UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TEMA

"SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO PARA EL MUNICIPIO DE MONTÚFAR."

APLICATIVO

"REDISEÑO E IMPLEMENTACIÓN DE LOS MÓDULOS DE FACTURACIÓN Y RECAUDACIÓN."

Autor: Edwin Ramiro Madruñero Padilla

Director: Ing. Mauricio Rea.

San Gabriel - Carchi – Ecuador

2010 – 2011

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE CONTACTO	
CEDULA DE IDENTIDAD	0401581731
APELLIDOS Y NOMBRES	MADRUÑERO PADILLA EDWIN RAMIRO
DIRECCIÓN	SAN GABRIEL - PANAMERICANA NORTE - SECTOR EL CAPULI
EMAIL	edwmadru@hotmail.com
TELÉFONO FIJO	062291323
TELÉFONO MOVIL	093924203

DATOS DE LA OBRA	
TITULO	"SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO PARA EL MUNICIPIO DE MONTÚFAR" con el aplicativo " REDISEÑO E IMPLEMENTACIÓN DE LOS MÓDULOS DE FACTURACIÓN Y RECAUDACIÓN."
AUTOR	EDWIN RAMIRO MADRUÑERO PADILLA
FECHA	09 DE ENERO DEL 2012
PROGRAMA	PREGRADO
TITULO POR EL QUE	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. MAURICIO REA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Edwin Ramiro Madruñero Padilla, con cedula de identidad Nro. 0401581731, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de educación Superior Articulo143.

……..

Firma

Nombre: Edwin Ramiro Madruñero Padilla

Cédula: 0401581731

Ibarra a los 9 días del mes de enero del 2012

UNIVERSIDAD TÉCNICA DEL NORTE

CESION DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Edwin Ramiro Madruñero Padilla, con cedula de identidad Nro. 0401581731, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, articulo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: "**SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO PARA EL MUNICIPIO DE MONTÚFAR"** con el aplicativo **" REDISEÑO E IMPLEMENTACIÓN DE LOS MÓDULOS DE FACTURACIÓN Y RECAUDACIÓN**.", que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte

……..

Firma

Nombre: Edwin Ramiro Madruñero Padilla

Cédula: 0401581731

Ibarra a los 9 días del mes de enero del 2012

CERTIFICACIÓN

Certifico que la Tesis "**SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO PARA EL MUNICIPIO DE MONTÚFAR"** con el aplicativo **" REDISEÑO E IMPLEMENTACIÓN DE LOS MÓDULOS DE FACTURACIÓN Y RECAUDACIÓN**." ha sido realizada en su totalidad por la señor:

Edwin Ramiro Madruñero Padilla

……

Ing. Mauricio Rea.

Director de la Tesis.

DECLARACIÓN

Yo, Edwin Ramiro Madruñero Padilla, con cédula de identidad Nro. 0401581731, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, y que este no ha sido previamente presentado para ningún grado o calificación profesional.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Universidad Técnica del Norte, según lo establecido por las leyes de Propiedad Intelectual y Normativa vigente de la Universidad Técnica del Norte.

……..……

Edwin Ramiro Madruñero Padilla

C.I. 0401581731

DEDICATORIA

A mi madre

Con mucho amor y
cariño

Le dedico todo mi
esfuerzo

Y trabajo puesto para

La realización de esta
tesis.

Edwin Ramiro Madruñero Padilla

DEDICATORIA

Dedico este proyecto de tesis a Dios y a mis padres. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi inteligencia y capacidad. Es por ello que soy lo que soy ahora.

Edwin Ramiro Madruñero Padilla

AGRADECIMIENTO

Los resultados de este proyecto, están dedicados a todas aquellas personas que, de alguna forma, son parte de su culminación.

Mis sinceros agradecimientos están dirigidos hacia Dr. Juan Acosta Alcalde del Gobierno Municipal de Montúfar, como también a sus respectivos concejales por haber brindado la oportunidad de implementar mis conocimientos a favor de la **SOCIEDAD MONTUFAREÑA**.

Al Analista Marco Pozo jefe del Departamento de Sistemas de la Municipalidad, al tecnólogo Juan Carlos Bedon técnico del mismo, Tecnólogo Darwin Martínez de igual menara Técnico del Departamento de Sistemas, por haber brindado su apoyo y de igual manera despejado varias inquietudes que surgieron en el transcurso del desarrollo del proyecto.

Así extender mis agradecimientos al Ing. Carlos Oña, Director del Departamento de Agua potable y Alcantarillado, así mismo a sus colaboradores en el mismo quienes con su ayuda desinteresada, me brindaron información relevante.

A mi familia por siempre brindarme su apoyo, tanto sentimental, como económico. Pero, principalmente mis agradecimientos están dirigidos hacia mi director de proyecto de tesis y amigo, Ing. Mauricio Rea, sin el cual no hubiésemos podido salir adelante.

Gracias Dios, gracias padres y hermanos, y en especial, gracias MI DIOS

Edwin Ramiro Madruñero Padilla

RESUMEN

Este proyecto de tesis consiste en el estudio, rediseño e implementación de dos módulos de un Sistema de Comercialización de Agua Potable y Alcantarillado (Facturación y Recaudación) para el Gobierno Municipal de Montúfar de la ciudad de San Gabriel, Provincia del Carchi.

Para la realización del sistema se realizó un previo estudio teórico sobre las tecnologías para el desarrollo de aplicaciones web para luego proceder al desarrollo del proyecto, utilizando la metodología de desarrollo RUP (Proceso Unificado de Rational).

Gracias a las aplicaciones web, la ciudanía del Cantón Montufar podrá conocer y utilizar los últimos avances tecnológicos y con ello la Municipalidad seguir innovando los procesos que se manejan de manera manual o por medio de sistemas rudimentarios que no brindan las facilidades al cliente ni los controles a los usuarios.

El sistema de Agua Potable cumple con todos los requerimientos funcionales (Ingresos, Cobros, Actualizaciones, etc.), que el departamento de Agua Potable de la Municipalidad

Una vez concluido este proceso, el sistema se implementará en la dependencia del Municipio y de esta manera obtener beneficios para los abonados y proveer la eficiencia del sistema de información a la comunidad.

SUMMARY

This thesis project is the study, redesign and implementation of two modules of marketing drinking water and sewer system (billing and collection) to Municipal Government of Montúfar in San Gabriel, Carchi Province.

For the realization of the system was performed prior theoretical study on technologies for web application development and then proceed to project development, using development methodology RUP (Rational Unified Process).

With web applications, the citizen of Montufar may know and use the latest technology and with it the city are innovating the processes that are handled manually or by elementary systems that do not provide the facilities or controls to the customer users.

The Drinking Water System meets all functional requirements (Revenue, Invoicing, Updates, etc.), that the Drinking Water Department of the Municipality.

Once this process has concluded, the system will be implemented in the dependence of the Municipality and thus obtain benefits for the subscribers and provide efficient information system to the community.

CONTENIDO

FIGURAS

TABLAS

1. TEMA

“SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO PARA EL MUNICIPIO DE MONTÚFAR” **con el aplicativo** " REDISEÑO E IMPLEMENTACIÓN DE LOS MÓDULOS DE FACTURACIÓN Y RECAUDACIÓN**.”**

2. OBJETIVOS

2.1. OBJETIVO GENERAL

- Rediseño e Implementación de los módulos de Facturación y Recaudación del Sistema para la Comercialización del Servicio de Agua Potable y Alcantarillado del Municipio de Montúfar con el fin de aumentar la productividad y brindar un servicio de calidad a la ciudadanía.

2.2. OBJETIVOS ESPECÍFICOS

- Analizar la documentación existe en la Municipalidad con el fin de conocer el manejo y funcionamiento de los procesos de Administración de Agua Potable y Alcantarillado.

- Recomendar la plataforma tecnológica a utilizar para la restructuración del Sistema de Administración del Servicio de Agua Potable e implementación de los nuevos módulos (Recaudaciones y Facturación).

- Diseñar e Implementar la Base de Datos de los nuevos módulos (Recaudaciones y Facturación) del sistema de Comercialización de Agua Potable y Alcantarillado.

- Disponer de información consolidada e integrada a través de los reportes que ayudarán a la toma de decisiones

3. INTRODUCCIÓN

3.1. EL MUNICIPIO DE MONTÚFAR, POBLACIÓN, UBICACIÓN, HISTORIA.

La Municipalidad de Montúfar es una institución pública dedicada a trabajar en beneficio de la comunidad.

La innovación tecnológica ha sido una de los ejes principales de su actual administración, para ofrecer una mejor atención a la ciudadanía en sus trámites y transacciones.

El Cantón Montúfar, es el segundo Cantón de la Provincia. Se encuentra ubicado en la región Centro Norte de la Provincia del Carchi. Limita por el Norte con los cantones de: Tulcán y Huaca; Por el Este, con la Provincia de Sucumbíos, cuya barrera natural lo constituye la cordillera central; y por el Sur y Oeste, con los cantones de Bolívar y Espejo. El Cantón tiene 400,4 Km2, y la ciudad de San Gabriel, su cabecera, se asienta en una extensión de 4,4 Km2. En el cantón Montufar las alturas van de los 2200msnm a los 3800 msnm. El clima se caracteriza por ser templado-frio con una temperatura media multianual de 12.5ºC. Las precipitaciones mínimas se ubican en agosto y enero con 38mm y 72mm respectivamente; y las máximas en abril y noviembre con 113mm y 109mm respectivamente.

La ciudad de San Gabriel, cuenta con 6 parroquias rurales y 2 parroquias urbanas. Distribuidas de la siguiente manera:

DISTRIBUCIÓN PARROQUIAL DE LA POBLACIÓN DEL CANTÓN MONTÚFAR			
Parroquias	**Total**	**Hombres**	**Mujeres**
San Gabriel (Área Urbana)	12.575	5.965	6.610
San Gabriel (Periferia)	6.655	3.288	3.367
Cristóbal Colón	2.932	1.447	1.485
Chitán de Navarrete	672	330	342
Fernández Salvador	1.393	727	666
La Paz	3.201	1.520	1.681
Piartal	1.148	612	536
Total	**28.576**	**13.889**	**14.687**

Tabla 1 Distribución Parroquial de la población

Fuente: INEC 2001

El cantón Montufar se encuentra localizado al norte del Ecuador y forma parte de la provincia del Carchi, su barrera natural lo constituye la cordillera central, se encuentra a una altitud entre los 2200- 3800 msnm y posee una extensión de 390 km^2, su temperatura promedio oscila entre los 12,55º C.

Figura 1 Mapa geográfico de Ecuador - Carchi

San Gabriel, cabecera del cantón de Montúfar, está localizada en el 000º36' latitud Norte, y 77º50' longitud Oeste. Tiene 362 km2 y está emplazada a una altitud de 2870 msnm. Los ramales de las cordilleras de Los Altos de Boliche y Azufral cierran a San Gabriel por las orientaciones Norte-Este y Sur. Tiene 275 km de carreteras y caminos vecinales y la densidad demográfica es de 56 habitantes/km2. La pluviosidad anual es de 1000 mm/año y la temperatura media de 9ºC. La ciudad está conformada por dos parroquias urbanas: González Suárez y San José, integradas por 13 barrios distribuidos de la siguiente manera:

3.1.1. BARRIOS BENEFICIARIOS DE LA CIUDAD DE SAN GABRIEL

PARROQUIAS	BARRIOS	TIPO	Nº HABITANTES
González Suárez	27 de Septiembre	Urbano	1.424
González Suárez	San Pedro	Urbano	458
González Suárez	San Vicente	Urbano	554
González Suárez	Santa Clara	Urbano	2977
San José	Centenario	Urbano	2.334
San José	San Antonio	Urbano	1.274
San José	San José	Urbano	4.260
San José	San Pedro de Los Cipreses	Urbano	210

San José	Santa Martha de Indújel	Urbano	435
San José	Santa Rosa	Urbano	1.171
San José	San Andrés	Urbano	245
San José	Unión y Progreso	Urbano	240
TOTAL			**16.136**

Tabla 2 Barrios de la Ciudad de San Gabriel

Fuente: Censo del Gas, 2010; Departamento de Patrimonio.

3.1.2. RESEÑA HISTÓRICA.

Fuente: Por Zenón Ponce Ch, Monografía del Cantón Montúfar, edición 1955-2010, Ilustre Municipalidad de Montúfar.

3.1.2.1. FUNDACIÓN ESPAÑOLA DE LA POBLACIÓN TUSA

EL 05 agosto de 1535, el Capitán Tapia, enviado por Sebastián de Benalcazar realizó la fundación española de la población de Tusa, actualmente San Gabriel.

3.1.2.2. PRIMERA IGLESIA PARROQUIAL DE TUSA

En 1568 se techó la primera iglesia parroquial de Tusa, así lo atestigua una viga (fechada) encontrada entre los escombros de la iglesia en el terremoto de 1868.

Para 1647 el pueblo de Tusa, estaba conformada por ciento noventa y cuatro indios (sin tomar en cuenta los enviados a las mitas), agrupados en dieciséis ayllus.

Hacia 1750 Pedro Vicente Maldonado, realizó el levantamiento de la "Carta de la Provincia de Quito y sus adyacentes" en la que consta el pueblo de Tusa.

3.1.2.3. PRIMERAS AUTORIDADES CIVILES Y ECLESIÁSTICAS DEL PUEBLO TUSA

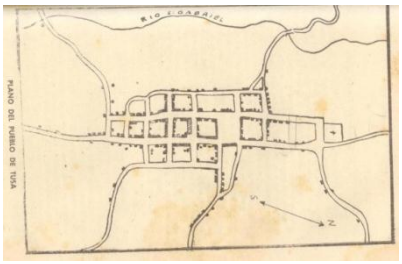
En 1831 el pueblo de Tusa contaba con un juez civil que ejercía las funciones de Teniente Político en la Parroquia Civil de Tusa en 1848 como autoridad eclesiástica estaba el Fray Mercedario Agustín Valdospinos, mientras que el cargo de Teniente Político se lo

desempeñaba con las funciones también de síndico de la Cofradía de la iglesia.

3.1.2.4. FUNDACIÓN DE LA PRIMERA ESCUELA FISCAL DE "PRIMERAS LETRAS"

Zenón Ponce sostiene que en el período republicano, particularmente entre 1835 y 1839, hubo una especial preocupación por el mejoramiento de la educación, se crearon escuelas en todos los lugares de la República y en Tusa se fundó una escuela fiscal de "Primeras Letras", seguramente mixta. En 1853, en la presidencia del general José María Urbina, ya se habla del funcionamiento de una escuela mixta en Tusa, pagada por el Gobierno Central, dependiente hasta entonces del Colegio de San Diego de Ibarra que de sus fondos atendía a las escuelas que de él dependían, o sea a todas las de la provincia de Imbabura que se extendía en aquel tiempo hasta el territorio de la actual provincia del Carchi (Ponce, 1955, 216- 218).

3.1.2.5. RECONSTRUCCIÓN DE LA CIUDAD Y EL PRIMER PLANO EN DAMERO



El terremoto de Ibarra en 1868 tuvo repercusión en una amplia zona del norte del país, en San Gabriel destruye una gran parte de la ciudad, como la Iglesia Matriz, y es el entonces Jefe Civil y Militar de Imbabura Dr. García Moreno quien ordena la reconstrucción de la ciudad, realizándose para el efecto el primer plano de San Gabriel (en ese momento ciudad de Tusa), basado en el diseño utilizado por los españoles, es decir, el trazado en cuadrícula o damero.

3.1.2.6. POR MANDATO LEGISLATIVO TUSA CAMBIA DE NOMBRE POR EL DE SAN GABRIEL

El 23 de abril de 1884, por mandato del Poder Legislativo, se cambia el nombre de Tusa por el de la ciudad de San Gabriel, en

homenaje al Dr. Gabriel García Moreno.

3.1.2.7. CANTONIZACIÓN DE MONTÚFAR

El 27 de Septiembre 1905 el Congreso de la República del Ecuador Vista la solicitud de los vecinos de San Gabriel:
Art. 1 Elévese a la categoría de Cantón las Parroquias de San Gabriel, La Paz, San Isidro, Mira y San Pedro de Piquer.
Art.2 El nuevo cantón se llamará Montúfar y su capital será San Gabriel.
Art. 3 La elección de Concejales se hará en tiempo y forma que determina las Leyes vigentes, correspondiendo el primer escrutinio a la Municipalidad de Tulcán.

3.1.2.8. PRIMER CONCEJO MUNICIPAL DE MONTÚFAR

En la ciudad de San Gabriel, capital del Cantón Montúfar, a las 12:30 del 21 de diciembre de 1905, se instaló en el salón de sesiones, el Primer Concejo Municipal de Montúfar, presidido por el Sr. Félix Oña, Jefe Político.

De acuerdo con la Ley, la elección de concejales se realizó en el mes de noviembre de 1905. Verificados los escrutinios en la ciudad de Tulcán según lo dispuesto en el decreto, los resultados favorecieron a los señores: Carlos Oña Benavides, José María Navarrete B., Isaac Acosta, José María Oña y Carlos Vinuesa, designados por voluntad de los Sangabrieleños, como miembros del Primer Concejo Municipal de Montúfar.

3.1.2.9. PRIMEROS BARRIOS Y MANZANAS DE SAN GABRIEL

La formación de los barrios de San Gabriel es muy antigua, los mismos que se ponen de manifiesto en las mingas para contribuir en las obras de beneficio para todo el cantón .En 1915, cuando se trasladaba la madera hasta el lugar donde funciona la escuela Calderón, y por evitar problemas de tránsito en la ciudad, el Vicario Ángel Guzmán, con los señores Gregorio Chamorro, Manuel Carrera y Secundino Chamorro organizaron el traslado del material por sectores, de manera que quedaron a su vez fijadas las manzanas que deberían trabajar en las mingas; este es también el origen de los barrios de San Gabriel, y que de sur a norte son: San José, San Antonio, San Pedro, San Vicente y Santa Clara

Para la década de los cincuenta ya existían varios edificios como la Casa Municipal en la Plaza González Suárez; el Salón de la Ciudad y Teatro, la Iglesia, que fue reconstrucción el siglo XIX; el colegio José Julián Andrade, entre otros.

3.1.2.10. PRIMERAS PLANTAS HIDROELÉCTRICAS DEL CANTÓN MONTÚFAR

La Luz Eléctrica**. "**Antes de 1954, el alumbrado público se hacía con faroles. En 1919 se gestionó ante el gobierno de Vaquerizo Moreno, la instalación de una planta hidroeléctrica; pedido y gestión que fue realizado por el Concejo Municipal y la Sociedad Obrera del Cantón. Su instalación se la hizo a través de mingas. El mes de mayo de 1923 el Concejo Municipal, presidido por Juan Agreda, instaló la primera planta de luz, marca Pelton de 27 Kw de capacidad, movida por las aguas del río Paluz.

En noviembre de 1945 se instala una nueva planta de luz de una capacidad de 150 Kw, debido a las nuevas necesidades de abastecimiento, con lo cual se dio servicio a Cristóbal Colón. En octubre del mismo año se celebró un contrato con la empresa Electro

Ecuatoriana para la adquisición de una nueva planta con una capacidad de 375 Kw, que funcionó desde septiembre de 1955, año del cincuentenario de la cantonización de San Gabriel. En esa misma época ya había el proyecto de electrificación para todo el Cantón, con una planta en la Gruta de Rumichaca".

3.1.2.11. MINGAS PARA LA CONSTRUCCIÓN DE LA CARRETERA ORIENTAL

Agosto de 1925, La Junta de Gobierno Provisional expide el decreto ordenando la construcción de una vía carrozable a la frontera norte. Ésta se llamaría "Carretera 9 de Julio". Para el efecto se realizaría el estudio de reconocimiento y factibilidad, considerando tres vías posibles:

- Chota-San Gabriel-Las Juntas, vía oriental
- Chota-El Ángel-Las Juntas, vía occidental
- Chota-Pucará-Las Juntas, vía central

Abril de 1926, con una gran concurrencia, conformada por las delegaciones de las parroquias del cantón mediante votación, se elige a los señores: Manuel Bastidas, como presidente; Sacerdote Nicanor Gavilanes, como vicepresidente; Gabriel Landázuri, como tesorero y el Prof. Miguel Aristizábal, como secretario. El nombre del comité, elegido en Asamblea, fue Comité Pro carretera Oriental del Carchi

En enero de 1927, se realiza una asamblea general en San Gabriel en la que se insta a los ciudadanos a construir la carretera por cuenta propia; ésta arrancaría desde la hacienda San Alfonso, en la cuenca del Chota, pasaría por el río en el punto El Juncal, atravesaría Íntegramente la banda oriental de la provincia y desembocaría en el punto Las Juntas cerca de Tulcán. El señor Presidente del Comité Pro carretera, don Manuel J. Bastidas, con su gran talento llamó a la conciencia de los concurrentes y arrancó un solemne juramento de construir la carretera con 'nuestros propios esfuerzos intelectuales, físicos y económicos. Fue continuamente interrumpido en su intervención por nutridos aplausos y vivas que llenos de emoción le brindaban los concurrentes a la Asamblea".

Junio de 1927, se realiza una sesión ampliada con la asistencia del Concejo Provincial, comité y delegados de los caseríos y barrios de la ciudad. Se resuelve a empezar la construcción de la carretera oriental el 27 de junio.

Junio 27 de 1927 el día empieza con el programa de inauguración:

Entonando el himno nacional todos los concurrentes, el señor Manuel J. Bastidas, Presidente del Comité Pro carretera Oriental del Carchi, en una entusiasta y patriótica proclama, declara inaugurados los trabajos de construcción de la vía; como se dijo, asisten a este acto muchas delegaciones; se reparten tarjetas conmemorativas. La presencia de la mujer Sangabrieleña es invalorable, han confeccionado ramilletes de flores naturales anudados con cintas de seda que tenían leyendas referentes al acto. Éstas eran prendidas en el pecho de los ciudadanos concurrentes. El acto fue solemne: todos los miembros del Comité toman sus herramientas de trabajo y a una señal previamente convenida dan la primera palada, que retumba en la tierra y su sonido se expande por los aires como pregón de fiesta; estas primeras paladas fueron en el barrio "El Hueco", frente a la casa del señor Secundino Chamorro"

Agosto 31 de 1927, la construcción de la vía llega a la plaza del caserío Caico, hoy Cristóbal Colón. Se decide realizar una gran minga que cubra desde la plaza de Caico hasta Paja Blanca en el sector de Huaca. Ésta se realizaría desde San Luis, los días 25, 26 y 27 de septiembre en homenaje al aniversario de la erección del cantón.

Septiembre 27 de 1927, el primer automóvil entra a San Gabriel por la carretera.
Febrero de 1928, se termina y aprueba el trazado hasta la parroquia La Paz. Se hace entonces un nuevo llamado a la ciudadanía para empezar los trabajos de construcción de la vía que una a San Gabriel con La Paz.

Abril de 1928, se inicia los trabajos hacia Bolívar; se trabaja los días viernes y sábado de cada semana.

Octubre de 1928, los trabajos se encuentran muy cerca de la parroquia Bolívar y se efectúa el trazado hasta Duendes.

Octubre 25 de 1928, se inicia la minga en Duendes con la asistencia de 2000 Sangabrieleños. Se trabajan 1500 metros de carretera en un lugar que se creía imposible de modificar por cuanto el terreno era rocoso y de barrancos profundos.

Enero de 1929, los trabajos de construcción habían pasado la población de Bolívar; se colocaban puentes en las quebradas para dar paso a los vehículos motorizados.
Fines de 1929, la carretera estaba totalmente concluida y en servicio hasta la quebrada de Duendes, al lado de Bolívar.

Inicios de1930, se restablecen las mingas diarias con la concurrencia de barrios y lomas de San Gabriel y de los demás pueblos del Cantón.

Julio de 1930, la vía estaba concluida hasta la quebrada que dividía la hacienda Chulunguza de Cunquer.

El hecho sobresaliente en los días 25, 26 y 27 de septiembre de 1930 cuando se congregaron 15.000 hombres con sus herramientas bajo el brazo, rompieron la tierra e hicieron aparecer como un milagro la cinta curvada, en los barrancos profundos de la hacienda Cunquer, que es la carretera oriental del Carchi

En la gran minga estaban presentes los siguientes pueblos: Julio Andrade, Huaca, Cristóbal Colón, San Gabriel, La Paz, Bolívar, Los Andes, García Moreno, Mira, San Isidro, San Vicente de Pusir, Pimampiro, Ambuquí y la peonada de la hacienda Pimán; la presidencia recibió los partes enviados con los comisionados indicando el total de trabajadores que alcanzó a 13.580 ciudadanos que cubrían una longitud de cerca de 30 Km.

Septiembre de 1930el aniversario de la creación del cantón Montúfar, sus hijos desean rendirle un sublime homenaje; por eso redoblan sus esfuerzos, ya están a la vista grandes trechos de la vía, terminados; no descansan porque no pueden desperdiciar el tiempo ya que la carretera debe quedar terminada ese día, es el GRAN HOMENAJE A LA PATRIA CHICA.

Los trabajos realizados afanosamente se iniciaron en junio de 1927 y se termina en septiembre de 1930, lo que ocurrió a las cinco de la tarde del 27 de septiembre de dicho año".

Esa memorable fecha entregaba el cantón Montúfar al Carchi, a la Nación y al mundo más de 120 Km de carretera trabajados en unidad de acto, cubriendo los gastos necesarios con las donaciones de cada ciudadano sin que el Estado los sufragara.

Octubre 26,27, y 28 de 1936, se realiza la última gran minga, entregándose el último día el trecho terminado. Dando fe de lo expuesto y a pesar de haber sido rectificada la carretera, aún quedan trechos largos en servicio y, sobre todo, está la Quebrada de Duendes como un hito eterno, testigo mudo del esfuerzo físico, moral y económico de un pueblo que escribió en las rocas eternas donde quedará petrificada para siempre esa hermosa palabra llamada patriotismo.

3.1.2.12. CONGRESO NACIONAL OTORGA EL GALARDÓN DE PROCERATO DEL TRABAJO A SAN GABRIEL.

El pueblo de San Gabriel se destaca por su espíritu minguero, donde se pone de manifiesto la unión y solidaridad entre los habitantes; muchas obras se han ejecutado en el cantón a través de este sistema, lo que ha contribuido a su desarrollo; este hecho hizo que en 1934 el Honorable Congreso Nacional, otorgue a San Gabriel el galardón de "El Procerato de Trabajo", distinción que hasta el momento no ha sido conseguida por ninguna otra población.

3.1.2.13. EL TRANSPORTE EN SUS INICIOS.

Después de terminada la vía Oriental, en la década de los treinta, se inició el desarrollo de los sistemas de trasporte terrestre con la creación de cooperativas.

3.1.2.14. OBRAS DE CANALIZACIÓN EN LA CIUDAD DE SAN GABRIEL

En cuanto al problema de la canalización iniciaron las gestiones para solucionarlo en 1940 dotando de este servicio a San Gabriel. En 1940 se inauguró la piscina municipal, las lavanderías públicas y baños de duchas en la Plaza Amazonas.

El Agua Potable y Canalización ante el aumento de las necesidades de la población, en 1951, el Concejo Municipal firmó un contrato con el Servicio Cooperativo Interamericano de Salud Pública para la adquisición de un nuevo sistema de agua potable que fue inaugurado en junio de 1954. Hasta esta fecha el servicio de agua potable se hacía a través de las instalaciones que había hecho el Concejo Municipal.

3.1.2.15. PRIMERAS CALLES Y PLAZAS IMPORTANTES DE SAN GABRIEL

En los años cincuenta la ciudad se componía de cinco calles tradicionales longitudinales y diecisiete transversales. Rocafuerte, Montúfar, Bolívar, Los Andes y 27 de Septiembre eran las longitudinales; Carrera Maldonado, Pichincha, Calderón, Sucre, Colón, García Moreno, Olmedo, Montalvo, Mejía, Salinas, Ricaurte, Ángel P. Chávez e Ibarra, transversales.

Las plazas eran cuatro: González Suárez (principal). Para esta plaza Luis Mideros, en el año de 1955, diseñó el monumento al Procerato del Trabajo, que demoró muchos años en construirse. Hacia el norte, la Plaza Amazonas, destinada al mercado de animales y víveres de los días sábados. Al sur, frente al Hospital Civil y la Escuela Pío XII, está la Plaza José Peralta. En la Carrera Los Andes, se consiguió una extensión de terreno para el Estadio Municipal, en donde se desarrollaban juegos de Pelota Nacional, fútbol, etc.

3.1.2.16. URBANISMO Y PATRIMONIO NACIONAL.

El 11 de noviembre de 1992 se expidió la declaratoria del centro histórico de San Gabriel como Patrimonio Cultural de la Nación lo que supuso a continuación el levantamiento de un inventario arquitectónico urbano de la ciudad en el que se describen las casas del centro y de otras. La mayor parte de casas pertenecen a inicios del siglo XX aunque también hay construcciones de finales del siglo XIX.

3.2. ORGANIGRAMA FUNCIONAL DE LA MUNICIPALIDAD.

Fuente: Manual de Funciones del Gobierno Municipal de Montúfar; Departamento de Talento Humano.

REGLAMENTO ORGANICO FUNCIONAL DE LA I. MUNICIPALIDAD DEL CANTÓN MONTÚFAR

EL I. CONCEJO MUNICIPAL DEL CANTON MONTUFAR

C O N S I D E R A N D O

QUE, El Reglamento Orgánico Funcional Vigente no está de acuerdo a las disposiciones de la Ley de Régimen Municipal.

QUE, No responde a las necesidades que el Municipio de Montúfar requiere para el cumplimiento eficiente de sus funciones;

QUE, Los retos que se ha planteado la Municipalidad requieren definir y delimitar, así como coordinar las actividades y funciones del Municipio, y por consiguiente de cada una de sus áreas;

QUE, Es necesario que sea el Municipio quien atienda y resuelva problemas inminentes como son medio ambiente y desarrollo sustentable, educación y cultura;

En ejercicio de las atribuciones de la Ley de Régimen Municipal en los artículos 64 numeral 49; y 72 numeral 27.

RESUELVE:

Emitir la siguiente Ordenanza que contiene el Reglamento Orgánico Funcional para la I. Municipalidad de Montúfar.

TITULO I

DEL GOBIERNO MUNICIPAL

Art. 1. El I. Municipio de Montúfar, constituye una persona jurídica de derecho público, de conformidad con lo dispuesto en la Ley de Régimen Municipal, es una Entidad Autónoma, subordina al orden jurídico constitucional del Estado, cuya finalidad es el bien común local, y dentro de éste en forma primordial la atención de las necesidades de su ciudad, sus parroquias urbanas y rurales.

Art. 2. El gobierno del Municipio, de acuerdo a lo establecido en la Ley de Régimen Municipal, lo conforma el Concejo y el Alcalde.

Art. 3. El Concejo está integrado por 7 ediles, designados por votación popular, siendo sus funciones las previstas en la ley de Régimen Municipal.

Art. 4. El Alcalde, de conformidad con lo dispuesto en la Constitución Política de la República es el órgano ejecutivo de la municipalidad, además de ser el Superior Jerárquico de la Administración Municipal, según dispone el Art. 26 de la Ley de Régimen Municipal; sus funciones se encuentran ampliamente establecidas en dicho cuerpo legal.

Art. 5. Los niveles de actividades que contemplará la estructura orgánica funcional de la Administración del Municipio de Montúfar, son:

- Directivo
- Asesor y,
- Operativo

Previstos en la Ley de Régimen Municipal, éstos se establecen de conformidad con lo determinado en la misma Ley para la estructura

administrativa y, en las funciones que se especifican en la presente Ordenanza.

FUNCIONES DEL CONCEJO
ATRIBUCIONES Y DEBERES

Art. 6. La acción del Concejo está dirigida al cumplimiento de los fines del Municipio, para lo cual tiene los siguientes deberes y atribuciones generales:

1. Normar a través de ordenanzas, reglamentos, acuerdos o resoluciones, la política a seguirse y fijar las metas en cada uno de los ramos propios de la administración municipal.

2. Conocer y aprobar la programación técnica de corto y largo plazo elaborada por los respectivos departamentos y aprobada por las comisiones pertinentes.

3. Dirigir el desarrollo físico del cantón y el ordenamiento urbanístico, de acuerdo con las previsiones especiales de la Ley de Régimen Municipal y las generales sobre la materia.

4. Aprobar los planes reguladores del desarrollo físico cantonal y los planes reguladores de desarrollo urbano, formulados de conformidad con las normas de esta Ley.

5. Controlar el uso del suelo en el territorio del Cantón, de conformidad con las leyes sobre la materia, y establecer el régimen urbanístico de la tierra.

6. Aprobar o rechazar los proyectos de parcelaciones o de reestructuraciones parcelarias formulados dentro de un plan regulador de desarrollo urbano.

7. Autorizar la suspensión hasta por un año del otorgamiento de licencias de parcelación de terrenos y de edificaciones en sectores comprendidos en un perímetro determinado, con el fin de estudiar el plan regulador de desarrollo urbano o sus reformas.

8. Aprobar el plan de obras locales contenidas en los planes reguladores de desarrollo urbano, todas las demás obras que interesen al vecindario y las necesarias para el Gobierno y administración municipal.
9. Decidir cuáles de las obras públicas locales deben realizarse por gestión municipal, bien sea directamente o por contrato o concesión, y cuáles por gestión privada y, si es el caso, autorizar la participación de la Municipalidad en sociedades de economía mixta.

10. Decidir el sistema mediante el cual deben ejecutarse los planes de urbanismo y las obras públicas.

Y las demás contempladas en la Constitución Política del Estado, Ley de Régimen Municipal, y demás Leyes Conexas.

FUNCIONES DEL ALCALDE
DE LOS DERECHOS Y ATRIBUCIONES

Art. 7. Son deberes y atribuciones del Alcalde:

1. Cumplir y hacer cumplir la Constitución y leyes de la República y las ordenanzas, reglamento, acuerdos y resoluciones del Concejo.

2. Representar, junto con el Procurador Síndico Municipal, judicial o extrajudicial a la Municipalidad.
3. Convocar al Concejo a sesiones ordinarias y extraordinarias, de conformidad con lo que sobre la materia dispone la Ley de Régimen Municipal.

4. Presidir las sesiones del Concejo, dar cuenta a éste de cuanto le corresponda resolver, y orientar sus discusiones.

5. Integrar y presidir la Comisión de Mesa.

6. Nombrar las comisiones permanentes que no hubiese integrado el Concejo o la Comisión de Mesa, y las especiales que estime convenientes.

7. Aprobar, con la Comisión de Mesa, las actas de las sesiones del Concejo cuando éste no lo hubiera hecho.

8. Intervenir en el trámite de los actos municipales cuya resolución corresponda al Concejo.

9. Suscribir las actas de la sesión del Concejo y de la Comisión de Mesa.

10. Conceder licencia a los concejales para que no actúen en una comisión, de acuerdo con lo que dispone la Ley de Régimen Municipal.

Y las demás contempladas en la Constitución Política del Estado, Ley de Régimen Municipal, y demás Leyes Conexas.

FUNCIONES DE SECRETARÍA GENERAL

Art. 8. Son funciones del Secretario(a) las siguientes:

1. Dar fe de los actos del Concejo, de la Comisión de Mesa y de la Alcaldía.

2. Redactar y suscribir las actas de las sesiones del Concejo y de la Alcaldía.

3. Cuidar del oportuno trámite de los asuntos que deba conocer la Corporación en Pleno, y atender el despacho diario de los asuntos resueltos por el Concejo.
4. Formar un protocolo encuadernado y sellado, con su respectivo índice numérico de los actos decisorios del Concejo, de cada año y conferir copia de esos documentos conforme a la Ley.

5. Llevar y mantener al día el archivo de documentos del Concejo y atender el trámite de la correspondencia.

6. Coordinación con todas las secretarías departamentales.

7. Las demás que le señale la Ley y los Reglamentos, y sean confiadas por el Alcalde

Art. 9. **DE LA PROSECRETARÍA**

Son sus funciones

1. Clasificar, elaborar y distribuir toda la correspondencia que ingresa al Concejo.

2. Mantener actualizados los archivos de documentos para suministrar información.

3. Revisar y proponer reformas al sistema de documentación y archivo del Concejo.

4. Canalizar previa autorización del Alcalde las órdenes de pago correspondientes.

5. Subrogar al Secretario a) General en la ausencia de este.

6. Y las demás que fueren confiadas por el Alcalde o el Secretario General del Concejo.

DE LA ESTRUCTURA ADMINISTRATIVA

Se establece la siguiente estructura administrativa del municipio, de conformidad a lo establecido en la Ley de Régimen Municipal como la estructura mínima que deben tener las municipalidades, la cual en el Art. 173 prevé la facultad de adaptarla considerando las características propias de cada municipio.

Art. 10. El Municipio de Montúfar, estará integrado por las siguientes Direcciones.

- Dirección Administrativa - Financiera
- Dirección de Obras Públicos
- Dirección de Agua Potable y Alcantarillado.
- Dirección de Planificación.
- Dirección de Asistencia Social

Art. 11. El nivel asesor en la municipalidad de Montúfar estará conformado por:

- Planificación Urbana;

- Programación Presupuestaria y Estadística;
- Asesoría Jurídica;
- Administración de Personal; y,
- Organización y Métodos.

A este nivel le corresponde prestar asistencia técnica a los niveles directivo y operativo en cuestiones de planeación, programación, presupuesto, administración de recursos humanos y proyección de las actividades municipales en materias legales y en asuntos de organización administrativa.

Los planes y programa que presenten deberán integrar un plan de desarrollo municipal que aceptado por el alcalde, deberá ser aprobado por el Concejo.

Art. 12. El nivel Operativo, estará conformado por los distintos departamentos que integran las diferentes direcciones, debiendo encargarse de las funciones que en cada uno de los ramos de la actividad municipal les corresponda, de conformidad al presente reglamento.

DIRECCIÓN ADMINISTRATIVA - FINANCIERA

Integrada por los siguientes Departamentos:

1. Tesorería
2. Presupuesto
3. Contabilidad
4. Comprobación y Rentas
5. Proveeduría y Bodega
6. Recursos Humanos
7. Centro de Cómputo

DIRECCIÓN DE OBRAS PÚBLICAS

Integrada por los siguientes Departamentos:

1. Fiscalización
2. Transporte

DIRECCIÓN AGUA POTABLE Y ALCANTARILLADO

Siendo los Departamentos que la conforman los siguientes:

1. Agua Potable y Alcantarillado

DIRECCIÓN DE PLANIFICACIÓN

Siendo los Departamentos que la conforman los siguientes:

1. Medio Ambiente
2. Avalúos y Catastros
3. Comisaría

DIRECCIÓN DE ASISTENCIA SOCIAL

Siendo los departamentos que la conforman los siguientes:

1. Cultura, Educación y Deportes
2. Salud Pública

ESTRUCTURA FUNCIONAL

Párrafo 1°

Art. 13. **DEL PROCURADOR SINDICO**

Son funciones del Procurador Síndico:

1. Representar, conjuntamente con el Alcalde, judicial y extrajudicialmente a la Municipalidad.

2. Dar asesoría al Acalde y a los Concejales; así como a los demás órganos administrativos de la municipalidad.

3. Asistir a las sesiones del Concejo obligatoriamente, como voz asesora.

4. Absolver las consultas que se presenten con relación a la aplicación de la ley; sean estas planteadas por los funcionarios del municipio, o por sus usuarios.

5. Estudio de los problemas legales relacionados con la Municipalidad.

6. Vigilar que los reclamos de los contribuyentes sean atendidos en la forma prevista en las leyes, dependiendo de la materia, y elaborar los informes jurídicos, tanto para los reclamos administrativos tributarios, como para cualquiera que se presente al municipio.

7. Emitir dictámenes legales sobre asuntos que debe conocer la Administración Municipal.

8. Formar parte del comité de contratación, estando obligado a asistir a toda reunión de éste, y velar por el cumplimiento de las disposiciones legales.

9. Presentar al Alcalde los informes jurídicos sobre la aplicabilidad de los proyectos de ordenanzas o reformas de ordenanzas.

10. En coordinación con el Tesorero y el Director Financiero tramitar los juicios de coactivas.

11. Revisar todos los contratos que deba suscribir conjuntamente con el Alcalde, y mantener el Archivo de éstos.

12. Las demás que le asigne el Concejo y el Alcalde.

Párrafo 2°
DE LAS FUNCIONES DE LA DIRECCIÓN ADMINISTRATIVA-FINANCIERA Y SUS DEPARTAMENTOS

Art. 14. **DE LA DIRECCIÓN ADMINISTRATIVA – FINANCIERA**

1. Planificar, organizar, ejecutar y controlar las actividades administrativas y financieras de la Municipalidad.

2. Asesorar a los diferentes niveles directivos de la Institución en materia administrativa y financiera.

3. Administrar los recursos financieros en forma eficiente, efectiva y económica.

4. Realizar estudios sobre el contenido de las ordenanzas, mediante las cuales se regula la recaudación de los diversos ingresos y proponer reformas que tiendan a mejorar los ingresos respectivos.

5. Elaborar y mantener estadísticas económicas de la entidad.

6. Dirigir y ejecutar la administración tributaria municipal, de conformidad con la Ley de Régimen Municipal, las leyes tributarias específicas, las ordenanzas y demás normas y procedimientos legales y técnicos vigentes sobre la materia.

7. Proporcionar información financiera a los niveles internos, cuando lo soliciten las Autoridades y a las diversas unidades administrativas.

8. Presentar con oportunidad los correspondientes estados y anexos a los organismos públicos que por Ley Corresponda.

9. A través del Departamento de Presupuesto, coordinar, ejecutar y controlar las actividades relacionadas con la realización del Presupuesto Municipal de acuerdo al Plan programático anual de la Municipalidad.

10. Organizar y ejecutar las actividades contables y verificar la documentación previa a la realización de pagos, a través del Departamento de Contabilidad.

11. Organizar, ejecutar, controlar y verificar la emisión de Títulos de Crédito, y realizar los correspondientes reportes, a través del Departamento de Comprobación y Rentas.

12. Planificar, organizar y ejecutar las actividades de custodia, pagos, recaudación y recuperación de valores municipales, así como también la emisión de Títulos de Crédito, especies valoradas, de impuestos, tasas y demás de recaudación a través del Departamento de Tesorería.

13. Coordinar las acciones de custodia de los bienes muebles e inmuebles de propiedad municipal, proveer en stock los requerimientos de consumo interno y coordinar las adquisiciones a través del Departamento de Bodega y Proveeduría.

14. Coordinar y controlar las actividades de los empleados y trabajadores y verificar el cumplimiento de sus funciones a través del Departamento de Recursos Humanos.

15. Planificar, organizar y ejecutar el funcionamiento del sistema informático, de radio y telecomunicaciones de la municipalidad y su correcta utilización a través del Centro de Cómputo.

16. Participar como asesores en las sesiones que mantienen los miembros del Concejo Municipal para tratar asuntos administrativos, económicos y financieros de la entidad.

Art. 15. **DEL DEPARTAMENTO DE PRESUPUESTO**

Son funciones del Departamento de Presupuesto las siguientes:

1. Planificar, organizar, coordinar, ejecutar y controlar las actividades relacionadas con el Presupuesto Municipal, según los programas y proyectos previstos en el Plan Operativo Anual de la Entidad.

2. Estructurar el anteproyecto de Ordenanza al Presupuesto del Ilustre Municipio de Montúfar y someterlo a su aprobación.

3. Analizar y proponer mejoras en los sistemas y procedimientos del control presupuestario.

4. Analizar, estudiar e informar periódicamente a la Dirección Administrativa - Financiera sobre los requisitos de la distribución, incremento, reducción y traspasos presupuestarios.

5. Elaborar informes mensuales de la ejecución presupuestaria y formular informes para uso de las unidades correspondientes.

6. Elaborar y tramitar las solicitudes realizadas por las Autoridades de cupos de transferencias, traspasos de crédito, modificaciones presupuestarias, incremento o reducción al presupuesto y otras acciones de carácter económico que se requiere y someterlas a su aprobación.

7. Efectuar los registros y controles presupuestarios relacionados con el compromiso, obligación y pagos asignados en la acción administrativa sobre la base de los comprobantes de soporte.

8. Recopilar la información relacionada con los ingresos, a fin de formular las diferentes proyecciones.

9. Proyectar los egresos, con sujeción a la estructura programática.

10. Recopilar los requerimientos de los recursos humanos, materiales y financieros de las distintas unidades administrativas, en concordancia con el distributivo de sueldos, el plan anual de adquisiciones y el plan anual de inversiones.

11. Participar en la formulación del plan anual de adquisiciones.

12. Mantener coordinación con las demás unidades administrativas y solicitar oportunamente informes para el registro eficiente de las operaciones económico financieras del Municipio.

13. Elaborar balances y cédulas presupuestarias, de conformidad con las disposiciones legales vigentes.

14. Coordinar con la unidad de Contabilidad para formular hasta el 31 de enero de cada año, la liquidación del presupuesto, la misma que incluirá un detalle pormenorizado de la ejecución presupuestaria del año anterior, y;

15. Las demás que le asigne el Alcalde y/o el Director Administrativo - Financiero en el área de su competencia.

Art. 16. **DEL DEPARTAMENTO DE TESORERIA**

Son funciones de este departamento:

1. Planificar, ejecutar y controlar las funciones y actividades del departamento.

2. Recibir los títulos de crédito, especies valoradas y demás documentos que amparan los ingresos municipales, verificar su legalidad y contabilización.
3. Clasificar por partidas de ingresos, los títulos de crédito, especies valoradas y demás documentos emitidos para el cobro de ingresos municipales.

4. Firmar los títulos de crédito, especies valoradas y más documentos emitidos para la recaudación de los ingresos municipales.

5. Recibir los pagos que se realicen a la municipalidad ya sea directamente o a través de Recaudadores, el Tesorero no puede negarse a recibir el pago de cualquier crédito, sea éste total o parcial, tributario, de conformidad con lo establecido en el Art. 468 de la Ley de Régimen Municipal.

6. Verificar y controlar el cobro de los intereses en mora y multas previstos en las leyes y ordenanzas, ya sea por ingresos tributarios o no tributarios.

7. Diseñar y someter a consideración del Director Administrativo - Financiero mecanismos de recaudación.

8. Llevar el control del movimiento efectivo de los ingresos propios corrientes y de capital, así como las transferencias corrientes y de capital; y de los desembolsos provenientes del crédito público.

9. Realizar los pagos de las obligaciones devengadas, con orden de autoridad competente y de conformidad con lo previsto en la Ley de Régimen Municipal y LOAFYC.

10. Depositar diariamente los ingresos recaudados en la cuenta bancaria que mantiene la municipalidad.

11. Ejercer la facultad coactiva, de conformidad con lo previsto en el Código Tributario y ley de Régimen Municipal, para lo cual deberá coordinar con el Procurador Síndico Municipal.

12. Cumplir y hacer cumplir las leyes y reglamentos, los procedimientos técnicos y administrativos sobre la Tesorería.

13. Solicitar al Servicio de Rentas Internas la devolución del Impuesto al Valor Agregado (IVA) qué se paga en la adquisición de bienes y servicios, a través del Sistema COA (Confrontación de Operaciones Auto declaradas) y las donaciones voluntarias del Impuesto a la Renta.

14. Las demás que le fueren asignadas por el Alcalde y el Director Administrativo –Financiero en el área de su competencia.

Art. 17. **DEL DEPARTAMENTO DE CONTABILIDAD**

Son funciones de este departamento:

1. Planificar y poner en conocimiento del Director Administrativo - Financiero, las funciones del Departamento.

2. Implementar el sistema de Contabilidad Gubernamental, de conformidad con las normas técnicas previstas para el efecto.

3. De conformidad con el plan de cuentas para los municipios establecido por el Ministerio de Finanzas, mantener los registros y auxiliares necesarios.

4. Preparar trimestralmente los balances presupuestarios de ingresos, gastos y flujo del efectivo, y presentarlos al Director Administrativo - Financiero.

5. Cumplir y hacer cumplir las disposiciones y procedimientos de control interno, previo y concurrentemente, conforme normas

técnicas dictadas por la Contraloría General del Estado y reglamentos internos.

6. En coordinación con el departamento de presupuesto elaborar y presentar trimestralmente las cédulas presupuestarias y demás informes relacionados con las operaciones contables y financieras a los organismos de control.

7. Efectuar sistemáticamente conciliaciones bancarias, pruebas de verificación, autenticidad de saldos, registros contables y realizar los ajustes correspondientes, en coordinación con el Director Administrativo - Financiero.

8. Proporcionar información financiera a los niveles internos, cuando lo soliciten las autoridades y a las diversas unidades administrativas.
9. En coordinación con el Director Administrativo – Financiero, realizar arqueos sorpresivos a los títulos de crédito, especies valoradas y fondo de caja chica.

10. Elaboración mensual de roles de pago.

11. Las demás que le fueren asignadas por el Alcalde y el Director Administrativo – Financiero en el área de su competencia.

Art. 18. **DEL DEPARTAMENTO DE COMPROBACION Y RENTAS**

Son funciones de este departamento.

1. Planificar y someter a aprobación del Director Administrativo - Financiero las actividades y funciones del Departamento.

2. Emitir con la debida oportunidad, y con los requisitos previstos en la Ley de Régimen Municipal y Código Tributario los títulos de crédito, las especies valoradas y demás documentos para la recaudación de los ingresos municipales.

3. Comprobar los boletines de emisión de títulos de crédito, especies valoradas y más documentos para la recaudación de los ingresos municipales.

4. Realizar la verificación de los impuestos municipales, y elaborar los actos de determinación tributaria para la firma del Director Administrativo – Financiero, con arreglo a lo dispuesto en la Ley de Régimen Municipal, Código Tributario y más leyes sobre la materia.

5. Dar trámite oportuno a los reclamos de los contribuyentes y usuarios del municipio, y preparar los actos administrativos para la firma del Director Administrativo - Financiero, de conformidad con las disposiciones de la Ley de Régimen Municipal, Código Tributario y más leyes sobre la materia.

6. Mantener el archivo clasificado y cronológico de las emisiones de boletines de títulos de crédito, especies valoradas y demás documentos para la recaudación de los ingresos municipales.
7. Verificar que las órdenes de emisión de títulos de crédito reúnan los requisitos previstos en el Código Tributario;

8. Las demás que le asigne el Director Administrativo – Financiero en el área de su competencia.

Art. 19. **DEL DEPARTAMENTO DE PROVEEDURÍA Y BODEGA**

Son funciones de este departamento:
1. Elaborar el plan de Adquisiciones de los bienes que requiera la Institución de forma trimestral, semestral y anual.

2. Solicitar al Director Administrativo - Financiero con tiempo suficiente la adquisición de los bienes y materiales que requiera la institución.
3. Recibir, almacenar, custodiar los bienes, suministros, materiales, y cuidar de su mantenimiento y distribución.

4. Mantener un registro de la entrega de bienes y materiales a las distintas dependencias de la institución, de conformidad con las disposiciones impartidas para este efecto por la Contraloría General del Estado.

5. Mantener el Archivo de actas entrega - recepción de bienes, a los funcionarios.

6. En coordinación con la Dirección Administrativa - Financiera, mantener el inventario de bienes muebles y materiales.

7. Igualmente, en coordinación con la Dirección Administrativa - Financiera, mantener el registro de los bienes inmuebles de la institución, con la especificación de si son de uso público o privado.

8. Mantener un registro de los proveedores de la municipalidad.

9. Receptar los informes con el pedido de materiales, repuestos y combustibles, de todas y cada una de las dependencias municipales.

10. Proceder al pedido de proformas a los proveedores para la adquisición de bienes de acuerdo al Reglamento Interno de Adquisiciones.

11. Elaborar el cuadro comparativo de los bienes a adquirirse y presentarlos al Comité de Adquisiciones para su resolución.

12. Ejecutar labores de recepción, clasificación, acondicionamiento de materiales, bienes muebles y otros.

13. Mantener un registro de existencias de los bienes de inversión y consumo interno de la Institución.

14. Colaborar con el planeamiento de la política de adquisiciones.

15. Las demás que le asigne el Director Administrativo – Financiero en el área de su competencia.

Art. 20. **DEL DEPARTAMENTO DE RECURSOS HUMANOS.**

Son funciones de este departamento:

1. Asesorar a las distintas dependencias en la Administración del Personal, en coordinación con el Director Administrativo - Financiero y en la selección del mismo.

2. Preparar proyectos de capacitación y evaluación del personal, los cuales se aplicarán en ascensos, remuneraciones, etc.

3. Mantener actualizado el expediente personal de cada funcionario, empleado y trabajador de la institución.

4. Llevar el registro de vacantes, nombramientos, movimientos, asistencia, permisos y tramitar las sanciones disciplinarias del personal de la institución.

5. Establecer prácticas adecuadas de supervisión del personal, evaluar el rendimiento, condiciones de trabajo y otras.

6. Preparar los registros y estadísticas del personal del municipio y planificar el calendario de vacaciones.

7. Informar mensualmente al Alcalde y Director Administrativo – Financiero, de las actividades realizadas.
8. Preparar y someter a consideración del Alcalde, reformas a los Reglamentos Internos de Administración de Recursos Humanos, Orgánico Funcional, Orgánico Estructural y otros, que mejoren la relación obrero patronal.

9. Participar en la elaboración del presupuesto de sueldos y salarios.

10. Participar en la negociación del contrato colectivo.

11. Las demás previstas en la Ley, Ordenanzas, Reglamentos Internos, y las que le asigne el Director Administrativo – Financiero en el área de su competencia.

Art. 21. **DEL DEPARTAMENTO DE SISTEMAS**

Son funciones de este departamento:

1. Ejecutar, evaluar, analizar, diseñar e implementar, los programas de Sistemas Informáticos y de radio y telecomunicaciones

2. Administración y mantenimiento de la red y sistemas.

3. Actualización y mantenimiento de equipos informáticos y de telecomunicaciones.

4. Desarrollar proyectos de telecomunicaciones, voz, datos y video.

5. Ingreso, Mantenimiento y Depuración de Bases de Datos.

6. Apoyo y fortalecimiento de sistemas de comunicación.

7. Las demás que le asigne el Director Administrativo – Financiero en el área de su competencia.

Párrafo 3°
FUNCIONES DE LA DIRECCIÓN DE OBRAS Y SERVICOS PÚBLICOS

Art. 22. **DE LA DIRECCIÓN DE OBRAS Y SERVICIOS PUBLICOS**

Son funciones de la Dirección de Obras y Servicios Públicos.

1. La Dirección de Obras Publicas tendrá a su cargo la programación, proyección y construcción de todas las obras públicas locales, por administración directa, contrato o concesión, cumpliendo las normas constructivas.

2. Coordinar los programas y proyectos de obras públicas locales, servicios públicos y someterlos a consideración del Alcalde.

3. Llevar a cabo la construcción y ejecución de los programas y proyectos aprobados.

4. Supervisar el cumplimiento de las ordenanzas municipales y demás normas relativas al tránsito en las calles, caminos y

paseos públicos en coordinación con Sindicatura y Dirección de Planificación.

5. Certificar informes de pago de combustible, lubricantes y demás requerimientos del Departamento de Transporte.

6. Reparar y mantener calles, caminos, aceras, plazas, parques y demás bienes de uso público del cantón.

7. Las demás establecidas en las Leyes sobre la materia y las que le asigne el Alcalde en el área de su competencia.

Art. 23. **DEL DEPARTAMENTO DE FISCALIZACIÓN**

Son funciones de este departamento:

1. Vigilar y responsabilizarse por el fiel y estricto cumplimiento de las cláusulas del contrato de construcción a fin de que el proyecto se ejecute de acuerdo a sus diseños definitivos, especificaciones técnicas, programas de trabajo, recomendaciones de los diseñadores y normas técnicas aplicables.

2. Detectar oportunamente errores y/u omisiones de los diseñadores, así como imprevisiones técnicas que requieran de acciones correctivas inmediatas que conjuren la situación.

3. Garantizar la buena calidad de los trabajos ejecutados.

4. Conseguir de manera oportuna las soluciones técnicas a problemas surgidos durante la ejecución de la obra.

5. Obtener que el equipo y personal técnico de las constructoras sea idóneo y suficiente para la obra.

6. Conseguir que los ejecutivos de la entidad contratante se mantengan oportunamente informados del avance de obra y problemas surgidos en la ejecución del proyecto.

7. Revisión y actualización de los programas y cronogramas presentados por el contratista.

8. Evaluación periódica del grado de cumplimiento de los programas de trabajo.

9. Sugerir durante el proceso constructivo la adopción de las medidas correctivas y/o soluciones técnicas que estime necesarias.

10. Medir las cantidades de obra ejecutadas y con ellas elaborar, verificar y certificar la exactitud de las planillas de pago, incluyendo la aplicación de las fórmulas de reajuste de precios.

11. Examinar cuidadosamente los materiales a emplear y controlar su buena calidad y la de los rubros de trabajo, a través de ensayos de laboratorio que deberá ejecutarse directamente o bajo la supervisión de su personal.

12. Las previstas en la Ley de Contratación Pública y las que dentro de su campo de responsabilidad específica le asigne el Director de Obras y Servicios Públicos.

Art. 24. **DEL DEPARTAMENTO DE TRANSPORTE.**

Son funciones de este departamento:

1. Vigilar y mantener los vehículos y maquinarias de la institución.

2. Controlar que los vehículos y maquinarias sean utilizados para fines oficiales.

3. Elaborar el presupuesto para mantenimiento de los vehículos y maquinarias.

4. Coordinar el trabajo de los vehículos y maquinarias con las demás dependencias del Municipio y establecer un cronograma de actividades de acuerdo a las necesidades que le planteen las distintas dependencias

5. Controlar el Kilometraje de los vehículos y maquinaria, a través de una hoja de ruta.

6. Cumplir y hacer cumplir las disposiciones de la LOAFYC en relación al uso que debe darse a los vehículos oficiales.

7. Emitir informes periódicos al Director de Obras y Servicios Públicos de los trabajos ejecutados por los vehículos y maquinaria.

8. Coordinar con el Departamento de Proveeduría y Bodega la emisión de las órdenes de adquisiciones.

9. Emitir informes de consumo de combustible, lubricantes y mantenimiento de los vehículos y maquinaria.

10. Las demás que le asigne el Director de Obras y Servicios Públicos en el área de su competencia.

Párrafo 4°
FUNCIONES DE LA DIRECCIÓN DE AGUA POTABLE Y ALCANTARILLADO

Art. 25. **DE LA DIRECCIÓN DE AGUA POTABLE Y ALCANTARILLADO.**

Son funciones de la dirección de Agua potable y Alcantarillado las siguientes:

1. Formular los planes y programas para la ejecución de obras de agua potable y alcantarillado, presentarlos al Alcalde a fin de que sean incluidos en los programas de desarrollo del Cantón.

2. Recomendar normas y procedimientos para reglamentar, racionalizar el uso y mantenimiento de estos servicios.

3. Proveer de agua potable a los tres sectores que conforman la Ciudad de San Gabriel.

4. Preparar informes técnicos concernientes a la dotación, mantenimiento y mejoramiento de los servicios de agua potable y alcantarillado.

5. Llevar a cabo la limpieza y mantenimiento de los elementos que conforman el sistema de agua potable y alcantarillado.

6. Efectuar conexiones y re conexiones de servicio de agua potable y suspender el mismo por falta de pago.

7. Realizar estudios necesarios, que serán puestos a consideración del Alcalde para que el Concejo cuente con los elementos de juicio necesarios para establecer presupuestos y/o aprobar tarifas de los servicios públicos de agua potable y alcantarillado.

8. Reglamentar la construcción de desagües de las aguas lluvias, servidas y autorizar su construcción.

9. Las demás que le asigne el Alcalde en el área de su competencia, y más normas sobre la materia.

Párrafo 5°

FUNCIONES DE LA DIRECCIÓN DE PLANIFICACIÓN Y URBANISMO

Art. 26. **DE LA DIRECCIÓN PLANIFICACIÓN Y URBANISMO.**

Son funciones de esta Dirección:

1. Preparar el plan de desarrollo municipal, destinado a prever, dirigir, ordenar y estimular el desenvolvimiento del Cantón, formular los planes de desarrollo físico y reguladores de desarrollo urbano.

2. Elaborar programas y proyectos de desarrollo cantonal.

3. Elaborar programas de urbanización y dictaminar los que se presenten.

4. Proceder a la zonificación y establecer las zonas de expansión urbana.

5. Elaborar los informes a fin de que el Concejo pueda autorizar las parcelaciones y reestructuración parcelaria.

6. Emitir líneas de fábrica y permisos provisionales, luego de efectuar la inspección correspondiente.

7. Revisar y aprobar los planos de planificación y construcción.

8. Seguimiento en el desarrollo y proceso constructivo de los planos aprobados.

9. Efectuar levantamientos planimétricos y topográficos solicitados por las diferentes Direcciones y contribuyentes del Cantón.

10. Asesorar a la Comisión de Centro Histórico en los diferentes proyectos y solicitudes que se presenten.

11. Preservar y cuidar de que no se destruyan los bienes, inmuebles inventariados como Patrimonio Nacional.

12. Participar como miembro de la comisión de Centro Histórico.

13. Coordinar a través de la Comisaría Municipal, los trabajos relacionados al orden, control y distribución de puestos de mercado, lugares de ventas en la ciudad y demás trabajos que realiza este Departamento.

14. Coordinar a través del Departamento de Medio Ambiente los diferentes proyectos presentados al seno del Concejo para su aprobación y posterior ejecución, al igual que los demás trabajos que realiza este Departamento.

15. Coordinar a través del Departamento de Avalúos y Catastros, el levantamiento y actualización del catastro en las áreas urbanas y rurales del cantón y demás trabajos que se realizan en este departamento.

16. Estudiar las ordenanzas referentes a control, regulación del uso del suelo, construcciones y proponer reformas.

17. Las demás que le asigne el Alcalde en el área de su competencia y demás normas sobre la materia.

Art. 27. **DEL DEPARTAMENTO DE MEDIO AMBIENTE.**

Son funciones de este departamento:

1. Elaborar Planes, Programas y Proyectos Productivos Comunitarios Agrícolas, Pecuarios, Forestales y Turísticos, con sus respectivos cronogramas de ejecución para someterlos a su aprobación.

2. Diseñar políticas de preservación y mejoramiento de la calidad de los Recursos Naturales.

3. Informar periódicamente al Director de Planificación y Urbanismo, sobre las actividades que estén en ejecución o evaluación.

4. Presentar proyectos para declaratoria de áreas protegidas a los lugares con mayor riesgo de deterioro ambiental.

5. Promover campañas de sensibilización con instituciones: educativas, públicas, privadas, comunidades, grupos organizados y otros, en educación ambiental, producción y turismo

6. Coordinar la suscripción de convenios con organizaciones Públicas, Privadas para el cumplimiento de sus objetivos.

7. Sugerir políticas de control y gestión local (ordenanzas).

8. Coordinar acciones para el manejo de desechos sólidos y relleno sanitario.

9. Implementar actividades de ornato y mantenimiento de áreas verdes, en el Cantón.

10. Coordinar el manejo adecuado de fuentes hídricas de consumo humano y riego, según concesiones.

11. Emitir informes para sancionar de acuerdo a la Ley y demás normas sobre la materia, las infracciones relacionadas con la protección de los Recursos Naturales del Cantón.

12. Llevar el catastro, registro y archivo de los expedientes correspondientes a las actividades productivas, potencialmente deterioradas del ambiente del cantón, y en general de la documentación que se origine en el desempeño de sus labores.

13. Las que le asigne el Director de Planificación y Urbanismo en el área de su competencia y demás normas sobre la materia.

Art. 28. **DEL DEPARTAMENTO DE AVALUOS Y CATASTROS.**

Son funciones de este departamento:

1. Programar, dirigir, organizar las actividades catastrales en el Cantón.

2. Mantener y actualizar los catastros de bienes inmuebles urbanos y rurales del Cantón.

3. Receptar y procesar la información para elaborar las tablas de valores de tierras y construcciones.
4. Realizar los avalúos especiales o individuales de los inmuebles urbanos - rurales del cantón de conformidad con la Ley y las ordenanzas aprobadas para el efecto.

5. Realizar avalúos y/o inventarios de valoración de los predios, en cualquier tiempo o a solicitud de los propietarios, para fines comerciales o de otra índole.

6. Mantener actualizados los registros catastrales por impuestos prediales, urbanos y rústicos, y adicionales de ley.

7. Mantener el archivo clasificado de los expedientes catastrales, con identificación del predio, propietario, descripción física del inmueble y descripción legal.

8. Informar oportunamente, sobre los reclamos de los contribuyentes en materia de avalúos, catastros e impuestos a los predios, y ejecutar las resoluciones y sentencias emanadas de autoridad competente.

9. Enviar oportunamente al Departamento de Comprobación y Rentas, los documentos sustenta torios para la emisión de los títulos de crédito. y,

10. Las demás que le asigne el Director de Planificación y Urbanismo en el área de su competencia y demás normas sobre la materia.

Art. 29. **DEL DEPARTAMENTO DE COMISARIA**

Son funciones de este departamento:

1. Cumplir y hacer cumplir las leyes, ordenanzas y reglamentos en el ámbito de su competencia.

2. Aplicar las sanciones que correspondan ante el cometimiento de infracciones o contravenciones, en materia municipal.

3. Mantener y garantizar la exactitud de Pesas y Medidas.

4. Controlar que se cumplan las disposiciones sobre higiene, salubridad, construcción, uso de vías y lugares públicos.
5. Controlar, autorizar y coordinar con la Dirección Financiera y el Departamento de Educación y Cultura la realización de espectáculos públicos dentro del cantón.

6. Organizar y dirigir el trabajo de la Policía Municipal.

7. Será el responsable de establecer con el personal de la Policía Municipal la seguridad de los personeros del Concejo, y del Alcalde.

8. Disponer la presencia de la Policía Municipal para todo acto público Organizado por la Municipalidad.

9. Planificar programas, dirigir, coordinar y evaluar el funcionamiento de los mercados.

10. Coordinar las actividades para el buen funcionamiento del Camal Municipal.

11. Vigilar el fiel cumplimiento de las normas que regulan la comercialización y manipulación de artículos para el consumo humano.

12. Conocer y resolver los reclamos del público, los arrendatarios y usuarios de los bienes municipales, y;

13. Las que le asigne el Director de Planificación y Urbanismo en el área de su competencia y, las normas sobre la materia.

Párrafo 6°
FUNCIONES DE LA DIRECCIÓN DE SERVICIOS SOCIALES Y SUS DEPARTAMENTOS

Art. 30. DE LA DIRECCIÓN DE SERVICIOS SOCIALES

Son funciones de la Dirección de Servicios Sociales:

1. Organizar y mantener servicios de salud pública y asistencia social.

2. Coadyuvar al desarrollo de la educación, cultura y comunicación del cantón.
3. Fomentar la educación pública y el deporte, en coordinación con las instituciones respectivas.

4. Elaborar planes, programas y proyectos de acuerdo al plan de desarrollo cantonal anual.

5. Proponer políticas instrumentales en temas de educación, capacitación, investigación, deporte, para la generación de conocimientos y técnicas en elementos de cultura ancestral.

6. Coordinar con organismos gubernamentales y no gubernamentales actividades dentro de su competencia.

7. Las demás que le asigne el Alcalde en el área de su competencia y las normas sobre la materia.

Art. 31. **DEL DEPARTAMENTO DE EDUCACIÓN, CULTURA Y COMUNICACIÓN**

Son funciones de este departamento:

1. Coadyuvar al desarrollo de la educación, cultura, comunicación del cantón, fomentar la educación pública y el deporte.

2. Promover planes de becas que está facultado a conceder el Municipio, de conformidad con lo establecido en la Ley de Régimen Municipal.

3. Dirigir, organizar, mantener la biblioteca pública municipal; fomentar la creación de otras, de museos de historia y arte.

4. Establecer el programa de actividades de la Banda Municipal.

5. Procurar el buen funcionamiento del Comisariato Municipal.

6. Las demás que le asigne el Director de Servicios Sociales en el área de su competencia y más normas sobre la materia.

Art. 32. **DEL DEPARTAMENTO SALUD PUBLICA**

Son funciones de este departamento:

1. Elaborar planes, proyectos, campañas con sus respectivos cronogramas de ejecución, someterlos a consideración del Director de Servicios Sociales, para la aprobación por parte del

Alcalde, buscando desarrollar una gestión planificada y sostenida a corto, mediano y largo plazo.

2. Procurar un equilibrio del capital humano por medio de la atención médica, preventiva y curativa.

3. Expender medicamentos a precios módicos y bajo prescripción médica.

4. Coordinación de actividades y programas con organismos gubernamentales y no gubernamentales para la consecución de jornadas de especialización y quirúrgicos de ayuda social.

5. Enviar a la Dirección Administrativa – Financiera recomendaciones presupuestarias para atender los requerimientos económicos de los planes y programas, en coordinación con la Dirección de Asistencia Social.

6. Presentar el plan operativo anual a la Dirección de Asistencia Social, en el primer mes de cada año.

7. Establecer normas de manejo de desechos hospitalarios.

8. Llevar un historial clínico y archivo de los beneficiarios de la atención médica, así como también un registro contable de ingresos y egresos de fármacos.

9. Las funciones que le asigne el Director de Asistencia Social en el área de su competencia y más normas pertinentes.

- **DISPOSICIONES GENERALES**

PRIMERA: En todo lo que no contemple la presente Ordenanza que contiene el presente Reglamento, se observará lo que dispone la Ley de Régimen Municipal, Ley Orgánica de Administración Financiera y Control, Ley de Servicio Civil y Carrera Administrativa sus Reglamentos Generales, Código del Trabajo y demás leyes que regulan la actividad de los Municipios.

SEGUNDA: Derogase Ordenanzas o Reglamentos en relación al Orgánico Funcional, emitidos con anterioridad al presente documento.

- **DISPOSICIÓN TRANSITORIA**

PRIMERA: La presente Ordenanza que contiene el Reglamento Orgánico Funcional entrará en vigencia a partir de su Aprobación definitiva por parte del I. Concejo Municipal.

3.3. DEPARTAMENTO DE AGUA POTABLE Y ALCANTARILLADO.

Fuente: Memorias del Sr. Antolín Revelo, miembro del Departamento de Obras Públicas del Gobierno Municipal de Montúfar.

Cuando empieza a trabajar el Sistema de Agua Potable de la Ciudad de San Gabriel, no existía un manejo técnico de volúmenes para la dotación del servicio a la población.

El ingreso del nuevo presidente del Concejo Dr. Hugo Navarrete aproximadamente en el año de 1988, se permitió una restructuración de los funcionarios municipales, asignando al Sr Armando Arévalo como Jefe de Agua Potable y Alcantarillado y al Sr. Antolín Revelo como Jefe de Avalúos y Catastros, ante esta situación, se hizo una nueva consideración y definitivamente se asignó al Sr Vinicio Oñate como Jefe de Avalúos y Catastros y al Sr. Antolín Revelo como Jefe de Agua Potable y Alcantarillado dependencia de la Dirección de Obras Públicas.

El Sistema de Agua Potable de la Ciudad de San Gabriel, fue manejado por personas que no conocían de la materia y técnicamente no distribuían ni caudales, ni pagos por el servicio, se efectuaba cartas de pago estableciendo el número de llaves de agua que existía en cada domicilio esto en el siguiente caso:

Una familia de clase media que vivían 12 personas muchas veces tenían 2 llaves de Agua, asunto contradictorio a consumo y a pago como también se daban casos que en una familia igual de clase media utilizaban el servicio 3 personas y el número de llaves eran de 9 a 10, el pago entonces decía que quien tiene mayor número de llaves debería pagan más sin tomar en cuenta consumos.

En el año de 1977 aproximadamente se inició la construcción de la planta de tratamiento de aguas ubicada en el sitio conocido como el Mirador, que servía para tratar aguas de la captación de Ramos Bajo y posteriormente de Ramos Alto. Agua que en tiempo de verano ingresaba a la planta en poco volumen y era clara mientras que en tiempo de invierno era de alta turbulencia de mayor caudal.

La planta de debía tratar unos 25 litros/segundo por lo que en verano no existía mayor variación en el caudal, en cambio en temporada de invierno por el ingreso de varia de 40 – 45 – 50 litros/segundo no se podía tratar esa turbulencia y el servicio por aquel motivo era pésimo.

La planta estaba constituida por floculadores, sedimentadores, filtros y cloraciones.
En esta planta se realizaban el retro lavado, sistema que no funciono por falta de cierre métrico de válvulas.

El sistema de Agua Potable anteriormente mencionado del año 1988 al 1999 se sufría por:

- La tubería de Agua Potable.
- Falta de Caudal
- Mal manejo del cobro de tarifas
- En tiempo de Estiaje, obligado a realizar el sistema de circuitos (distribución de agua por sectores de la ciudad) desde luego produciendo las quejas constantes de los usuarios.

Aproximadamente en el año de 1994 se efectuó un cambio de redes del Servicio de Agua Potable utilizando materiales de Asbesto, cemento y tubería PVC y el sistema que cubre la red alta y se lo considero para abastecer como la fuente de Agua Potable de la comunidad de Tanguis, de esta manera mejorando el sistema de redes mas no caudales.

En el año de 1999 se construye el nuevo sistema de Agua Potable con la Compañía COANDES S.A. la misma que realizo la captación de agua en la Comunidad del Chamizo, línea de conducción con tubería de presión PVC y diámetro 315mm y tubería de acero especialmente donde se arman los sifones, válvulas de desagüe y aire.

Dos tanques de reserva, uno ubicado en Comunidad de Chiles en el que se realiza la cloración y vertederos para medición de Volúmenes de agua y otro en el arrayan.

Un sistema nuevo de cloración se ubica en los tanques en el sector del Mirador, además se construye redes de distribución en la ciudad de San Gabriel, con sus respectivas válvulas para armar circuitos en la ciudad.

El sistema queda constituido por una red alta que abarca los barrios: Santa Clara, 27 de septiembre y parte de Cuyan.

La red media abarca los Barrios: San Vicente, San Pedro, San Antonio, San José hasta la panamericana y la red Baja que cubre los Barrios de Santa Rosa, Centenario, Los Ciprés, Santa Marta, San Andrés, Jardín del Norte y parte de Cuyan.

Con este nuevo sistema de agua potable se realiza la tarifación para el consumo y pago del servicio.

En el año de 1996 la jefatura re agua potable y alcantarillado es declarada como Departamento de Agua potable y alcantarillado y es puesta a cargo del Ing. Carlos Oña.

3.4. DESARROLLO INFORMÁTICO.

Las tecnologías de la información, actualmente son elementos fundamentales para la superación y desarrollo de un país. Por eso, los países desarrollados basan su crecimiento en la aplicación y la programación estratégica de las herramientas computacionales y han definido políticas que los inducirán a su permanencia en el dinamismo mundial de los próximos años.

El desarrollo informático es uno de los factores que siempre están en constante crecimiento, gracias a esto las empresas públicas y privadas pueden ofrecer nuevos servicios e implementar nuevas estrategias que ayuden a estar a la vanguardia, utilizando los últimos procesos con el fin de ofrecer calidad a toda la ciudadanía.

3.4.1. INTERNET

El origen de INTERNET se sitúa en la década de los años 60, como una estrategia del Departamento de Defensa de los Estados Unidos, encaminada a proveer un medio de comunicación eficiente, que soportara fallas parciales de llegarse a presentar eventuales bombardeos en su territorio; recordemos que por esa época el mundo se caracterizaba por la rivalidad entre las dos potencias de ese entonces. ARPA (Advanced Research Project Agency), la Agencia de Investigaciones de Proyectos Avanzados del Departamento de Defensa de los Estados Unidos, implementa para tal fin un Laboratorio experimental en redes, ARPAnet, que al final permitió la ampliación de enlaces y la asignación de recursos de cómputo compartidos para los demandantes de los mismos en ese país.

El funcionamiento que se le quiso dar a la red experimental, se basaba en la conversión de la información a transportar en pequeños paquetes, cada uno de los cuales se etiquetaba con la dirección electrónica de su destino final, para así poder ser enviados por diferentes puntos. Si un paquete encontraba alguna línea interrumpida, o algún impedimento similar, de inmediato y en forma automática podía localizar una trayectoria diferente para trasladarse, lo cual le permitía llegar a su destino de cualquier manera. En el extremo receptor, la computadora enlazada se encargaba de volver a ensamblar las piezas de información e iniciaba el procedimiento relacionado con el mensaje recibido. En el caso de no localizar ningún componente o de detectar mal funcionamiento de cierto elemento, la máquina emitía una nueva petición para volver a recibir el mensaje y combinar los elementos otra vez; a la utilización de este procedimiento o algoritmo se llamó Protocolo IP.

Esta es una breve reseña histórica que nos permite conocer cómo avanza el internet y con él la tecnología actual en los últimos tiempos
Para los usuarios de este servicio, los servicios básicos de INTERNET incluyen:

a) Comunicación.
b) Acceso remoto.
c) Transferencia de archivos.
d) Comunicación para acceso a información (texto, gráficas, audio y video).

Figura 2 Internet

Fuente: 10/10/201; http://es.wikipedia.org/wiki/Internet

3.5. POBLACIÓN ACTUAL.

La población del Cantón Montúfar, es un ejemplo de cordialidad y esfuerzo, siempre está buscando las maneras de salir adelante implantando nuevas herramientas, procesos para hacer de la vida un poco más sostenible.

La población se la ha distribuido entre mujeres y hombres según la fuente de la información aumento constantemente según el censo poblacional del 2011 existen las siguientes cifras.

Mujeres	**15.601**
Hombres	14.910
Total	30.511

Tabla 3 Población de la Zona

Fuente: Datos del INEC censo 2010.

4. ARQUITECTURA DEL SISTEMA

4.1. HERRAMIENTAS DE DESARROLLO

4.1.1. IDE ECLIPSE

4.1.1.1. INTRODUCCIÓN

Gran parte de la programación de Eclipse fue realizada por IBM antes de que se creara el proyecto eclipse como tal. El antecesor de Eclipse fue VisualAge y se construyó usando Smalltalk en un entorno de desarrollo llamado Envy. Con la aparición de Java en la década de los noventa, IBM desarrollo una máquina virtual valida tanto para Smalltalk como para Java. El rápido crecimiento de Java y sus beneficios con miradas en el internet en plena expansión obligaron a IBM a plantearse el abandono de esta máquina virtual dual y la construcción de una nueva plataforma basada en Java desde el principio. El Producto resultante fue Eclipse, ya que había costado unos 40 millones de dólares a IBM en el año 2001

A fines del 2001 IBM, junto a Borland, crea la fundación sin fines de lucro Eclipse, abriéndose así al mundo del código abierto. A este consorcio se ha unido poco a poco importantes empresas de desarrollo de software del mundo como: Oracle, Rational Software, Red Hat, SuSE, HP, Serena, Ericsson, y Novell y muchas más

4.1.1.2. DEFINICIÓN

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama “Aplicación de Cliente Enriquecido”, opuesta a las aplicaciones de “Cliente-livianos” basadas en navegadores. Esta plataforma, típicamente ha sido desarrollada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de java llamado Java Development Toolk (JDT) y el compilador (ECJ) que se entrega como parte del eclipse y que son usadas también para desarrollar el mismo.

4.1.1.3. ARQUITECTURA DEL IDE ECLIPSE

Un concepto fundamental de Eclipse, necesario para comprender lo que sigue, es el recurso. En Eclipse, un recurso básico es simplemente un fichero ASCII que contiene código fuente para el lenguaje de programación.
El IDE Eclipse se compone de componentes:

a) EL PLATFORM RUNTIME

Se encarga de gestionar los recursos y los plug-ins, además de permitir el arranque de la plataforma. Cuando se arranca Eclipse, este fichero se encarga de buscar los ficheros de manifiesto de los plug-ins (que son archivos XML que describen los plug-ins), y carga esta información en un registro. Solamente cuando se requiere por primera vez un plug-in, el Platform runtime lo ejecuta; este componente descubre de forma dinámica plug-ins durante el tiempo de ejecución. A grosso modo, el Platform runtime define los puntos de extensión y el modelo de plug-ins.

b) EL WORKSPACE (ESPACIO DE TRABAJO)

Permite gestionar el acceso a ficheros tanto a alto como a bajo nivel. Actúa como un componente que encapsulara la gestión de archivos, permitiendo que los plug-ins utilicen sus métodos sin tener que trabajar directamente con distintos sistemas de archivos, según la plataforma que se utilice.

De la misma manera refleja el estado actual de los proyectos locales, con su código fuente y sus ficheros compilados, que estén en la memoria activa. Al cerrar el Eclipse se gurda el estado actual del Workspace local, de modo que cuando se reinicia Eclipse vuelve al estado que se cerró.

c) EL WORKBENCH (BANCO DE TRABAJO)

Se encarga de la presentación de la información al usuario y de la gestión del dialogo con el mismo. Proporciona la interface gráfica de Eclipse y constituye uno de sus puntos más cuidados y atractivos. Desde el punto de vista del usuario una ventana del Workbench consiste en vista y editores. Tanto el API como la implementación del Workbench se han realizado mediante SWT y JFace.

- **SWT (STANDARD WIDGET TOOLKIT)**

  SWT es un conjunto de componentes para construir interfaces graficas en Java, (widgets) desarrollados por el proyecto Eclipse; además recupera la idea original de la biblioteca AWT de utilizar

componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero evita caer en las limitaciones de esta.

- **JFACES**

  JFaces es un conjunto de herramientas de interface de usuario con las clases para el manejo de muchas tareas comunes de programación de interfaz de usuario. JFaces es la ventana-independiente del sistema, tanto en su API y la aplicación y está diseñado para trabajar con SWT.

EL WORKBENCH O BANCO DE TRABAJO PROPORCIONA LO SIGUIENTE:

- **EDITORES**

  Es un componente que permite interactuar con los contenidos de un fichero (no solo con el código fuente, sino también con los ficheros XML asociado, sus propiedades, etc.) y modificarlos.

- **VISTAS**

  Una vista proporciona metadatos sobre el recurso que se haya seleccionado; organización de un recurso dentro de un paquete o proyecto, estado de la compilación, etc.

a) EL COMPONENTE DE AYUDA (HELP)

Permite a los plug-ins proporcionar documentación HTML que pueda ser presentada contextualmente por el Workbench.

b) EL COMPONENTE DEL EQUIPO (TEAM O TEAM SUPPORT)

Se define un modelo de programación en equipo para crear y mantener un registro de las versiones de las aplicaciones que se desarrolle. Este componente permite que diferentes plug-ins de repositorio convivan dentro de la plataforma; del mismo modo, añade las vistas que el usuario necesite para interaccionar con cualquier sistema de control de versiones que se esté usando. Tal como se ha mencionado ya, Eclipse incluye de forma estándar

un plug-in CVS, pero puede añadirse otro repositorio, independiente del VCM (versión Control System) que se utilice, la interface no cambia.

c) EL COMPONENTE DE DEPURACIÓN (DEBUG)

Proporciona un modelo genérico de depuración, en el que permite expresiones, puntos de interrupción, acciones habituales de depuración, junto a una interface gráfica genérica de depuración. Cualquier plug-in puede aprovechar los mecanismos de depuración que proporciona este componente.

Se debe mencionar que Platform runtime no es un plug-in; los otros cinco componentes son plug-ins, cualquiera plug-in se incorpora directamente a Eclipse se integra con este de la misma manera que cualquiera de los plug-ins que incluye de forma estándar. El workbench y el workspace son dos plug-ins indispensables porque proporcionan puntos de extensión usados por casi todos los plug-ins pero su funcionamiento no difiere de otro.

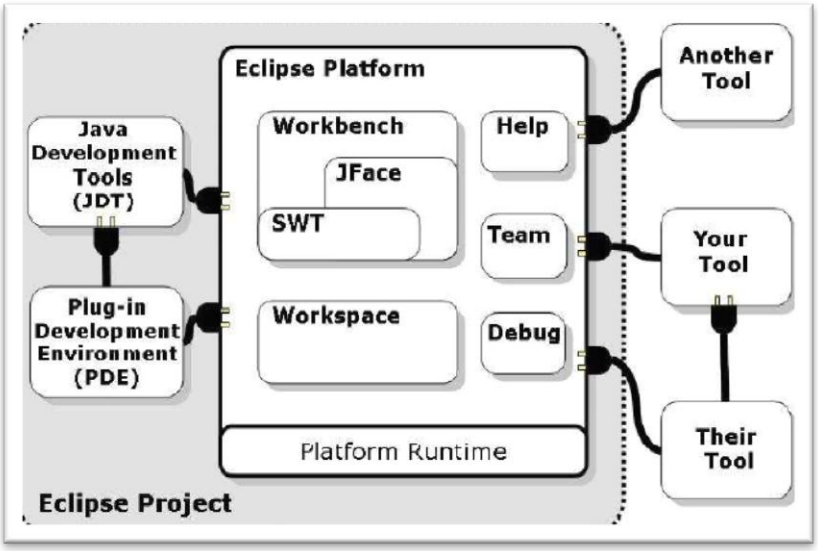


Figura 3 Arquitectura de la Plataforma Eclipse

4.1.1.4. ESTRUCTURA DE ECLIPSE

La Estructura del IDE Eclipse está compuesta por la Máquina virtual de Java y el SDK o Kit de Desarrollo Estándar de Eclipse que consta de tres elementos.

- La plataforma de Eclipse.
- La JDT (Herramienta de Desarrollo Java)
- El PDE (Entorno de Desarrollo de Plug-in)

a) LA PLATAFORMA DE ECLIPSE

Es el núcleo básico o el kernel del Eclipse; emplea una estructura abierta de plug-ins (extensiones) que permite expandir las capacidades de la plataforma base; es de arquitectura abierta, por tanto es un producto de código abierto u open source, está escrita en Java

Provee de las capas superiores de servicios tales como editor de código fuente, infraestructura para depuración independiente del lenguaje de programación, soporte de versiones, búsqueda, compilación, asistentes para creación, etc.

b) LA JDT (HERRAMIENTAS DE DESARROLLO JAVA)

Agrupa un conjunto de plug-ins que extienden la plataforma básica proporcionando características de edición, compilación, depuración y ejecución de código Java; además explica cómo entender Java, está incluido en el SDK.

c) EL PDE (ENTORNO DE DESARROLLO DE PUG-IN)

Proporciona herramientas y asistentes que automatizan y facilitan la creación, desarrollo, depuración y distribución de plug-ins.

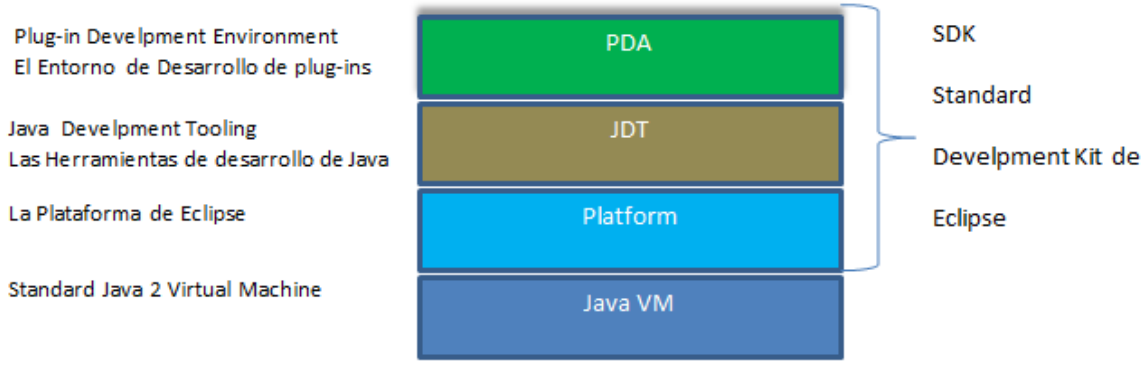


Figura 4 Estructura de IDE Eclipse

4.1.1.5. CARACTERÍSTICAS

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant.
- Asistentes (wizards): para creación de proyectos, clases, tests, exploración e importación de proyectos; para generar esqueletos de códigos (templates), etc.
- Refactorización.
- Multiplataforma (GNU/Linux, Solaris, Mac OSX, Windows)
- Soportado para distintas arquitecturas (x86, 64,…)
- Así mismo, a través de "plug-ins" libremente disponibles es posible añadir:
- Control de versiones con Subversión.
- Integración con Hibérnate
- Integración con Jboss Tools
- Portabilidad

4.1.1.6. EL PROYECTO ECLIPSE

El IDE Eclipse es, únicamente, una de las herramientas que se engloba bajo en denominado Proyecto Eclipse. El Proyecto Eclipse aún tanto en desarrollo del IDE Eclipse como de algunos de los plug-ins más importantes

(como el JDT, plug-in para el lenguaje Java o el CDT, plug-in para el lenguaje C/C++).

Este proyecto alcanza a las librerías que sirven como base para la construcción del IDE Eclipse (pero se puede utilizar de forma independiente), como por ejemplo, las librerías de widgets SWT.

4.1.1.7. SUB-PROYECTOS DE ECLIPSE

El proyector global "Eclipse" se compone de diversos sub-proyectos:

a) Proyecto Eclipse: Orientado a la realización de herramientas de desarrollo altamente integrada.
b) Proyecto de Herramientas de Eclipse
c) CDT: IDE para C/C++
d) Graphical Editor Framework (GEF): para crear editores gráficos.
e) EMF: se crea herramientas y aplicaciones que se usa el modelo estructurado
f) Cobol: IDE para cobol
g) Eclipse Virtual Editor (VE): se crea constructores de GUIs para Eclipse, utilizando Swing, SWT, etc.
h) UML2: para herramientas de modelización UML.
i) Proyecto de herramientas web de Eclipse (WTP): orientado a la creación de herramientas para el desarrollo web (editores, modelos, wizards, etc.), por ejemplo Standard Tools (WST), J2EE Standard Tools (JST), JavaServer Faces Tools (JSF), etc.
j) Herramientas de monitorización, pruebas y análisis de rendimiento (Test & Performance Tools Platform, TPTP).
k) Herramientas de Business Intelligence (BIRT).
l) Data Tools Platform (DTP).

4.1.2. LENGUAJE JAVA

Fuente: Javier García de Jalón, José Ignacio Rodríguez; 10/10/2011; Apuntes de Java;

4.1.2.1. INTRODUCCIÓN

Java se fundó como parte de un proyecto de investigación para el desarrollo de software avanzado para una amplia variedad de dispositivos de red y sistemas embebidos. La meta era diseñar una plataforma operativa sencilla, fiable, portable, distribuida y de tiempo real. Cuando se inició el proyecto, C++ era el lenguaje del momento. Pero a lo largo del tiempo las dificultades encontradas con C++ crecieron hasta el punto en que se pensó que los problemas podrían resolverse mejor creando una plataforma de lenguaje completamente nueva. Se extrajeron decisiones de diseño y arquitectura de una amplia variedad de lenguajes como Eiffel, SmallTalk, Objetive C y Cedar/Mesa. El resultado es un lenguaje que se ha mostrado ideal para desarrollar aplicaciones de usuario final seguras, distribuidas y basadas en red en un amplio rango de entornos desde los dispositivos de red embebidos hasta los sistemas de sobremesa e Internet.

4.1.2.2. OBJETIVOS DE DISEÑO DE JAVA

Java fue diseñado para ser:

- **SENCILLO, ORIENTADO A OBJETOS Y FAMILIAR**: Sencillo, para que no requiera grandes esfuerzos de entrenamiento para los desarrolladores. Orientado a objetos, porque la tecnología de objetos se considera madura y es el enfoque más adecuado para
  las necesidades de los sistemas distribuidos y/o cliente/servidor. Familiar, porque aunque se rechazó C++, se mantuvo Java lo más parecido posible a C++, eliminando sus complejidades innecesarias, para facilitar la migración al nuevo lenguaje.

- **ROBUSTO Y SEGURO**: Robusto, simplificando la gestión de memoria y eliminando las complejidades de la gestión explicita de punteros y aritmética de punteros del C. Seguro para que pueda operar en un entorno de red.

- **INDEPENDIENTE DE LA ARQUITECTURA Y PORTABLE**: Java está diseñado para soportar aplicaciones que serán instaladas en un entorno de red heterogéneo, con hardware y sistemas operativos diversos. Para hacer esto posible el compilador Java genera 'bytecodes', un formato de código independiente de la plataforma diseñado para transportar código eficientemente a través de múltiples plataformas de hardware y software. Es además portable en el sentido de que es rigurosamente el mismo lenguaje en todas las plataformas. El 'bytecode' es traducido a código máquina y ejecutado por la Java Virtual Machine, que es la implementación Java para cada plataforma hardware-software concreta.

- **ALTO RENDIMIENTO**: A pesar de ser interpretado, Java tiene en cuenta el rendimiento, y particularmente en las últimas versiones dispone de diversas herramientas para su optimización. Cuando se necesitan capacidades de proceso intensivas, pueden usarse llamadas a código nativo.

- **INTERPRETADO, MULTI-HILO Y DINÁMICO**: El intérprete Java puede ejecutar bytecodes en cualquier máquina que disponga de una Máquina Virtual Java (JVM). Además Java incorpora capacidades avanzadas de ejecución multi-hilo (ejecución simultánea de más de un flujo de programa) y proporciona mecanismos de carga dinámica de clases en tiempo de ejecución.

4.1.2.3. CARACTERÍSTICAS

- Lenguaje de propósito general.
- Lenguaje Orientado a Objetos.
- Sintaxis inspirada en la de C/C++.
- Lenguaje multiplataforma: Los programas Java se ejecutan sin variación (sin recompilar) en cualquier plataforma soportada (Windows, UNIX, Mac...)
- Lenguaje interpretado: El intérprete a código máquina (dependiente de la plataforma) se llama Java Virtual Machine (JVM). El compilador produce un código intermedio independiente del sistema denominado bytecode.

- Lenguaje gratuito: Creado por SUN Microsystems, que distribuye gratuitamente el producto base, denominado JDK (Java Development Toolkit) o actualmente J2SE (Java 2 Standard Edition).
- API distribuida con el J2SE muy amplia. Código fuente de la API disponible.

4.1.2.4. QUE INCLUYE EN J2SE (JAVA 2 STANDARD EDITION)

- Herramientas para generar programas Java. Compilador, depurador, herramienta para documentación, etc.
- La JVM, necesaria para ejecutar programas Java.
- La API de Java (jerarquía de clases).
- Código fuente de la API (Opcional).
- Documentación.

4.1.2.5. QUE ES EL JRE (JAVA RUNTIME ENVIRONMENT)

JRE es el entorno mínimo para ejecutar programas Java 2. Incluye la JVM y la API. Está incluida en el J2SE aunque puede descargarse e instalarse separadamente. En aquellos sistemas donde se vayan a ejecutar programas Java, pero no compilarlos, el JRE es suficiente.

El JRE incluye el Java Plug-in, que es el 'añadido' que necesitan lo navegadores (Explorer o Netscape) para poder ejecutar programas Java 2. Es decir que instalando el JRE se tiene soporte completo Java 2, tanto para aplicaciones normales (denominadas 'standalone') como para Applets (programas Java que se ejecutan en una página Web, cuando esta es accedida desde un navegador).

4.1.2.6. COMPONENTES

- Las Bibliotecas de Java, que son el resultado de compilar el código fuente desarrollado por quien implementa la JRE, y que ofrecen apoyo para el desarrollo en Java. Algunos ejemplos de estas bibliotecas son:

    - Las bibliotecas centrales, que incluyen:

    - Una colección de bibliotecas para implementar estructuras de datos como listas, arrays, árboles y conjuntos.
    - Bibliotecas para análisis de XML.
    - Seguridad.
    - Bibliotecas de internacionalización y localización.

- Bibliotecas de integración, que permiten la comunicación con sistemas externos. Estas bibliotecas incluyen:

    - La API para acceso a bases de datos JDBC (Java DataBase Conectivity).
    - La interfaz JNDI (Java Naming and Directory Interface) para servicios de directorio.
    - RMI (Remote Method Invocation) y CORBA para el desarrollo de aplicaciones distribuidas.
    - Bibliotecas para la interfaz de usuario, que incluyen:
    - El conjunto de herramientas nativas AWT (Abstract Windowing Toolkit), que ofrece componentes GUI (Graphical User Interface), mecanismos para usarlos y manejar sus eventos asociados.
    - Las Bibliotecas de Swing, construidas sobre AWT pero ofrecen implementaciones no nativas de los componentes de AWT.
    - APIs para la captura, procesamiento y reproducción de audio.

- Una implementación dependiente de la plataforma en que se ejecuta de la máquina virtual de Java (JVM), que es la encargada de la ejecución del código de las bibliotecas y las aplicaciones externas.

- Plug-ins o conectores que permiten ejecutar applets en los navegadores Web.

- Java Web Start, para la distribución de aplicaciones Java a través de Internet.

- Documentación y licencia.

4.1.2.7. APIS

Sun define tres plataformas en un intento por cubrir distintos entornos de aplicación. Así, ha distribuido muchas de sus APIs (Application Program Interface) de forma que pertenezcan a cada una de las plataformas:

- Java ME (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.

- Java SE (Java Platform, Standard Edition) o J2SE — para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.

- Java EE (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.

- Las clases en las APIs de Java se organizan en grupos disjuntos llamados **paquetes**. Cada paquete contiene un conjunto de interfaces, clases y excepciones relacionadas. La información sobre los paquetes que ofrece cada plataforma puede encontrarse en la documentación de ésta.

- El conjunto de las APIs es controlado por Sun Microsystems junto con otras entidades o personas a través del programa JCP (Java Community Process). Las compañías o individuos participantes del JCP pueden influir de forma activa en el diseño y desarrollo de las APIs, algo que ha sido motivo de controversia.

4.2. MODELO MVC

Después de un cuidadoso análisis de los objetivos del proyecto, se determinó que la mejor manera de estructurar el sistema era haciendo uso del patrón de diseño Model- View – Controller.

4.2.1. MODEL – VIEW – CONTROLLER

MVC (por sus siglas en inglés) es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan una gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitado la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en 3 estratos:

Modelo, Vista, Controlador los cuales se explican en breve:

4.2.1.1. MODELO

Es la representación de la información que maneja la aplicación. El modelo en si son los datos puros que puestos en contextos del sistema proveen de información al usuario o a la aplicación misma.

4.2.1.2. VISTA

Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web, la "Vista" es una página HTML con contenido dinámico sobre el cual el usuario puede utilizar operaciones.

4.2.1.3. CONTROLADOR

Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

4.2.2. CICLO DE VIDA DEL MVC

El ciclo de vida del MVC es normalmente representado por las 3 capas presentadas anteriormente y el cliente (también conocido como usuario). El siguiente diagrama representa el ciclo de vida de manera sencilla:

Figura 5 Funcionamiento MVC

El primer paso del ciclo de vida empieza cuando el usuario hace una solicitud al controlador con información con información sobre lo que el

usuario desea realizar. Entonces el controlador decide a quien delegar la tarea y es aquí donde el modelo empieza su trabajo. En esta etapa, el Modelo se encarga de realizar las operaciones sobre la información que maneja para cumplir con lo que le solicita el Controlador. Una vez que termina su labor, le regresa al Controlador la información resultante de sus operaciones, el cual a su vez dirige a la Vista. La vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al Controlador y éste se encarga de transmitírsela al usuario. El ciclo entero puede empezar nuevamente si el usuario así lo requiere.

4.2.3. VENTAJAS Y DESVENTAJAS DEL MVC

Las principales ventajas de hacer uso del patrón de diseño MVC son:

- La separación del modelo de la vista, es decir, separa los datos de la representación visual de los mismos.
- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.

- Facilita agregar nuevos tipos de datos según sea requerida por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso de ser requerido.

Las desventajas de seguir el planteamiento de MVC son:

- La separación en conceptos en capas agrega complejidad al sistema
- La cantidad de archivos a mantener y desarrollar aumenta considerablemente
- La curva de aprendizaje del patrón de diseño es mal alta que usando otros modelos más sencillos.

Cabe mencionar que la comparación de ventajas y desventajas de MVC puede ser un tema muy subjetivo y se puede presentar como tema de debate, sin embargo se tomó la decisión usando principalmente los puntos

mencionados anteriormente ya que en términos generales la balanza se inclina a favor del MVC en vez de a su contra.

4.2.4. MVC EN USO

Actualmente existen muchos Framework disponibles para el desarrollo de aplicaciones MVC; sin embargo su uso requiere de una compresión completa de su funcionamiento merecedora de temas de tesis más complejas. En este caso se deseaba obtener resultados rápidos sin involucrar más complejidad que la que el tema de propio requiere, así que se decidió seguir el esquema que plantea MVC sin hacer uso de frameworks externos.

Con el propósito de seguir el patrón de diseño MVC se determinó utilizar la tecnología JavaServer Faces para cumplir las funciones y cumplir con el diseño del framework. Para satisfacer las necesidades del Modelo se hará uso de una plataforma llamada Hibérnate, la cual se explica más adelante.

4.3. BASE DE DATOS

Fuente: Wikipedia; http://en.wikipedia.org/wiki/PostgreSQL

4.3.1. POSTGRES 8.4

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

4.3.1.1. NOMBRE DEL PRODUCTO

El uso de caracteres en mayúscula en el nombre PostgreSQL puede confundir a algunas personas a primera vista. Las distintas pronunciaciones de "SQL" pueden llevar a confusión. Los desarrolladores de PostgreSQL. Es también común oír abreviadamente como simplemente "Postgres", el que fue

su nombre original. Debido a su soporte del estándar SQL entre la mayor parte de bases de datos relacionales, la comunidad consideró cambiar el nombre al anterior Postgres. Sin embargo, el PostgreSQL Core Team anunció en 2007 que el producto seguiría llamándose PostgreSQL. El nombre hace referencia a los orígenes del proyecto como la base de datos "post-Ingres", y los autores originales también desarrollaron la base de datos Ingres.

4.3.1.2. HISTORIA

PostgreSQL ha tenido una larga evolución, la cual se inicia en 1982 con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con él mismo, Michael decidió volver a la Universidad en 1985 para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En Postgres la base de datos «comprendía» las relaciones y podía obtener información de tablas relacionadas utilizando reglas. Postgres usó muchas ideas de Ingres pero no su código.

La siguiente lista muestra los hitos más importantes en la vida del proyecto Postgres.

- 1986: se publicaron varios informes que describían las bases del sistema.
- 1988: ya se contaba con una versión utilizable.
- 1989: el grupo publicaba la versión 1 para una pequeña comunidad de usuarios.

- 1990: se publicaba la versión 2 la cual tenía prácticamente reescrito el sistema de reglas.
- 1991: publicación de la versión 3, esta añadía la capacidad de múltiples motores de almacenamiento.
- 1993: crecimiento importante de la comunidad de usuarios, la cual demandaba más características.
- 1994: después de la publicación de la versión 4, el proyecto terminó y el grupo se disolvió.

Después de que el proyecto POSTGRES terminara, dos graduados de la universidad, Andrew Yu y Jolly Chen, comenzaron a trabajar sobre el código de POSTGRES, esto fue posible dado que POSTGRES estaba licenciado bajo la BSD, y lo primero que hicieron fue añadir soporte para el lenguaje SQL a POSTGRES, dado que anteriormente contaba con un intérprete del lenguaje de consultas QUEL (basado en Ingres), creando así el sistema al cual denominaron Postgres95.

Para el año 1996 se unieron al proyecto personas ajenas a la Universidad como Marc Fournier de Hub.Org Networking Services, Bruce Momjian y Vadim B. Mikheev quienes proporcionaron el primer servidor de desarrollo no universitario para el esfuerzo de desarrollo de código abierto y comenzaron a trabajar para estabilizar el código de Postgres95.

En el año 1996 decidieron cambiar el nombre de Postgres95 de tal modo que refleje la característica del lenguaje SQL y lo terminaron llamando PostgreSQL, cuya primera versión de código abierto fue lanzada el 1 de agosto de 1996. La primera versión formal de PostgreSQL (6.0) fue liberada en enero de 1997. Desde entonces, muchos desarrolladores entusiastas de los motores de base de datos se unieron al proyecto, coordinaron vía Internet y entre todos comenzaron a incorporar muchas características al motor.

Aunque la licencia permitía la comercialización de PostgreSQL, el código no se desarrolló en principio con fines comerciales, algo sorprendente considerando las ventajas que PostgreSQL ofrecía. La principal derivación se originó cuando Paula Hawthtorn (un miembro del equipo original de Ingres que se pasó a Postgres) y Michael Stonebraker conformaron Illustra Information Technologies para comercializar Postgres.

En 2000, ex inversionistas de Red Hat crearon la empresa Great Bridge para comercializar PostgreSQL y competir contra proveedores comerciales de

bases de datos. Great Bridge auspició a varios desarrolladores de PostgreSQL y donó recursos de vuelta a la comunidad, pero a fines de 2001 cerró debido a la dura competencia de compañías como Red Hat y pobres condiciones del mercado.

En 2001, Command Prompt, Inc. lanzó Mammonth PostgreSQL, la más antigua distribución comercial de PostgreSQL. Continúa brindando soporte a la comunidad PostgreSQL a través del auspicio de desarrolladores y proyectos, incluyendo PL/Perl, PL/php y el alojamiento de proyectos de comunidades como PostgreSQL Build Farm.
En enero de 2005, PostgreSQL recibió apoyo del proveedor de base de datos Pervasive Software, conocido por su producto Btrieve que se utilizaba en la plataforma Novell Netware. Pervasive anunció soporte comercial y participación comunitaria y logró algo de éxito. Sin embargo, en julio de 2006 dejó el mercado de soporte de PostgreSQL.

A mediados de 2005 otras dos compañías anunciaron planes para comercializar PostgreSQL con énfasis en nichos separados de mercados. EnterpriseDB añadió funcionalidades que le permitían a las aplicaciones escritas para trabajar con Oracle ser más fáciles de ejecutar con PostgreSQL. Greenplum contribuyó mejoras directamente orientadas a aplicaciones de Data Warehouse e Inteligencia de negocios, incluyendo el proyecto BizGres.

En octubre de 2005, John Loiacono, vicepresidente ejecutivo de software en Sun Microsystems comentó: "No estamos yendo tras el OEM de Microsoft pero estamos viendo a PostgreSQL ahora", aunque no se dieron especificaciones en ese momento. Para noviembre de 2005, Sun Solaris 10 (lanzamiento 6/06) incluía PostgreSQL.

En agosto de 2007 EnterpriseDB anunció el Postgres Resource Center y EnterpriseDB Postgres, diseñados para ser una completamente configurada distribución de PostgreSQL incluyendo muchos módulos contribuidos y agregados. EnterpriseDB Postgres fue renombrado Postgres Plus en marzo de 2008.

El proyecto PostgreSQL continúa haciendo lanzamientos principales anualmente y lanzamientos menores de reparación de bugs, todos disponibles bajo la licencia BSD, y basados en contribuciones de

proveedores comerciales, empresas aportantes y programadores de código abierto mayormente.

4.3.1.3. CARACTERÍSTICAS

Algunas de las características de PostgreSQL son:

4.3.1.3.1. ALTA CONCURRENCIA

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.....

4.3.1.3.2. AMPLIA VARIEDAD DE TIPOS NATIVOS

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

4.3.1.3.3. OTRAS CARACTERÍSTICAS

Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
Disparadores (triggers): Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro

de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:

- El nombre del disparador o trigger
- El momento en que el disparador debe arrancar
- El evento del disparador deberá activarse sobre...
- La tabla donde el disparador se activará
- La frecuencia de la ejecución
- La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

Soporte para transacciones distribuidas. Permite a PostgreSQL integrase en un sistema distribuido formado por varios recursos (p.ej, una base de datos PostgreSQL, otra Oracle, una cola de mensajes IBM MQ JMS y un ERP SAP) gestionado por un servidor de aplicaciones donde el éxito ("commit") de la transacción global es el resultado del éxito de las transacciones locales.

4.3.1.4. FUNCIONES

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.

Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos.
Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de Oracle).

- C.
- C++.
- JavaPL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.
- Lenguaje para aplicaciones estadísticas R por medio de PL/R.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).

4.3.1.5. PRODUCTOS ALREDEDOR DE POSTGRES

El PGDG solo desarrolla el Motor de Datos y un número pequeño de utilidades, para potenciar el trabajo con PostgreSQL suele ser necesario añadir utilidades externas creadas especialmente para este motor, algunas de estas herramientas son:

4.3.1.6. ALTERNATIVAS COMERCIALES

Gracias a su licencia BSD, se permite la utilización del código para ser comercializado. Uno de los casos ejemplo es la de Enterprise DB (Postgresql Plus), la cual incluye varios agregados y una interfaz de desarrollo basada en Java. Entre otras empresas que utilizan PostgreSQL para comercializar se encuentra CyberTech (Alemania), con su producto CyberCluster.

4.3.1.7. GIS

PostGIS: Extensión que añade soporte de objetos geográficos a PostgreSQL y permite realizar análisis mediante consultas SQL espaciales o mediante conexión a aplicaciones GIS (Sistema de Información Geográfica).

4.3.1.8. REPLICACIÓN

- PgCluster: Replicación multi maestro.
- Slony-IReplicación maestro esclavo.
- PyReplica: Replicación maestro esclavo y multi maestro asincrónica

4.3.1.9. HERRAMIENTAS ADMINISTRATIVAS

- PgAdmin3: Entorno de escritorio visual.
- PgAccess: Entorno de escritorio visual.
- PhpPgAdmin: Entorno web.
- Psql Cliente de consola.
- Database MasterEntorno de escritorio visual.

4.3.1.10. USUARIOS DESTACADOS

- La American Chemical Society.
- BASF.
- IMDb.
- Skype.
- TiVo.
- Penny Arcade.
- Sony Online.2
- U.S. Departamento de Trabajo.
- USPS.
- VeriSign.
- Pictiger.com
- Wisconsin Circuit Court Access con 6 * 180GB DBs replicados en tiempo real.
- OpenACS y .LRN.
- INEGI.
- IFE.

4.4. SERVIDOR DE APLICACIONES

4.4.1. TOMCAT

4.4.1.1. INSTRUCCIÓN

Tomcat es un contenedor de Servlets con un entorno JSP. Un contenedor de Servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Podemos dividir los contenedores de Servlets en:

- **CONTENEDORES DE SERVLETS STAND-ALONE (INDEPENDIENTES)**

Estos son una parte integral del servidor web. Este es el caso cuando usando un servidor web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWeb Server (actualmente sustituido por **iPlanet**). Este el modo por defecto usado por Tomcat.
Sin embargo, la mayoría de los servidores, no están basados en Java, los que nos lleva los dos siguientes tipos de contenedores

- **CONTENEDORES DE SERVLETS DENTRO-DE-PROCESO**

El contenedor Servlet es una combinación de un plug-in para el servidor web y una implementación de contenedor Java. El plug-ind del servidor web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. Si una cierta petición debería ejecutar un servlet, el plug-in toma el control sobre la petición y lo pasa al contenedor Java (usando JNI). Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad

- **CONTENEDORES DE SERVLETS FUERA-DE-PROCESO**

El contenedor Servlet es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets TCP/IP). Si una cierta petición debería ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior,

pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (actualmente se soportan los servidores Apache, IIS y Netscape). Esto significa que siempre que despleguemos Tomcat tendremos que decidir cómo usarlo, y, si seleccionamos las opciones 2 o 3, también necesitaremos instalar un adaptador de servidor web

4.4.1.2. ESTRUCTURA DE DIRECTORIOS DE TOMCAT

Asumiendo que hemos descomprimido la distribución binaria del Tomcat deberíamos tener la siguiente estructura de directorios:

NOMBRE DEL DIRECTORIO	DESCRIPCIÓN
bin	Contiene los scripts de arrancar/parar
conf	Contiene varios ficheros de configuración incluyendo server.xml (el fichero de configuración principal de Tomcat) y web.xml que configura los valores por defecto para las distintas aplicaciones desplegadas en Tomcat
doc	Contiene varia documentación sobre Tomcat (Este manual, en inglés).
lib	Contiene varios ficheros jar que son utilizados por Tomcat. Sobre UNIX, cualquier fichero de este directorio se añade al classpath de Tomcat.
los	Aquí es donde Tomcat sitúa los ficheros de diario
src	Los ficheros fuentes del API Servlet. ¡No te excites, todavía! Esto son sólo los interfaces vacíos y las clases abstractas que debería implementar cualquier contenedor de servlets
webapps	Contiene aplicaciones Web de Ejemplo.

Tabla 4 Estructura de Directorios de Tomcat

Adicionalmente podemos, o Tomcat crea los siguientes directores:

NOMBRE DEL DIRECTORIO	DESCRIPCIÓN
work	Generado automáticamente por Tomcat, este es el sitio donde Tomcat sitúa los ficheros intermedios (como las páginas JSP compiladas) durante su trabajo. Si borramos este directorio mientras se está ejecutando Tomcat no podremos ejecutar páginas JSP.
Clases	Podemos crear este directorio para añadir clases adicionales al classpath. Cualquier clase que añadamos a este directorio encontrará un lugar en el classpath de Tomcat.

Tabla 5 Directorios creados por Tomcat

4.4.1.3. LOS SCRIPTS DE TOMCAT

Tomcat es un programa Java, y por lo tanto es posible ejecutarlo desde la línea de comandos, después de configurar varias variables del entorno. Sin embargo, configurar cada variable de entorno y seguir los parámetros de la línea de comandos usados por el Tomcat es tedioso y propenso a errores.

NOMBRE DEL SCRIPT	DESCRIPCIÓN
Tomcat	El script principal. Configura el entorno apropiado, incluyendo CLASSPATH, TOMCAT_HOME y JAVA_HOME, y arranca Tomcat con los parámetros de la línea de comando apropiados
Startup	Arrancar Tomcat en segundo plano. Acceso directo para Tomcat start
Shutdown	Para Tomcat (lo apaga). Acceso directo para Tomcat stop;

Tabla 6 Scripts de Tomcat

El script más importante para los usuarios es Tomcat (tomcat.sh/tomcat.bat). Los otros scripts relacionados con Tomcat sirven como un punto de entrada simplificado a una sola tarea (configuran diferentes parámetros de la línea de comandos, etc.).

4.4.1.4. FICHEROS DEL TOMCAT

La configuración de Tomcat se basa en dos ficheros

- server.xml - El fichero de configuración global de Tomcat.
- web.xml - Configura los distintos contextos en Tomcat.

Esta sección trata la forma de utilizar estos ficheros. No vamos a cubrir las interioridades de web.xml, esto se cubre en profundidad en la especificación del API Servlet. En su lugar cubriremos el contenido de server.xml y discutiremos el uso de web.xml en el contexto de Tomcat

➢ **SERVER.XML**

Es el fichero de configuración principal de Tomcat. Sirve para dos objetivos:

1. Proporcionar configuración inicial para los componentes de Tomcat.

2. Especifica la estructura de Tomcat, lo que significa, permitir que Tomcat arranque y se construya a sí mismo ejemplarizando los componentes especificados en server.xml.

Los elementos más importantes de server.xml se describen en la siguiente tabla

ELEMENTO	DESCRIPCIÓN
Server	El elemento superior del fichero server.xml. Server define un servidor Tomcat. Generalmente no deberíamos tocarlo demasiado. Un elemento Server puede contener elementos Logger y ContextManager.
Logger	Este elemento define un objeto logger. Cada objeto de este tipo tiene un nombre que lo identifica, así como un path para el fichero log que contiene la salida y un verbosityLevel (que especifica el nivel de log). Actualmente hay loggeres para los servlets (donde va el ServletContext.log()), los ficheros JSP y el sistema de ejecución tomcat.
ContextManager	Un ContextManager especifica la configuración y la estructura para un conjunto de ContextInterceptors,

	RequestInterceptors, Contexts y sus Connectors. El ContextManager tiene unos pocos atributos que le proporcionamos con: Nivel de depuración usado para marcar los mensajes de depuración La localización base para webapps/, conf/, logs/ y todos los contextos definidos. Se usa para arrancar Tomcat desde un directorio distinto a TOMCAT_HOME. El nombre del directorio de trabajo. Se incluye una bandera para controlar el seguimiento de pila y otra información de depurado en las respuestas por defecto.
ContextInterceptor & RequestInterceptor	ContextInterceptor escucha los eventos de arrancada y parada de Tomcat, y RequestInterceptor mira las distintas fases por las que las peticiones de usuario necesitan pasar durante su servicio. El administrador de Tomcat no necesita conocer mucho sobre los interceptores; por otro lado, un desarrollador debería conocer que éste es un tipo **global** de operaciones que pueden implementarse en Tomcat (por ejemplo, loggin de seguridad por petición).
Connector	El Connector representa una conexión al usuario, a través de un servidor Web o directamente al navegador del usuario (en una configuración independiente). El objeto connector es el responsable del control de los threads en Tomcat y de leer/escribir las peticiones/respuestas desde los sockets conectados a los distintos clientes. La configuración de los conectores incluye información como: La clase handler. El puerto TCP/IP donde escucha el controlador. el backlog TCP/IP para el server socket del controlador. Describiremos cómo se usa esta configuración de conector más adelante.
Context	Cada Context representa un path en el árbol de tomcat donde situamos nuestra aplicación web. Un Context Tomcat tiene la siguiente configuración: El path donde se localiza el contexto. Este puede ser

	un path completo o relativo al home del ContextManager. Nivel de depuración usado para los mensaje de depuración. Una bandera reloadable. Cuando se desarrolla un servlet es muy conveniente tener que recargar el cambio en Tomcat, esto nos permite corregir errores y hacer que Tomcat pruebe el nuevo código sin tener que parar y arrancar. Para volver a recargar el servlet seleccionamos la bandera reloadable a true. Sin embargo, detectar los cambios consume tiempo; además, como el nuevo servlet se está cargando en un nuevo objeto **class-loader** hay algunos casos en los que esto lanza errores de forzado (cast). Para evitar estos problemas, podemos seleccionar la bandera reloadable a false, esto desactivará esta característica.

Tabla 7 Elementos del server.xml

Se puede encontrar información adicional dentro del fichero server.xml.

- **ARRANQUE DE TOMCAT DESDE OTRO DIRECTORIO.**

Por defecto tomcat usará TOMCAT_HOME/conf/server.xml para su configuración. La configuración por defecto usará TOMCAT_HOME como la base para sus contextos.

Podemos cambiar esto usando la opción -f /path/to/server.xml, con un fichero de configuración diferente y configurando la propiedad home del controlador de contexto. Necesitamos configurar los ficheros requeridos dentro del directorio home:

- Un directorio webapps/ (si hemos creado uno) - todos los ficheros **war** se desplegará y todos sus subdirectorios se añadirán como contextos.

- Directorio conf/ - podemos almacenar tomcat-users.xml y otros ficheros de configuración.

- logs/ - todos los logs irán a este directorio en lugar de al principal TOMCAT_HOME/logs/.

- work/ - directorio de trabajo para los contextos.

- Si la propiedad ContextManager.home de server.xml es relativa, será relativa al directorio de trabajo actual

➢ **WEB**

Podemos encontrar una detallada descripción de web.xml y la estructura de la aplicación web (incluyendo la estructura de directorios y su configuración) en los capítulo 9, 10 y 14 de la Servlet API Spec en el site de Sun Microsystems.

Hay una pequeña característica de Tomcat que está relacionada con web.xml. Tomcat permite al usuario definir los valores por defecto de web.xml para todos los contextos poniendo un fichero web.xml por defecto en el directorio conf. Cuando construimos un nuevo contexto, Tomcat usa el fichero web.xml por defecto como la configuración base y el fichero web.xml específico de la aplicación (el localizado en el WEB-INF/web.xml de la aplicación), sólo sobrescribe estos valores por defecto.

4.4.2. JBOSS

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa
- Adaptabilidad, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java

- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

4.4.2.1. VERSIONES

- 4.0: Servidor de aplicaciones Java EE 1.4.

- Contenedor de servlet **Apache Tomcat 5.5** embebido. Soporta JVM v1.4 – v1.6. Corre en cualquier OS con una JVM apropiada

- 4.2: También funciona como Java EE 1.4, pero destaca EJB 3.0 por defecto. (Requiere JDK 5)

- 5.1: Opera como un servidor de aplicaciones Java EE 5. Es un update menor sobre la versión 5.0 que estuvo en desarrollo por más de 3 años construida sobre un nuevo microcontenedor.

- 6.0: Opera bajo la especificación EE6. SoportaJSF 2 (AJAX entre otros). Validación de Beans.CDI (Administración de contexto), balanceo descarga inteligente.

  Fuente: Wikipedia; http://es.wikipedia.org/wiki/Tomcat

4.5. FRAMEWORK

4.5.1. JAVA SERVER FACES (JSF)

La tecnología Java Server Faces es un framework de interface de componentes de usuarios del lado del servidor para las aplicaciones web basadas en la tecnología Java.
El objetivo de la tecnología Java Server Faces es desarrollar aplicaciones web de forma parecida a como se construyen aplicaciones locales con Java Swing, AWT (Abstract WindowToolkit), SWT (Standard Widget Toolkit) o cualquier otra API similar.

Tradicionalmente, las aplicaciones web se han codificado mediante páginas JSP (Java Server Pages) que recibían peticiones a través de formularios y construían como respuesta páginas HTML (Hiper Text Markup Language) mediante ejecución directa o indirecta -a través de bibliotecas de etiquetas- de código Java, lo que permitía, por ejemplo, acceder a bases de datos para obtener los resultados a mostrar una vez realizada las operaciones básicas de una aplicación como insertar o modificar, borrar información seleccionada por el usuario.

JavaServer Faces pretende facilitar la construcción de estas aplicaciones proporcionando un entorno de trabajo (framework) vía web que gestiona las acciones producidas por el usuario en su página HTML y las traduce a eventos que son enviados al servidor con el objetivo de regenerar la página original y reflejar los cambios pertinentes provocados por dichas acciones.

En definitivas cuentas, se trata de hacer aplicaciones Java en las que el cliente no es una ventana de la clase JFrame o similar, sino una página HTML.

Como podemos apreciar, cualquier evento realizado sobre una página JSF incurre en una carga sobre la red, ya que el evento debe enviarse a través de ésta al servidor, y la respuesta de éste debe devolverse al cliente; por ello, el diseño de aplicaciones JavaServer Faces debe hacerse con cuidado cuando se pretenda poner las aplicaciones a disposición del mundo entero a través de internet. Aquellas aplicaciones que vayan a ser utilizadas en una intranet podrán aprovecharse de un mayor ancho de banda y producirán una respuesta mucho más rápida

4.5.1.1. CARACTERÍSTICAS DEL JSF

La tecnología JavaServer Faces constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).

Los principales componentes de la tecnología JavaServer Faces son:

- Una API y una implementación de referencia para:
- Representar componentes de interfaz de usuario (UI-User Interface) y manejar su estado
- Manejar eventos, validar en el lado del servidor y convertir datos
- Definir la navegación entre páginas
- Soportar internacionalización y accesibilidad, y
- Proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Este modelo de programación bien definido y la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado del servidor. Con un mínimo de esfuerzo, es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado servidor.
- Mapear componentes UI a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

En la siguiente figura podemos observar la interfaz del usuario que se crea con la tecnología JSF que se encuentra representada por el ítem **miUI** se crea en el lado del cliente y se ejecuta en el lado del servidor de aplicaciones.

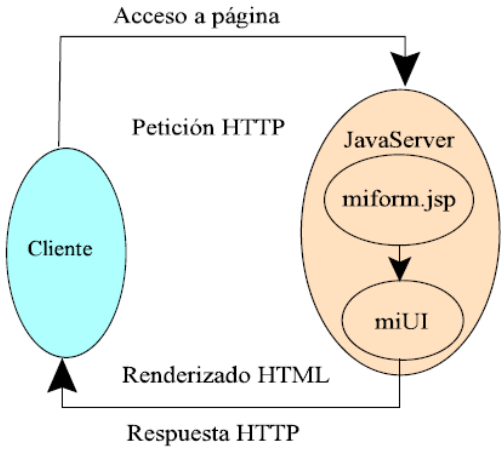


Figura 6 Diagrama de una aplicación JSF

La página JSP, miform.jsp, especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología JavaServer Faces. LA UI de la aplicación web (representada por miUI en la figura) maneja los objetos referenciados por la JSP, que pueden ser de los siguientes tipos:

- Objetos componentes que mapean las etiquetas sobre la página JSP.
- Oyentes de eventos, validadores y conversores registrados y asociados a los componentes.
- Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica de negocio).

4.5.1.2. BENEFICIOS DE LA TECNOLOGÍA JAVASERVER FACES

- Una de las grandes ventajas de la tecnología JavaServer Faces es que ofrece una clara separación entre el comportamiento y la presentación. Las aplicaciones Web construidas con tecnología JSP conseguían parcialmente esta separación. Sin embargo, una aplicación JSP no puede mapear peticiones HTTP al manejo de eventos específicos de los componentes o manejar elementos UI como objetos con estado en el servidor.

- La tecnología JavaServer Faces permite construir aplicaciones Web que implementan una separación entre el comportamiento y la presentación tradicionalmente ofrecida por arquitectura UI del lado del cliente. JSF se hace fácil de usar al aislar al desarrollador del API de Servlet.

- La separación de la lógica de la presentación también le permite a cada miembro del equipo de desarrollo de una aplicación Web enfocarse en su parte del proceso de desarrollo, y proporciona un sencillo modelo de programación para enlazar todas las piezas.

- Otro objetivo importante de la tecnología JavaServer Faces es mejorar los conceptos familiares Disadvantage de componente-UI y capa-Web sin limitar a una tecnología de script particular o un lenguaje de marcas. Aunque la tecnología JavaServer Faces incluye una librería de etiquetas JSP personalizadas para representar componentes en una página JSP, los APIs de la tecnología JavaServer Faces se han creado directamente sobre el API JavaServlet. Esto permite hacer algunas cosas: usar otra tecnología de presentación junto a JSP, crear componentes propios personalizados directamente desde las clases de componentes, y generar salida para diferentes dispositivos cliente. Así, se podrán encapsular otras tecnologías como Ajax en componentes JSF, haciendo su uso más fácil y productivo, al aislar al programador de ellas.

- JavaServer Faces ofrece una gran cantidad de componentes opensource para las funcionalidades que se necesiten. Los componentes Richfaces, Tomahawk de MyFaces y ADFFaces de Oracle son un ejemplo. Además, también existe una gran cantidad de herramientas para el desarrollo IDE en JSF al ser el estándar de JAVA.

- La tecnología JavaServer Faces proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario, y manejar eventos.

- Además, ofrece una rápida adaptación para nuevos desarrolladores.

4.5.1.3. DISTINTAS IMPLEMENTACIONES DE JSF

Actualmente existen muchas librerías de etiquetas JSF que pueden complementar a la implementación de la especificación oficial. La elección no tiene porqué cerrarse sobre una de ellas sino que pueden combinarse según interese.

4.5.1.3.1. MYFACES TOMAHAWK

Desarrollado por Apache. Este conjunto de componentes también es compatible con la implementación de SUN, así como con cualquier implementación compatible con JSF 1.1.
Pueden verse los distintos componentes de MyFaces Tomahawk en el siguiente enlace:
http://www.irian.at/myfaces/
http://www.marinschek.com/myfaces/tiki/tiki-index.php?page=Features

4.5.1.3.2. MYFACES SANDBOX

Desarrollado por Apache: http://myfaces.apache.org/sandbox/ Sandbox es un subproyecto de MyFaces que sirve como base de pruebas para las nuevas incorporaciones al proyecto de Tomahawk. Consiste sobre todo en componentes, pero como el proyecto de Tomahawk, puede también contener otras utilidades para JSF.

4.5.1.3.3. ICEFACES

Desarrollado por ICEsoft: http://www.icesoft.com/products/icefaces.html
ICEFaces proporciona un entorno de presentación web para aplicaciones JSF que mejora el framework JSF estándar y el ciclo de vida con características interactivas basadas en AJAX. Para trabajar con ICEfaces puede elegirse cualquiera de las dos implementaciones estándar. En la siguiente dirección web pueden encontrarse demos sobre sus componentes: http://www.icesoft.com/products/demos_icefaces.html

4.5.1.3.4. RICHFACES

JBoss RichFaces es una biblioteca de componentes para JSF. Ahora, RichFaces no sustituye a la norma JSF; utilizar RichFaces, ya sea con el JSF mojara (Sun RI) la aplicación o la MyFaces implementación. RichFaces simplemente proporciona listas para utilizar los componentes Ajax para permitir la creación de aplicaciones basadas en Ajax. Otra forma de verlo es como un montón de extras más allá de los componentes JSF lo que el estándar de JSF proporciona. Estos componentes proporcionan todo lo necesario JavaScript, por lo que casi nunca se tiene que trabajar con él directamente.

Ahora, para tomar un paso más allá, RichFaces es en realidad un marco. Una de sus características principales son los componentes ricos que ofrece. Los componentes se dividen en las bibliotecas de etiquetas. Además, RichFaces proporciona una kinnability (temas) de características y el desarrollo de componentes Ajax4jsf Kit (CDK)

Figura 7 Arquitectura RichFaces

Son características de RichFaces las siguientes:

- Se integra perfectamente en el ciclo de vida de JSF.
- Incluye funcionalidades Ajax, de modo que nunca vemos el JavaScript y tiene un contenedor Ajax propio.
- Contiene un set de componentes visuales, los más comunes para el desarrollo de una aplicación web rica (Rich Internet Application), con un número bastante amplio que cubren casi todas nuestras necesidades.
- Soporta facelets, soporta css themes o skins.
- Es un proyecto opensource, activo y con una comunidad también activa

El funcionamiento del framework es sencillo. Mediante sus propias etiquetas se generan eventos que envían peticiones al contenedor Ajax. Estos eventos se pueden ejecutar por pulsar un botón, un enlace, una región específica de la pantalla, un cambio de estado de un componente, etc. Esto significa que no nos preocuparemos de crear el código Javascript y el objeto XMLHttpRequest para que envíe la petición al servidor ya que el framework lo hará por nosotros. Algunas etiquetas son:

4.5.1.4. AJAX4JSF

ETIQUETA	DESCRIPCIÓN
aj4:support	Etiqueta que se puede añadir a cualquier otra etiqueta JSF para dotarla de funcionalidad Ajax. Permite al componente generar peticiones asíncronas mediante eventos (onclick, onblur, onchange,...) y actualizar campos de un formulario de forma independiente, sin recargar toda la página.
aj4:poll	Realiza cada cierto tiempo una petición al servidor
aj4:commandButton	Botón de envío de formulario similar al de JSF. La principal diferencia es que se puede indicar que únicamente actualice ciertos componentes evitando la recarga de todo el formulario.
aj4:htmlCommandLink	Muy parecida a la etiqueta anterior con pequeñas diferencias en la generación de links y cuando se utilizan etiquetas < f:param >
aj4:region	Determina un área a decodificar en el servidor después de la petición Ajax
aj4:status	Muestra el estado de la petición Ajax. Hay 2 estados posibles: procesando petición y petición terminada. Por ejemplo mientras dure el proceso de la llamada al servidor y la evaluación de la petición se puede mostrar el texto "procesando..." y cuando termine la petición y se devuelva la respuesta a la página se cambia el texto por "petición finalizada".
aj4:form	Similar al < h:form> con la diferencia de que se puede enviar previamente el contenido al contenedor Ajax.
aj4:actionparam	Etiqueta que combina la funcionalidad de la etiqueta < f:param> y < f:actionListener >.
aj4:outputPanel:	Se utiliza para agrupar componentes para aplicarles similares propiedades, por ejemplo a la hora de actualizar sus valores tras la petición Ajax.
aj4:ajaxListener	Similar a la propiedad actionListener o valueChangeListener pero con la diferencia de que la petición se hace al contenedor Ajax.

Tabla 8 Algunos componentes Ajax4jsf

4.5.1.5. RICHFACES

ETIQUETA	DESCRIPCIÓN
rich:calendar	Este componente se utiliza para crear elementos de calendario
rich:comboBox	Este es un componente, que proporciona combo Box editable
rich:componentControl	Este permite llamar a funciones API de JavaScript en los componentes definidos después de los acontecimientos
rich:datascroller	El componente diseñado para proporcionar la funcionalidad de los cuadros de desplazamiento utilizando Ajax solicitudes
rich:columns	Es un componente, que le permite crear una columnas dinámica
rich:columnGroup	Este componente nos permite combinar las columnas en una fila para organizar
rich:dataTable	Este componente nos permite crear tablas de datos.
rich:dataGrid	Este componente permite ver los datos como una rejilla que nos deja elegir los datos.
rich:menuItem	Este componente se utiliza para la definición de un único punto dentro de una lista emergente
rich:message	El componente se utiliza para hacer un solo mensaje a un componente específico.
rich:menuGroup	Este componente se utiliza para definir un ampliable grupo de temas dentro de una lista emergente u otro grupo.

Tabla 9 Algunos componentes RichFaces

4.5.2. HIBÉRNATE

Fuente: Héctor Suárez González; 21/03/2003; Manual Hibérnate.

4.5.2.1. CAPAS DE PERSISTENCIA Y POSIBILIDADES QUE ESTA OFRECE

En el diseño de una aplicación (me referiré a una aplicación a desarrollar utilizando Java) una parte muy importante es la manera en la cual accedemos a nuestros datos en la base de datos (BBDD) determinar esta parte se convierte en un punto crítico para el futuro desarrollo.
La manera tradicional de acceder seria a través de JDBC directamente conectado a la
BBDD mediante ejecuciones de sentencias SQL

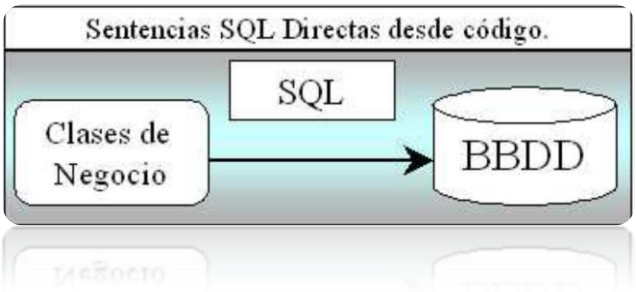


Figura 8 Acceso a la base de datos de forma tradicional

Esta primera aproximación puede ser útil para proyectos o arquitecturas sin casi clases de negocio, ya que el mantenimiento del código está altamente ligado a los cambios en el modelo de datos relacional de la BBDD, un mínimo cambio implica la revisión de casi todo el código así como su compilación y nueva instalación en el cliente.

Aunque no se puede desechar su utilidad. El acceso a través de sentencias SQL directas puede ser utilizado de manera puntual para realizar operaciones a través del lenguaje SQL lo cual sería mucho más efectivo que la carga de gran cantidad de objetos en memoria. Si bien un buen motor de persistencia debería implementar mecanismos para ejecutar estas operaciones masivas sin necesidad de acceder a este nivel.

Una aproximación más avanzada seria la creación de unas clases de acceso a datos (DAO Data Acces Object). De esta manera nuestra capa de negocio

interactuaría con la capa DAO y esta sería la encargada de realizar las operaciones sobre la BBDD.

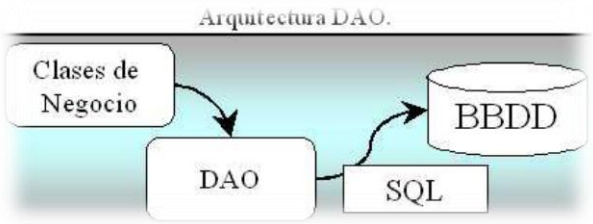


Figura 9 Ejemplo de DAO (Data Access Object)

Los problemas de esta implementación siguen siendo el mantenimiento de la misma así como su portabilidad. Lo único que podemos decir es que tenemos el código de transacciones encapsulado en las clases DAO.

Lo que parece claro es que debemos separar el código de nuestras clases de negocio de la realización de nuestras sentencias SQL contra la BBDD. Por lo tanto Hibernate es el puente entre nuestra aplicación y la BBDD, sus funciones van desde la ejecución de sentencias SQL a través de JDBC hasta la creación, modificación y eliminación de objetos persistentes

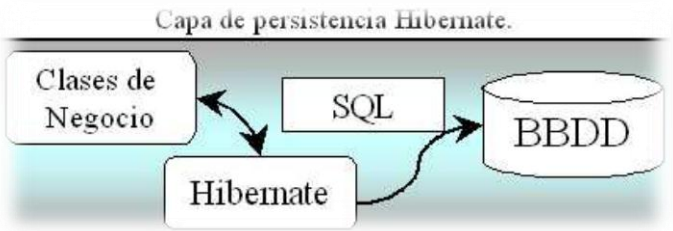


Figura 10 Persistencia con Hibernate

Con la creación de la capa de persistencia se consigue que los desarrolladores no necesiten conocer nada acerca del esquema utilizado en la BBDD. Tan solo conocerán el interface proporcionado por el motor de persistencia. De esta manera conseguimos separar de manera clara y definida, la lógica de negocios de la aplicación con el diseño de la BBDD.

4.5.2.2. QUE ES HIBERNATE

Es un framework de mapeo Objeto Relacional open source. Una de las principales características de Hibernate es que permite tener persistencia con los datos transforman modelos de diseño de datos a formato XML de acuerdo a DTD definidos por Hibernate

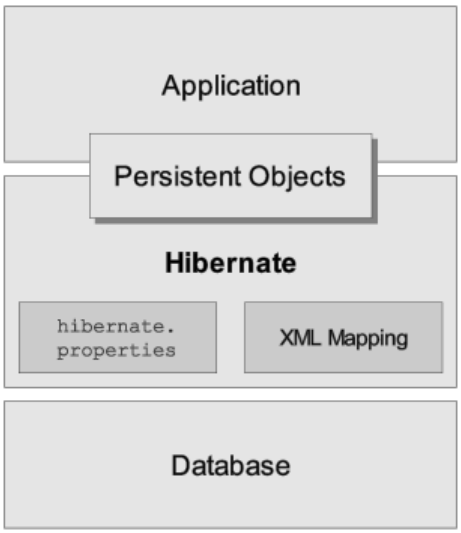


Figura 11 Diagrama de Bloques Hibernate

Hibernate es un framework que se ubica en la capa de persistencia objeto/relacional y es un generador de sentencias SQL, permite manejar objetos persistentes, que podrán incluir características tales como: polimorfismo, colecciones, relaciones, y múltiples tipos de datos. De manera muy optimizada y rápida se pueden generar mapeos Objeto Relacionales en cualquiera de los entornos soportados (tiene soporte para todos los sistemas gestores de bases de datos y se integra de forma elegante y sin restricción alguna con los más conocidos contenedores web y servidores de aplicaciones J2EE, también puede utilizarse en aplicaciones standalone). Hibernate es la solución ORM más conocida en las aplicaciones Java. Permite implementar clases persistentes a partir de pojos no importando si incluyen composición, colecciones de objetos, polimorfismo, herencia y asociación. Hibernate provee una forma muy elegante para acceder a la información de las bases de datos relacionales, esto es gracias a su lenguaje de consultas HQL (Hibernate Query Language), el cual tiene un diseño como una mínima extensión orientada a objetos de SQL.

Hibernate también permite realizar consultas basadas en criterios o en SQL puro

4.5.2.3. ARQUITECTURA

El API de Hibernate es una arquitectura de dos capas (Capa de persistencia y Capa de Negocio).

En la siguiente figura se muestran los roles de las interfaces Hibernate más importantes en las capas de persistencia y de negocio de una aplicación J2EE. La capa de negocio está situada sobre la capa de persistencia, debido a que actúa como un cliente de la capa de persistencia.

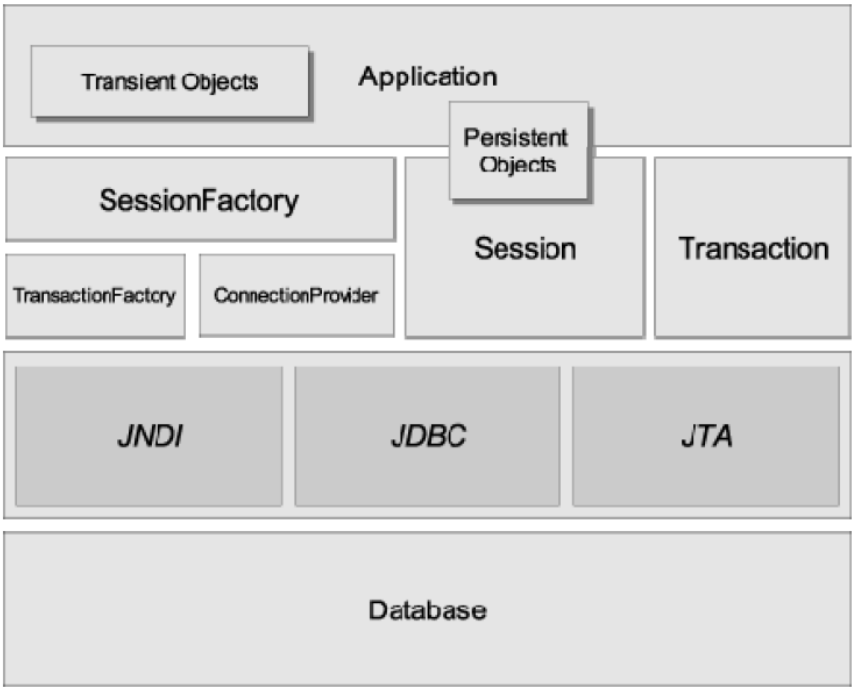


Figura 12 Arquitectura Hibernate

➢ Las Interfaces mostradas se clasifican de la siguiente forma:

- Interfaces llamadas por la aplicación para realizar operaciones básicas.

- **SESSION:** interfaz primaria utilizada en cualquier aplicación Hibernate (SessionFactory).

- **TRANSACTION**: Esta interfaz se utiliza para el manejo de transacciones, las cuales se listaran más adelante.

- **QUERY:** permite realizar peticiones a la base de datos y controla cómo se ejecuta dicha petición (query). Las peticiones se escriben en HQL o en SQL nativo de la base de datos que estamos utilizando. Una instancia. Query se utiliza para enlazar los parámetros de la petición, limitar el número de resultados devueltos por la petición y para ejecutar dicha petición.

- **INTERFACES** llamadas por el código de la infraestructura de la aplicación para configurar Hibernate. La más importante es la clase “Configuration”**,** esta clase se utiliza para configurar y "arrancar" Hibernate. La aplicación utiliza una instancia de configuración para especificar la ubicación de los documentos que indican el mapeado de los objetos y propiedades específicas de Hibernate, y a continuación crea la Session Factory.

- **INTERFACES CALLBACK** que permiten a la aplicación reaccionar ante determinados eventos que ocurren dentro de la aplicación, tales como Interceptor, Lifecycle y Validatable

- **INTERFACES** que permiten extender las funcionalidades de mapeado de Hibernate, como por ejemplo UserType, CompositeUserType e IdentifierGenerator

Además, Hibernate hace uso de API’s de Java, tales como JDBC, JNDI(Java Naming Directory Interface) y JTA (Java Transaction Api).

4.5.2.4. CARACTERÍSTICAS CLAVE

A continuación se enlistaran una serie de características claves oportunas de Hibernate.

- **MODELO PROGRAMACIÓN ORIENTADA A OBJETOS**: Hibernate tiene soporte para todas las características del paradigma orientado a objetos, pudiendo utilizar sin ningún problema, características como: herencia, polimorfismo, composición y el framework de colecciones de Java.

- **MEJORAMIENTO DEL CÓDIGO COMPILADO (BYTECODE)**: No se necesita realizar un procesamiento del bytecode ni generar código en tiempo de compilación

- **SOPORTE PARA MÚLTIPLES MODELOS DE OBJETOS**: Permite múltiples tipos de mapeos entre objetos dependientes y colecciones.

- **SOPORTE PARA TRANSACCIONES**: Hibernate tiene soporte para transacciones largas y gestiona la política optimistic locking de forma automática

- **MANEJO AUTOMÁTICO DE LLAVES PRIMARIAS**: Hibernate permite asignaciones de identificadores automáticos por la aplicación, así como también tiene soporte de los distintos tipos de generación de llaves que proporciona los manejadores de bases de datos, como ejemplo: secuencias, columnas autoincreméntales, entre otras.

- **LENGUAJE DE CONSULTAS HQL:** Hibérnate proporciona su propio lenguaje de consultas, el uso de este lenguaje crea independencia del gestor de base de datos que se esté utilizando. Este lenguaje puede ser utilizado tanto para el almacenamiento de objetos como para la extracción de los mismos.

- **PERSISTENCIA TRANSPARENTE:** Una gran ventaja Hibérnate es que la persistencia se realiza de forma transparente para el desarrollador, el desarrollador nunca tiene que construir una sentencia SQL.

- **ESCALABILIDAD:** Hibernate posee una gran escalabilidad, posee una caché de dos niveles que puede ser usado en un clúster, además de esto también permite inicialización perezosa de objetos y colecciones

4.5.2.5. SIMPLICIDAD Y FLEXIBILIDAD

En lugar de un conjunto de clases y configuraciones requeridas por algunas soluciones de persistencia tal como EJB, Hibérnate requiere un único archivo de configuración en tiempo de ejecución y un documento de especificación para cada objeto persistente. La configuración en tiempo de ejecución puede ser seteada por valor o por medio de un archivo XML, alternativamente se puede configurar parte de Hibérnate programáticamente en tiempo de ejecución.

El formato XML de los documentos de mapeo puede ser muy corto, dejando al propio framework determinar las demás características. Adicionalmente, puede proveer al framework con más información para especificar propiedades adicionales.

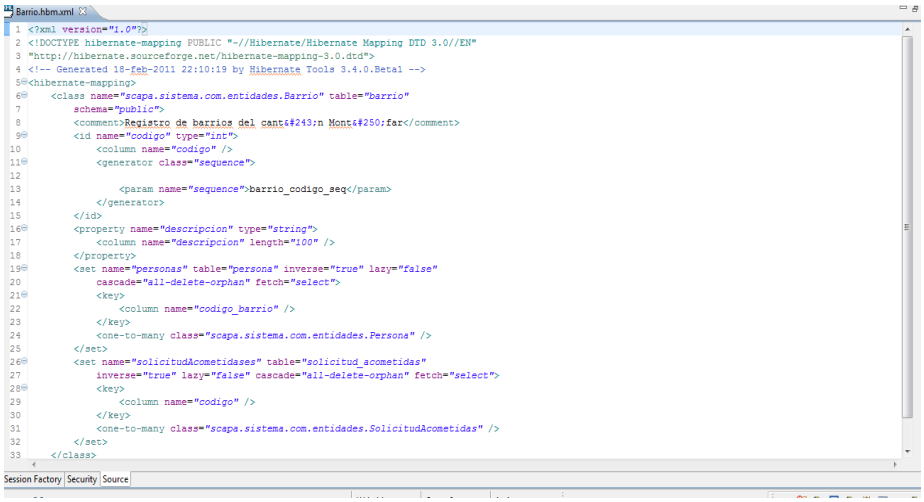


Figura 13 Ejemplo del archivo de XML

Al utilizar algunos framework de persistencia como EJB, la aplicación se torna dependiente del framework. Hibérnate no crea esta dependencia adicional. Los objetos persistentes en la aplicación no contienen información extra de Hibernate en su semántica. Simplemente se crean objetos de java asociados a los documentos de especificación.

A diferencia de EJB's, Hibernate no requiere de un especial contenedor para funcionar, puede ser utilizado en cualquier entorno desde un aplicación standalone hasta un servidor de aplicaciones empresarial

4.5.2.6. RENDIMIENTO

Una idea errónea acerca de los ORM, es que existe un gran impacto en el rendimiento de las aplicaciones, este no es el caso de Hibérnate. La forma clave de medir el rendimiento de un ORM, es si utiliza el número mínimo de acceso a la base datos. Esto es válido para inserción, modificación y extracción de datos. Muchos framework basados en jdbc realizan la modificación en la base de datos de los objetos aun si estos no han cambiado, Hibernate asegura una actualización si solo si el estado del objeto ha cambiado

La carga perezosa provee otra forma de mejorar el rendimiento, puesto que para un objeto ya cargado en memoria, se llenan las colecciones (hijos) hasta que estén a punto de accederse. Esto asegura que se tendrán cargados solo los objetos necesarios lo que tiene un gran impacto en el rendimiento.

Otra forma de mejorar el rendimiento es la capacidad de deshabilitar selectivamente objetos asociados que se recuperan cuando el objeto principal es obtenido. Esto se logra mediante el establecimiento de las regiones a la cual está asociada la propiedad de asociación. Por ejemplo, imaginemos que se tiene una clase Usuario con una asociación one-to-one con una clase Departamento. Cuando se recupera la clase usuario no siempre se querrá recuperar el Departamento asociado, por lo que se puede declarar que el objeto Departamento sea recuperado solo cuando este sea necesario mediante la configuración outer-join false

También es posible declarar un proxy de objetos, dando como resultado objetos poblados solo cuando el desarrollador los solicita. Cuando se crea un proxy, Hibernate crea una representación no poblada de clases persistentes. El proxy se reemplazará por la instancia actual de la clase persistente llamando a un método del objeto de acceso

El proxy está relacionado con la configuración outer-join que se acaba de mencionar. Utilizando el mismo ejemplo: si la clase Departamento es declarada para tener un proxy, no se necesita establecer outer-join false sobre la clase
Usuario para el departamento (El servicio de Hibernate solo poblará los Departamentos cuando sean necesarios).

La caché de los objetos también juega un rol importante en el mejoramiento del rendimiento de la aplicación. Hibernate soporta varios productos open source y comerciales para manejo de la caché. La caché puede ser habilitada para clases persistentes o para colecciones de objetos persistentes. Los resultados de las consultas también pueden utilizar caché pero estos solo se benefician las consultas con los mismos parámetros. La caché de consultas no garantiza un aumento en el rendimiento en las aplicaciones pero está disponible para ser utilizada en casos especiales

4.5.2.7. INTERFACES BÁSICAS

- **SESSION**: Interfaz primaria, se crean y destruyen muchas instancias en la ejecución, pero sus instancias son objetos ligeros. Una sesión está entre una conexión y una transacción. También hace de cache de objetos cargados

- **SESSIONFACTORY**: Devuelve instancias de sesiones, se necesita una instancia diferente por cada tipo de BD accedida.

- **CONFIGURATION**: para indicar la localización de los ficheros de mapeo.

- **TRANSACTION**: (opcional) Abstracción de una transacción concreta (JDBC, JTA, CORBA).

- **QUERY**: para realizar consultas, y controlar su ejecución, en HQL o SQL.

- **CRITERIA**: Para crear y ejecutar OO consultas (parecido a Query).

    - Interfaces CallBack: para recibir notificaciones de eventos sobre objetos (cargado, guardado, borrado). Ej.: Lifecycle, Validatbale, Interceptor.

    - Type: mapea el tipo de un objeto a el tipo de una tabla(s). Se pueden añadir tipos de usuario con las interfaces UserType y CompositeUserType.

4.5.2.8. CONFIGURACIÓN BÁSICA DEL HIBERNATE

Para utilizar Hibérnate en una aplicación, es necesario conocer cómo configurarlo. Hibérnate puede configurarse y ejecutarse en la mayoría de aplicaciones Java y entornos de desarrollo. Generalmente, Hibernate se utiliza en aplicaciones cliente/servidor de dos y tres capas, desplegándose únicamente en el servidor. Las aplicaciones cliente normalmente utilizan un navegador web, pero las aplicaciones swing y AWT también son usuales. Aunque solamente vamos a ver cómo configurar Hibernate en un entorno no gestionado, es importante comprender la diferencia entre la configuración de Hibernate para entornos gestionados y no gestionados

- Entorno gestionado: los pools de recursos tales como conexiones a la base de datos permiten establecer los límites de las transacciones y la seguridad se debe especificar de forma declarativa, es decir, en sus metadatos. Un servidor de aplicaciones J2EE, tal como JBoss, Bea WebLogic o IBM WebSphere implementan un entorno gestionado para Java.

- Entorno no gestionado: proporciona una gestión básica de la concurrencia a través de un pooling de threads. Un contenedor de servlets, como Tomcat proporciona un entorno de servidor no gestionado para aplicaciones web Java. Una aplicación stand-alone también se considera como no gestionada. Los entornos no gestionados no proporcionan infraestructura para transacciones automáticas, gestión de recursos, o seguridad. La propia aplicación es la que gestiona las conexiones con la base de datos y establece los límites de las transacciones

  Tanto en un entorno gestionado como en uno no gestionado, lo primero que debemos hacer es iniciar Hibernate. Para hacer esto debemos crear una SessionFactory desde la clase Configuration.

4.5.2.9. ESPECIFICACIÓN DE OPCIONES DE CONFIGURACIÓN

Una instancia de org.hibernate.cfg.Configuration representa un conjunto completo de correspondencias entre los tipos Java de una aplicación y los tipos de una base de datos SQL, además de contener un conjunto de propiedades de configuración. Una lista de las posibles propiedades de configuración y su explicación la podemos consultar en el manual de referencia de Hibernate incluido en la distribución (directorio doc\reference\en\pdf). Para especificar las opciones de configuración, se pueden utilizar cualquiera de las siguientes formas:

- Pasar una instancia de java.util. Properties a Configuration.setProperties().
- Establecer las propiedades del sistema mediante java-Dproperty=value.
- Situar un fichero denominado hibernate.properties en el classpath.
- Incluir elementos <property> en el fichero hibernate.cfg.xml en el classpath.

Las dos primeras opciones no se suelen utilizar, excepto para pruebas rápidas y prototipos. La mayoría de las aplicaciones requieren un fichero de configuración estático. Las dos últimas opciones sirven para lo mismo configurar Hibernate, elegir entre una u otra depende simplemente de nuestras preferencias sintácticas.

4.5.2.10. CREACIÓN DE UNA SESSIONFACTORY

Para crear una SessionFactory, primero debemos crear una única instancia de configuración durante la inicialización de la aplicación y utilizarla para determinar la ubicación de los ficheros mapeados. Una vez configurada la instancia de configuración se utiliza para crear la SessionFactory. Una vez creada la SessionFactory.

```
package scapa.sistema.com.hibernate;

import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory=construirSessionFactory();
    private static SessionFactory construirSessionFactory(){
        try{
            //Crea un nuevo SessionFactory a partir de hibernate.cfg.xml
            return new Configuration().configure().buildSessionFactory();
        }catch(Throwable ex){
            //Si existe error al inicializar
            System.out.println("Error creando SessionFactory: "+ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory(){
        return sessionFactory;
    }
}
```

Figura 14 Clase Hibérnate Útil

4.5.2.11. CONFIGURAR LA CONEXIÓN A LA BASE DE DATOS.

En un entorno no gestionado, como por ejemplo un contenedor de servlets, la aplicación es la responsable de obtener las conexiones JDBC.

Hibernate es parte de la aplicación, por lo que es responsable de obtener dichas conexiones. Generalmente, no es conveniente crear una conexión cada vez que se quiere interactuar con la base de datos. En vez de eso, las aplicaciones Java deberían usar un pool de conexiones. Hay tres razones por las que usar un pool:

- Conseguir una nueva conexión resulta costoso.
- Mantener muchas conexiones ociosas tiene un alto costo.
- Crear la preparación de sentencias es también caro para algunos drivers

La siguiente figura muestra el papel de un pool de conexiones JDBC en un entorno de ejecución de una aplicación web (sin utilizar Hibernate). Ya que este entorno es no gestionado, no implementa el pooling de conexiones, por lo que la aplicación debe implementar su propio algoritmo de pooling o

utilizar alguna librería como por ejemplo el pool de conexiones de libre distribución C3P0. Sin Hibernate, el código de la aplicación normalmente llama al pool de conexiones para obtener las conexiones JDBC y ejecutar sentencias SQL

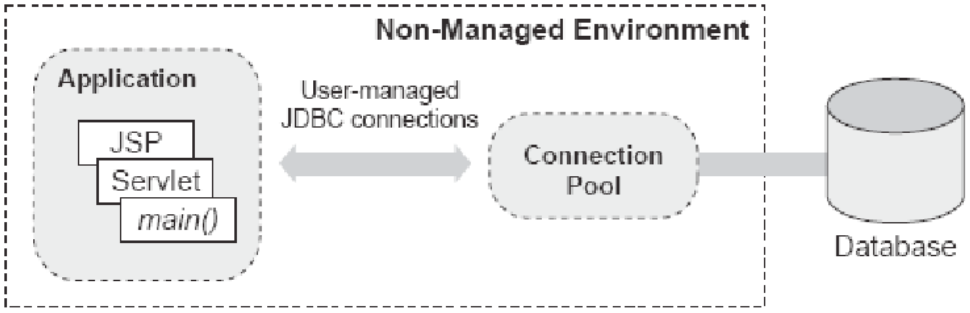


Figura 15 Pool de conexiones JDBC en un entorno no gestionado

Con Hibernate, este escenario cambia: Hibernate actúa como un cliente del pool de conexiones JDBC, tal y como se muestra en la siguiente Figura. El código de la aplicación utiliza los API's Session y Query para las operaciones de persistencia y solamente tiene que gestionar las transacciones a la base de datos, idealmente, utilizando el API Hibernate Transaction

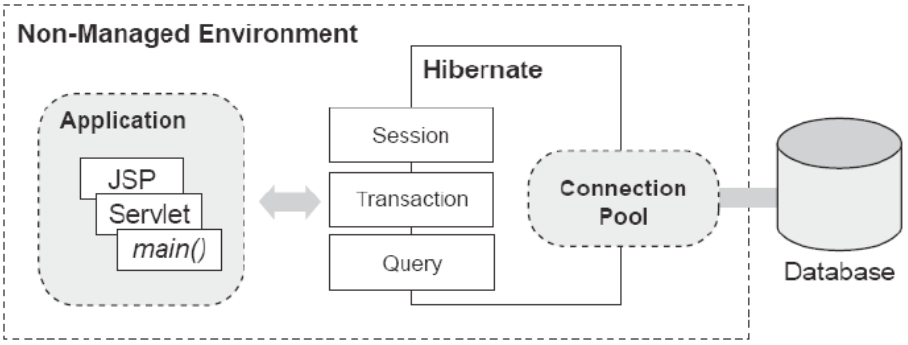


Figura 16 Hibérnate con un pool de conexiones en un entorno no gestionado

Hibérnate define una arquitectura de plugins que permite la integración con cualquier pool de conexiones. Puesto que Hibérnate ya incluye soporte para C3P0, vamos a ver cómo usarlo.

Hibernate actualizará la configuración del pool por nosotros con las propiedades que determinemos. Un ejemplo de un fichero hibernate.cfg.xml utilizando C3P0 se muestra en el siguiente listado:

```
<property name="hibernate.connection.url">jdbc:postgresql://localhost/SCAPA</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="show_sql">true</property>
<!-- Bind the getCurrentSession() method to the thread. -->
<property name="current_session_context_class">thread</property>
<property name="connection.provider_class">org.hibernate.connection.C3P0ConnectionProvider</pr
<property name="c3p0.acquire_increment">1</property>
<!-- seconds -->
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_periods">3000</property>
<!-- seconds
Disable the second-level cache -->
<property name="hibernate.show_sql">true</property>
<mapping resource="scapa/sistema/com/entidades/EstadoUsuario.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/RolUsuarios.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Usuarios.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Persona.hbm.xml"/>
```

Figura 17 Fichero hibernate.cfg.xml utilizando C3P0 Hibernate

En la figura se muestran las siguientes líneas de código:

- El nombre de la clase Java que implementa el Driver JDBC (el fichero JAR del driver debe estar en el classpath de la aplicación).
- La URL JDBC que especifica el host y nombre de la base de datos para las conexiones JDBC.
- El usuario de la base.
- La contraseña de la base de datos para los usuarios especifico.
- Un Dialect para la base de datos. A pesar de esfuerzo de estandarización de ANSI, SQL se implementa de forma diferente por diferentes vendedores, por lo que necesitamos especificar un Dialect. Hibernate soporta el SQL de las bases de datos más populares
- El número mínimo de conexiones JDBC que C3P0 mantiene preparadas.
- El número máximo de conexiones en el pool. Se lanzará una excepción si este número se sobrepasa en tiempo de ejecución.
- El período de tiempo (en este caso, 5 minutos o 300 segundos) después del cual una conexión no usada se eliminará del pool.
- El número máximo de sentencias preparadas que serán almacenadas en una memoria intermedia (caché). Esto es esencial para un mejor rendimiento de Hibernate.

- El tiempo en segundos que una conexión debe estar sin utilizar para que se valide de forma automática dicha conexión.

El especificar las propiedades de la forma hibernate.c3p0.* selecciona C3P0 como el pool de conexiones para Hibernate (sin necesidad de ninguna otra acción). Otros pools de conexiones soportados son Apache DBCP y Proxool.

4.5.2.12. USO DE CONFIGURACIÓN BASADA EN XML

Podemos utilizar un fichero de configuración XML para configurar completamente una SessionFactory. A diferencia del fichero hibernate.properties, que contiene solamente parámetros de configuración, el fichero hibernate.cfg.xml puede especificar también la ubicación de los documentos de mapeado. Muchos usuarios prefieren centralizar la configuración de Hibernate de esta forma, en vez de añadir parámetros a Configuration en el código de la aplicación

```
<property name="hibernate.connection.password">edwin</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost/SCAPA</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="show_sql">true</property>
<!-- Bind the getCurrentSession() method to the thread. -->
<property name="current_session_context_class">thread</property>
<property name="connection.provider_class">org.hibernate.connection.C3P0ConnectionProvider</property>
<property name="c3p0.acquire_increment">1</property>
<!-- seconds -->
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_periods">3000</property>
<!-- seconds
Disable the second-level cache -->
<property name="hibernate.show_sql">true</property>
<mapping resource="scapa/sistema/com/entidades/EstadoUsuario.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/RolUsuarios.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Usuarios.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Persona.hbm.xml"/>
<mapping resource="                com/entidades/Contrato.hbm.xml"/>
<mapping resource="                com/entidades/ClausulasContrato.hbm.xml"/>
<mapping resource="                com/entidades/FichaMunicipal.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Sucursal.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/EstadoCuenta.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/UsoConexion.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/CuentaPersona.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/ParametrosCuenta.hbm.xml"/>
<mapping resource="scapa/sistema/com/entidades/Medidor.hbm.xml"/>
```

Attribute : resource
Data Type : CDATA
Press 'F2' for focus

Figura 18 Configuración Xml Hibernate

La declaración <!DOCTYOPE se usa por el analizador de XML para validar este documento frente a la DTD de configuración de Hibernate. El atributo "name" opcional es equivalente a la propiedad hibernate.session_factory_name y se usa para el enlazado JNDI de la SessionFactory.

Las propiedades de Hibernate pueden especificarse sin el prefijo Hibernate. Los nombres de las propiedades y valores son, por otro lado, idénticos a las propiedades de configuración especificadas mediante programación.

Los documentos de mapeado pueden especificarse como recursos de la aplicación. Ahora podemos inicializar Hibernate utilizando SessionFactory sessions = new Configuration().configure().buildSessionFactory();.

Cuando se llama a configure(), Hibernate busca un fichero denominado hibernate.cfg.xml en el classpath. Si se quiere utilizar un nombre de fichero diferente o hacer que Hibernate busque en un subdirectorio, debemos pasar una ruta al método “configure(): SessionFactory ses = new Configuration().configure("/hibernate-config/configHibernate.cfg.xml).buildSessionFactory();”.

Utilizar un fichero de configuración XML ciertamente es más cómodo que usar un fichero de propiedades o mediante programación. El hecho de que se puedan tener ficheros de mapeado externos al código fuente de la aplicación es el principal beneficio de esta aproximación. Así, por ejemplo, podemos usar diferentes conjuntos de ficheros de mapeado (y diferentes opciones de configuración), dependiendo de la base de datos que utilicemos y el entorno (Desarrollo o producción), y cambiar entre ellos mediante programación. Si tenemos ambos ficheros, hibernate.properties, e hibernate.cfg.xml, en el classpath, las asignaciones del fichero de configuración XML prevalecen sobre las del fichero de propiedades. Esto puede resultar útil si queremos guardar algunas propiedades base y sobrescribirlas para cada despliegue con un fichero de configuración XML.

4.5.2.13. CONFIGURACIÓN DE LOGGING.

Hibernate (y muchas otras implementaciones de ORMs) ejecuta las sentencias SQL de forma asíncrona. Una sentencia INSERT normalmente no se ejecuta cuando la aplicación llama a Session.save(); una sentencia UPDATE no se ejecuta cuando la aplicación llama a Item.addBid(). En vez de eso, las sentencias SQL se ejecutan al final de una transacción.

Esta característica evidencia el hecho de seguir una traza y depurar el código ORM es a veces no trivial. Una forma de ver qué es lo que está sucediendo internamente en Hibernate es utilizar el mecanismo de logging. Para ello tendremos que asignar a la propiedad hibernate.show_sql el valor true, lo que permite hacer logs en la consola del código SQL generado

Hay veces en las que mostrar el código SQL no es suficiente. Hibernate "muestra" todos los eventos interesantes utilizando el denominado

commonslogging de Apache, una capa de abstracción que redirige la salida al log4j de Apache (si colocamos log4j.jar en le classpath) o al logging de JDK1.4 (si estamos ejecutando bajo JDK1.4 o superior y log4j no está presente). Es recomendable utilizar log4j, ya que está más maduro, es más popular, y está bajo un desarrollo más activo.

Para ver las salidas desde log4j necesitamos un fichero denominado log4j.properties en nuestro classpath. El siguiente ejemplo redirige todos los mensajes log a la consola

```
###Dirigir los mensajes a la consola###
log4j.appender.stdout.Target = System.out
##Enviarlo a consola
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
###Opciones de root###
log4j.rootLogger=warn, stdout
###Opciones de Hibernate Log.###
log4j.logger.net.sf.hibernate=info
###cache, log###
log4j.logger.net.sf.hibernate.ps.PreparedStatementCache=error
###parámetros log JDBC###
log4j.logger.net.sf.hibernate.type=info
```

Figura 19 Re direccionar el log de Hibernate a la salida de la consola

Con esta configuración, no aparecerán muchos mensajes de log en tiempo de ejecución. Si reemplazamos info por debug en la categoría log4.logger.net.sf.hibernate se mostrará el trabajo interno de Hibernate

4.5.2.14. TRABAJOS CON OBJETOS MAPEADOS

Al trabajar con objetos mapeados, Hibernate provee una gran cantidad de funciones y formas de mapeo para manipular los datos resultantes, se pueden establecer relaciones de many to many, one to many, one to one con el atributo inverse le indicamos a Hibernate que la relación es bidireccional.

Con el atributo cascade=”save-update”, le indicamos a hibernate que persista todos los extremos de las relaciones sin necesidad de realizar un sabe explicito, con el atributo “all-delete-orphan” indicamos que el padre es el responsable del ciclo de vida del hijo.

Hibernate soporta modelos de granulo fino, por lo que se pueden mapear varias clases en una sola tabla, también cuenta con persistencia transitiva lo que significa que todos los objetos desde una instancia persistente, se vuelven persistentes.

4.5.2.15. CREAR UN OBJETO

Si se desea insertar un objeto a la base de datos, basta con que la clase este mapeada e incluida en el archivo de configuración de Hibernate, luego con la interfaz session, se procede a ejecutar la función saveOrUpdate.

```
public void insertarBarrio(Barrio b) throws ExcepcionesSCAPA {
    sesion = HibernateUtil.getSessionFactory().openSession();
    try {

        sesion.beginTransaction();
        sesion.clear();
        sesion.save(b);
        sesion.getTransaction().commit();

        sesion.close();
    } catch (HibernateException e1) {
        sesion.getTransaction().rollback();
        sesion.close();
        throw new ExcepcionesSCAPA("No ha guardados el barrio : "
                + e1.getMessage());
    }

}
```

Figura 20 Método de Guardar un Objeto

4.5.2.16. MODIFICAR UN OBJETO

Por defecto se escriben todas los campos de la tabla, con la propiedad "dynamic-update = true", solo se modifican los campos que han sido modificados, es decir se genera una sentencia update que contiene solo las columnas que fueron modificadas

```
public void actualizarBarrio(Barrio c) throws ExcepcionesSCAPA {
    sesion = HibernateUtil.getSessionFactory().openSession();
    try {

        sesion.beginTransaction();
        Barrio b;
        sesion.clear();
        b = (Barrio) sesion.get(Barrio.class, c.getCodigo());
        b.setDescripcion(c.getDescripcion());
        sesion.update(b);
        sesion.getTransaction().commit();
        // escribirMensaje(mensage1);
        sesion.close();
    } catch (HibernateException e1) {
        sesion.getTransaction().rollback();
        sesion.close();
        throw new ExcepcionesSCAPA("Búsqueda de barrio no realizada : "
                + e1.getMessage());
    }
}
```

Figura 21 Modificar un objeto con Hibernate

4.5.2.17. OBTENER UNA LISTA

En Hibérnate existen varias formas de obtener los objetos de la base de datos.

- Navegar en el grafo de objetos persistidos, empezando por un objeto ya cargado
- Cargándolo por medio de su identificador ya sea por medio de get (Class, Id), o por medio de load (Class, Id), load a diferencia de get envía una excepción si no se encontrase el objeto, mientras que get retorna null.
- Utilizar HQL, Utilizar la API Criteria que permite hacer consultas, permite especificar restricciones dinámicamente sin trabajar directamente con cadenas en tiempo de compilación

```
public List<Barrio> findAll() throws ExcepcionesSCAPA {
    sesion = HibernateUtil.getSessionFactory().openSession();
    try {

        sesion.beginTransaction();
        Query q;
        sesion.clear();

        q = sesion.createQuery("From Barrio order by codigo");
        @SuppressWarnings("unchecked")
        List<Barrio> listadoComentarios = q.list();
        sesion.getTransaction().commit();
        sesion.close();

        return listadoComentarios;
    } catch (HibernateException e1) {
        sesion.getTransaction().rollback();
        sesion.close();
        throw new ExcepcionesSCAPA("Búsqueda de barrio no realizada: "
                + e1.getMessage());
    }

}
```

Figura 22 Búsqueda con HQL

4.5.2.18. ELIMINAR UN OBJETO

Para eliminar un objeto de la base de datos con Hibernate, simplemente es necesario ejecutar la función delete desde la interfaz sesión, esto produce que el objeto se vuelva transitivo

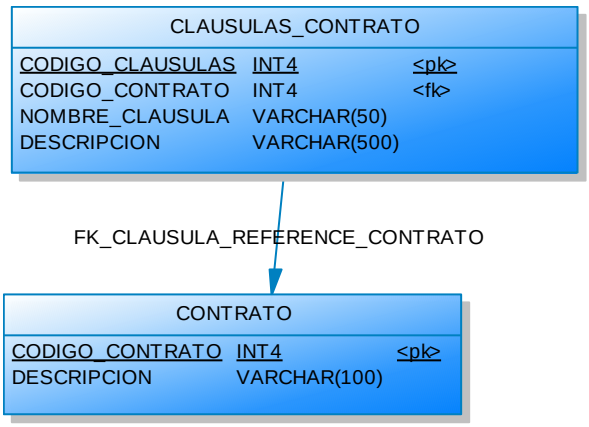
```
public void eliminarBarrio(int codigo) throws ExcepcionesSCAPA {
    sesion = HibernateUtil.getSessionFactory().openSession();
    try {

        sesion.beginTransaction();
        Barrio c;
        sesion.clear();
        c = (Barrio) sesion.get(Barrio.class, codigo);
        sesion.delete(c);
        sesion.getTransaction().commit();
        // escribirMensaje(mensage1);// mensaje de Confirmacion
        sesion.close();
    } catch (HibernateException e1) {
        sesion.getTransaction().rollback();
        sesion.close();
        throw new ExcepcionesSCAPA("No ha eliminado el barrio : "
                + e1.getMessage());
    }

}
```

Figura 23 Borrar un objeto con Hibernate

4.5.2.19. COLECCIONES

Cuando se tienen relaciones de uno a muchos entre dos clases que son persistentes, como ejemplo la tabla contrato y clausulas contrato, dentro de la clase contrato, se debe colocar una colección (Array) de Objetos de la clase clausulas contrato.

Podemos visualizar esta relación en la base de datos por medio de dos tablas: contrato y clausulas contrato, donde en clausulas contrato se encuentra un campo que es la llave extranjera de la tabla contrato.




```
<class name="scapa.sistema.com.entidades.Contrato" table="contrato"
    schema="public">
    <comment>CONTRATO</comment>
    <id name="codigoContrato" type="int">
        <column name="codigo_contrato" />
        <generator class="sequence">
            <param name="sequence">secuencia_ingreso_contrato</param>
        </generator>
    </id>
    <property name="descripcion" type="string">
        <column name="descripcion" length="100" />
    </property>
    <set fetch="select" inverse="true" lazy="false" name="clausulasContratos"
        table="clausulas_contrato" cascade="all-delete-orphan" order-by="codigo_clausulas">
        <key foreign-key="clausulasContratos.fk_clausula_reference_contrato">
            <column name="codigo_contrato" />
        </key>
        <one-to-many class="scapa.sistema.com.entidades.ClausulasContrato" />
    </set>

    <set name="cuentaPersonas" table="cuenta_persona" inverse="true"
        lazy="false" fetch="select" cascade="all-delete-orphan">
        <key>
            <column name="codigo_contrato" />
        </key>
        <one-to-many class="scapa.sistema.com.entidades.CuentaPersona" />
    </set>
</class>
```

Figura 24 Mapeo de las tablas

De manera predeterminada, cuando intentemos recuperar los objetos de la clase classA, Hibernate 3 realizará una consulta al repositorio de datos para

extraer la información correspondiente a la tabla C, pero Hibernate no extraerá la información inmediatamente, solamente proporcionará un proxy

Este proxy es el encargado de realizar las consultas al repositorio de datos cuando verdaderamente se intente acceder a la colección de Objetos de la D.

Según esta forma de acceso, para extraer la información de la tabla C y sus objetos relacionados de tabla D es necesario realizar n+1 (donde n es el número de registros de classC) accesos al repositorio de datos

En la mayoría de casos esta forma de acceso es adecuada, puesto que no se recuperan absolutamente todos los registros de la tabla “tabla D” hasta que sean realmente necesarios. Esta forma de acceso evita traer todo el repositorio de datos a memoria cuando se tienen muchas tablas relacionadas.

Este mecanismo puede ser inadecuado, ya que este realiza múltiples accesos pues cuando se pudiera realizar la extracción de datos con una sola consulta, pero mejoramos el rendimiento de nuestra aplicación al momento de realizar consultas por separado.

4.5.2.20. FECTH & LAZY

Para mejorar el rendimiento de las aplicaciones, debemos comprender como y cuando Hibérnate obtiene la información del repositorio de datos.

4.5.2.21. LAZY LOADING

Al momento de realizar una relación en Hibernate utilizamos el atributo “lazy” para definir cuándo se obtendrá esta información. Por defecto la propiedad “lazy” tiene el valor “true”, esto quiere decir que la colección de Objetos se recupera cuando se realice una acción sobre este.

4.5.2.22. FETCH

Por otra parte, el atributo “fetch” define como obtendrá Hibernate la información del repositorio de datos. Por defecto esta propiedad tiene el valor “select”. Cuando se tiene el valor “select” en esta propiedad, Hibernate realiza un acceso a la base de datos para obtener cada objeto class C. Al

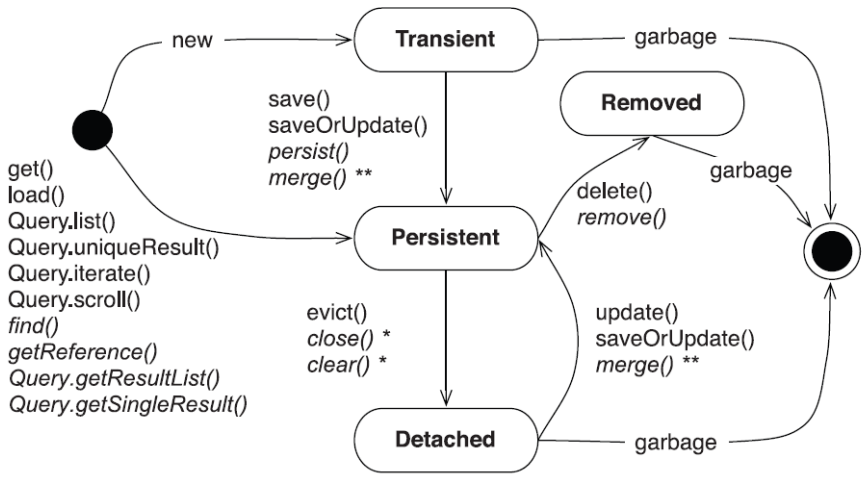
establecer fetch con valor “join”, Hibernate obtendrá todos los datos con una sola consulta, simplemente realiza un left outer join con las tablas relacionadas en una misma sentencia SQL.

4.5.2.23. ESTADO DE LOS OBJETOS DE HIBERNATE

Las diferentes soluciones ORM usan diferentes terminologías y definen diferentes estados y estados de transición para el ciclo de vida de los objetos.

Por otra parte, los estados de los objetos utilizados pueden ser internamente diferentes a lo presentado en la aplicación del cliente. Hibernate define solo 4 estados, ocultando la complejidad de la implementación interna del código del cliente.

Los estados de los objetos definidos por Hibernate se presentan en la figura siguiente, se puede notar que cada método invocado por el administrador de persistencia es el disparador para la transición de estado. El Api en Hibernate es el objeto Session.



Figuran 25 Estados de objetos en Hibernate y sus disparadores

4.5.2.24. CACHE

La arquitectura de caché de Hibernate es realmente potente. Su caché de dos niveles ofrece una gran flexibilidad al tiempo que un gran rendimiento tanto en sistemas standalone como en cluster.

Al comprender completamente el funcionamiento de las caches de Hibernate, se podrá obtener el máximo de posibilidades, teniendo un aumento significativo en la escalabilidad y el rendimiento de nuestras aplicaciones, evitando así problemas de sincronización y concurrencia.

- **LA ARQUITECTURA DE LA CACHE DE HIBERNATE**

  La arquitectura de caché de Hibernate se compone de dos niveles, cada uno con tareas muy diferentes. El primer nivel se refiere a la caché de sesión, esta viene activada por defecto y no se puede desactivar.

  La caché de segundo nivel en Hibernate se maneja de forma externa por medio de proveedores de caché, el alcance que podría tener es de proceso o de clúster. El uso de la caché de segundo nivel es opcional y puede ser asignada clase por clase o realizando una asociación base.

  En la siguiente figura, se puede observar cómo está constituida la arquitectura de caché de dos niveles de Hibernate, tal como se describe anteriormente la caché de primer nivel se refiere a la sesión misma e interactúa con la caché de segundo nivel.

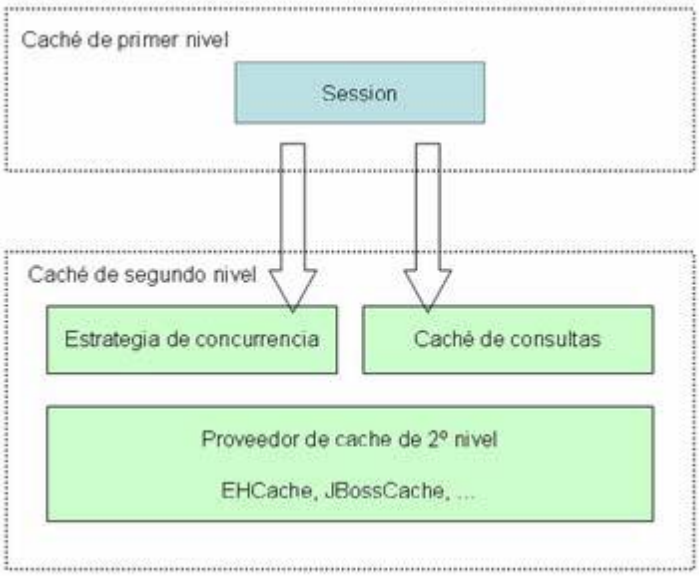


Figura 26 Arquitectura de caché de Hibernate

- **CACHES Y CONCURRENCIAS**

Cualquier implementación ORM que permita múltiples unidades de trabajo para compartir la misma instancia de persistencia, debe proveer algún tipo de nivel de bloqueo para asegurar la sincronización de accesos concurrentes. Usualmente esta es implementada utilizando candados de lectura y escritura junto con la detección de deadlock. Implementaciones como Hibernate que mantienen un conjunto de instancias por unidad de trabajo, evitan estos tipos de problema en gran medida.

Un mecanismo de caché, se utiliza para almacenar en memoria objetos que pueden ser utilizados posteriormente, aumentando con esto el rendimiento de las aplicaciones y reduciendo los accesos a las bases de datos. Cuando se utiliza un mecanismo de caché, la forma de interactuar sería la siguiente

- Se realiza la consulta para obtener los objetos.

- Se realiza una búsqueda dentro de la caché de primer nivel, para verificar si los objetos solicitados se encuentran allí, si esto es así entonces se devuelven de forma directa. Si dichos objetos no estuvieran en la caché, se procede a realizar la recuperación desde algún recurso externo y se guardan en la caché, por si se llegaran a necesitar en un futuro, nos ahorraríamos este paso, que sin lugar a duda es más costoso.

- **LA CACHE DE PRIMER NIVEL**

La arquitectura de caché de Hibernate se compone de dos niveles, cada uno con tareas my diferentes. El primer nivel se refiere a la caché de sesión, esta viene activada por defecto y no se puede desactivar

- **LA CACHÉ DE PRIMER NIVEL**

La caché de session o caché de primer nivel, asegura que cuando la aplicación requiera el mismo objeto persistente en dos ocasiones en una particular sesión, este devuelve la misma instancia. Esto ayuda a

no crear tráfico innecesario en la base de datos, pero lo más importante de esto es lo siguiente:

- La capa de persistencia no es vulnerable a desbordamiento de pila en el caso de tener referencia circular en el grafo de objetos.

- No puede existir conflicto al finalizar una transacción, esto se debe a que solo un objeto representa una fila en la base de datos. Todos los cambios en los objetos pueden escribirse de forma segura en la base de datos.

- Los cambios realizados en una particular unidad de trabajo están disponibles inmediatamente para todo lo que se ejecute dentro de la misma

No se necesita realizar nada en especial para activar la caché de sesión.
Esta siempre está activada y por razones evidentes no se puede desactivar

Cada vez que se realice una acción sobre un objeto tales como: save(), update() o saverOrUpdate() y siempre que se recupere un objeto utilizando load(), find(), list(), itérate() o filter(), este objeto es agregado a la caché de primer nivel. El estado del objeto será sincronizado con la base de datos cuando se llama el método flush(). Si no se desea que la sincronización ocurra, o si se está utilizando un número grande de objetos y se necesita liberar memoria, se puede llamar la instrucción evict() para remover los objetos y sus colecciones de la caché de primer nivel, esto puede ser útil en muchas situaciones

Cuando se necesite realizar update o deletes masivos, no es recomendable utilizar la caché de primer nivel, esto puede causar un error de tipo OutOfMemoryException, es mejor realizar estas operaciones mediante procedimientos almacenados que realicen las modificaciones o eliminaciones directamente en la base de datos. Si por el contrario es necesario tener los objetos en memoria, se recomienda ejecutar la sentencia evict inmediatamente para liberar memoria. Para eliminar completamente todos los objetos de la caché de sesión, se debe llamar la instrucción Session.clear().

Cuando se requiere una sesión, Hibernate necesita obtener una conexión al repositorio de datos del pool de conexiones de Hibernate. Estas conexiones se liberan solamente cuando se cierra la sesión utilizando la instrucción close ().

Es evidente que si se consumen muchas conexiones del pool, este podría quedarse sin conexiones. Hibernate proporciona la instrucción disconnect() que permite desconectar por momentos las conexiones, evitando que el pool se quede sin conexiones para servir

- **LA CACHÉ DE SEGUNDO NIVEL**

La caché de segundo nivel de Hibernate, tiene alcance de proceso o clúster; todas las sesiones comparten la misma caché de segundo nivel. La caché de segundo nivel tiene el alcance de SessionFactory.

Las instancias persistentes son guardadas en la caché de segundo nivel de forma separada. La separación es un proceso a nivel de bits como la serialización (este algoritmo es mucho más eficiente que la serialización de Java).

La implementación interna del alcance proceso/cluster no es muy importante, es más importante el uso de las políticas, estrategias y proveedores físicos de caché.
Dependiendo el tipo de información que se maneje, requerirá un tipo especial de políticas de caché. En las aplicaciones, las relaciones de escritura/lectura y el tamaño de las tablas son variantes, además algunas tablas pueden ser compartidas con otras aplicaciones externas. La caché de segundo nivel es configurable para soportar este tipo de granularidad, esto permite desactivar la caché de segundo nivel para clases que son críticas, tal es el caso de las tablas financieras.

La política de caché consiste en realizar las actividades siguientes:

- Activar la caché de segundo nivel.
- Establecer la estrategia de concurrencia de Hibernate
- Configurar la política de expiración de la caché (tiempo de espera, LRU).
- Establecer el formato físico de la caché (archivos de índices, memoria, cluster replicado)

No es conveniente habilitar la caché para todas las clases, la caché solo debe estar disponible para aquellas clases cuyos accesos de lecturas sean considerables. Si se tienen clases en las cuales se realizan muchas actualizaciones, no se recomienda que se habilite la caché de segundo nivel

La caché de segundo nivel puede ser un problema cuando se comparte con aplicaciones que realizan muchos acceso de escritura a las tablas.

La configuración de la caché de hibérnate, se realiza en dos simples pasos.

- Primero se debe decidir cuál estrategia de concurrencia utilizar.
- Después de esto, se necesita configurar el tiempo de expiración y los atributos del formato físico de la caché utilizada por el proveedor de caché.
- Este tipo de caché, ayuda a resolver el problema de las actualizaciones concurrentes, además esta se acoplara a la caché de sesión resolviendo todos los fallos que en esta se produzcan.

- **CACHE EN HIBERNATE**

Hibernate permite habilitar individualmente la caché para cada una de las entidades. De este modo, se puede decidir que clases se beneficiarán del uso de una caché, ya que como se explicaba anteriormente, puede que no todas las clases de nuestro sistema se beneficien. Además de esto, al habilitar la caché es necesario establecer la estrategia de concurrencia que Hibernate utilizará para sincronizar la caché de primer nivel con la caché de segundo nivel, y ésta última con la base de datos. Hay cuatro estrategias de concurrencia predefinidas, a continuación aparecen listadas por orden de restricciones en términos de aislamiento transaccional

- **TRANSACTIONAL**: Es una estrategia que solo puede ser utilizada en clúster con cachés distribuidas, garantiza un nivel de aislamiento hasta repeatable read, si es necesario. Es el nivel más estricto, se recomienda su uso cuando no podamos permitirnos datos que queden desfasados

- **READ-WRITE**: Esta estrategia es recomendable en la misma situación que la transactional, con la única diferencia es que esta no puede ser utilizada con cachés distribuidas, su alcance es hasta el nivel de commit y utiliza un sistema de marcas de tiempo.

- **NONSTRICT-READ-WRITE**: No garantiza la consistencia entre la caché y la base de datos. Si existe la posibilidad de acceso concurrente a una entidad, se debe configurar el tiempo de expiración suficientemente corto. Se debe utilizar esta estrategia cuando los datos raramente cambian (horas, días o semanas) y la información histórica no es de crítica importancia. Hibernate invalida la caché de un objeto que sea modificado y volcado a la base de datos, pero esta operación se realiza de forma asíncrona, sin ningún bloqueo de caché o garantía que la información recuperada por otro usuario sea la última versión.

- **READ-ONLY**: Es la estrategia de concurrencia para datos que nunca cambian. Esta estrategia se utiliza solo para referencia de datos

- **PROVEEDORES DE CACHE.**

Para la caché de segundo nivel de Hibernate, se debe elegir un proveedor pues Hibérnate no incluye ningún proveedor de caché, pero si posee gran soporte para estos. Los proveedores que se pueden utilizar con Hibernate son los siguientes:

- **EHCACHE** es destinado a un proceso simple en una sola JVM. Puede mantener la caché en memoria o en disco, y soporta la caché de consultas de Hibernate.

- **OPENSYMPHONY OSCACHE** es una librería que soporta caché en memoria y en disco en una sola JVM, con un amplio conjunto de políticas de expiración y soporta la caché de consultas.

- **SWARMCACHE** es un clúster de caché basado en JGroups. Puede ser utilizada en un clúster pero no soporta la caché de consultas de Hibernate.

- **JBOSSCACHE** está completamente orientado a un clúster transaccional replicado de caché. También está basado en la librería JGroupsmulticast. Suporta la caché de consulta de Hibernate. Asume que los relojes de sincronización están en el clúster.

Existe una forma de hacer adaptaciones de otros productos simplemente implementando la interfaz net.sf.hibernate.cache.CacheProvider.

No todos los proveedores de caché son compatibles con todas las estrategias de concurrencias, la siguiente matriz ayuda a elegir una apropiada combinación.

Cache Provider	read-only	nonstrict-read-write	read-write	transactional
EHCache	X	X	X	
OSCache	X	X	X	
SwarmCache	X	X		
JBossCache	X			X

Figura 27 Tabla de compatibilidad

La configuración de la caché consiste básicamente en dos pasos:

- En los archivos de mapeo para las clases persistentes, elegir la estrategia de concurrencia que se desea utilizar.

- Habilitar el proveedor de caché en la configuración global de Hibernate y personalizar las configuración especifica del proveedor de caché, por ejemplo si se desea utilizar el proveedor OSCache, se necesita editar el archivo oscache.properties o EHCache el archivo ehcache.xml en el classpath.

Para establecer la configuración del proveedor se necesita agregar la siguiente línea en el archivo global de configuración de Hibernate.

hibernate.cache.provider_class =net.sf.ehcache.hibernate.Provider

4.5.2.25. CACHE DISTRIBUIDAS

En una aplicación empresarial donde existen decenas de miles de usuarios, es necesario que las aplicaciones se ejecuten en un ambiente distribuido, en otras palabras en un clúster. En estas circunstancias, la única solución es utilizar la caché de segundo nivel.

Existen dos formas de implementar una caché de segundo nivel forma distribuida. La primera y más sencilla, consiste en colocar una instancia del

proveedor de caché en cada nodo, esperando que cada instancia en los nodos se sincronicen entre ellos, esta forma es muy simple pues no existen retardos de sincronización.

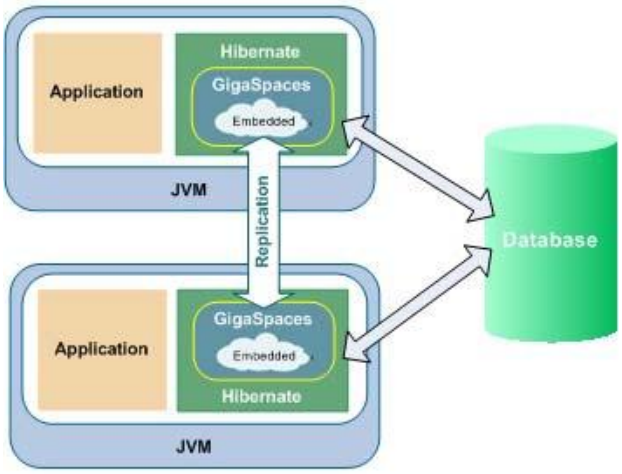


Figura 28 Modelo de caché Hibernate, replicación embebida

La segunda forma consiste en permitir que un proveedor de caché más sofisticado realice el trabajo de sincronización entre las cachés de los diferentes nodos. Uno de los proveedores que se recomienda para Hibernate es JBossCache, ya que es un proveedor totalmente transaccional que se basa en la librería de multicasting, JGroups.

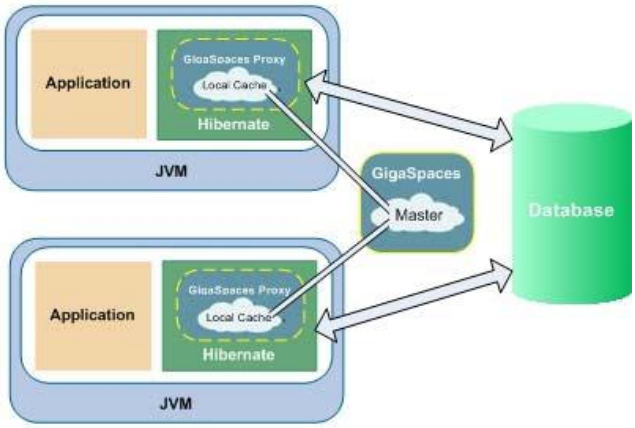


Figura 29 Modelo de caché Hibérnate, topología Máster-Local

De la misma forma en que se configura el proveedor EHCache, se necesita añadir solamente una línea al archivo de configuración de hibérnate para activar el proveedor de caché:

hibernate.cache.provider_class=net.sf.hibernate.cache.TreeCacheProvider

La configuración de JBossCache se realiza mediante la utilización del archivo treecache.xml, este archivo debe establecerse en cada uno de los nodos del clúster. Este archivo es más complejo que el de EHCache pues en este se debe especificar la configuración del clúster, las políticas de sincronización entre los nodos, etc. El siguiente ejemplo sé cómo podría ser la configuración de JBossCache

4.5.2.26. QUIENES UTILIZAN HIBERNATE

Listado de empresas que utilizan Hibernate.
Wilos - http://www.wilos-project.org/
Company name, Location: SoftSlate Commerce, NY, USA
2Fi Business Solutions Ltd. (Hong Kong)
argus Barcelona (Europe)
GPI Argentina, La PLata, Buenos Aires, Argentina
TDC Internet (Warsaw, Poland)
Proyecto Open Source itracker
TerraContact Inc.,
LF Inc. (Tampa, FL)
Ubik-Ingénierie, ubik-ingenierie.com, Roubaix, France
Sony Computer Entertainment Europe, SCEE, Studio Liverpool, Liverpool,
United Kingdom
Elastic Path Software (Vancouver, BC, Canada)
Skillserv, skillserv.com, San Francisco, California, USA
Fedelta POS, fedeltapos.com, Brisbane, Australia
AT&T Labs, Tampa (Florida)
AonCHOR http://www.aonchor.aon.com
Jteam (Amsterdam, The Netherlands)
1Genia (Paris, France)

Tabla 10 Entidades que utilizan Hibernate

4.5.2.27. LICENCIA

Hibérnate es software libre, distribuido bajo los términos de la licencia GNU LGPL

4.5.2.28. PORTE Y CAPACITACIONES

Todo el soporte y consultoría para Hibérnate es proporcionado mediante Jboss, una división de Red Hat.

El soporte profesional ayuda a sobrellevar todos los problemas relacionados con Hibernate, incluyendo bugs y administración de parches, soporte de producción y asistencia en despliegue. Están disponibles tres niveles de soporte, que van desde soporte de 8x5 con respuesta a las 24 horas a 24x7 con respuesta a las 2 horas.
Jboss básicamente ofrece servicios como: Certificaciones, cursos de capacitación y Consultoría.

4.5.2.29. HERRAMIENTAS DE APOYO

Hibérnate cuenta con muchas herramientas de apoyo para generar el xml que lo describe y ahorra mucho trabajo al momento de mapear las tablas, algunas generan el xml y el código java a partir del esquema de base de datos, otros generan el xml a partir de metadata creada en las clases java, etc.,

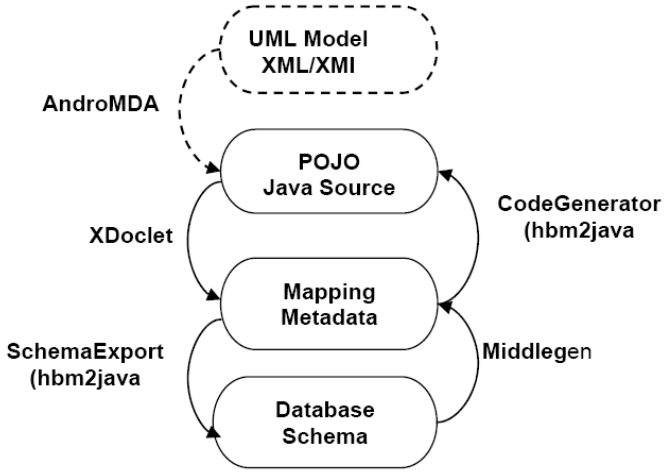


Figura 30 Ciclo de Generación de código y Herramientas

Estas herramientas nos ayudan desde que se inicia el ciclo de vida del software con el modelado. Poseidón puede generar los modelos de entidad relación, luego estos son traducidos por androMDA a objetos planos de Java y luego utilizando XDoclet se crean los descriptores xml, estas tareas pueden ser automatizadas a través de tareas ANT

Una opción alternativa consiste en crear los scripts para la base de datos mediante una herramienta CASE, luego generar los descriptores xml mediante Middlegen y por último generar los objetos planos de Java con hbm2java

A continuación se listan algunas de las herramientas que se pueden utilizar de apoyo para Hibernate

- **HIBERNATE TOOLS**: Ahorra mucho trabajo puesto que al utilizarlo con Ant o con Eclipse puede generar los archivos xml y pojos, correspondiente a las tablas. Referencia: http://www.hibernate.org/255.html.

- **HIBERNATOR**: Herramienta que se Utiliza para generar los descriptores xml y los pojos a partir de un esquema de base de datos. Referencia: http://hibernator.sourceforge.net/

- **XDCLET**: Es una herramienta para la generación de código o XML a partir de "doclets", marcas que se incluyen en los comentarios de un programa. Referencia: http://xdoclet.sourceforge.net/olddocs/

- **ANDROMDA:** Se pronuncia "Andromeda" es un programa informático de tipo framework, de generación extensible de código que se adhiere al paradigma de la arquitectura dirigida por modelos. Referencia: http://www.javahispano.org/canyamo.action

- **MIDDLEGEN:** Es un producto Open source de generación de código, este genera los archivos de configuración utilizando herramientas como JDBC, Velocity, Ant y XDoclet.

4.5.3. EJB 3.0

Fuente: Esta página fue modificada por última vez el 19 ago 2011, a las 02:43.; Wikipedia® es una marca registrada de la Fundación Wikimedia, Inc

Enterprise Java Beans (EJB) es una plataforma para construir aplicaciones de negocio portables, reusables y escalables usando el lenguaje de programación Java. Desde el punto de vista del desarrollador, un EJB es una porción de código que se ejecuta en un contenedor EJB, que no es más que un ambiente especializado (runtime) que provee determinados componentes

de servicio.

Los EJBs pueden ser vistos como componentes, desde el punto de vista que encapsulan comportamiento y permite reutilizar porciones de código, pero también pueden ser vistos como un framework, ya que, desplegados en un contenedor, proveen servicios para el desarrollo de aplicaciones enterprise, servicios que son invisibles para el programador y no ensucian la lógica de negocio con funcionalidades trasversales al modelo de dominio (a menudo requerimientos no funcionales o aspectos). En la especificación 3.0, los EJB no son más que POJOs (clases planas comunes y corrientes de Java) con algunos poderes especiales implícitos, que se activan en runtime cuando son ejecutados en un contenedor de EJBs.

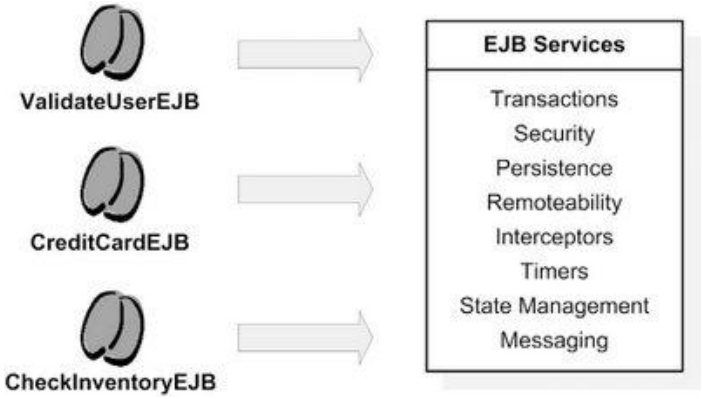


Figura 31 Figura de EJBs

Los servicios que debe proveer el contenedor de EJBs deben ser especificados por el programador a través de metadata de configuración que puede escribirse como:

- Anotaciones de Java5 intercaladas en el código de las clases.
- Descriptores XML (archivos XML separados).
  A partir de EJB 3 se puede usar cualquiera de estas dos técnicas. Las técnicas no son exclusivas, pueden coexistir anotaciones con descriptores XML y, en el caso de superponerse la metadata, los XML tendrán prioridad y podrán sobrescribir las anotaciones.

  Algunos ejemplos de los contenedores más populares que hay actualmente en el mercado son: Glassfish, de Sun Microsystem, JBoss Application Server, de Red Hat, BEA Weblogic Server y Oracle Application Server, ambos de Oracle y WebSphere de IBM

- **UNA ANOTACIÓN TRANSFORMA UN SIMPLE POJO EN UN EJB**

Figura 32 Transformación simple

4.5.3.1. TIPOS DE EJBS

Existen tres tipos de EJBs:

- **SESSION BEANS:** en una aplicación enterprise típica, dividida en cuatro grandes capas o layers (presentación, lógica de negocio (business logic), persistencia y base de datos (DBMS)), los Session Beans viven en la lógica de negocio. Hay dos grandes tipos de **Session Beans:** Stateless y Stateful, el primero no conserva el estado de ninguno de sus atributos de la invocación de un método a otro y el segundo conserva el estado a lo largo de toda una sesión. Los Session Beans Stateless son los únicos que pueden exponerse como web services.

- **ENTITIES**: los entities viven en la capa de persistencia y son los EJBs que manejan la Java Persistence API (JPA), también parte de la especificación de EJB 3.0. Los entities son POJOs con cierta metadata que permite a la capa de persistencia mapear los atributos de la clase a las tablas de la base de datos y sus relaciones.

- **MESSAGE-DRIVEN BEANS (MDBS)**: también viven en la lógica de negocio y los servicios que proveen son parecidos a los Session Beans, con la diferencia de que los MDBs son usados para invocar métodos de forma asincrónica. Cuando se produce la invocación de un método de un MDB desde un cliente, la llamada no bloquea el código del cliente y el mismo puede seguir con su ejecución, sin tener que esperar indefinidamente por la respuesta del servidor. Los MDBs encapsulan el popular servicio de mensajería de Java, JMS. Hay una analogía muy interesante en el libro que dice que los MDBs son a JMS lo que JDBC es a SQL.

Los Session Beans son invocados por el cliente con el propósito de ejecutar operaciones de negocio específicas, como por ejemplo podría ser chequear la historia crediticia del cliente de un banco. El nombre sesión implica que la instancia del bean estará disponible durante una unidad de trabajo (unit of work) y no sobrevivirá a una caída del servidor. Un bean de sesión sirve para modelar cualquier funcionalidad lógica de una aplicación.

Los MDBs también procesan lógica de negocio, pero un cliente nunca invoca a un método de un MDB directamente. El sistema de mensajería asincrónica propone la utilización de una capa intermedia en la comunicación entre el productor y el consumidor del mensaje. En EJB 3, esta capa se llama MOM (Message-oriented middleware). Básicamente la MOM es un software que permite funcionar como servidor de mensajería, reteniendo los mensajes del productor y enviándolos posteriormente al consumidor en el momento en que esté disponible para recibirlo. (Es un funcionamiento similar al de un servidor de correo electrónico.) Algunos ejemplos típicos de servidores de mensajería son WebSphere MQ de IBM, SonicMQ, Advanced Queueing de Oracle y TIBCO.

4.5.3.2. ENTITIES Y LA JAVA PERSISTENCE API

Debido al auge de los frameworks ORM (Hibernate, TopLink, etc), Sun tuvo que replantear su complicada y anti-natural especificación de persistencia, que tanto dolores de cabeza le daba a los programadores que usaban EJB 2, al extremo que optó por reescribirla casi por completo. Así nació JPA.

JPA persiste automáticamente los objetos Java usando la técnica de ORM (mapeo objeto-relacional). Los ORM del mercado se han adaptado a esta especificación y, hoy en día, cualquier framework de persistencia ORM soporta JPA.

El estándar JPA define:

- La configuración ORM mediante metadata que mapea **entidades** a **tablas relacionales**.

- La interface EntityManager, que define una API estándar para realizar las operaciones de persistencia (CRUD) de las entidades.

- El Java Persistence Query Language (**JPQL**), para consultas y lecturas de datos de aplicación persistidos (algo así como un SQL orientado a objetos).

En general, los contenedores proveen su ORM. Por ejemplo, JBoss provee Hibernate, Glassfish provee TopLink. JPA puede funcionar independientemente del resto de los componentes de EJB 3 y hasta puede ser usado en una aplicación desktop común y corriente, una aplicación Java SE.

Los entities son los objetos Java que se persisten en la base de datos. Mientras que los Session Beans son los **verbos** del sistema, las entidades son los **sustantivos**.

4.5.3.3. CONTENEDOR DE EJBS

Así como cuando compilamos una clase simple de Java, necesitamos una Java Virtual Machine (JVM) para ejecutarla, necesitamos un contenedor de EJBs para ejecutar los Session Beans y los MDBs.

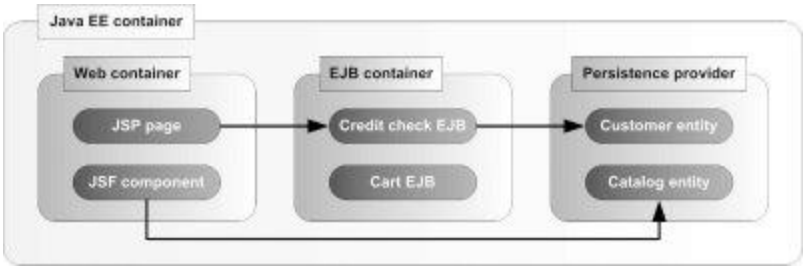


Figura 33 Contenedores de EJBs

Un contenedor Java EE es un servidor de aplicaciones que es capaz de ejecutar EJBs, puede servir como web container y además puede incluir otras APIS y servicios, como por ejemplo el de persistencia. Algunos servidores de aplicaciones pueden proveer solamente un contenedor web, como es el caso de Apache Tomcat, o sólo proveer servicios de persistencia, como es el caso de Hibernate. Un servidor de aplicaciones como JBoss trae un servidor Apache Tomcat y un servidor Hibernate, que se ejecutan dentro de forma transparente.

4.5.3.4. QUÉ SERVICIO PROVEEN LOS EJBS

- **INTEGRACIÓN**: Proveen una forma de acoplar en tiempo de ejecución diferentes componentes, mediante simple configuración de anotaciones o XMLs. El acoplamiento se puede hacer mediante Inyección de Dependencia (DI) o usando JNDI, como se hacía en EJB 2 (explicaré el concepto de Inyección de Dependencia en detalle en el próximo post). La integración es un servicio que proveen los beans de sesión y los MDBs.

- **POOLING:** El contenedor de EJBs crea para componentes EJB un pool de instancias que es compartido por los diferentes clientes. Aunque cada cliente ve como si recibiera siempre instancias diferentes de los EJB, el contenedor está constantemente rehusando objetos para optimizar memoria. El pooling es un servicio que se aplica a los Stateless Session Beans y a los MDBs.

- **THREAD-SAFELY:** El programador puede escribir componentes del lado del servidor como si estuviera trabajando en una aplicación sencilla con un solo thread (hilo). El contenedor se encarga de que los EJBs tengan el soporte adecuado para una aplicación multi-usuario (como son en general las aplicaciones enterprise) de forma transparente, asegurando el acceso seguro, consistente y performante. Aplica a los beans de sesión y a los MDBs.

- **ADMINISTRACIÓN DE ESTADOS**: El contenedor de EJBs almacena y maneja el estado de un Stateful Session Bean de forma transparente, lo que significa que el programador puede mantener el estado de los miembros de una clase como si estuviera desarrollando una aplicación desktop ordinaria. El contenedor maneja los detalles de las sesiones.

- **MENSAJERÍA:** Mediante los MDBs es posible desacoplar por completo dos componentes para que se comuniquen de forma asincrónica, sin reparar demasiado en los mecanismos de la JMS API que los MDBs encapsulan.

- **TRANSACCIONES:**EJB soporta el manejo de transacciones declarativas que permiten agregar comportamiento transaccional a un componente simplemente usando anotaciones o XMLs de

configuración. Esto significa que cuando un método de un EJB (Session Bean o MDB) se completa normalmente, el contenedor se encargará de commitear la transacción y efectivizar los cambios que se realizaron en los datos de forma permanente. Si algo fallara durante la ejecución del método (una excepción o cualquier otro problema), la transacción haría un rollback y es como si el método jamás se hubiera invocado.

- **SEGURIDAD:**EJB soporta integración con la Java Authentication and Authorization Service (JAAS) API, haciendo casi transparente el manejo transversal de la seguridad. Aplica a todos los Session Beans.

- **INTERCEPTORS**:EJB introduce un framework liviano y simple para AOP (programación orientada a aspectos). No es tan robusto y completo como otros, pero es lo suficientemente útil para que sea utilizado por los demás servicios del contenedor para brindarnos de forma invisible los crosscutting concerns de seguridad, transacciones, thread-safely. Además, nosotros, como programadores, podemos agregar nuevos aspectos como logging o auditoria y demás de forma configurable.

- **ACCESO REMOTO:** Es posible acceder de forma remota a distintos EJBs de forma sencilla, simplemente mediante la Inyección de Dependencia. El procedimiento para inyectar un componente local o uno remoto es exactamente el mismo, abstrayéndonos de las complicaciones específicas de RMI o similares. Este servicio aplica únicamente a los Session Beans.

- **WEB SERVICES:** Un Stateless Session Bean puede publicar sus métodos como web services mediante una sencilla anotación

- **PERSISTENCIA:** EJB 3 provee la especificación JPA para el mapeo de objetos (Entities) a tablas.

- **CATCHING AND PERFORMANCE:**JPA provee de forma transparente un importante número de servicios que permiten usar un caché de entidades en memoria y una lectura y escritura sobre la base de datos altamente per formante.

5. DESARROLLO DEL SISTEMA

5.1. ANÁLISIS DE REQUERIMIENTOS

5.1.1. ACTA DE TRABAJO NUMERO 1

ACTA DE TRABAJO 001		
Proyecto: SCAPA		
Tema a tratar: Conocimiento del estado actual del sistema de Comercialización de Agua Potable		
Fecha: 01 de Junio del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Carlos Oña	Director del Departamento de Agua Potable.	
Betty Aldás	Cajera del Dep.de Agua Potable	
Marco Pozo	Director del Departamento de Sistemas	
Observaciones:		
Se procedió a explicar cuál es procedimiento para realizar el procedo de recaudación y facturación del Agua Potable y Alcantarillado: Explicación del proceso de registro de actividades en el proceso de registro de nuevas acometidas. Se explica el proceso de ingreso de nuevas lecturas, actualizaciones de las mismas, ingreso de Nuevos previos, Usuarios, venta de medidores, como también de ciertas consultas que realiza el sistema con una interface realizada en .Net Por lo que manifiestan existe un problema de actualizar la información ingresada, como también el momento que realiza los ingresos al sistema existen procesos que debe llevárselos a cabo manualmente Se explica la codificación de los materiales ingresados al departamento y que no pueden llevar un control de existencias de los mismos. Explica de los nuevos procesos y servicios que desea implementar el departamento para la ciudadanía y que no se pueden realizar debido al tiempo en el que incurre y al costo del proveedor. Exponen que ellos no poseen los códigos fuentes del actual sistema motivo, motivo por el cual no realizan las modificaciones respectivas e		

implementación de los nuevos servicios. Explica también la necesidad de implementar nuevos reportes que ayuden de manera gerencial. Se concluye que es necesario: Realizar un nuevo sistema utilizando tecnología actualizada para el manejo de Servicio de Agua Potable que cumpla con los requerimientos de la municipalidad. Mejorar el proceso de cobros a través la implementación de nuevas ventanillas de cobros. Capacitar al personal para el proceso de captura de datos. Incluir seguridades y controles para restringir el ingreso de datos erróneos asegurando así la confiablidad de los datos Desarrollar interfaces para hacer los ingresos automáticos y disminuir así procesos manuales
Responsabilidades asumidas:

Tabla 11 Acta de Trabajo 1

3.1.1. ACTA DE TRABAJO NÚMERO 2

ACTA DE TRABAJO 002		
Proyecto: SCAPA		
Tema a tratar: Procesos y Actividades y nuevos requerimientos del SCAPA		
Fecha: 30 de Junio del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Caros Oña	Departamento de Agua Potable y Alcantarillado	
Betty Aldás	Departamento de Agua	

	Potable y Alcantarillado	
Marco Pozo	Director del Departamento de Sistemas	

Observaciones:

Se procede a explicar los requerimientos del nuevo sistema de Comercialización de Agua Potable y Alcantarillado:

En el tema de Ingresos al Sistema:

- Permitir ingresar nuevas lecturas, acometidas, tarifas de cobro, valores de venta de materiales y otros.
- Ingreso de Nuevas solicitudes para un Nuevo derecho.
- Ingresar información del usuario como sus datos personales, dirección, barrio, sector, parroquia, etc.

Procesos que debe realizar el Sistema:

- Debe realizar el proceso de asignación de consumo dinámicamente si no contare con el ingreso pertinente hasta una fecha específica, el cálculo es realizar un promedio de los 3 anteriores meses y asignarle al cuarto mes.
- En caso de no tener cancelado los 3 meses debe emitir una alerta de morosidad para el respectivo cliente.
- Contará con actualizaciones dinámicas para todo el sistema.

Reportes que debe Brindar el Sistema:

- Metros cúbicos facturados.
- Emisión de factura web, factura en formato PDF.

Reportes Estadísticos:

- Seguridad que debe brindar el Sistema:
  - Debe contar con la validación respectiva de la información ingresada.
  - No permitirá el ingreso de lecturas menores a la anterior.

○ Tendrá un control del monto total recaudado anualmente. ○ Contará con usuarios ingresados al sistema y solo el administrador dará de alta a los mismos. ○ Control en el proceso de facturación y recaudación. ○ Solo el administrador del sistema realizará las modificaciones y actualizaciones de la información. ➢ El sistema debe permitir la venta de m3 de agua potable. ➢ Se me entrega un documento explicando la ordenanza municipal, con la que se regirá el Sistema de Comercialización de Agua Potable y Alcantarillado.
Responsabilidades asumidas:
El Director del departamento de Agua Potable se compromete a brindar toda la información requerida para el desarrollo del nuevo sistema (Ing. Carlos Oña). El Director del departamento de Sistemas de la municipalidad se compromete a brindar el equipamiento logístico, como tecnológico para el desarrollo del sistema (Ing. Marco Pozo). Realización de un procedimiento que se encargue de la actualización automática de la información. Realizar varios procedimientos para cumplir con los requerimientos y expectativas del sistema de Comercialización de Agua Potable y Alcantarillado.

Tabla 12 Acta de Trabajo 2

3.1.2. ACTA DE TRABAJO NÚMERO 3

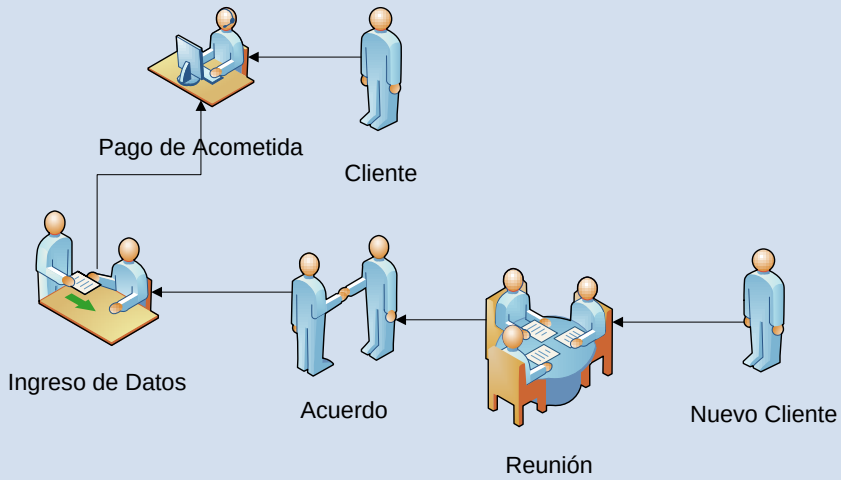
ACTA DE TRABAJO 003		
Proyecto: SCAPA		
Tema a tratar: Manejo del proceso de Recaudación y Facturación, requerimientos por parte de Señora cajera.		
Fecha: 04 de Agosto del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Betty Aldás	Cajera del Servicio de Agua Potable.	

Hugo Enríquez	Personal del Departamento de Agua Potable.	

Observaciones:

Se procedió a explicar el proceso de Recaudación y Facturación del servicio y el proceso de ingreso de nuevas acometidas.

El registro de nuevas acometidas es primero ingresando la solicitud de adquisición de nueva acometidas, el proceso se lleva a cabo en el departamento de Agua Potable.



Paso 1: En el primer paso se mantiene una reunión y se establece los lineamientos para la obtención del servicio.

Paso 2: Se llega a un acuerdo de la reunión.

Paso 3: Se ingresan los datos del Usuario, y su estado en el sistema como no activo.

Paso 4: En el último paso se cancela el costo de la nueva acometida y se camia es estado del cliente en Acometida Activa.

Responsabilidades asumidas:

Realización de una interface para el registro de los ítems que conforman los pasos de ingreso de una nueva acometida.

Tabla 13 Acta de Trabajo 3

3.1.3. ACTA DE TRABAJO NÚMERO 4

ACTA DE TRABAJO 004		
Proyecto: SAPAL		
Tema a tratar: Obtener requerimientos con los cuales contará el nuevo módulo de Recaudación.		
Fecha: 11 de Agosto del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Edwin Madruñero	Desarrollador	
Marco Pozo	Director del Dep. de Informática	
Ing. Carlos Oña (11-08-2010)	Director del Dep. de Agua Potable y Alcantarillado	
Observaciones:		

Se procede a obtener la información y los procesos que maneja cada ítem.
Módulo de Recaudación.
Proceso de captura de Datos.

Gestión Catastral.

Parámetros

- Tipos de identificación (ingreso de código catastral)
- Control de ingreso de cédula (dígito de verificación)
- Determinación de tipo y categoría de clientes
- Estados de clientes
- Rutas, Sectores, Ciudadelas, Manzanas, etc.
- Parámetros personalizados de acuerdo a la oficina de gestión.
- Servicio de Alcantarillado

Administrativa

- Ingreso/Mantenimiento clientes
- Generación de cuentas por oficina.
- Legalización – contratos, órdenes de instalación.
- Medidores, órdenes de instalación
- Cierres temporales(Activación)
- Ingreso de Servicios.
- Generación de inspecciones
- Contrato medidor

- Generar número de localización.

Consultas y Reportes

- Consulta de clientes por múltiples alternativas de selección
- Consulta de medidores, por fecha, agencia, zona, sector, contratista, etc.
- Estado de medidores.
- Estadísticas de medidores (reportes de barras)
- Estadísticas de tipo de conexión (reportes de barras)
- Estadísticas por tipos de clientes y categorías (reportes de barras)
- Clientes pendientes, activos
- Cuentas con medidores, sin medidores
- Medidores instalados por oficina, ruta, sector
- Servicios Disponibles para el Cliente.
- Consultas por el tipo de cliente.

Ingreso de Lecturas

Parámetros

- Mantenimiento de ciclos de lecturas
- Parametrización de ciclos de lecturas(años)
- Parametrización de rangos de lecturas
- Administrativa
- Hoja para tomas de lecturas por oficina,
- Captura de lecturas individuales
- Reingreso de lecturas
- Consultas y Reportes
- Historial de lecturas por cliente
- Lecturas críticas o fuera de rango
- Órdenes de inspección por lecturas críticas o fuera de rango
- Anomalías y verificación de lecturas

Proceso de Cobro.

Recaudación

Parámetros

- Mantenimiento de formas de pagos
- Motivos de devoluciones
- Autorizaciones de devoluciones

Administrativas

- Recepción de pagos (formas de pago)
- Ingreso del detalle de pago de acuerdo a la forma de pago
- Reversos de pagos
- Consultas y Reportes
- Resumen / Detalle de Recaudación por cajero, día, mes, año
- Resumen / Detalle de Recaudación por rubro, día, mes, año
- Reporte de Auditoría de reversos
- Estadística de la gestión de recaudación

Manejo de Ventanillas

Parámetros

- Mantenimiento de Cajas

- Administrativas
  - Cierre de día
  - Registro depósito bancario
- Consultas y Reportes
  - Resumen / Detalle de Recaudación por cajero, día, mes, año
  - Resumen / Detalle de Recaudación por rubro, día, mes, año
  - Cierre del Día.
  - Reporte de lo Depositado en la Entidad Bancaria.

Depósitos

- Parámetros
  - Mantenimiento de los estados de clientes
  - Registro del pago del Cliente.
- Administrativas
  - Mantenimiento del Estado de pago
  - Registro del pago de los clientes
- Consultas y Reportes
  - Consulta de Clientes con estado pagado al día, mes, año.
  - Consulta del monto recaudado al día, mes, año.

Seguridad y Validación

- Parámetros
  - Mantenimiento de Tipos de Usuarios

- Mantenimiento de Tipos de Acceso
- Mantenimiento de Caducidad de Clave de Acceso

➢ Administrativas

- Ingreso / Actualización de Usuarios
- Altas y Bajas de Usuarios
- Administración de Claves de Acceso

➢ Consultas y Reportes

- Detalle de Usuarios por Tipo de Acceso
- Detalle de Tipos de Acceso
- Detalle de procesos específicos por usuarios

Gestión de Cartera y Morosidad

➢ Parámetros
  - Parametrización de tipos de notificaciones
  - Ingreso de Parámetros de Interés según el INEN
➢ Administrativas
  - Generación de notificaciones y avisos de pagos individuales y/o masivas
  - Mantenimiento de Parámetros de Interés
  - Cálculo de interés por mora a partir de los 3 meses.
➢ Consultas y Reportes
  - Impresión de notificaciones individuales y masivas
  - Reportes de registros de notificaciones y avisos de pagos
  - Estadística de la gestión de notificaciones y avisos de pagos
  - Consultas de los parámetros de interés

Atención al cliente (módulo Básico)

➢ Parámetros
  - Mantenimiento de tipos de Reclamos
  - Mantenimiento de venta de Agua
  - Venta de materiales
➢ Administrativas
  - Recepción Reclamos
  - Ingreso/Mantenimiento de Venta de Agua Potable
  - Ingreso/Mantenimiento de venta de Materiales
➢ Consultas y Reportes
  - Resumen / Detalle de Clientes atendidos
  - Estadísticas de Servicios al Cliente

○ Factura de venta de agua Potable ○ Factura de venta de Materiales. ○ Reporte de Materiales Vendidos ○ Orden de despacho a bodega ○ Consultas del valor a pagar Reportes Generales ➢ Metros cúbicos recaudados ➢ Metros cúbicos no recaudados. ➢ Total de lectura mensual, anual ➢ Numero de acometidas, clientes ➢ Reporte de cobros anuales por tipos

Tabla 14 Acta de Trabajo 4

3.1.4. ACTA DE TRABAJO NÚMERO 5

ACTA DE TRABAJO 005		
Proyecto: SCAPA		
Tema a tratar: Obtener requerimientos del módulo de Facturación		
Fecha: 11 de Agosto del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Edwin Madruñero	Desarrollador	
Ing. Carlos Oña(11-08-2010)	Director del Dep. de Agua Potable	
Observaciones:		
Se procede a obtener la información y los procesos que maneja cada ítem. Módulo de Facturación. ➢ Captura de Datos. ➢ Mantenimiento pliego tarifario ➢ Calculo de Valores a pagar. ➢ Configuración de facturación ➢ Asignación de Rubros extras ➢ Ingreso de rubros por prestación de servicios. ➢ Ingreso de cobros de multas. ➢ Ingreso de ventas Ingreso de costo de servicios.		

➢ Emisión de Factura ➢ Generación de facturas por ruta Reporte ○ Resumen / Detalle facturación por categorías de clientes ○ Resumen / Detalle Facturación por sectores ○ Facturación por tarifa ○ Facturación por tarifa – rubros ○ Evolución de consumos por oficina, zona, sector, manzana ○ Balance Facturación Vs. Recaudación por oficina, zona, sector, manzana ○ Cartera vencida día, mes, año ○ Saldos por usuario ○ Gráfica de resumen de facturación ○ Estadística de consumo facturado
Responsabilidades asumidas:
Realizar una interfaz como prototipo de desarrollo.

Tabla 15 Acta de Trabajo 5

3.1.5. ACTA DE TRABAJO NÚMERO 6

ACTA DE TRABAJO 006		
Proyecto: SCAPA		
Tema a tratar: Presentación de 70 % de las formas del Sistema de Agua Potable		
Fecha: 15 de Noviembre del 2010		
Participantes:		
Nombre	**Cargo**	**Firma**
Juan Carlos Bedón	Administrador de sistemas	
Marco Pozo	Administrador del Proyecto	
Hugo Enríquez	Director del Departamento de Agua Potable	
Observaciones:		

La reunión que se mantuvo fue para realizarles la presentación del 70% de los formularios del Sistema de Agua Potable y Alcantarillado del Gobierno Municipal del Cantón Montúfar en la cual se realizaron las siguientes correcciones: ➢ Corrección del nombre del Ilustre Municipalidad del Cantón Montúfar a Gobierno Municipal del Cantón Montúfar ➢ En las búsquedas de los clientes incluir la parte de ingreso del código catastral ➢ Cambiar el nombre de ciertos campos ➢ El código de toda transacción es el número de cuenta ➢ El formulario de recaudación debe tener una opción de realizar un re cálculo del monto a pagar
Responsabilidades asumidas:
Realizar los cambios correspondientes para la comodidad del cliente.

Tabla 16 Acta de Trabajo 6

5.2. DOCUMENTOS DE MISIÓN Y VISIÓN

5.2.1. VISIÓN

Fecha	Versión	Descripción	Autor
02/08/2010	1.0	Creación del documento de visión	Edwin Madruñero
05/08/2010	1.0	Actualización de la Información de responsables	Edwin Madruñero
06/08/2010	1.0	Revisión del Documento	Ing. Marco Pozo
30/05/2011	1.0	Actualización del Documento	Edwin Madruñero

Tabla 17 Revisiones del Documento de Visión

5.2.1.1. INTRODUCCIÓN

5.2.1.1.1. PROPÓSITO

El propósito de este documento en definir a alto nivel los requerimientos del "Sistema de Comercialización de Agua Potable y Alcantarillado (SCAPA)".

Este sistema se encargara de administrar los procesos relacionados con el manejo de la Recaudación y Facturación del suministro de Agua Potable y Alcantarillado.
Realizando tareas como: Realiza el registro de clientes, los cuales tendrán acceso a diferentes servicios ofrecidos por parte del sistema conjuntamente con el departamento de Agua Potable y Alcantarillado.

El sistema ha sido diseñado para evitar procesos repetitivos y lograr la integración, disponibilidad y confiabilidad de la información, ayudando a agilizar los procesos que tienen que ver con la gestión del Agua Potable y del Alcantarillado.
El detalle de cómo el sistema cubrirá las necesidades de los usuarios se especifica en los casos de uso, que son información adicional no especificada en este documento.

5.2.1.1.2. ALCANCE

Este documento de visión se aplica al "Sistema de Comercialización de Agua Potable y Alcantarillado" que será implementado por el personal del Centro de Cómputo de la Municipalidad del Cantón Montúfar, bajo la supervisión de los técnicos del mismo.

5.2.1.1.3. ARQUITECTURA DEL SOFTWARE

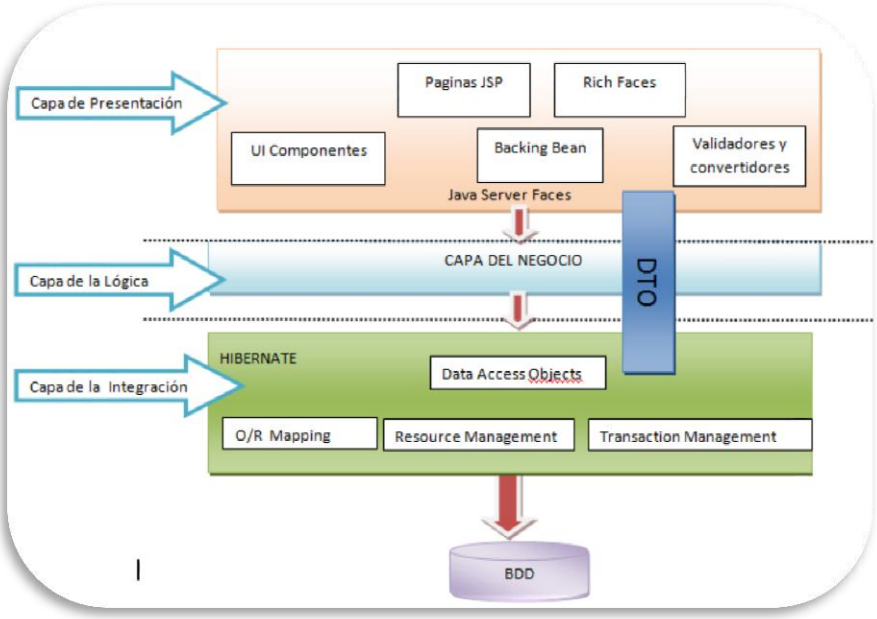


Figura 34 Arquitectura

5.2.1.1.4. DEFINICIONES, SIGLAS Y ABREVIATURAS

Ver Glosario.

5.2.1.1.5. REFERENCIAS

- Glosario: Definiciones abreviaturas y siglas;
- Resumen de los requerimientos de los interesados;
- Resumen del modelo de casos de uso.

5.2.2. POSICIONAMIENTO

5.2.2.1. OPORTUNIDAD DE NEGOCIO

A partir de los procedimientos ya establecidos en el Departamento de Agua Potable y Alcantarillado, y como parte del plan de automatización establecido por el Centro de Cómputo de la Municipalidad, se determina el mejoramiento del sistema de Agua Potable y Alcantarillado que permita optimizar la gestión de las actividades relacionadas al manejo y gestión del servicio de Agua Potable y Alcantarillado, automatizando los módulos de Recaudación y Facturación.

5.2.2.2. DEFINICIÓN DEL PROBLEMA

EL PROBLEMA DE	El Departamento de Agua Potable y Alcantarillado de Municipio de Montúfar utiliza un sistema informático que no cubre las necesidades actuales de la organización, pues utilizan tecnología desactualizada que limita realizar un mantenimiento y adecuación de nuevos requerimientos, además el Municipio no cuenta con los programas fuente que le ayudarían en caso de realizar cambios inmediatos o posibles incorporaciones de servicio ya que disponen de un Departamento de Informática para realizarlo, sin embargo ahora cada modificación la tienen que solicitar al proveedor y su actualización depende del tiempo del mismo, motivos por los cuales se desarrollará un sistema que les permita incorporar tecnología actual y a la vez que el departamento de tecnología posea el código fuente de la aplicación para futuras incorporaciones
QUE AFECTA A	Toda la ciudadanía del Cantón Montúfar, como también a la Municipalidad y a su respectivo Departamento de Agua Potable y Alcantarillado ya que la falta de información rápida y oportuna que permita hacer un análisis y control sobre la gestión del servicio.
EL IMPACTO DE ELLO ES	Que no se puede realizar nuevas actualizaciones con los requerimientos del día a día. No poder realizar una gestión correcta de los recursos, ya que no conoce la información necesaria e actualizada No puede implementar nuevos servicios. No posee un sistema centralizado de la información. El sistema no ofrece los controles de recaudación requeridos por el Centro de Cómputo y por el Departamento de Agua Potable

UNA SOLUCIÓN EXITOSA DEBERÍA	Rediseño e Implementación de los módulos de Facturación y Recaudación del Sistema para la Comercialización del Servicio de Agua Potable y Alcantarillado del Municipio de Montúfar con el fin de aumentar la productividad y brindar un servicio de calidad soportada por una metodología eficiente de desarrollo de software.

Tabla 18 Definición del Problema

5.2.2.3. SENTENCIA QUE DEFINE LA POSICIÓN DEL PRODUCTO.

PARA	El Departamento de Agua Potable y Alcantarillado del Gobierno Municipal de Montúfar,
QUIENES	Director del Departamento, Cajero, Supervisor, Ciudadanía.
EL NOMBRE DEL PRODUCTO	"Sistema de Comercialización de Agua Potable y Alcantarillado SCAPA"
QUE	Almacena información importante y en algunos casos sensibles que apoyan al buen funcionamiento del Departamento de Agua el cual aprovecha sustancialmente
NO COMO	La forma anterior que no contaba con reportes óptimos y controles eficientes
NUESTRO PRODUCTO	Permite automatizar los procesos de recaudación y facturación, lo que implica un correcto registro, control y resguardo de la información; esto se lo realiza mediante una interfaz gráfica, sencilla y amigable. Además proporciona un acceso rápido y actualizado a la información desde cualquier lugar del Gobierno Municipal de Montúfar GMM

Tabla 19 Sistema de definición del problema

5.2.3. DESCRIPCIÓN DE LOS INTERESADOS Y USUARIOS

5.2.3.1. RESUMEN DE LOS INTERESADOS

Interesados son todas aquellas personas directamente involucradas en la definición y alcance del proyecto.

A continuación se presenta la lista de los interesados:

NOMBRE	DESCRIPCIÓN	RESPONSABILIDAD
Coordinador del proyecto	Responsable a nivel directivo del Municipio del Cantón Montúfar del proyecto.	Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los diferentes requerimientos que surjan en el desarrollo del sistema.
Responsable del proyecto	Responsable del proyecto por parte del Área de Sistemas del Municipio del Cantón Montúfar.	Responsable del análisis y diseño del proyecto. Gestiona el correcto desarrollo del proyecto en lo referente a la construcción e implantación.
Responsable funcional	Responsable del proyecto por parte del Departamento de Agua Potable y Alcantarillado.	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y la correcta concepción del sistema.
Empleados Municipales	Responsables del ingreso y manejo de la información.	Responsable de definir la información que se utilizará para generar los procesos correspondientes

Tabla 20 Resumen de Interesados

5.2.3.2. RESUMEN DE LOS USUARIOS

Los usuarios son todas aquellas personas involucradas directamente en el uso del sistema “SCAPA” continuación se presenta una lista de los usuarios:

NOMBRE	DESCRIPCIÓN	RESPONSABILIDAD
Administrador del sistema	Personal del Departamento de Informática del Gobierno Municipal de Montúfar encargada de la administración del Sistema	Administrar eficazmente el sistema (gestionar acceso a usuarios, facilitar mantenimiento al sistema frente a nuevos requerimientos).
Usuario del	Personal del Gobierno	Ingresar y consultar la

sistema	Municipal de Montúfar como son: Recaudadores Supervisor Directores departamentales.	información de cada uno usuario de la Municipalidad que permitirá administrar correctamente los mismos.

Tabla 21 Resumen de Usuarios

5.2.3.3. ENTORNO DE USUARIO

El Sistema, beneficiara al Gobierno Municipal del Montúfar y a la ciudadanía en general ya que permitirá registrar todos los procesos necesarios para realizar la gestión, control y recaudación del servicio de Agua Potable y Alcantarillado

El proceso a seguir para el correcto funcionamiento del sistema es:

1. El usuario final (Cliente) realiza una solicitud de nueva acometida la cual será revisada por el encargado departamental.

2. El usuario final (Cliente), liego podrá acercarse a las oficinas correspondientes y realizar la apertura de la cuanta donde el personal encargado realizara el ingreso de información correspondiente al sistema.

3. Una vez dada de alta la cuanta esta apta para realizar el seguimiento de las lecturas.

4. Después de haber registrado las lecturas puede generar la carta de consumo del respectivo mes.

5.2.3.4. PERFILES DE LOS INTERESADOS

- **COORDINADOR DEL PROYECTO**

Representante	Marco Pozo
Descripción	Responsable a nivel directivo del proyecto.
Tipo	Director
Responsabilidades	Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los diferentes

	requerimientos que surjan en el desarrollo del sistema.
Criterio de éxito	Mantener una funcionalidad integral en los sistemas. Mantener activa la aplicación luego de ser implantada.
Implicación	Revisor de la administración (Management Reviewer)
Entregable	N/A
Comentarios	Mantener una relación constante con el desarrollo del proyecto. Brindar apoyo a nivel gerencial cuando sea necesario.

Tabla 22 Coordinador del Proyecto

- **RESPONSABLE DEL PROYECTO**

Representante	Ing. Juan Carlos Bedón
Descripción	Responsable del proyecto por parte del Departamento de Sistemas del Gobierno Municipal de Montúfar
Tipo	Analista de sistemas
Responsabilidades	Responsable del análisis y diseño del proyecto. Gestiona el correcto desarrollo del proyecto en lo referente a la construcción e implantación.
Criterios de éxito	Cumplir con el cronograma determinado. Obtener un sistema de calidad que cumpla con los requerimientos funcionales establecidos.
Implicación	Jefe de proyecto (Project Manager)
Entregables	Documento de visión Glosario Lista de riesgos Resumen del modelo de casos de uso Especificaciones del modelo de casos de uso Especificaciones complementarias
Comentarios	

Tabla 23 Responsable del Proyecto

- **RESPONSABLE FUNCIONAL**

Representante	Ing. Carlos Oña
Descripción	Responsable del proyecto por parte del Departamento Agua Potable y Alcantarillado
Tipo	Experto en el tema
Responsabilidades	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y la correcta concepción del sistema.

	Coordinar las pruebas de validación del nuevo sistema. Coordinar y asegurar la capacitación de los usuarios.
Criterios de éxito	Obtener un sistema de calidad que cumpla con los requerimientos funcionales establecidos.
Implicación	Aprueba las especificaciones funcionales y las pruebas realizadas.
Entregables	Documento de revisión de las especificaciones funcionales. Documento de revisión de las pruebas funcionales
Comentarios	

Tabla 24 Responsable Funcional

5.2.3.5. PERFILES DE USUARIO

- **ADMINISTRADOR DEL SISTEMA**

Representante	Ing. Juan Carlos Bedón
Descripción	Persona del Centro de Cómputo que administra el sistema SCAPA.
Tipo	Operador, Analista de Sistemas
Responsabilidades	Administrar funcionalmente el sistema: gestionar acceso a usuarios, dar mantenimiento al sistema frente a nuevos requerimientos.
Criterios de éxito	N/A
Implicación	N/A
Entregables	Bitácora de control de nuevos requerimientos.
Comentarios	N/A

Tabla 25 Administrador del Sistema

- **ADMINISTRADOR FUNCIONAL DEL SISTEMA**

Representante	Hugo Enríquez
Descripción	Persona del Departamento de Agua Potable y Alcantarillado del Municipio de Montufar que administra el sistema SCAPA.
Tipo	Secretario del Departamento de Agua Potable y Alcantarillado.
Responsabilidades	Administrar funcionalmente el sistema: definición de Clientes a registrar.

	Ingreso de Información de los nuevos clientes. Ingreso de Lecturas. Actualización de Información Emisión de Reportes de Acuerdo al rol asignado
Criterios de éxito	N/A
Implicación	N/A
Entregables	N/A
Comentarios	N/A

Tabla 26 Administrador funcional del Sistema

- **USUARIO DEL SISTEMA**

Representante	N/A
Descripción	Personal del Departamento de Agua Potable y Alcantarillado del Ilustre Municipalidad del Cantón Montúfar que hará uso del sistema
Tipo	Personal del departamento de Agua Potable y Alcantarillado
Responsabilidades	Ingresar la información de los clientes del servicio, realiza las peticiones de cobro, imprime la respectiva factura. Ingreso de nuevas lecturas. Emisión de Reportes de Acuerdo al rol asignado Consultar el estado de las cuentas.
Criterios de éxito	N/A
Implicación	N/A
Entregables	N/A
Comentarios	N/A

Tabla 27 Usuarios del Sistema

- **USUARIO DE GESTIÓN DEL SISTEMA**

Representante	Betty Aldás
Descripción	Personal del Departamento de Agua Potable y Alcantarillado
Tipo	Cajera
Responsabilidades	Emisión de Facturas. Emisión de Reportes de acuerdo al rol asignado.
Criterios de éxito	N/A
Implicación	N/A
Entregables	N/A

Comentarios	N/A

Tabla 28 Perfil de Usuarios del Sistema

5.2.3.6. NECESIDADES DE LOS INTERESADOS Y USUARIOS

NECESIDADES	PRIORIDAD	INQUIETUDES	SOLUCIÓN ACTUAL	SOLUCIÓN PROPUESTA
Diseñar un sistema que facilite la consolidación e integración de información concerniente al registro de consumo del Agua Potable, nuevas Acometidas, Materiales y permita automatizar los cálculos.	Alta	El sistema debe consolidar la información para facilitar el registro y administración del Servicio de Agua Potable y Alcantarillado.	Tiene varios inconvenientes: carece de facilidades, controles e integración.	Rediseñar e implementar los módulos de facturación y recaudación del Servicio de Agua Potable y Alcantarillado para el Gobierno Municipal de Montúfar.
Mantener un registro detallado de los procesos concernientes a la gestión y control del Servicio de suministro de Agua Potable y Alcantarillado	Alta	N/A	Actualmente se registra en forma independiente, cantidades	Desarrollar interfaces que faciliten la integración.
Elaborar el sistema utilizando herramientas que facilite y agilice su	Alta	Se debe utilizar las herramientas existentes o adquirir nuevo	N/A	Desarrollar el sistema utilizando herramientas libres con Tecnología

desarrollo.		software de desarrollo.		Java.
La interfaz del sistema debe ser fácil de manejar, cumpliendo con todos los requerimientos establecidos.	Alta	Cumplir con todos los requerimientos de los usuarios.	Existen interface poco amigable y no cuentan con el control suficiente.	Desarrollo con la ayuda de los expertos en el tema.
Los códigos fuentes deber encontrarse a la disposición del Departamento de Cómputo de La municipalidad de Montúfar para posteriores modificaciones e incremento de nuevos servicios	Alta	Se debe Comprar los fuentes del Actual sistema o realizar un sistema con tecnología Actualizada	N/A	Desarrollar un nuevo Sistema de Comercialización de Agua Potable y Alcantarillado acorde a las necesidades del Municipio.

Tabla 29 Necesidades de los Interesados

5.2.3.7. ALTERNATIVAS Y COMPETENCIA

- **ADQUIRIR UN SISTEMA DESARROLLADO EXTERNAMENTE.**

Se ha mostrado interés en buscar alternativas externas para solucionar los diversos requerimientos. Pero en la actualidad existen herramientas en el mercado, las cuales son de un gran consto y no se ajustan a los requerimientos de la Municipalidad, puesto que la entidad necesita un sistema personalizado.

5.2.4. VISTA GENERAL DEL PRODUCTO

Esta sección provee de información a alto nivel de la funciones del módulo a implementar partiendo de procesos principales como son: Ingreso de clientes, ingreso de lecturas, etc. Realizados en el departamento de Agua Potable y Alcantarillado del Gobierno Municipal de Montúfar.

Es presente sistema permite registrar y controlar la gestión del comercialización del agua potable y del alcantarillado realizado por el personal del departamento

5.2.4.1. PERSPECTIVA DEL PRODUCTO

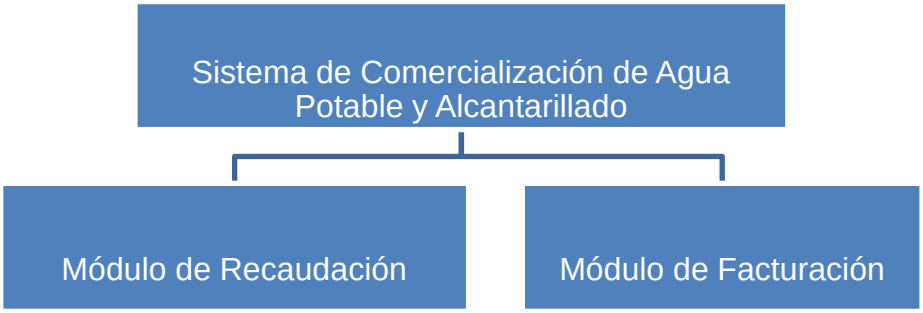


Figura 35 Perspectivas del Producto

5.2.4.2. RESUMEN DE CAPACIDADES

- **SISTEMA DE COMERCIALIZACIÓN DE AGUA POTABLE Y ALCANTARILLADO**

BENEFICIOS PARA EL USUARIO	CARACTERÍSTICAS QUE LO SOPORTAN
El registro de Nuevas Acometidas, tarifas de consumo, venta de materiales, será completo y correcto automatizando cálculos.	El Ingreso de la información por el personal del Departamento de Agua Potable y Alcantarillado será de forma sistematizada y ordenada. El personal del departamento tendrá una herramienta de registro de datos congruente y sincronizada con el módulo de facturación
Los usuarios contarán con una herramienta unificada.	Se evitara realizar cálculos de manera separada.
Se tendrá alta disponibilidad.	El acceso a la información a través de la Web permitirá a los usuarios un acceso inmediato desde cualquier punto de la intranet del Municipio.
Facilidades para el análisis de la información.	Brindará diferentes reportes y funciones de consulta.
Contará con un control de Cartera, morosidad de pagos.	Tendrá un control de la recaudación del dinero por parte de las cajas.
Los usuarios podrán realizar actualizaciones de la información sin tener el peligro de que la información no esté sincronizada entre módulos	Brindará formas de actualizar información ingresada y se actualizará en los dos módulos, con ello lograr tener información sincronizada.

Tabla 30 Sistema de Comercialización de Agua Potable y Alcantarillado

5.2.4.3. SUPOSICIONES Y DEPENDENCIAS

N/A

5.2.4.4. COSTOS Y PRECIOS

N/A

5.2.4.5. LICENCIAMIENTO E INSTALACIÓN

- Para el desarrollo del sistema no es necesario la adquisición de la licencias, debido a que se lo desarrollara con tecnología java, utilizando el Framework con el que cuenta para aplicaciones web Java Server Faces.

- La instalación del producto será realizada por el personal del Departamento de Informática del Gobierno Municipal de Montúfar soporte del Centro de Cómputo ya que es un sistema que utilizará tecnología Web.

5.2.5. CARACTERÍSTICAS DEL PRODUCTO

5.2.5.1. FACILIDAD DE ACCESO Y USO

El SCAPA será desarrollado utilizando tecnología Web y las facilidades que ofrecen las herramientas de software libre, lo que permitirá a los usuarios un fácil acceso y uso.

5.2.5.2. UNIFICACIÓN DE LA INFORMACIÓN

Uno de los principales objetivos del SCAPA es registrar las peticiones de la ciudadanía con respecto al Servicio de Agua Potable e incorporar nueva información al sistema para ayuda de la población.

5.2.5.3. MEJOR CONTROL Y VALIDACIÓN DE LA INFORMACIÓN

Los usuarios del Departamento de Agua Potable y Alcantarillado contarán con facilidades para la verificación de la información del servicio de Agua Potable.

5.2.5.4. RESTRICCIONES

Debido a limitaciones que puede soportar el servidor de aplicaciones Tomcat 300 peticiones concurrentes aproximadamente.

5.2.5.5. RANGOS DE CALIDAD

El desarrollo del Sistema SCAPA se elaborará siguiendo la Metodología de Desarrollo de Software RUP, contemplando los parámetros de calidad que la metodología define.

5.2.5.6. PRECEDENCIA Y PRIORIDAD

N/A

5.2.5.7. OTROS REQUERIMIENTOS DEL PRODUCTO

N/A

5.3. DISEÑO

5.3.1. PLAN DE DESARROLLO DEL SOFTWARE

Fecha	Versión	Descripción	Autor
09/08/2010	1.0	Versión preliminar como propuesta de desarrollo.	Edwin Madruñero
12/06/2011	1.0	Revisión de la propuesta de desarrollo	Edwin Madruñero

5.3.1.1. INTRODUCCIÓN

Este Plan de Desarrollo del Software es una versión preliminar preparada para ser incluida en la propuesta elaborada como respuesta al proyecto SCAPA para el Gobierno Municipio de Montúfar. Este documento provee una visión global del enfoque de desarrollo propuesto.
El proyecto ha sido desarrollado en una metodología de Rational Unified Process con el fin de implementar un esquema inicial de ésta metodología para futuros desarrollo.

La orientación del desarrollo propuesto constituye una configuración del proceso RUP de acuerdo a las particularidades del proyecto, seleccionando los roles, actividades y artefactos (documentos entregables). Este documento es a su vez uno de los artefactos de RUP.

5.3.1.1.1. PROPÓSITO

El propósito del Plan de Desarrollo de Software es proporcionar la información necesaria para controlar el proyecto. En él se describe el enfoque de desarrollo del software.
Los usuarios del Plan de Desarrollo de Software son:

El jefe del proyecto lo utiliza para organizar la agenda y necesidades de recursos, y para realizar su seguimiento.

Los miembros del equipo de desarrollo lo usan para entender lo qué deben hacer, cuándo deben hacerlo y qué otras actividades dependen de ello**.**

5.3.1.1.2. ALCANCE

El Plan de Desarrollo del Software describe el plan completo usado para el desarrollo del SCAPA. El detalle de las iteraciones individuales se describe en los planes de cada iteración, documentos que se aportan en forma separada. Durante el proceso de desarrollo en el artefacto "Visión" se definen las características del producto a desarrollar, lo cual constituye la base para la planificación de las iteraciones.

Para la versión 1.0 del Plan de Desarrollo del Software, nos hemos basado en la captura de requisitos por medio del stakeholder representante del Gobierno Municipal de Montúfar para hacer una estimación aproximada, una vez comenzado el proyecto y durante la fase de Inicio se generará la primera versión del artefacto "Visión", el cual se utilizará para refinar este documento. Posteriormente, el avance del proyecto y el seguimiento en cada una de las iteraciones ocasionará el ajuste de este documento produciendo nuevas versiones actualizadas.

5.3.1.1.3. RESUMEN

Después de esta introducción, el resto del documento está organizado en las siguientes secciones:

- Vista General del Proyecto — proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.

- Organización del Proyecto — describe la estructura organizacional del equipo de desarrollo.

- Gestión del Proceso — explica los costos y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.

- Planes y Guías de aplicación — proporciona una vista global del proceso de desarrollo de software, incluyendo métodos, herramientas y técnicas que serán utilizadas.

5.3.1.2. VISTA GENERAL DEL PROYECTO

5.3.1.2.1. PROPÓSITO, ALCANCE Y OBJETIVOS

El Gobierno Municipal de Montúfar es una entidad de carácter público y tiene como misión especial brindar un servicio confiable y eficiente a la ciudadanía del norte del País
La información que a continuación se incluye ha sido extraída de reuniones que se han realizado con el stakeholder de la Municipalidad desde el inicio del proyecto.

A partir de los procedimientos ya establecidos en el Departamento de Informática conjuntamente con el Departamento de Agua Potable y Alcantarillado del Gobierno Municipal de Montúfar, y como parte del plan de automatización establecido por el Departamento de Informática se determina la modificación y migración del sistema existente de Recaudación de Agua Potable y Alcantarillado, de forma que permita mejorar el registro, administración y actividades relacionadas con la Recaudación y Facturación del servicio de Agua Potable.

Los módulos con los que contara el sistema son:

- Módulo de Recaudación
- Módulo de Facturación

5.3.1.2.2. SUPOSICIONES Y RESTRICCIONES

Las suposiciones y restricciones respecto del sistema, y que se derivan directamente de las entrevistas con el stakeholder de la Municipalidad son:

a) Debe calcular automáticamente el consumo, si el consumo actual es cero, realizando un promedio de los tres anteriores meses.

b) Realizar el control de la información ingresada e realizar los reportes exhaustivos del mismo

c) Debe estar preparado para pruebas a realizarse en el presente año 2011.

d) El Modulo será diseñado sobre plataforma WEB y cumplirá con los estándares de calidad vigentes para desarrollo de software. Esto se conseguirá cumpliendo con el estándar PMI para dirección de proyectos, metodología RUP para el proceso de ingeniería de software y herramientas java para la construcción de las aplicaciones.

Como es natural, la lista de suposiciones y restricciones se incrementará durante el desarrollo del proyecto, particularmente una vez establecido el artefacto “Visión”

5.3.1.2.3. ENTREGABLES DEL PROYECTO

A continuación se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto y que constituyen los entregables. Esta lista constituye la configuración de RUP desde la perspectiva de artefactos, y que proponemos para este proyecto.

Es preciso destacar que de acuerdo a la filosofía de RUP (y de todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos. Esto será indicado más adelante cuando se presenten los objetivos de cada iteración.
A continuación se presenta la lista de artefactos propuesta para el proyecto SCAPA:

1) **PLAN DE DESARROLLO DEL SOFTWARE**
   Es el presente documento.

2) **MODELO DE CASOS DE USO DEL NEGOCIO**
   Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso usando estereotipos específicos para este modelo.

3) **VISIÓN**

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

4) **GLOSARIO**

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología común.

5) **MODELO DE CASOS DE USO**

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

6) **ESPECIFICACIONES DE CASOS DE USO**

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

7) **ESPECIFICACIONES ADICIONALES**

Este documento capturará todos los requisitos que no han sido incluidos como parte de los casos de uso y se refieren requisitos no-funcionales globales. Dichos requisitos incluyen: requisitos legales o normas, aplicación de estándares, requisitos de calidad del producto, tales como: confiabilidad, desempeño, etc., u otros requisitos de ambiente, tales como: sistema operativo, requisitos de compatibilidad, etc.

8) **PROTOTIPOS DE INTERFACES DE USUARIO**

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de las interfaces que proveerá el sistema y así, conseguir retroalimentación de su parte respecto a los requisitos del sistema. Estos prototipos se realizarán como: dibujos a mano en papel, dibujos con alguna herramienta gráfica o prototipos ejecutables interactivos, siguiendo ese orden de acuerdo al avance del proyecto. Sólo los de este último tipo serán entregados al final de la fase de

Elaboración, los otros serán desechados. Asimismo, este artefacto, será desechado en la fase de Construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

9) MODELO DE ANÁLISIS Y DISEÑO

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

10) MODELO DE DATOS

Previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un Diagrama de Clases (donde se utiliza un profile UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

11) MODELO DE IMPLEMENTACIÓN

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. (Este modelo es sólo una versión preliminar al final de la fase de Elaboración, posteriormente tiene bastante refinamiento).

12) MODELO DE DESPLIEGUE

Este modelo muestra el despliegue la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes.

13) CASOS DE PRUEBA

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba, y dependiendo del tipo de prueba dicho procedimiento podrá ser automatizable mediante un script de prueba.

14) SOLICITUD DE CAMBIO

Los cambios propuestos para los artefactos se formalizan mediante este documento. Mediante este documento se hace un seguimiento de los defectos detectados, solicitud de mejoras o cambios en los requisitos del producto. Así se provee un registro de decisiones de cambios, de su evaluación de impacto, y se asegura que éstos sean conocidos por el equipo de desarrollo. Los cambios se establecen respecto de la última baseline (el estado del conjunto de los artefactos en un momento determinado del proyecto) establecida. En nuestro caso al final de cada iteración se establecerá una baseline.

15) PLAN DE ITERACIÓN

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados, dependencias entre ellas. Se realiza para cada iteración, y para todas las fases.

16) EVALUACIÓN DE ITERACIÓN

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

17) LISTA DE RIESGOS

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación.

18) MANUAL DE INSTALACIÓN

Este documento incluye las instrucciones para realizar la instalación del producto.

19) MATERIAL DE APOYO AL USUARIO FINAL

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Guías del Usuario, Guías de Operación, Guías de Mantenimiento

20) PRODUCTO

Los archivos del producto SCAPA empaquetados y almacenados en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de construcción es

desarrollado incrementalmente e interactivamente, obteniéndose una reléase al final de cada iteración.

Los artefactos 18, 19 y 20 se generarán a partir de la fase de Construcción, con lo cual se han incluido aquí sólo para dar una visión global de todos los artefactos que se generarán en el proceso de desarrollo.

5.3.1.2.4. EVOLUCIÓN DEL PLAN DE DESARROLLO DEL SOFTWARE

El Plan de Desarrollo del Software se revisará semanalmente y se refinará antes del comienzo de cada iteración.

5.3.1.3. ORGANIZACIÓN DEL PROYECTO

5.3.1.3.1. PARTICIPANTES EN EL PROYECTO

De momento no se incluye el personal que designará Responsable del Proyecto, Comité de Control y Seguimiento, otros participantes que se estimen convenientes para proporcionar los requisitos y validar el sistema.

El resto del personal del proyecto considerando las fases de Inicio, Elaboración y dos iteraciones de la fase de Construcción, estará formado por los siguientes puestos de trabajo y personal asociado:

- **JEFE DE PROYECTO**. Con una experiencia en metodologías de desarrollo, herramientas CASE y notaciones, en particular la notación UML y el proceso de desarrollo RUP.

- **ANALISTA DE SISTEMAS**. El perfil establecido es: Ingeniero en Informática con conocimientos de UML, uno de ellos al menos con experiencia en sistemas afines a la línea del proyecto

- **ANALISTAS - PROGRAMADORES**. Con experiencia en el entorno de desarrollo del proyecto, con el fin de que los prototipos puedan ser lo más cercanos posibles al producto final.

- **INGENIERO DE SOFTWARE**. El perfil establecido es: Ingeniero en Informática que participará realizando labores de gestión de requisitos, gestión de configuración, documentación y diseño de datos.

Encargada de las pruebas funcionales del sistema, realizará la labor de Tester.

5.3.1.3.2. INTERFACES EXTERNAS

El Gobierno Municipal de Montúfar se encargara de define los participantes del proyecto que proporcionarán los requisitos del sistema, y entre ellos quiénes serán los encargados de evaluar los artefactos de acuerdo a cada módulo y según el plan establecido.
El equipo de desarrollo interactuará activamente con los participantes para especificación y validación de los artefactos generados.

5.3.1.3.3. ROLES Y RESPONSABILIDADES

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP.

PUESTO	RESPONSABILIDAD
Jefe de Proyecto	El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto.
Analista de Sistemas	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.
Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario
Ingeniero de Software	Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 31 Tabla de Responsabilidades

5.3.1.4. GESTIÓN DEL PROCESO

5.3.1.4.1. PLAN DEL PROYECTO

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto.

5.3.1.4.1.1. ESTIMACIÓN DEL PROYECTO

El presupuesto del proyecto y los recursos involucrados se adjuntan en un documento separado.

- **PLAN DE LAS FASES**

  El desarrollo se efectuará en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar)

FASE	NRO. ITERACIONES	DURACIÓN
Fase de Inicio	1	4 Meses
Fase de Elaboración	2	5 Meses
Fase de Construcción	2	4 Meses
Fase de Transición	2	2 Meses

Tabla 32 Tabla de faces del RUP

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

DESCRIPCIÓN	HITO
Fase de Inicio	En esta fase desarrollará los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente / usuario del artefacto Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera versión de la fase de Construcción

	deben estar analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.

Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis / Diseño. El producto se construye en base a 3 iteraciones, cada una produciendo una reléase a la cual se le aplican las pruebas y se valida con el cliente / usuario. Se comienza la elaboración de material de apoyo al usuario. El hito que marca el fin de esta fase es la versión de la reléase 3.0, con la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregada a los usuarios para pruebas beta.
Fase de Transición	En esta fase se prepararán dos raleases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

- **OBJETIVOS DE LAS ITERACIONES**

FASE	ITERACIÓN	DESCRIPCIÓN	HITOS ASOCIADOS	RIESGOS DIRECCIONADOS
Incepción	Iteración Preliminar	Define el modelo del	Revisión de Casos de	Aclarar los requerimientos.

		negocio, requerimientos del producto, plan de proyecto, y casos de negocio.	Uso	Desarrollar planes de proyecto realista y alcance. Determina la viabilidad de los proyectos desde el punto de vista empresarial.
Fase de Elaboración	Desarrollar una arquitectura prototipo.	Completar el análisis y desarrollo de todos los casos de Usos. Desarrollar arquitectura prototipo.	Arquitectura Prototipo	Arquitectura del Software clarificada. Técnicas de mitigación de riesgos. Prototipo para que el usuario revise.
Fase de Construcción	C1 Iteración – Desarrollo Beta	Implementar y probar casos de uso para proveer la versión Beta.	Beta	Todas las características claves y arquitectura prospectiva usado en la versión Beta... Opinión de los usuarios antes de lanzar la versión del software.
	C2 Iteración – desarrollo versión inicial	Implementar y probar casos de uso restantes, Corregir defectos en versión Beta, e incorporar los comentarios del usuario.	Software	Software revisado por los usuarios. Producto de alta calidad Minimizar defectos Costes de calidad reducido.

		Desarrollar el sistema inicial.		
	C3 Iteración – desarrollar modulo completo	Incorporar mejoras a de defectos en la versión inicial. Desarrollar sistema completo.	Software	Liberación rápida asegurando la satisfacción del cliente. Todas las funciones claves proporcionando el modulo completo
Fase de Transición	Sistema	Empaquetar, distribuir, instalar el sistema	Sistema	

Tabla 33 Objetivos de las Interacciones

5.3.1.4.1.2. CALENDARIO DEL PROYECTO

A continuación se presenta un calendario de las principales tareas del proyecto incluyendo sólo las fases de Inicio y Elaboración. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto. La siguiente figura ilustra este enfoque del rup:

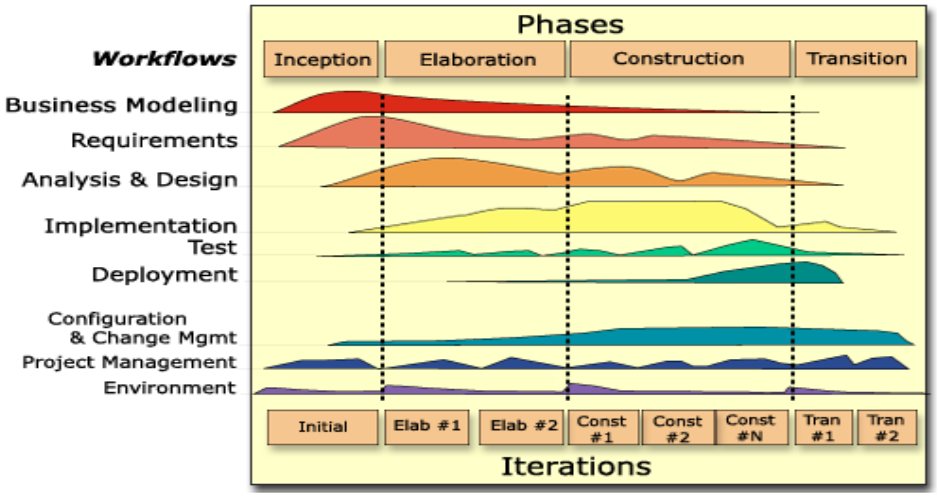


Figura 36 RUP

Para este proyecto se ha establecido el siguiente calendario.

5.3.1.4.2. SEGUIMIENTO Y CONTROL DEL PROYECTO

- **GESTIÓN DE REQUISITOS**

  Los requisitos del sistema son especificados en el artefacto Visión. Cada requisito tendrá una serie de atributos tales como importancia, estado, iteración donde se implementa, etc. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito. Los cambios en los requisitos serán gestionados mediante una Solicitud de Cambio, las cuales serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

- **CONTROL DE PLAZOS**

  El calendario del proyecto tendrá un seguimiento y evaluación semanal por el jefe de proyecto y por el Comité de Seguimiento y Control.

- **CONTROL DE CALIDAD**

  Los defectos detectados en las revisiones y formalizados también en una Solicitud de Cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias Para la revisión de cada artefacto y su correspondiente garantía de calidad se utilizarán las guías de revisión y checklist (listas de verificación) incluidas en RUP.

- **GESTIÓN DE RIESGOS**

  A partir de la fase de Inicio se mantendrá una lista de riesgos asociados al proyecto y de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia. Esta lista será evaluada al menos una vez en cada iteración.

- **GESTIÓN DE CONFIGURACIÓN**

  Se realizará una gestión de configuración para llevar un registro de los artefactos generados y sus versiones. También se incluirá la gestión de las Solicitudes de Cambio y de las modificaciones que éstas

produzcan, informando y publicando dichos cambios para que sean accesibles a todo los participantes en el proyecto. Al final de cada iteración se establecerá líneas de base (un registro del estado de cada artefacto, estableciendo una versión), la cual podrá ser modificada sólo por una Solicitud de Cambio aprobada.

5.3.1.5. REFERENCIAS

- Software Development form Small Teams – A RUP-Centric Aproach, Addison Wesley
- Documentación de Rational Unified Process, manuals de ayuda, tutoriales, etc.
- Ordenanza Municipal para el cobro del Servicio de Agua Potable

5.3.2. MODELO DE CASOS DE USO

5.3.2.1. CAPTURA DE DATOS

5.3.2.1.1. REGISTRO DE CLIENTES

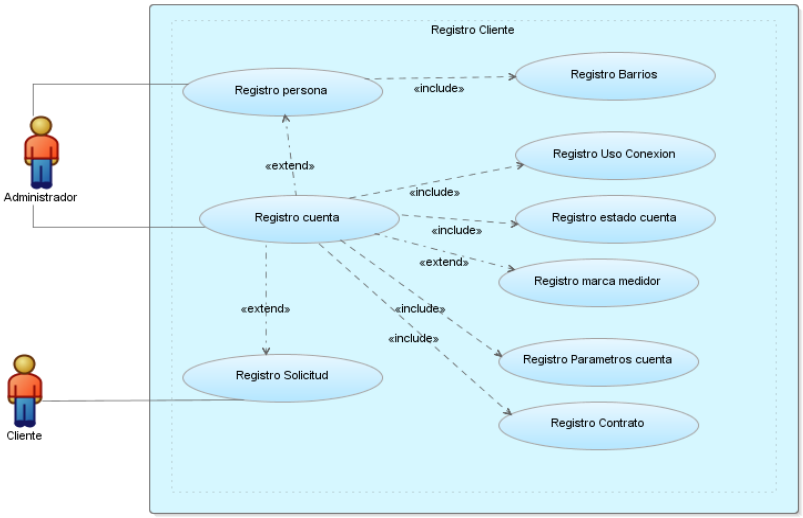


Figura 37 Registro de Clientes

5.3.2.1.2. INGRESO DE LECTURAS

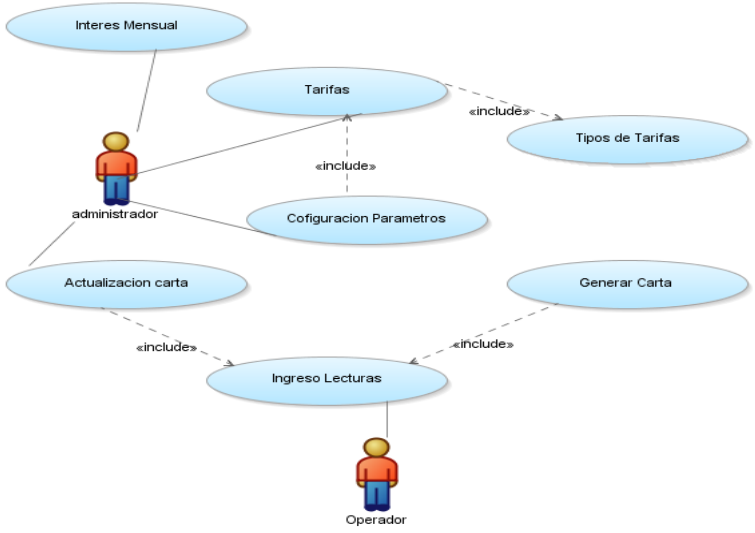


Figura 38 Ingreso de Lecturas

5.3.2.2. MANEJO DE VENTANILLAS

5.3.2.2.1. GESTIÓN DE CAJAS

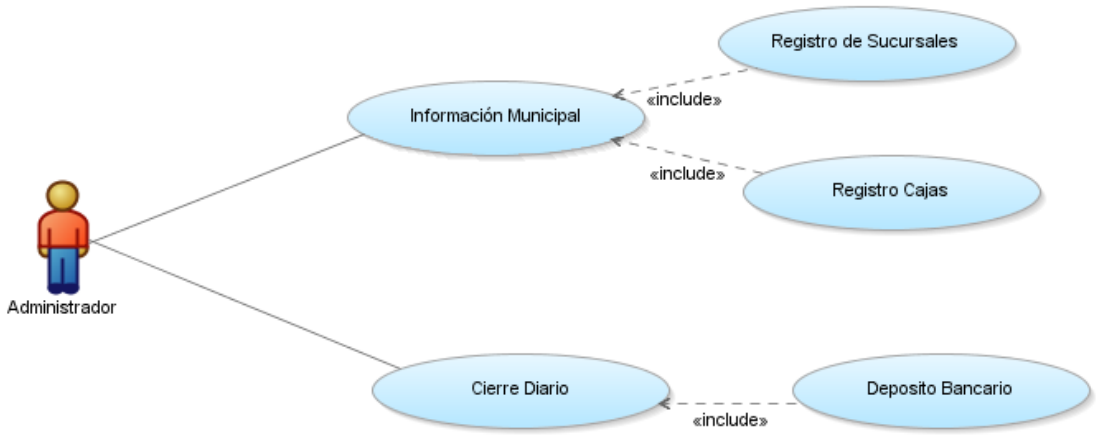


Figura 39 Gestión de Cajas

5.3.2.3. GESTIÓN DE CARTERA Y MOROSIDAD

5.3.2.3.1. GESTIÓN DE MOROSIDAD

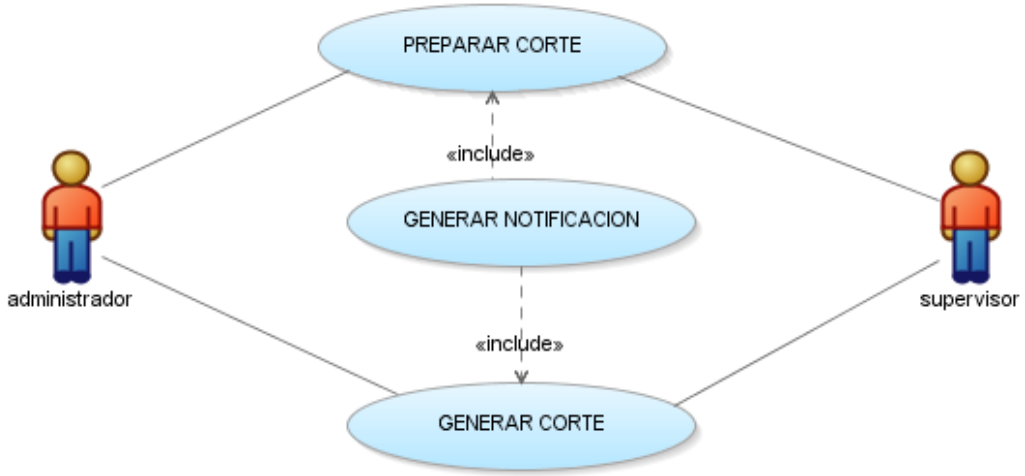


Figura 40 Gestión de Morosidad

5.3.2.4. SEGURIDAD Y VALIDACIONES

5.3.2.4.1. REGISTRO DE USUARIOS

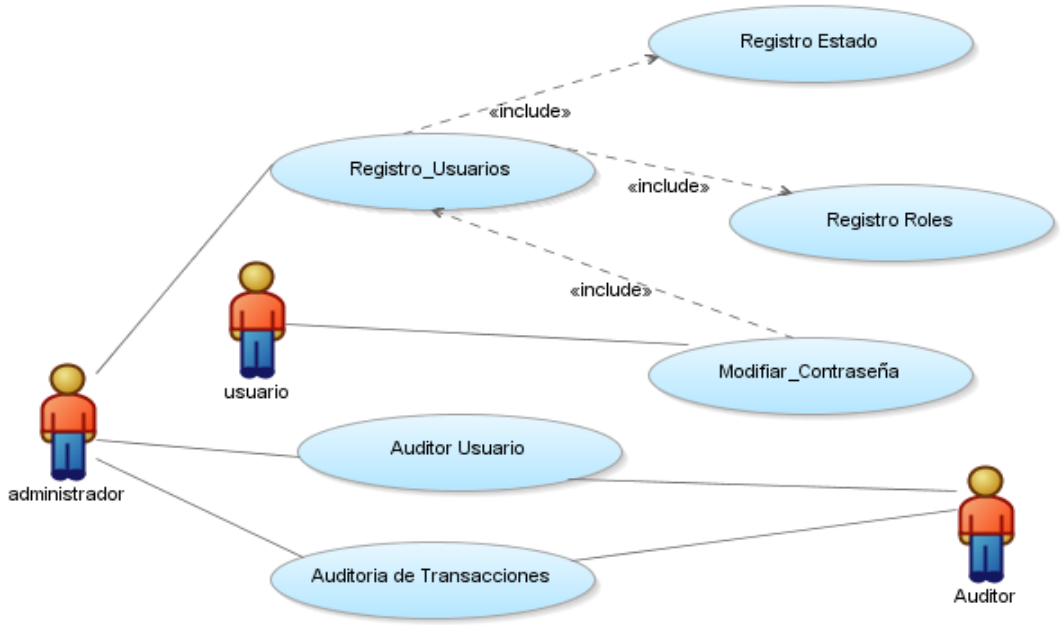


Figura 41 Seguridad del Sistema

5.3.2.5. VENTAS

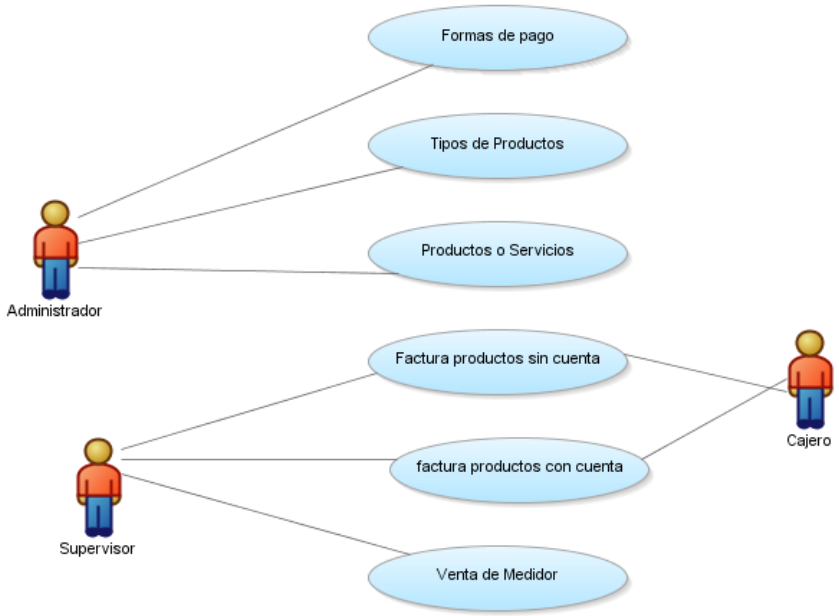


Figura 42 Ventas

5.3.2.6. ATENCIÓN AL CLIENTE

Figura 43 Atención al Cliente

5.3.2.7. RECAUDACIÓN

Figura 44 Recaudación

5.4. ESPECIFICACIÓN DE CASOS DE USO

5.4.1. CAPTURA DE DATOS

5.4.1.1. REGISTRO DE CLIENTES.

5.4.1.1.1. REGISTRO DE ESTADOS DE LA CUENTA

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin

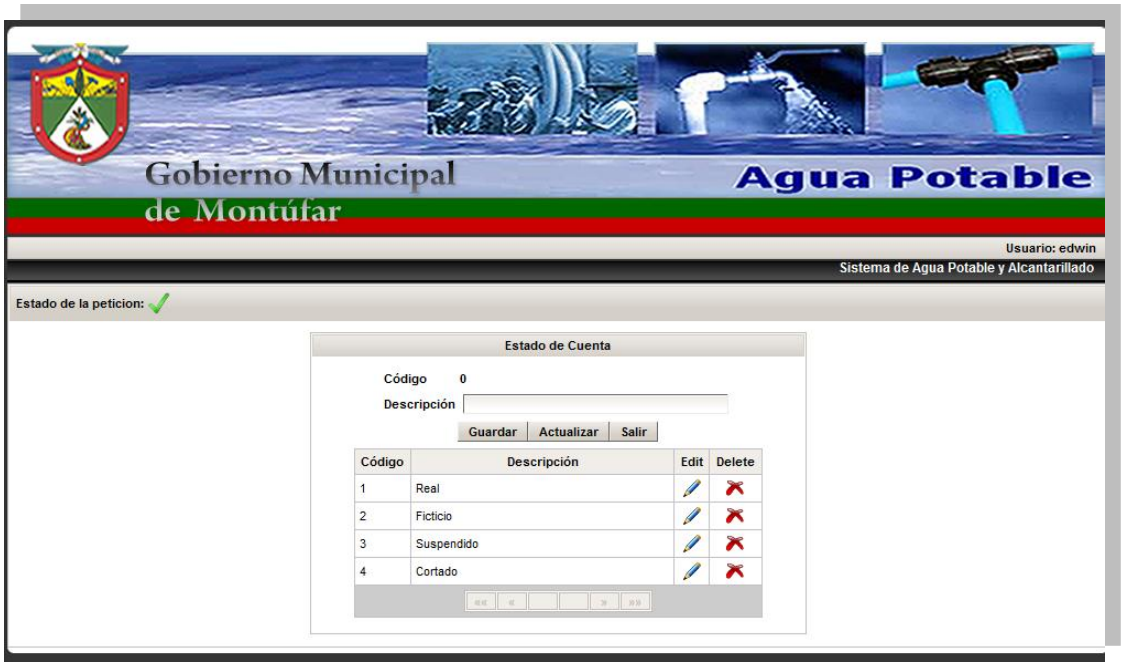
	Madruñero

DESCRIPCIÓN BREVE

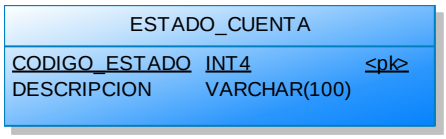
- El caso de uso describe el registro de los diferentes estados de cuenta que puede tener un cliente

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los estados de una cuenta los mismos que serán detallados en la descripción.



- El sistema almacena la información registrada por el usuario.



5.4.1.1.2. ESPECIFICACIÓN DE CASO DE USO CONEXIÓN

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

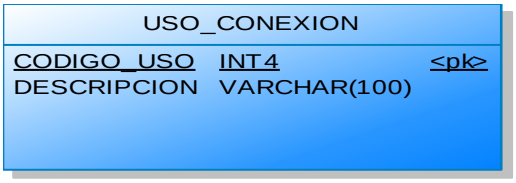
DESCRIPCIÓN BREVE

- El caso de uso describe el registro de los diferentes tipos de cuantas o conexiones que existen

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar las diferentes conexiones, las cuales se detallan en el campo descripción.

- El sistema almacena la información registrada por el usuario.



5.4.1.1.3. ESPECIFICACIÓN DE CASO DE USO REGISTRO DE BARRIOS

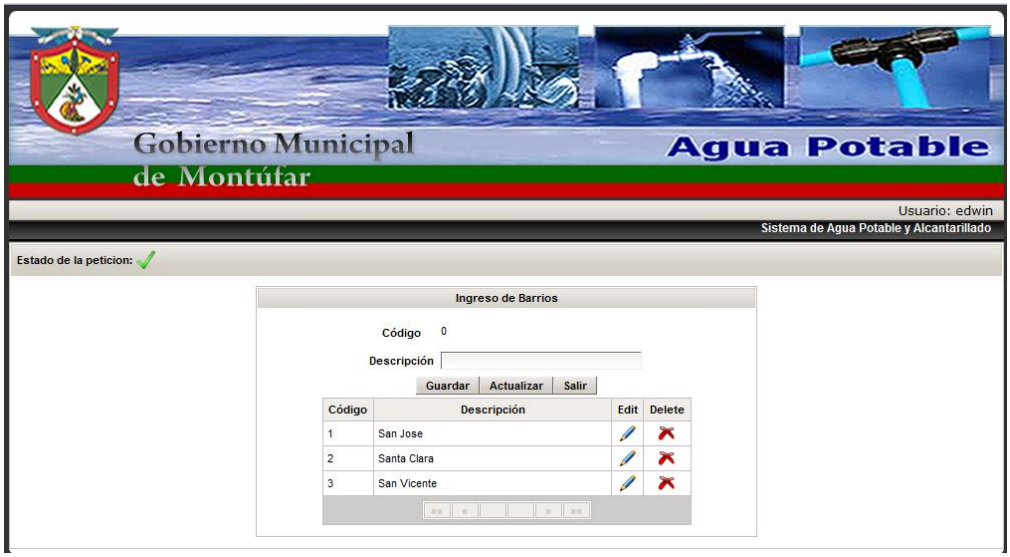
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

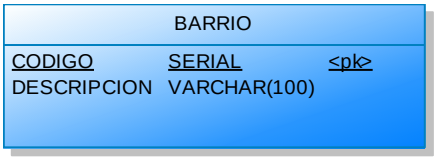
- El caso de uso describe el registro de los barrios que existen en el Cantón Montúfar

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los barrios del Cantón Montúfar



- El sistema almacena la información registrada por el usuario.



5.4.1.1.4. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE MARCAS DE MEDIDORES

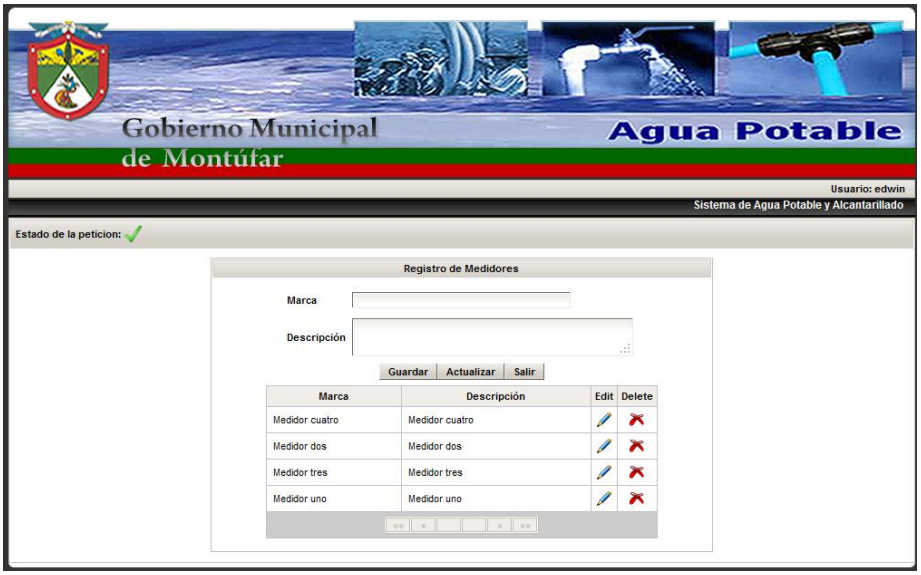
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

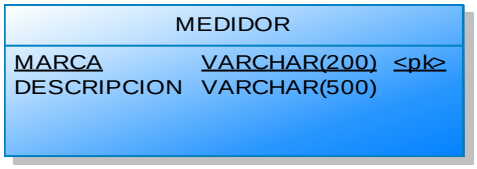
- El caso de uso describe el registro de las marcas de los medidores que existen en el Cantón Montúfar

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar la marca de los medidores.



- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

  En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error

5.4.1.1.5. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE CONTRATOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

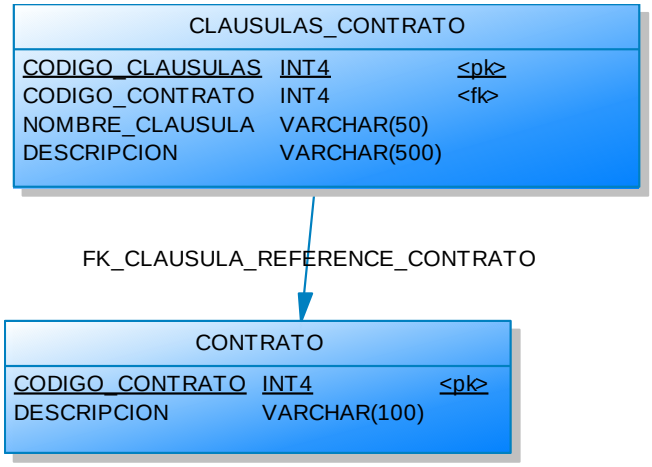
- El caso de uso describe el registro del contrato con sus respectivas clausulas.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar el contrato con sus respectivas clausulas, las cuales se regirán los clientes del sistema.

- Registro de las cláusulas del contrato con los siguientes ítems:
  - Clausula
  - Descripción de la Clausula

- El sistema almacena la información registrada por el usuario.



5.4.1.1.6. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE PARÁMETROS DE CUENTA

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el ingreso de los parámetros que son necesarios en la cuenta.

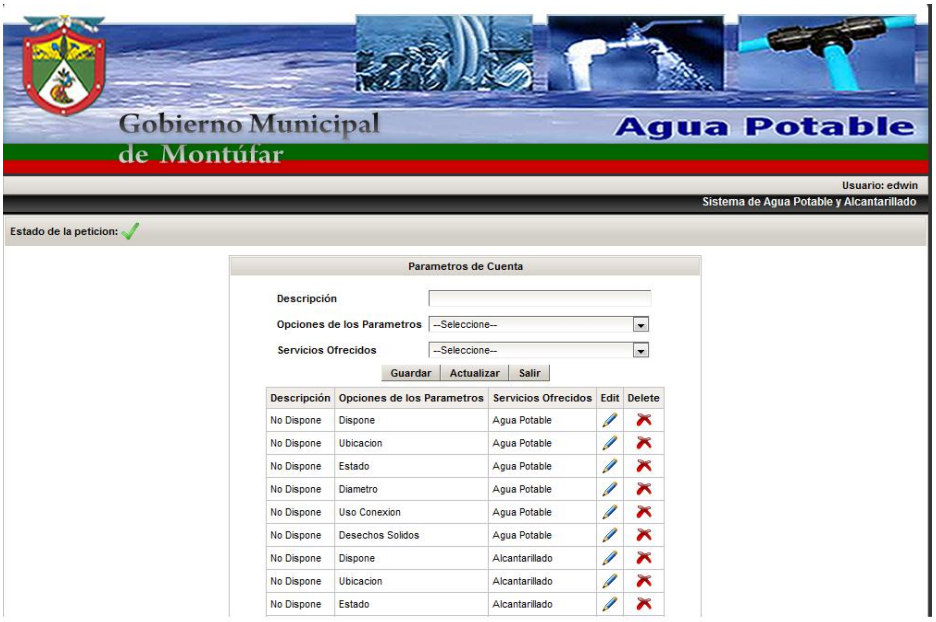
FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los parámetros que son necesarios para crear la nueva cuenta en el sistema.

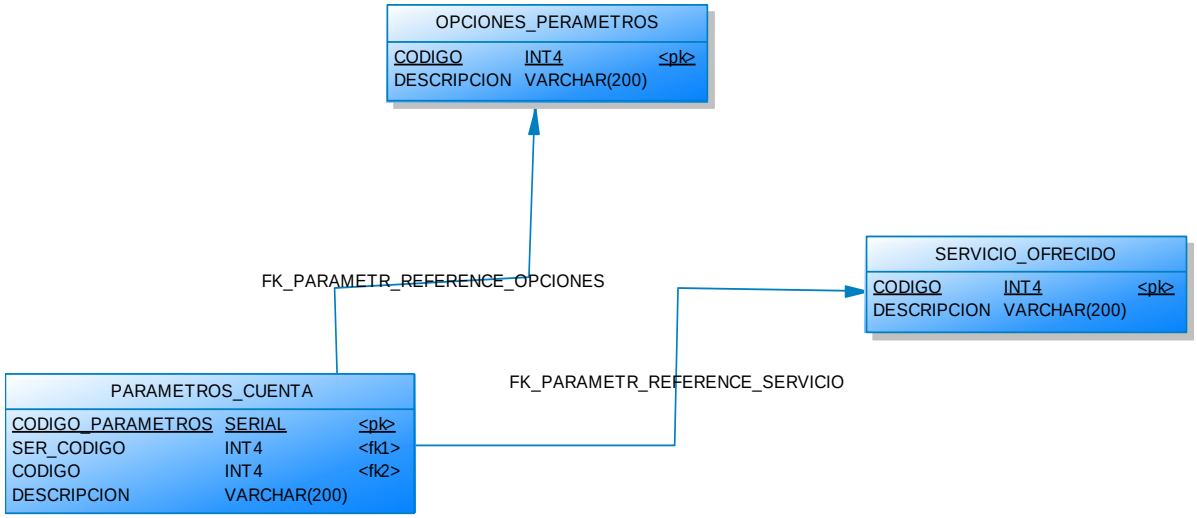
  - Pantalla de servicios ofrecidos

  - Pantalla de Opciones de parámetros

  - Pantalla de parámetros de la cuenta



- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

  En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la infurción.

5.4.1.1.7. ESPECIFICACIÓN CASO DE USO REGISTRO DE CUENTA.

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el registro de la nueva cuenta en el sistema

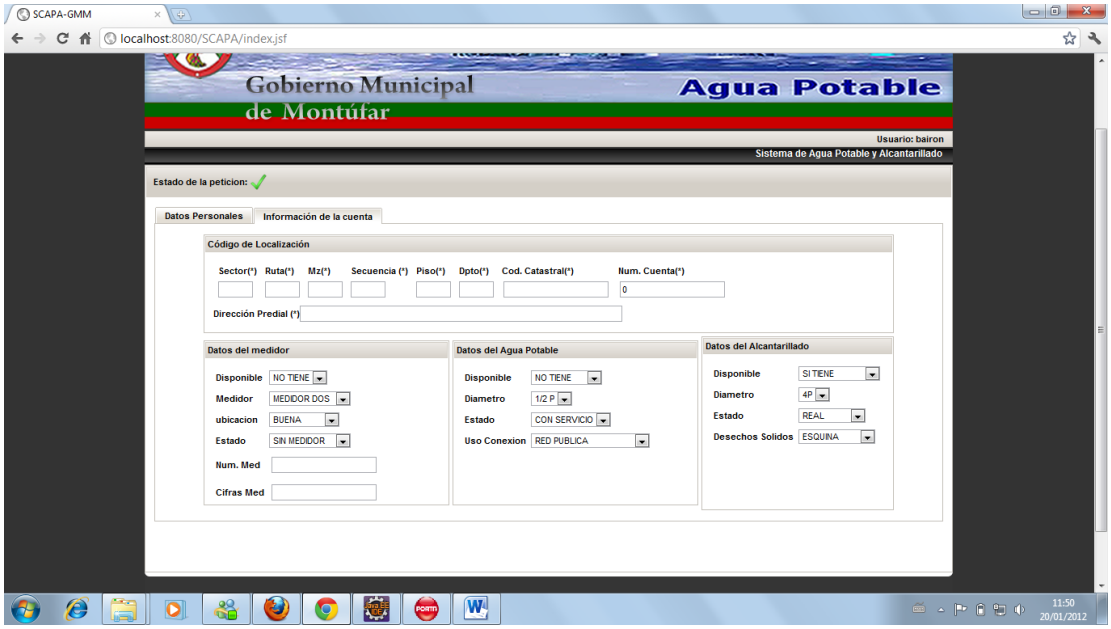
FLUJO BÁSICO DE EVENTOS

- El usuario del sistema se encargara de registrar los datos requeridos para la nueva cuenta

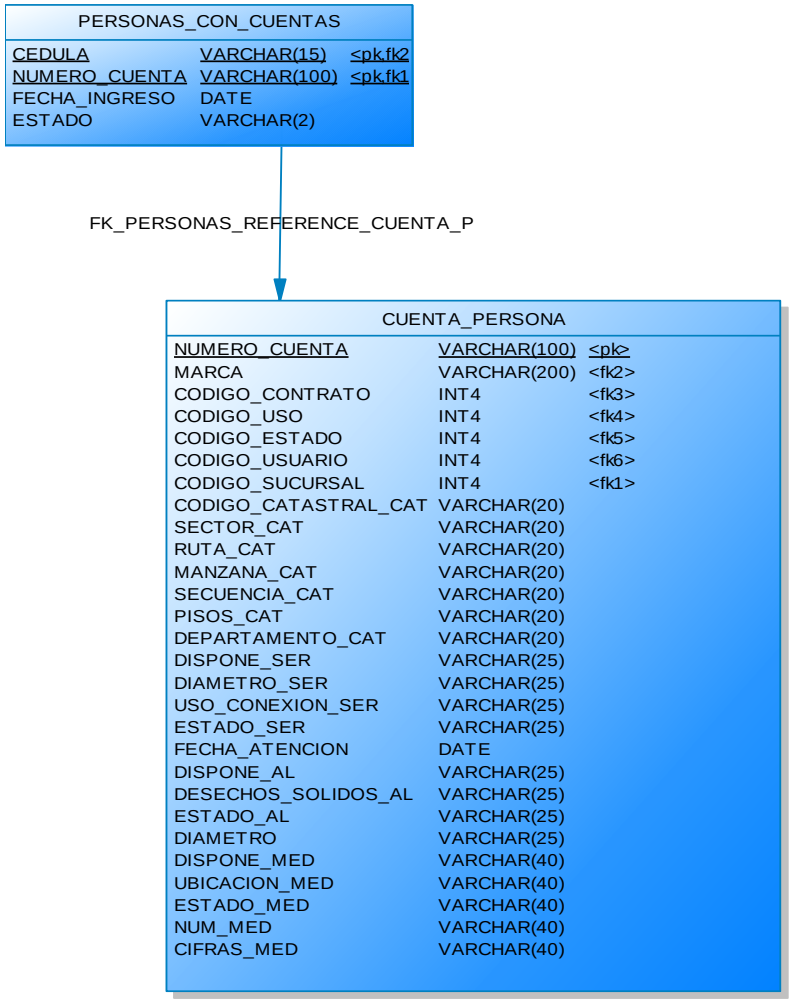
  - Pantalla número uno

  - Pantalla número dos

    Donde ingresamos los datos catastrales de la cuenta, con el fin de identificar el predio, además mediante el ingreso de la ruta se genera automáticamente el número de cuenta, si la ruta es nueva el sistema inicia desde 1 caso contrario busca el último número de cuenta de esa ruta.



- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

  En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la infurción.

5.4.1.2. INGRESO DE LECTURAS

5.4.1.2.1. ESPECIFICACIÓN DEL CASO DE USO TIPOS DE TARIFAS

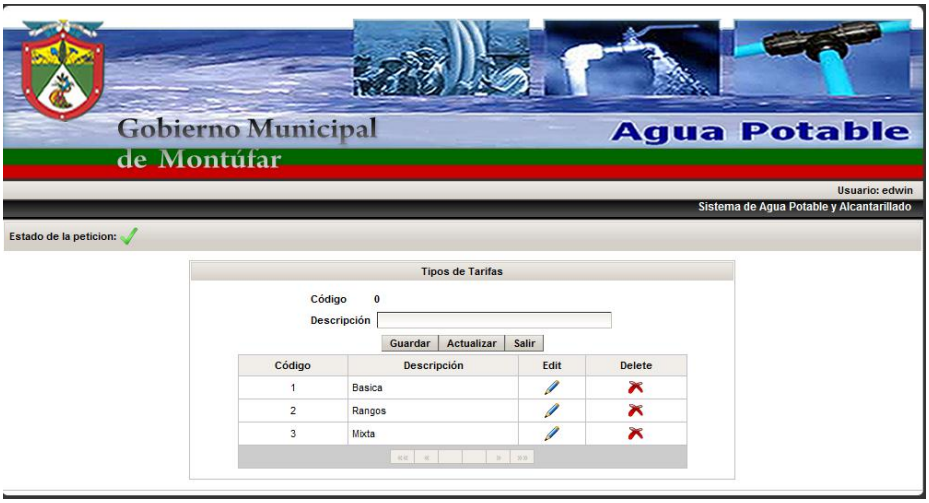
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el registro de loa tipos de tarifas que pueden existir

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los tipos de las tarifas que existan



- El sistema almacena la información registrada por el usuario.

TIPO_TARIFA		
CODIGO_TIPO_TARIFA	INT4	<pk>
DESCRIPCION	VARCHAR(100)	

.

5.4.1.2.2. ESPECIFICACIÓN CASO DE USO TARIFAS.

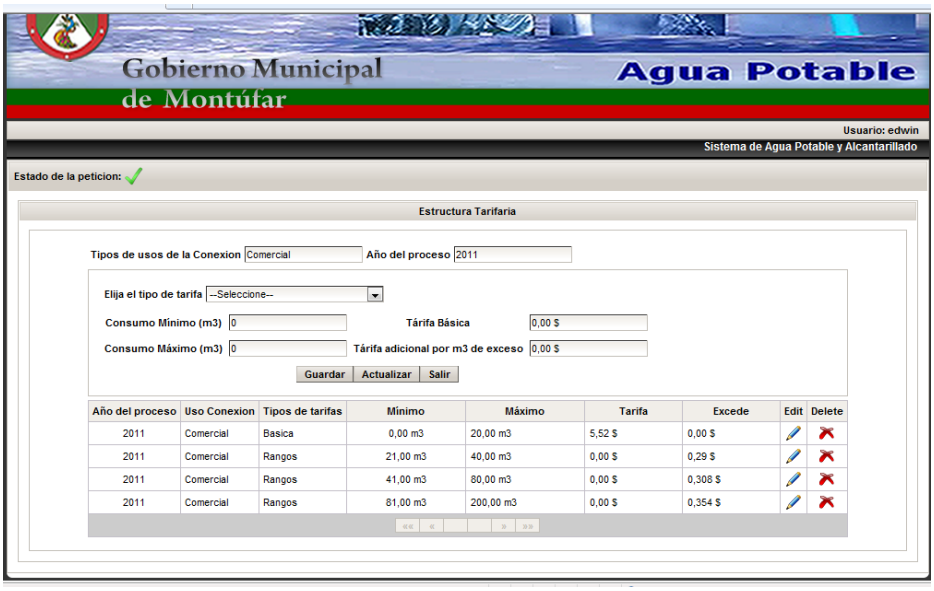
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

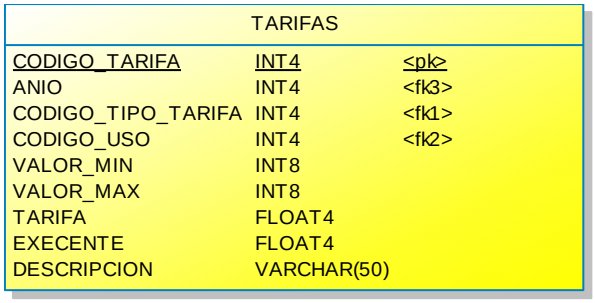
- El caso de uso describe el registro de las tarifas del sistema

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encarga de registrar las tarifas utilizadas para la generación de la carta del agua.



- El sistema almacena la información registrada por el usuario.



5.4.1.2.3. ESPECIFICACIÓN DEL CASO DE USO CONFIGURACIÓN PARÁMETROS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el registro de los parámetros de la configuración de la cuanta que son utilizados principalmente para la generación de la cartas.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encarga de registrar los parámetros de configuración necesarios para la cuenta, estos parámetros son necesarios para el funcionamiento y cálculos del sistema.

- El sistema almacena la información registrada por el usuario.

CONFIG_PARAMETROS		
ANIO	INT4	<pk>
PORCENTAJE_CRITICO	FLOAT8	
FECHA	DATE	
COMETARIO	VARCHAR(50)	
VALOR_ADMIN	FLOAT8	
OTROS_VALOR	FLOAT8	
VALOR_PROCESO	FLOAT8	
CONEXION	FLOAT8	
RECONEXION	FLOAT8	
MULTAS	FLOAT8	
MESES_CORTE	INT4	
MESES_MORA	INT4	
PORCENTAJE_AL	FLOAT8	
POCENTAJE_BAS	FLOAT8	
MESES_PROMEDIO	INT4	

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En caso de que trate de registrar el mismo código de parámetros de configuración, el sistema emitirá un mensaje de error de que el código que en este caso es el año ya está ingreso.

5.4.1.2.4. ESPECIFICACIÓN DEL CASO DE USO INTERÉS MENSUAL

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

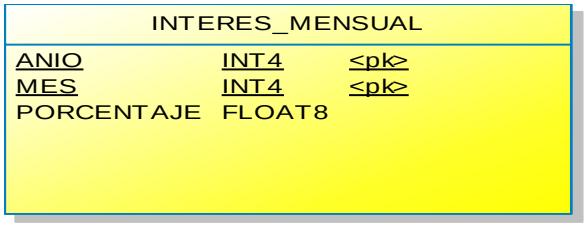
DESCRIPCIÓN BREVE

- El caso de uso describe el registro de la tasa de interés por mora según el mes que se encuentre.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar la tasa de interés por mora, la cual se entrega o se saca de las publicaciones que realiza el estado

- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En caso de suplicación el sistema no dejara ingresar un nuevo y emitirá un mensaje de error.

5.4.1.2.5. ESPECIFICACIÓN DEL CASO DE USO INGRESO DE LECTURAS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

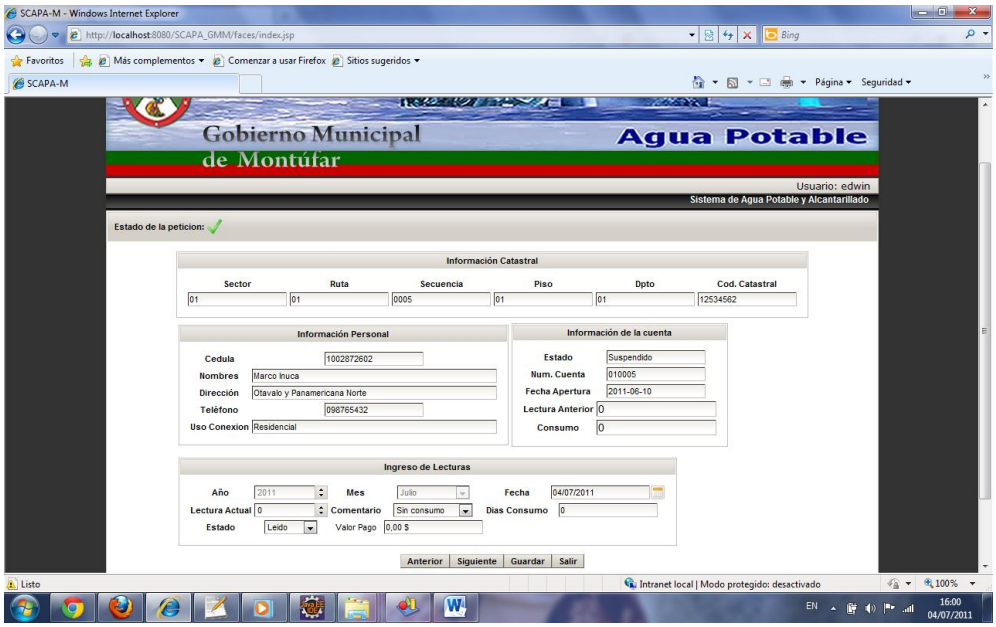
DESCRIPCIÓN BREVE

- El caso de uso describe el registro la lectura por cada uno de los clientes del servicio.

FLUJO BÁSICO DE EVENTOS

- El operador o encargado de buscar según las diferentes opciones existentes

- Luego de buscar la o las cuantas a las cuales les vamos a ingresar su respectiva lectura mensual presionamos clic en aceptar para que el sistema genera la forma de ingreso



- Una vez ingresada la lectura guardamos y el sistema automáticamente genera la próxima cuenta.
- El sistema almacena la información registrada por el usuario.

LECTURAS		
NUMERO_CUENTA	VARCHAR(100)	<pk,fk2>
ANIO	VARCHAR(10)	<pk>
MES	VARCHAR(10)	<pk>
CODIGO_USO	INT4	<fk1>
CODIGO_USUARIO	INT4	<fk3>
FECHA_LECTURA	DATE	
VALOR_LECTURA	INT4	
COMENTARIO	VARCHAR(100)	
CONSUMO	INT4	
DIAS_CONSUMO	INT4	
ESTADO_LECTURA	INT4	
LECTURA_ANTERIOR	INT4	

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En caso de suplicación el sistema no dejara ingresar un nuevo y emitirá un mensaje de error.

5.4.1.2.6. ESPECIFICACIÓN DEL CASO DE USO GENERAR CARTAS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
04/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el momento que se ingresa la lectura el sistema internamente lanza un proceso el cual crea la carta y genera los valores correspondientes, como al finalizar llama a otro proceso el cual actualiza las cartas que no han sido canceladas.

FLUJO BÁSICO DE EVENTOS

- El operador o el personal a cargo del ingreso de las lecturas realiza e ingreso.
- La lectura al momento de seleccionar el tipo de ingreso automáticamente realiza el proceso de generar la carta y el costo de la misma
- Después al momento de realizar el registro de la lectura en la base después se realiza el ingreso de la carta a la tabla pertinente.

- Después de registrar la carta, existe un proceso interno que realiza un chequeo si el estado de pago y si se encuentra en estado de cortad se cambia automáticamente.
- Una vez realizado y para finalizar existe un proceso el cual realiza una actualización de las cartas deudoras para calcular el interés de mora según la tasa de interés ingresada.
- Tabla donde se realiza el ingreso

CARTAS		
NUMERO_CUENTA	VARCHAR(100)	<pk,fk1>
ANIO	VARCHAR(10)	<pk,fk1>
MES	VARCHAR(10)	<pk,fk1>
CODIGO_USO	INT4	<fk3>
CON_ANIO	INT4	<fk2>
CONSUMO	FLOAT8	
VALOR_CONSUMO	FLOAT8	
VALOR_BASURA	FLOAT8	
VALOR_ALCANTARILLADO	FLOAT8	
VALOR_ADMIN	FLOAT8	
VALOR_OTROS	FLOAT8	
PROCESO	FLOAT8	
VALOR_CONEXION	FLOAT8	
VALOR_RECONEXION	FLOAT8	
VALOR_MULTAS	FLOAT8	
VALOR_VARIOS	FLOAT8	
INTERES_MORA	FLOAT8	
INTERES	FLOAT8	
TOTAL_LIQUIDO	FLOAT8	
TOTAL_FACTURA	FLOAT8	
FECHA_LIQUIDACION	DATE	
FECHA_FACTURACION	DATE	
FECHA_PROCESO	DATE	
ESTADO_LIQUIDACION	INT4	
VALOR_IVA	FLOAT8	
VALOR_DESCUENTO	FLOAT8	
COMPRA_MEDIDOR	VARCHAR(2)	
COMPRA_VARIOS	VARCHAR(2)	
CODIGO_TRANASCCION	INT4	

FLUJOS ALTERNATIVOS

- En el caso de existir algún error en este proceso el sistema emitirá un mensaje de erros y ser reversara toda la transacción.

5.4.1.2.7. ESPECIFICACIÓN DEL CASO DE USO ACTUALIZACIÓN LECTURAS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
03/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe cuando el usuario administrador o personal encargado de actualizar la lectura si por alguna existe algún inesperado error.

FLUJO BÁSICO DE EVENTOS

- Búsqueda de la cuenta que va a actualizar la lectura

- El administrador del sistema se encargara de registrar cada una de las lecturas o las lecturas que se encuentren mal ingresadas.
- Aquí se realiza una búsqueda del mes que deseen actualizas e ingresa la nueva lectura.

- La tabla que registra las actualizaciones es:

AUTORIZACION_CAMBIO		
CODIGO_AUTORIZACION	INT4	<pk>
NUMERO_CUENTA	VARCHAR(100)	<fk>
ANIO	VARCHAR(10)	<fk>
MES	VARCHAR(10)	<fk>
FECHA_CAMBIO	DATE	
VALOR_ANTERIOR	INT4	
VALOR_ACTUAL	INT4	
DESCRIPCION	VARCHAR(200)	

5.4.2. MANEJO DE VENTANILLAS

5.4.2.1. GESTIÓN DE CAJAS

5.4.2.1.1. ESPECIFICACIÓN DEL CASO DE USO FICHA MUNICIPAL

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
20/08/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el proceso de registro de la información del Gobierno Municipal de Montúfar, la cual es requerida por el sistema.

FLUJO BÁSICO DE EVENTOS

- El usuario ingresa al sistema y en el menú busca el ítem que dice ingresar ficha municipal.
- El usuario ingresa los datos del Gobierno Municipal de Montúfar que el sistema necesita para el funcionamiento.
    - El usuario ingresara los siguiente información:
        - Ruc
        - Descripción
        - Teléfono
        - Fax
        - IVA
        - Email

- El sistema almacena la información registrada por el usuario la información es registrada en la siguiente tabla.

FICHA_MUNICIPAL		
RUC	VARCHAR(50)	<pk>
DESCRIPCION	VARCHAR(200)	
DIRECCION	VARCHAR(100)	
TELEFONO	VARCHAR(20)	
FAX	VARCHAR(20)	
EMAIL	VARCHAR(100)	
IVA	FLOAT8	
DESCUENTO	FLOAT8	

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**
  En el caso de ingresar un RUC ya existente, el sistema dará un mensaje de error anunciando su existencia

5.4.2.1.2. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE SUCURSALES

Fecha	Versión	Descripción	Autor
27/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
20/09/2011	<1.0>	Revisión del caso de Uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el proceso a seguir para el registro de las sucursales que puede tener el Gobierno Municipal de Montufar para efecto de la recaudación por la prestación del servicio de Agua Potable

FLUJO BÁSICO DE EVENTOS

- El usuario Administrador se encarga de registra los parámetros requeridos para crear una nueva sucursal
  - Se ingresan los siguientes campos:
  - Descripción.
  - Teléfono.
  - Dirección.

- El sistema almacena la información registrada por el usuario

SUCURSAL		
CODIGO_SUCURSAL	INT4	<pk>
RUC	VARCHAR(50)	<fk>
DESCRIPCION	VARCHAR(100)	
TELEFONO	VARCHAR(20)	
DIRECCION	VARCHAR(500)	

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

No existirá la duplicación del código debido a que se genera automáticamente desde la base de datos.

5.4.2.1.3. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE CAJAS

Fecha	Versión	Descripción	Autor
27/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
20/09/2011	<1.0>	Revisión del caso de Uso	Edwin Madruñero

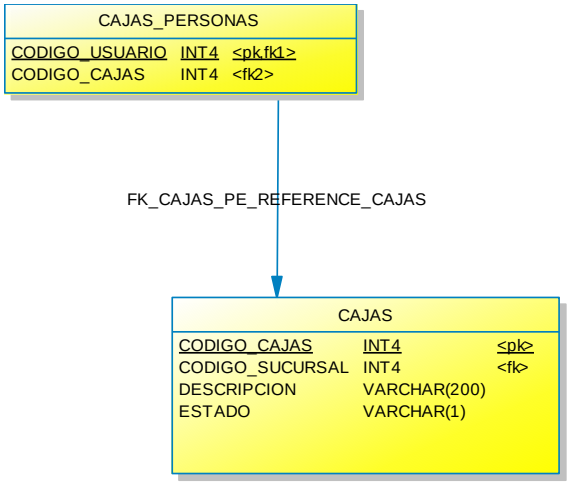
DESCRIPCIÓN BREVE

- El caso de uso describe el proceso de registro de las cajas que se encuentran en cada una de las sucursales.

FLUJO BÁSICO DE EVENTOS

- El usuario Administrador se encargara de registra la información con los siguientes campos:
  a. Código
  b. Descripción
  c. Seleccionamos al encargado.
  d. Seleccionamos la sucursal.
  e. Estado de la caja.

- El sistema almacena la información registrada por el usuario



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

No existirá la duplicación del código debido a que se genera automáticamente desde la base de datos.

5.4.2.1.4. ESPECIFICACIÓN DEL CASO DE USO CIERRE DE CAJA

Fecha	Versión	Descripción	Autor
27/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
06/11/2011	<1.0>	Revisión del caso de Uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el proceso que debe realizar el proceso de cierre de caja y la generación del asiento contable

FLUJO BÁSICO DE EVENTOS

- El usuario ingresará a la pantalla de cierre de caja, en esta parte seleccionamos el usuario encargado del cobro y la fecha de cierre.

- Este proceso luego de haberse realizado se guarda en una tabla de la base de datos como es:

CIERRE_CAJA		
FECHA_CIERRE	DATE	<pk>
CODIGO_CAJAS	INT4	<fk2>
CODIGO_USUARIO	INT4	<fk1>
CONSUMO_TOTAL	FLOAT8	
VALOR_CONSUMO	FLOAT8	
VALOR_BASURA	FLOAT8	
VALOR_ALACANTARILLADO	FLOAT8	
VALOR_ADMIN	FLOAT8	
VALOR_OTROS	FLOAT8	
VALOR_PROCESO	FLOAT8	
VALOR_CONEXION	FLOAT8	
VALOR_RECONEXION	FLOAT8	
VALOR_MILTAS	FLOAT8	
VALOR_VARIOS	FLOAT8	
INTERES_MORA	FLOAT8	
TOTAL_LIQUIDO	FLOAT8	
VALOR_VENTA_MED	FLOAT8	
VALOR_VENTA_VARIOS	FLOAT8	

5.4.2.1.5. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE DEPÓSITO

Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el ingreso del depósito diario de lo recaudado el día anterior en la caja del Departamento de Agua Potable

FLUJO BÁSICO DE EVENTOS

- Ingresar al sistema y luego buscamos en el menú la pestaña que nos presenta la pantalla para registrar el depósito bancario del cierre de cada día

- Almacena en la tabla de la base de datos.

DEPOSITO_BANCARIO		
CODIGO_DEPOSITO	INT4	<pk>
FECHA_CIERRE	DATE	<fk>
FECHA_DEPOSITO	DATE	
ENTIDAD_FINANCIERA	VARCHAR(4000)	
NUMERO_CUENTA	VARCHAR(100)	
VALOR_DEPOSITO	FLOAT8	
OBSERVACION	VARCHAR(4000)	

5.4.3. GESTIÓN DE CARTERA Y MOROSIDAD

5.4.3.1. GESTIÓN DE MOROSIDAD

5.4.3.1.1. ESPECIFICACIÓN DEL CASO DE USO CORTES

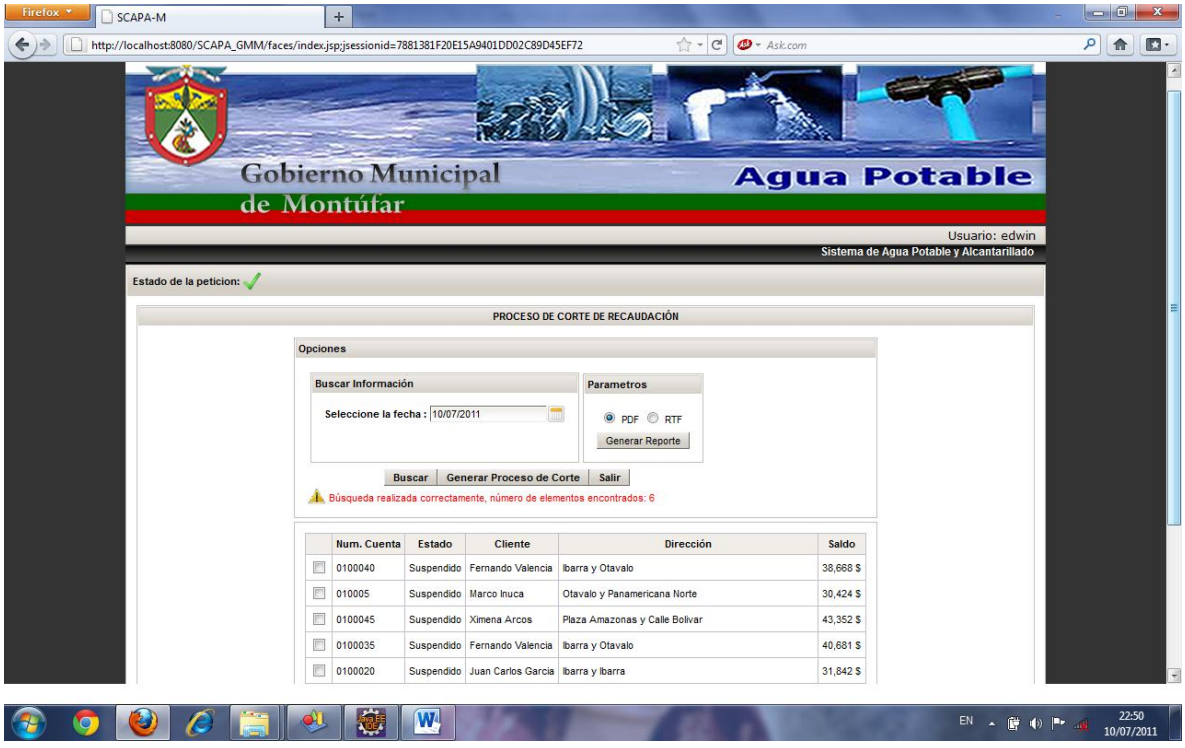
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
10/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe cuando una cuenta se ha demorado sin realizar el pago, un tiempo determinado por las autoridades competentes

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema ingresa al sistema e inicia el proceso de cálculo y realización de corte, en ese instante la cuenta asume un estado de preparado para el corte.



- Una vez generado el proceso inicial del corte la cuanta asume un estado de suspendido o listo para el corte, de esta lista se genera un reporte y con la comprobación de las cuantas se accede al recurso y realiza el corte de la cuenta

5.4.3.1.2. ESPECIFICACIÓN DEL CASO DE USO TIPOS DE NOTIFICACIONES

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
12/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el registro de los tipos de notificaciones que pueden existir.

FLUJO BÁSICO DE EVENTOS

- El administrador se encargara de registrar los tipos de notificaciones en la siguiente forma.

- El sistema almacena la información registrada por el usuario.

TIPO_NOTIFICACION		
CODIGO	INT4	<pk>
DESCRIPCION	VARCHAR(100)	
NUMERO_MESES	INT4	

FLUJOS ALTERNATIVOS

- **CASO DE REPETIR INFORMACIÓN.**

La información no se repetirá debido a que estamos autogenerando el código de la base de datos.

5.4.3.1.3. ESPECIFICACIÓN DEL CASO DE USO GENERAR NOTIFICACIONES

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
14/08/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

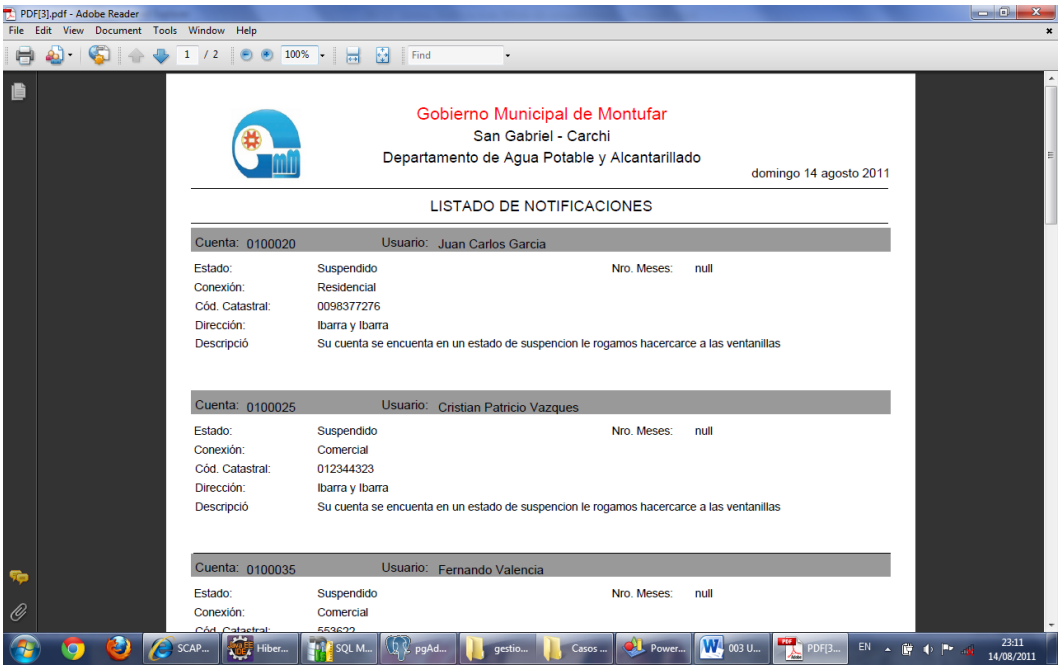
- El caso de uso describe las notificaciones que se encuentren ya en ejecución.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema ingresa a la forma donde realizara la búsqueda de las notificaciones activas y genera el reporte

- Aquí se visualizan las notificaciones en estado activo, para obtener la información lista para imprimir pulsamos el respectivo botón

- Seleccionamos en tipo de reporte que queremos y generamos.



- El sistema almacena la información registrada por el usuario.

NOTIFICACIONES		
CODIGO_NOTIFICACION	INT4	<pk>
CODIGO	INT4	<fk2>
CEDULA	VARCHAR(15)	<fk1>
NUMERO_CUENTA	VARCHAR(100)	<fk1>
FECHA_NOTIFICACION	DATE	
FECHA_IMPRESION	DATE	
ESTADO_IMPRESION	VARCHAR(2)	
MESES_DEUDA	INT4	

FLUJOS ALTERNATIVOS

- **CASO DE REPETIR INFORMACIÓN.**

La información no se repetirá debido a que estamos autogenerando el código de la base de datos.

5.4.4. SEGURIDAD Y VALIDACIÓN

5.4.4.1. GESTIÓN DE USUARIOS

5.4.4.1.1. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE USUARIOS

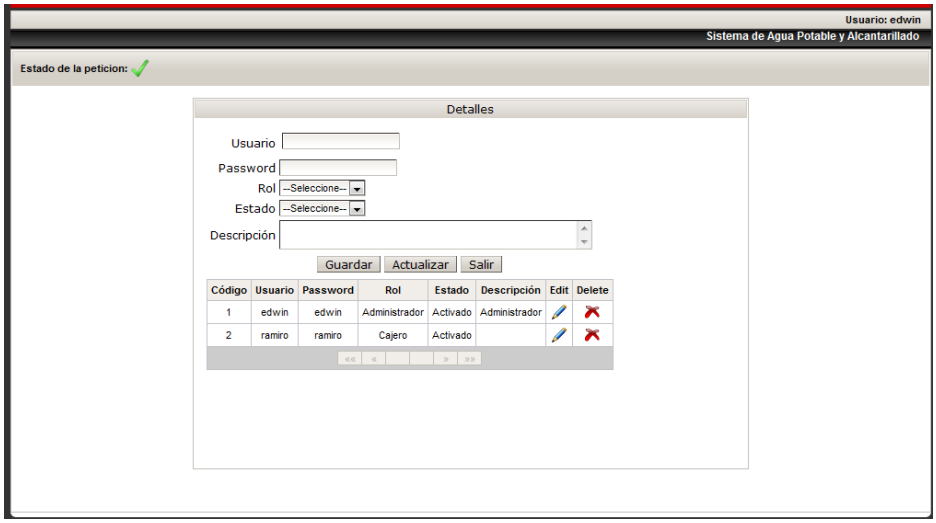
Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
26/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe la forma de cómo registrar los datos del usuario del sistema

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar el usuario a el usuario para el sistema como también su rol, el cual se lo utilizara para ingresar al sistema



- La información se registrara en la siguiente tabla de la base de datos

5.4.4.1.2. ESPECIFICACIÓN DEL CASO DE USO CAMBIO DE CLAVE DE ACCESO

Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/09/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el registro la actualización de las claves de acceso del usuario al momento de iniciar la primera vez

FLUJO BÁSICO DE EVENTOS

- Una vez registrada la clave por parte del administrado del sistema, el usuario al momento de iniciar por primera vez la sesión le perdida que cambie la clave de acceso obligatoriamente

- El sistema actualizara la clave de acceso en la base de datos.

USUARIOS		
CODIGO_USUARIO	INT4	<pk>
CODIGO_ESTADO	INT4	<fk2>
CODIGO_ROL	INT4	<fk3>
CEDULA	VARCHAR(15)	<fk1>
USUARIO	VARCHAR(50)	
CLAVE	VARCHAR(100)	
DESCRIPCION	VARCHAR(100)	
IMPORTANCIA	VARCHAR(2)	

FLUJOS ALTERNATIVOS

- Si no actualiza la clave de acceso el sistema con continuar a la página de administración administrador

5.4.4.1.3. ESPECIFICACIÓN DEL CASO DE USO ESTADOS DE UN USUARIO

Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
26/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

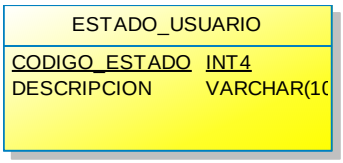
DESCRIPCIÓN BREVE

- El caso de uso describe la forma de cómo registrar los estados de los usuarios.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los estados de los usuarios que va a tener el sistema

- La información se registrara en la siguiente tabla de la base de datos



5.4.4.1.4. ESPECIFICACIÓN DEL CASO DE USO ROLES DE UN USUARIO

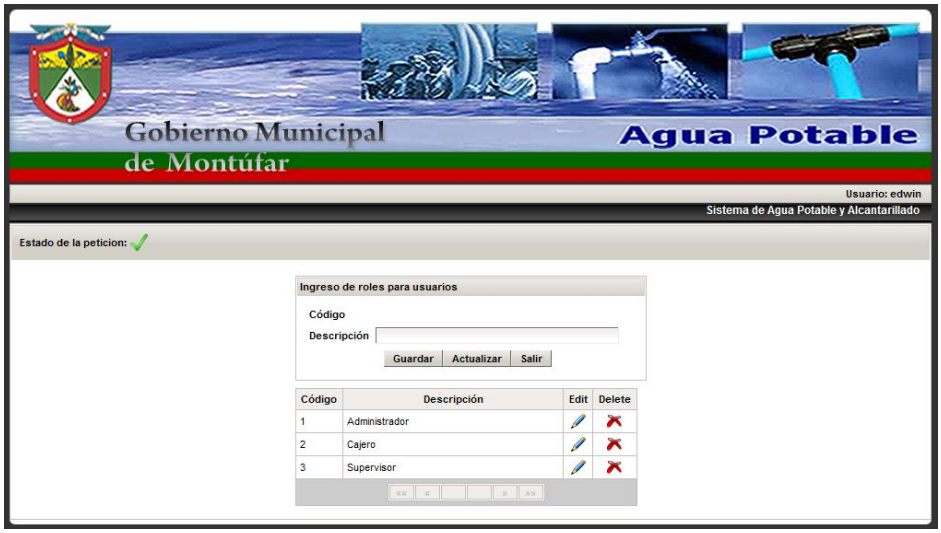
Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
26/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe la forma de cómo registrar el rol que utilizara cada uno de los usuarios.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los roles para los usuarios que va a tener el sistema



- La información se registrara en la siguiente tabla de la base de datos

ROL_USUARIOS		
CODIGO_ROL	INT4	<pk>
DESCRIPCION	VARCHAR(100)	
IMPORTANCIA	INT2	

FLUJOS ALTERNATIVOS

- En el caso de encontrar un que se registre la misma clave el sistema emitirá un mensaje de error indicando el percance.

5.4.4.1.5. ESPECIFICACIÓN DEL CASO DE USO AUDITORIA DE INGRESO DEL USUARIOS

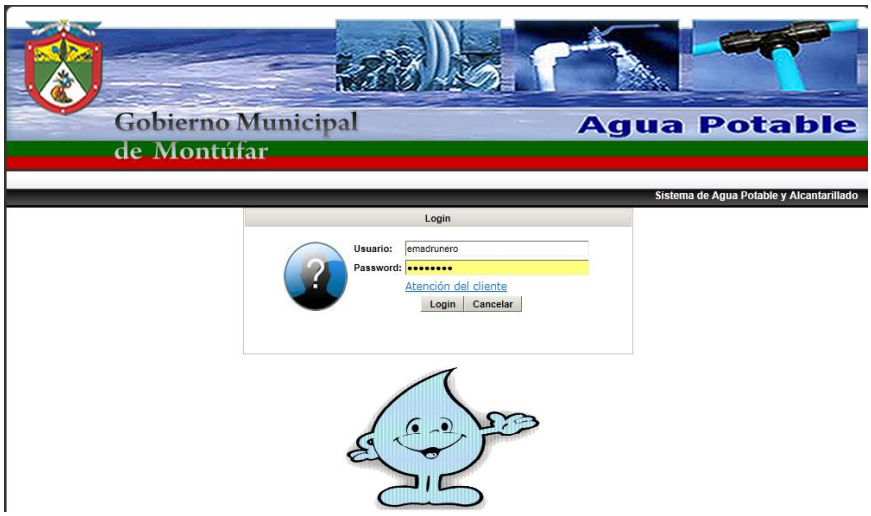
Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe cuando un usuario ingresa al sistema este caso de uso registra ese evento como un control de auditoría

FLUJO BÁSICO DE EVENTOS

- El usuario ingreso su usuario y contraseña para proceder al sistema



- Si el usuario ingresa su usuario y contraseña correctamente nuestra tabla de registro de ingreso del usuario ingresara los correspondiente datos

AUDITORIA_USUARIOS		
CODIGO_INGRESO	INT4	<pk>
CODIGO_USUARIO	INT4	<fk>
HORA_ENTRADA	TIME	
HORA_SALIDA	TIME	
FECHA	FECHA	
HOST	VARCHAR(200)	

5.4.4.1.6. ESPECIFICACIÓN DEL CASO DE USO AUDITORÍA DE TRANSACCIONES

Fecha	Versión	Descripción	Autor
05/09/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe la acción que realiza varios disparadores creados en la base de datos para un fin determinado, cuando se realiza una actualización o una eliminación esta registrara en la tabla correspondiente para que el auditor pueda visualizar todas

las transacciones realizadas

FLUJO BÁSICO DE EVENTOS

- La actualización y la eliminación de las 4 tablas que hemos considerado más sensibles se registran en una tabla de auditoría y el nombre de los trigges son:
    i. FUN_AUDITAR_CARTAS
    ii. FUN_AUDITORIA_LECTURAS

- Si el usuario ingresa su usuario y contraseña correctamente nuestra tabla de registro de ingreso del usuario ingresara los correspondiente datos

AUDITORIA_TRANSACCIONES		
CODIGO_TRANSACCION	SERIAL	<pk>
CODIGO_USUARIO	INT4	<fk>
NOMBTABLA	VARCHAR(150)	
ESQUEMA	VARCHAR(150)	
OPERACION	VARCHAR(200)	
TRANSACCION_ANTERIOR	TEXT	
TRANSACCION_ACTUAL	TEXT	
FECHA_HORA	TIMESTAMP WITH TIME ZONE	

5.4.5. VENTAS

5.4.5.1. ESPECIFICACIÓN DEL CASO DE USO FORMAS DE PAGO

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

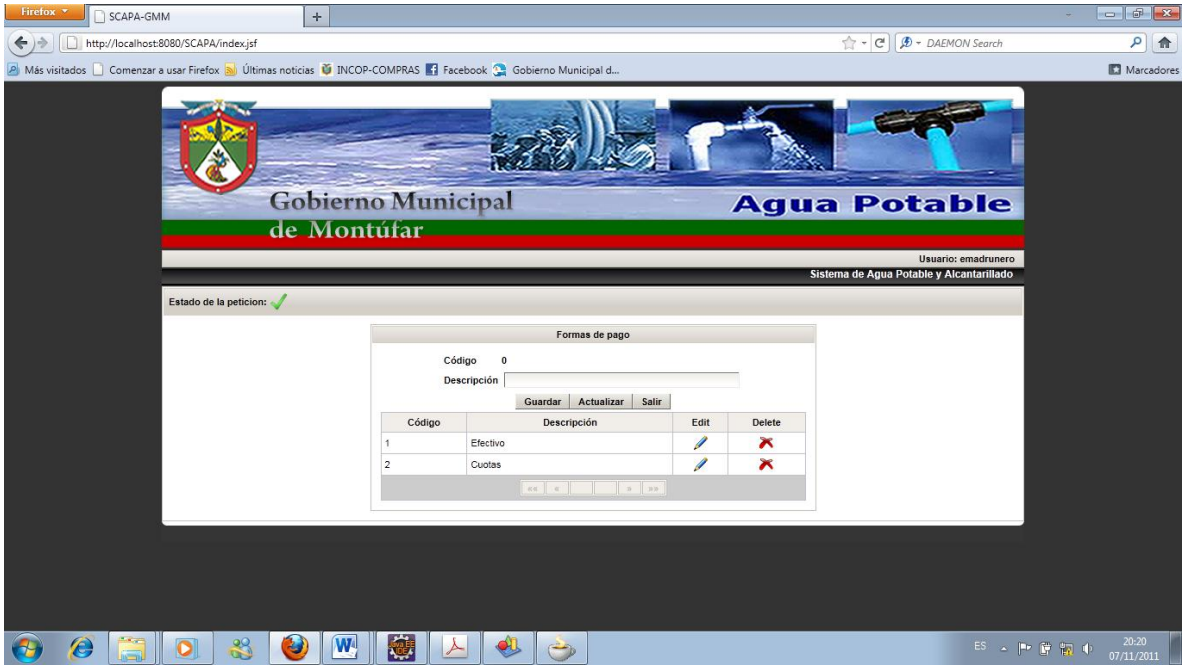
DESCRIPCIÓN BREVE

- El caso de uso describe las formas de pago que puede realizar un cliente al realizar la compra de algún utilitario

FLUJO BÁSICO DE EVENTOS

- El administrador registra los dos tipos de pagos que existen

- ✓ Efectivo
- ✓ Cuotas.



- ➢ El sistema almacena la información registrada por el usuario.

FORMA_PAGO		
CODIGO_FORMA	INT4	<pk>
DESCRIPCION	VARCHAR(200)	

5.4.5.2. ESPECIFICACIÓN DEL CASO DE USO VENTAS DE MEDIDOR

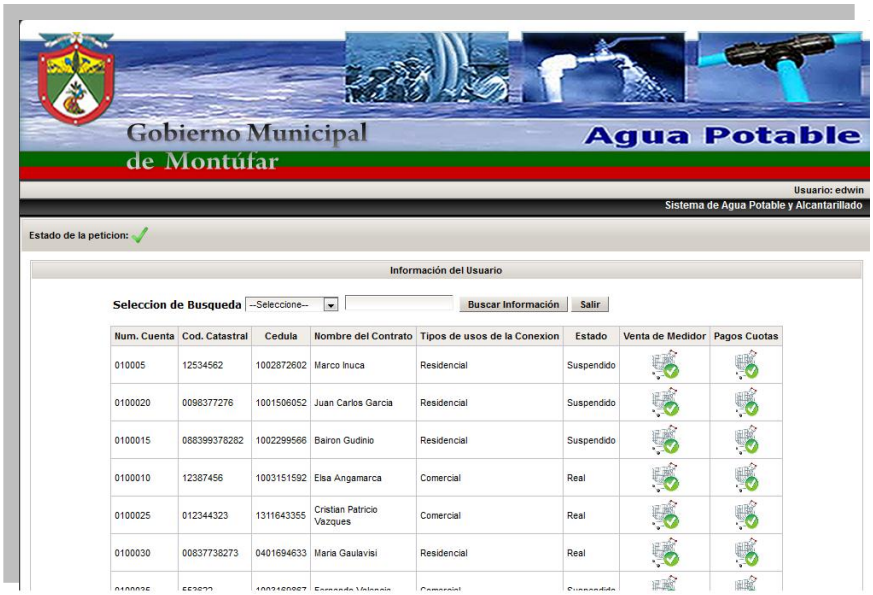
Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
06/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe la venta de un medidor con todos sus implementos, los cuales se realizan el momento de que se hace la apertura una nueva cuenta.

FLUJO BÁSICO DE EVENTOS

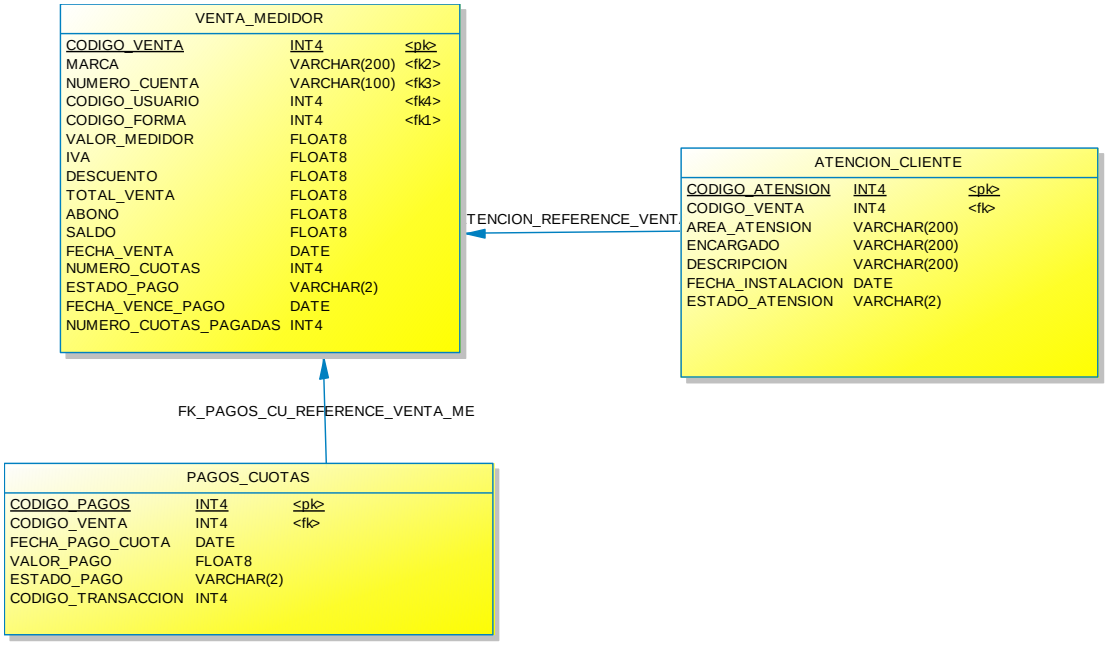
- El administrador o la persona en cargada de registrar la apertura de la nueva cuenta, es quien se encargara de registrar la venta del medidor.
- Como primer paso buscamos la cuanta a la cual ve vamos a vender el medido.



- Se realiza la búsqueda y una vez encontrada la cuenta pulsamos la figura del carrito de compras en la columna que dice Venta de Medidor, se mostrara la siguiente pantalla en la cual se registra la venta.

- Una vez ingresado la información de la pantalla principal automáticamente se genera las cuentas y la información de la atención al cliente.

- El sistema almacena la información registrada por el usuario.



5.4.5.3. ESPECIFICACIÓN DEL CASO DE USO TIPOS DE PRODUCTOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe el ingreso de los tipos de productos que puede vender.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema se encargara de registrar los tipos de productos

- El sistema almacena la información registrada por el usuario.

TIPO_PRODUCTOS		
CODIGO_TIPO	INT4	<pk>
DESCRIPCION	VARCHAR(200)	

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la información

5.4.5.4. ESPECIFICACIÓN DEL CASO DE USO PRODUCTOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
10/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

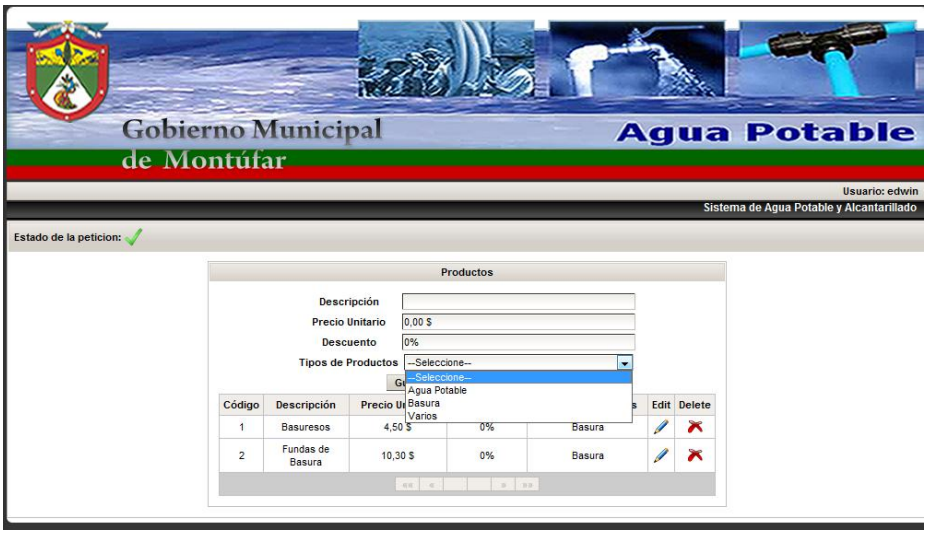
DESCRIPCIÓN BREVE

- El caso de uso describe el registro de los diferentes productos que se realizara la venta.

FLUJO BÁSICO DE EVENTOS

- El administrador del sistema ingresa los diferentes productos o servicios respectivamente primero seleccionando su categoría.

a) Registramos los parámetros requeridos en la figura.



- El sistema almacena la información registrada por el usuario.

FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la infurción.

- **SELECCIÓN TIPOS DE PRODUCTO**

Si no ha seleccionado la categoría a la cual pertenece el producto, el sistema emitirá un mensaje de error indicando que falta la selección correspondiente.

5.4.5.5. ESPECIFICACIÓN DEL CASO DE USO FACTURA DE PRODUCTOS SIN CUENTA

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
10/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

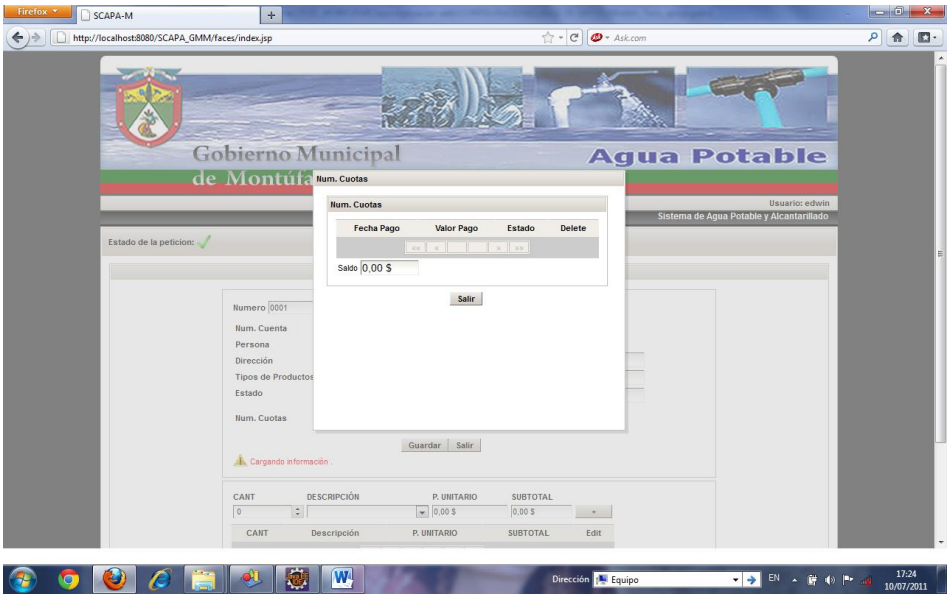
- El caso de uso describe el registro de las ventas de diversos servicios como productos del departamento de agua potable a la ciudadanía en general.

FLUJO BÁSICO DE EVENTOS

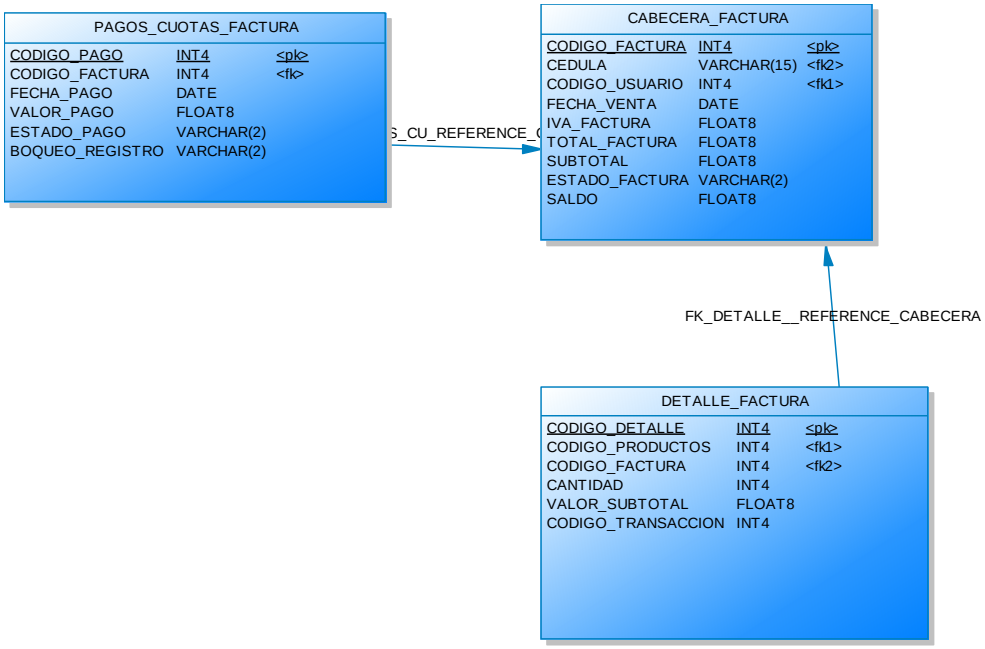
- El administrador o el supervisor se encargara de buscar a la persona en el sistema, cabe señalar que la persona debe estar ingresada en el sistema con anterioridad.

- Una vez encontrada la persona se la selecciona dando clic en la figura, una vez hecho esto el sistema automáticamente encontrara la cuenta de esta persona para la venta a realizar, cabe resaltar que si la persona tiene 2 cuentas al momento de realizar la búsqueda se mostraran las dos.

- Ingresamos los datos requeridos en la pantalla, las cuotas se generan automáticamente al momento de seleccionar las cuotas, las podemos ver dando clic donde dice cuotas



- Una vez ingresada la información presionamos clic en guardar.
- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la infurción.

- **SELECCIÓN TIPOS DE PRODUCTO**

Si no ha seleccionado la categoría a la cual pertenece el producto, el sistema emitirá un mensaje de error indicando que falta la selección correspondiente.

5.4.5.6. ESPECIFICACIÓN DEL CASO DE USO FACTURA DE PRODUCTOS CON CUENTA

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
10/07/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

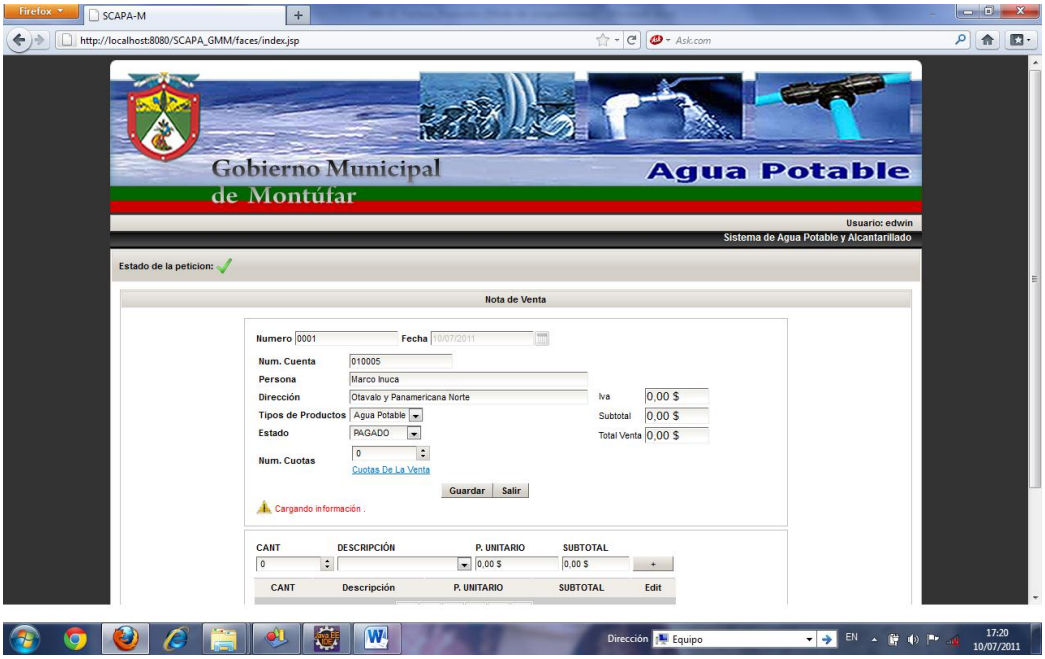
DESCRIPCIÓN BREVE

- El caso de uso describe el registro de las ventas de diversos servicios como productos del departamento de agua potable a la ciudadanía en general.

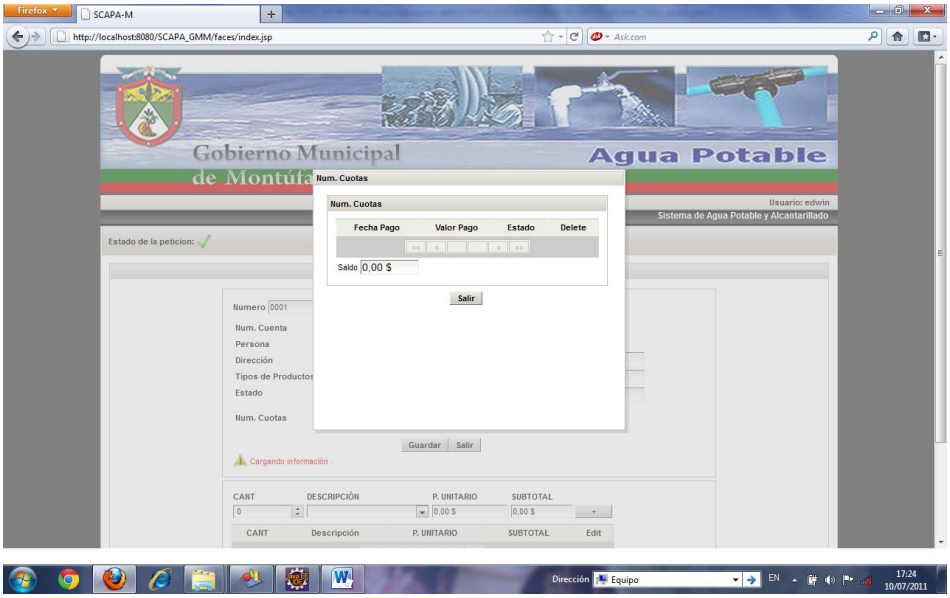
FLUJO BÁSICO DE EVENTOS

- El administrador o el supervisor se encargara de buscar a la persona en el sistema, cabe señalar que la persona debe estar ingresada en el sistema con anterioridad.

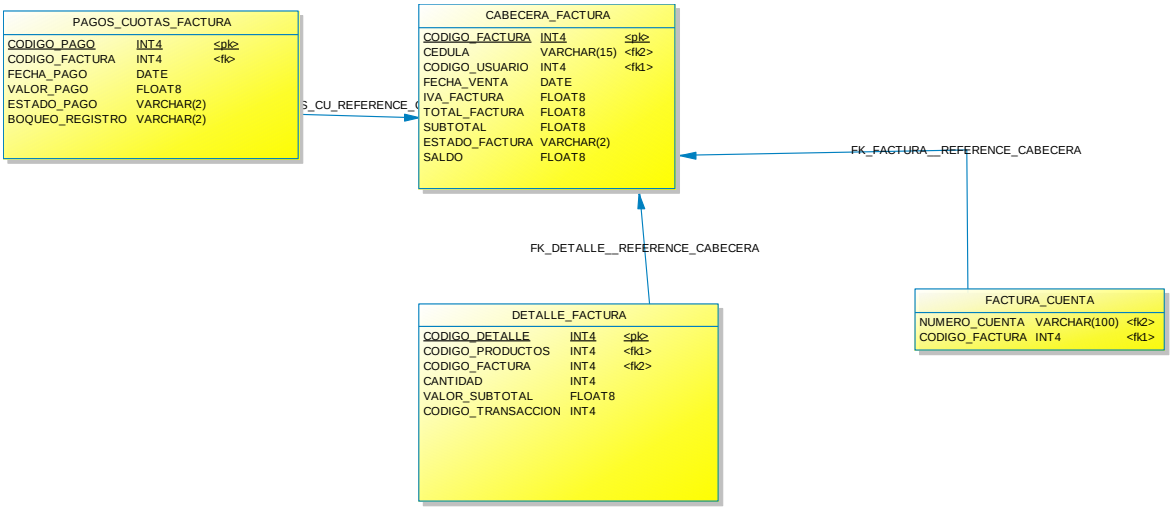
- Una vez encontrada la persona se la selecciona dando clic en la figura, una vez hecho esto el sistema automáticamente encontrara la cuenta de esta persona para la venta a realizar, cabe resaltar que si la persona tiene 2 cuentas al momento de realizar la búsqueda se mostraran las dos.



- Ingresamos los datos requeridos en la pantalla, las cuotas se generan automáticamente al momento de seleccionar las cuotas, las podemos ver dando clic donde dice cuotas



- Una vez ingresada la información presionamos clic en guardar.
- El sistema almacena la información registrada por el usuario.



FLUJOS ALTERNATIVOS

- **DUPLICIDAD DE DATOS**

En el caso de ingresar al sistema un código repetido el sistema emitirá un mensaje de error indicando que no se ha podido registrar la infurción.

- **SELECCIÓN TIPOS DE PRODUCTO**

Si no ha seleccionado la categoría a la cual pertenece el producto, el sistema emitirá un mensaje de error indicando que falta la selección correspondiente.

5.4.6. ATENCIÓN AL CLIENTE

5.4.6.1. DESCRIPCIÓN DEL CASO DE USO REGISTRO DE RECLAMOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

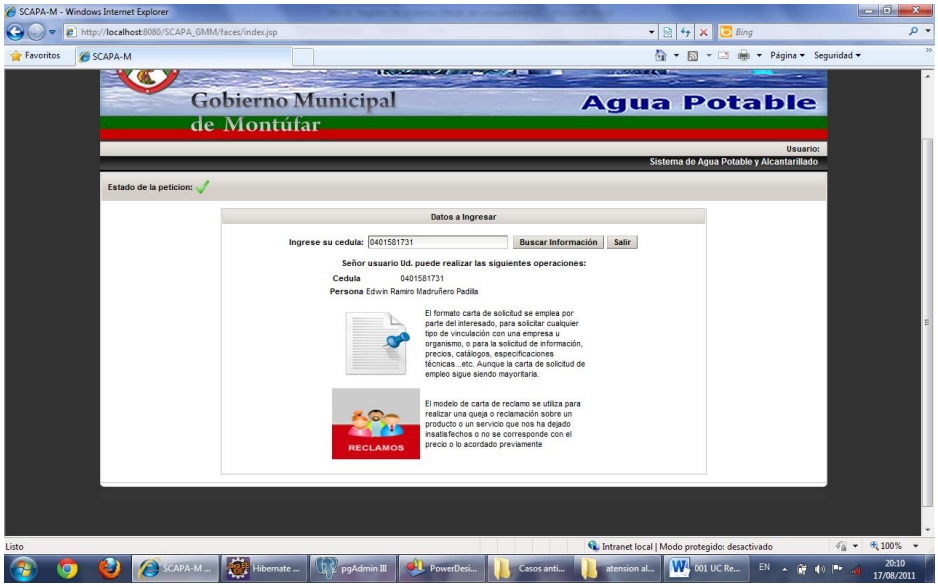
- El caso de uso describe el registro del reclamo que realice el usuario por cualquier eventualidad

FLUJO BÁSICO DE EVENTOS

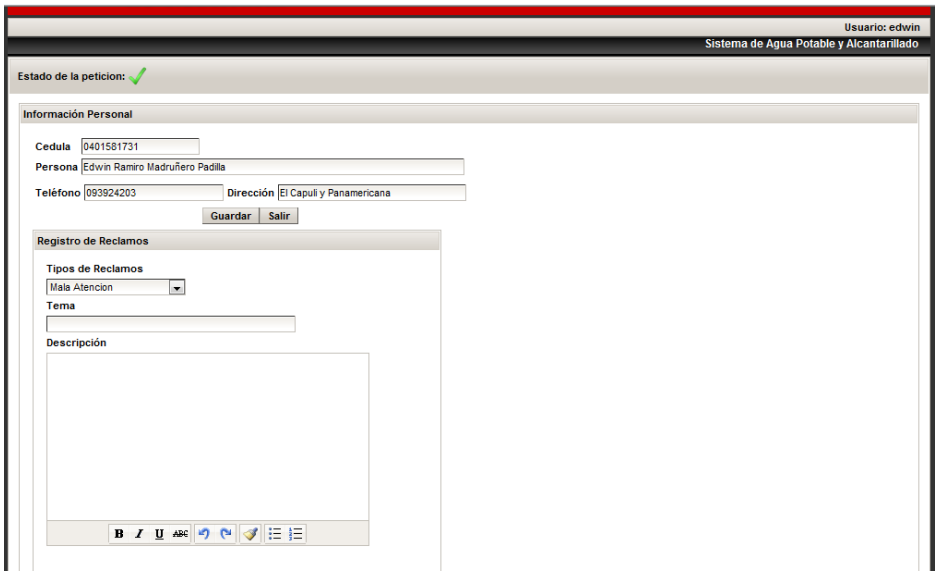
- El cliente deberá acceder desde el internet al sistema.

- Una vez ingresado al sistema damos clic en la parte donde dice **ATENCION AL CLIENTE** la cual nos permitirá ingresar a la siguiente forma.

    a. Ingresamos la cedula y pulsamos el botón buscar.
    b. Una vez ingresado y si el sistema encuentra al usuario ya ingresado se habilitaran las opciones que son de ingreso de reclamos y de búsqueda de cartas no pagadas.
    c. Damos clic en donde dice ingreso de reclamos, donde se nos presentara la siguiente pantalla.



- Pulsamos el botón registro de reclamos y se mostrara la siguiente forma en la cual tienen que registrar la información solicitada.
  a) Aquí el usuario ingresara el tipo de reclamo que va a realizar
  b) Ingresara el tema del reclamo
  c) El cliente en este caso describirá el reclamo que desea realizar



- Una vez realizado el registro, el sistema le presentara al costado derecho el reclamo ya ingresado con su respectivo estado, al momento que el reclamo sea atendido, el usuario en la parte indicada podrá observar la respuesta a su petición.

- El sistema almacena la información registrada por el usuario.

5.4.6.2. ESPECIFICACIÓN DEL CASO DE USO REGISTRO TIPOS DE RECLAMOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

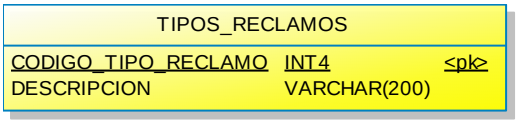
DESCRIPCIÓN BREVE

- El caso de uso describe el registro de los diferentes tipos de reclamos que puede solicitar el cliente

FLUJO BÁSICO DE EVENTOS

- El administrador se encargara de registrar los diferentes tipos de reclamos que puede realizar el cliente

- El sistema almacena la información registrada por el usuario.



5.4.6.3. ESPECIFICACIÓN DEL CASO DE USO RECEPCIÓN DE RECLAMOS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

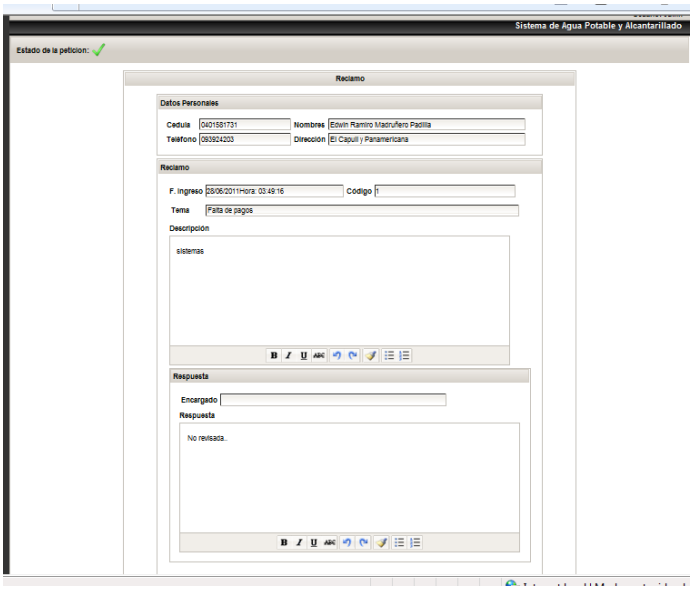
DESCRIPCIÓN BREVE

- El caso de uso describe la revisión de los reclamos registrados por la ciudadanía.

FLUJO BÁSICO DE EVENTOS

- El administrador entra al sistema e ingresa donde se visualiza los diferentes reclamos

- Damos clic en contestar para dar la atención al reclamo solicitado esto lo realiza el administrador del sistema revisa las solicitudes realizadas y que están pendientes de revisar y las actualiza cambiándoles el estado de no revisado a revisado



- El sistema almacena la información registrada por el usuario.

5.4.6.4. ESPECIFICACIÓN DEL CASO DE USO CONSULTA SALDOS DE LOS CLIENTES

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
17/08/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

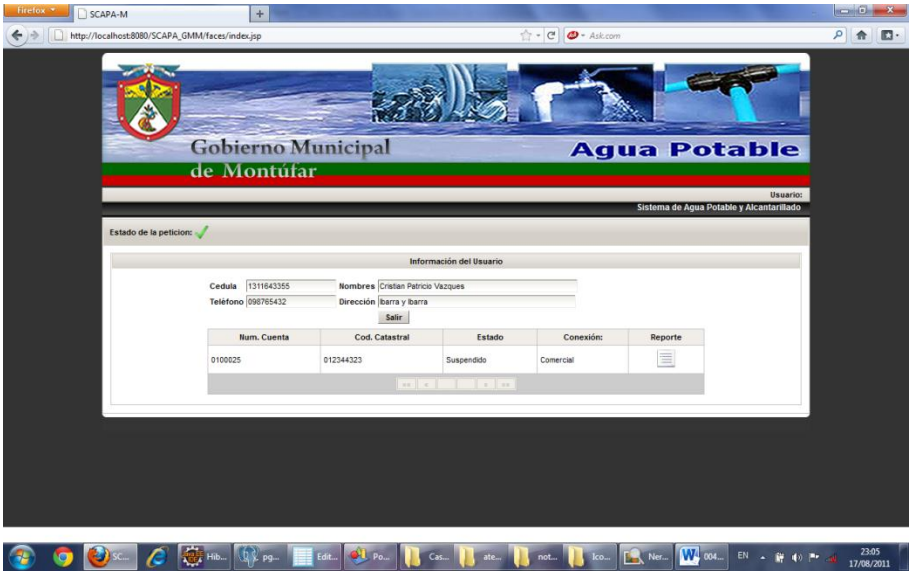
- El caso de uso describe la forma con un usuarios puede generar un reporte de cuantos meces de deuda tiene.

FLUJO BÁSICO DE EVENTOS

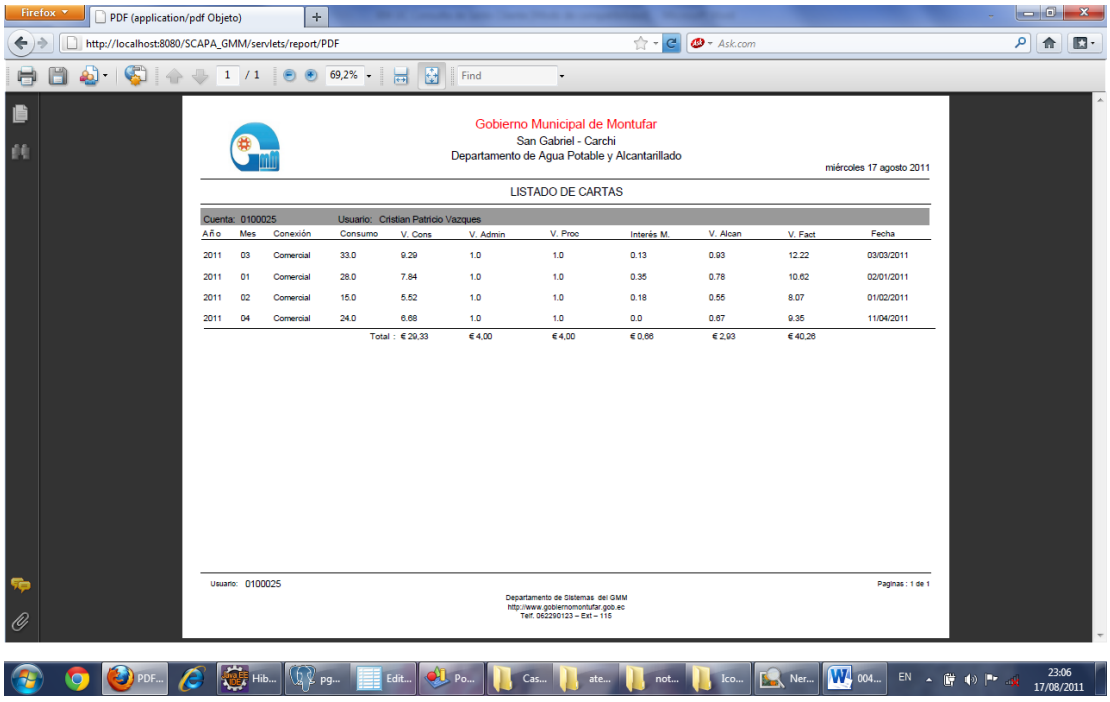
- El Cliente ingresa al sistema donde le mostrara la siguiente pantalla

- Ingresamos a una pantalla donde tendremos que ingresar la cedula de identidad de la persona y pulsar el botón buscar para que el sistema pueda habilitar las opciones indicadas

- En la parte inferior existe un icono el cual nos muestra la pantalla donde tenemos que seleccionar la cuanta de la cuan queremos consultar



- Aquí tenemos la pantalla y en el costado derecho el icono que damos clic y se nos muestra en que formato deseamos nuestro reporte.



5.4.6.5. ESPECIFICACIÓN DEL CASO DE USO REGISTRO DE SOLICITUDES

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
23/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

- El caso de uso describe la información necesaria para una solicitud de una nueva acometida

FLUJO BÁSICO DE EVENTOS

- Ingresamos al sistema y daremos clic donde dice ingreso de solicitudes
- El cliente del sistema se encargara de ingresar al sistema e ingresar la información requerida y necesaria

- Una vez que hayamos dado clic, el sistema nos muestra la siguiente pantalla, donde el cliente ingresara los datos personales o más bien la información requerida para registrar la solicitud

- El sistema almacena la información registrada por el usuario.

SOLICITUD_ACOMETIDAS		
CODIGO_SOLICITUD	INT4	<pk>
CODIGO_USO	INT4	<fk1>
CODIGO	INT4	<fk2>
ESTADO_SOLICITUD	VARCHAR(2)	
CEDULA	VARCHAR(15)	
NOMBRES	VARCHAR(200)	
APELLIDOS	VARCHAR(200)	
DIRECCION	VARCHAR(200)	
NUMERO_CASA	VARCHAR(200)	
FECHA_INGRESO	DATE	

- El usuario puede registrar el número de solicitudes que estime conveniente según la cuenta que desee

5.4.7. RECAUDACIÓN

5.4.7.1. ESPECIFICACIÓN DEL CASO DE USO REGISTRO PAGO CARTAS

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

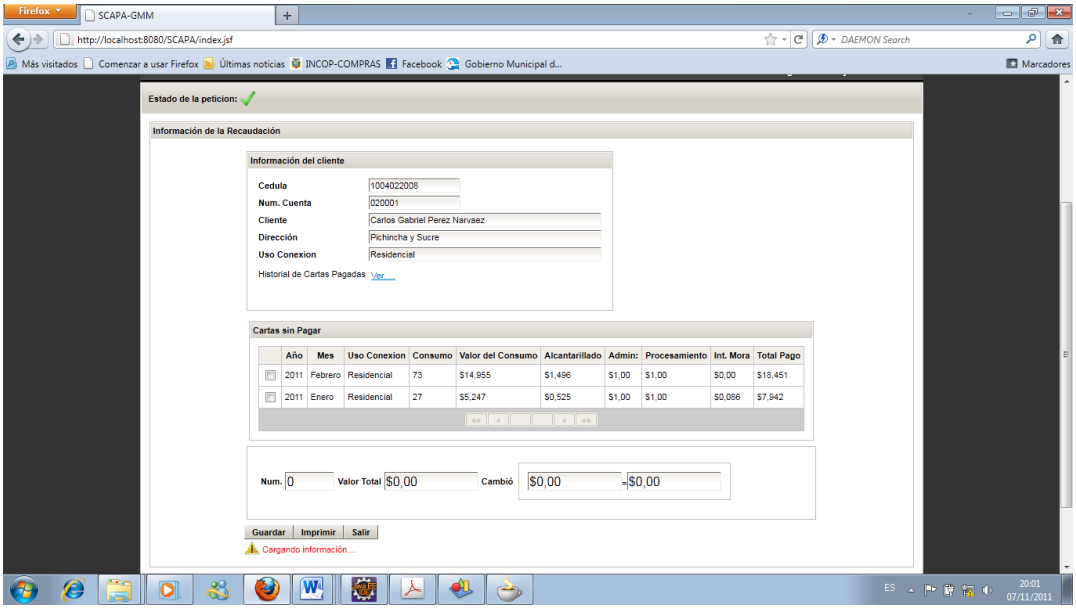
- El caso de uso describe la forma como el cajero realiza la cancelación del consumo del Agua Potable según el consumo, este puede darse de una sola carta o de varias cartas a la vez

FLUJO BÁSICO DE EVENTOS

- El cliente deberá acceder desde el internet al sistema.

- El usuario cajero debe buscar en el menú de la parte de arriba la pestaña donde dice RECAUDACIÓN

- Una ven encotrado nos presentara la siguiente pantalla en la cual debemos seleccionar al cliente despues de haberle buscado, se le presentara la siguiente pantalla donde debe registrar el pago.



- Una vez registrado el pago correctamente el sistema registrara una actualización en las cartas que se has pagado cambiando el estado y registrando la fecha de pago y además asignándoles un número de transacción necesario para realizar el reverso de la misma.

CARTAS		
NUMERO_CUENTA	VARCHAR(100)	<pk,fk1>
ANIO	VARCHAR(10)	<pk,fk1>
MES	VARCHAR(10)	<pk,fk1>
CODIGO_USO	INT4	<fk4>
CON_ANIO	INT4	<fk2>
CODIGO_USUARIO	INT4	<fk3>
CONSUMO	FLOAT8	
VALOR_CONSUMO	FLOAT8	
VALOR_BASURA	FLOAT8	
VALOR_ALCANTARILLADO	FLOAT8	
VALOR_ADMIN	FLOAT8	
VALOR_OTROS	FLOAT8	
PROCESO	FLOAT8	
VALOR_CONEXION	FLOAT8	
VALOR_RECONEXION	FLOAT8	
VALOR_MULTAS	FLOAT8	
VALOR_VARIOS	FLOAT8	
INTERES_MORA	FLOAT8	
INTERES	FLOAT8	
TOTAL_LIQUIDO	FLOAT8	
TOTAL_FACTURA	FLOAT8	
FECHA_LIQUIDACION	DATE	
FECHA_FACTURACION	DATE	
FECHA_PROCESO	DATE	
ESTADO_LIQUIDACION	INT4	
VALOR_IVA	FLOAT8	
VALOR_DESCUENTO	FLOAT8	
COMPRA_MEDIDOR	VARCHAR(2)	
COMPRA_VARIOS	VARCHAR(2)	
CODIGO_TRANASCCION	INT4	
CODIGO_USO_ANTERIOR	INT4	

FLUJOS ALTERNATIVOS

- En el caso de que suceso algún evento fortuito en el pago el sistema reversara toda la transacción dejando las cartas en su estado original.

5.4.7.2. ESPECIFICACIÓN DEL CASO DE USO REVERSO PAGO

Fecha	Versión	Descripción	Autor
31/08/2010	<1.0>	Descripción inicial del caso de uso	Edwin Madruñero
28/06/2011	<1.0>	Revisión del caso de uso	Edwin Madruñero

DESCRIPCIÓN BREVE

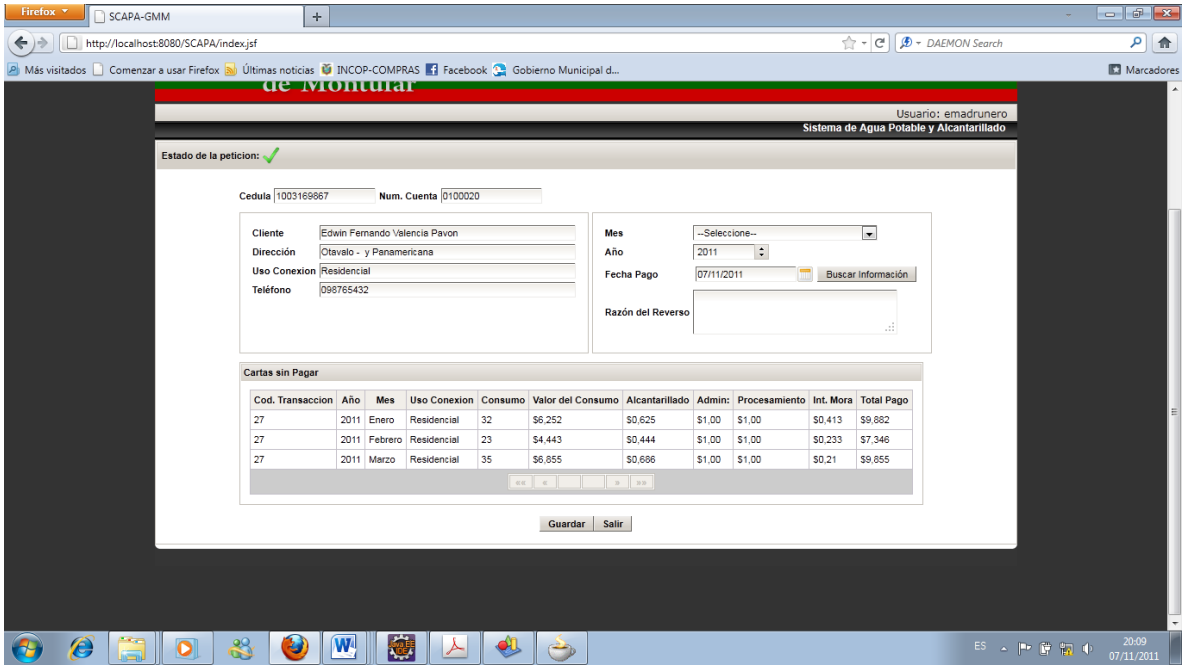
- El caso de uso describe la forma como el sistema realizara el reverso de toda la transacción de pago incluido si en esta se realizaron el pago de cuotas.

FLUJO BÁSICO DE EVENTOS

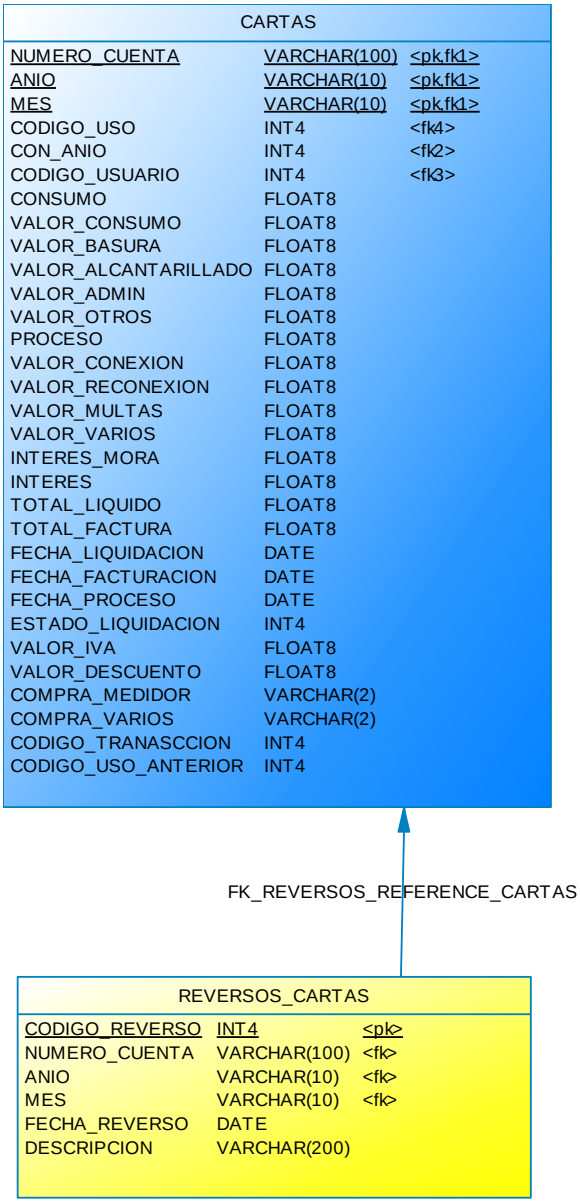
- El cliente deberá acceder desde el internet al sistema.

- El usuario cajero debe buscar en el menú de la parte de arriba la pestaña donde dice Reversar Pago

- Aquí encontramos la pantalla en alcual re realiza el reverso del pago, para ello solo basta con elcontrar una sola de las varias transacciones realizadas y como tenemos un codigo de trasaccion podemos encontrar las demas transacciones realizadas.



- Cuando el reverso se ha realizado también el sistema registra en una tabla cada una de las transacciones reversadas.


CARTAS
NUMERO_CUENTA VARCHAR(100) <pk,fk1>
ANIO VARCHAR(10) <pk,fk1>
MES VARCHAR(10) <pk,fk1>
CODIGO_USO INT4 <fk4>
CON_ANIO INT4 <fk2>
CODIGO_USUARIO INT4 <fk3>
CONSUMO FLOAT8
VALOR_CONSUMO FLOAT8
VALOR_BASURA FLOAT8
VALOR_ALCANTARILLADO FLOAT8
VALOR_ADMIN FLOAT8
VALOR_OTROS FLOAT8
PROCESO FLOAT8
VALOR_CONEXION FLOAT8
VALOR_RECONEXION FLOAT8
VALOR_MULTAS FLOAT8
VALOR_VARIOS FLOAT8
INTERES_MORA FLOAT8
INTERES FLOAT8
TOTAL_LIQUIDO FLOAT8
TOTAL_FACTURA FLOAT8
FECHA_LIQUIDACION DATE
FECHA_FACTURACION DATE
FECHA_PROCESO DATE
ESTADO_LIQUIDACION INT4
VALOR_IVA FLOAT8
VALOR_DESCUENTO FLOAT8
COMPRA_MEDIDOR VARCHAR(2)
COMPRA_VARIOS VARCHAR(2)
CODIGO_TRANASCCION INT4
CODIGO_USO_ANTERIOR INT4
FK_REVERSOS_REFERENCE_CARTAS
REVERSOS_CARTAS
CODIGO_REVERSO INT4 <pk>
NUMERO_CUENTA VARCHAR(100) <fk>
ANIO VARCHAR(10) <fk>
MES VARCHAR(10) <fk>
FECHA_REVERSO DATE
DESCRIPCION VARCHAR(200)

5.5. VISIÓN LÓGICA

5.5.1. MODELO ENTIDAD RELACIÓN.

5.5.2. MODELO FÍSICO.

5.6. PRUEBAS.

5.6.1. IMPLEMENTACIÓN DE PRUEBAS

5.6.1.1. CASOS DE PRUEBA.

Para la elaboración de las pruebas se crearon usuario para cada uno de los roles en la tabla "Usuarios", se le proporciono los permisos necesarios para poder acceder a cada uno de los respectivos módulos.

5.6.1.1.1. ESPECIFICACIÓN DE CASO DE PRUEBA: REGISTRO DE CUENTA

DESCRIPCIÓN.

Este artefacto cubre el conjunto de pruebas realizadas sobre el caso de uso ingreso de cuentas. Se realizaron tres pruebas a este caso de uso se comprobó que faltaban campos de información como puede ser la dirección del predio donde se instalara la acometida, debido a que la dirección que ingresamos en la información de persona contiene la dirección exacta del dueño del lugar, pero esto difiere cuando el usuario compra un predio en otro lugar pero no vive allí. El entorno para realizar la prueba es el formulario de entrada de la aplicación.

COMPROBAR LA MANIPULACIÓN DE DATOS.

En este caso de uso se utilizó un usuario creado, y asignado los permisos de funcionalidad necesarios.

Previamente se registró la información necesaria o más bien los parámetros requeridos para el ingreso o ceración de una nueva cuenta

El usuario una vez ingresado al sistema, busca en la barra de menú la pestaña de mantenimiento, allí encontrar un menú donde dice que desea crear cuenta o actualizar cuenta, debe dar clic en crear cuenta.

Una vez ya ingresado a la forma de ingreso el usuario registrara la información que se encuentra señalada en el formulario, y posteriormente debe cambia de TAB para poder registrar las opciones

catastrales de la cuenta, una vez realizado este proceso damos clic en la opción guardar.

CONDICIONES DE EJECUCIÓN.

Las condiciones de ejecución del caso de prueba son: es que el usuario que realice esta “pruebas” este dado de alta en la base de datos, tenga asignado un los roles necesario para realizar las operaciones de funcionalidad.

ENTRADA.

- ingresar al sistema con el usuario “pruebas” y su respectiva contraseña;
- ingresar al menú principal, escoge menú mantenimiento, cliente, Registro de Cuenta;
- Registramos la información que la forma nos solicita como es cedula, nombre, apellidos calle principal, calle secundaria, seleccionamos el barrio, seleccionamos el contrato, seleccionamos el estado de la cuenta, seleccionamos el tipo de uso conexión, sucursal y demás parámetros que se encuentra en la forma.
- llenar datos relevantes de la forma y guardar;
- el ingreso de la información se realiza en tres tablas como son "personas", "personas_con_cuenta" y en la tabla "cuenta_persona"

RESULTADO ESPERADO.

Que la cuenta se haya creado satisfactoriamente y que el usuario obtenga un reporte de la cuenta lista para asignación de lecturas.

EVALUACIÓN DE LA PRUEBA.

Prueba superada con éxito, para mayor funcionalidad se agregó algunas sentenciones en los botones para cuando se está realizado la transacciones el mimo se bloquea hasta terminar la transacción solicitada.

5.6.1.1.2. ESPECIFICACIÓN DE CASO DE PRUEBA: INGRESO DE LECTURAS.

DESCRIPCIÓN.

Este artefacto cubre el conjunto de pruebas realizadas sobre el caso de uso "Ingreso de Lecturas", el cual es muy utilizado ya que la base de la generación de las cartas son las lecturas.
Pruebas a realizar a este caso de uso:

- Consultar la lecturas anteriores
- Verificar que el al ingresar la lectura y registrar el comentario, el sistema realice los procesos necesarios para el cálculo del valor.
- Verificar que cuando no exista lecturas y el usuario seleccione el estado de la lectura en promedio, el sistema busque el número de cartas anteriores y con ellas realice el promedio.
- Comprobar que el usuario manipule bien la información;

COMPROBAR LA MANIPULACIÓN DE DATOS.

El usuario del sistema "pruebas ", ingresa al sistema, para realizar la búsqueda de las cuentas por diferentes parámetros, pero el más utilizado es el de RUTA y SECUENCIA, cuando se hayan encontrado los elementos pulsamos el botón aceptar y el sistema nos muestra la pantalla en la cual registraremos la lectura.
Una vez ingresado la lectura seleccionamos el comentario de la lectura, y guardamos.

CONDICIONES DE EJECUCIÓN.

Las condiciones de ejecución del caso de prueba son: es que el usuario “pruebas” este dado de alta en la base de datos, tenga asignado los permisos necesarios (ROL).

ENTRADA.

- Ingresar al sistema con el usuario “pruebas” y su respectiva contraseña;
- Ingresar al menú principal, escogemos menú mantenimiento, lecturas, Ingreso de lecturas;

- Aparece la pantalla de búsqueda de cuentas, con varias opciones para realizar la misma;
- Realizada la búsqueda pulsamos aceptar y se nos presenta la pantalla de ingreso de lecturas en la cual ingresamos la lectura y su respectivo comentario.
- Una vez que el sistema realizo el cálculo del valor de la lectura y valor de pago pulsamos guardad y el sistema asmáticamente me cambia a la siguiente cuenta a ingresar.

RESULTADO ESPERADO.

Cuando se seleccione el comentario indicado que el sistema realice las operaciones indicadas para el ingreso de la lectura y cuando el usuario ingrese el estado de la lectura en promedio que el sistema busque el número de meses anteriores indicados en los parámetros para realizar el cálculo.

EVALUACIÓN DE LA PRUEBA.

Prueba superada con éxito.

5.6.1.1.3. ESPECIFICACIÓN DE CASO DE PRUEBA: REGISTRO DE PAGO.

DESCRIPCIÓN.

Este artefacto cubre el conjunto de pruebas realizadas sobre el caso de uso "Registro de pago", el cual es muy utilizado

COMPROBAR LA MANIPULACIÓN DE DATOS.

Ingresa en el sistema como el usuario "pruebas", asignado el correspondiente rol para la correcta funcionalidad del sistema, una vez ingresado al sistema buscamos en el menú recaudación, seleccionamos recaudación y empezamos la manipulación de datos.

CONDICIONES DE EJECUCIÓN.

Las condiciones de ejecución del caso de prueba son: es que el usuario “pruebas” este dado de alta en la base de datos, tenga asignado los permisos necesario es decir los roles.

ENTRADA.

1. ingresar al sistema con el usuario “pruebas” y su respectiva contraseña;
2. ingresar al menú principal, escogemos menú recaudación, recaudación;
3. aparece la pantalla de búsqueda de cuentas, con varias opciones para realizar la misma;
4. Realizada la búsqueda y pulsamos en ver pago, este paso nos pide confirmar la transacción esto es porque se realiza el proceso de búsqueda de cartas deudoras.
5. Cuando ya se nos presenta la pantalla de recaudación el usuario debe seleccionar las cartas a pagar y en este momento si existe cuotas a pagar de la venta de un medidor seleccionara por cara mes una cuota.
6. Guardar los datos;

RESULTADO ESPERADO.

- El sistema al seleccionar las cuotas, toco implementar un control más, el cual nos ayuda a controlar cuando existen mes cartas que meces y menos también

EVALUACIÓN DE LA PRUEBA.

- Prueba superada con éxito

5.7. IMPLEMENTACIÓN DE PRUEBAS

5.7.1. DAIGRAMA DE ACTIVIDADES

- **REGISTRO DE CUENTA**

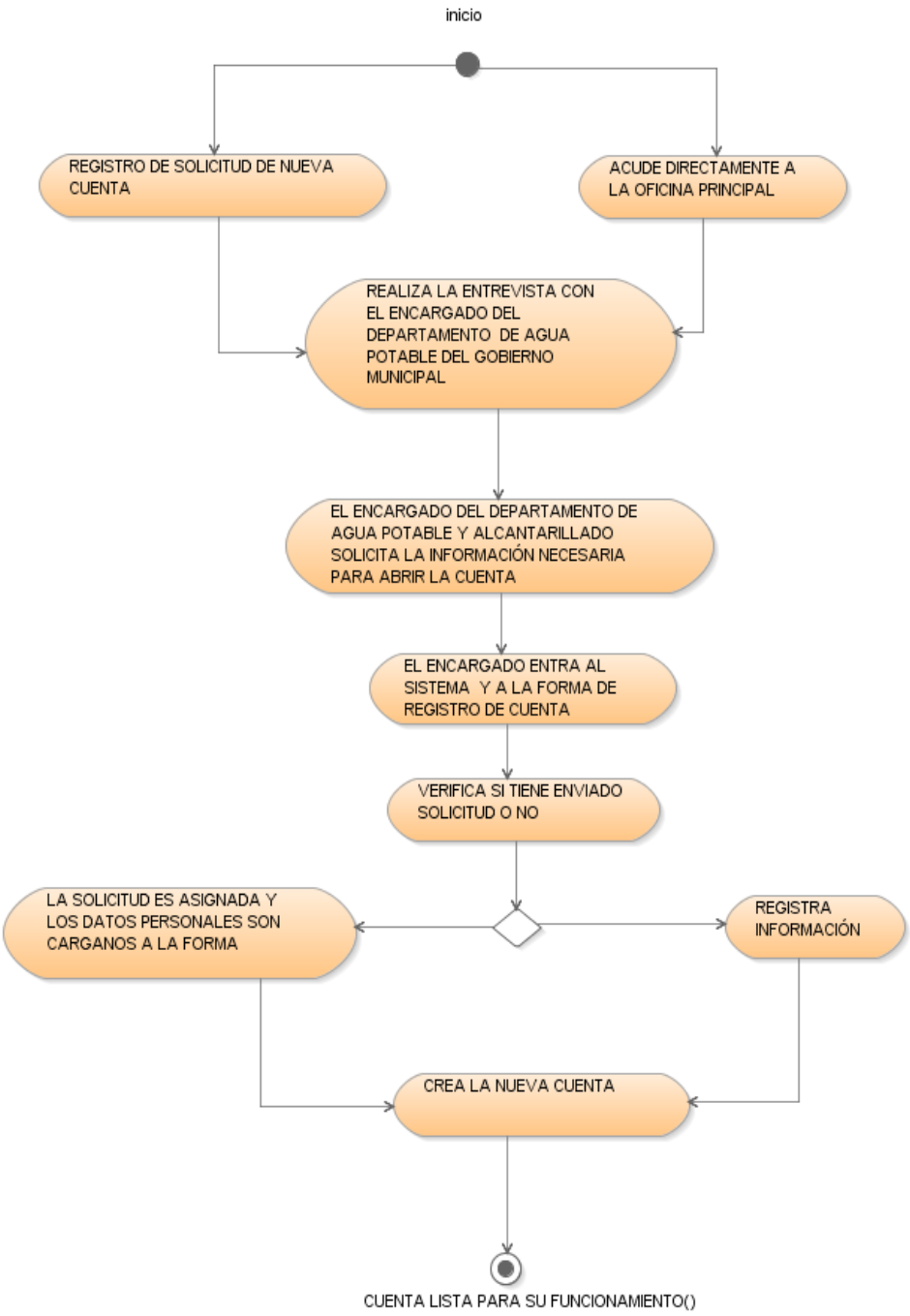


Figura 45 Diagrama de actividades registro lecturas

- **INGRESO DE LECTURAS**

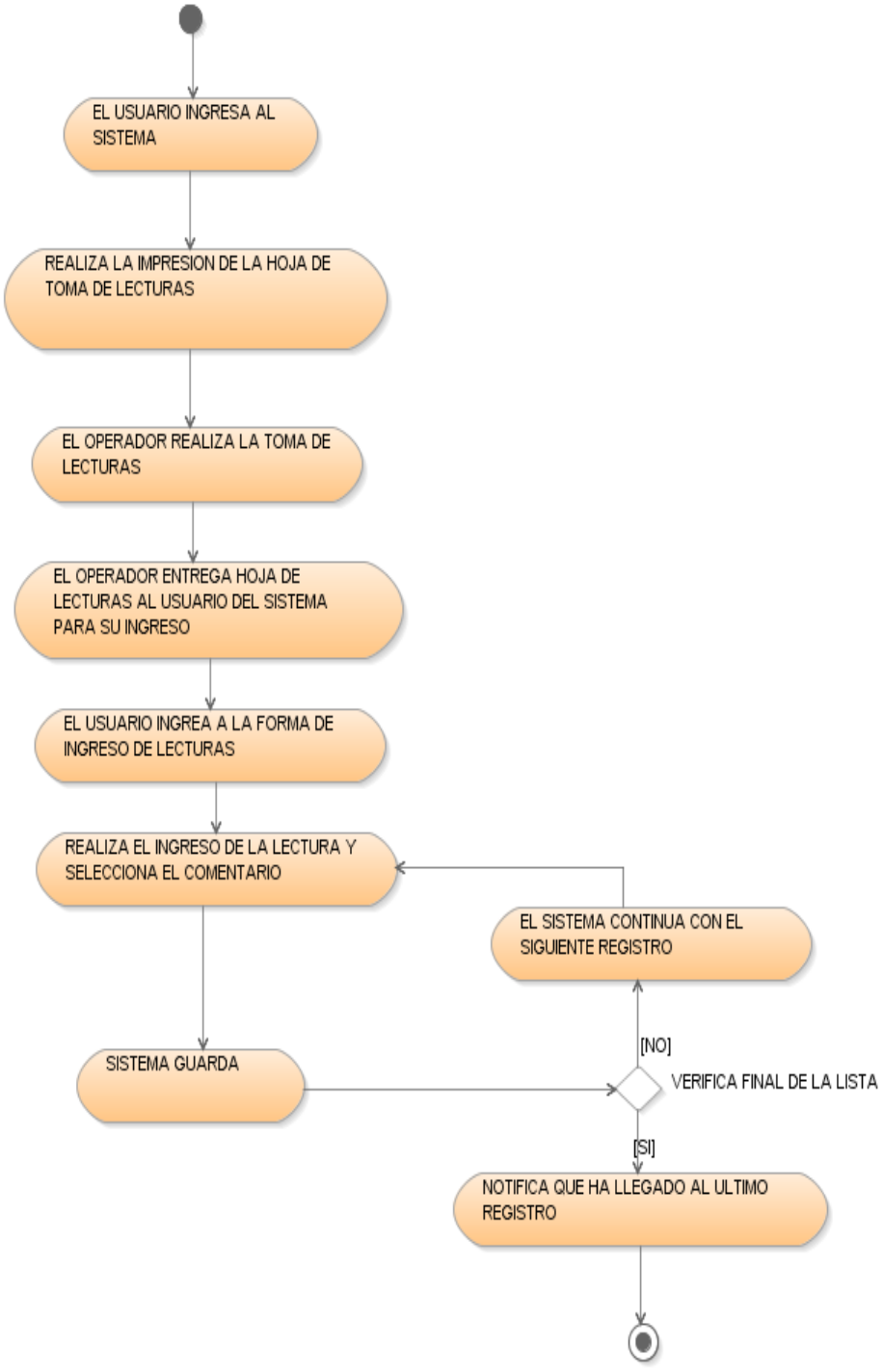


Figura 46 Diagrama de actividades ingreso de lecturas

- **REGISTRO DE PAGOS**

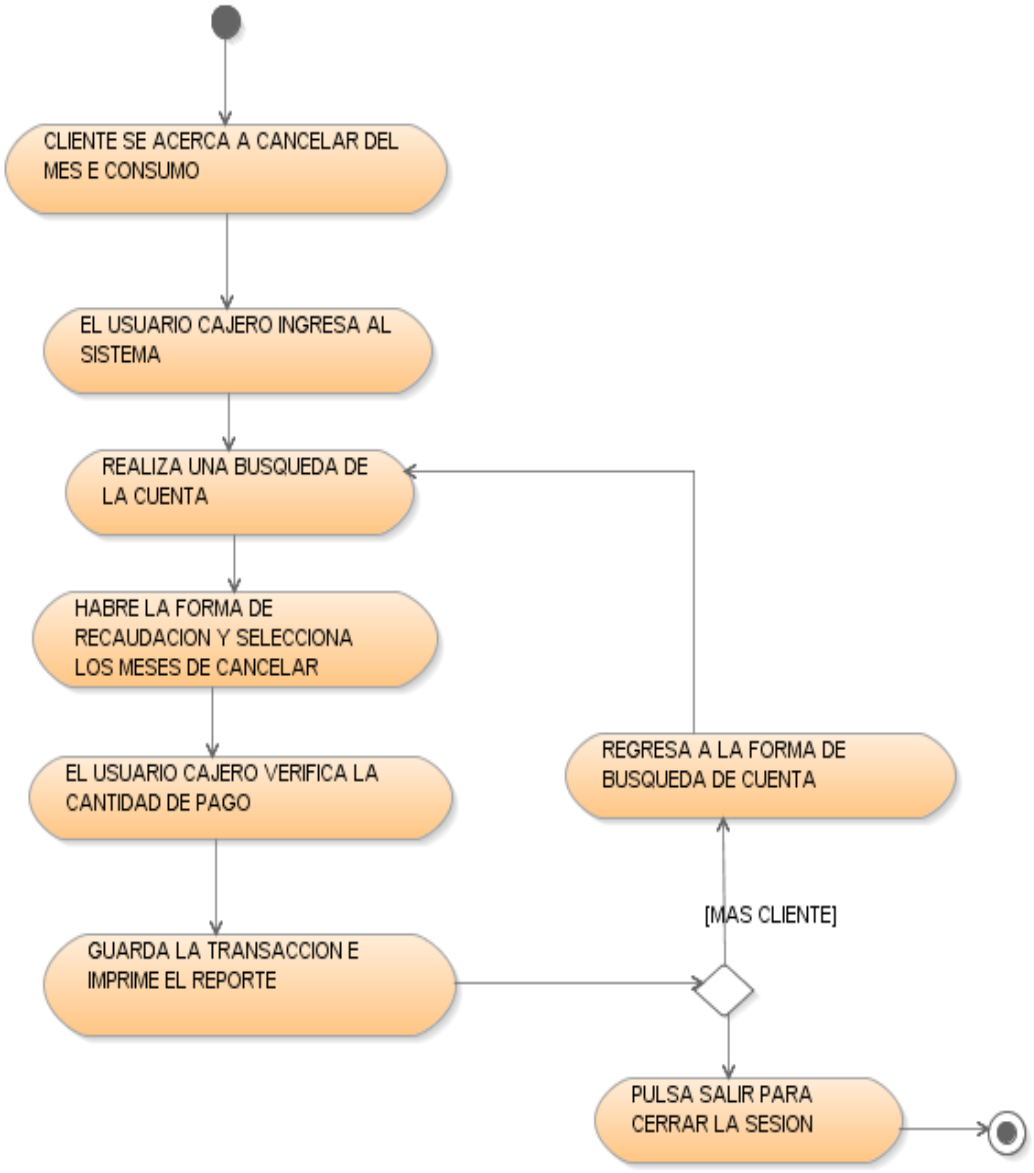


Figura 47 Diagrama de actividades Registro pago

5.7.2. DAIGRAMA DE ARQUITECTURA

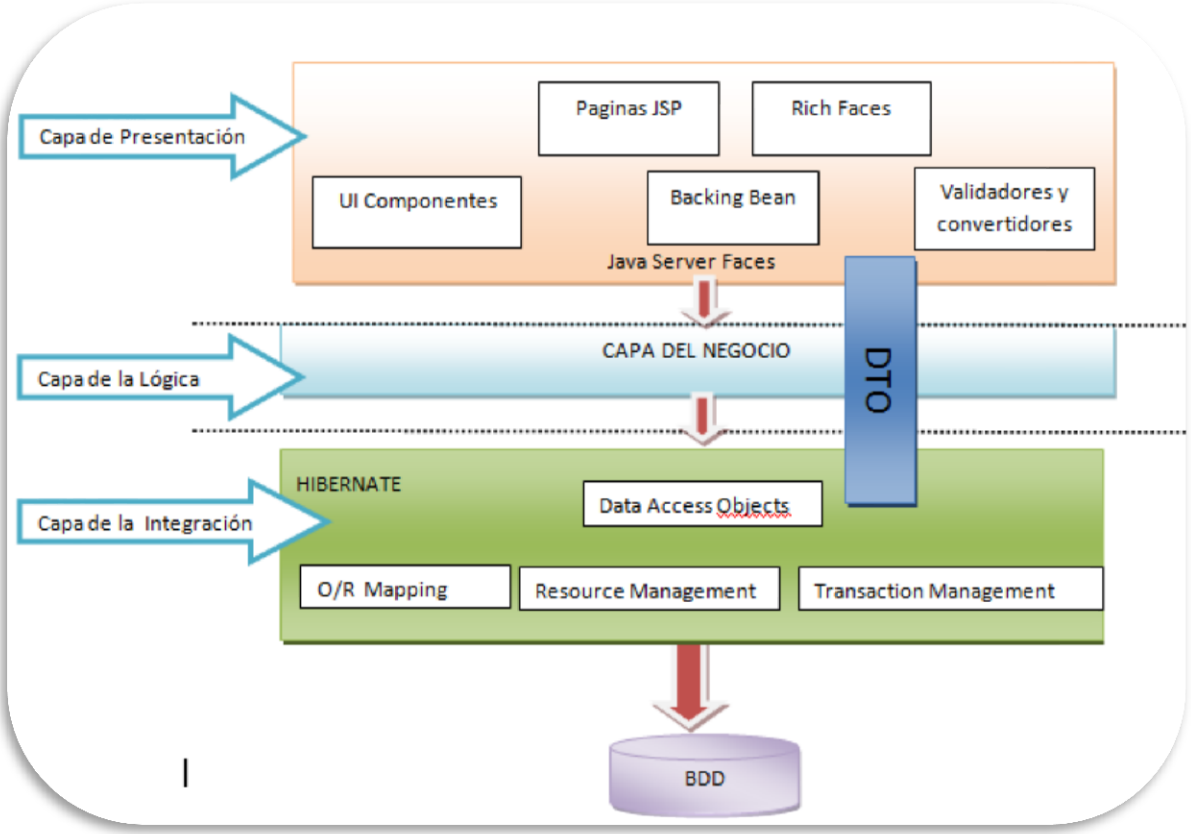


Figura 48 Arquitectura

6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- La Implementación de los Módulos de Facturación y Recaudación optimiza los procesos, elimina tareas innecesarias generando resultados más rápidos.

- El aplicativo desarrollado aporta significativamente al departamento de Agua Potable del Gobierno Municipal de Montúfar agilizando procesos que antes no se encontraban disponibles, además brindando una interface con el cliente para ofrecer un servicio a la ciudadanía

- El motor de base de datos PostgreSQL 8.4, proporciona solidez, confiabilidad, pertenencia, seguridad, integridad, disponibilidad, relevancia con datos almacenados sobre la información del departamento.

- Al utilizar Tecnología Java para el desarrollo de este aplicativo de software, concluyo que es una herramienta excelente y de gran funcionalidad, además de disminuir costos de licenciamiento, posee características que la hacen solida al manejo de un alto volumen de datos, así también otras características que hacen que estas herramientas estén entre unas de las más recomendada para aplicaciones empresariales.

6.2. RECOMENDACIONES

- El aplicativo desarrollado, de acuerdo a sus características de desarrollo debe ser utilizado en el departamento con el fin de brindar un servicio eficiente y dar a conocer a la ciudadanía el nuevo servicio que puede tener al alcance de sus manos.

- Desarrollar los reportes que sus respectivos usuarios crean convenientes a fin de ofrecer información veraz y confiable.

- Desarrollar una depuración de la información, debido a que un estudio realizado dio como resultado índices que nos indican inconsistencia en la información.

GLOSARIO

APLICACIÓN WEB: Aplicación informática que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet.

TURBULENCIA: La turbulencia ocurre cuando un gas o un fluido, como el aire o el agua, es impelido a alta velocidad o en grandes cantidades, y se caracteriza por modelos de flujo caóticos, aparentemente al azar. Debido a su complejidad, la turbulencia es muy eficaz para mezclar: la solución de dos líquidos, como la crema y el café, se producirá mucho más rápidamente si el flujo es turbulento que si no lo es.

FLOCULADORES: La floculación es un proceso químico mediante el cual, con la adición de sustancias denominadas floculantes, se aglutinan las sustancias coloidales presentes en el agua, facilitando de esta forma su decantación y posterior filtrado.[cita requerida] Es un paso del proceso de potabilización de aguas de origen superficial y del tratamiento de aguas servidas domésticas, industriales y de la minería.

LA SEDIMENTACIÓN: es un fenómeno natural que sustenta una de las operaciones básicas de más solera en ingeniería de procesos, cuyas aplicaciones más eficientes y económicas, y cuyos más estimulantes requerimientos, tienen lugar con frecuencia en el ámbito del tratamiento de efluentes residuales.

API: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Representa una interfaz de comunicación entre componentes
Software.

ARQUITECTURA: Representación abstracta de los componentes de un sistema y su comportamiento

CLASE: Definición de un objeto.

DAO: componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

HIBERNATE: Herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de Hibérnate) que facilita

el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

JAVA: Lenguaje de Programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

JDBC: Java DataBase Connectivity es el API de Java que define como una aplicación cliente accederá a una base de datos, independientemente del motor de base de datos al que accedamos.

MAPPING: Proceso de conectar objetos/atributos a tablas/columnas.

MVC: Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Persistencia: Capacidad de almacenar y recuperar el estado de los objetos, de forma que sobreviran a los procesos que los manipulan.

POJO (Plain Old Java Object): Simple clase Java que tiene métodos get y set para cada uno de los atributos.

OBJETO: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

ORM (Object/Relational Mapping): Técnica que realiza la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

SERVLET deriva de otra anterior, applet, que se refería a pequeños programas que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

APPLET es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por 'applets'.

FRAMEWORK DE PERSISTENCIA: Componente de software encargado de traducir entre objetos y registros (de la base de datos relacional). Es decir es el encargado de que el programa y la base de datos se “entiendan”.

RICHFACES: es una librería de código abierto basada en Java que permite crear aplicaciones web con Ajax.

CARROZABLE: Dicho de un camino: Destinado al tránsito de vehículos
SCAPA: Sistema de Comercialización de Agua Potable y Alcantarillado, sistema a ser desarrollado para la Municipalidad de Montúfar.

SERVICIO DE AGUA POTABLE: Servicio que ofrece la Municipalidad a la ciudadanía del cantón Montúfar

ACOMETIDA: Es una derivación de una red de consumo del suministro de agua potable hacia una vivienda, predio, etc.

PREDIO: Terreno, bien inmueble que posee una persona.

ALTAS: Ingreso de nuevas Acometidas.

BAJAS: Egreso de las acometidas

TRASPASO: Cambio de Usuario final.

SISTEMA DE CATASTROS: El sistema encargado del registro de cada predio y asignación de un código único.

SISTEMA DE BODEGA: Permite registrar los bienes que ingresan y salen de la Municipalidad.

BIBLIOGRAFÍA

INTERNET

- [WWW1] http://es.wikipedia.org/wiki/PostgreSQL, Esta página fue modificada por última vez el 7 dic 2011, a las 12:01; Wikipedia® es una marca registrada de la Fundación Wikipedia, Inc., una organización sin ánimo de lucro.

- [WWW2] http://www.programacion.com/articulo/tomcat_-_introduccion_134; Publicado por Eloy A. Esteban; Copyright © 1998-2011 Programación en Castellano.

- [WWW3] http://livedemo.exadel.com/richfaces-demo/richfaces/tabPanel.jsf

- [WWW4] http://blog.jotadeveloper.com/2008/09/07/ejemplo-richfaces-jsf

- [WWW5] http://frankseguel.blogspot.com/2008/05/introduccin-richfaces.html

- [WWW6] http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html/index.html; Publicado May 05, 2010; Copyright © 2008, 2009 Red Hat.

- [WWW7] http://www.davidmarco.es/tutoriales/hibernate-reference/; publicado por Gavin King, Christian Bauer, Max Rydahl Andersen, Emmanuel Bernard, y Steve Ebersole; Copyright © 2004 Red Hat Middleware, LLC.

- [WWW8] http://docs.jboss.org/ejb3/docs/tutorial/1.0.7/html/index.html ; publicado por Bill Burke, Jaikiran Pai

- [WWW9] http://tomcat.apache.org/ ; Apache Software Fundation; *Copyright © 1999-2012, The Apache Software Foundation*

- [WWW10]
  http://es.wikipedia.org/wiki/Tomcat Esta página fue modificada por última vez el 9 dic 2011, a las 11:20; Wikipedia® es una marca registrada de la Fundación Wikipedia, Inc.

- [WWW11]
  http://en.wikipedia.org/wiki/CentOS

- [WWW12]
  http://www-01.ibm.com/software/awdtools/rup/

- [WWW13]
  www.google.com

- [WWW14]
  www.wikipedia.org

- [WWW15]
  http://www.itinerario.psico.edu.uy/NormasAPA.htm

LIBROS

- [LIB01]
  Monografía del Cantón Montúfar, por Zenón Ponce Ch, Ilustre Municipalidad de Montúfar; edición 1955-2010 .

- [LIB02]
  Manual Hibérnate, Fecha de creación: 21.03.2003, Héctor Suárez González (hsg arroba telecable punto es)

- [LIB03]
  Practical RichFaces, por Max Kats; Copyright © 2008 by Max Katz

- [LIB03]
  RichFaces Developer Guide, RichFaces framework with a huge library of rich components and skinnability support