UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

ARTÌCULO CIENTÌFICO

TEMA:

“ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM.

APLICATIVO: APLICACIÓN WEB PARA MANTENIMIENTO MECÁNICO EN INDUSTRIAS CÁRNICAS PARA LA EMPRESA PÚBLICA MUNICIPAL DE FAENAMIENTO Y PRODUCTOS CÁRNICOS DE IBARRA.”

AUTOR: ROBINSON DANILO CHÁVEZ CABRERA

DIRECTOR: ING. XAVIER MAURICIO REA PEÑAFIEL

IBARRA – ECUADOR

2016

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 DEL CONTACTO			
CEDULA DE IDENTIDAD	0401466503		
APELLIDOS Y NOMBRES	CHÁVEZ CABRERA ROBINSON DANILO		
DIRECCIÓN	JUAN MARTÍNEZ DE ORBE 2-79 Y ZUMBA		
EMAIL	rdchavezc@utn.edu.ec		
TELÉFONO FIJO	062643074	TELÉFONO MÓVIL	0989313653
DATOS DE LA OBRA			
TÍTULO	"ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM. APLICATIVO: APLICACIÓN WEB PARA MANTENIMIENTO MECÁNICO EN INDUSTRIAS CÁRNICAS PARA LA EMPRESA PÚBLICA MUNICIPAL DE FAENAMIENTO Y PRODUCTOS CÁRNICOS DE IBARRA."		
FECHA	ENERO DE 2015		
PROGRAMA	PREGRADO		
TÍTULO POR EL QUE OPTA	INGENIERO EN SISTEMAS COMPUTACIONALES		
DIRECTOR	ING. MAURICIO REA PEÑAFIEL		

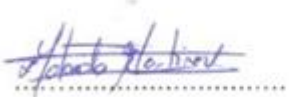
2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Robinson Danilo Chávez Cabrera, con cédula de ciudadanía Nº 0401466503, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago la entrega del ejemplar respectivo en formato 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 artículo 144.

3. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asumo la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Danilo Chávez Cabrera
0401466503



Ing. Betty Chávez
Jefe de Biblioteca

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Robinson Danilo Chávez Cabrera, con cédula de ciudadanía Nº 0401466503, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: "**ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM. APLICATIVO: APLICACIÓN WEB PARA MANTENIMIENTO MECÁNICO EN INDUSTRIAS CÁRNICAS PARA LA EMPRESA PÚBLICA MUNICIPAL DE FAENAMIENTO Y PRODUCTOS CÁRNICOS DE IBARRA**", ha sido desarrollado para optar por el título de Ingeniero 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, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

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: Robinson Danilo Chávez Cabrera

Cédula: 0401466503

Ibarra, Agosto del 2016

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

Certifico que el trabajo de grado titulado: "**ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM. APLICATIVO: APLICACIÓN WEB PARA MANTENIMIENTO MECÁNICO EN INDUSTRIAS CÁRNICAS PARA LA EMPRESA PÚBLICA MUNICIPAL DE FAENAMIENTO Y PRODUCTOS CÁRNICOS DE IBARRA.**", ha sido realizado en su totalidad por el egresado Robinson Danilo Chávez Cabrera portador de la cédula de ciudadanía Nº. 0401466503.

Ing. Xavier Rea Peñafiel

DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÒN

Yo, Robinson Danilo Chávez Cabrera, declaro bajo juramento que el trabajo aquí descrito es de mí autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo los derechos de propiedad intelectual correspondiente a este trabajo, a la Universidad Técnica del Norte, según lo establecido por la Ley de Propiedad Intelectual por su reglamento y por la normatividad institucional vigente.

Firma: .....

Nombre: Robinson Danilo Chávez Cabrera

Cédula: 0401466503

Ibarra, Agosto del 2016

EMPRESA PÚBLICA
MUNICIPAL DE FAENAMIENTO
Y PRODUCTOS CÁRNICOS DE IBARRA

IBARRA

ING. KLEVER ALEJANDRO TABOADA SALAZAR, en mi calidad de INGENIERO MECÁNICO-JEFE de Mantenimiento de la Empresa Pública Municipal de Faenamiento y Productos Cárnicos de Ibarra, tengo a bien;

C E R T I F I C A R

Que: el señor Egresado de la Universidad Técnica del Norte, Facultad de Ingeniería en Ciencias Aplicadas, Escuela de Ingeniería en Sistemas Computacionales, CHAVEZ CABRERA ROBINSON DANILO, con cédula de ciudadanía 040146650-3, elaboró para la institución el Sistema de Mantenimiento Mecánico "SIMMEC" mismo que se instaló y funciono luego de las respectivas pruebas y capacitación recibida.

Además, debo indicar que no dejo el respaldo del sistema por cuanto la empresa no contaba con la partida presupuestaria para la adquisición del sistema, sin embargo su compromiso era el de seguir dando el soporte técnico cuando la empresa así lo requiera.

Es lo que puedo certificar y faculto al interesado hacer uso del presente como estime conveniente dentro del marco legal.

Ibarra, 15 de enero de 2016

EMPRESA PÚBLICA FAENAMIENTO IBARRA SECRETARIA EP-FYPROCAI

Ing. Klever Taboada
INGENIERO MECÁNICO
JEFE DE MANTENIMIENTO EP-FYPROCAI

DEDICATORIA

El desarrollo del presente trabajo está dedicado a las personas que directa o indirectamente aportaron en la culminación del proyecto de tesis.

A mis padres por su apoyo incondicional y abnegado a lo largo de mi vida, a quienes aportaron con su granito de arena en la culminación de esta nueva meta, llena de sacrificios, esfuerzo y perseverancia.

Robinson Danilo Chávez Cabrera

AGRADECIMIENTO

A Dios por su incondicional amistad guiando mis pasos en este largo camino de aprendizaje, quien ha fortalecido mi paciencia en las adversidades y fortaleza en las dificultades.

A la Universidad Técnica del Norte por permitirme formarme académica, deportiva y políticamente en sus aulas.

A mi familia que siempre me apoyo dándome aliento, confianza y cariño en cada meta que tenía que cumplir.

A mis compañeros de facultad por su acolite, consejos y amistad.

A los docentes de la Facultad de Ingeniería en Ciencias Aplicadas por los conocimientos impartidos, al Ing. Xavier Rea Peñafiel, por su guía en el desarrollo del presente trabajo de grado.

Robinson Danilo Chávez Cabrera

RESUMEN

El objetivo principal de este proyecto de tesis es el desarrollo del Sistema de Mantenimiento Mecánico o SIMMEC, para la Empresa Pública de Faenamiento y Productos Cárnicos de Ibarra, que está desarrollado en Django un framework open source, basado en python un lenguaje programación interpretado; dicho framework por sus características ha sido utilizado para agilizar el desarrollo de una aplicación web para la industria cárnica. El sistema trabaja sobre MariaDB un fork open source de MySQL, una base de datos relacional compatible con el framework, muy popular en nuestro medio por su configuración y por contar con una comunidad a nivel mundial de soporte como de desarrollo; la metodología utilizada Scrum es una metodología ágil que utiliza ciclos para una tarea específica por medio de equipos multidisciplinarios.

Los resultados obtenidos han sido satisfactorios para la empresa ya que ha logrado en gran medida solucionar el problema de la maquinaria al poder tener un control del mantenimiento rutinario, no rutinario y correctivo; así como poder registrar las actividades a realizarse durante dicho mantenimiento, ahorrando tiempo y evitando detener el proceso de faenamiento de la empresa municipal.

SUMMARY

Mechanical Maintenance System or SIMMEC, for Public Company slaughtering and meat products Ibarra, Django is developed in an open source framework based on an interpreted programming language python; said framework for its characteristics has been used to accelerate the development of a web application for the meat industry. The system works on open source MariaDB a fork of MySQL, a database-compliant relational database with the framework, very popular in our country for its configuration and have a worldwide community support and development; the methodology Scrum is an agile methodology used for a specific task cycles through multidisciplinary teams.

The results have been satisfactory for the company has since achieved largely solve the problem of power machinery have control routine maintenance, non-routine and corrective; and to record the activities carried out during such maintenance, saving time and avoiding stop the butchering process of the municipal company.

ÍNDICE DE CONTENIDO

AUTORIZACIÓN DE USO Y PUBLICACIÓN II

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE IV

CERTIFICACIÓN V

DEDICATORIA VIII

AGRADECIMIENTO IX

RESUMEN X

SUMMARY XI

ÍNDICE CONTENIDO XII

ÍNDICE DE ILUSTRACIONES XX

ÍNDICE DE TABLAS XXI

CAPÍTULO I 1

1 INTRODUCCIÓN 1

1.1 INTRODUCCIÓN 1

1.2 PROBLEMA 1

1.3 OBJETIVOS 2

1.3.1 GENERAL 2

1.3.2 ESPECÍFICOS 2

1.4 SITUACIÓN ACTUAL DE LOS CENTROS DE FAENAMIENTO. 3

1.5 ALCANCE 5

1.5.1 DE LA INVESTIGACIÓN 5

1.5.2 DEL APLICATIVO 5

1.6 JUSTIFICACIÓN 8

1.6.1 DE LA INVESTIGACIÓN 8

1.6.2 DEL APLICATIVO 8

CAPÌTULO II 10
2 MARCO TEÓRICO 10
2.1 PYTHON 10
2.1.1 ¿QUÉ ES PYTHON? 10
2.1.2 CARACTERÍSTICAS 10
2.1.3 FRAMEWORKS DE PYTHON 11
2.2 FRAMEWORK DJANGO 12
2.2.1 ¿QUÉ ES UN FRAMEWORK? 13
2.2.2 HISTORIA 13
2.2.3 DEFINICIÓN 13
2.2.4 CARACTERÍSTICAS 14
2.2.5 VENTAJAS Y DESVENTAJAS 14
2.2.6 SISTEMAS DE SEGURIDAD INCLUIDOS 14
2.2.7 ARQUITECTURA MVC 18
2.2.8 ESTRUCTURA MVC EN EL FRAMEWORK DJANGO 19
2.2.9 TAXONOMÍA DE DJANGO 21
2.2.10 ESTRUCTURA DE UNA APLICACIÓN DJANGO 22
2.3 PALABRAS RESERVADAS 23
2.3.1 CAMPOS 23
2.3.2 AUTO FIELD 24
2.3.3 BOOLEANFIELD 24
2.3.4 CHARFIELD 25
2.3.5 COMMASEPARATEDINTEGERFIELD 25
UN CAMPO DE ENTEROS SEPARADOS POR COMAS. IGUAL QUE EN CHARFIELD, SE REQUIERE EL ARGUMENTO MAX_LENGTH. 25
2.3.6 DATETIMEFIELD 25
2.3.7 FILEFIELD 25
2.3.8 FILEPATHFIELD 26

2.3.9 FLOATFIELD 27
2.3.10 PHONENUMBERFIELD 27
2.3.11 POSITIVESMALLINTEGERFIELD 28
2.4 OPCIONES UNIVERSALES DE CAMPO 28
2.4.1 NULL 28
2.4.2 BLANK 29
2.4.3 CHOICES (ESCOGER) 29
2.4.4 DB_COLUMN 30
2.4.5 DB_INDEX 31
2.4.6 PRIMARY_KEY 31
2.4.7 RADIO_ADMIN 31
2.4.8 VERBOSE_NAME 32
2.5 RELACIONES 33
2.5.1 RELACIONES MUCHOS-A-UNO 33
2.5.2 RELACIONES MUCHOS-A-MUCHOS 35
2.6 OPCIONES DE LOS METADATOS DEL MODELO 36
2.6.1 DB_TABLE 37
2.6.2 GET_LATEST_BY 37
2.6.3 ORDER_WITH_RESPECT_TO 38
2.6.4 ORDERING 38
2.6.5 PERMISSIONS 39
2.6.6 UNIQUE_TOGETHER 39
2.6.7 VERBOSE_NAME 40
2.6.8 VERBOSE_NAME_PLURAL 40
2.7 MANAGERS 40
2.7.1 NOMBRES DE MANAGER 41
2.7.2 MANAGERS PERSONALIZADOS 41
2.7.3 AGREGANDO MÉTODOS EXTRA AL MANAGER 41

2.7.4 MODIFICANDO LOS QUERYSETS INICIALES DEL MANAGER 43
2.8 MÉTODOS PARA MODELOS ESPECIALES 44
2.8.1 __STR__ 44
2.8.2 GET_ABSOLUTE_URL 45
2.9 MIDDLEWARE 47
2.9.1 ¿POR QUÉ LOS MIDDLEWARES? 47
2.9.2 ¿QUÉ ES UN MIDDLEWARE? 47
2.9.3 INSTALACIÓN DE MIDDLEWARE 48
2.9.4 MIDDLEWARE DE SOPORTE PARA AUTENTICACIÓN 48
2.9.5 MIDDLEWARE “COMMON” 48
2.9.9 MIDDLEWARE DE SOPORTE PARA SESIONES 50
2.9.11 MIDDLEWARE DE TRANSACCIÓN 50
2.9.12 MIDDLEWARE “X-VIEW” 50
2.10 BASE DE DATOS RELACIONAL MARIADB 51
2.10.1 INTRODUCCIÓN 51
2.10.2 ¿POR QUÉ UN FORK? 52
2.10.3 MOTOR DE ALMACENAMIENTO XTRADB 52
2.10.4 ¿QUÉ ES SQL? 54
2.10.5 ¿CUÁLES SON LAS VENTAJAS DE SQL? 54
2.10.6 CONCEPTO ACID 54
2.10.7 CARACTERÍSTICAS 57
2.11 MOTORES DE ALMACENAMIENTO 58
2.11.1 ARIA 58
2.11.2 XTRADB (REEMPLAZO DE INNODB) 59
2.11.3 PBXT (MARIADB 5.1, 5.2 Y 5.3. DESACTIVADO EN MARIADB 5.5.) 60
2.11.4 FEDERATEDX 60
2.11.5 SOPORTE DE PLUG-INS. 61
2.11.6 OQGRAPH (MARIADB 5.1, 5.2 Y 5.3. DESACTIVADO 5.5.) 61

2.11.7 SPHINXSE....61
2.11.8 TOKUDB....62
2.11.9 CASSANDRA....62
2.11.10 MEJORAS DE VELOCIDAD....63
2.11.11 EXTENSIONES....64
2.12 ESTADÍSTICAS DE USUARIOS....65
2.12.1 AUTENTICACIÓN CONECTABLE....65
2.12.2 LAS MEJORAS EN LA TABLA INFORMACIÓN SCHEMA.PLUGINS....65
2.12.3 GRUPO DE ENVÍO AL REGISTRO BINARIO....69
2.12.4. COLUMNAS DINÁMICAS....70
2.13 VENTAJAS Y DESVENTAJAS....71
2.13.1 VENTAJAS....71
2.13.2. DESVENTAJAS....71
2.14 INTEGRACIÓN DEL FRAMEWORK DJANGO CON MARIADB....72
2.14.1 ¿QUÉ ES UNA API?....72
2.14.2 API'S PARA MARIADB....73
2.15 MARIADB/MYSQL EN DJANGO....76
2.15.1 ASPECTOS GENERALES....76
2.16 MARIADB....77
2.16.1 MOTORES DE ALMACENAMIENTO....77
2.16.2 MYSQLDB....78
2.16.3 DEFINICIONES DE ZONA HORARIA....78
2.16.4 CONEXIÓN A LA BASE DE DATOS....78
2.16.5 LOS NOMBRES DE TABLAS....79
2.16.6 LOS PUNTOS DE RETORNO....79
2.16.7 CAMPOS DE CARACTERES....79
2.17 COMPATIBILIDAD MARIADB Y MYSQL....79
2.18 HERRAMIENTAS DE MARIADB....81

2.18.1 EMS SQL MANAGER FOR MYSQL 81
2.18.2 PHPMYADMIN 82
2.18.3 DBDESIGNER 82
2.18.4 HEIDISQL 82
2.18.5 UMBRELLO 83
2.19 METODOLOGÍA ÁGIL DE DESARROLLO 83
2.19.1 MANIFIESTO PARA DESARROLLO ÁGIL 83
2.20 VENTAJAS Y DESVENTAJAS AL UTILIZAR DJANGO CON MARIADB. 85
2.20.1 VENTAJAS 85
2.20.2 DESVENTAJAS 85
2.21 PRINCIPALES METODOLOGÍAS ÁGILES 86
2.21.1 SCRUM 86
2.21.2 XP O XTREAM PROGRAMMING 86
2.21.3 DESARROLLO LEAN 86
2.22 METODOLOGÍA DE SCRUM 87
2.22.1 ARTEFACTOS Y EVENTOS DE SCRUM 88
2.22.2 CONTROL DE EVOLUCIÓN DEL PROYECTO 88
2.22.3 VISIÓN DEL PROCESO 89
CAPÍTULO III 91
3 DISEÑO ARQUITECTÓNICO DE LA SOLUCIÓN 91
3.1 HERRAMIENTAS DE DESARROLLO 91
3.1.1 PLATAFORMA 91
3.1.2 LENGUAJE DE PROGRAMACIÓN 92
3.1.3 ENTORNO 93
3.1.4 MODELADO LÓGICO 94
3.1.5 PENCIL 95
3.1.6 SERVIDOR HTTP 95

CAPÍTULO IV 97
4 DISEÑO Y DESARROLLO E IMPLEMENTACIÓN DEL APLICATIVO. 97
4.1 ANÁLISIS Y DEFINICIÓN DE LOS REQUERIMIENTOS CON EVENTOS ("SPRINTS") 97
4.1.1 PROPÓSITO DE ESTE DOCUMENTO 97
4.1.2 ALCANCE 97
4.1.3 DESCRIPCIÓN GENERAL DE LA METODOLOGÍA 97
4.1.4 ARTEFACTOS 99
4.1.5 RESPONSABILIDADES Y ACTIVIDADES DEL EQUIPO. 100
4.1.6 PILA DEL SPRINT 101
4.2 VISTA LÓGICA DE LA BASE DE DATOS 107
4.3 MODELO DE CASOS DE USO 108
CAPÍTULO V 111
5 ANÁLISIS COSTO, ANÁLISIS IMPACTO, CONCLUSIONES Y RECOMENDACIONES. 111
5.1 ANÁLISIS DE COSTO 111
5.1.1 COSTOS EN HARDWARE. 111
5.1.2 COSTOS EN SOFTWARE 111
5.1.3 COSTO SERVICIOS 112
5.1.4 COSTOS MATERIALES DE OFICINA 112
5.1.5 COSTO DE DESARROLLO 112
5.1.6 COSTO TOTAL 113
5.2 ANÁLISIS DE IMPACTO 113
5.2.1 ÁMBITO OPERATIVO 113
5.2.2 ÁMBITO SOCIAL 114
5.2.3 ÁMBITO TECNOLÓGICO 114
5.2.4 ÁMBITO AMBIENTAL 115
5.3 CONCLUSIONES 116

5.4 RECOMENDACIONES 117

5.5 BIBLIOGRAFÍA 118

ÍNDICE DE ILUSTRACIONES

ILUSTRACIÓN 1: Arquitectura a usarse.....6

ILUSTRACIÓN 2: Subprocesos del módulo Mantenimiento Mecánico 6

ILUSTRACIÓN 3: Logo Django Framework 12

ILUSTRACIÓN 4: Patrón de desarrollo MVC 18

ILUSTRACIÓN 5: Patrón MTV Django. 20

ILUSTRACIÓN 6: MariaDB 10.0 versión en debian.....57

ILUSTRACIÓN 7: Motores de almacenamiento en MySQL 5.5 Windows Xp 58

ILUSTRACIÓN 8: Motores de almacenamiento en MariaDB 5.6 mejorados.....58

ILUSTRACIÓN 9: Ciclo de vida de SCRUM 87

ILUSTRACIÓN 10: Interfaz de la distribución Debian 7 91

ILUSTRACIÓN 11: Interfaz del IDE Pycharm 3.0. 93

ILUSTRACIÓN 12: Interfaz de Umbrello 94

ILUSTRACIÓN 13: Interfaz de Pencil.....95

ILUSTRACIÓN 14: Sprint Blocklog.....102

ILUSTRACIÓN 15: Gráfico de tareas de la Metodología Scrum.....105

ILUSTRACIÓN 16: Gráfico de esfuerzo 105

ILUSTRACIÓN 17: Gráfico individual.....106

ILUSTRACIÓN 18: Diagrama general de caso de uso. 108

ILUSTRACIÓN 19: Diagrama de caso de uso del proceso Auditoría.....110

ÍNDICE DE TABLAS

TABLA 1: Patrones de desarrollo MVC y MTV 20

TABLA 2: Argumentos opcionales extra de DateField 25

TABLA 3: Opciones extra de FileField 25

TABLA 4: Opciones extras de FilePathField 26

TABLA 5: Opciones opcionales de FloatField 27

TABLA 6: Opciones de ManyToManyField 36

TABLA 7: Logo MariaDB 51

TABLA 8: API's para la implementación con MariaDB (MariaDB Org., 2015) 73

TABLA 9: Comparación de MariaDB y MySQL para versiones 5.x 80

TABLA 10: Equipo SCRUM 99

TABLA 11: Cuadro de prioridades asignadas al equipo 101

TABLA 12: Descripción de los caso de uso general 109

TABLA 13: Casos de uso para subproceso auditoria 110

TABLA 14: Costos en hardware. 111

TABLA 15: Costos de software. 111

TABLA 16: Costos servicios. 112

TABLA 17: Costos materiales de oficina. 112

TABLA 18: Costo de desarrollo. 112

TABLA 19: Costo total del análisis de costo. 113

TABLA 20: Impacto Ámbito económico. 113

TABLA 21: Mantenimientos programados con y sin sistema. 114

TABLA 22: Maquinaría con mantenimiento diario. 114

TABLA 23: Impacto ambiental 115

CAPÌTULO I

1 INTRODUCCIÓN

1.1 Introducción

Las herramientas tecnológicas nos sirven para desarrollar nuestras capacidades de ejercer de manera eficiente las actividades cotidianas que cumplimos, por lo tanto la solución informática que se le pueda dar a nuestra rutina, implica que debemos buscar alternativas existentes en el marcado para cumplir dicha actividad.

1.2 Problema

La tecnología pone a disposición marcos de desarrollo con el fin de desarrollar aplicaciones ágiles e intuitivas; diseñando una forma de acceso a la información ya sea móvil o web; podrán ser instaladas y/o cargadas en computadoras, en teléfonos celulares, en vehículos, en semáforos, en cajeros automáticos, etcétera. Con el fin de satisfacer oportunamente y en el menor tiempo posible las necesidades del cliente o de sí mismo. Muchos de estos marcos de desarrollo no son conocidos en nuestro medio; poco se sabe de sus ventajas, desventajas y cuando utilizarlos, por lo tanto no se los utiliza.

La tendencia en el desarrollo de soluciones marca un requerimiento que es el de acoplar una aplicación con una fuente de almacenamiento que ahorre recursos al sistema y con ello poder generar una interacción con base de datos que permita un sistema relacional y/o ser usada como puente SQL, No SQL y el tradicional orientado a objetos, como sistema de almacenamiento.

Por lo tanto es necesario que haya un estudio de estas tecnologías tanto el marco de desarrollo Django como el RDBMS MariaDB, por la importancia que van adquiriendo, sirviendo de referencia para estudiantes y/o autodidactas; y por qué no, en un futuro cercano ser tomadas en cuenta como materia de estudio ya que están disponibles sus códigos fuente y se las puede utilizar sin ningún problema de índole privativa; logrando captar más adeptos y profundizando en sus características, ventajas y desventajas; con el fin de lograr suplir sus desventajas en ventajas en el desarrollo de soluciones.

La Empresa Pública de Faenamiento y Productos Cárnicos de Ibarra pierde recursos al no contar con la automatización de las actividades de mantenimiento que facilite la intervención oportuna y adecuada de la maquinaria, que es utilizada en el proceso de faenamiento, por cuanto los operadores de dicha maquinaria informan de algún desperfecto cuando dicha maquinaria deja de funcionar.

1.3 Objetivos

1.3.1 General

Desarrollar una aplicación informática para la gestión del mantenimiento mecánico en industrias cárnicas utilizando el marco de desarrollo Django y la base de datos RDBMS MariaDB.

1.3.2 Específicos

- Documentar la investigación y el desarrollo del aplicativo.
- Estudiar el lenguaje de programación Python.
- Estudiar el marco de desarrollo de Django.
- Estudiar la base de datos RDBMS MariaDB.
- Determinar los beneficios y desventajas al utilizar Django con MariaDB.
- Estudiar los middlewares necesarios para la integración de Django y MariaDB.
- Aplicar al desarrollo la metodología SCRUM.
- Determinar los requerimientos para la implementación del módulo.
- Determinar la situación actual de los centros de faenamiento enfocado al inventario de la maquinaria.
- Probar las bondades de la tecnología propuesta en el desarrollo de una aplicación web de mantenimiento mecánico para empresas que se dedican a la industrialización cárnica.

1.4 Situación Actual de los centros de faenamiento.

La Empresa de Rastro de la ciudad de Ibarra es una empresa de carácter público administrada por el Municipio de Ibarra que brinda los servicios de faenamiento de animales mayores y menores.

El sacrificio de ganado en el domicilio ha pasado a ser considerado una práctica irregular, prohibida en muchos países; la utilización de maquinaria en este proceso ha ido mejorando de tal manera que pueda evitar el estrés de los animales, conservando así los componentes nutricionales de la carne.

En la empresa de faenamiento su maquinaria ha cumplido su etapa de vida útil y la empresa se ha visto obligada muchas veces a fabricar sus propios repuestos con el fin de mantener el proceso de faena, con esto mantener la calidad e higiene de la carne que estaría lista para el consumo con un proceso ordenado con la cadena de producción. Al no contar con la automatización del mantenimiento que coadyuven a evitar los desperfectos que se ocasionan en dicho proceso.

Proceso

Proceso de recepción: Se recibe a los animales según documentación de Guía de Movilización emitido por AGROCALIDAD, los animales son identificados, pesados y ubicados en los corrales, para cumplir con las medidas sanitarias de prevención, durante el tiempo que determine la ley.

Proceso de corralaje: Durante este proceso los animales cumplen un tiempo de estancia en el que son hidratados y pasan por un proceso de descanso y relajación muscular.

Proceso de arreo y duchado: Cumplido con los tiempos sanitarios acordados y habiéndose aceptado y cancelado las tasas correspondientes por el servicio de faenamiento de los animales que van al proceso de faenamiento, se trasladan a los mismos al duchado, para someterlos a una higienización inicial.

Proceso de noqueo: El noqueo del animal es físico mediante la aplicación o uso de una pistola neumática, se insensibiliza al animal a ser sacrificado para evitarles sufrimiento a la hora del degüello.

Proceso de izado: El animal es colgado de los cuartos traseros, en un gancho adherido a un riel para facilitar su movilidad en el proceso de desangrado y posteriores pasos del proceso de faena.

Proceso de sangrado y degüello: Se aplica un corte en las arterias del cuello del animal (estando boca abajo) para que el animal se desangre, la sangre es recogida en una canaleta especial, para su posterior procesamiento convirtiéndola en harina de sangre.

Proceso de corte de patas y cabeza: Se procede a cortar las patas y la cabeza del animal.

Proceso de desollado: Procedimiento que se realiza entre el cuero y la carnosidad, para facilitar el desollado del animal, proceso realizado mecánicamente.

Proceso de eviscerado: Procedimiento en el que se extrae los órganos internos de cada animal, llamados víscera.

Proceso de fisurado: Incisión longitudinal del esternón y la columna vertebral, que se realiza sobre el animal faenado, mediante una sierra eléctrica.

Proceso de inspección veterinaria post mortem: La carne de los animales faenados, son revisados por el veterinario para determinar su integridad orgánica y estado sanitario.

Proceso de higiene y desinfección: Es la aplicación de agua a presión y/o ácido orgánico sobre las superficies corporales, para desinfectar al animal de posibles contaminaciones propias del manipuleo y el eviscerado. (Empresa Pública Metropolitana de Rastro, 2014)

1.5 Alcance

1.5.1 De la Investigación

La presente investigación analizará las nuevas herramientas tecnológicas para el desarrollo de aplicaciones web con el marco de desarrollo Django donde se explicará el patrón de desarrollo MVC dando a conocer las características, bondades y desventajas de la misma; creando una aplicación CRUD con Django, que se desarrolla con el lenguaje de alto nivel Python, proporcionando una breve introducción de sus características, ventajas y desventajas.

Para la integración se analizará la base de datos MariaDB, donde se dará una introducción y se darán a conocer características, ventajas y desventajas así como sus requerimientos y compatibilidad con Django para un desarrollo de soluciones web y móvil.

Una vez lograda la investigación, y para el desarrollo del aplicativo se integrarán estas dos nuevas tecnologías haciendo uso de la metodología de desarrollo SCRUM, una metodología vanguardista y de fácil aplicación para desarrollo ágil, constante e ininterrumpido con el uso de eventos propios de la metodología.

1.5.2 Del Aplicativo

La aplicación corresponde a un sistema web de mantenimiento mecánico en el cual se creará un historial de mantenimiento de la maquinaria y demás piezas mecánicas utilizadas en dicha actividad para de esta manera ayudar al personal a coordinar sus actividades dentro de la empresa en especial en la planta de operaciones.

Mantener reportes del stock de repuestos y accesorios e informando a qué maquinaria tiene que realizarse el mantenimiento en un periodo, indicando costo por repuesto y/o mano de obra si fue requerido.

El módulo también contará con los niveles de seguridad de usuario asignándoles roles o perfiles de usuario.

La solución que se va a implementar tendrá licencia open source; reduciendo los riesgos de restricción de uso y modificación, facilitando su escalabilidad y mantenimiento.

Arquitectura a usarse

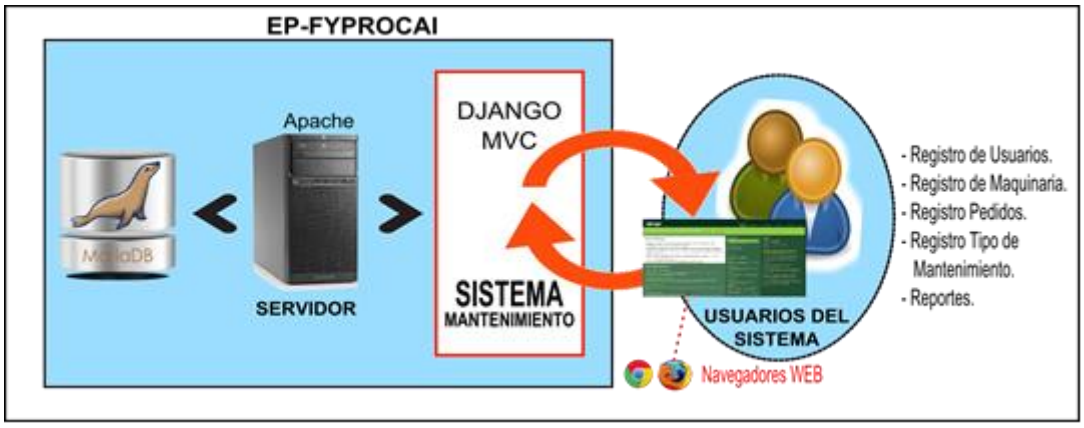


ILUSTRACIÓN 1: Arquitectura a usarse

Fuente: Autor

En la ilustración 1 se indica la arquitectura web, y la estructura básica funcional que tendrá el sistema. Este se ejecutará a través de un servidor y los usuarios podrán acceder al sistema por medio de navegadores web (Mozilla Firefox, Google Chrome, Opera) instalado en su Computador desde la cual se podrá acceder desde diferentes sistemas operativos o distribuciones GNU/Linux o MS. Windows.

Esquema del aplicativo

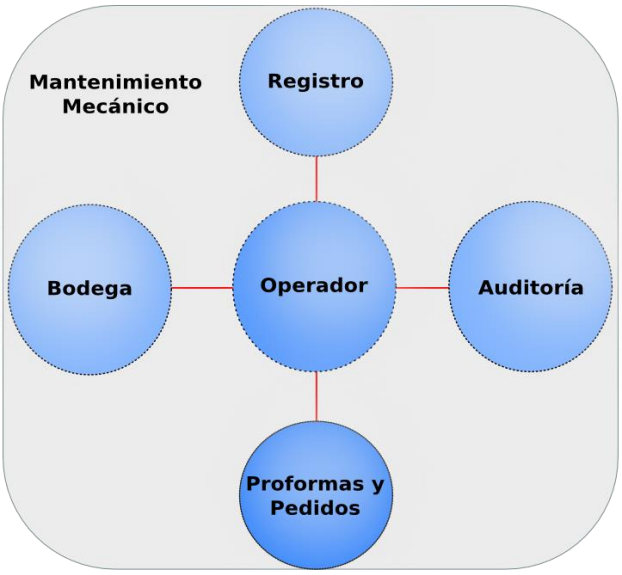


ILUSTRACIÓN 2: Subprocesos del módulo Mantenimiento Mecánico

Fuente: Autor

En la Ilustración 2 se muestran los subprocesos del módulo Mantenimiento Mecánico y se detallan a continuación.

Registro

Este subproceso registrará la información necesaria para poder realizar los cronogramas de mantenimientos y de reclamos de funcionamiento de la maquinaria.

Se tendrán los formulario de ingresos y modificación de todos los maestros para la realización de los cronogramas de mantenimiento: materiales utilizados, mano de obra (si fue necesario) y tiempos de ejecución de la solución o del trabajo requerido; estos datos se actualizan.

Se registrarán las plantas con las que cuente la empresa si están separadas o si están en la misma planta.

Proformas y Pedidos

En este subproceso se registraran las proformas solicitadas para trabajos especializados y que en el taller no se las puede realizar, así como para los pedidos se propondrá un formulario para solicitar material o accesorio del taller.

Operador

Será encargado de mantener el historial de la maquinaria y reparaciones de toda la planta sea mecánica, eléctrica o civil; con un registro de mantenimientos correctivo y preventivo con su documento de informe de estado del mantenimiento, tiempos de ejecución y responsable.

Bodega

Este subproceso se encargará de proporcionar la información del existente de materiales, también el de herramientas disponibles y alertar cuando el existente de material esté por terminar para una próxima provisión.

Auditoría

Este subproceso se encargará de realizar toda la auditoría de asignación y cuanto material o para qué fue utilizado previo a la autorización de la autoridad.

Se mantendrá un control de todos los cambios que se realicen en el sistema, datos del usuarios que ingresa, que acciones realizó.

Este módulo generará los informes pertinentes para auditoría interna para su revisión y control.

1.6 Justificación

1.6.1 De la Investigación

Las herramientas tecnológicas propuestas para ser estudiadas han tenido un crecimiento importante en los últimos años ya que facilitan el desarrollo de las aplicaciones web; en la actualidad una herramienta fácil de utilizar y debidamente documentada gana terreno y adeptos.

Los estándares de desarrollo con el tiempo han ido cambiando y las empresas deben cambiar sus formas de organizarse ya que requieren equipos multidisciplinarios que colaboren en el desarrollo de la aplicación.

La empresa en la que se aplicara el sistema busca procesos centralizados y seguros, era necesario ir creando una base de información única que vaya creciendo de acuerdo a las necesidades de la empresa, manejar un mismo lenguaje y en el nombre de las herramientas, materiales y maquinaria e implementar un estándar.

Con esta solución informática la empresa contará con sistema propio y no será necesario que para suplir dicha necesidad lo haga por medio de terceros.

Los conocimientos técnicos adquiridos en las aulas nos ayudaran a analizar las herramientas presentadas ayudaran a llevar a cabo el desarrollo y la implementación del sistema.

1.6.2 Del aplicativo

Las empresas que se dedican a este tipo de servicios mantienen un trabajo muy activo y cuando está en proceso de faenamiento es cuando más cuidado se debe tener para mantener la maquinaria funcionando en perfectas condiciones y así evitar inconvenientes en el proceso del producto.

Los centros de rastro, están siendo modernizados y hasta reconstruidos en el país por cuanto los organismos de salud pública piden que se verifique los procesos mecánicos y de salubridad de este tipo de empresas, por lo tanto mejoran la calidad del producto terminado y cuidan la cadena de frío implementando un estándar internacional, con el fin de en un futuro cercano poder exportar cortes selectos.

CAPÌTULO II

2 MARCO TEÓRICO

2.1 Python

El lenguaje para programación por excelencia de todos las personas que deseen aprender a programar o desarrollar software de calidad y factible ya que es multiplataforma y fácil de comprender, ya que mantiene persistente la codificación con las iteraciones, declaraciones e inicializaciones.

2.1.1 ¿Qué es Python?

Lenguaje de programación desarrollado por Guido Van Rossum en los finales de los 80 y principios de los 90; muy parecido al lenguaje Perl pero más amigable con el desarrollador y con el entorno de ejecución. Lenguaje cuya característica principal es que es un lenguaje interpretado o de como también se lo llama script que se ejecuta con un intérprete y no requiere de compilador ya que no requiere de ser transformado a lenguaje máquina.

2.1.2 Características

Lenguaje interpretado o de script

Se ejecuta en un entorno interpretado no compilado la ventaja es su fácil portabilidad y flexibilidad; también posee la capacidad dependiendo del entorno de ejecución ser semi interpretado ya que se lo puede embeber con otros lenguajes como java, ruby entre otros.

Tipado dinámico

La característica más debatible por los desarrolladores de lenguajes compilados es que no se requiere declarar variables con el tipo de dato ya que trabaja con el valor que se le asigne y es delimitado por comillas simples o dobles, puntos, corchetes, paréntesis, llaves, clases u otras.

El tipado no permite cambiar el dato de la variable que desde un comienzo se le asignó; solo caben las transformaciones propias de enteros a cadena o viceversa si el valor es el que dice ser.

Multiplataforma

Al ser un lenguaje interpretado o semi interpretado su ejecución es más flexible en entornos como UNIX, GNU/Linux, Linux, DOS/Windows, MAC OS, etc. ya que ofrece un sinnúmero que librerías que los sistemas vienen preconfigurados y la configuración es mínima.

Orientado a Objetos

P.O.O. Es un paradigma de programación que consiste en dar solución a un problema desde clases y objetos.

Python permite programación interpretativa, funcional, y programación orientada a aspectos.

2.1.3 Frameworks de Python

Una lista de frameworks están propuestos en la página oficial de Python org. Teniendo en cuenta que todos los marcos de desarrollo estén afianzados en el paradigma DRY (Don't Repeat Yourself) o en español sería NTR (No te repitas); a continuación veremos los frameworks más populares como son:

Grok

Registrado por la organización Python en el 2011; construido en base a las librerías de un framework Zope organización que adopto una versión “mejorada” si cabe el termino; orientado al desarrollo ágil de las aplicaciones web. Para profundizar más sobre el tema recomendamos visitar la página web del framework. (http://grok.zope.org).

Pylons

Registrado por la organización Python en el 2010; un framework que por sus características adopta las mejores ideas de estructura de ruby, python y perl, haciendo de este un framework para el desarrollo rápido de las aplicaciones ya que es ligero en flexibilidad. Para mayor información para los autodidactas que deseen profundizar en el framework sería (http://pylonshq.com/)

TurboGears

Registrado por la organización Python en el 2014; un framework para desarrollo web y que ha tenido avances muy importantes en ya que presenta una estructura adaptable para trabajar con base de datos no relacionales y con relacionales haciendo del framework escalable y amigable con el desarrollador. Para mayor información para los autodidactas que deseen profundizar en el framework pueden visitar. (Turbogears.org, 2010)

Django

Registrado por la Organización Python en el 2014; un framework para desarrolladores perfeccionistas como se lo ha denominado ya que es un marco que garantiza las mejores aplicaciones web porque para desarrollar hace que su estructura requiera menos código y los resultados son aplicaciones elegantes.

Django ha sido escogido para nuestra investigación ya que presenta las características propias ya es su paradigma de desarrollo en el mapeo de base de datos relacional y la capacidad conectar múltiples base de datos.

2.2 FRAMEWORK DJANGO

ILUSTRACIÓN 3: Logo Django Framework

Fuente: projectdjango.org

2.2.1 ¿Qué es un framework?

Un framework es un conjunto de herramientas, artefactos y estándares que dan funcionalidad extra a una aplicación y facilita su comprensión y desarrollo; por lo tanto estos frameworks sirven para acortar el tiempo de desarrollo y reducir el costo de la inversión que esto implica; no todos los desarrolladores están de acuerdo con estas herramientas ya que las llaman fuentes de mediocridad por que los programadores más experimentados los hacen ellos solos desde cero y por lo tanto es un motivo de polémica hasta hoy en día el uso o no de un framework.

Los frameworks tienen como objetivo ofrecer una funcionalidad definida, auto contenido, siendo construidos usando patrones de diseño, y su característica principal es su alta cohesión y bajo acoplamiento. Para acceder a esa funcionalidad, se construyen piezas, objetos, llamados objetos calientes, que vinculan las necesidades del sistema con la funcionalidad que este presta. [3W02]

Adicionalmente como dato real se podría decir que: en la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte de todos los graduados del año 2013 que presentaron proyectos de desarrollo propusieron frameworks.

2.2.2 Historia

Django nace en el año 2003 en un medio de comunicación escrita *Lawrence Journal-World*, es desarrollado por Adrián Holovaty y Simmon Willison; en 2005 lanzaron al público la primera versión de Django bajo la licencia BSD; mediante la Fundación de Software Django (DSF), algo restrictivo de la licencia es que se debe pedir autorización para crear una bifurcación (fork); su nombre proviene del guitarrista de jazz Jean "Django" Reinhardt. (Rossum, 2009)

2.2.3 Definición

Django es un framework de código libre que permite el desarrollo web ágil y utilizando el menor número de código posible; requiere también de un nivel de conocimientos básicos del lenguaje de programación Python.

2.2.4 Características

¿Qué lo hace uno de los mejores?

En el análisis del open source y con el avance que está dando la implementación de soluciones tecnológicas basadas en la información o TIC; es un lenguaje más adaptable porque rompe el paradigma de programación modelo-vista-controlador y lo relaciona con modelo-plantilla-controlador; su política es siempre la rigurosidad de Python y con el termino DRY (Don't repeat yourself) en español “no te lo repitas“.

2.2.5 Ventajas y Desventajas

Un framework capaz de construir en el menor tiempo posible las aplicaciones con las características en desarrollo, producción y pruebas. Contiene un servidor incluido para la ejecución en desarrollo y se lo puede desactivar para integrarlo al desarrollo en producción con los servidores de aplicaciones como Apache. Cuenta con un mapeo objeto relacional que se utiliza en la definición de los modelos y en el desarrollo ayuda tener casi cero interacción directa con la base de datos ya que Django administra la base de datos con las ordenes de los comandos embebidos en el framework.

Django se tiene un manejador de url's que lo denominan “pretty” y ayuda a direccionar las páginas que desea mostrar o generar. Plantillas heredadas es otra de las ventajas apremiantes del framework ya que no es necesario repetir el código HTML en cada página sino que se declaran bloques y se los utiliza de manera dinámica.

Una de las desventajas es un lenguaje poco conocido por que la mayoría de desarrolladores en nuestro medio que se enfocan a las aplicaciones con lenguajes ensamblados y no interpretados y eso conlleva a que su desarrollo sea más extenso.

2.2.6 Sistemas de Seguridad incluidos

En Django la seguridad en la ejecución de las aplicaciones guarda algunas características para evitar ciertos ataques externos y evitar problemas de integridad pero sin embargo “Nunca -- bajo ninguna circunstancia -- confíes en datos enviados por un navegador.”

Django intenta mitigar esta dificultad. Está diseñado para protegerte automáticamente de muchos de los errores de seguridad comunes que cometen los nuevos (e incluso los experimentados) desarrolladores Web. Aun así, es importante entender de qué se tratan dichos problemas, cómo es que Django te protege, y – esto es lo más importante – los pasos que puedes tomar para hacer tu código aún más seguro.

Inyección de SQL

La inyección de código SQL es un ataque en el cual se inserta código malicioso en las cadenas que posteriormente se pasan a una instancia de SQL Server para su análisis y ejecución. (Microsoft, 2015) Por ejemplo, se escribe una función para obtener una lista de información de contacto desde una página de búsqueda. Para prevenir que los spammers lean todas las direcciones de email en nuestro sistema, vamos a exigir al usuario que escriba el nombre de usuario del cual quiere conocer sus datos antes de proveerle la dirección de email respectiva.

Cross-Site Scripting (XSS)

El Cross-site scripting (XSS) (Scripting inter-sitio), puede encontrarse en aplicaciones Web que fallan a la hora de escapar en forma correcta contenido provisto por el usuario antes de renderizarlo en HTML. Esto le permite a un atacante insertar HTML arbitrario en tu página Web, usualmente en la forma de etiquetas <script>.

Los atacantes a menudo usan ataques XSS para robar información de cookies y sesiones, o para engañar usuarios y lograr que proporcionen información privada a la persona equivocada (también conocido como phishing).

Las plantillas Django han evitado este comportamiento debido a que esto cambia sutilmente algo que debería ser un comportamiento no complejo (la visualización de variables). Es un asunto no trivial y una decisión de compromiso difícil de evaluar. Agregando comportamiento implícito y oculto va contra los ideales de base de Django (y los de Python), pero la seguridad es igual de importante.

Cross-Site Request Forgery

La Cross-site request forgery (CSRF) (Falsificación de peticiones inter-sitio) sucede cuando un sitio Web malicioso engaña a los usuarios y los induce a visitar una URL desde un sitio ante el cual ya se han autenticado – por lo tanto saca provecho de su condición de usuario ya autenticado.

Session Forging/Hijacking

El framework de sesiones de Django simplemente no permite que las URLs contengan sesiones.

No almacenar datos en cookies en forma directa; en cambio, almacena un identificador de sesión que esté relacionado a datos de sesión almacenados en el back-end.

Si usas el framework de sesiones incluido en Django (o sea request.session), eso es manejado en forma automática. La única cookie que usa el framework de sesiones es un identificador de sesión; todos los datos de la sesiones se almacenan en la base de datos.

Recuerda escapar los datos de la sesión si los visualizas en la plantilla. Revisa la sección previa sobre XSS y recuerda que esto se aplica a cualquier contenido creado por el usuario así como a cualquier dato enviado por el navegador. Debes considerar la información de sesiones como datos creados por el usuario.

A pesar de que es prácticamente imposible detectar a alguien que se ha apropiado de un identificador de sesión, Django incluye protección contra un ataque de sesiones de fuerza bruta. Los identificadores de sesión se almacenan como hashes (en vez de números secuenciales) lo que previene un ataque por fuerza bruta, y un usuario siempre obtendrá un nuevo identificador de sesión si intenta usar uno no existente, lo que previene la captura o el set de la cookie.

Inyección de cabeceras de email

Conocidos como Man-in-the-middle, se captura datos de sesión mientras estos viajan por la red cableada o inalámbrica.

Session forging (Falsificación de sesión), Se usa un identificador de sesión para simular ser otro usuario.

Session fixation (fijación de sesión), se burla a usuario y logra cambiar un nuevo valor o cambiar el valor existente del identificador de su sesión.

Session poisoning (envenenamiento de sesión), inyecta datos potencialmente peligrosos en la sesión de un usuario; usualmente a través de un formulario.

Para reducir la ejecución de este tipo de engaños el framework no permite que las url's contengan sesiones de usuarios.

No almacena datos en cookies en forma directa; en cambio, almacena un identificador de sesión que esté relacionado a datos de sesión almacenados en el back-end (o lo que se podría llamar la parte que no ve el usuario).

Django usa las sesiones (o sea request.session), eso es manejado en forma automática. La única cookie que usa el framework de sesiones es un identificador de sesión; todos los datos de la sesiones se almacenan en la base de datos.

Recuerda escapar los datos de la sesión si los visualizas en la plantilla. Revisa la sección previa sobre XSS y recuerda que esto se aplica a cualquier contenido creado por el usuario así como a cualquier dato enviado por el navegador. Debes considerar la información de sesiones como datos creados por el usuario.

Exposición de mensajes de error

“La filosofía de Django es que los visitantes al sitio nunca deben ver mensajes de error relacionados a una aplicación. Si tu código genera una excepción no tratada, un visitante al sitio no debería ver un traceback completo – ni ninguna pista de fragmentos de código o mensajes de error (destinados a programadores) de Python. En cambio, el visitante debería ver un amistoso mensaje “. (Holovaty Adrian, 2007)

2.2.7 Arquitectura MVC

¿Qué es una Arquitectura?

"La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución".[1]

¿Qué es un patrón?

Es lo pragmático de la programación la definición de los patrones de desarrollo ya que sin ellos se vería un desarrollador forzado a cambios que lleven mucho tiempo y costo; por lo tanto no se optimizaría el desarrollo de una solución y no es más que la separación de la lógica de negocios y los datos.

Patrón MVC

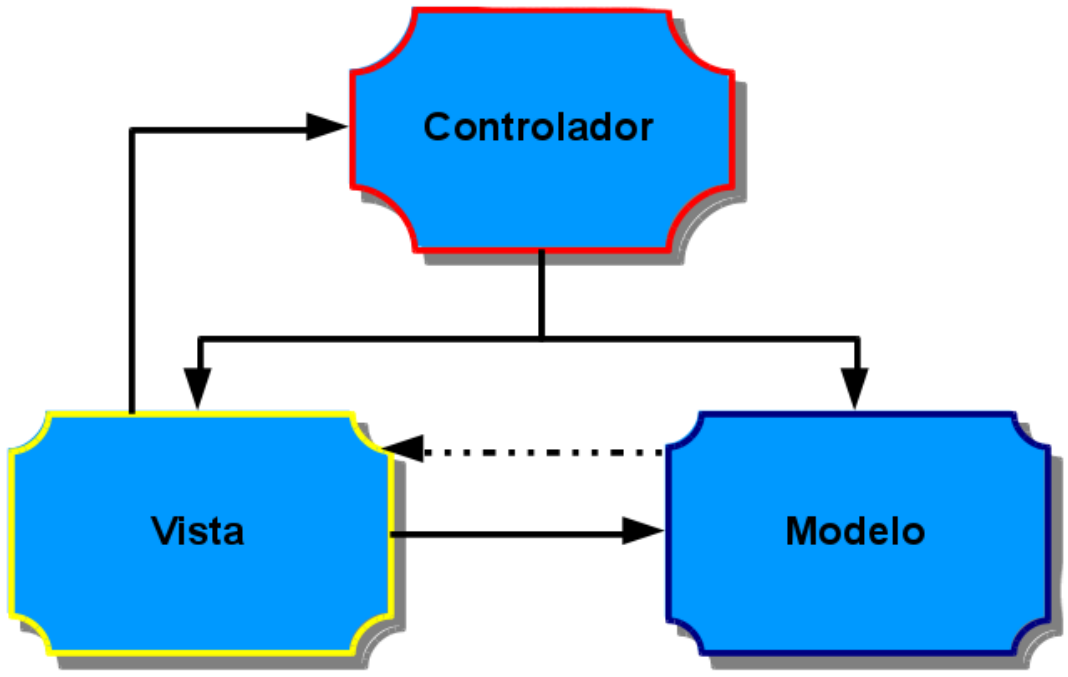


ILUSTRACIÓN 4: Patrón de desarrollo MVC

Fuente: Autor

[1] Arquitectura del Software en la IEEE STD 1471-2000

Es un paradigma de la programación que viene con la Arquitectura de Software orientada a patrones; estos componentes relacionados independizan y separan la lógica de los datos de una aplicación para representar la información con el usuario.

Modelo

Este componente se encarga de las operaciones lógicas y de gestionar el manejo de información como consultas y su función es enviar a la vista la información redundante solicitada por los usuarios; esta información es canalizada por el controlador y por lo tanto recibe del controlador las “ordenes” de actualización y manipulación.

Vista

Este componente muestra la información y la lógica del negocio del modelo; al usuario en una interfaz de usuario.

Controlador

Este componente actúa con cada petición que realiza el usuario o con alguna automatización que le envíe el componente modelo o actúe en la vista con la manipulación del controlador por ejemplo el desplazamiento con el scroll de la información de una base de datos.

Este componente es el intermediario entre el componente vista y el componente modelo.

2.2.8 Estructura MVC en el Framework Django

En Django el paradigma MVC ha evolucionado y hace una excepción en la intermediación entre el componente modelo y la vista ya que el framework se encarga de ello; se puede decir que Django hace la función del controlador; sin alterar el concepto del patrón MVC.

Django sigue este patrón MVC; por lo que se denominaría un Framework MVC.

ILUSTRACIÓN 5: Patrón MTV Django.

Fuente: projectdjango.org

Entendiendo que el controlador (C) es gestionado por el propio framework y la mayor parte de la solución se encuentra en los modelos, plantillas y vistas; Django se ha redefinido como un framework MTV.

Modelo, este componente se encarga de gestionar el acceso a datos: cómo acceder a ella, cómo validar, cómo se comporta y las relaciones entre los grupos de datos.

Template (Plantilla), este "componente" se encarga de la presentación; contiene las decisiones relacionadas con la presentación en forma de página web o algún otro documento.

Vista, este componente es una capa de lógica del negocio, hace una intermediación entre los modelos y las plantillas; contiene la lógica de la porción de datos del modelo que serán enviadas a la plantilla(s).

TABLA 1: Patrones de desarrollo MVC y MTV

PATRONES DE DESARROLLO	
MVC	**MTV**
Modelo	Modelo
Vista	Vista y Plantilla*
Controlador	Django

Fuente: Autor

2.2.9 Taxonomía de Django

Esquema de carpetas y archivos en Django

Cuando se ejecuta el comando *django-admin.py startproject simmec* en la carpeta que va contener el o los proyectos; aparecerá el siguiente esquema.

simmec/

- *__init__.py*
- *manage.py*
- *settings.py*
- *urls.py*
- *apps/*
- *static/*
- *css/*
- *img/*
- *js/*
- *font/*
- *media/*
- *videos/*
- *fotos/*
- *templates/*

Dónde:

El archivo __init__.py, es requerido para que Python reconozca este archivo y la carpeta que lo contiene como un paquete de aplicaciones o el proyecto.

El archivo manage.py, para se lo utiliza para ejecutar comandos en la terminal que afectan al proyecto.

El archivo settings.py, de opciones y/o configuraciones para este proyecto de Django.

El archivo urls.py, de las declaraciones de las direcciones URL para un proyecto de Django; una "tabla de contenidos"(tupla) de tu sitio hecho con Django.

Los archivos anteriormente mencionados se crean por defecto cuando se crea un proyecto en Django.

Las carpetas que vamos a ver a continuación son las que el desarrollador debe crear para darle un orden al proyecto y para que el controlador que en este caso es Django gestione bien el proyecto

La carpeta apps/, es para contener las aplicaciones de ese proyecto tantas cuantas sean necesarias; la carpeta es opcional por eso que el framework no las genera; sino más bien las propongo para darle orden al proyecto.

La carpeta static/, el archivo de configuración del proyecto (settings.py) ya hace referencia a esta carpeta que contendrá todos los archivos estáticos del proyecto cabe recalcar que los archivos de audio y vídeo.

Fotos, documentos, tipos de letras, plantillas html, etc. son también estáticos eso depende de la jerarquía y orden que se le quiera dar al proyecto.

La carpeta media/, es la carpeta que contendrá los archivos multimedia de las aplicaciones que se generen.

La carpeta templates/, es la carpeta que contiene los archivos de html de la aplicación.

Lo antes detallado es como Django propone como se deben llamar las carpetas y el lugar se ha criterio técnico de los lugares donde estarán físicamente; la jerarquía propuesta es una apreciación personal es indistinto como relacione su jerarquía lo importante es hacer la referencia correcta.

2.2.10 Estructura de una aplicación Django

Cuando se genera un proyecto la filosofía de Django es dividirlo en varias aplicaciones.

simmec/

…

apps/

miaplicacion/

migracion/

__init__.py

admin.py

models.py

test.py

views.py

En el directorio miaplicacion/ que se genera una subestructura con directorio y archivos que contiene el core de la apicación.

La particularidad del framework es la capacidad de contener en la estructura principal unas subestructuras que dan robustez al proyecto.

Nótese que el directorio templates/ no existe ya que será creada un nivel más alto en la jerarquía del proyecto.

2.3 Palabras Reservadas

2.3.1 Campos

La capa del modelo necesita de las palabras reservadas para generar la base de datos relacional como las necesitemos hacer.

Restricciones en el nombre de los campos

El framework tiene dos restricciones en el nombre de los campos:

Un nombre de campo no puede ser una palabra reservada de Python, porque eso ocasionaría un error de sintaxis en Python, por ejemplo:

class Miclase(models.Model):

pass = models.IntegerField() # donde pass es una palabra reservada de Python

El nombre del campo no puede contener dos o más guiones bajos consecutivos, por la forma que Python reserva la sintaxis de las consultas de búsqueda de, por ejemplo:

class Example(models.Model):

e__str__ = models.IntegerField() # __str__ tiene dos guiones bajos!

Las palabras reservadas no pueden dar mayor problema por cuanto no se requiere que los campos no tienen que llamarse como las columnas de las tablas a generar; aunque es una mejor práctica para el desarrollo comprensivo e intuitivo.

Las palabras reservadas de SQL, como join, where, o select, son permitidas como nombres de campo, dado que Django al generar las tablas la encierra entre comillas.

Cada campo del modelo debe ser una instancia de campo apropiada. Django usa los tipos de clase campo (Field) para determinar algunas cosas:

- El tipo de columna de la base de datos (ej.:, INTEGER, VARCHAR).
- El widget a usar en la interfaz de administración de Django, si vas a usarla (ej., <input type="text">, <ul>).
- Los requerimientos mínimos de validación.

A continuación, una lista completa de las clases de campo, ordenadas alfabéticamente. Los campos de relación (ForeignKey, ManytoMany, etc.) se tratan a continuación.

2.3.2 Auto Field

Un ***IntegerField*** que se incrementa automáticamente de acuerdo con los IDs disponibles. Se generan automáticamente cuando se genera el modelo en la base de datos; son la clave primaria automáticamente al modelo si no especificas una clave primaria.

2.3.3 BooleanField

Un campo de Verdadero y Falso hay que tener en cuenta que en MySQL los campos se guardan como un TyNYINT y se le da un valor de 1 o 0.

2.3.4 CharField

Un campo string, para cadenas cortas o largas. Para grandes cantidades de texto, usa ***TextField*** requiere un argumento extra, max_length, que es la longitud máxima (en caracteres) del campo. Esta longitud máxima es reforzada a nivel de la base de datos y en la validación de Django.

2.3.5 CommaSeparatedIntegerField

Un campo de enteros separados por comas. Igual que en CharField, se requiere el argumento max_length.

2.3.6 DateTimeField

Un campo de fecha y hora. Tiene las mismas opciones extra que DateField.

TABLA 2: Argumentos opcionales extra de DateField

Argumento	Descripción
auto_now	Asigna automáticamente al campo el valor que marca el reloj en el instante que guarda el objeto. (ej.:"última modificación"). (Django Software Fundation, 2016)
auto_now_add	Asigna automáticamente al campo el valor que marca el reloj en el instante que se crea el objeto.(ej.: "fecha de envío"). (Django Software Fundation, 2016)

Fuente: Autor

2.3.7 FileField

TABLA 3: Opciones extra de FileField

Argumento	Descripción
upload_to	Una ruta del sistema de archivos local que se agregará a la configuración de MEDIA_ROOT para determinar el resultado de la función de ayuda get_<fieldname>_url()

Fuente: Autor

Al momento de cargarla el framework reemplaza su nombre y le asigna fecha y hora con una función propia de Python strftime.

El uso de un FileField o un Image Field en un modelo; deben cumplir los pasos requeridos como tener en el archivo de configuración la ruta de la carpeta

MEDIA_ROOT; que por rendimiento de la aplicación no se almacenan en la base de datos:

Todo lo que se va a almacenar en la base de datos es la ruta al archivo (relativa a MEDIA_ROOT). Si se prefiere usar la función get_<fieldname>_url provista por Django. Por ejemplo, si una imagen se llama utn_logo, puedes obtener la URL absoluta a tu imagen en una plantilla con {{object.get_utn_logo_url }}.

Por ejemplo, digamos que tu MEDIA_ROOT es '/home/danilo/simmec/media', y upload_to es 'fotos/%Y/%m/%d'. La parte '%Y/%m/%d' de upload_to es formato strftime; '%Y' es el año en cuatro dígitos, '%m' es el mes en dos dígitos, y '%d' es el día en dos dígitos. Si subes un archivo el 9 de mayo de 2014, será guardado en /home/danilo/simmec/media/fotos/2014/05/09.

Si se requiere recuperar el nombre en disco del archivo subido, o una URL que se refiera a ese archivo, o el tamaño del archivo, puedes usar los métodos get_FIELD_filename(), get_FIELD_url(), y get_FIELD_size().

2.3.8 FilePathField

Un campo cuyas opciones están limitadas a los nombres de archivo en un cierto directorio en el sistema de archivos. Tiene tres argumentos especiales, que se muestran en la Tabla B-3.

TABLA 4: Opciones extras de FilePathField

Argumento	Descripción
Path *requerido*	La ruta absoluta en el sistema de archivos hacia el directorio del cual esté FilePathField debe tomar sus opciones (ej.: "/home/images")
Match *opcional*	Una expresión regular como string, que FilePathField usará para filtrar los nombres de archivo. Observar que la regex será aplicada al nombre de archivo base, no a la ruta completa (ej.: "foo.*\.txt^", va a comparar con un archivo llamado foo23.txt, pero no con bar.txt o foo23.gif).
Recursive *opcional*	True o False. El valor por omisión es False. Especifica si deben incluirse todos los subdirectorios de path.

Fuente: Autor

Por supuesto, estos argumentos pueden usarse juntos.

El único 'gotcha'[2] potencial es que match se aplica sobre el nombre de archivo base, no la ruta completa. (Holovaty Adrian, 2007) De esta manera, este ejemplo:

FilePathField(path="/home/images", match="tec.", recursive=True)*

La sentencia va a comparar (matchear) con /home/images/tec.gif pero no con /home/images/tec/bar.gif porque el match se aplica al nombre de archivo base (tec.gif y bar.gif).

2.3.9 FloatField

El número de punto flotante, representado en Python por una instancia de float. Tiene dos argumentos requeridos.

TABLA 5: Opciones opcionales de FloatField

Argumento	Descripción
max_digits	La cantidad máxima de dígitos permitidos en el número.
decimal_places	La cantidad de posiciones decimales a almacenar con el numero

Fuente: Autor

Por ejemplo, para almacenar números hasta 9999 con una resolución de dos decimales, hay que usar:

models.FloatField(..., max_digits=6, decimal_places=2)

IntegerField.- Un entero.

2.3.10 PhoneNumberField

Un CharField que verifica el valor de un teléfono válido este E.C. (ej.: xxx-xxx-xxxx)

[2] Un gotcha se ha convertido en un término de programación que se caracteriza por gastar bromas por mostrar un comportamiento que es diferente a lo que se espera.

Si se necesita representar teléfonos de otros países, consulta el paquete *django.contrib.localflavor* para ver si ya están incluidas las definiciones del país que se requiera.

PositiveIntegerField.- Similar a IntegerField, pero debe ser positivo.

2.3.11 PositiveSmallIntegerField

Similar a PositiveIntegerField, solo permite valores por debajo de un límite. El valor máximo permitido para estos campos depende de la base de datos, pero como las bases de datos tienen un tipo entero corto de 2 bytes, el valor máximo positivo usualmente es 65,535.

SmallIntegerField.- Similar a IntegerField, que permite valores en un cierto rango dependiente de la base de datos (usualmente -32,768 a +32,767).

TextField.- Un campo de texto de longitud ilimitada.

TimeField.- Un campo de hora. Acepta las mismas opciones de autocompletación de DateField y DateTimeField.

URLField.- Un campo para una URL. Si la opción verify_exists es True (valor por omisión), se chequea la existencia de la URL dada (la URL carga y no da una respuesta 404).

Como los otros campos de caracteres, URLField toma el argumento max_length. Si no se especifica, el valor por omisión es 200.

2.4 Opciones Universales de Campo

Los siguientes argumentos están disponibles para todos los tipos de campo. Todos son opcionales.

2.4.1 Null

Si está en True, Django almacenará valores vacíos como NULL en la base de datos. El valor por omisión es False.

Los valores de string nulo siempre se almacenan como strings vacíos, no como NULL. Utiliza null=True solo para campos que no sean cadenas, como enteros, booleanos y fechas.

En los dos casos, también necesitarás establecer blank=True si deseas permitir valores vacíos en los formularios, ya que el parámetro null solo afecta el almacenamiento en la base de datos.

Tomar en cuenta que utilizar null en campos basados en cadenas como CharField y TextField. Si un campo basado en string tiene null=True, eso significa que tiene dos valores posibles para "sin datos": NULL y el string vacío. En la mayoría de los casos, esto es redundante; la convención de Django es usar el string vacío, no NULL.

2.4.2 Blank

Si está en True, está permitido que el campo esté en blanco. El valor por omisión es False.

Observar que esto es diferente de null. null solo se relaciona con la base de datos, mientras que blank está relacionado con la validación. Si un campo tiene blank=True, la validación del administrador de Django permitirá la entrada de un valor vacío. Si un campo tiene blank=False, es un campo requerido.

2.4.3 Choices (Escoger)

Un grupo de datos (ej.: una lista, tupla, u otro objeto iterable de Python) de dos tuplas para usar como opciones para este campo.

Si esto está dado, la interfaz de administración de Django utilizará un cuadro de selección en lugar del campo de texto estándar, y limitará las opciones a las dadas.

Una lista de opciones se ve así:

```
CIUDADES_EN_ECUADOR_CHOICES =(
('UIO', 'Quito'),
('GYE', 'Guayaquil'),
('IBR', 'Ibarra'),
```

('STOD', 'Santo Domindo de los Tsachilas'),

('TCN', 'Tulcan'),

)

El primer elemento de cada tupla es el valor real a ser almacenado. El segundo elemento es el nombre legible por humanos para la opción.

La lista de opciones puede ser definida también como parte de la clase del modelo:

class Escoger(models.Model):

GENERO_CHOICES = (

('M', 'Masculino'),

('F', 'Femenino'),

)

gender = models.CharField(maxlength=1, choices=GENERO_CHOICES)

o fuera de la clase del modelo:

GENERO_CHOICES = (

('M', 'Masculino'),

('F', 'Femenino'),

)

class Foo(models.Model):

gender = models.CharField(maxlength=1, choices=GENERO_CHOICES)

Para cada campo del modelo que tenga establecidas choices, Django agregará un método para recuperar el nombre legible por humanos para el valor actual del campo.

2.4.4 db_column

El nombre de la columna de la base de datos a usar para este campo. Si no está dada, Django utilizará el nombre del campo. Esto es útil cuando estás definiendo un modelo sobre una base de datos existente.

Si tu nombre de columna de la base de datos es una palabra reservada de SQL, o contiene caracteres que no están permitidos en un nombre de variable de Python (en particular el guión simple), no hay problema. Django quotea los nombres de columna y tabla detrás de la escena.

2.4.5 db_index

Si está en True, Django creará un índice en la base de datos para esta columna cuando cree la tabla (es decir, cuando ejecute manage.py syncdb).

Default.- El valor por omisión del campo.

Editable.- Si es False, el campo no será editable en la interfaz de administración o vía procesamiento de formularios. El valor por omisión es True.

Help_text.- Texto de ayuda extra a ser mostrado bajo el campo en el formulario de administración del objeto. Es útil como documentación aunque tu objeto no tenga formulario de administración.

2.4.6 Primary_key

Si es True, este campo es la clave primaria del modelo.

Su no se especifica primary_key=True para ningún campo del modelo, Django agregará automáticamente este campo:

id = models.AutoField('ID', primary_key=True)

Por lo tanto, no necesitas establecer primary_key=True en ningún campo, salvo que quieras sobrescribir el comportamiento por omisión de la clave primaria.

primary_key=True implica blank=False, null=False, y unique=True. Solo se permite una clave primaria en un objeto.

2.4.7 Radio_admin

Por omisión, el administrador de Django usa una interfaz de cuadro de selección (<select>) para campos que son ForeignKey o tienen choices. Si radio_admin es True, Django utilizará una interfaz radio-button en su lugar.

No utilice esto para un campo que no sea ForeignKey o no tenga choices.

Unique.- Si es True, el valor para este campo debe ser único en la tabla.

Unique_for_date.- Asignar como valor el nombre de un DateField o DateTimeField para requerir que este campo sea único para el valor del campo tipo fecha, por ejemplo:

class Historia(models.Model):

fecha_publicacion = models.DateTimeField()

slug = models.SlugField(unique_for_date="fecha_publicacion")

En este código, Django no permitirá la creación de dos historias con el mismo slug publicados en la misma fecha. Esto difiere de usar la restricción unique_together en que solo toma en cuenta la fecha del campo pub_date; la hora no importa.

Unique_for_month.- Similar a unique_for_date, pero requiere que el campo sea único con respecto al mes del campo dado.

Unique_for_year.- Similar a unique_for_date y unique_for_month, pero para el año.

2.4.8 Verbose_name

Cada tipo de campo, excepto ForeignKey, ManyToManyField, y OneToOneField, toma un primer argumento posicional opcional un nombre descriptivo. Si el nombre descriptivo no está dado, Django lo creará automáticamente usando el nombre de atributo del campo, convirtiendo guiones bajos en espacios.

En este ejemplo, el nombre descriptivo es "Primer nombre de personas":

primer_nombre = models.CharField("Primer nombre de personas", maxlength=30)

En este ejemplo, el nombre descriptivo es "primer nombre":

primer_nombre = models.CharField(maxlength=30)

ForeignKey, ManyToManyField, y OneToOneField requieren que el primer argumento sea una clase del modelo, en este caso hay que usar verbose_name como argumento con nombre:

encuesta = models.ForeignKey(Encuesta, verbose_name="las encuestas relacionadas")

sitios = models.ManyToManyField(Sitio, verbose_name="lista de sitios")

lugares = models.OneToOneField(Lugar, verbose_name="related place")

La convención es no capitalizar la primera letra del verbose_name. Django convertirá a mayúscula automáticamente la primera letra cuando lo necesite.

2.5 Relaciones

Es claro que el poder de las bases de datos se basa en relacionar tablas entre sí. Django ofrece formas de definir los tres tipos de relaciones más comunes en las bases de datos: muchos-a-uno, muchos-a-muchos, y uno-a-uno.

2.5.1 Relaciones Muchos-a-Uno

Para definir una relación muchos-a-uno, usa ForeignKey. Se usa como cualquier otro tipo Field: incluyéndolo como un atributo de clase en tu modelo.

ForeignKey requiere un argumento posicional: la clase a la cual se relaciona el modelo.

Por ejemplo, si un modelo Vehículo tiene un Fabricante; es decir, un Fabricante fabrica varios vehículos pero cada vehículo tiene solo un fabricante usa la siguiente definición:

class Fabricante(models.Model):

...

class Vehículo(models.Model):

fabricante = models.ForeignKey(Fabricante)

...

Para crear una relación recursiva, un objeto que tiene una relación muchos-a-uno con él mismo para ello usa *models.ForeignKey('self'):*

class Trabajador(models.Model):

gerente = models.ForeignKey('self')

Si necesitas crear una relación con un modelo que aún no se ha definido, puedes usar el nombre del modelo en lugar del objeto modelo:

class Vehículo(models.Model):

fabricante = models.ForeignKey('Fabricante')

...

class Fabricante(models.Model):

...

Todas formas solo puedes usar cadenas para hacer referencia a modelos dentro del mismo archivo models.py; no puedes usar una cadena para hacer referencia a un modelo en una aplicación diferente, o hacer referencia a un modelo que ha sido importado de cualquier otro lado.

Django cuando se genera el modelo agrega "_id" al nombre de campo para crear su nombre de columna en la base de datos. En el ejemplo anterior, la tabla de la base de datos correspondiente al modelo Vehículo, tendrá una columna fabricante_id. (Puedes cambiar esto explícitamente especificando db_column; pero no es requerido que el nombre de un campo ForeignKey (fabricante en el ejemplo) sea el nombre del modelo en minúsculas. Por supuesto, puedes ponerle el nombre que quieras. Por ejemplo:

class Vehículo(models.Model):

compania_que_los_fabrica = models.ForeignKey(Fabricante)

 # ...

2.5.2 Relaciones Muchos-a-Muchos

Para definir una relación muchos-a-muchos, usa ManyToManyField. Al igual que ForeignKey, ManyToManyField requiere un argumento posicional: la clase a la cual se relaciona el modelo.

Por ejemplo, si una Pizza tiene varios ingredientes y variaciones de pizzas y cada Pizza tiene múltiples ingredientes debe representarse así:

class Ingredientes(models.Model):

...

class Pizza(models.Model):

ingredientes = models.ManyToManyField(Ingredientes)

...

Como sucede con ForeignKey, una relación de un objeto con sí mismo puede definirse usando el string 'self' en lugar del nombre del modelo, y puedes hacer referencia a modelos que todavía no se definieron usando un string que contenga el nombre del modelo. De todas formas solo se puede usar strings para hacer referencia a modelos dentro del mismo archivo models.py; no puedes usar una cadena para hacer referencia a un modelo en una aplicación diferente, o hacer referencia a un modelo que ha sido importado de cualquier otro lado.

Se sugiere, pero no es requerido, que el nombre de un campo ManyToManyField (varias, en el ejemplo) sea un término en plural que describa al conjunto de objetos relacionados con el modelo.

Django crea automáticamente una tabla join intermedia para representar la relación muchos-a-muchos.

No importa cuál de los modelos tiene el ManyToManyField, pero es necesario que esté en uno de los modelos no en los dos.

TABLA 6: Opciones de ManyToManyField

Argumento	Descripción
related_name	El nombre a utilizar para la relación desde el objeto relacionado hacia este objeto.
filter_interface	Usa una interfaz de "filtro" JavaScript agradable y discreta en lugar de la menos cómoda <select multiple> en el formulario administrativo de este objeto. El valor debe ser models.HORIZONTAL o models.VERTICAL (es decir, la interfaz debe aplicarse horizontal o verticalmente).
limit_choices_to	Limita el rango a escoger.
symmetrical	Solo utilizado en la definición de ManyToManyField sobre sí mismo. Considera el siguiente modelo: *class Persona(models.Model):* *friends = models.ManyToManyField("self")* Cuando Django procesa este modelo, identifica que tiene un ManyToManyField sobre sí mismo, y como resultado, no agrega un atributo persona_set a la clase Persona. En lugar de eso, se asumen que el ManyToManyField es simétrico, si yo soy tu amigo, entonces tú eres mi amigo. Si no deseas la simetría en las relaciones ManyToMany con self, establece symmetrical en False. Django agregará el descriptor para la relación inversa, permitiendo que las relaciones ManyToMany sean asimétricas.
db_table	El nombre de la tabla a crear para almacenar los datos de la relación muchos-a-muchos. Se asumirá un nombre por omisión basado en los nombres de las dos tablas a ser vinculadas.

Fuente: Autor

2.6 Opciones de los Metadatos del Modelo

Los metadatos específicos de un modelo viven en una class Meta definida en el cuerpo de tu clase modelo:

class Libro(models.Model):

titulo = models.CharField(maxlength=100)

class Meta:

 # model metadata options go here

Los metadatos del modelo son opciones de ordenamiento, nombres de tablas, etc.

2.6.1 db_table

El nombre de la tabla de la base de datos a usar para el modelo.

Django deriva automáticamente el nombre de la tabla de la base de datos a partir del nombre de la clase modelo y la aplicación que la contiene. Un nombre de tabla de base de datos de un modelo se construye uniendo la etiqueta de la aplicación del modelo y el nombre que usaste en la aplicación con el nombre de la clase modelo, con un guión bajo entre ellos.

Por ejemplo, si tienes una aplicación libros, un modelo definido como class Libro tendrá una tabla en la base de datos llamada libros.

Para sobrescribir el nombre de la tabla de la base de datos, use el parámetro db_table dentro de class Meta:

class Libro(models.Model):

nombre = models.CharField(max_length=30)

class Meta:

db_table = 'para_leer'

2.6.2 Get_latest_by

El nombre de un DateField o DateTimeField del modelo. Esto especifica el campo a utilizar por omisión en el método latest() del Manager del modelo.

class OrdenCliente(models.Model):

fecha_orden = models.DateTimeField()

...

class Meta:

get_latest_by = "fecha_orden"

2.6.3 Order_with_respect_to

Marca este objeto como "ordenable" con respecto al campo dado. Esto se utiliza casi siempre con objetos relacionados para permitir que puedan ser ordenados respecto a un objeto padre. Por ejemplo, si una pregunta tiene más de una respuesta, dada su relación y el orden de las respuestas importa:

class Respuesta(models.Model):

pregunta = models.ForeignKey(Pregunta)

 # ...

class Meta:

order_with_respect_to = 'pregunta'

2.6.4 Ordering

El ordenamiento por omisión del objeto, utilizado cuando se obtienen listas de objetos:

class Libro(models.Model):

titulo = models.CharField(maxlength=100)

class Meta:

ordering = ['titulo']

Esto es una tupla o lista de strings. Cada string es un nombre de campo con un prefijo opcional, que indica orden descendiente. Los campos sin un precedente se ordenarán en forma ascendente. Use el string "?" para ordenar al azar.

Por ejemplo, para ordenar por un campo título en orden ascendente (A-Z), usa esto:

ordering = ['titulo']

Para ordenar por título en orden descendente (Z-A), usa esto:

ordering = ['-titulo']

Para ordenar por título en orden descendente, y luego por autor en orden ascendente, usa esto:

ordering = ['-titulo', 'autor']

Observar que no importa cuántos campos haya en ordering, el sitio de administración usa sólo el primer campo.

2.6.5 Permissions

Permisos extra para almacenar en la tabla de permisos cuando se crea este objeto. Se crean automáticamente permisos para agregar, eliminar y cambiar para cada objeto que tenga establecida la opción admin. Este ejemplo especifica un permiso extra, puedo_pedir_pizzas:

class Trabajador(models.Model):

...

class Meta:

permissions = (("puedo_pedir_pizzas", "Puedo pedir pizzas"),)

2.6.6 Unique_together

Conjuntos de nombres de campo que tomados juntos deben ser únicos:

class Trabajador(models.Model):

departamento = models.ForeignKey(Departamento)

extension = models.CharField(maxlength=10)

...

class Meta:

unique_together = [("departamento", "extension")]

Esto es una lista de listas de campos que deben ser únicos cuando se consideran juntos. Es usado en la interfaz de administración de Django y se refuerza a nivel de base de datos (esto es, se incluyen las sentencias UNIQUE apropiadas en la sentencia CREATE TABLE).

2.6.7 Verbose_name

Un nombre legible por humanos para el objeto, en singular:

class OrdenCliente(models.Model):

fecha_orden = models.DateTimeField()

...

class Meta:

verbose_name = "orden"

Si no se define, Django utilizará una versión adaptada del nombre de la clase.

2.6.8 Verbose_name_plural

El nombre del objeto en plural:

class Casa(models.Model):

...

class Meta:

verbose_name_plural = "casas"

Si no se define, Django agregará una “s” al final del verbose_name.

2.7 Managers

Un Manager es la interfaz a través de la cual se proveen las operaciones de consulta de la base de datos a los modelos de Django. Existe al menos un Manager para cada modelo en una aplicación Django.

Esta sección trata específicamente las opciones del modelo que personaliza el comportamiento del Manager.

2.7.1 Nombres de Manager

Django agrega un Manager llamado objects a cada clase modelo de Django. De todas formas, si se desea usar objects como nombre de campo, o quieres usar un nombre distinto de objects para el Manager, puedes renombrarlo en cada uno de los modelos. Para renombrar el Manager para una clase dada, define un atributo de clase de tipo models.Manager() en ese modelo, por ejemplo:

from django.db import models

class Persona(models.Model):

...

gente = models.Manager()

Usando este modelo de ejemplo, Persona objects generará una excepción AttributeError ya que el objeto Persona no tiene un atributo objects sino gente, pero Persona.gente.all() devolverá una lista de todos los objetos Persona.

2.7.2 Managers Personalizados

Puedes utilizar un Manager personalizado en un modelo en particular extendiendo la clase base Manager e instanciando tu Manager personalizado en tu modelo.

Hay dos razones por las que puedes querer personalizar un Manager: para agregar métodos extra al Manager, y/o para modificar el QuerySet inicial que devuelve el Manager.

2.7.3 Agregando Métodos Extra al Manager

Agregar métodos extra al Manager es la forma preferida de agregar funcionalidad a nivel de tabla a tus modelos.

Un método Manager personalizado puede retornar cualquier cosa que necesites. No tiene que retornar un QuerySet.

Por ejemplo, este Manager personalizado ofrece un método with_counts(), que retorna una lista de todos los objetos OpinionPoll, cada uno con un atributo extra num_responses que es el resultado de una consulta agregada:

from django.db import connection

class AdministradorEncuestas(models.Manager):

def with_counts(self):

cursor = connection.cursor()

cursor.execute(""" SELECT p.id, p.pregunta, p.fecha_encuesta, COUNT(*)

FROM encuesta_opinionencuesta p, encuestas_respuesta r

WHERE p.id = r.poll_id

GROUP BY 1, 2, 3 ORDER BY 3 DESC""")

result_list = []

for row in cursor.fetchall():

p = self.model(id=row[0], question=row[1], poll_date=row[2])

p.num_responses = row[3]

result_list.append(p)

return result_list

class OpinionEncuesta(models.Model):

pregunta = models.CharField(maxlength=200)

fecha_encuesta = models.DateField()

objects = PollManager()

class Respuesta(models.Model):

encuesta = models.ForeignKey(Poll)

nombre_persona = models.CharField(maxlength=50)

respuesta = models.TextField()

En este ejemplo, puedes usar OpinionEncuesta.objects.with_counts() para retornar la lista de objetos OpinionEncuesta con el atributo num_responses.

Otra cosa a observar en este ejemplo es que los métodos de un Manager pueden acceder a self.model para obtener la clase modelo a la cual están anexados.

2.7.4 Modificando los QuerySets iniciales del Manager

Un QuerySet base de un Manager devuelve todos los objetos en el sistema. Por ejemplo, usando este modelo:

class Libro(models.Model):

titulo = models.CharField(maxlength=100)

autor = models.CharField(maxlength=50)

La sentencia Book.objects.all() retornará todos los libros de la base de datos.

Puedes sobrescribir el QuerySet base, sobre-escribiendo el método Manager.get_query_set(). get_query_set() debe retornar un QuerySet con las propiedades que se requiere.

Por ejemplo, el siguiente modelo tiene dos managers; uno que devuelve todos los objetos, y otro que retorna solo los libros de Roald Dahl:

 # Primero, define el Manager() de subclases.

class LibroBenedettiManager(models.Manager):

def get_query_set(self):

return super(LibroBenedettiManager, self).get_query_set().filter(autor='Mario Benedetti')

class Libro(models.Model):

titulo = models.CharField(maxlength=100)

autor = models.CharField(maxlength=50)

objects = models.Manager() # manager.

benedetti_objects = LibroBenedettiManager() # Los libros de Bendetti manager.

Con este modelo de ejemplo, Libro.objects.all() retornará todos los libros de la base de datos, pero Libro.benedetti_objects.all() solo retornará aquellos escritos por Mario Benedetti.

Como get_query_set() devuelve un objeto QuerySet, se puede usar filter(), exclude(), y todos los otros métodos de QuerySet sobre él. Por lo tanto, se pueden usar estas sentencias

Libro.benedetti_objects.all()

Libro.benedetti_objects.filter(titulo='Despistes y Franquezas')

Libro.benedetti_objects.count()

2.8 Métodos para Modelos especiales

2.8.1 __str__

Método de Python que define lo que debe ser devuelto si llamas a str() sobre el objeto. Django usa str(obj), en varios lugares, particularmente como el valor mostrado para hacer el render de un objeto en el sitio de administración de Django y como el valor insertado en una plantilla cuando muestra un objeto. Por eso, siempre debes retornar un string agradable y legible por humanos en el

__str__ de un objeto. A pesar de que esto no es requerido, es altamente recomendado.

class Persona(models.Model):

primer_nombre = models.CharField(maxlength=50)

apellido = models.CharField(maxlength=50)

def __str__(self):

return '%s %s' % (self.primer_nombre, self.apellido)

2.8.2 Get_absolute_url

Define un método get_absolute_url() para decirle a Django cómo calcular la URL de un objeto, por ejemplo:

def get_absolute_url(self):

return "/gente/%i/" % self.id

Django usa esto en la interfaz de administración. Si un objeto define get_absolute_url(), la página de edición del objeto tendrá un enlace "View on site", que te llevará directamente a la vista pública del objeto, según get_absolute_url().

También un par de otras partes de Django, como el framework de sindicación de feeds, usan get_absolute_url() como facilidad para recompensar a las personas que han definido el método.

Es una buena práctica usar get_absolute_url() en plantillas, en lugar de codificar en duro las URL de tus objetos. Por ejemplo, este código de plantilla es malo:

{{ object.name }}

Mejor por:

{{ object.name }}

El problema con la forma en que simplemente escribimos get_absolute_url() es que viola levemente el principio DRY: la URL de este objeto de define dos veces, en el archivo URLconf y en el modelo.

Además puedes desacoplar tus modelos del URLconf usando el decorador permalink. A este decorador se le pasa función de view, una lista de parámetros posicionales, y un diccionario de parámetros por nombre. Django calcula la URL completa correspondiente usando el URLconf, sustituyendo los parámetros que le has pasado en la URL. Por ejemplo, si el URLconf contiene una línea como ésta:

url(r'^people/(\d+)/$', 'people.views.details'),

tu modelo puede tener un método get_absolute_url como éste:

@models.permalink

def get_absolute_url(self):

return ('people.views.details', [str(self.id)])

En forma similar, si tienes una entrada en URLconf que se ve como esta:

url(r'/archive/(?P<year>\d{4})/(?P<month>\d{1,2})/(?P<day>\d{1,2})/$', archive_view)

puedes hacer referencia a la misma usando permalink() como sigue:

@models.permalink

def get_absolute_url(self):

return ('archive_view', (), {

'year': self.created.year,

'month': self.created.month,

'day': self.created.day})

Observar que especificamos una secuencia vacía para el segundo argumento en este caso, porque sólo queremos pasar argumentos por clave, no argumentos por nombre.

De esta forma, estás ligando la URL absoluta del modelo a la vista que se utiliza para mostrarla, sin repetir la información de la URL en ningún lado. Aún puedes usar el método get_absolute_url en plantillas.

2.9 Middleware

2.9.1 ¿Por qué los middlewares?

Cuando se necesite ejecutar una porción de código en todas las peticiones del framework éste código puede necesitar modificar la petición antes de que la vista se encargue de ella, puede necesitar registrar información sobre la petición para propósitos de ejecución por ejemplo.

Lo que se recomienda es hacer un middleware para eso Django, que es un conjunto de partes dentro del procesamiento de post/request de Django. Que es capaz de alterar de forma global tanto la entrada como la salida de Django.

Los middlewares estas más cerca del desarrollo que casi pasan desapercibidos por ejemplo los utilizamos en el inicio de sesión para ellos se utiliza porciones pequeñas de código(middleware) haciendo que request.session y request.user estén disponibles en la capa vista.

La memoria cache global del aplicación web es solo una pieza de middleware que desvía la llamada a tu función de vista si la respuesta para esa vista ya fue almacenada en la cache.

Así como la paginación, redirecciones y el cifrado de csrf llaman a un middleware.

2.9.2 ¿Qué es un middleware?

Un componente middleware es simplemente una clase Python que se ajusta a una cierta API para obtener un cierto resultado como saber la IP de las páginas que visitan tu página o los usuarios en red.

2.9.3 Instalación de Middleware

Para activar un componente middleware, agregarlo a la tupla MIDDLEWARE_CLASSES en tu archivo de configuración (settings.py).

En MIDDLEWARE_CLASSES, cada componente middleware se representa con un string: la ruta Python completa al nombre de la clase middleware; que por defecto se crea cuando creamos un proyecto nuevo.

MIDDLEWARE_CLASSES = (

'django.middleware.common.CommonMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.middleware.doc.XViewMiddleware'

)

El componente middleware que es importante es CommonMiddleware; ya que es importante para controlar el arranque del servidor.

2.9.4 Middleware de soporte para autenticación

Clase middleware: django.contrib.auth.middleware.AuthenticationMiddleware.

Este middleware permite el soporte para autenticación. Agrega el atributo request.user, que representa el usuario actual registrado, a todo objeto HttpRequest que se recibe.

2.9.5 Middleware “Common”

Clase middleware: django.middleware.common.CommonMiddleware.

Este middleware agrega algunas conveniencias para los perfeccionistas:

Prohíbe el acceso a los agentes de usuario especificados en la configuración "DISALLOWED_USER_AGENTS": Si se especifica, esta configuración debería

ser una lista de objetos de expresiones regulares compiladas que se comparan con el encabezado user-agent de cada petición que se recibe. Aquí está un pequeño ejemplo de un archivo de configuración:

import re

DISALLOWED_USER_AGENTS = (

re.compile(r'^OmniExplorer_Bot'),

re.compile(r'^Googlebot')

)

El import re, ya que DISALLOWED_USER_AGENTS requiere que sus valores sean expresiones regulares compiladas. El archivo de configuración es un archivo común de Python, por lo tanto es perfectamente adecuado incluir sentencias import en él.

Realiza re-escritura de URL basado en las configuraciones APPEND_SLASH y PREPEND_WWW: Si APPEND_SLASH es igual a True, las URLs que no poseen una barra al final serán redirigidas a la misma URL con una barra al final, a menos que el último componente en el path contenga un punto. De esta manera foo.com/bar es redirigido a foo.com/bar/, pero foo.com/bar/file.txt es pasado a través sin cambios.

Si PREPEND_WWW es igual a True, las URLs que no poseen el prefijo “www.” serán redirigidas a la misma URL con el prefijo “www.”.

Ambas opciones tienen por objeto normalizar URLs. La filosofía es que cada URL debería existir en un y sólo un lugar. Técnicamente la URL example.com/bar es distinta de example.com/bar/, la cual a su vez es distinta de www.example.com/bar/. Un motor de búsqueda indexador trataría de forma separada estas URLs, lo cual es perjudicial para la valoración de tu sitio en el motor de búsqueda, por lo tanto es una buena práctica normalizar las URLs.

Maneja ETags basado en la configuración USE_ETAGS: ETags son una optimización a nivel HTTP para almacenar condicionalmente las páginas en la caché. Si USE_ETAGS es igual a True, Django calculará una ETag para cada petición mediante la generación de un hash MD5 del contenido de la página, y se hará cargo de enviar respuestas Not Modified, si es apropiado.

Nota también que existe un middleware de GET condicional, que veremos en breve, el cual maneja ETags y hace algo más.

2.9.9 Middleware de soporte para sesiones

Clase middleware: django.contrib.sessions.middleware.SessionMiddleware.

Este middleware habilita el soporte para sesión.

2.9.11 Middleware de transacción

Clase middleware: django.middleware.transaction.TransactionMiddleware.

Este middleware asocia un COMMIT o ROLLBACK de la base de datos con una fase de petición/respuesta. Si una vista de función se ejecuta con éxito, se emite un COMMIT. Si la vista provoca una excepción, se emite un ROLLBACK.

El orden de este middleware en la pila es importante. Los módulos middleware que se ejecutan fuera de este, se ejecutan con commit-on-save el comportamiento por omisión de Django. Los módulos middleware que se ejecutan dentro de este estarán bajo el mismo control de transacción que las vistas de función.

2.9.12 Middleware “X-View”

Clase middleware: django.middleware.doc.XViewMiddleware.

Este middleware envía cabeceras HTTP X-View personalizadas a peticiones HEAD que provienen de direcciones IP definidas en la configuración INTERNAL_IPS. Esto es usado por el sistema automático de documentación de Django.

2.10 Base de Datos relacional MariaDB

TABLA 7: Logo MariaDB

Fuente: mariadb.org

2.10.1 Introducción

MariaDB la salida técnica del activismo open source; es un fork de MySQL que hace poco era parte de Sun Microsystems empresa que fue adquirida por Oracle. En 2010. En 2008 Sun Microsystems adquiere MySQL y se hizo de una gran comunidad de desarrolladores, luego que fue adquirida por Oracle la situación cambió y actualmente hay plugins de MySQL de propiedad exclusiva de Oracle por ejemplo el plugin de NoSQL.

MariaDB al ser una bifurcación (fork) de MySQL que es la base de datos más popular para administración de base de datos; al heredar y mejorar la solidez, seguridad y rapidez bajo el soporte del fundador de MySQLAB Michael “Monty” Widenius; ha sido capaz de generar otro producto con mejores características de su padre.

Monty (Machael Widenius) decidió crear esta variante porque estaba convencido de que el único interés de Oracle en MySQL era reducir la competencia que MySQL daba al mayor vendedor de bases de datos relacionales del mundo que es Oracle. (Follet, 2014)

2.10.2 ¿Por qué un fork?

En los proyectos de código abierto o de software libre sacar una rama de un software en principio es técnico, es tratar de evitar en lo más posible que un programa se expanda con características que no son funcionales para los usuarios sea cual sea su interés; con versiones de MySQL 4 no son las mismas que las tenemos en MySQL 5; ya que la primera era más sencilla y rápida y no tenía código oculto para poder ser incorporadas a los repositorios oficiales de las distribuciones GNU/Linux, principalmente.

Los forks también que no ofrecen características nuevas o mejoras de las ya existentes uno de ellos son los motores de almacenamiento MyISAM e InnoDB; para otras ramas está desarrollando un nuevo motor de almacenamiento ideal para su propósito. Lo que estaba por hacer era reemplazar las MySQL con cualquier fork que venga de este lo que se denomina “drop in” que en un principio utilizan el mismo código, interfaces y middlewares.

La compatibilidad de los forks depende del equipo o grupo de personas que darán soporte y que pueden alterar drásticamente su código haciendo de esta versión incompatible con su predecesor. (Widenius, 2013)

Oracle al comprar Sun que compró MySQLAB; controla el desarrollo que a ciencia cierta no se conoce cuál es el fin pero Monty ya lo predijo de cierta manera ya que Oracle es un DBMS comercial; y la única teoría que se maneja es quitarle la popularidad que actualmente tiene MySQL, por eso que decidió junto con su equipo sacar una rama.

2.10.3 Motor de almacenamiento XtraDB

Percona es la encargada de lanzar el modificado motor que no es más que un InnoDB mejorado para soportar sitios “modernos” que tiene alta disponibilidad y flexibilidad ya que soportaría servidores de alta gama sacándole el máximo provecho a dicho motor, mejorando sustancialmente su funcionalidad.

La rama XtraDB tenía otro objetivo ser una simple gota en el reemplazo para el motor de almacenamiento InnoDB, por lo que los usuarios podrían simplemente cambiar su motor de almacenamiento sin tener que cambiar nada de su código de la aplicación subyacente. XtraDB tenía que ser compatible hacia atrás con InnoDB, además de proporcionar todas las nuevas características y mejoras que se querían añadir.

Continuación se presentara algunas ramas que tienen como origen MySQL.

Percona

Es un producto de base de datos independiente que ofrece a los usuarios la posibilidad de intercambiar su instalación de MySQL y de intercambio en el producto servidor Percona, y de este modo, aprovechar el motor de almacenamiento XtraDB. Esto hace que la afirmación de que es totalmente compatible con MySQL, así que no hay que cambiar ningún código en el software. Eso es definitivamente una gran ventaja, y es genial para el control de calidad cuando usted está buscando para una rápida mejora en el rendimiento. Por lo tanto, una buena razón para mirar servidor Percona es tomar ventaja del motor XtraDB como con algunos cambios en su código principal como sea posible.

Drizzle

El fork diferente que añade unas características importantes y mejora los motores de almacenamiento haciéndola no compatible con MySQL; pese a que es un fork; principalmente es una reescritura importante de los motores MySQL, con las características calificadas de no-óptimo e innecesario, y con gran parte del código reescrito para ser optimizado, incluso yendo tan lejos sería comparable tanto como decir de pasar de C a C++ para el código.

MariaDB

Otro de los productos que ofrece el mercado con el motor de almacenamiento XtraDB que es muy compatible con MySQL por lo que ofrece motores de almacenamiento estándar y de por lo tanto altamente compatible con los motores de almacenamiento de MySQL es decir MyISAM y InnoDB los hace poco motores por defecto de MySQL.

2.10.4 ¿Qué es SQL?

SQL (S-Q-L) Es una abreviatura para Structured Query Language. SQL es un lenguaje diseñado específicamente para la comunicación con las bases de datos.

A diferencia de otras lenguas (como el inglés o de programación como Python o Perl), SQL se compone de muy pocas palabras, también se podría decir que está diseñado para hacer una cosa y hacerla bien, proporcionando una sencilla y eficiente manera de leer y escribir datos desde una base de datos.

2.10.5 ¿Cuáles son las ventajas de SQL?

SQL no es un lenguaje propio de las empresas que proveen base de datos.

Casi todos los principales DBMS soportan SQL, por lo que el aprendizaje de éste lenguaje le permite interactuar con casi toda base de datos.

SQL es fácil de aprender. Los comandos se describen en inglés y son muy.

A pesar de su aparente simplicidad, SQL es en realidad un lenguaje poderoso, y hábilmente utilizando sus elementos de lenguaje puede realizar operaciones simples, complejas o sofisticadas.

2.10.6 Concepto ACID

El las base de datos relacionales es casi imprescindibles hablar sobre el concepto ACID[3]., dicho acrónimo que acoge cuatro propiedades que toda base de datos transaccional debe cumplir sin dilaciones, estas son atomicidad, consistencia, aislamiento y durabilidad.

Atomicidad

Establece que, las modificaciones en la bases de datos debe ejecutarse sin fallas si es así la transacción se dice que es atómica. Si una parte de la transacción falla, toda la transacción falla. Es fundamental que el sistema de gestión de base de datos mantenga la naturaleza atómica de transacciones, a pesar de cualquier DBMS, sistema operativo o de hardware. (Janssen)

[3] En 1983, Andreas Reuter y Theo Harder acuñaron el acrónimo ACID para describirlos. Haerder, T.; Reuter, A. (1983). "Principles of transaction-oriented database recovery". ACM Computing Surveys 15 (4): 287.

Usualmente, los sistemas implementan la atomicidad mediante algún mecanismo que indica qué transacción comenzó y cuál finalizó; o manteniendo una copia de los datos antes de que ocurran los cambios. Las bases de datos en general implementan la atomicidad usando algún sistema de logging para seguir los cambios. El sistema sincroniza los logs a medida que resulta necesario una vez que los cambios ocurren con éxito. Luego, el sistema de recuperación de caídas simplemente ignora las entradas incompletas.

En los sistemas de almacenamiento NoSQL con consistencia eventual, la atomicidad se especifica de forma más débil que en los sistemas relacionales, y existe sólo para las filas.

Consistencia

La consistencia asegura que los cambios a los valores de una instancia son consistentes con cambios a otros valores de la misma instancia. Una restricción de consistencia es un predicado sobre los datos que funcionan como precondición, pos condición, y condición de transformación en cualquier transacción. El sistema de la base de datos asume que la consistencia se mantiene para cada transacción en las instancias. Por otro lado, asegurar la propiedad de consistencia de la transacción es responsabilidad del usuario.

Asegura que sólo los datos válidos siguientes todas las reglas y restricciones se escriben en la base de datos. Cuando una transacción se traduce en datos no válidos, la base de datos vuelve a su estado anterior, que se rige por las reglas y limitaciones habituales. (Janssen)

Aislamiento

Cuando se intenta mantener el más alto nivel de aislamiento, las bases de datos adquieren un bloqueo o implementan un control concurrente multiversión, que puede resultar en pérdida de concurrencia. Esto genera que la aplicación agregue lógica para funcionar correctamente.

Se asegura de que las transacciones se procesan de forma independiente y al mismo tiempo sin interferencias, pero no garantiza el orden de transacciones. Por ejemplo, el usuario A se retira $ 100 y el usuario B se retira 250 dólares de la cuenta de usuario de Z, que tiene un saldo de $ 1000. Dado que tanto A y B sorteo de la cuenta del Z, uno de los usuarios es necesario esperar hasta que se complete la otra transacción de usuario, evitando datos inconsistentes. Si se requiere B que esperar, entonces B debe esperar hasta que se complete la transacción de A, y el balance de la cuenta del Z cambia a $ 900. Ahora, B puede retirar $ 250 desde $ 900 este equilibrio. (Janssen)

La mayoría de las bases de datos ofrecen una cantidad de niveles de aislamiento de transacciones, que controlan el grado de bloqueo que ocurre cuando se seleccionan datos.

Para muchas aplicaciones, se pueden construir la mayoría de las transacciones evitando los niveles más altos de aislamiento (por ejemplo, el nivel SERIALIZABLE), y por lo tanto reduciendo la carga en el sistema por bloqueos. El programador tiene que analizar el código de acceso para asegurar que se pueda relajar el nivel de aislamiento sin generar bugs que luego serían difíciles de encontrar. Por otro lado, si se usan niveles más altos de aislamiento, es más probable la aparición de dealocks, lo cual requieren de análisis y la implementación de técnicas de programación para evitarlos.

Durabilidad

La durabilidad es la propiedad que garantiza que las transacciones que tuvieron un commit sobrevivan de forma permanente.

La durabilidad puede lograrse almacenando los registros de log de la transacción en un medio de almacenamiento no-volatil antes de aceptar los commit.

En las transacciones distribuidas, todos los participantes deben coordinarse antes de aceptar un commit. Esto se suele realizar mediante el protocolo de "commit en dos fases".

Muchas bases de datos implementan la durabilidad escribiendo las transacciones en un log de transacciones que puede ser reprocesado para recrear el estado del sistema antes de una falla.

En el ejemplo anterior, el usuario B podrá retirar $ 100 sólo después de la transacción del usuario A se completa y se actualiza en la base de datos. Si el sistema falla antes de la transacción de A se registra en la base de datos, A no puede retirar dinero, y cuenta vuelva de Z a su estado anterior consistente. (Janssen)

2.10.7 Características

MariaDB posee varias características que la hacen mejor que su predecesor como se los describe a continuación.

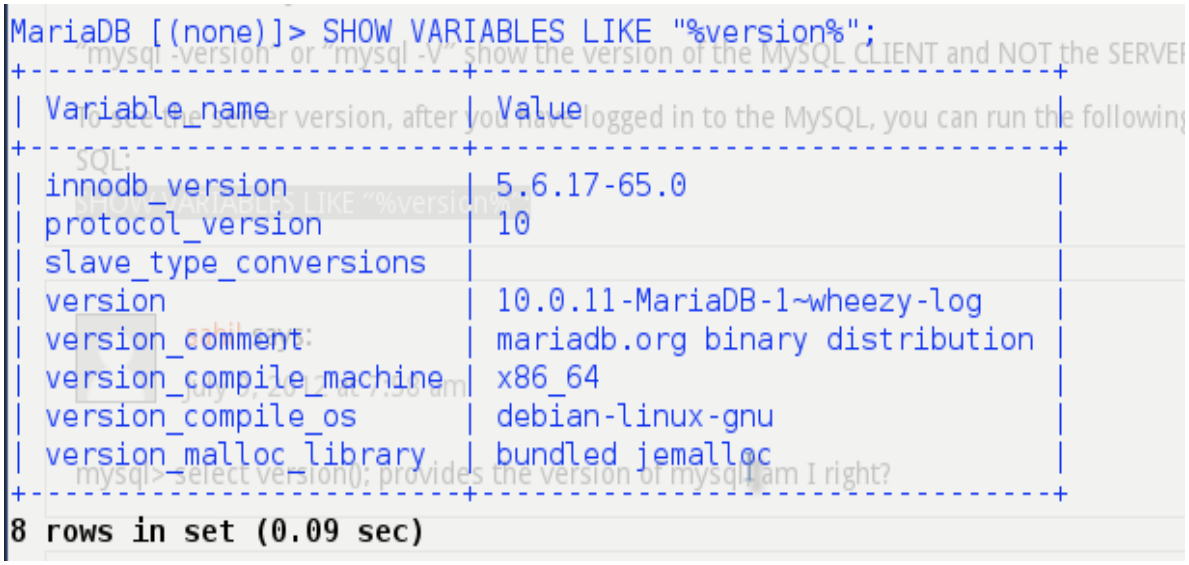


ILUSTRACIÓN 6: MariaDB 10.0 versión en debian

Fuente: Autor

```
ysql> show engines;
+------------+---------+------------------------------------------------------------+
| Engine     | Support | Comment                                                    |
+------------+---------+------------------------------------------------------------+
| MyISAM     | DEFAULT | Default engine as of MySQL 3.23 with great performance     |
| MEMORY     | YES     | Hash based, stored in memory, useful for temporary tables  |
| InnoDB     | YES     | Supports transactions, row-level locking, and foreign keys |
| BerkeleyDB | NO      | Supports transactions and page-level locking               |
| BLACKHOLE  | YES     | /dev/null storage engine (anything you write to it disappears) |
| EXAMPLE    | NO      | Example storage engine                                     |
| ARCHIVE    | YES     | Archive storage engine                                     |
| CSV        | NO      | CSV storage engine                                         |
| ndbcluster | NO      | Clustered, fault-tolerant, memory-based tables             |
| FEDERATED  | YES     | Federated MySQL storage engine                             |
| MRG_MYISAM | YES     | Collection of identical MyISAM tables                      |
| ISAM       | NO      | Obsolete storage engine                                    |
+------------+---------+------------------------------------------------------------+
2 rows in set (0.00 sec)
```

ILUSTRACIÓN 7: Motores de almacenamiento en MySQL 5.5 Windows Xp

Fuente: Autor

```
MariaDB [(none)]> show engines
    -> ;
+--------------------+---------+--------------------------------------------------------------+--------------+------+-------------+
| Engine             | Support | Comment                                                      | Transactions | XA   | Savepoints  |
+--------------------+---------+--------------------------------------------------------------+--------------+------+-------------+
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables    | NO           | NO   | NO          |
| MRG_MyISAM         | YES     | Collection of identical MyISAM tables                        | NO           | NO   | NO          |
| MyISAM             | YES     | MyISAM storage engine                                        | NO           | NO   | NO          |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO         | NO   | NO          |
| CSV                | YES     | CSV storage engine                                           | NO           | NO   | NO          |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                           | NO           | NO   | NO          |
| ARCHIVE            | YES     | Archive storage engine                                       | NO           | NO   | NO          |
| InnoDB             | DEFAULT | Percona-XtraDB, Supports transactions, row-level locking, and foreign keys | YES | YES | YES |
| FEDERATED          | YES     | FederatedX pluggable storage engine                          | YES          | NO   | YES         |
| Aria               | YES     | Crash-safe tables with MyISAM heritage                       | NO           | NO   | NO          |
+--------------------+---------+--------------------------------------------------------------+--------------+------+-------------+
10 rows in set (0.00 sec)
```

ILUSTRACIÓN 8: Motores de almacenamiento en MariaDB 5.6 mejorados

Fuente: Autor

2.11 Motores De Almacenamiento

2.11.1 Aria

El motor de almacenamiento Aria está compilado por defecto en MariaDB 5.1 y es necesario que esté activado cuando se inicia el proceso mysqld.

Además, las tablas internas en disco están en el formato de tabla Aria en lugar del formato de tabla MyISAM; logrando acelerar las funciones de agregación como GROUP BY y DISTINCT consultas porque Aria tiene mejor almacenamiento en caché de MyISAM. La inclusión de ARIA es una de las diferencias entre MariaDB 5.1 y MySQL 5.1.

El motor de almacenamiento Aria era conocida anteriormente como María (ver el nombre Aria para obtener detalles sobre el cambio de nombre) y en versiones anteriores de MariaDB el motor todavía se llamaba María. (Widenius, 2013) Que fue votado en y la comunidad escogió un nombre para Maria y había votado mayoritariamente por Aria. (Mneptok, 2010)

2.11.2 XtraDB (reemplazo de InnoDB)

MariaDB por defecto utiliza el motor de almacenamiento XtraDB, un rendimiento mejorado del motor de almacenamiento InnoDB. Por razones de compatibilidad, las variables del sistema aún conservan sus prefijos innodb originales, y en el que el texto en esta base de conocimiento se refiere a InnoDB, por lo general se puede entender como una referencia al tenedor XtraDB.

Tenga en cuenta que es posible utilizar InnoDB como un plugin en MariaDB lugar de XtraDB.

Tenga en cuenta que antes de MariaDB dev. 10.0.9, MariaDB incluido InnoDB de MySQL como el predeterminado, con XtraDB de Percona disponible como un plug-in dinámico. Para utilizar XtraDB en 10.0.8 y posteriores, la variable ignore_builtin_innodb debe establecerse, y el plugin-load = opción ha_xtradb proporcionado.

Por otro lado Percona XtraDB incluye todo el diseño de InnoDB robusto, fiable; cumple las reglas ACID y arquitectura MVCC avanzada, y se basa en que la base sólida con más funciones, más capacidad de ajuste, más métricas, y más escalabilidad. En particular, está diseñado para escalar mejor en muchos núcleos, para usar la memoria de manera más eficiente, y para ser más conveniente y útil. Las nuevas características son especialmente diseñadas para aliviar algunas de las limitaciones de InnoDB. Se eligieron características y correcciones en base a las peticiones de los clientes y en nuestro mejor juicio de necesidades del mundo real como usuario. (Widenius M. "., 2010)

2.11.3 PBXT (MariaDB 5.1, 5.2 y 5.3. Desactivado en MariaDB 5.5.)

En la versión 5.5, el motor de almacenamiento PBXT está desactivada por defecto y debe ser construido expresamente para utilizarlo. La razón es que PBXT no se mantiene de forma activa, tiene algunos errores que no son fijos y no se utiliza activamente. Las versiones de PBXT en MariaDB son:

- versión 1.0.08d en MariaDB 5.1.44b
- versión 1.0.11 en MariaDB 5.1.47

PBXT es un motor de almacenamiento transaccional de propósito general. PBXT es totalmente "Acid" compatible, lo que significa que se puede utilizar como una alternativa a otros motores transaccionales MariaDB (tales como XtraDB o InnoDB).

2.11.4 FederatedX

Cisco necesitaba un motor de almacenamiento de MySQL, que les permita consolidar tablas remotas en algún tipo de dispositivo de enrutamiento, al ser capaz de interactuar con estas tablas remotos como si fueran locales al dispositivo, pero no realmente en el dispositivo, ya que el dispositivo de enrutamiento tenido sólo tanto espacio de almacenamiento, fue lanzado al público en un comunicado de MySQL 5.0.

Cuando MySQL 5.1 se convirtió en la versión de producción de MySQL, Federated tenía más características y mejoras:

Nuevo servidor federado añade al analizador. Esto era algo que Cisco necesitaba que lo hiciera posible para cambiar los parámetros de conexión para numerosas tablas federadas a la vez sin tener que alterar o volver a crear las tablas federadas.

2.11.5 Soporte De Plug-ins.

Cada motor de almacenamiento tiene que implementar métodos estándar en la clase de controlador para que trabaje un motor de almacenamiento. FederatedX no es diferente en ese sentido. La gran diferencia es que FederatedX necesita implementar estos métodos en una clase controlador sería como construir sentencias SQL para ejecutar en un host remoto y si hay un conjunto de resultados, procesar este conjunto de datos en el formato de controlador interno para que el resultado se devuelve al de usuario.

2.11.6 OQGRAPH (MariaDB 5.1, 5.2 y 5.3. desactivado 5.5.)

El motor de cálculo Open Query GRAPH o OQGRAPH como se lo llama, le permite manejar jerarquías (estructuras de árbol) y gráficos complejos (nodos que tienen muchas conexiones en varias direcciones).

OQGRAPH es a diferencia de otros motores de almacenamiento, que consta de una arquitectura completamente diferente a un motor de almacenamiento regular, tales como Aria, MyISAM o InnoDB.

Se destinado a ser utilizado para recuperar información jerárquica, tales como los utilizados para gráficos, rutas o relaciones sociales, en planos SQL.

2.11.7 SphinxSE

SphinxSE no almacena datos. SphinxSE está incorporado en el cliente que permite MariaDB comunicarse con el demonio de búsqueda de Sphinx (ejecutar consultas, obtener resultados). La indexación y la búsqueda se llevan a cabo por Sphinx.

Dos usos potenciales se presentan:

En una aplicación existente que hace uso de la búsqueda de texto completo en MyISAM, llevarla a cabo de forma fácil.

Si usted está usando un lenguaje de programación que Sphinx no tiene API nativa, la integración será fácil usando MariaDB/SphinxSE.

Sphinx realiza la ordenación, el filtrado y el corte de la hoja de resultados porque optimiza y se comporta mejor en estas tareas.

2.11.8 TokuDB

Presente en las versiones de MariaDB 5.5 y 10.0; el motor de almacenamiento TokuDB es para uso en alto rendimiento y para escritura en entornos de manera intensiva, ofreciendo una mayor compresión y mejor rendimiento.

Está disponible ya que es de código abierto, incluido en MariaDB de 64 bits (no está activo por defecto).

Las especificaciones oficiales de TokuDB así como manuales de productos están disponibles en el sitio web Tokutek.

2.11.9 Cassandra

El motor de almacenamiento que no está instalado; el motor de almacenamiento Cassandra permite el acceso a datos en un clúster de Cassandra en MariaDB, y es similar a la del motor de almacenamiento NDB.[4]

Puede acceder a la misma agrupación Cassandra desde varias instancias de MariaDB, siempre y cada uno de ellos corre el motor de almacenamiento Cassandra:

El objetivo principal de CassandraSE (motor de almacenamiento) es la integración de datos entre el SQL y NoSQL. Alguna vez se necesita para:

Tomar un poco de los datos de Cassandra desde su interfaz web, o una consulta SQL

Insertar algunos registros en Cassandra de alguna parte de su aplicación.

Ahora, esto es muy posible. CassandraSE hace que la familia la columna de Cassandra aparece como una tabla en MariaDB que puede insertar, actualizar, y elegir. Puedes escribir se une en contra de esta tabla, es posible unirse a los datos que se almacenan en MariaDB con los datos que se almacenan en Cassandra.

[4] NDB es un cluster de MySQL y sirve para acceder a los datos de forma independiente con la API respectiva.

2.11.10 Mejoras de velocidad

- Una gran cantidad de mejoras del optimizador en MariaDB 5.3. Las subconsultas son ahora utilizables.

- Más rápido y más seguro la replicación: Esto hace que muchas configuraciones que utilizan la replicación y la gran cantidad de cambios de más de 2x veces más rápido. (Forta, 2012)

- CHECKSUM[5] TABLE es más rápida.

- Mejoramiento del rendimiento de la conversión de juegos de caracteres (y conversiones removidas cuando no eran necesarias). Mejora general de velocidad es de 1-5 % (según sql-bench) pero puede ser mayor para los grandes conjuntos de resultados con todos los caracteres entre 0x00-0x7f. (Stroganov, 2014)

- Maneja 200.000 conexiones+ y con una mejora de la velocidad notable cuando se utiliza muchas conexiones gracias a un pool. (Soto, 2013)

- Hay algunas mejoras en el código DEBUG para hacer su ejecución más rápida cuando la depuración está compilando pero no se utiliza. (MariaDB Org., 2015)

- El motor de almacenamiento Aria permite consultas complejas más rápidas (consultas que normalmente utilizan tablas temporales en disco). El motor de almacenamiento Aria se utiliza para las tablas temporales internas, que debe dar un aumento de velocidad al hacer selecciona complejos. (MariaDB Org., 2015)

- El conjunto de pruebas se ha ampliado y ahora funciona mucho más rápido que antes, a pesar de que las pruebas de más cosas. (MariaDB Org., 2015)

[5] Es usado para verificar que un archivo compartido a un usuario es idéntico bit a bit al publicado en su fuente original

2.11.11 Extensiones

Microsegundos en MariaDB

(MariaDB Org., 2015)

En MariaDB, los tipos de datos TIME, DATETIME y TIMESTAMP, junto con las funciones temporales, CAST y columnas dinámicas, son compatibles con microsegundos. La precisión de fecha y hora de una columna puede especificarse al crear la tabla con CREATE TABLE, por ejemplo:

CREATE TABLE alumno(

col_microseg DATETIME(6),

col_miliseg TIME(3)

);

Tabla Eliminación

Es la propiedad de hacer consultas con modelados altamente normalizados lo que se conoce como "anchor modeling", es decir un atributo una tabla.

Columnas Virtuales

Hay dos tipos de columnas virtuales: PERSISTENTES, que se almacenan en la tabla, y virtual, que se generan cuando se consulta la tabla ya el valor predeterminado es virtual.

CREATE TABLE operaciones (
num10 INT NOT NULL,
num11 INT NOT NULL,
cade12 VARCHAR(32),
num14 INT AS (a mod b) VIRTUAL, #calcula el residuo de la división
cade15 VARCHAR(5) AS (left(c,5)) PERSISTENT);#toma los 5 primeros caracteres de la cadena.);
para insertar:
INSERT INTO operaciones VALUES (1,3,'some text',default,default);

2.12 Estadísticas de usuarios

La implementación MariaDB proporciona la misma funcionalidad que el parche serstatv2 pero una gran cantidad de cambios se han hecho para que sea más rápido y para ajustarse mejor a la infraestructura MariaDB.

Como las estadísticas de acceso, conexión, de usuario, de tablas.

- Eliminar todas las consultas para un usuario

KILL QUERY ID: termina la consulta por query_id, dejando la conexión intacta

2.12.1 Autenticación conectable

Manejo de autenticación mediante sesión del sistema operativo. Ya no necesita poner contraseñas aunque esto no quiera decir que es para cualquier usuario

Motor de Almacenamiento específico CREATE TABLE

Es un API que permite a MariaDB 5.2, especificar atributos adicionales del motor de un motor de almacenamiento por índice, campo o tabla. El motor tiene que declarar que atributos se introduce.

2.12.2 Las mejoras en la tabla INFORMACIÓN SCHEMA.PLUGINS

Con el comando show plugins ya podemos ver los diferentes componentes que trabajan con nuestro entorno

MariaDB [(none)]> show plugins

-> ;

+ -+ - - - - - - + - - - - - - - - - - - - - - - - + - - - - - - + - - - - -
+

| *Name* | *Status* | *Type* | *Library* |
License |

+ -+ - - - - - - + - - - - - - - - - - - - - - - - + - - - - - - + - - - - -
+

| *binlog* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *mysql_native_password* | *ACTIVE* | *AUTHENTICATION* | *NULL* | *GPL* |

| *mysql_old_password* | *ACTIVE* | *AUTHENTICATION* | *NULL* | *GPL* |

| *MEMORY* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *MyISAM* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *CSV* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *MRG_MyISAM* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *PERFORMANCE_SCHEMA* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *InnoDB* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *XTRADB_READ_VIEW* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *XTRADB_RSEG* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_TRX* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_LOCKS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_LOCK_WAITS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CMP* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CMP_RESET* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CMPMEM* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CMPMEM_RESET* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CMP_PER_INDEX* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_BUFFER_PAGE* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_METRICS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_FT_DELETED* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_FT_BEING_DELETED* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_FT_CONFIG* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_FT_INDEX_CACHE* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_FT_INDEX_TABLE* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_TABLES* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_TABLESTATS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_INDEXES* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_COLUMNS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_FIELDS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_FOREIGN* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_FOREIGN_COLS* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_TABLESPACES* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_SYS_DATAFILES* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *INNODB_CHANGED_PAGES* | *ACTIVE* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *ARCHIVE* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *Aria* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *FEDERATED* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *BLACKHOLE* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

| *FEEDBACK* | *DISABLED* | *INFORMATION SCHEMA* | *NULL* | *GPL* |

| *partition* | *ACTIVE* | *STORAGE ENGINE* | *NULL* | *GPL* |

+ - - - - - - - - - - - - - - - - - - - + - - - - - - + - - - - - - - - - - - - - - - + - - - - - - + - - - -+

(MariaDB Org., 2015)

2.12.3 Grupo de envío al registro binario.

Esto hace que la replicación notablemente más rápido. MariaDB 5.3 cuando el servidor se ejecuta con innodb_flush_logs_at_trx_commit = 1 y/o sync_binlog = 1. Se necesitan estos ajustes para garantizar que si un servidor falla, cualquier transacción que se informó como commit antes de la hora del accidente todavía estará presente en la base de datos después de la recuperación de bloqueo ("durabilidad", la "D" en ACID).

Ambas opciones también son necesarias para poder recuperar a un estado "original" después de un accidente; si los comandos no están seteados, es posible que después de un accidente la transacción no haya guardado en InnoDB/XtraDB, o viceversa, lo que rompe la replicación de esclavos.

Cuando estos valores son, en efecto, el servidor hace un sincronización con fsync() (o fdatasync()) llamada en el archivo de registro de transacciones XtraDB/archivo de registro binario COMMIT durante la cual se garantiza que los datos se almacenan de forma duradera en el disco. Un fsync() es una operación que consume tiempo, y puede limitar fácilmente el rendimiento en términos del número de confirmaciones por segundo que se pueden sostener.

- Añadida "rewrite-db" opción mysqlbinlog para cambiar la base de datos utilizada
- Información sobre los progresos de ALTER TABLE y LOAD DATA INFILE.
- JOINS y subconsultas rápidas.
- HandlerSocket conector más rápido.

Nos permite tener acceso directo a los motores de almacenamiento InnoDB/XtraDB y SPIDER.

HandlerSocket es un plugin para una administración de la base de datos NoSQL para MariaDB y que soporta también MySQL. Funciona como un proceso(demonio) de mysqld, aceptando conexiones TCP, y la ejecución de las

solicitudes de los clientes; no soporta consultas SQL. En lugar de ello, es compatible con las operaciones CRUD sencillas sobre tablas.

HandlerSocket puede ser mucho más rápido que mysqld/libmysql en algunos casos, ya que tiene una unidad de procesamiento más baja, el disco y sobrecarga de la red: esto con el fin de reducir el uso de CPU que no analiza SQL; en lotes procesa solicitudes cuando sea posible, lo que reduce aún más el uso de CPU y reduce el uso del disco.

2.12.4. Columnas dinámicas.

MariaDB permiten una columna para almacenar diferentes conjuntos de sub-columnas para cada fila de una tabla. Funciona mediante el almacenamiento de un conjunto de columnas en una burbuja y que tiene un pequeño conjunto de funciones para manipularlo estas deben ser utilizados cuando no es posible utilizar columnas regulares.

Un caso de uso típico es cuando se necesita almacenar elementos que pueden tener muchos atributos diferentes (como el tamaño, color, peso, etc.), y el conjunto de atributos posibles es muy grande y/o conozca con antelación. En ese caso, los atributos se pueden poner en columnas dinámicas.

create table articulo(
id_articulo int autoincrement primary key,
nombre_articulo varchar(100) not null,
características_dinámica blob #declaración de columna dinámica.
);
INSERT INTO articulo VALUES ('Mueble de Sala', COLUMN_CREATE('color', 'ocre', 'altura', 1.50, 'numpuertas', 2, 'acabados', 'pino tallado', 'modelo', 'Luis XV'));
SELECT nombre_articulo, COLUMN_GET(características_dinámica, 'color' as char) AS color FROM articulo;

+----------+-------+
| *nombre_articulo* | *color* |
+----------+-------+
| *Mueble de sala* | *ocre* |
+----------+-------+

- Funcionalidad GIS. que es la misma de MySQL ya que soporta índices espaciales.
- Replicación de múltiples fuentes.
- SHOW EXPLAIN da el detalle de una consulta que se ejecuta en otro hilo.

Permite obtener el detalle de una consulta que se está ejecutando.

2.13 Ventajas y Desventajas

2.13.1 Ventajas

- Es un programa open source y es desarrollada una comunidad a nivel mundial.
- Su implementación y elaboración son de bajo costo, no por ello quiere decir que sea malo.
- Capacidad de portabilidad y adaptabilidad a distintos entornos ya sean escasos recursos en hardware.
- Garantiza la integridad de los datos.
- Los procesos de ejecución de comandos como Alter Table ya son más visibles.

2.13.2. Desventajas

En el transcurso de este estudio de MySQL y sus ramas como MariaDB se ha encontrado que si bien cumple con lo antes mencionado ya que hay varios blogs de desarrolladores “famosos” que lo confirman y reportan, la desventaja que le encuentro por el momento es la falta de soporte técnico que garantice rendimiento ya que es una base de datos sencilla, liviana y fácil de usar adaptar a otros sistemas y para migraciones.

La mayoría de páginas oficiales, de gobierno la podrían utilizar si hay algún problema en lo posterior con su predecesora ya que el modelo de desarrollo planteado por el actual gobierno hace de las soluciones open source y libres sean adaptables al modelo político, económico y soberano.

2.14 Integración del Framework Django con MariaDB

2.14.1 ¿Qué es una API?

Una interfaz de programación de aplicaciones (API) es un conjunto de instrucciones de programación y las normas para el acceso a una aplicación de software basada en la Web o una herramienta Web.

Una compañía de software libera su API para el público para que otros desarrolladores de software puedan diseñar productos que funcionan con su servicio.

Por ejemplo, Amazon.com[6] lanzó su API para que los desarrolladores de sitios web puedan acceder más fácilmente a la información de productos de Amazon. Usando la API de Amazon, un sitio Web de terceros puede publicar enlaces directos a los productos de Amazon, con precios actualizados y una opción de "comprar ahora".

Una API es una interfaz de software para el software, no una interfaz de usuario. Con las API, las aplicaciones se comuniquen entre sí sin ningún conocimiento o intervención del usuario. Cuando usted compra de entradas de cine en línea y entrar en su información de la tarjeta de crédito, el sitio Web boleto para el cine utiliza una API para enviar su información de tarjeta de crédito a una aplicación remota que verifica si la información es correcta. Una vez que se confirme el pago, la aplicación remota envía una respuesta al sitio Web de entrada de cine diciendo que está bien para emitir los billetes.

Hoy existen algunas APIs muy conocidas y que tienen una predominancia en el desarrollo de aplicaciones web como son las API's de Youtube, Twitter, Facebook, etc.. como sugerencia en la siguiente dirección web se puede encontrar un recopilatorio algunas otras más: www.codecademy.com/tracks/apis.

[6] Tienda en línea de venta al detalle, fundada en julio de 1994; por Jeff Bezos

2.14.2 API's para MariaDB

En la siguiente tabla mencionaremos algunas de las DB API's más usadas y que son recomendadas para MariaDB/MySQL. Para mayor información en el pie de la imagen.

TABLA 8: API's para la implementación con MariaDB (MariaDB Org., 2015)

API	URL	Licencia	Plataformas	Descripción
MySQL para Python	http://sourceforge.net/projects/mysql-python	Licencia Pública General de GNU (GPL), Python Licencia (CNRI Python License) Licencia Pública de Zope	Independiente del SO	
PyMySQL	http://www.pymysql.org/	MIT	Independiente del SO, CPython 2.xy 3.x, PyPy, Jython, IronPython Python 2.4 a 3.2	Pure-Python centra en la simplicidad y compatibilidad. Prácticamente el 100% compatible con MySQLdb Mejor rendimiento
mxODBC	http://www.egenix.com/products/python/mxODBC/	eGenix Comercial	Windows, Linux, MacOS X, FreeBSD, Solaris, AIX, Python 02.04 a 02.07	mxODBC es compatible con el controlador ODBC de MySQL en Windows y Unix.
yodbc	http://code.google.com/p/pyodbc	MIT	Windows, Linux, MacOS X, FreeBSD, Solaris, Python 2.4 +	Mantenido activamente proyecto Open Source. Binarios precompilados están disponibles para

				Windows. Red Hat Enterprise Linux, CentOS, Fedora y han precompilado RPMs disponibles en sus repositorios Extras.
MySQL Connector/Python	https://dev.mysql.com/downloads/connector/python/	GNU GPL v2 con excepción de licencia de software libre	Python v2.6, v2.7 y v3.1 para Python 3.3 (Ver relación de versiones)	Implementa la API de Python DB 2.0 (PEP 249). Implementación de Python puro del protocolo MySQL. Activamente desarrollado y mantenido por Oracle. Incluye Django backend de base de datos.
mypysql	http://sourceforge.net/projects/mypysql/	GNU GPL v3 +	Python 3	Este módulo proporciona (todavía) incompleta funcionalidad PEP 249 Implementación en C del conector de base de datos MySQL. La mayoría de las órdenes se implementan. Todavía experimental,

				pero desarrollado activamente.
PyPyODBC (Pure Python ODBC)	http://code.google.com/p/pypyodbc	MIT	Windows, Linux, Python 2.4 a 3.3	Ejecute SQLAlchemy en PyPy´ Una secuencia de comandos de Python puro, ejecuta en CPython / IronPython / PyPy, versión 3.3 / 3.2 / 3.1 / 2.4 / 2.5 / 2.6 / 2.7, Win / Linux, de 32/64 bits. Utilizo similar como pyodbc (puede ser visto como una reimplementación de pyodbc en Python puro). Simple - todo el módulo se implementa en una única secuencia de comandos python con menos de 3.000 líneas.

2.15 MariaDB/MySQL en Django

(MariaDB, 2015)

Django intenta apoyar tantas características como sea posible en todos los backends de bases de datos. Sin embargo, no todos los backends de bases de datos son iguales, y hemos tenido que tomar decisiones de diseño en el que cuenta para apoyar y qué supuestos se puede hacer de manera segura.

2.15.1 Aspectos generales

Las conexiones persistentes

El objetivo de esta característica procura evitar la sobrecarga de volver a establecer una conexión con la base de datos en cada petición. Están controlados por el CONN_MAX_AGE parámetro que define la duración máxima de una conexión. Se puede configurar de forma independiente para cada base de datos.

El valor por defecto es 0, preservando el comportamiento histórico de cierre de la conexión de base de datos al final de cada solicitud. Para habilitar las conexiones persistentes, establecer CONN_MAX_AGE a un número positivo de segundos. Para conexiones persistentes ilimitadas, ponerlo en Ninguno.

Gestión de conexiones

Django abre una conexión con la base de datos cuando se hace primero una consulta de base de datos. Se mantiene esta conexión abierta y reutiliza en solicitudes posteriores. Django cierra la conexión una vez que se supera el tiempo máximo definido por CONN_MAX_AGE.

En detalle, Django se abre automáticamente una conexión a la base de datos cada vez que necesita uno ya sea porque se trata de la primera conexión, o porque la conexión anterior se cerró.

Al comienzo de cada solicitud, Django cierra la conexión si se ha alcanzado su tiempo de vida máximo. Si su base de datos finaliza las conexiones inactivas después de algún tiempo, se debe configurar CONN_MAX_AGE a un valor más bajo, así que Django no intenta utilizar una conexión que haya dado por terminada por el servidor de base de datos.

Al final de cada petición, Django cierra la conexión si se ha llegado a su tiempo de vida máximo o si se encuentra en un estado de error irrecuperable. Si se produce un error de base de datos durante el procesamiento de las solicitudes, Django comprueba si la conexión sigue funcionando, y la cierra si no lo hace. Por lo tanto, los errores de la base de datos afectan a lo sumo una solicitud; Si la conexión se vuelve inutilizable, la siguiente petición para crear una nueva conexión.

Cuando Django establece una conexión con la base de datos, en ella se establece parámetros apropiados, dependiendo en el backend que se utiliza. Si habilita las conexiones persistentes, esta configuración ya no se repite es cada petición. Si modifica los parámetros tales como nivel de aislamiento de la conexión o la zona horaria, debe ya sea restaurar los valores predeterminados de Django al final de cada solicitud, forzar un valor adecuado al comienzo de cada solicitud o desactivar las conexiones persistentes.

2.16 MariaDB

La compatibilidad de versiones

Django soporta MariaDB 5.1 y superior.

MariaDB 5.1 agrega la base de datos INFORMATION_SCHEMA, que contiene datos detallados sobre todo el esquema de base de datos con la característica inspectdb que Django utiliza.

2.16.1 Motores de almacenamiento

Desde MariaDB 5.1, el motor de almacenamiento por defecto es InnoDB/XtraDB. Este motor es completamente transaccional y soporta referencias de claves externas. Es probablemente la mejor opción en este momento, teniendo en cuenta que el contador autoincrement InnoDB se pierde en un reinicio de MySQL, ya que no recuerda el valor AUTO_INCREMENT, en lugar de recrear como "max (id) 1". Esto puede dar lugar a una reutilización inadvertida de AutoField valores.

Si actualiza un proyecto existente a MariaDB 5.5 y posteriormente agrega algunas tablas, asegúrese de que las tablas están utilizando el mismo motor de

almacenamiento, de lo contrario si las tablas tienen una ForeignKey entre ellas utiliza diferentes motores de almacenamiento, es posible que aparezca un error. Como el siguiente al ejecutar syncdb:

2.16.2 MySQLdb

Es la interfaz de Python para MySQL/MariaDB versión 1.2.1p2 o posterior se requiere para el pleno apoyo de MariaDB en Django.

2.16.3 Definiciones de zona horaria

Si usted planea usar Django apoyo zona horaria, utilice mysql_tzinfo_to_sql para cargar tablas de zonas horarias en la base de datos MariaDB. Esto se debe hacer sólo una vez para el servidor MariaDB, no por base de datos.

2.16.4 Conexión a la base de datos

Para la sincronización del Framework Django con MariaDB de las múltiples formas de clientes se ha tomado en cuenta para nuestra implementación al DB API Mypysql; que en lo personal me está dando mejores resultados sincronizándose con el motor InnoDB/XtraDB; cuando se genera un proyecto en Django se genera un archivo __init__.py en el escribimos lo siguiente.

```
import pymysql #importamos el API

pymysql.install_as_MySQLdb() #inicializamos el API
```

de igual manera en el archivo settings.py en la tupla DATABASES, procedemos a configurarlo.

```
DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql', #gestor de almacenamiento

'NAME': 'sisman', #nombre de la base de datos

'USER': 'root', #nombre usuario
```

'PASSWORD': 'danilo14', #contraseña

'HOST': 'localhost', #terminal donde está alojado servidor 'PORT': '3306', #puerto por defecto para MariaDB.

}

}

Con estos sencillos pasos ya tendremos sincronizado nuestro API y el framework Django.

2.16.5 Los nombres de tablas

Django utiliza nombres de tabla en minúsculas cuando se genera automáticamente los nombres de tablas a partir de modelos, así que esto es principalmente una consideración si va a reemplazar el nombre de la tabla a través de la db_table parámetro.

2.16.6 Los puntos de retorno

Tanto el ORM[7] de Django y MySQL (cuando se utiliza el motor de almacenamiento XtraDB/InnoDB) de base de datos de soporta puntos de retorno.

2.16.7 Campos de caracteres

Cualquier campo que se almacenan con los tipos de columna VARCHAR tienen su max_length limita a 255 caracteres si utiliza = únicas verdadero para el campo. Esto afecta CharField , SlugField y CommaSeparatedIntegerField .

2.17 Compatibilidad MariaDB y MySQL

En la siguiente tabla se hace una tabla comparativa propuesta por DB-Engines (Solid IT, 2014) en la que propone algunas propiedades.

[7] Mapeo de Objetos relacionales. Que trabaja sin que nosotros manipulemos con SQL nuestra base de datos.

TABLA 9: Comparación de MariaDB y MySQL para versiones 5.x

Propiedad	MariaDB	MySQL
Descripción	Un fork de MySQL con el objetivo de ofrecer un mejorado y el reemplazo de MySQL desarrollada por la comunidad.	De código abierto ampliamente usada RDBMS.
Ranking de los motores de almacenamiento. (Solid IT, 2014)	27	2
Esquema de guía. (Solid IT, 2014)	14.734	1309.550
Sitio WEB	http://mariadb.org	http://www.mysql.com
Documentación Técnica	https://mariadb.com/kb/en/mariadb/	http://dev.mysql.com/doc/
Desarrollador y Soporte	Monty Program AB	Oracle
Año de lanzamiento	2009	1995
Tipo Licencia	Open Source	Open Source
Lenguaje de implementación	C y C++	C y C++
Plataformas	FreeBSD, Linux, OS X, Solaris y Windows.	FreeBSD, Linux, OS X, Solaris y Windows.
Modelo de base de datos	Relacional DBMS (con un API para acceder directo al motor de almacenamiento)	Relacional DBMS (con un API para acceder directo al motor de almacenamiento)
Esquema de datos	Si	Si
Soporta tipos de datos float o date	Si	Si
Índices secundarios	Si	Si
Soporta SQL	Si (depende del tipo de requerimiento)	Si (depende del requerimiento)
API''s y otros métodos de conexión y acceso	ADO.NET, JDBC y ODBC	ADO.NET, JDBC y ODBC

Propiedad	MariaDB	MySQL
Soporta lenguajes de programación	Ada, C, C#, C++, D, Eiffel, Erlang, Haskell, Java, Objective-C, OCaml, Perl, PHP, Python, Ruby, Scheme y Tcl.	Ada, C, C#, C++, D, Eiffel, Erlang, Haskell, Java, Objective-C, OCaml, Perl, PHP, Python, Ruby, Scheme y Tcl.
Procedimientos almacenados y/o Triggers	Si (sintaxis con algunas mejoras)	Si
Soporta Métodos de particionamiento (clúster)	Si (a partir de la 5.4)	Si
Métodos de replicación	Master-master replication Master-slave replication MariaDB Galera Cluster	Master-master replication Master-slave replication MySQL Cluster
Control de acceso	Configurable	Configurable
ACID	Si	Si

Fuente: (MariaDB, 2015)

Para todos los propósitos prácticos, MariaDB es un fork binario en sustitución de la misma versión de MySQL (por ejemplo MySQL 5.1 -> 5.1 MariaDB , MariaDB 5.2 y MariaDB 5.3 son compatibles, MySQL 5.5 será compatible con MariaDB 5.5).

2.18 Herramientas de MariaDB

2.18.1 EMS SQL Manager for MySQL

EMS SQL Manager for MySQL es una herramienta de alto rendimiento para la administración de bases de datos MySQL y el desarrollo. Funciona con cualquier versión de MySQL desde 4.1 a la más reciente y soporta todas las últimas características incluyendo disparadores MySQL, vistas, procedimientos almacenados y funciones, llaves extranjeras InnoDB de datos Unicode, etc.

SQL Manager for MySQL permite crear/editar todos los objetos de base de datos MySQL, diseño de bases de datos MySQL visualmente, ejecutar scripts SQL, importar y exportar datos de bases de datos MySQL, MySQL administrar usuarios y sus privilegios y tiene muchas otras características útiles para la eficiente administración de MySQL.

2.18.2 PhpMyAdmin

phpMyAdmin es una herramienta de software libre escrito en PHP , destinada a manejar la administración de MySQL a través del Web Wide World. phpMyAdmin es compatible con una amplia gama de operaciones con MySQL. Las operaciones más utilizadas son compatibles con la interfaz de usuario (bases de datos de gestión, tablas, campos, relaciones, índices, usuarios, permisos, etc), mientras que usted todavía tiene la capacidad de ejecutar cualquier sentencia SQL directamente. (PhpMyAdmin , 2013)

2.18.3 DBDesigner

DBDesigner 4 es un sistema de base de datos de diseño visual de base de datos que integra el diseño, modelado, creación y mantenimiento en un único entorno sin fisuras. Combina características profesionales y una interfaz de usuario clara y sencilla de ofrecer la forma más eficiente para manejar sus bases de datos. DBDesigner 4 se comparan con productos como diseñador de Oracle, IBM Rational Rose, ERwin de Computer Associates y DataArchitect theKompany es un proyecto de código abierto disponible para Microsoft Windows y GNU/Linux. (Fab Force, 2014)

2.18.4 HeidiSQL

HeidiSQL es una herramienta útil y fiable diseñada para los desarrolladores web que utilizan el popular servidor MySQL y bases de datos SQL de Microsoft. Le permite navegar y editar los datos, crear y editar tablas, vistas, procedimientos, triggers y eventos programados. Además, puede exportar la estructura y los datos ya sea a un archivo SQL, el portapapeles o en otros servidores.

2.18.5 Umbrello

Umbrello UML Modeller es un programa de modelado de lenguaje unificado(UML) basado en tecnología KDE.[8]

UML permite crear diagramas de software y otros sistemas en un formato estándar para documentar o diseñar la estructura de sus programas.

Umbrello viene con KDE SC, incluido en todas las distribuciones de Linux y está disponible a través de su gestor de paquetes. Consulte Instalación para instalar Umbrello.

2.19 Metodología ágil de desarrollo

Aunque los creadores e impulsores de las metodologías ágiles más populares lograron desarrollar un manifiesto para el desarrollo ágil, coinciden con los principios existen una gran variedad de metodologías basadas en estos principios, cada una tiene características propias y enfatiza en algunos aspectos específicos.

2.19.1 Manifiesto para desarrollo ágil

Los principios sobre los que se basan los métodos alternativos, en un documento denominado “Manifiesto Ágil”.

Manifiesto ágil está fundamentado en los siguientes valores.

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el (Independent Signatories of The Manifesto for Agile Software Development) equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades; software que funciona más que conseguir una buena documentación.

[8] KDE Software Compilation; es una comunidad que desarrolla software libre y de código abierto con su propio entorno de escritorio para los sistemas Unix.

La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta. Los valores enumerados anteriormente inspiran los doce principios del manifiesto, estos son:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.

VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento. (Q7 Enterprises, Inc., 2014)

2.20 Ventajas y desventajas al utilizar Django con MariaDB.

2.20.1 Ventajas

- Tanto Django como MariaDB son Open Source.
- Bajo costo en requerimientos para la elaboración de bases de datos puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de Sistemas Operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor.
- Su conectividad y seguridad hacen de Django y MariaDB altamente apropiado para acceder bases de datos en Internet.

2.20.2 Desventajas

- La documentación de las API es variada y extensa con respecto a MariaDB que con otros sistemas gestores de datos.

2.21 Principales metodologías ágiles

En el mundo del desarrollo de software con su constante avaluación tenemos el día de hoy:

2.21.1 SCRUM

Scrum es una metodología ágil muy útil para el desarrollador. Se basa en la asignación de tareas diarias, reuniones rápidas y control de la evolución de los procesos. Que lleva un seguimiento de las tareas (sprints) que se están ejecutando e ir conociendo paralelamente las tareas rezagadas o retrasadas que eso repercute al equipo. La profundidad de las tareas que se asignan en SCRUM tiende a ser incremental, y esto coincide exactamente con el progreso normal de un desarrollo. (Hundhausen, 2012)

2.21.2 XP o Xtream Programming

Programación Extrema se suele utilizar en equipos con pocos desarrolladores, pocos procesos abiertos al mismo tiempo. La metodología trata de diseñar, implementar, programar e implantar lo más "rápido" posible en equipos de programadores, suprimiendo un poco la documentación que en ciertos momentos se hace complejo y los procedimientos tradicionales. Se fundamenta en la comunicación del equipo y estar predispuesto al ensayo y error inherentes en un desarrollador. La ventaja más importante es la capacidad de respuesta ante los errores propios del desarrollo.

2.21.3 Desarrollo Lean

Lean Software Development, también conocido como Lean Programming es una metodología que contiene técnicas de desarrollo ágil con el fin de conseguir exactamente lo que necesita el cliente. Es una evolución del "Método Toyota de Producción" aplicado al desarrollo de software y que está muy de moda entre los equipos de desarrollo en startups. Principalmente consiste en hacer períodos de desarrollo que se van incrementando en los que se post-ponen las decisiones lo más posible hasta haber obtenido un feedback del cliente y así reaccionar lo más rápido y eficazmente posible a sus necesidades.

Se caracteriza por estar formado de un equipo eficaz y comprometido con el principio de aprendizaje continuo sobre el producto.

El Desarrollo Lean una metodología fantástica para startups que se la está actualmente utilizando para videojuegos y/o aplicaciones para móviles.

2.22 Metodología de SCRUM

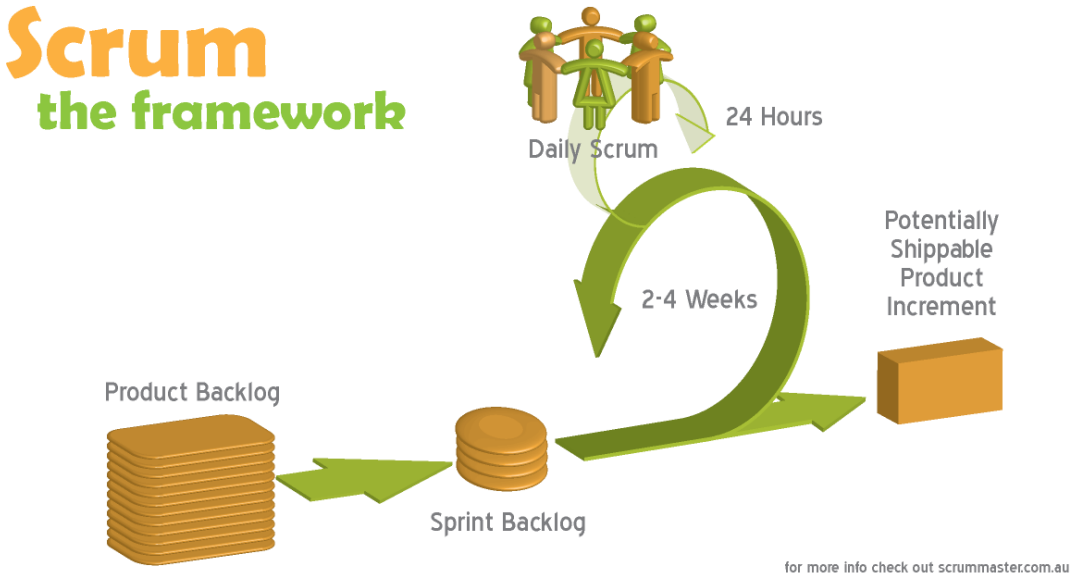


ILUSTRACIÓN 9: Ciclo de vida de SCRUM

Fuente: scrummaster.com.au

Scrum es una guía para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados para el desarrollo ágil de software.

Puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas: Scrum de Scrums, una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (normalmente de 30 días).

Cada uno de estos periodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto. Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.

2.22.1 Artefactos y eventos de SCRUM

- Pila de producto (requerimientos funcionales y no funcionales).
- Pila del sprint (Responsabilidades).
- Sprints.
- Gráfica de producto (Burn Up).
- Gráfica de avance (Burn Down).
- Reunión técnica (Sprint Planning Meeting).
- Reunión de cierre de sprint y entrega del incremento (Block Log).

2.22.2 Control de evolución del proyecto

Scrum controla empírica y adaptablemente la evolución del proyecto usando: Revisión de las Iteraciones: Al fin de cada iteración se lleva a cabo una revisión con todas las personas implicadas en el proyecto.

Incremental:

El desarrollo incremental implica que al final de cada iteración se usa una parte del producto que se puede inspeccionar y evaluar.

Desarrollo evolutivo:

Los modelos ágiles se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos. Intentar predecir en las fases iniciales ¿cómo será el producto final no es realista?

Auto-organización:

En Scrum los equipos son auto-organizados (no auto-dirigidos), con decisión suficiente para tomar las acciones que consideren oportunas.

Colaboración:

El entorno de trabajo ágil facilita la colaboración del equipo.

2.22.3 Visión del proceso

Scrum denomina “sprint” a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días. El sprint es por tanto el núcleo central que proporciona la base de desarrollo iterativo e incremental.

Los elementos que conforman el desarrollo Scrum son:

Las reuniones

- Planificación de sprint: Previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en esa iteración.

- Reunión diaria: Breve revisión del equipo del trabajo realizado hasta la fecha y la previsión para el día siguiente.

- Revisión de sprint: Análisis y revisión del incremento generado.

Valores

Scrum es una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil.

- Delegación de atribuciones (empowerment) al equipo para que pueda auto-organizarse y tomar las decisiones sobre el desarrollo.
- Respeto entre las personas. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.
- Responsabilidad y auto-disciplina (no disciplina impuesta).
- Trabajo centrado en el desarrollo de lo comprometido.
- Información, transparencia y visibilidad del desarrollo del proyecto.

CAPÍTULO III

3 DISEÑO ARQUITECTÓNICO DE LA SOLUCIÓN

3.1 Herramientas de desarrollo

Las herramientas a utilizar se describen a continuación:

3.1.1 Plataforma

Distribución GNU/Linux Debian 7[9]

ILUSTRACIÓN 10: Interfaz de la distribución Debian 7
Fuente: (Debian, 2013)

Debian GNU/Linux es una distribución del sistema operativo Linux, junto a numerosos paquetes que se ejecutan en él.

Debian GNU / Linux es:

- Completo: Debian incluye más de 37400 paquetes de software en la actualidad. Los usuarios pueden seleccionar qué paquetes instalar; Debian proporciona una herramienta para este propósito. Usted puede encontrar una lista y descripciones de los paquetes actualmente disponibles con Debian en cualquiera de los servidores réplica de Debian.

[9] Distro que se la conoce con el nombre de wheezy, fue liberada el 4 de mayo de 2013.

- Libre de usar y redistribuir: No hay participación en el consorcio, ni pago solicitado para participar en su distribución y desarrollo. Todos los paquetes que formalmente son parte de Debian GNU/Linux son libres para ser redistribuidos, normalmente bajo los términos especificados por la Licencia Pública General GNU.
- Los archivos FTP de Debian también tienen aproximadamente 187 paquetes de software (en las secciones non-free y contrib), que son distribuibles bajo términos específicos que se incluyen en cada paquete. (Debian, 2013)
- Dinámico: Con alrededor de 990 voluntarios constantemente contribuyendo con código nuevo y mejorado, Debian evoluciona rápidamente. Los archivos FTP se actualizan dos veces al día.

La mayoría de los usuarios de Linux se ejecutan una distribución específica de Linux, como GNU/Linux Debian. Con la particularidad de que el usuario puede compilar el kernel y adaptarlo a sus necesidades; se pueden obtener el código fuente para muchas aplicaciones de la misma forma, compilar los programas y luego instalarlos en sus sistemas.

De programas complicados, este proceso puede ser no sólo tiempo, sino a errores. Para evitarlo, los usuarios a menudo optan por obtener el sistema operativo y los paquetes de la aplicación de uno de los distribuidores de Linux.

Lo que distingue a las distintas distribuciones de Linux son los programas, los protocolos y las prácticas que utilizan para empaquetar, instalar, y los paquetes de aplicaciones de seguimiento de los sistemas de los usuarios, junto con herramientas de instalación y mantenimiento, documentación, y otros servicios.

3.1.2 Lenguaje de Programación

Python 3

Lenguaje de programación de propósito general, orientado a objetos, que también puede utilizarse para el desarrollo web.

Lenguaje de programación de propósito general, cuya expansión y popularidad es relativamente reciente. Se trata de Python, una apuesta por la simplicidad, versatilidad y rapidez de desarrollo.

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

3.1.3 Entorno

PyCharm IDE [10]

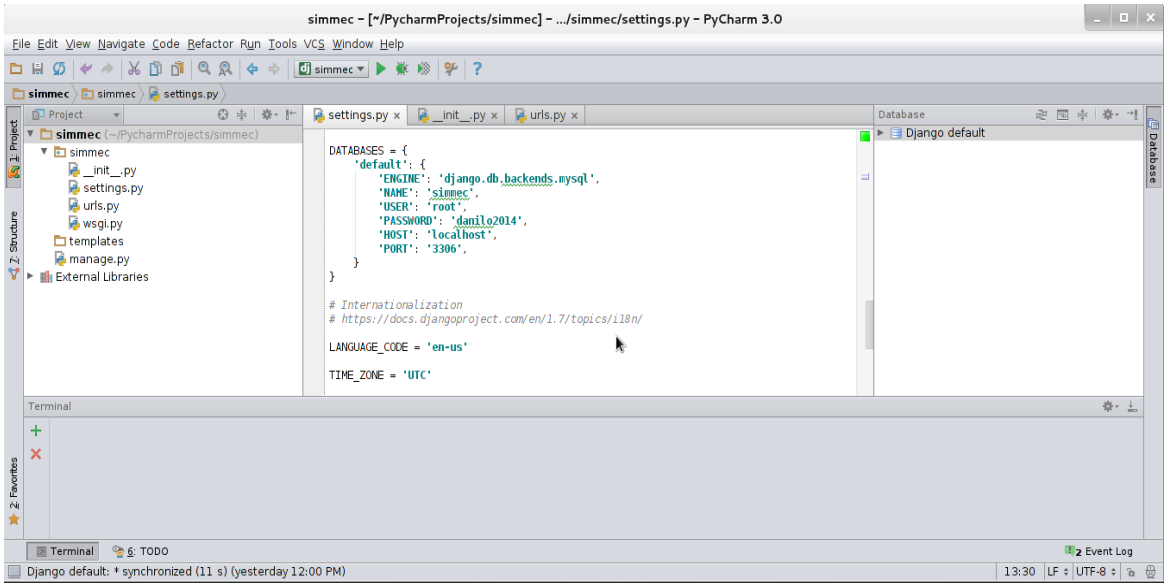


ILUSTRACIÓN 11: Interfaz del IDE Pycharm 3.0.

Fuente: Autor

Es uno de los mejores IDEs de python. PyCharm ofrece asistencia código, análisis de código, navegación del proyecto, desarrollo web con Django, depurador gráfico, probador de la unidad integrada, integración VCS/DVCS. PyCharm es desarrollado por JetBrains.

PyCharm es el desarrollo con el lenguaje de programación Python. Y funciona en Windows, Mac OS X y Linux. PyCharm Free Community Edition se lo puede utilizar sin licencia.

[10] La versión cummunity aunque también dispone de la versión de pago.

PyCharm fue lanzado al mercado de los IDE's de Python para competir con PyDev de Eclipse y Komodo IDE. La versión beta del producto fue lanzado en julio de 2010, y la versión 1.0 llegando a los 3 meses después.

3.1.4 Modelado lógico

Umbrello

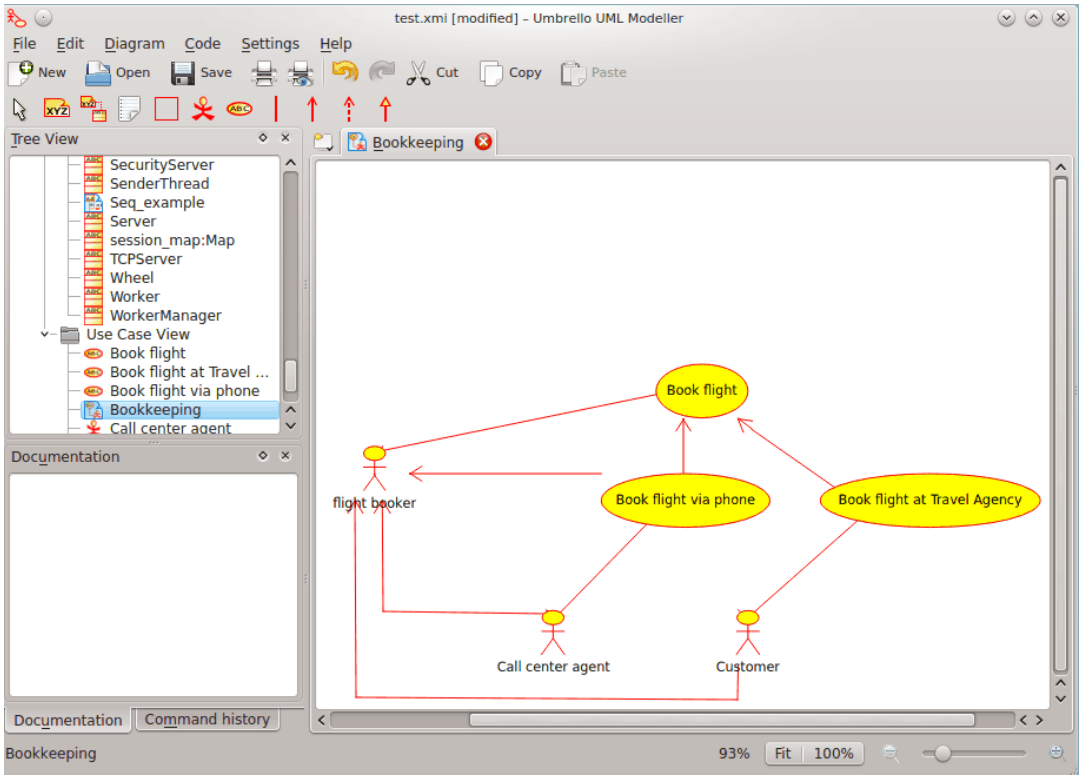


ILUSTRACIÓN 12: Interfaz de Umbrello

Fuente: umbrello.org

Umbrello UML Modeller es un programa de modelado de lenguaje unificado (UML) basado en tecnología KDE.

UML permite crear diagramas de software y otros sistemas en un formato estándar para documentar o diseñar la estructura de sus programas.

Umbrello viene con KDE SC, incluido en todas las distribuciones de Linux y está disponible a través de su gestor de paquetes. Consulte Instalación para instalar Umbrello.

3.1.5 Pencil

Pencil es un proyecto de open source que brinda a los usuario las herramientas tipo widget para diseñar el prototipo de las interfaces de los diferentes GUI en diferentes plataformas como: Android, IOS, WEB, Mobile.

ILUSTRACIÓN 13: Interfaz de Pencil

Fuente: pencil.evolus.vn

3.1.6 Servidor HTTP

Apache

Acrónimo de "a patchy server"(Un servidor de parches). Servidor web de distribución libre y de código abierto, siendo el más popular del mundo desde abril de 1996, con una penetración actual del 50% del total de servidores web del mundo.

La principal competencia de Apache es el IIS (Microsoft Internet Information Services) de Microsoft.

Apache es desarrollado y mantenido por una comunidad abierta de desarrolladores bajo el auspicio de la Apache Software Foundation.

La aplicación permite ejecutarse en múltiples sistemas operativos como Windows, Novell NetWare, Mac OS X y los sistemas basados en Unix. (Apache Org, 2014)

Características de Apache

(Apache Org, 2014)

- Soporte para los lenguajes perl, python, tcl y PHP.
- Módulos de autenticación: mod_access, mod_auth y mod_digest.
- Soporte para SSL y TLS.
- Permite la configuración de mensajes de errores personalizados y negociación de contenido.
- Permite autenticación de base de datos basada en SGBD.

CAPÍTULO IV

4 DISEÑO Y DESARROLLO E IMPLEMENTACIÓN DEL APLICATIVO.

4.1 Análisis y Definición de los requerimientos con Eventos (“Sprints”)

Este capítulo describe la implementación de la metodología de trabajo scrum en la Empresa Pública de Faenamiento y Productos Cárnicos de Ibarra para la gestión del desarrollo el proyecto Sistema de Mantenimiento Mecánico - SIMMEC.

Incluye junto con la descripción de este ciclo de vida iterativo e incremental para el proyecto, los artefactos o documentos con los que se gestionan las tareas de adquisición y suministro: requisitos, monitorización y seguimiento del avance, así como las responsabilidades y compromisos de los participantes en el proyecto.

4.1.1 Propósito de este documento

Facilitar la información de referencia necesaria a las personas implicadas en el desarrollo del sistema SIMMEC.

4.1.2 Alcance

El proyecto servirá para llevar un historial del mantenimiento mecánico de la maquinaria a empresas que se dedique al brindar el servicio de faenamiento y las personas que tienen que ver con el Sistema de Mantenimiento Mecánico SIMMEC. Y todas las demás puestas en la sección. 0

4.1.3 Descripción General de la Metodología

Fundamentos

Las principales razones del uso de un ciclo de desarrollo iterativo e incremental de tipo scrum para la ejecución de este proyecto son:

-Sistema modular. Las características del sistema SIMMEC permiten desarrollar una base funcional mínima y sobre ella ir incrementando las funcionalidades o modificando el comportamiento o apariencia de las ya implementadas.

- Entregas frecuentes y continuas al cliente de los módulos terminados, de forma que puede disponer de una funcionalidad básica en un tiempo mínimo y a partir de ahí un incremento y mejora continua del sistema.
- Previsible inestabilidad de requisitos.
  - Es posible que el sistema incorpore más funcionalidades de las inicialmente identificadas.
  - Es posible que durante la ejecución del proyecto se altere el orden en el que se desean recibir los módulos o historias de usuario terminadas.
  - Para el cliente resulta difícil precisar cuál será la dimensión completa del sistema, y su crecimiento puede continuarse en el tiempo suspenderse o detenerse.
- La información no esté disponible al momento de ponerla el modulo en producción.
- El cambio de autoridades en las empresas en la que se va a implementar.
- Reformas al presupuesto en los que se refieren a recortes de la empresa.

Valores de trabajo

Los valores que deben ser practicados por todos los miembros involucrados en el desarrollo y que hacen posible que la metodología Scrum tenga éxito son:

- Autonomía del equipo
- Respeto en el equipo
- Responsabilidad y auto-disciplina
- Foco en la tarea
- Información transparencia y visibilidad.

Personas y roles del proyecto.

Los siguientes actores están relacionados con el proyecto.

TABLA 10: Equipo SCRUM

Integrante	Contacto	Rol
Kléver Taboada	062546230	Coordinador/Scrum Manager
Luis Ramírez	062546230	Técnico
Xavier Rea	0986099536	Asesor
Danilo Chávez	0989313653	Desarrollador

Fuente: Autor

4.1.4 Artefactos

- Documentos
  - Pila de producto o Product Backlog
  - Pila de sprint o Sprint Backlog
- Sprint
- Incremento
- Gráficas para registro y seguimiento del avance**.**
  - Gráfica de producto o Burn Up
  - Gráfica de avance o Burn Down.
- Comunicación y reporting directo.
- Reunión de inicio de sprint
- Reunión técnica diaria
- Reunión de correcciones.
- Reunión de cierre de sprint y entrega del incremento
- Pila de producto

4.1.5 Responsabilidades y actividades del equipo.

Responsabilidad del gestor de producto

- Registrar la lista de pila del producto de las historias de usuario que definen el sistema.
- Mantenimiento actualizado de la pila del producto en todo momento durante la ejecución del proyecto.
  - Orden en el que desea quiere recibir terminada cada historia de usuario.
  - Incorporación / eliminación /modificaciones de las historias o de su orden de prioridad.
  - Disponibilidad: la información y los más

Responsabilidades del Scrum Manager

Supervisión de la pila de producto, y comunicación con el gestor del producto para pedirle aclaración de las dudas que pueda tener, o asesorar para la subsanación de las deficiencias que observe.

Responsabilidades del equipo técnico

Conocimiento y comprensión actualizada de la pila del producto.

Resolución de dudas o comunicación de sugerencias con el trabajo diario de las experiencias obtenidas.

Responsabilidades del resto de implicados

Conocimiento y comprensión actualizada de la pila del producto.

Resolución de dudas o comunicación de sugerencias con el equipo de trabajo de ser caso el coordinador.

Si lo necesita, el gestor de producto puede solicitar asesoría al Coordinador del proyecto o personal técnico del equipo para conocer la estimación temprana de las historias de usuario cuyo tamaño aproximado le presenten dudas.

TABLA 11: Cuadro de prioridades asignadas al equipo

Id.	Prioridad	Descripción	Estimado	Responsable
1	Alta	Control de la información con datos reales. Documento de procesos de mantenimiento.	30	Klever Taboada
2	Alta	Infraestructura tecnológica. Interfaz de Usuario. Registro de maquinaria. Control de flujo y orden de	120	Danilo Chávez
3	Alta	Características técnicas de la maquinaria y herramienta.	30	Luís Ramírez
4	Alta	Revisión de los objetivos alcanzados.	30	Xavier Rea

Fuente: Autor

4.1.6 Pila del Sprint

Es el documento de registro de los requisitos detallados o tareas que va a desarrollar el equipo técnico en la iteración (actual o que está preparándose para comenzar)

Responsabilidades del gestor de producto

Presencia en las reuniones en las que el equipo elabora la pila del sprint. Resolución de dudas sobre las historias de usuario que se descomponen en la pila del sprint.

Responsabilidades del Scrum Manager

Supervisión y asesoría en la elaboración de la pila de la pila del sprint.

Responsabilidades del equipo técnico:

- Elaboración de la pila del sprint.
- Resolución de dudas o comunicación de sugerencias sobre las historias de usuario con el gestor del producto.

SPRINT	INICIO	DURACIÓN
6	1-ene-15	22

	J	V	L	M	X	J	V	L	M	X	J	V	L	M	X	J	V	L	M	X	J	V	L	
	1-ene	2-ene	5-ene	6-ene	7-ene	8-ene	9-ene	12-ene	13-ene	14-ene	15-ene	16-ene	19-ene	20-ene	21-ene	22-ene	23-ene	26-ene	27-ene	28-ene	29-ene	30-ene	2-feb	
Tareas pendientes	17	17	17	17	17	6	6	6	11	6	6	11	6	6	11	6	6	11	6	6	6	1		
Horas de trabajo pendientes	138	117	123	123	123	9	9	9	14	9	9	14	9	9	14	9	9	14	9	9	6	9		

PILA DEL SPRINT					ESFUERZO																				
Backlog	Tarea	Tipo	Estad	Responsal																					
Inicial	Establecer actores	Análisis	Terminada	Klever	16	16	2	2	2																
I-1	Establecer procesos	Prototipado	Terminada	Danilo	8	4	8	8	8																
I-2	Establecer Entidades	Pruebas	Terminada	Danilo	5	8	8	8	8																
I-3	Establecer Software Ofinatico	Codificación	Terminada	Danilo	4	4	8	8	8																
I-4	Establecer calendarios	Codificación	Terminada	Klever	16	16	8	8	8																
I-5	Instalar distro Debian	Pruebas	Terminada	Danilo	8	8	8	8	8																
I-6	Instalar Python	Codificación	Terminada	Danilo	8	4	8	8	8																
I-7	Instalar middlewares	Codificación	Terminada	Danilo	8	8	8	8	8																
I-8	Instalar Django	Pruebas	Terminada	Danilo	8	4	8	8	8																
I-9	Instalar Base de datos	Pruebas	Terminada	Danilo	8	8	8	8	8																
I-10	Corrección de bugs de instalación	Codificación	Terminada	Danilo	8	4	8	8	8																
I-11	Instalar middlewares II	Codificación	Terminada	Danilo	8	8	8	8	8																
I-12	Modelo Entidad Relación	Codificación	Terminada	Danilo	8	4	8	8	8																
I-13	Modelo Prototipo	Codificación	Terminada	Danilo	8	8	8	8	8																
I-14	Modelo Frontend	Codificación	Terminada	Danilo	8	4	8	8	8																
I-15	Respaldo	Codificación	Terminada	Danilo	8	8	8	8	8																
Segundo Registro																									
S-1	Desarrollo de las interfaces	Codificación	Terminada	Danilo						2	2	2	1												
S-2	Desarrollo de los modelos	Codificación	Terminada	Danilo						2	2	2	1												
S-3	Desarrollo de las vistas	Codificación	Terminada	Danilo						2	2	2	1												
S-4	Desarrollo de los controladores	Codificación	Terminada	Danilo						1	1	1	1												
S-5	Configuración URLs	Codificación	Terminada	Danilo						1	1	1	1												
Tercero Bodega																									
T-1	Desarrollo de las interfaces	Codificación	Terminada	Danilo									2	2	2	1									
T-2	Desarrollo de los modelos	Codificación	Terminada	Danilo									2	2	2	1									
T-3	Desarrollo de las vistas	Codificación	Terminada	Danilo									2	2	2	1									
T-4	Desarrollo de los controladores	Codificación	Terminada	Danilo									1	1	1	1									
T-1	Configuración URLs	Codificación	Terminada	Danilo									1	1	1	1									
Cuarto Proformas y Pedidos																									
C-1	Desarrollo de las interfaces	Codificación	Terminada	Danilo												2	2	2	1						
C-2	Desarrollo de los modelos	Codificación	Terminada	Danilo												2	2	2	1						
C-3	Desarrollo de las vistas	Codificación	Terminada	Danilo												2	2	2	1						
C-4	Desarrollo de los controladores	Codificación	Terminada	Danilo												1	1	1	1						
C-5	Configuración URLs	Codificación	Terminada	Danilo												1	1	1	1						
Quinto Auditoría																									
Q-1	Desarrollo de las interfaces	Codificación	Terminada	Danilo															2	2	2	1			
Q-2	Desarrollo de los modelos	Codificación	Terminada	Danilo															2	2	2	1			
Q-3	Desarrollo de las vistas	Codificación	Terminada	Danilo															2	2	2	1			
Q-4	Desarrollo de los controladores	Codificación	Terminada	Danilo															1	1	1	1			
Q-5	Configuración URLs	Codificación	Terminada	Danilo															1	1	1	1			
Sexto Operador																									
X-1	Desarrollo de las interfaces	Codificación	Terminada	Danilo																		2	2	2	1
X-2	Desarrollo de los modelos	Codificación	Terminada	Danilo																		2	2	2	1
X-3	Desarrollo de las vistas	Codificación	Terminada	Danilo																		2	2	2	1
X-4	Desarrollo de los controladores	Codificación	Terminada	Danilo																		1	1	1	1
X-5	Configuración URLs	Codificación	Terminada	Danilo																		1	1	1	1
Septimo Documentación																									
P-1	Elaboración del docuemtación	Codificación	Terminada	Danilo	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	9

ILUSTRACIÓN 14: Sprint Blocklog

Fuente: Autor

Sprint

Cada una de las iteraciones del ciclo de vida iterativo Scrum. La duración de cada sprint es de tres días laborables en modo de bucle por cada sprint. Que para nuestra solución son cinco.

Incremento

Parte o subsistema que se produce en un sprint y se entrega al gestor del producto completamente terminado y operativo.

Gráfica de producto (Burn Up)

Representación gráfica del plan de producto previsto por el gestor de producto. Es una gráfica que representa los temas o del sistema en el orden que se desean, y el tiempo en el que se prevé su ejecución.

Responsabilidades del gestor de producto

- Elaboración.
- Mantenimiento actualizado en todo momento durante la ejecución del proyecto.
  - Orden en el que desea disponer de los temas o “epics” del sistema, e hitos del producto (versiones).
  - Incorporación / eliminación /modificaciones de los temas, de su orden de prioridad, estimaciones o hitos.
  - Disponibilidad: El coordinador y el equipo se comunicaran por medio de los correos o llamada de trabajo.

Responsabilidades del Scrum Manager

Supervisión del gráfico de producto, y comunicación con el gestor del producto para pedirle aclaración de las dudas que pueda tener, o asesorarle para la subsanación de las deficiencias que observe.

Responsabilidades del equipo técnico

- Conocimiento y comprensión actualizada del plan del producto.
- Resolución de dudas o comunicación de sugerencias con el coordinador de la empresa.

Responsabilidades del resto de implicados

- Conocimiento y comprensión actualizados del plan de producto.
- Resolución de dudas o comunicación de sugerencias con del desarrollador del equipo.
- La línea de velocidad proyecta sobre el eje X la fecha o sprint en el que previsiblemente se completarán las versiones representadas en el eje Y.

Gráfica de avance (Burn Down)

Gráfico que muestra el estado de avance del trabajo del sprint en curso.

Responsabilidades del gestor de producto

Sin responsabilidades específicas, más allá de mantenerse regularmente informado del avance del sprint y disponible para atender decisiones para la resolución de opciones en sprints sobrevalorados o infravalorados (la gráfica de avance predice una entrega anterior o posterior a la fecha prevista)

Responsabilidades del Scrum Manager

- Supervisión de la actualización diaria por parte del equipo.
- Responsabilidades del equipo técnico
- Actualización diaria del gráfico de avance.
- Desarrollo de la solución.

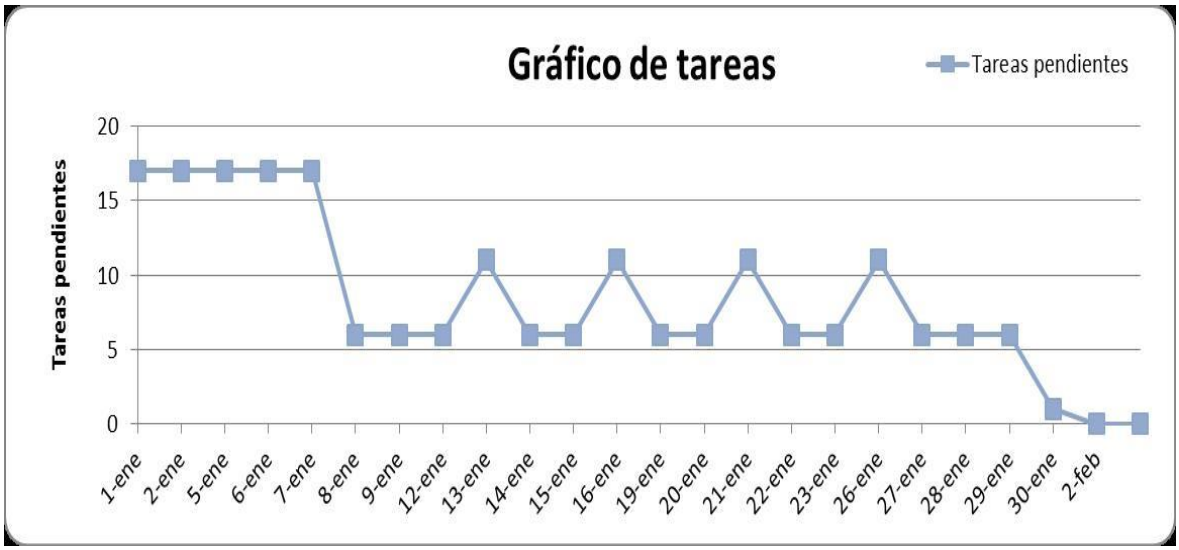


ILUSTRACIÓN 15: Gráfico de tareas de la Metodología Scrum

Fuente: Autor

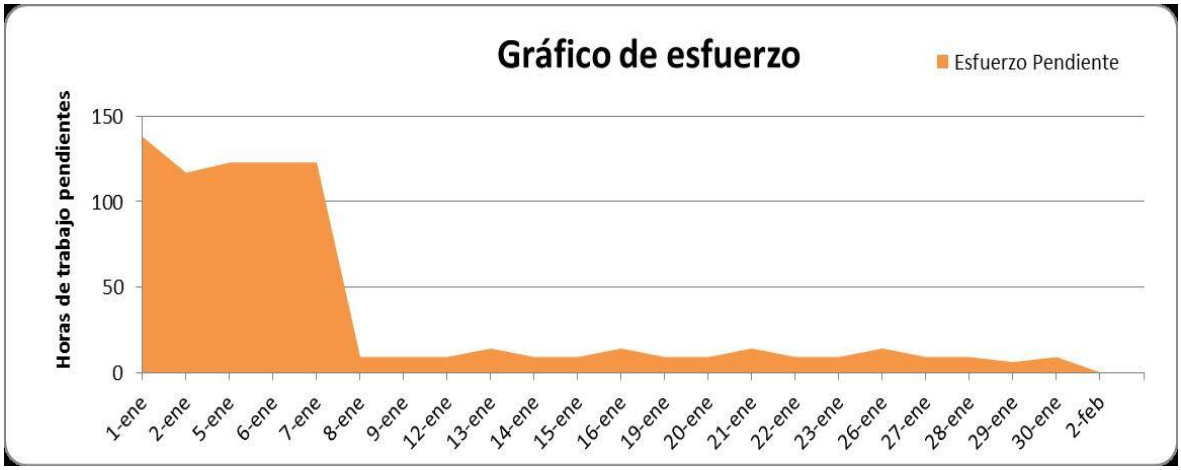


ILUSTRACIÓN 16: Gráfico de esfuerzo

Fuente: Autor

Reunión de inicio de sprint

Reunión para determinar las funcionalidades o historias de usuario que se van a incluir en el próximo incremento.

Responsabilidades del gestor de producto

Asistencia a la reunión.

Exposición y explicación de las historias que necesita para la próxima iteración y posibles restricciones de fechas que pudiera tener.

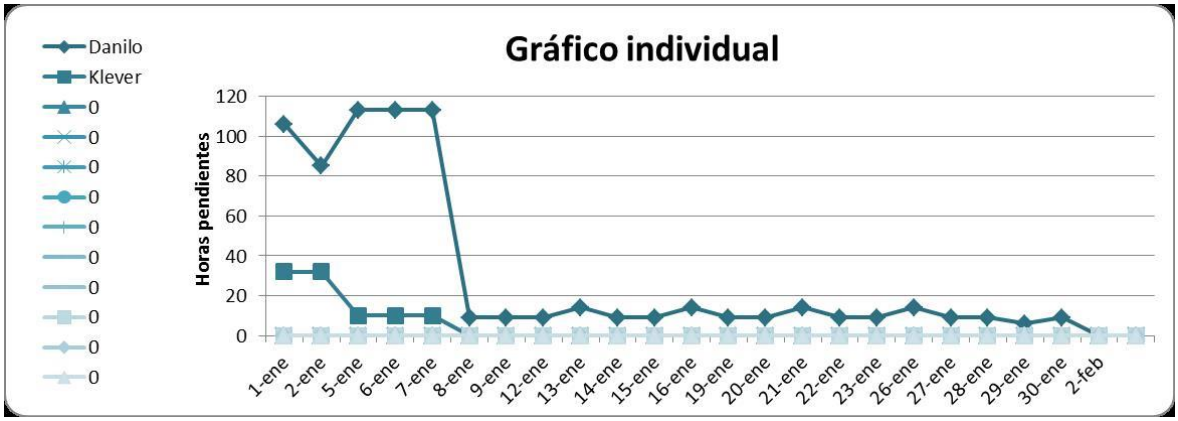
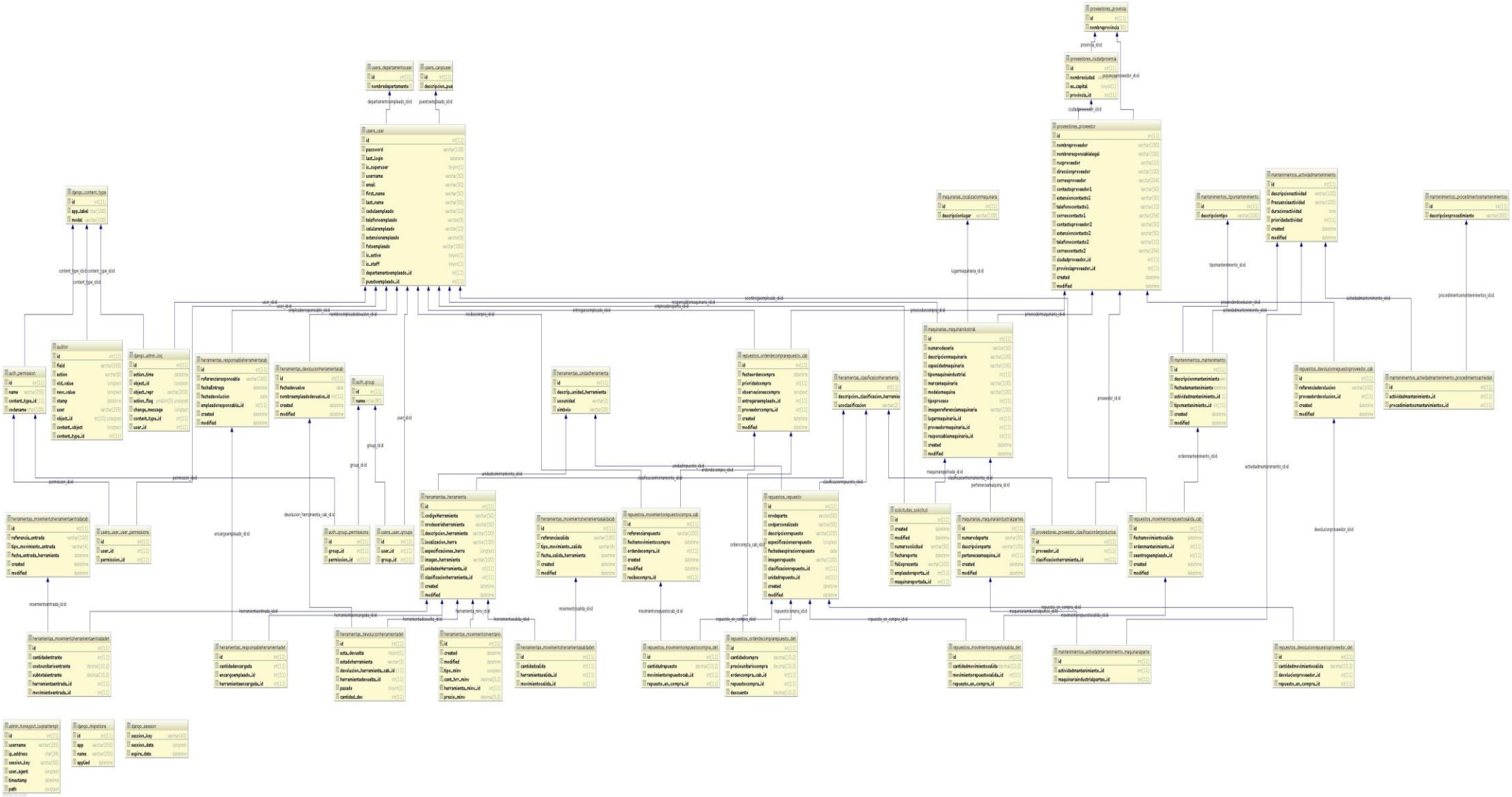


ILUSTRACIÓN 17: Gráfico individual

Fuente: Autor

4.2 Vista lógica de la Base de Datos



4.3 Modelo de casos de uso

A continuación presentaremos los modelos de caso de uso general para el Sistema de Mantenimiento Mecánico SIMMEC.

Los casos de uso es un diagrama que indica acciones y procesos que cumple un sistema cuando se le ordena un evento.

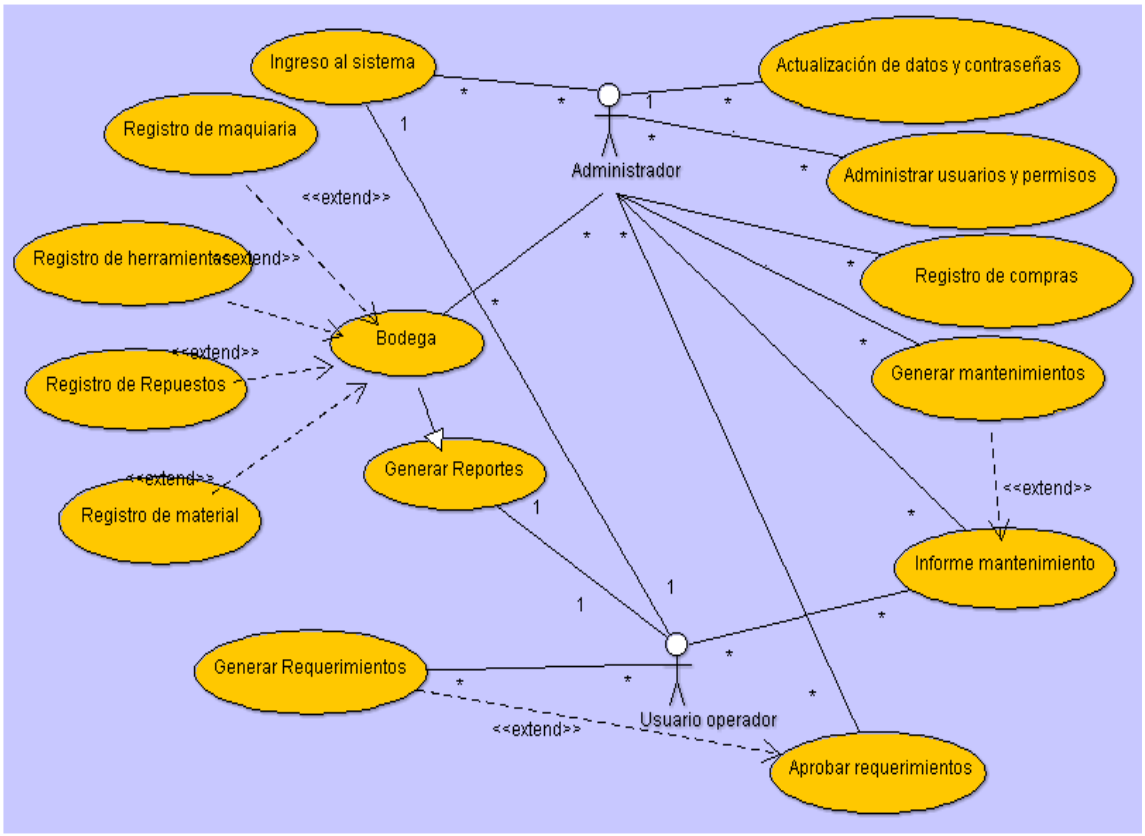


ILUSTRACIÓN 18: Diagrama general de caso de uso.

Fuente: Autor

TABLA 12: Descripción de los caso de uso general

CASO DE USO	DESCRIPCIÓN
Ingreso al sistema	Realiza el ingreso de usuario y su contraseña para que el programa active las funciones que le corresponden.
Registro de Maquinaria	Realizar el registro de la maquinaria, con códigos de fábrica y con los asignados por la empresa.
Registro de Herramienta	Realizar el registro de la herramienta que sirva para tal propósito.
Registro de Repuestos	Realizar el registro de los repuestos de la maquinaria con partes.
Registro de Material	Realizar el registro del material disponible para ser utilizado en el mantenimiento.
Bodega	Manejo de una Bodega
Generar Reportes	Generar los reportes que se generen de la utilización de los mantenimientos y existencias.
Generar Requerimientos	Realizar la petición para nueva adquisición de bienes o servicios.
Actualización de datos y contraseñas	Actualizar los datos de los usuarios que se lo requiera.
Administrar usuarios y permisos	Realizar el alta del sistema y asignar permisos que correspondan.
Registro de compras	Realizar los registros de compra del bien o servicio con su respectiva factura.
Generar Mantenimiento	Establecer el tipo de mantenimiento y tiempo para el mantenimiento dependiendo de la jefatura técnica.
Informe Mantenimiento	Realizar un informe de la acción realizada con un bien.
Aprobar requerimientos	Realizar la autorización de requerimiento del bien o servicio.

Fuente: Autor

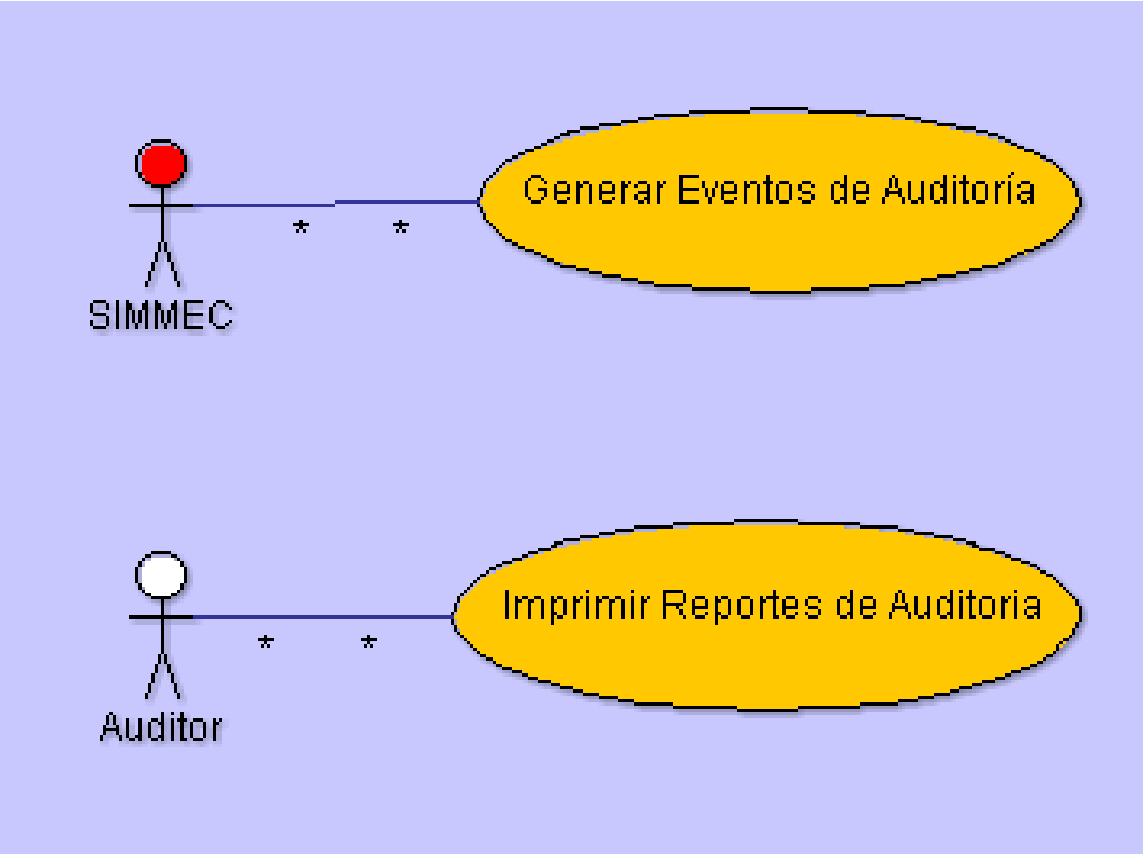


ILUSTRACIÓN 19: Diagrama de caso de uso del proceso Auditoría.

Fuente: Autor

TABLA 13: Casos de uso para subproceso auditoria

CASO DE USO	DESCRIPCIÓN
Generar eventos de auditoría	Registrar registros de auditoria.
Imprimir Reportes de Auditoria	Generar archivos de la auditoria.

Fuente: Autor

CAPÍTULO V

5 ANÁLISIS COSTO, ANÁLISIS IMPACTO, CONCLUSIONES Y RECOMENDACIONES.

En este capítulo se detalla lo siguiente.

5.1 ANÁLISIS DE COSTO

5.1.1 Costos en hardware.

TABLA 14: Costos en hardware.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Computador	689,00	689,00
Impresora y copiadora	250,00	250,00
Servidor Base de datos	0,00	5000,00
Servidor de aplicaciones	0,00	5000,00
TOTAL	939,00	10939,00

Fuente: Autor

5.1.2 Costos en software

TABLA 15: Costos de software.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Umbrello UML y ArgoUML (diagramas de base de datos y casos de uso)	0,00	0,00
PyCharm community IDE django-python	0,00	0,00
RDBMS MariaDB	0,00	0,00
GNU/Linux Debian 7	0,00	0,00
TOTAL	0,00	0,00

Fuente: Autor

5.1.3 Costo servicios

TABLA 16: Costos servicios.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Internet 2Mb (mes)	20,00	20,00
Curso framework django	120,00	120,00
Heroku servidor de aplicaciones	0,00	0,00
Fotocopias, anillados y empastados.	120,00	120,00
TOTAL	260,00	260,0

Fuente: Autor

5.1.4 Costos materiales de oficina

TABLA 17: Costos materiales de oficina.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Resmas papel bond A4	20,00	20,00
CD’s, esferos, memotips.	40,00	40,00
TOTAL	60,00	60,00

Fuente: Autor

5.1.5 Costo de desarrollo

TABLA 18: Costo de desarrollo.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Costo Tesista	4.800,00	4.800,00
TOTAL	4.800,00	4.800,00

Fuente: Autor

5.1.6 Costo Total

TABLA 19: Costo total del análisis de costo.

DESCRIPCIÓN	COSTO REAL	COSTO REFERENCIAL
Costo Hardware	939,00	10939,00
Costo Software	0,00	0,00
Costo Servicios	260,00	260,00
Costo materiales de oficina	60,00	60,00
Costo de desarrollo	4800,00	4800,00
TOTAL GENERAL	6.059,00	16.059,00

Fuente: Autor

5.2 ANÁLISIS DE IMPACTO

5.2.1 Ámbito operativo

El sistema al ser implementado se podrá apreciar un cambio en la forma de trabajo de las personas que están implicadas en dicho proceso al ahorrarles tiempo, evitar que la maquinaria se vea sobre usada sin un mantenimiento previo ya que los trabajadores olvidaban su mantenimiento.

TABLA 20: Impacto Ámbito económico.

	Tiempo	Repuesto y material	Observación
Sin SIMMEC	Verificar permanentemente la maquinaria.	Si los había, o por lo contrario paralizar labores de faenado.	Usuarios molestos, quejas y extender el horario de salida del personal
SIMMEC	2 a 3 horas asignando mantenimiento y observar las existencias.	Tener ya en existencia.	Usuarios informados de los mantenimientos.

Fuente: Autor

5.2.2 Ámbito Social

El sistema no ha podido ser implemento por cuanto está en versión de prueba, por tal motivo no se ha podido medir el impacto en el ámbito señalado.

5.2.3 Ámbito Tecnológico

Con la implementación se logrará observar que se dejó el esfero, el papel y la hoja de cálculo de programas de ofimática y se lo reemplazó por un computador que el administrador técnico de la planta industrial se pueda transportar sin problemas.

TABLA 21: Mantenimientos programados con y sin sistema.

	Tareas Programadas	**Tareas ejecutadas**	**Tiempo (horas)**
Sin SIMMEC	15	11	4
Con SIMMEC	15	15	2

Fuente: Autor

A continuación se muestra la tabla de la maquinaria que era utilizada para dicho fin.

TABLA 22: Maquinaría con mantenimiento diario.

Maquinaria	**Mantenimientos**	**Herramientas**	**Material**
Chamuscador	5 diferentes situaciones	Estuvo disponible en buen estado.	Existencia verificable.
Sierra corta canales	5 por diferentes ocasiones.	Disponible y en buen estado.	Las hojas se las adquiere con anticipación.
Tina de escaldado.	4 por diferentes ocasiones y 1 por imprevisto, se generó un informe del mantenimiento.	Estuvo disponible y en buen estado	Mantener la existencia del líquido ablandador de las cerdas del ganado porcino.

Fuente: Autor

5.2.4 Ámbito Ambiental

Con la implementación del sistema se logrará observar que la asepsia del lugar de trabajo tuvo un mejoramiento ya que el mantenimiento se lo procura hacer cuando no hay faenamiento o antes de iniciar las etapas de los faenamientos y los trabajadores no manipulan maquinaria con fluidos propios de los animales sacrificados. Que serán apreciables en la planta de la empresa.

Por otro lado como se muestra a continuación se tendría un ahorro de papel principalmente ya que otra sería útil de oficina en menor cantidad.

TABLA 23: Impacto ambiental

	Resmas papel (mes)	Precio unitario	Costo Total
Sin SIMMEC	4	4.80	19,20
Con SIMMEC	1	4,80	4,80

Fuente: Autor

5.3 CONCLUSIONES

- Los trabajos de tesis relacionados con el desarrollo de software contribuyen en la automatización de procesos de las diferentes industrias o empresas donde se desarrollan.

- El desarrollar con el leguaje python permite a los programadores mantener un orden y una ética de programación, esto logrará aplicaciones más limpias y con el menor número de líneas de código posible ya que tiene no se repite código.

- El framework Django acelera el desarrollo de aplicaciones debido a que genera automáticamente código listo para ser usado.

- Django es un framework que va ganado adeptos y mantiene una comunidad que se mantiene constante actualización ya que python es un lenguaje muy conocido por los desarrolladores de software libre.

- La base de datos relacional MariaDB es una base de datos relacional robusta que soporta la carga del framework ya que es un fork de MySQL, una base de datos reconocida mundialmente.

- La metodología de desarrollo usada en este proyecto fue SCRUM permitió un desarrollo de calidad y con el menor número de errores, para ello fue necesario contar con un equipo de desarrollo plenamente identificado conociendo los puntos fuertes de cada uno de los integrantes.

- SCRUM es recomendable para proyectos medianos y pequeños donde se requiera una solución ágil y de resultados visibles en corto tiempo.

- El Software Libre nos ayuda a reducir notablemente los costos en el desarrollo de un proyecto, ya que el desarrollador o el equipo no tendrá que verse afectado por la compra de licencias de uso que en muchos casos es elevada.

- AL mantener los procesos automatizados se asegura que el servicio de dicha empresa va a mejorar notablemente.

5.4 RECOMENDACIONES

- La UTN como parte de su programa de pasantías pre profesionales deben incluir este tipo de industrias ya que actualmente el Estado Ecuatoriano busca fortalecer su matriz productiva.

- Se proponga la enseñanza del lenguaje python en las aulas de la carrera de sistemas, ya que es un lenguaje muy intuitivo y su comunidad provee de manuales, libros y cursos donde el estudiante pueda aprender a aprender.

- Formar una comunidad en la facultad para seguir la enseñanza de la lógica de programación, para personas de todas las edades.

- Utilizar bases de datos relacionales y no relacionales en la enseñanza supone una independencia de la formación y optar por una herramienta que soluciona el problema, y no sea la más popular.

- MariaDB es un administrador de datos poco conocida y la que se recomienda su uso tanto relacional como no relacional.

- Utilizar una buena metodología de desarrollo ayuda a garantizar el éxito en el desarrollo de un proyecto que emprendamos.

- Se sugiere aplicar en las aulas la metodología SCRUM ya que puede llegar a ser muy dinámica y aplicable a casi todos los proyectos.

- Incluir en las materias impartidas la enseñanza de Software Libre tanto en herramientas como en Sistemas Operativos derivados de UNIX, con ello ganaremos independencia y no anclarnos a un solo siguiente infinito.

5.5 Bibliografía

Alchin, M. (2013). *Pro Django, Second Edition.* LA California: Apress.

Apache Org. (29 de Agosto de 2014). *FAQ Apache Org*. Obtenido de http://www.apache.org/foundation/preFAQ.html

Bennett, J. (2009). *Practical Django Projects Second Edition.* New York, UE: Duncan Parkes.

Briggs, J. R. (2007). *Snake Wrangling for Kids, Learning to Program with Python.* San Francisco, California.: Creative Commons 3.0 España.

Debian. (2013). *Debian*. Obtenido de Caracteristicas: https://www.debian.org/intro/about

Downey, A., Elkner, J., & Meyers, C. (2002). *Aprenda a Pensar Como un Programador con Python.* Wellesley, Massachusetts: Green Tea Press.

Empresa Pública Metropolitana de Rastro. (28 de Octubre de 2014). *Empresa Pública Metropolitana de Rastro Quito*. Obtenido de http://www.epmrq.gob.ec

Fab Force. (2014). *Dbdesigner4*. Obtenido de http://www.fabforce.net/dbdesigner4/

Follet, M. (24 de Noviembre de 2014). *Soy Admin*. Obtenido de Instalar MariaDB en centOS con Mysql en producción: http://soyadmin.com/2014/11/instalar-mariadb-en-centos-con-mysql-en-produccion/

Forta, B. (2012). *MariaDB Crash Course.* Crawfordsville, Indiana: Addison Wesley.

Holovaty Adrian, K.-M. J. (2007). *El libro de Django 1.0.* Obtenido de Libros Web: http://librosweb.es/libro/django_1_0/capitulo_19/exposicion_de_mensajes_de_error.html

Hundhausen, R. (2012). *Professional Scrum Development with Microsoft® Visual Studio® 2012.* Redmond, Washington: Christian Holdener, S4 Carlisle Publishing Services.

Janssen, C. (s.f.). *ACID*. Obtenido de Atomicity Consistency Isolation Durability (ACID): http://www.techopedia.com/definition/23949/atomicity-consistency-isolation-durability-acid

MariaDB. (Septiembre de 2015). *MariaDB versus MySQL - Features*. Obtenido de https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-features/

MariaDB Org. (Septiembre de 2015). *MariaDB Knowledge Base*. Obtenido de https://mariadb.com/kb/en/

Mneptok. (20 de Julio de 2010). Obtenido de Rename Maria Contest Winner: http://blogs.gnome.org/mneptok/2010/07/20/rename-maria-contest-winner/

Palacio, J. (2007). *Flexibilidad con Scrum.* SafeCreative.

PhpMyAdmin . (2013). *PhpMyAdmin - Features*. Obtenido de https://www.phpmyadmin.net/

Q7 Enterprises, Inc. (20 de Junio de 2014). *Manifesto for Agile Software Development* . Obtenido de http://www.agilemanifesto.org/iso/es/principles.html

Rossum, G. v. (2009). *El tutorial de Python (Traducción).* Argentina: Fred L. Drake, Jr. Python Software Foundation.

Solid IT. (2014). *DB.Engine*. Obtenido de http://db-engines.com/en/system/MariaDB

Soto, J. (Agosto de 2013). *Tux Apuntes*. Obtenido de Conociendo un Poco de MariaDB: http://tuxapuntes.com/2013/08/conociendo-un-poco-de-mariadb/

Stroganov, A. (29 de 01 de 2014). *Percona*. Obtenido de Percona Server: Thread Pool Improvements for Transactional Workloads: http://www.percona.com/blog/2014/01/29/percona-server-thread-pool-improvements/

Turbogears.org. (Junio de 2010). *TurboGears*. Obtenido de About: http://www.turbogears.org

Widenius, M. ". (2010). *Storage Engines*. Obtenido de Information on Storage Engines included in MariaDB.: https://mariadb.com/kb/en/mariadb/documentation/storage-engines/

Widenius, M. (2013). *About MariaDB*. Obtenido de Mariadb.org: https://mariadb.com/kb/en/