UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN SISTEMAS COMPUTACIONALES

TEMA:

“ESTUDIO DE WEB SERVICES CASO ESTUDIO OPENSHIFT. APLICATIVO PARA LA GESTIÓN DE LA INFORMACIÓN DEL CLUB DEPORTIVO FORMATIVO ESPECIALIZADO JUVENIL CALEÑO”

AUTOR:

ESTALIN DAVID TRUJILLO CASANOVA

DIRECTOR:

ING. DIEGO JAVIER TREJO ESPAÑA

IBARRA - ECUADOR

2018

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

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional. Para lo cual pongo a disposición la siguiente información.

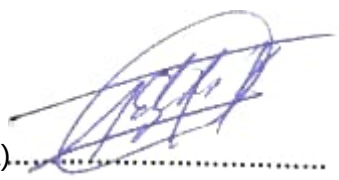
DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD	100359200-1
APELLIDOS Y NOMBRES	TRUJILLO CASANOVA ESTALIN DAVID
DIRECCIÓN	Hernan Gonzales de Saa 26-37 y Princesa Paccha
EMAIL	edtc9301@gmail.com
TELÉFONO FIJO	2652-723
TELÉFONO MOVIL	0992475099

DATOS DE LA OBRA	
TÍTULO	"ESTUDIO DE WEB SERVICES CASO ESTUDIO OPENSHIFT. APLICATIVO PARA LA GESTIÓN DE LA INFORMACIÓN DEL CLUB DEPORTIVO FORMATIVO ESPECIALIZADO JUVENIL CALEÑO"
AUTOR	TRUJILLO CASANOVA ESTALIN DAVID
FECHA	02/10/2018
SÓLO PARA PROYECTOS DE GRADO	
PROGRAMA	■ PREGRADO ☐ POSTGRADO
TÍTULO POR EL QUE OPTA	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. DIEGO TREJO

2. CONSTANCIAS

El autor (es) manifiesta (n) 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 (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

EL AUTOR:



(Firma).....

ESTALIN DAVID TRUJILLO CASANOVA

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DIRECTOR DE TESIS

Por medio del presente yo MSc. Trejo España Diego Javier, certifico que el trabajo de grado **"ESTUDIO DE WEB SERVICES CASO ESTUDIO OPENSHIFT. APLICATIVO PARA LA GESTIÓN DE LA INFORMACIÓN DEL CLUB DEPORTIVO FORMATIVO ESPECIALIZADO JUVENIL CALEÑO.",** ha sido desarrollado en su totalidad por la Sr. Trujillo Casanova Estalin David, portador de cédula de identidad Nro. 100359200-1.

Es todo cuanto puedo certificar en honor a la verdad.

MSc. Diego Trejo

DIRECTOR

CLUB DEPORTIVO ESPECIALIZADO FORMATIVO
"JUVENIL CALEÑO"
FUNDADO EL 27 DE NOVIEMBRE DE 1999
CREACION MD-CZ1-2016-0262
IBARRA - ECUADOR

Ibarra, 26 de septiembre de 2018

CERTIFICADO DE RECEPCION DE SOFTWARE

Con el presente documento el club deportivo formativo especializado juvenil caleño certifica que el Sr. **TRUJILLO CASANOVA ESTALIN DAVID,** con cedula de ciudadanía Nº **100359200-1**, estudiante de la Universidad Técnica del Norte, ha implementado el proyecto de trabajo de titulado **"ESTUDIO DE WEB SERVICES CASO ESTUDIO OPENSHIFT. APLICATIVO PARA LA GESTIÓN DE LA INFORMACIÓN DEL CLUB DEPORTIVO FORMATIVO ESPECIALIZADO JUVENIL CALEÑO"**, en nuestra distinguida institución se encuentra en perfecto estado de funcionamiento.

Nos es grato informar que cumplió con satisfacción la pruebas funcionales y requerimientos, por lo que recibió el proyecto como culminado y realizado por parte del Sr. Estudiante.

Atentamente

Lic. Pedro Carlosama
COORDINADOR

DEDICATORIA

A **DIOS** por haberme permitido llegar a este punto de mi vida, para culminar esta meta; pero lo más importante, me ha entregado una familia hermosa que ha estado siempre apoyándome.

A mi madre **GINA** por ser el pilar fundamental en el transcurso de mi vida y formar a una persona con valores. Por darme tanto amor y apoyarme incondicionalmente en mis decisiones, tanto académicamente, como en la vida cotidiana. Usted es la fuente de inspiración de todo este trabajo.

A **toda mi familia**, por todos sus consejos y palabras de aliento que me han ayudado a salir a delante.

AGRADECIMIENTO

Al Lcdo. Pedro Carlosama

Por todo su apoyo, facilidades brindadas al realizar este proyecto, sus consejos e indicaciones que me permitieron cumplir este objetivo.

Un agradecimiento muy especial al Ing. Diego Trejo

Director de trabajo de titulación, guía fundamental en la elaboración de este proyecto de grado, hasta la culminación.

A Cintya Albán

Por apoyarme en el transcurso de esta etapa de mi vida, por estar siempre dándome ánimos para no decaer y terminar esta meta.

A la UTN

Por brindarme sus conocimientos y poder conocer a tan buenos catedráticos y amigos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

El Club Deportivo Formativo Especializado Juvenil Caleño con acuerdo ministerial número MD-CZ1-2016-0262 del 15 de julio de 2016, es una entidad deportiva que tiene como fin específico: fomentar el deporte en jóvenes y niños. Razón por la cual siguiendo este lineamiento se realiza este proyecto fundamentado en **"ESTUDIO DE WEB SERVICES CASO ESTUDIO OPENSHIFT. APLICATIVO PARA LA GESTIÓN DE LA INFORMACIÓN DEL CLUB DEPORTIVO FORMATIVO ESPECIALIZADO JUVENIL CALEÑO".** El documento consta de cuatro capítulos en los que se desglosa de manera clara, paso a paso el desarrollo de las diferentes etapas hasta la culminación del proyecto.

En el Capítulo I Introducción, se detalla el problema y la razón de crear este proyecto, su análisis, justificación y objetivos que son necesarios para el desarrollo de esta Aplicación Web.

En el Capítulo II Marco Teórico, se hace una investigación de lo que es SOAP y REST, sus beneficios y cuál es el óptimo de usar.

En el Capítulo III Desarrollo, se generan los documentos de la metodología SCRUM que se implementará en este proyecto según los requerimientos del Club, utilizando Openshift como plataforma y características de herramientas utilizadas en este proyecto.

En el Capítulo IV, se plantea las conclusiones y recomendaciones de la presente aplicación.

ABTRACT

The Specialized Training Youth Club Caleño with ministerial agreement number MD-CZ1-2016-0262 of July 15, 2016, is a sports entity whose specific purpose: to promote sport in young people and children. Reason for which following this guideline is carried out this project based on **"STUDY OF WEB SERVICES CASE STUDY OPENSHIFT. APPLICATION FOR THE INFORMATION MANAGEMENT OF THE SPORTS CLUB SPORTS CLUB, SPECIALIZED JUVENIL CALEÑO ".** The document consists of four chapters that clearly break down, step by step, the development of the different stages until the completion of the project.

In Chapter I Introduction, the problem and the reason for creating this project, its analysis, justification and objectives that are necessary for the development of this Web Application are detailed.

In Chapter II Theoretical Framework, an investigation is made of what SOAP and REST are, their benefits and what is the optimum to use.

In Chapter III Development, the documents of the SCRUM methodology that will be implemented in this project are generated according to the requirements of the Club, using Openshift as a platform and features of the tools used in this project.

In Chapter IV, the conclusions and recommendations of the present application are presented.

INTRODUCCIÓN

Este capítulo recopila la información acerca de la definición del problema, los objetivos planteados para este proyecto, la delimitación del problema y la justificación del tema.

1.1 **Definición del problema**.

“Los servicios web surgieron ante una necesidad de estandarizar la comunicación entre distintas plataformas y lenguajes de programación. Anteriormente se habían realizado intentos de crear estándares, pero fracasaron o no tuvieron el suficiente éxito, algunos de ellos son DCOM y CORBA, por ser dependientes de la implementación del vendedor DCOM Microsoft, y CORBA de ORB. Otro gran problema es que se hacía uso de RPC para realizar la comunicación entre diferentes nodos.” (Brea, s.f.)

“DCOM se lanzó en 1996 y sólo está disponible para plataformas de 32 bits. El equipo de Visual Basic de Microsoft siempre pensó en utilizar la automatización para la comunicación de sus componentes. La falta de una versión distribuida limitó en gran medida el uso de estas características en los entornos empresariales; por ello, el equipo que desarrollaba Visual Basic 4.0 Edición empresarial decidió investigar la creación de su propio conjunto de componentes remotos para las partes de automatización de OLE y COM. Claramente, un objetivo fundamental era asegurarse de que el resultado fuera compatible con DCOM y pudiera ser reemplazado por esta tecnología cuando estuviera disponible. Así, implementaron la Automatización remota (RA, Remote Automation) para plataformas Windows de 16 bits y 32 bits.” (Microsoft, s.f.)

Los Web Services aparecieron para poder solucionar la comunicación entre las diferentes plataformas.

En los años recientes, ha habido una explosión en la rama de "servicios hospedados en la nube". Estos son ofrecidos en demanda directamente desde el proveedor en lugar de ser hospedados en la infraestructura del cliente. Los servicios relacionados con la nube están entre los más estratégicos para los departamentos de tecnología durante estos años.

En los últimos años apareció una arquitectura de software conocida como REST (Representational State Transfer). Este nuevo servicio es una opción más que los programadores pueden hacer uso.

1.2 Definición de objetivos.

1.2.1 Objetivo General.

- Desarrollar un aplicativo web para el club deportivo formativo especializado Juvenil Caleño, usando web services mediante la implementación de la metodología SCRUM.

1.2.2 Objetivos Específicos.

- Realizar un estudio de los requerimientos necesarios para que la aplicación funcione correctamente de acuerdo con las necesidades del club.
- Diseñar la base de datos en MySQL.
- Establecer un método de comparación entre los webs services más populares y determinar cuál es la mejor alternativa a ser aplicada.
- Desarrollar el software aplicando la metodología SCRUM.

1.3 Delimitación del problema.

Se estudia la arquitectura que manejan y en que herramientas de programación pueden ser utilizadas, además se observó los tipos de web service que existen en el mercado, haciendo énfasis en las más usadas que son los web services SOAP y los web services REST.

Para ver cómo funcionan los webs services se realizó un aplicativo el cual tiene la arquitectura que se muestra en Ilustración 1.

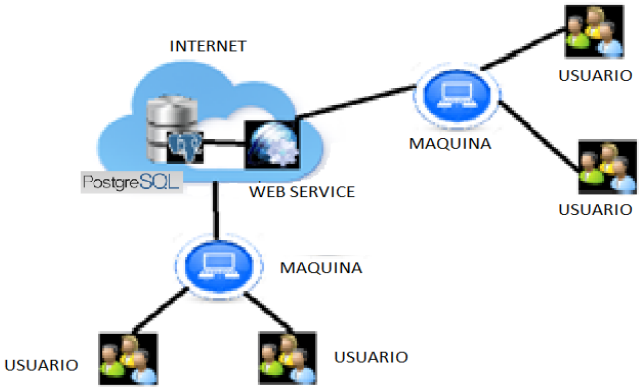


Ilustración 1 Arquitectura del sistema

Fuente: Propia.

Para la realización de este aplicativo pondremos en funcionamiento únicamente dos módulos los cuales son:

- **Módulo Campeonato**

En este módulo se detalla todo lo referente a lo que es el campeonato de futbol del club deportivo formativo especializado Juvenil Caleño, este tendrá el control de la tabla de posiciones de los diferentes clubes que participen en dicho campeonato, los partidos realizados y calendario de juegos.

- **Módulo Ficha Jugador**

En este módulo se encontrarán todos los datos referentes a los niños que son participes del club deportivo formativo especializado Juvenil Caleño, en el cual se encuentra una ficha con los datos de cada uno de ellos incluido los pagos realizados.

1.4 Justificación del tema.

1.4.1 Justificación Tecnológica.

Para el desarrollo del trabajo de titulación se utilizó la metodología SCRUM, la cual permitió una mejor implementación del proyecto. En la ilustración 2 se detalla cómo está estructurada la metodología.

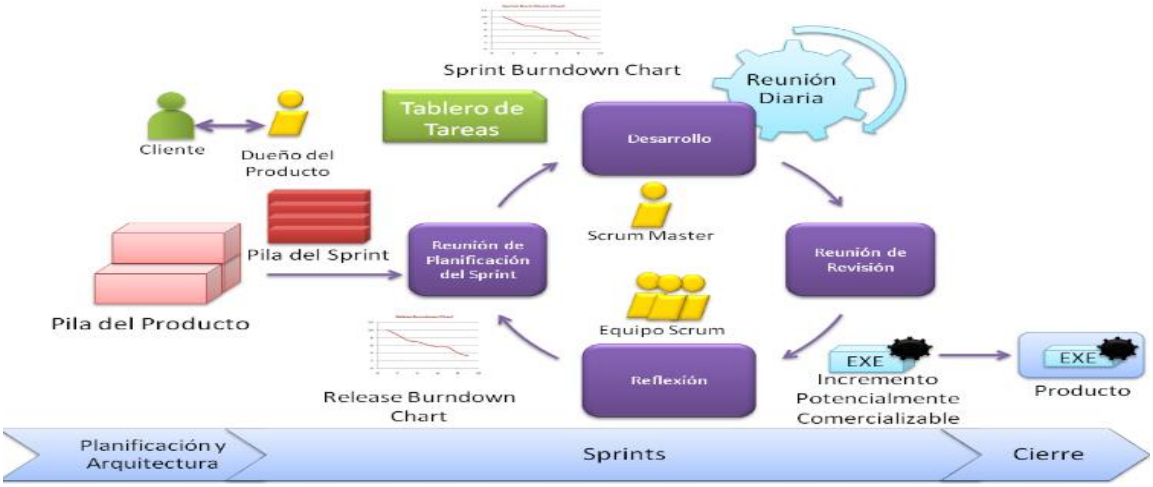


Ilustración 2 Metodología SCRUM

Fuente: http://escritura.proyectolatin.org/gestion-de-proyectos-de-software/ejemplos-de-procesos.

El estudio de los webs services contribuye un aporte a las personas que lean este trabajo de titulación, en el cual estará detallado el funcionamiento de estos, es decir: la estructura, los modelos y arquitecturas.

Se desarrolló un aplicativo para demostrar el funcionamiento de estos servicios, se utilizó la herramienta opensift para subir los datos al servidor, esta es una herramienta gratuita, pero brinda un espacio adecuado para la realización del aplicativo.

1.4.2 Ciclo de vida de la metodología SCRUM.

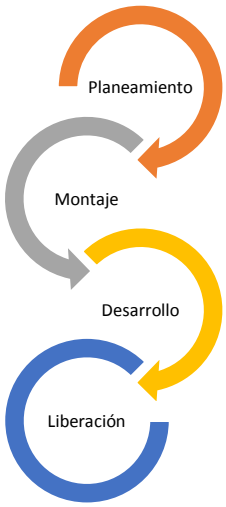


Ilustración 3 Ciclo de vida de SCRUM

Fuente: Propia

El ciclo de vida es un proyecto que puede empezar con cualquier actividad y se puede pasar de una actividad a otra en cualquier momento maximizando la flexibilidad y la productividad del equipo como se muestra en la ilustración 3. “En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas si, así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento del producto final” (Heredia, 2011)

1.4.3 Elementos de la metodología SCRUM

Pila del producto

- Lista de requisitos de usuario.

Pila del sprint

- Lista de los trabajos que debe realizar el equipo durante la iteracion.

Incremento

- Resultado de cada iteración.

Ilustración 4 Elementos de la metodología SCRUM

Fuente: propia

1.4.4 Roles de la metodología SCRUM

(Sutherland, 2016) define los roles de SCRUM son:

- Jefe de proyecto o SRUM Master: Vigila que todas las reglas se cumplan y se encarga de guiar el desarrollo.
- Propietario del proyecto o Product Owner: Fija tareas y prioridades.
- Equipo de desarrolladores o Team Members: Su función es entregar el producto.
- Los extremos interesados o Stakeholders: Que pueden asistir a las reuniones, pero pueden hablar.

1.4.5 Impacto económico

El impacto económico que presenta este servicio es que al implantar en una organización sus costos van a disminuir, debido a que ya no necesitaran una infraestructura muy grande para esto, solo necesitara que sus usuarios consuman los servicios en la nube.

1.4.6 Impacto social y/o ambiental

El mayor impacto ambiental es la disminución de energía y gastos en papel, gastos en energía por que la empresas ya no necesitaran tener sus mega servidores en cada empresa, con los web services estos estarán en un solo servidor al que se accede a través de internet, mientras que al implantar esto en el club deportivo formativo especializado Juvenil Caleño los costos en papel disminuirán drásticamente en vista que cambiaran su método de llevar lo registros, no usaran papel en exceso y todos los datos estarán en la nube.

CAPÍTULO 2

Este capítulo inicia explicando la definición de los servicios web, continuando con la arquitectura SOA y además se explica y compara los servicios SOAP y REST.

2.1 Servicios Web

Existen varias definiciones acerca de los servicios web. IBM los define como: “Tanto un servicio web como los servicios web son auto contenidos, aplicaciones modulares que pueden ser descritas, publicadas, localizadas, e invocadas a través de una red, en general, la World Wide Web." (O'Neill, 2003)

“Otra definición de servicio web explica: “Un servicio web se describe así mismo y a las aplicaciones empresariales modulares que exponen la lógica de negocio como servicios sobre Internet a través de interfaces programables y el uso de protocolos de Internet con el propósito de proporcionar formas de buscar, suscribirse e invocar esos servicios.” (Ramesh Nagappan, 2003)

Algunos ejemplos del uso de los servicios web son:

- Verificar la validez de las tarjetas de crédito ya sea en la misma entidad bancaria o en otra.
- Consultas enviando mensajes de texto desde teléfonos celulares.
- Servicios de búsqueda en internet utilizando Google.

¿Para qué sirven?

“Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.” (w3c_es, s.f.)

La ilustración 5 muestra cómo interactúa un conjunto de Servicios Web:

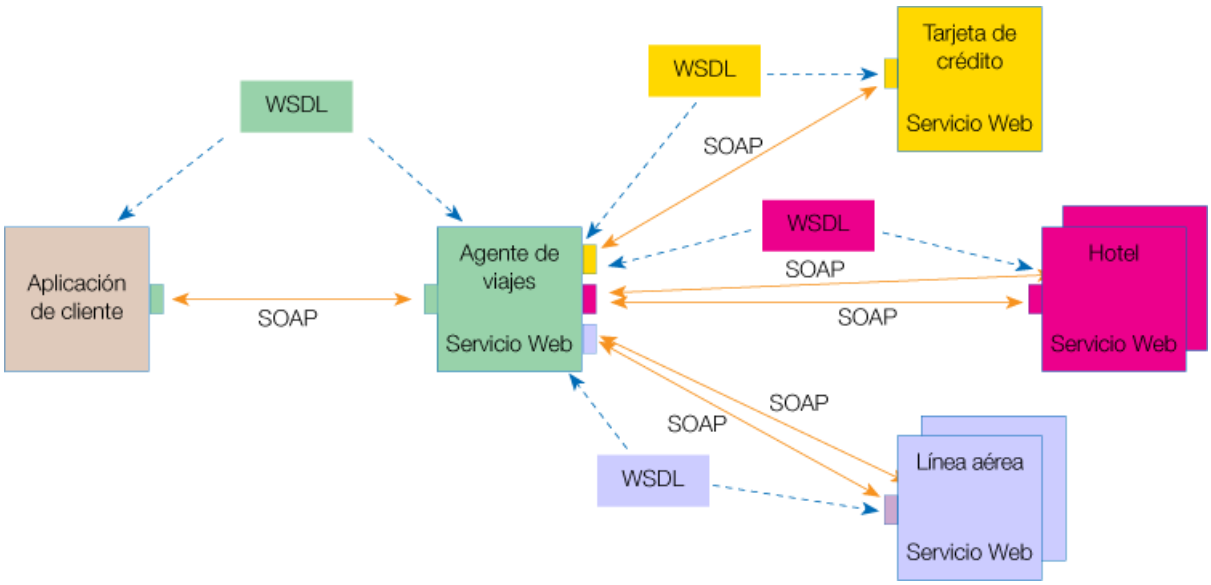


Ilustración 5 Funcionamiento de los servicios web

Fuente: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

"Según la ilustracion 5, un usuario (que juega el papel de cliente dentro de los Servicios Web), a través de una aplicación, solicita información sobre un viaje que desea realizar haciendo una petición a una agencia de viajes que ofrece sus servicios a través de Internet. La agencia de viajes ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia de viajes solicita a su vez información a otros recursos (otros Servicios Web) en relación con el hotel y la compañía aérea. La agencia de viajes obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada sobre el hotel y la línea aérea. Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago." (w3c_es, s.f.)

“En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (Protocolo Simple de Acceso a Objetos). Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP , SMTP . SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo) los cuales se muestran en la ilustracion 6.” (w3c_es, s.f.)

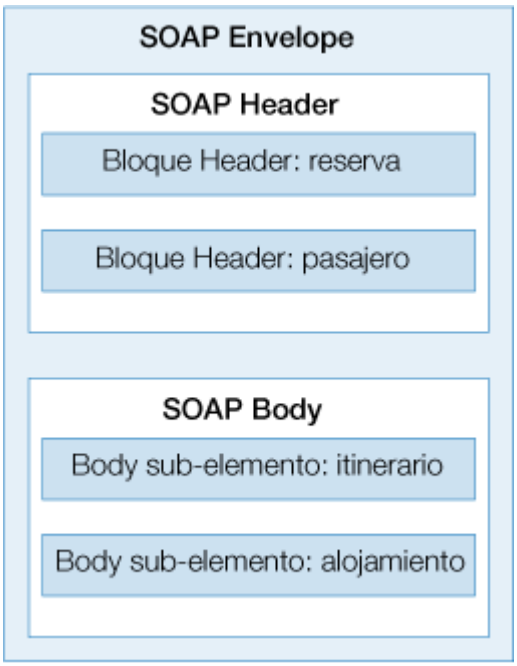


Ilustración 6 Estructura de los mensajes

Fuente: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

“Para optimizar el rendimiento de las aplicaciones basadas en Servicios Web, se han desarrollado tecnologías complementarias a SOAP, que agilizan el envío de los mensajes (MTOM) y los recursos que se transmiten en esos mensajes (SOAP-RRSHB).

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL

representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes.” (w3c_es, s.f.)

“Durante la evolución de las necesidades de las aplicaciones basadas en Servicios Web de las grandes organizaciones, se han desarrollado mecanismos que permiten enriquecer las descripciones de las operaciones que realizan sus servicios mediante anotaciones semánticas y con directivas que definen el comportamiento. Esto permitiría encontrar los Servicios Web que mejor se adapten a los objetivos deseados. Además, ante la complejidad de los procesos de las grandes aplicaciones empresariales, existe una tecnología que permite una definición de estos procesos mediante la composición de varios Servicios Web individuales, lo que se conoce como coreografía.” (w3c_es, s.f.)

Ejemplo

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-
policy"xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:ws
am="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schem
as.xmlsoap.org/wsdl/soap/"xmlns:tns="http://service/" xmlns:xsd="http://www
.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetName
space="http://service/" name="service">
<types>
<xsd:schema>
<xsd:import namespace="http://service/" schemaLocation="http://edtrujillo:8
080/PagosWeb/service?xsd=1"/>
</xsd:schema>
</types>
<message name="crearFactura">
<part name="parameters" element="tns:crearFactura"/>
</message>
<message name="crearFacturaResponse">
<part name="parameters" element="tns:crearFacturaResponse"/>
</message>
<message name="objetofact">
<part name="parameters" element="tns:objetofact"/>
</message>
<message name="objetofactResponse">
<part name="parameters" element="tns:objetofactResponse"/>
</message>
<portType name="service">
<operation name="crearFactura">
<input wsam:Action="http://service/service/crearFacturaRequest" message="tn
s:crearFactura"/>
<output wsam:Action="http://service/service/crearFacturaResponse" message="
tns:crearFacturaResponse"/>
</operation>
<operation name="objetofact">
<input wsam:Action="http://service/service/objetofactRequest" message="tns:
objetofact"/>
<output wsam:Action="http://service/service/objetofactResponse" message="tn
s:objetofactResponse"/>
</operation>
</portType>
```

```
<binding name="servicePortBinding" type="tns:service">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="docum
ent"/>
<operation name="crearFactura">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="objetofact">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
<service name="service">
<port name="servicePort" binding="tns:servicePortBinding">
<soap:address location="http://edtrujillo:8080/PagosWeb/service"/>
</port>
</service>
</definitions>
```

2.2 Arquitectura SOA

2.2.1 Qué es la arquitectura orientada a servicios SOA

El acrónimo S.O.A. proviene de las palabras inglesas Service Oriented Architecture.

SOA es un concepto de arquitectura de software que da forma a los procedimientos para crear y usar diversos procesos, herramientas y modelos, reunidos en forma de servicios independientes y reutilizables con interfaces invocables bien definidas (independientes del hardware, sistema operativo y del lenguaje de programación), para dar soporte TI a los requisitos y necesidades de un negocio.

“Una SOA es una forma de mirar al mundo. Cuando adopta una visión orientada a servicios, todo cobra forma de servicio. Los servicios son los ladrillos con los que se construye una SOA. Son un medio para acceder a las capacidades que se repiten en un negocio.” (Matsumura, Brauel, & Shah, 2009)

SOA permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e

invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

Entonces decimos que el concepto de SOA puede confundirse en varios casos con una tecnología o bien por un producto de software, y estos dos aspectos son lo más alejado de la realidad. En el internet existen muchas definiciones de SOA y aunque en su gran mayoría estas son acertadas, unas explican de mejor manera que otras. hay que comprender que SOA es una arquitectura que quiere alinear las TI con el propio negocio de la organización. Esto es por lo que se sugiere la creación de: servicios y funcionalidades que sean reutilizables.

Estos servicios deben ser flexibles, seguros y, sobre todo, con una arquitectura basada en estándares.

Una encuesta realizada a 12 directores de tic de empresas españolas dio como resultado que alguno de los principales objetivos para implementar la arquitectura en sus empresas se muestra en la ilustración 7.

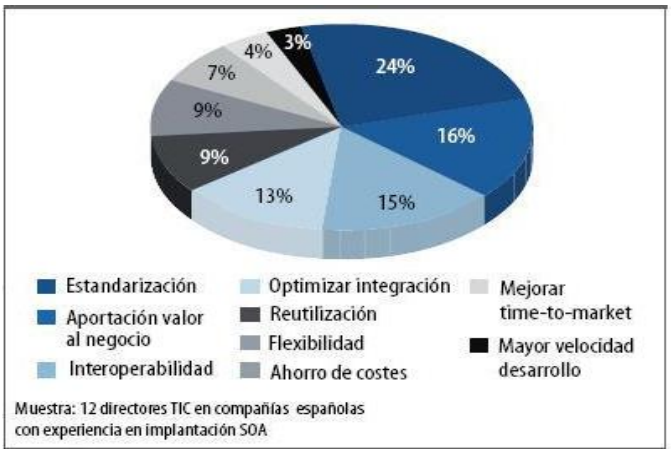


Ilustración 7 Principales objetivos para implantar SOA

Fuente: https://emergingtechuva.files.wordpress.com/2013/11/principales_objetivos.jpg

2.2.2 Actores en el SOA

Los roles que integran al SOA son los siguientes y pueden observarse en la ilustración 8:

- Consumidor de servicios: Es una aplicación, un módulo de software u otro servicio que demanda la funcionalidad proporcionada por un servicio, y la ejecuta en una interfaz definida.
- Proveedor de servicios: Es una entidad accesible a través de la red que acepta y ejecuta consultas de consumidores y publica sus servicios y su contrato de interfaces en el registro de servicios para que el consumidor pueda descubrir y acceder al servicio.
- Registro de servicios: Es un repositorio de servicios disponibles y permitiendo visualizar las interfaces de los proveedores de servicios a los consumidores interesados.

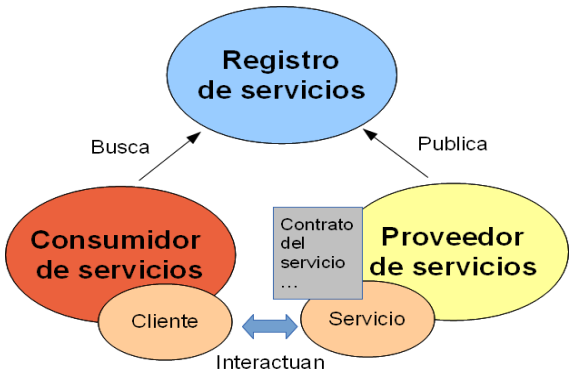


Ilustración 8 Funcionamiento roles SOA

Fuente: https://emergingtechuva.files.com/2013/11/roles_soa.png

2.2.3 Elementos de SOA

Los elementos que componen el SOA están indicados en la tabla 1

Tabla 1 Descripción elementos

Transporte	Es el mecanismo utilizado para llevar las demandas de un servicio desde un consumidor hacia un proveedor del servicio, y las respuestas desde el proveedor hacia el consumidor.
Protocolo de comunicación de servicios	Es el conjunto de reglas y estándares que controlan la secuencia de mensajes que

	ocurren durante una comunicación entre entidades que forman una red.
Descripción de servicio	Es el esquema (schema) fijado para describir la información y elementos que componen a un servicio.
Servicio	Describe un servicio actual que está disponible para ser utilizado.
Proceso de Negocio	Es una colección de servicios, invocados en una secuencia particular con un conjunto específico de reglas, para satisfacer un requisito de negocio.
Registro de servicios	Es un repositorio de descripciones de servicios y datos que pueden utilizar los proveedores para publicar sus servicios, así como los consumidores poder descubrir o hallar servicios disponibles.
Política	Es un conjunto de condiciones o reglas bajo las cuales un proveedor hace el servicio disponible para los consumidores.
Seguridad	Son las normas y reglas de autenticación, identificación y control de acceso a los consumidores y proveedores de servicios.
Transacciones	Es una interacción con una estructura de datos compleja, compuesta por varios procesos que se han de aplicar para que el servicio sea consistente.
Administración	Es el conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.

Fuente: Propia.

Las funciones que proporciona SOA y la calidad del servicio, presentando un modelo de construcción de sistemas distribuidos en el que la funcionalidad demandada será entregada a la aplicación a través de servicios como se muestra en la ilustración 9.

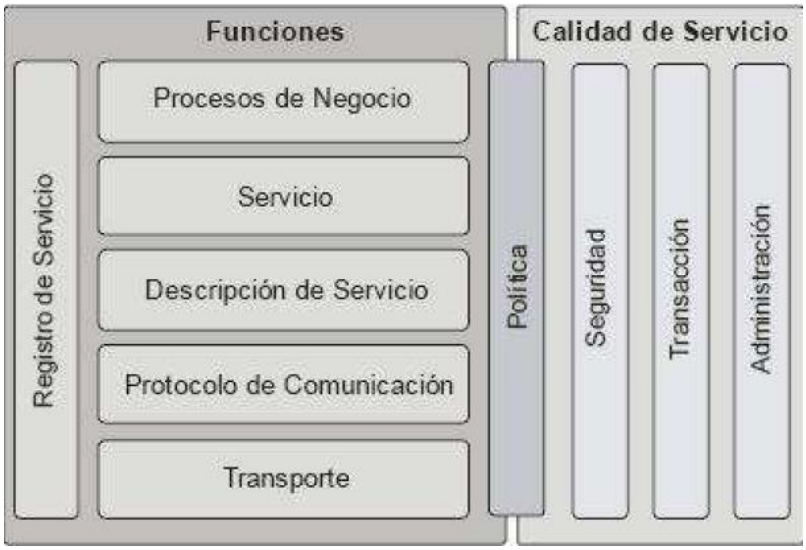


Ilustración 9 Elementos de SOA

Fuente: https://oposicionestic.blogspot.com/2012/08/arquitectura-soa-orientada-servicios.html

2.2.4 SOA como arquitectura de software

Estándares que se aplican a SOA:

Los relacionados con los servicios web: Simple Object Access Protocol – SOAP, Web Services Description Language – WSDL, en la ilustración 10 se muestra el funcionamiento de los servicios web.

Ilustración 10 Servicio Web

Fuente: http://www.palentino.es/servicios-web-utilidades-estandares-y-beneficios/

El relacionado con la ejecución de los procesos de negocio: Business Process Execution Language (BPEL) se muestra un ejemplo de cómo funciona en la ilustración 11.

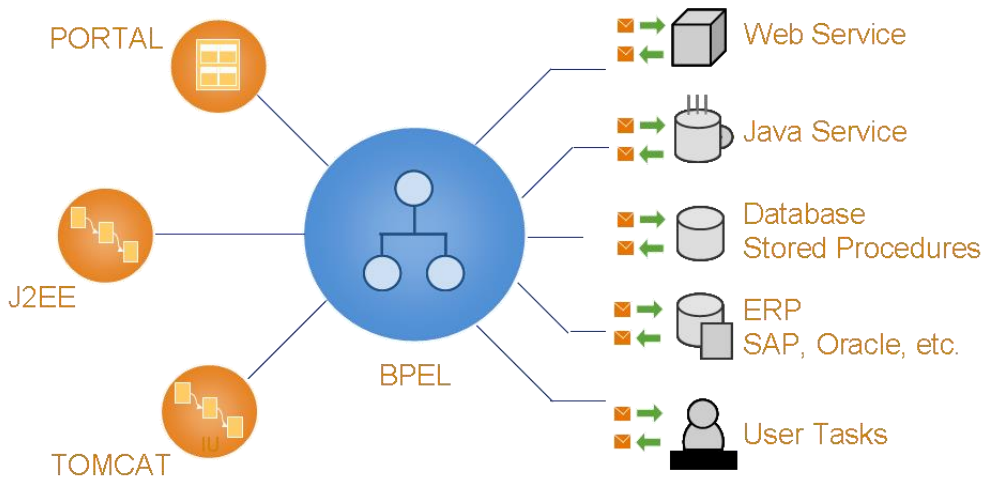


Ilustración 11 BPEL

Fuente: https://technology.amis.nl

Los componentes tecnológicos:

- Bus de Servicios (ESB), donde se despliegan y ejecutan los servicios.
- Registro de servicios, basado en el protocolo UDDI (Universal Description, Discovery and Integration).
- Business Process Management – BPM: componente para la orquestación de servicios en procesos de negocio.
- Business Activity Monitoring – BAM: componente para la visualización y el seguimiento de las actividades del negocio.

Protocolos, formatos.

- HTTP: HiperText Transfer Protocol
- SOAP.
- REST.
- URL: (Uniform Resource Locator) mecanismo de identificación de recursos
- XML/HTML/PNG/… distintos formatos de representación de recursos
- Tipos MIME como text/xml, text/html, image/png, etcétera.

Ventajas e inconvenientes:

Algunas de estas se muestran en la tabla 2, y vemos que existen más ventajas que inconvenientes para esta arquitectura.

Tabla 2 Ventajas y Desventajas

Ventajas	Inconvenientes
Respuesta rápida a nuevas necesidades de negocio	SOA depende de la implementación de estándares. Sin estándares, la comunicación entre aplicaciones requiere de mucho tiempo y código.
Reducción del costo de desarrollo de IT	SOA no es para: aplicaciones con alto nivel de transferencia de datos, aplicaciones que no requieren de implementación del tipo request/response y para aplicaciones que tienen un corto periodo de vida.
Capacidad de integrar a clientes y socios	Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio.
Capacidad de generar nuevos modelos de negocios	Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio.
Alinear objetivos de IT a objetivos de negocio	En la medida en que un servicio de negocio vaya siendo incorporado en la definición de los procesos de negocio, dicho servicio aumentara su nivel de criticidad. Con lo cual cada que se requiera efectuar una actualización en dicho servicio (por ejemplo, un cambio en el código, una interfaz nueva, etc.), deberá evaluarse previamente el impacto y tener mucho cuidado con su

	implementación. Sin embargo, parte de la problemática anterior, puede ser solventada en virtud de un buen diseño del servicio.
Apertura a nuevos mercados, canales y valor de sistemas existentes	
Permite la reutilización de componentes prefabricados de servicio para implementaciones lógicas	
Integra sistemas separados de distintas plataformas	
Eliminar duplicidad de sistemas	
Automatiza los procesos de negocios	
Mejorar la visibilidad de procesos de negocio	
Respuesta rápida a nuevas necesidades de negocio	

2.2.5 Qué es un ESB

Otro concepto muy asociado a SOA y que también conviene aclarar es el de ESB (Enterprise Service Bus). A diferencia de SOA, ESB sí es una tecnología o producto software.

Puede definirse un ESB como la Infraestructura que sirve como el backbone de las Arquitecturas Orientadas a Servicios (SOA). Un ESB permite a una empresa, conectar, mediar, y controlar la interacción entre diversas aplicaciones y servicios a lo largo de entornos altamente distribuidos y heterogéneos.

SOA de por sí puede aportar una gran potencia y flexibilidad a una organización sin necesidad de utilizar un ESB, pero sin un ESB es muy complicada la gestión y mantenimiento de los procesos SOA y el escalado de la arquitectura en caso de ser necesario. Normalmente un ESB debe proporcionar a una organización por un lado un sistema de mensajería robusto de alta disponibilidad y altamente escalable que permita garantizar la comunicación entre los distintos servicios y aplicaciones de la organización independientemente de su localización geográfica, y por otro lado un mecanismo que permita fácilmente la definición de nuevos procesos o su posterior modificación orquestando los servicios existentes en la organización.

Además, debe proporcionar funcionalidades avanzadas como la de enrutamiento de datos dinámico basado en el contenido de estos, en la ilustración 12 se detalla cómo es el funcionamiento del ESB.

Ilustración 12 ESB

Fuente: http://ains.com/enterprise-service-bus-esb/

Etapas en la evolución a SOA

Una vez que una organización decide implementar SOA, es muy típico pasar por todas estas fases en el camino de evolución hacia la madurez SOA:

- **Creación de los Servicios dentro de la organización.** Estos Servicios incluyen funcionalidades reutilizables, accesibles de manera estándar y que pueden formar parte de procesos de negocio.
- **Orquestación de los Servicios** previamente creados, mediante la definición de Procesos que utilizarán dichos servicios disponibles en la organización.
- **Construcción de nuevos servicios** según son necesarios para la orquestación de los nuevos Procesos y reutilización de los Servicios ya existentes en distintos Procesos.
- **Enterprise Service Bus (ESB).** Una vez que el número de procesos orquestados es importante, la organización descubre que es difícil controlarlos y es difícil escalar procesos existentes, y en ese momento es cuando las organizaciones realmente descubren que es muy complicado gestionar y escalar una Arquitectura SOA si se carece de un ESB.
- **Monitorización de los Procesos en SOA**. Una vez se definen los primeros procesos SOA, surge la necesidad de tener visibilidad en quién está utilizando estos servicios, qué tiempos de respuesta se están ofreciendo, etc. Este punto

será tenido en cuenta en distintos momentos en el camino hacia la madurez SOA

- **Integración Semántica de Datos.** Es la problemática conocida como la del "Año SOA + 1", ya que es una problemática que habitualmente no se tiene en cuenta a la hora de diseñar SOA y que se descubre cuando se empieza a utilizar. Toda la gestión de las transformaciones de datos, en muchas ocasiones se solventan a nivel del Orquestador o del BPM o incluso a nivel del ESB, implementando en muchas ocasiones complicadas transformaciones XSLT incluso para formatos no XML. Esto genera problemas a la hora de cambiar la estructura de datos de una aplicación, ya que es muy complejo identificar a qué puede afectar dicho cambio, todos los cambios a realizar en otras aplicaciones o transformaciones de datos para gestionar dicho cambio y dónde realizarlos.
- **Monitorización de negocio (BAM) de los procesos en ejecución en SOA.** Además de conocer los procesos que están en ejecución, surge la necesidad de tener información en tiempo real del negocio, es decir, cuántos pedidos estamos recibiendo en este momento, cuántos pagos no se han realizado y caducan en dos días, si hay retraso en la entrega de algún producto, etc. Una de las principales garantías de éxito para una organización a la hora de abordar una evolución hacia SOA es apoyarse en proveedores que realmente tengan experiencia en este tipo de proyectos, y que sean capaces de orientarla para evitar caer en errores ya vividos en otros proyectos SOA. Y cuando hablo de experiencia en proyectos SOA me refiero a proyectos SOA que realmente estén en producción, y no pilotos o puesta en producción de un cierto número de Servicios Web aislados.

2.3 Que es SOAP

SOAP es un protocolo simple y ligero basado en XML para intercambiar información estructurada y de tipos en la Web. El objetivo del diseño conjunto de SOAP es mantenerlo tan simple como sea posible y proporcionar un mínimo de funcionalidad. El protocolo define un marco de mensajería que no contiene ninguna aplicación o semántica de transporte. Como resultado, el protocolo es modular y muy extensible.

Viajando a través de los protocolos de transporte estándar, SOAP puede utilizar la arquitectura abierta existente de Internet y conseguir fácilmente la aceptación de cualquier sistema arbitrario capaz de admitir los estándares de Internet más básicos. Puede ver la infraestructura necesaria para admitir un servicio web de XML conforme con SOAP como bastante simplista, aunque eficaz, ya que agrega relativamente poco a la infraestructura

existente de Internet y aun así facilita el acceso universal a los servicios generados con SOAP.

La especificación del protocolo SOAP está compuesta de cuatro partes principales. La primera parte define un sobre extensible obligatorio para encapsular los datos. El sobre SOAP define un mensaje SOAP y es la unidad básica de intercambio entre los procesadores de mensajes SOAP. Ésta forma la única parte obligatoria de la especificación.

La segunda parte de la especificación del protocolo SOAP define reglas opcionales de codificación de datos para representar tipos de datos y gráficas dirigidas definidos por la aplicación, y un modelo uniforme para serializar los modelos de datos no sintácticos.

La tercera parte define un modelo de intercambio de mensajes de estilo RPC (solicitud/respuesta). Cada mensaje SOAP es una transmisión unidireccional. Aunque las raíces de SOAP están en RPC, no se limita a ser un mecanismo de solicitud/respuesta. Los servicios web de XML combinan a menudo los mensajes SOAP para implementar tales modelos, pero SOAP no asigna un modelo de intercambio de mensajes y esta parte de la especificación también es opcional.

La parte cuarta de la especificación define un enlace entre SOAP y HTTP. Sin embargo, esta parte también es opcional. Puede utilizar SOAP en combinación con cualquier protocolo de transporte o mecanismo que puede transportar el sobre SOAP, incluidos SMTP, FTP o incluso un disquete.

“Protocolo Simple de Acceso a Objetos (SOAP). Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winerenen 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los Servicios Web.” (ecured, s.f.)

En el núcleo de los servicios Web se encuentra el protocolo simple de acceso a datos SOAP, que proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido gran atención debido a que facilita una comunicación del estilo RPC entre un cliente y un servidor remoto. Pero existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC de Sum, DCE de Microsoft, RMI de Java y ORPC de CORBA.

¿Por qué se presta tanta atención a SOAP?

Una de las razones principales es que SOAP ha recibido un increíble apoyo por parte de la industria. SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por

todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores Compañías que soportan SOAP son Microsoft, IBM, SUN, Microsystems, SAP y Arriba.

“Es importante para las aplicaciones de Internet para poder comunicarse a través de Internet.

La mejor manera de comunicarse entre aplicaciones es a través de HTTP, HTTP porque es soportado por todos los navegadores de Internet y servidores. SOAP fue creado para lograr esto.

SOAP proporciona una manera de comunicarse entre aplicaciones que se ejecutan en sistemas operativos diferentes, con diferentes tecnologías y lenguajes de programación.

SOAP Building Blocks

Un mensaje SOAP es un documento XML que contiene ordinaria los siguientes elementos:

- Un elemento de sobres que identifica el documento XML como un mensaje SOAP.
- Un elemento de cabecera que contiene información de cabecera.
- Un elemento de cuerpo que contiene la información de llamada y respuesta.
- Un elemento de falta que contiene errores y la información de estado.” (W3C, s.f.)

2.4 Que es REST

“REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP.

En realidad, REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP. Estos dos significados pueden chocar o incluso solaparse. Es posible diseñar un sistema software de gran tamaño de acuerdo con la arquitectura propuesta por Fielding sin utilizar HTTP o sin interactuar con la Web. Así como también es posible diseñar una simple interfaz XML+HTTP que no sigue los principios REST, y en cambio seguir un modelo RPC.

Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aunque REST no es un estándar, está basado en estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/…
- Tipos MIME: text/xml, text/html,” (Marset, 2006)

2.4.1 Cuál es la motivación de REST

“La motivación de REST es la de capturar las características de la Web que la han hecho tan exitosa.

Si pensamos un poco en este éxito, nos daremos cuenta de que la Web ha sido la única aplicación distribuida que ha conseguido ser escalable al tamaño de Internet. El éxito lo debe al uso de formatos de mensaje extensibles y estándares, pero además cabe destacar que posee un esquema de direccionamiento global (estándar y extensible a su vez).

En particular, el concepto central de la Web es un espacio de URIs unificado. Las URIs permiten la densa red de enlaces que permiten a la Web que sea tan utilizada. Por tanto, ellos consiguen tejer una mega-aplicación.

Las URIs identifican recursos, los cuales son objetos conceptuales. La representación de tales objetos se distribuye por medio de mensajes a través de la Web. Este sistema es extremadamente desacoplado.

Estas características son las que han motivado para ser utilizadas como guía para la evolución de la Web." (Marset, 2006)

2.5 Principios de REST

"El estilo de arquitectura subyacente a la Web es el modelo REST. Los objetivos de este estilo de arquitectura se listan a continuación:

- Escalabilidad de la interacción con los componentes. La Web ha crecido exponencialmente sin degradar su rendimiento. Una prueba de ellos es la variedad de clientes que pueden acceder a través de la Web: estaciones de trabajo, sistemas industriales, dispositivos móviles.
- Generalidad de interfaces. Gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. Esto no es del todo cierto para otras alternativas, como SOAP para los Servicios Web.
- Puesta en funcionamiento independiente. Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestas en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URIs, a través de la habilidad para crear nuevos métodos y tipos de contenido.
- Compatibilidad con componentes intermedios. Los más populares intermediaros son varios tipos de proxys para Web. Algunos de ellos, las caches, se utilizan para mejorar el rendimiento. Otros permiten reforzar las políticas de seguridad: firewalls. Y, por último, otro tipo importante de intermediarios, gateway, permiten encapsular sistemas no propiamente Web. Por tanto, la compatibilidad con intermediarios nos permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas." (Marset, 2006)

“REST logra satisfacer estos objetivos aplicando cuatro restricciones:

- Identificación de recursos y manipulación de ellos a través de representaciones. Esto se consigue mediante el uso de URIs. HTTP es un protocolo centrado en URIs. Los recursos son los objetos lógicos a los que se le envían mensajes. Los recursos no pueden ser directamente accedidos o modificados. Más bien se trabaja con representaciones de ellos. Cuando se utiliza un método PUT para enviar información, se coge como una representación de lo que nos gustaría que el estado del recurso fuera. Internamente el estado del recurso puede ser cualquier cosa desde una base de datos relacional a un fichero de texto.
- Mensajes autodescriptivos. REST dicta que los mensajes HTTP deberían ser tan descriptivos como sea posible. Esto hace posible que los intermediarios interpreten los mensajes y ejecuten servicios en nombre del usuario. Uno de los modos que HTTP logra esto es por medio del uso de varios métodos estándares, muchos encabezamientos y un mecanismo de direccionamiento. Por ejemplo, las cachés Web saben que por defecto el comando GET es cacheable (ya que es side-effect-free) en cambio POST no lo es. Además, saben cómo consultar las cabeceras para controlar la caducidad de la información. HTTP es un protocolo sin estado y cuando se utiliza adecuadamente, es posible es posible interpretar cada mensaje sin ningún conocimiento de los mensajes precedentes. Por ejemplo, en vez de logearse del modo que lo hace el protocolo FTP, HTTP envía esta información en cada mensaje.
- Hipermedia como un mecanismo del estado de la aplicación. El estado actual de una aplicación Web debería ser capturada en uno o más documentos de hipertexto, residiendo tanto en el cliente como en el servidor. El servidor conoce sobre el estado de sus recursos, aunque no intenta seguirles la pista a las sesiones individuales de los clientes. Esta es la misión del navegador, él sabe cómo navegar de recurso a recurso, recogiendo información que el necesita o cambiar el estado que el necesita cambiar.” (Marset, 2006)

2.6 Ejemplo de diseño basado en REST

“De nuevo tomaremos como ejemplo a la Web. La Web evidentemente es un ejemplo clave de diseño basado en REST, ya que muchos principios son la base de REST. Posteriormente mostraremos un posible ejemplo real aplicado a Servicios Web.

La Web consiste en el protocolo HTTP, de tipos de contenido, incluyendo HTML y otras tecnologías tales como el Domain Name System (DNS).

Por otra parte, HTML puede incluir javascript y applets, los cuales dan soporte al codeon-demand, y además tiene implícitamente soporte a los vínculos. HTTP posee un interfaz uniforme para acceso a los recursos, el cual consiste de URIs, métodos, códigos de estado, cabeceras y un contenido guiado por tipos MIME.

Los métodos HTTP más importantes son PUT, GET, POST y DELETE. Ellos suelen ser comparados con las operaciones asociadas a la tecnología de base de datos, operaciones CRUD: CREATE, READ, UPDATE, DELETE. Otras analogías pueden también ser hechas como con el concepto de copiar-y-pegar (Copy&Paste). Todas las analogías se representan en la siguiente tabla:” (Marset, 2006)

Tabla 3 Métodos de HTTP

Acción	HTTP	SQL	Copy & Paste	Unix Shell
Create	PUT	Insert	Pegar	>
Read	GET	Select	Copiar	<
Update	POST	Update	Pegar después	>>
Delete	DELETE	Delete	Cortar	Del/rm

Fuente:

“Las acciones (verbos) CRUD se diseñaron para operar con datos atómicos dentro del contexto de una transacción con la base de datos. REST se diseña alrededor de transferencias atómicas de un estado más complejo, tal que puede ser visto como la transferencia de un documento estructurado de una aplicación a otra.

El protocolo HTTP separa las nociones de un servidor y un navegador. Esto permite a la implementación cada uno variar uno del otro, basándose en el concepto cliente/servidor. Cuando utilizamos REST, HTTP no tiene estado. Cada mensaje contiene toda la información necesaria para comprender la petición cuando se combina el estado en el recurso.” (Marset, 2006)

“Como resultado, ni el cliente ni el servidor necesita mantener ningún estado en la comunicación. Cualquier estado mantenido por el servidor debe ser modelado como un recurso.

La restricción de no mantener el estado puede ser violada mediante cookies que mantienen las sesiones. Fielding advierte del riesgo a la privacidad y seguridad que frecuentemente surge del uso de cookies, así como la confusión y errores que pueden resultar de las interacciones entre cookies y el uso del boton “Go back” del navegador.

HTTP proporciona mecanismos para el control del caching y permite que ocurra una conversación entre el navegador y la caché del mismo modo que se hace entre el navegador y el servidor Web.” (Marset, 2006)

2.7 Interfaz basada en REST

“En vez de cubrir esto desde un punto de vista arquitectural, es aconsejable realizarlo a modo de receta. Existen una serie de pasos a tomar y una serie de preguntas a responder.

Antes de empezar a pensar en el servicio, debemos responder las siguientes preguntas en orden:

• ¿Qué son las URIs? Las cosas identificadas por URIs son “recursos”. Aunque es más apropiado decir que los Recursos son identificados mediante URIs.

Si solo existe una única URI como punto de acceso, posiblemente se esté creando un protocolo RPC. En tal caso, se debe desmenuzar el problema en tipos de recursos que se quieren manipular. Dos lugares donde se debe considerar cuando se buscan los recursos potenciales son las colecciones y las interfaces de búsqueda. Una colección de recursos es en sí mismo un recurso. Una interfaz de búsqueda también lo es, ya que el resultado de una búsqueda es otro conjunto de recursos.

A modo de ejemplo, supongamos un sistema de mantenimiento de una lista de contactos de empleados. En tal sistema cada usuario debería tener su propia URI con una apropiada representación. Además, la colección de recursos es otro recurso.

Por tanto, hemos identificado dos tipos de recursos, por tanto, habrá dos tipos de URIs: o Employee (Una URI por empleado). o AllEmployee (Listado de los empleados).” (Marset, 2006)

“• ¿Cuál es el formato? Cuando hablamos informalmente de formato, estamos hablando de la representación. Como ya hemos comentado con anterioridad, no se puede acceder directamente al recurso, hay que obtener una representación. Esta representación puede ser un documento HTML, XML, una imagen, dependiendo de la situación.

Para cada uno de los recursos, se tiene que decidir cuál va a ser su representación. Si es posible, reutilizar formatos existentes, ayudará a incrementar la oportunidad de que nuestro sistema se componga con otros.

Continuando con nuestro ejemplo, el formato de representación va a ser XML, ya que es el principal formato para intercambio de información. El formato del empleado podría ser el siguiente:” (Marset, 2006)

```
<employee xmlns='HTTP://example.org/my-example-ns/'>
        <name>Full name goes here.</name>
        <title>Persons title goes here.</title>
        <phone-number>Phone number goes here.</phone-number>
 </employee>
```

Para el listado de empleados podríamos tomar este otro:

```
<employee-list xmlns='HTTP://example.org/my-example-ns/'>
      <employee-ref href="URI of the first employee"/> Full name of the first employee goes here.</employee>
   <employee-ref href="URI of employee #2"/>Full name</employee>
   <employee-ref href="URI of employee #N"/>Full name</employee>
</employee-list>
```

“¿Qué métodos son soportados en cada URI? En este apartado tenemos que definir cómo van a ser referenciados los recursos. Por medio de las URIs y los métodos soportados el acceso va a ser posible. El acceso de puede hacer de muchas formas, recibiendo una representación del recurso (GET o HEAD), añadiendo o modificando una representación (POST o PUT) y eliminando algunas o todas las representaciones (DELETE).” (Marset, 2006)

Tabla 4 Métodos Soportados

HTTP	CRUD	Descripción
POST	CREATE	Crear un nuevo recurso
GET	RETRIEVE	Obtener la representación de un recurso
PUT	UPDATE	Actualizar un recurso
DELETE	DELETE	Eliminar un recurso

Fuente:

Continuando con nuestro ejemplo, ahora vamos a combinar recursos, representación y métodos:

Tabla 5 Recursos y Métodos

Recurso	Método	Representación
Employee	GET	Formato del empleado
Employee	PUT	Formato del empleado
Employee	DELETE	-
All Employee	GET	Formato de la lista de empleados
All Employee	POST	Formato de empleado

Fuente:

“¿Por qué surge el debate entre los Servicios Web basados en REST y SOAP?

Muchos diseñadores de Servicios Web están llegando a la conclusión que SOAP es demasiado complicado. Por tanto, están comenzando a utilizar Servicios Web basados en REST para mostrar cantidades de datos masivos. Este es el caso de grandes empresas como eBay y Google.

El problema principal surge del propósito inicial de SOAP. Esta tecnología fue originalmente pensada para ser una versión, sobre Internet, de DCOM o CORBA. Así lo demuestra su predecesor, el protocolo XML-RPC. Estas tecnologías lograron un éxito limitado antes de ser adaptadas. Esto es debido a que este tipo de tecnologías, las basadas en modelos RPC (Remote Procedure Call) son más adecuadas para entornos aislados, es decir, entornos donde se conoce perfectamente el entorno. La evolución en este tipo de sistemas es sencilla, se modifica cada usuario para que cumpla con los nuevos requisitos.” (Marset, 2006)

2.8 Servicio Web basado en REST

“Las pautas que seguir en el diseño de servicios Web basados en REST son las mismas que las descritas en el apartado dedicado a crear una interfaz REST. Por otra parte, hemos ampliado dicha información en el contexto de los Servicios Web:

• Identificar todas las entidades conceptuales que se desean exponer como servicio.

• Crear una URL para cada recurso. Los recursos deberían ser nombres no verbos (acciones). Por ejemplo, no utilizar esto:

http://www.service.com/entities/getEntity?id=001

Como podemos observar, getEntity es un verbo. Mejor utilizar el estilo REST, un nombre: http://www.service.com/entities/001

• Categorizar los recursos de acuerdo con si los clientes pueden obtener una representación del recurso o si pueden modificarlo. Para el primero, debemos hacer los recursos accesibles utilizando un HTTP GET. Para el último, debemos hacer los recursos accesibles mediante HTTP POST, PUT y/o DELETE.

• Todos los recursos accesibles mediante GET no deberían tener efectos secundarios. Es decir, los recursos deberían devolver la representación del recurso. Por tanto, invocar al recurso no debería ser el resultado de modificarlo. “ (Marset, 2006)

"• Ninguna representación debería estar aislada. Es decir, es recomendable poner hipervínculos dentro de la representación de un recurso para permitir a los clientes obtener más información.

• Especificar el formato de los datos de respuesta mediante un esquema (DTD, W3C Schema). Para los servicios que requieran un POST o un PUT es aconsejable también proporcionar un esquema para especificar el formato de la respuesta.

• Describir como nuestro servicio ha de ser invocado, mediante un documento WSDL/WADL o simplemente HTML." (Marset, 2006)

2.9 Comparación entre los servicios

Con lo visto en todo este documento podemos obtener la siguiente tabla en la que veremos más claramente las diferencias entre REST y SOAP, ya sabiendo que SOAP tiene un acoplado muy fuerte lo que nos permite es poder revisar fallos antes de poner en marcha la aplicación, mientras que REST tiene una escalabilidad muy alta.

Tabla 6 Comparación entre SOAP y REST

	REST	**Nota**	**SOAP**	**Nota**
Características	• Las operaciones se definen en los mensajes. • Una dirección única para cada instancia del proceso. • Cada objeto soporta las operaciones estándares definidas. • Componentes débilmente acoplados	4	• Las operaciones son definidas como puertos WSDL. • Dirección única para todas las operaciones. • Múltiples instancias del proceso comparten la misma operación. • Componentes fuertemente acoplados.	3
Ventajas declaradas	• Bajo consumo de recursos. • Las instancias del proceso son creadas explícitamente. • El cliente no necesita información de	4	• Fácil (generalmente) de utilizar. • La depuración es posible. • Las operaciones complejas pueden ser	4

	enrutamiento a partir de la URI inicial. • Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones. • Generalmente fácil de construir y adoptar.		escondidas detrás de una fachada. • Envolver APIs existentes es sencillo • Incrementa la privacidad. • Herramientas de desarrollo.	
Posibles desventajas	• Gran número de objetos. • Manejar el espacio de nombres (URIs) puede ser engorroso. • La descripción sintáctica / semántica muy informal (orientada al usuario). • Pocas herramientas de desarrollo.	4	• Los clientes necesitan saber las operaciones y su semántica antes del uso. • Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. • Las instancias del proceso son creadas implícitamente.	3
Tecnología	• Interacción dirigida por el usuario por medio de formularios. • Pocas operaciones con muchos recursos. • Mecanismo consistente de nombrado de recursos (URI). • Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	5	• Flujo de eventos orquestados. • Muchas operaciones pocos recursos. • Falta un mecanismo de nombrado. • Se centra en el diseño de aplicaciones distribuidas.	4

Protocolo	• XML auto descriptivo. • HTTP • HTTP es un protocolo de aplicación. • Síncrono.	3	• Tipado fuerte, XML Schema. • Independiente del transporte. • HTTP es un protocolo de transporte. • Síncrono y Asíncrono.	4
Descripción del servicio	• Confía en documentos orientados al usuario que define las direcciones de petición y las respuestas. • Interactuar con el servicio supone horas de testeado y depuración de URLs. • No es necesario el tipado fuerte, si ambos lados están de acuerdo con el contenido. • WADL propuesto en noviembre de 2006.	5	• WSDL. • Se pueden construir automáticamente stubs (clientes) por medio del WSDL. • Tipado fuerte • WSDL 2.0.	3
Gestión del estado	• El servidor no tiene estado (stateless). • Los recursos contienen datos y enlaces representados transiciones a estados válidos. • Los clientes mantienen el estado siguiendo los enlaces. • Técnicas para añadir sesiones: Cookies	4	• El servidor puede mantener el estado de la conversación. • Los mensajes solo contienen datos. • Los clientes mantienen el estado suponiendo el estado del servicio. • Técnicas para añadir sesiones: Cabecera de sesión (no estándar).	3

Seguridad	• HTTPS. • Implementado desde hace muchos años. • Comunicación punto a punto segura.	4	• WS-Security. • Las implementaciones están comenzando a aparecer ahora. • Comunicación origen a destino segura	4
Metodología de diseño	• Identificar recursos a ser expuestos como servicios. • Definir URLs para direccionarlos. • Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE). • Implementar e implantar el servidor Web.	4	• Listar las operaciones del servicio en el documento WSDL. • Definir un modelo de datos para el contenido de los mensajes. • Elegir un protocolo de transporte apropiado y definir las correspondientes políticas QoS, de seguridad y transaccional. • Implementar e implantar el contenedor del servicio Web.	3
Sumatoria		37		31

Fuente: Propia

Al culminar esta tabla nos podemos ver que REST tiene una puntuación mientras que SOAP 31 por lo que utilizamos REST para creación del proyecto.

“Amazon posee ambos estilos de uso de sus servicios Web. Pero el 85% de sus clientes prefieren la interfaz REST. A pesar de la promoción que las empresas han invertido para ensalzar a SOAP, parece que es evidente que los desarrolladores prefieren, en algunos casos, la aproximación más sencilla: REST.

Todo lo visto a lo largo del documento como en el párrafo anterior, parece que nos da pistas para el futuro éxito de REST. Aunque, todos sabemos que en el mundo de la tecnología no siempre acaba triunfando la tecnología mejor, recuérdese el caso de VHS vs BetaMax.

Lo que está claro es que falta una pieza en la Web. La comunicación Hombre – Máquina parece que funciona, pero la comunicación Máquina – Máquina sigue siendo un reto. Recientemente, se ha presentado una propuesta donde los principios de REST han sido aplicados a los estándar y guías de diseño asociadas con la nueva versión de SOAP, es decir, SOAP podría ser utilizado de tal manera que no violara los principios de REST. Esto parece prometedor. Pero en mi opinión, este tipo de propuestas (incluyendo a REST) no triunfaran si la industria no apuesta realmente por ellas (herramientas y frameworks).” (Marset, 2006)

“¿Dónde es útil REST?

Tanto los arquitectos como los desarrolladores necesitan decidir cuál es el estilo adecuado para las aplicaciones. En algunos casos es adecuado un diseño basado en REST, se listan a continuación:

- El servicio Web no tiene estado. Una buena comprobación de esto consistiría en considerar si la interacción puede sobrevivir a un reinicio del servidor.
- Una infraestructura de caching puede mejorar el rendimiento. Si los datos que el servicio Web devuelve no son dinámicamente generados y pueden ser cacheados, entonces la infraestructura de caching que los servidores Web y los intermediarios proporcionan, pueden incrementar el rendimiento.
- Tanto el productor como el consumidor del servicio conocen el contexto y contenido que va a ser comunicado. Ya que REST no posee todavía (aunque hayamos visto una propuesta interesante) un modo estándar y formal de describir la interfaz de los servicios Web, ambas partes deben estar de acuerdo en el modo de intercambiar de información.
- El ancho de banda es importante y necesita ser limitado. REST es particularmente útil en dispositivos con escasos recursos como PDAs o teléfonos móviles, donde la sobrecarga de las cabeceras y capas adicionales de los elementos SOAP debe ser restringida.
- La distribución de Servicios Web o la agregación con sitios Web existentes puede ser fácilmente desarrollada mediante REST. Los desarrolladores pueden utilizar tecnologías como AJAX y toolkits como DWR (Direct Web Remoting) para consumir el servicio en sus aplicaciones Web.” (Marset, 2006)

“¿Dónde es útil SOAP?

Un diseño basado en SOAP es adecuado cuando:

- Se establece un contrato formal para la descripción de la interfaz que el servicio ofrece. El lenguaje de Descripción de Servicios Web (WSDL), como ya sabemos, permite describir con detalles el servicio Web.
- La arquitectura debe abordar requerimientos complejos no funcionales. Muchas especificaciones de servicios Web abordan tales requisitos y establecen un vocabulario común para ellos. Algunos ejemplos incluyen transacciones, seguridad, direccionamiento, … La mayoría de las aplicaciones del mundo real se comportan por encima de las operaciones CRUD y requieren mantener información contextual y el estado conversacional. Con la aproximación REST, abordar este tipo de arquitecturas resulta más complicado.
- La arquitectura necesita manejar procesado asíncrono e invocación. En estos casos, la infraestructura proporcionada por estándares como WSRM y APIs como JAX-WS junto con la asincronía por el lado del cliente nos permitirán el soporte de estas características.” (Marset, 2006)

2.10 Pruebas con JMeter

Se realizan las pruebas necesarias para ver el rendimiento de los webs services soap y rest, para ver los resultados que arrojan.

2.10.1 Métricas de JMeter con 50 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	50	912	384	4380	612.07	0.00%	9.7/sec	5.38	5.66
TOTAL	50	912	384	4380	612.07	0.00%	9.7/sec	5.38	5.66

Ilustración 13 Soap 50 usuarios

Fuente: Propia

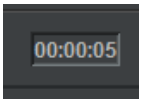


Ilustración 14 Tiempo de respuesta Soap 50 usuarios

Fuente: Propia.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec
HTTP Request	50	1700	319	2307	343.46	0.00%	19.1/sec	40.56	4.37
TOTAL	50	1700	319	2307	343.46	0.00%	19.1/sec	40.56	4.37

Ilustración 15 Rest con 50 usuarios

Fuente: Propia

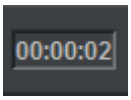


Ilustración 16 Tiempo de respuesta Rest con 50 usuarios

Fuente: Propia

2.10.2 Métricas de JMeter con 100 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	100	889	399	2097	410.58	0.00%	33.9/sec	18.78	19.78
TOTAL	100	889	399	2097	410.58	0.00%	33.9/sec	18.78	19.78

Ilustración 17 Soap con 100 usuarios

Fuente: Propia.

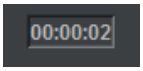


Ilustración 18 Tiempo de respuesta de Soap con 100 usuarios

Fuente: Propia.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec
HTTP Request	100	830	175	1475	401.48	0.00%	40.2/sec	85.40	9.20
TOTAL	100	830	175	1475	401.48	0.00%	40.2/sec	85.40	9.20

Ilustración 19 Rest con 100 usuarios

Fuente: Propia

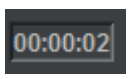


Ilustración 20 Tiempo de respuesta de Rest con 100 usuarios

Fuente: Propia.

2.10.3 Métricas de JMeter con 500 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	2204	406	16342	1866.75	0.00%	28.8/sec	15.95	16.79
TOTAL	500	2204	406	16342	1866.75	0.00%	28.8/sec	15.95	16.79

Ilustración 21 Soap con 500 usuarios

Fuente: Propia.

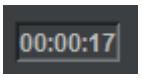


Ilustración 22 Tiempo de respuesta de Soap con 500 usuarios

Fuente: Propia.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	500	3861	151	7968	2543.24	20.20%	53.7/sec	67.82	9.84	1292.9
TOTAL	500	3861	151	7968	2543.24	20.20%	53.7/sec	67.82	9.84	1292.9

Ilustración 23 Rest con 500 usuarios

Fuente: Propia.

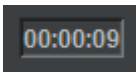


Ilustración 24 Tiempo de respuesta de Rest con 500 usuarios

Fuente: Propia.

2.10.4 Métricas de JMeter con 1000 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	1000	4284	492	21001	3039.27	13.00%	46.5/sec	25.59	27.07
TOTAL	1000	4284	492	21001	3039.27	13.00%	46.5/sec	25.59	27.07

Ilustración 25 Soap con 1000 usuarios

Fuente: Propia.

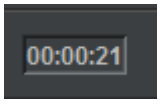


Ilustración 26 Tiempo de respuesta de Soap con 1000 usuarios

Fuente: Propia

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s	Sent KB/sec
HTTP Request	1000	9388	2220	27140	3215.82	4.10%	58.4/sec	182.88	2.18
TOTAL	1000	9388	2220	27140	3215.82	4.10%	58.4/sec	182.88	2.18

Ilustración 27 Rest con 1000 usuarios

Fuente: Propia.

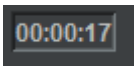


Ilustración 28 Tiempo de respuesta de Rest con 1000 usuarios

Fuente: Propia

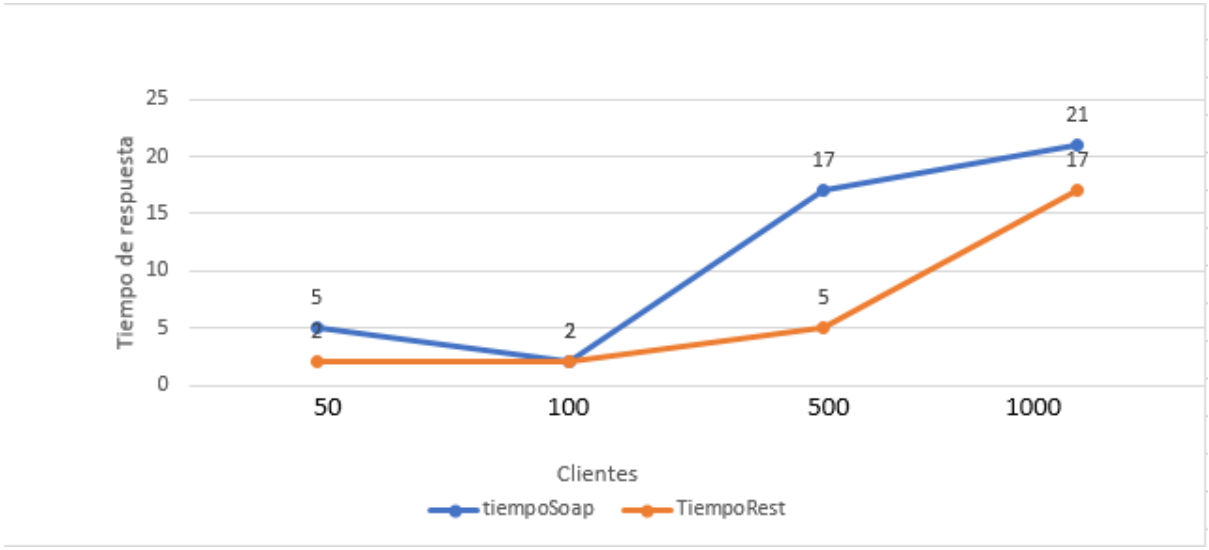


Ilustración 29 Comparación de Tiempo de Respuesta.

Fuente: Propia.

En la Ilustración 29 muestra que la diferencia del tiempo de respuesta de Soap y Rest de acuerdo con el número de usuarios que acceden a la información. La tendencia son líneas divergentes lo que implica que mientras más usuarios accedan a esta será mejor el rendimiento con REST y mayor será la diferencia en los tiempos de respuesta.

Tabla 7 Envíos de paquetes por segundo

Clientes	RendimientoSoap	RendimientoRest	TiempoSoap	TiempoRest
50	9,7	19,1	5	2
100	33,9	40,2	2	2
500	28,8	53,7	17	9
1000	46,5	58,4	21	17

Fuente: Propia.

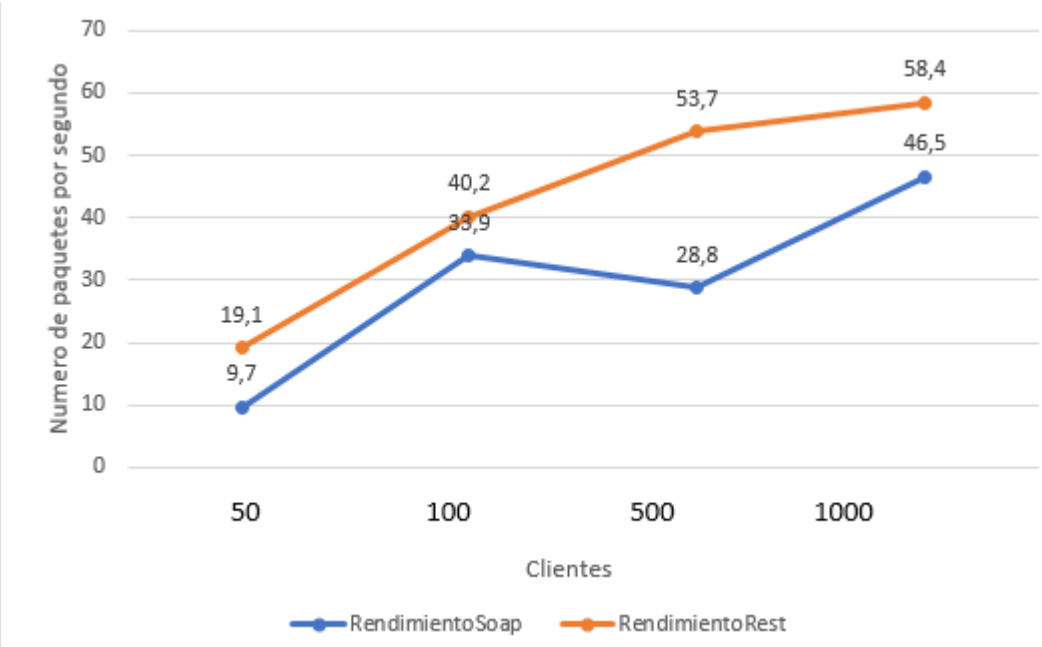


Ilustración 30 Rendimiento de Soap y Rest al envío de paquetes por segundo.

Fuente: Propia.

En la Ilustración 30 muestra la diferencia de rendimiento al momento de enviar paquetes por segundo que tiene Soap y Rest, de acuerdo con el número de usuarios que acceden a la información. En la ilustración se visualiza el incremento de REST con respecto a SOAP, evidenciando que REST envía más paquetes en menor tiempo haciéndolo más eficaz.

2.11 Red Hat OpenShift

“Red Hat OpenShift es una plataforma de aplicaciones en contenedores que proporciona a la empresa docker y kubernetes. Independientemente de la arquitectura de sus aplicaciones, OpenShift le permite diseñar, desarrollar e implementar de forma fácil y rápida en casi cualquier infraestructura, pública o privada. Ya sea de forma local, alojada o en una nube pública, usted tiene una plataforma galardonada para obtener su próxima gran idea para comercializar antes de que su competencia lo haga.

Kubernetes empresariales

Red Hat OpenShift es una plataforma de aplicaciones en contenedores completa que integra de forma nativa tecnologías como docker y kubernetes (un poderoso gestor de clúster de contenedores y sistema de coordinación) y las combina con una base empresarial en Red Hat Enterprise Linux.” (openshift, s.f.)

“OpenShift integra la arquitectura, los procesos, las plataformas y los servicios necesarios para habilitar el desarrollo y los equipos operativos.

Aplicaciones con estado y sin estado

Red Hat OpenShift ejecuta y da soporte a las aplicaciones con estado y sin estado. Esto le permite aprovechar los contenedores sin necesidad de modificar completamente la arquitectura de sus aplicaciones empresariales.

Modernice sus aplicaciones

Con OpenShift, usted puede fácilmente deshacerse y reemplazar los marcos costosos y desactualizados. Modernizar sus aplicaciones existentes para basarlas en microservicios y obtener beneficios de una mayor capacidad modular y de servicios. Tomar decisiones empresariales informadas mediante la integración completa de los datos y la creación de una mejor gestión de los procesos empresariales.

Diseñe una nube híbrida mejor y más potente

Usted puede desarrollar y dar soporte a OpenShift en cualquier lugar que Red Hat Enterprise Linux (la base para muchas nubes públicas y privadas) se implementa y admita. Esto incluye Amazon Web Services, Azure, Google Cloud Platform, VMware y más. Con OpenShift, usted puede proporcionar una plataforma única de aplicaciones de contenedor en todas estas nubes públicas y privadas.” (openshift, s.f.)

CAPÍTULO 3

En este capítulo se hace un estudio de la situación actual del club principalmente en la zona a efectuar la aplicación del presente proyecto de grado, los dispositivos electrónicos seleccionados y el resultado de este. La metodología SCRUM es el proceso de la ingeniería de software que permite elaborar un sistema de calidad bajo tareas asignadas. En este capítulo se desarrollan las fases de: inicio, elaboración, construcción y transición.

3.1 Instalaciones del club

Actual mente el club no tiene una instalación propia, realiza sus entrenamientos en el estadio de las palmas. Dentro de este no se encuentra ninguna oficina por lo que los datos son almacenados en la computadora personal del licenciado Pedro Carlosama.

3.2 Dispositivos y herramientas usadas

El club formativo especializado juvenil caleño lleva sus registros en una computadora personal que se muestra en la ilustración, en esta computadora se escriben los datos de los partidos en Excel y algunos datos en Word.

El proceso campeonato se encuentra la hoja de vocalía en donde se registran los datos de cada partido como se muestra en la ilustración.

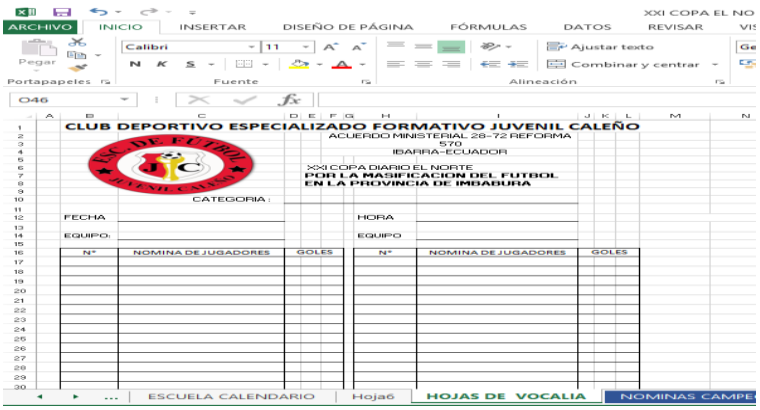


Ilustración 31 Hoja de vocalía

Fuente: propia

JDK 8.0

JDK 8 es un superconjunto de JRE 8 que contiene herramientas para compilar y depurar las aplicaciones en el lenguaje Java. Además, al JRE 8 proporciona las bibliotecas, la máquina virtual de Java (JVM) para ejecutar las aplicaciones.

Ilustración 32 Logo del JDK de Java

Fuente: https://www.hd-tecnologia.com/oracle-anuncia-java-8/

CodeIgniter

Es una librería muy potente que sirve para la conexión de web services Rest, gracias a su api Rest que facilita establecer la conexión con el servidor y poder manejar la información de manera más sencilla y óptima.

Ilustración 33 CodeIgniter

Fuente: www.codeigniter.com.

MySQL

“MySQL es la base de datos de código abierto más popular del mercado. Gracias a su rendimiento probado, a su fiabilidad y a su facilidad de uso, MySQL se ha convertido en la base de datos líder elegida para las aplicaciones basadas en web y utilizada por propiedades web de perfil alto, como Facebook, Twitter, YouTube y los cinco sitios web principales. Además, es una elección muy popular como base de datos integrada, distribuida por miles de ISV y OEM.” (MySQL, s.f.)

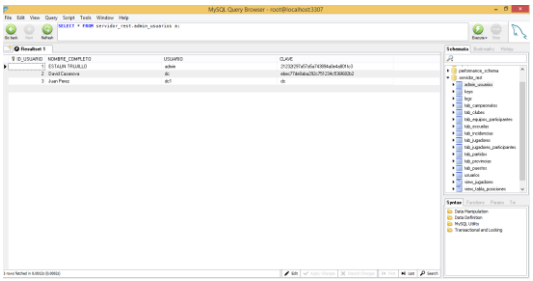


Ilustración 34 Entorno de desarrollo MySQL

Fuente: Propia

NetBeans IDE 8.2

Para el desarrollo de la aplicación se usa la herramienta NetBeans es un entorno de desarrollo integrado libre, realizado principalmente para Java.

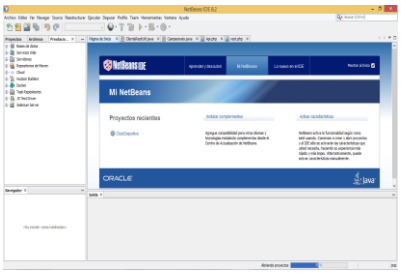


Ilustración 35 Entorno de Desarrollo NetBeans

Fuente: propia

3.3 Roles del sistema

En la metodología SCRUM, el equipo de trabajo prioriza en la construcción de un software de calidad, para lo cual realizarlo en equipo es esencial; cada participante del proyecto debe colaborar de la mejor manera por lo que deben tener claro el rol que desempeña y sus funciones a cargo.

A continuación, se presenta el equipo de trabajo para el desarrollo del proyecto.

Tabla 8 Roles del sistema.

Rol	Nombre	Cargo
Jefe de Proyecto SRUM Master	Estalin Trujillo	Desarrollador
Propietario del proyecto Product Owner	Lic. Pedro Carlosama Presidente del club deportivo Juvenil Caleño	Entidad Auspiciante
Equipo de Desarrolladores Team Masters	Estalin Trujillo	Desarrollador
Externos Interesados StakeHolders	Ing. Diego Trejo	Docente Tutor

Fuente: Propia.

3.4 Historias de usuarios y criterios de aceptación

En la metodología ágil SCRUM la forma de levantar los requerimientos de usuario es mediante el uso de historias de usuario, las cuales se enfocan en lo que el usuario necesita hacer, las historias de usuario pueden tener los siguientes campos:

- Identificador (ID) de la Historia: Identificador que se le asigna a la historia de usuario.
- Rol: Muestra el rol en el que el usuario puede realizar la tarea que se va a describir a
- continuación.
- Funcionalidad: Muestra la funcionalidad que se va a poder realizar en la aplicación.
- Número de escenario: Los números de escenarios que se puedan presentar realizando la
- funcionalidad antes mencionada.
- Criterio: Descripción de los diferentes escenarios que se pueden presentar.
- Resultado: Es el comportamiento que el sistema tomaría en cada criterio.

Tabla 9 Historias de usuario y criterios de aceptación

		Enunciado de la Historia			Criterios de Aceptación	
ID Historia	Solicitante	Rol	Funcionalidad	Escenario	Criterio	Resultado
0001	Lic. Pedro Carlosama	Como administrador	Necesito que existan varios usuarios que administren el programa.	1	_____	Se crea 3 diferentes roles para ingresar al sistema.
0002	Lic. Pedro Carlosama	Como administrador	Necesito que en otra página me muestre todos los campeonatos.	1	_____	Botón que diga actualizar, para realizar de nuevo la petición al servidor, y se muestre ordenadas por fechas de ingreso.
0003	Lic. Pedro Carlosama	Como administrador	Necesito una página me muestre todos los jugadores.	1	_____	Mostrar una página que muestre todos los jugadores.
0004	Lic. Pedro Carlosama	Como administrador	Necesito que se pueda buscar los jugadores por la CI o por nombre.	1	Resultado con al menos una coincidencia	Mostrar página con los jugadores encontrados.
				2	Sin Resultados	Mostrar página texto "No hay registro para mostrar".
0005	Lic. Pedro Carlosama	Como administrador	Necesito que en una página me muestre la lista de jugadores y se pueda editar los datos.	1	Si existe uno o más jugadores en la lista.	El empleado o administrador pueda editar la información de dicho jugador o eliminarlo con diferentes botones.
				2	Si la lista no contiene jugadores.	Mostrar una página que diga: “No hay registro para mostrar”.

0006	Lic. Pedro Carlosama	Como administrador	Necesito ingresar nuevos jugadores, equipos, campeonatos e infracciones.	1	————	En cada página se muestra los botones para poder realizar un nuevo ingreso.

Fuente: Propia.

3.5 Pila de producto

En esta pila se colocan los requisitos en los que se estará trabajando en la fase de iteraciones.

Tabla 10 Pila de productos

ID	NOMBRE	OBSERVACION
R1	Análisis y estructuración del proyecto,	Creación del proyecto, análisis del modelo de datos.
R2	Creación módulo club	Creación, edición y modificación de las interfaces y actividades del módulo de club.
R3	Implementación en plataforma OPENSHIFT.	Implementación y pruebas de todo el proyecto en la plataforma.

Fuente: Propia.

3.6 Pila de tareas

En la pila de tareas se analiza los requerimientos, haciendo el desglosamiento de estos en pequeñas tareas para que los desarrolladores puedan cumplir con su implementación en un corto periodo de tiempo.

Tabla 11 Análisis y estructuración del proyecto

Análisis y Estructuración del Proyecto		
ID	NOMBRE	OBSEVACIONES
T1	Instalación de MySQL.	Instalación y configuración del motor de base de datos con el cual se va a trabajar el servidor.
T2	Instalación de Apache.	Instalación de servidor apache para poder testear la aplicación.
T3	Instalación de codeigniter.	Instalación y configuración del framework codeigniter.
T4	Creación de la base de datos.	Modelación y creación de base de datos del sistema con sus respectivas tablas, claves primarias y foráneas.
T5	Creación de Proyecto.	Se crea el proyecto.

Fuente: Propia.

Tabla 12 Creación del módulo club

Creación de Módulo Club.		
ID	NOMBRE	OBSEVACIONES
T6	Creación de interfaces modulo club.	Se crea la plantilla para el módulo club.
T7	Configuración página usuarios	Ninguna
T8	Configuración página parámetros.	Ninguna
T9	Configuración página campeonato.	Ninguna
T10	Configuración página resultados.	Ninguna

Fuente: Propia.

Tabla 13 Implementación en plataforma Openshift

Implementación de una plataforma Openshift		
ID	NOMBRE	OBSEVACIONES
T11	Investigación plataforma Openshift	Ninguna
T12	Creación plataforma Openshift.	Creación y configuración de la plataforma.
T13	Implementación de aplicación en la plataforma.	Subir información de la aplicación
T14	Pruebas de la aplicación.	Test de todas las propiedades de la aplicación.

Fuente: Propia.

3.7 Planificación del Proyecto

En esta sección del proyecto se planifica cada iteración o sprint, es decir, se define qué requisitos y que tareas se van a cumplir en cada iteración, y además se determina las fechas de inicio y fin para la ejecución de estas.

Tabla 14 Planificación del Proyecto.

ID	NOMBRE	INICIO	FIN	REQUERIMIENTOS
I1	Análisis y estructuración del proyecto.	02 de febrero de 2018	16 de febrero de 2018	R1, T1, T2, T3, T4, T5.
I2	Creación del módulo club	19 de febrero de 2018	02 de marzo de 2018	R2, T6, T7, T8, T9, T10.
I3	Implementación en plataforma Openshift	18 de abril de 2018	18 de mayo de 2018	R3, T11, T12, T13, T14.

Fuente: Propia.

3.8 Iteraciones

3.8.1 Iteración 1 Análisis y estructuración del proyecto.

- **Requerimiento Nro. 1.-** Análisis y estructuración del proyecto.
- **Tarea Nro. 1.-** Instalación de MySql.

Interfaz visual de administración de MySQL, para realizar modificación e inserciones en la base de datos “servidor_rest”.

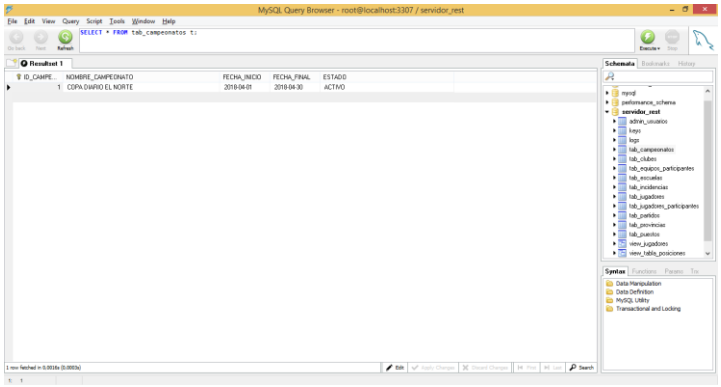


Ilustración 36 Instalación de MySQL

Fuente: propia

- **Tarea Nro. 2.-** Instalación de Apache.

El panel de control de XAMPP permite controlar el servicio de Apache.

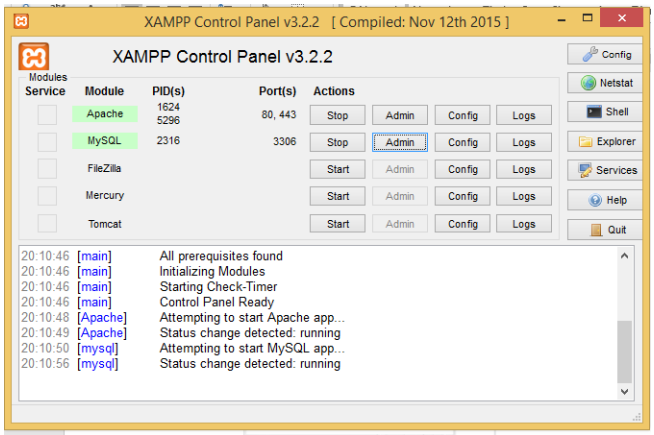


Ilustración 37 Instalación de Apache

Fuente: propia.

- **Tarea Nro. 3.-** Instalación de Codeigniter.

En la pila de tareas se analiza los requerimientos, haciendo el desglosamiento de estos en pequeñas tareas para que los desarrolladores puedan cumplir con su implementación en un corto periodo de tiempo.

- Para comenzar con la creación del API en PHP se tiene que descargar los siguientes recursos: - Codeigniter: https://codeigniter.com/
  Codeigniter-restserver: https://github.com/chriskacerguis/codeigniter-restserver
- Descomprimir Codeigniter
- Copiar los siguientes archivos de codeigniter-restserver a las siguientes carpetas de codeigniter (ambas carpetas tiene la misma estructura de archivos).

  application / libraries / Format.php

  application / libraries / REST_Controller.php

  application / config / rest.php application / language / *

  copier todos los archivos de lenguajes

- Ir a la carpeta raíz de codeigniter y crear el archivo .htaccess con el siguiente código.

```
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_FILENAME} !-d

RewriteRule ^(.*)$ index.php/$1 [L]
```

Ilustración 38 código de ingreso

Fuente: propia

- **Tarea Nro. 4.-** Creación de las bases de datos.

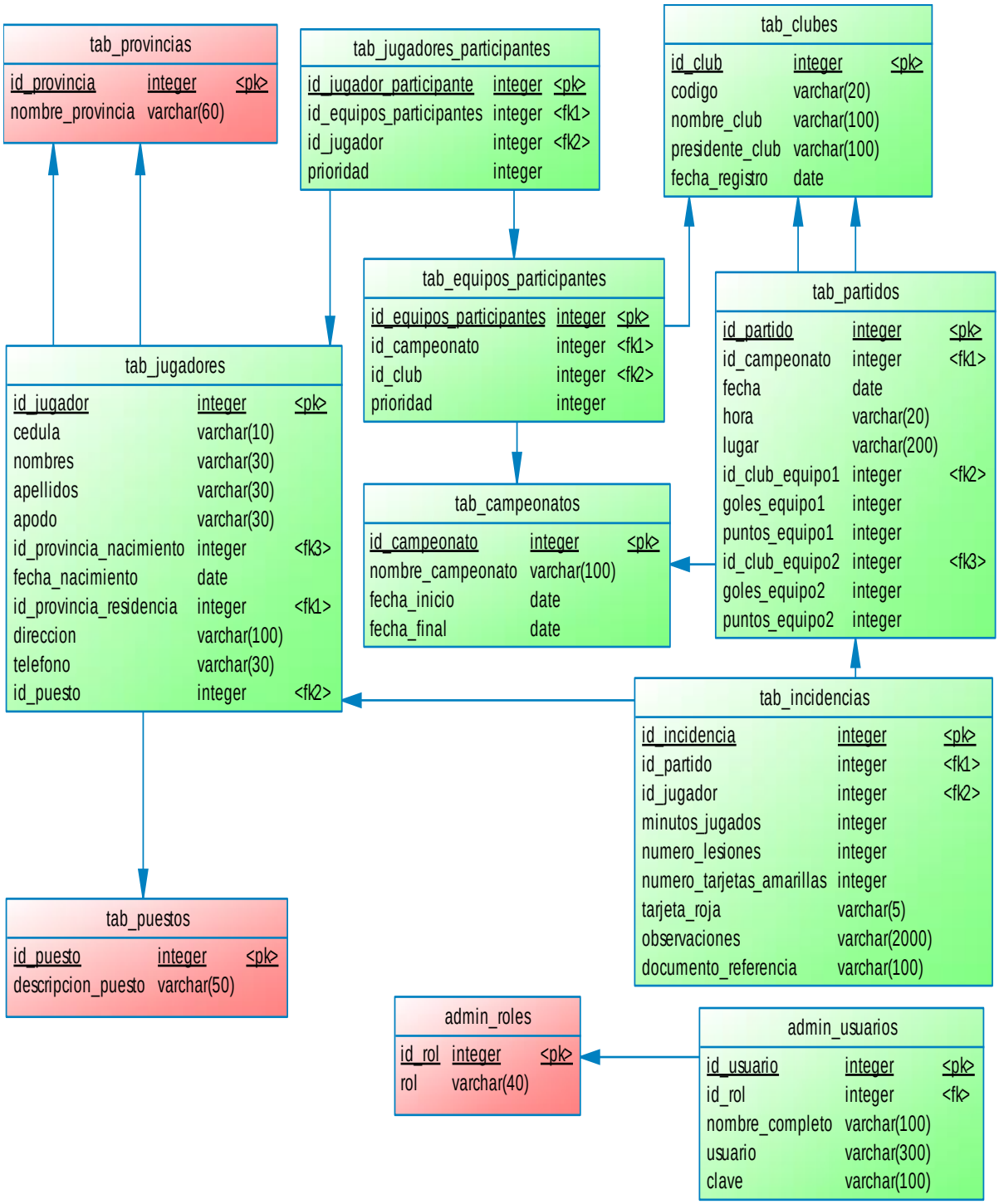


Ilustración 39 Modelo físico de la base de datos

Fuente: Propia.

Creación de tablas necesarias para el sistema con sus respectivas claves principales o foráneas.

- **Tabla Provincias:** esta tabla tiene los siguientes atributos:
  - Id_provincia: es integer, con este código se identifica las provincias, este atributo es la clave primaria de la tabla.
  - Nombre_provincia: es character varchar (60), es la descripción o nombre de la provincia.

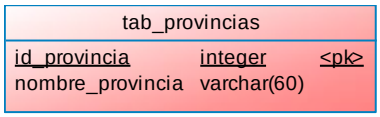


Ilustración 40 Tabla provincias

Fuente: Propia

- **Tabla Jugadores:** esta tabla tiene los siguientes atributos:
  - Id_jugador: es integer, con este código se identifica los jugadores, este atributo es la clave primaria de la tabla.
  - cedula: es character varchar (10), es el número de identificación de cada jugador.
  - nombres: es character varchar (30), son los nombres de cada jugador.
  - apellidos: es character varchar (30), son los apellidos de cada jugador.
  - apodo: es character varchar (30), es una identificación que se le da a los jugadores de acuerdo con las características de cada uno.
  - id_provincia_nacimiento: integer, es el código de identificación de la provincia y es clave foránea de la tabla provincias.
  - Fecha_nacimiento: es date, es la fecha de nacimiento de cada jugador.
  - id_provincia_residencia: integer, es el código de identificación de la provincia y es clave foránea de la tabla provincias.
  - direccion: es character varchar (100), lugar en donde se encuentra ubicada la residencia de los jugadores.
  - telefono: es character varchar (30), número de teléfono de los jugadores.
  - Id_puesto: integer, es el código de identificación de la posición en donde juega y es clave foránea de la tabla puesto.

tab_jugadores		
id_jugador	integer	<pk>
cedula	varchar(10)	
nombres	varchar(30)	
apellidos	varchar(30)	
apodo	varchar(30)	
id_provincia_nacimiento	integer	<fk3>
fecha_nacimiento	date	
id_provincia_residencia	integer	<fk1>
direccion	varchar(100)	
telefono	varchar(30)	
id_puesto	integer	<fk2>

Ilustración 41 Tabla Jugadores

Fuente: Propia

- **Tabla Puesto:** esta tabla tiene los siguientes atributos:
  - Id_puesto: es integer, con este código se identifica el puesto, este atributo es la clave primaria de la tabla.
  - Descripción_puesto: es character varchar (50), es la posición en la que juega el futbolista.

tab_puestos		
id_puesto	integer	<pk>
descripcion_puesto	varchar(50)	

Ilustración 42 Tabla Puestos

Fuente Propia

- **Tabla Jugadores Participantes:** esta tabla tiene los siguientes atributos:
  - Id_jugador_participante: es integer, con este código se identifica al jugador participante, este atributo es la clave primaria de la tabla.
  - Id_equipos_participantes: integer, es el código de identificación de los equipos participantes y es clave foránea de la tabla equipos participantes.
  - Id_jugador: integer, es el código de identificación del jugador y es clave foránea de la tabla jugadores.
  - Prioridad: es integer,

tab_jugadores_participantes		
id_jugador_participante	integer	<pk>
id_equipos_participantes	integer	<fk1>
id_jugador	integer	<fk2>
prioridad	integer	

Ilustración 43 Tabla jugadores participantes

Fuente: Propia

- **Tabla Equipos Participantes:** esta tabla tiene los siguientes atributos:
  - Id_equipos_participante: es integer, con este código se identifica a los equipos participantes, este atributo es la clave primaria de la tabla.
  - Id_campeonato: integer, es el código de identificación del campeonato y es clave foránea de la tabla campeonato.
  - Id_club: integer, es el código de identificación del club del jugador y es clave foránea de la tabla club.
  - Prioridad: es integer,

tab_equipos_participantes		
id_equipos_participantes	integer	<pk>
id_campeonato	integer	<fk1>
id_club	integer	<fk2>
prioridad	integer	

Ilustración 44 Tabla Equipos participantes

Fuente: Propia

- **Tabla Campeonatos:** esta tabla tiene los siguientes atributos:
  - Id_campeonato: es integer, con este código se identifica al campeonato, este atributo es la clave primaria de la tabla.
  - Nombre_campeonato: es character varchar (100), es el nombre de identificación del campeonato.
  - Fecha_Inicio: es date, la fecha en la que inicia el campeonato.
  - Fecha_Final: es date, la fecha en la que finaliza el campeonato.

tab_campeonatos		
id_campeonato	integer	<pk>
nombre_campeonato	varchar(100)	
fecha_inicio	date	
fecha_final	date	

Ilustración 45 Tabla Campeonatos

Fuete: Propia.

- **Tabla Incidencias:** esta tabla tiene los siguientes atributos:
  - Id_incidencia: es integer, con este código se identifica las incidencias, este atributo es la clave primaria de la tabla.
  - Id_partido: es integer, es el código de identificación del partido y es clave foránea de la tabla partidos.
  - Id_jugador: integer, es el código de identificación del jugador al que se sanciona y es clave foránea de la tabla jugadores.
  - minutos_jugados: es integer, los minutos que estuvo dentro de la cancha.
  - Numero_lesiones: es integer, las lesiones que se pudiesen presentar en el desarrollo del partido.
  - Numero_tarjetas_amarillas: es integer, numero acumulado de tarjetas amarillas a lo largo del campeonato.
  - Tarjeta_roja: es character varchar (5), si existiese o no una tarjeta roja.
  - Observacion: es character varchar (2000), las observaciones que hayan sucedido en el partido.

tab_incidencias		
id_incidencia	integer	<pk>
id_partido	integer	<fk1>
id_jugador	integer	<fk2>
minutos_jugados	integer	
numero_lesiones	integer	
numero_tarjetas_amarillas	integer	
tarjeta_roja	varchar(5)	
observaciones	varchar(2000)	
documento_referencia	varchar(100)	

Ilustración 46 Tabla incidencias

Fuente: Propia.

- **Tabla Clubes:** esta tabla tiene los siguientes atributos:
  - Id_club: es integer, con este código se identifica al campeonato, este atributo es la clave primaria de la tabla.
  - codigo: es character varchar (20), es el código de registro de cada club.
  - Nombre_club: es character varchar (100), es el nombre con el que se identifica cada club.
  - Presidente_club: es character varchar (100), nombre del presidente del club.
  - Fecha_registro: es date, fecha de registro del club.

tab_clubes		
id_club	integer	<pk>
codigo	varchar(20)	
nombre_club	varchar(100)	
presidente_club	varchar(100)	
fecha_registro	date	

Ilustración 47 Tabla Clubes

Fuente: Propia.

- **Tabla Partido:** esta tabla tiene los siguientes atributos:
  - Id_partido: es integer, con este código se identifica el partido, este atributo es la clave primaria de la tabla.
  - Id_campeonato: integer, es el código de identificación del campeonato y es clave foránea de la tabla campeonato.
  - Fecha: es date, fecha en la que se realizó el encuentro.
  - Hora: es character varchar (20), es la hora del partido.
  - lugar: es character varchar (200), es el lugar en donde se realizó el encuentro.
  - Id_club_equipo1: integer, es el código de identificación del club y es clave foránea de la tabla club.
  - Goles_equipo1: es integer, es el número de anotaciones del club.
  - Puntos_equipo1: es integer, es el número de puntos que se asigna al equipo dependiendo del resultado del partido.
  - Id_club_equipo2: integer, es el código de identificación del club y es clave foránea de la tabla club.
  - Goles_equipo2: es integer, es el número de anotaciones del club.
  - Puntos_equipo2: es integer, es el número de puntos que se asigna al equipo dependiendo del resultado del partido.

tab_partidos		
id_partido	integer	<pk>
id_campeonato	integer	<fk1>
fecha	date	
hora	varchar(20)	
lugar	varchar(200)	
id_club_equipo1	integer	<fk2>
goles_equipo1	integer	
puntos_equipo1	integer	
id_club_equipo2	integer	<fk3>
goles_equipo2	integer	
puntos_equipo2	integer	

Ilustración 48 Tabla Partidos

Fuente: Propia.

- **Tabla Admin Usuarios:** esta tabla tiene los siguientes atributos:
  - Id_usuario: es integer, con este código se identifica al usuario, este atributo es la clave primaria de la tabla.
  - Nombre_completo: es character varchar (100), son los nombres y apellidos del usuario.
  - Usuario: es character varchar (300), nombre de usuario para el ingreso a la sesión.
  - Clave: es character varchar (100), es la contraseña o clave para verificar el ingreso al sistema.

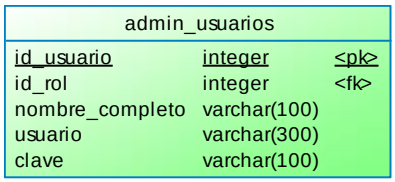


Ilustración 49 Tabla Administración usuarios

Fuente: Propia.

- **Tabla Admin Roles:** esta tabla tiene los siguientes atributos:
  - Id_rol: es integer, con este código se usa para saber el tipo de rol al que pertenece, este atributo es la clave primaria de la tabla.
  - Rol: es character varchar (40), diferentes roles que existen.

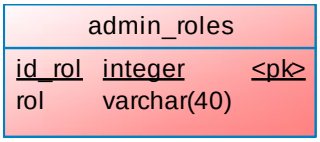


Ilustración 50 Tablas Roles

Fuente: Propia.

- **Tarea Nro. 5.-** Creación del proyecto.

Se crea los proyectos en Netbeans, uno echo en php y el otro realizo en formato web.

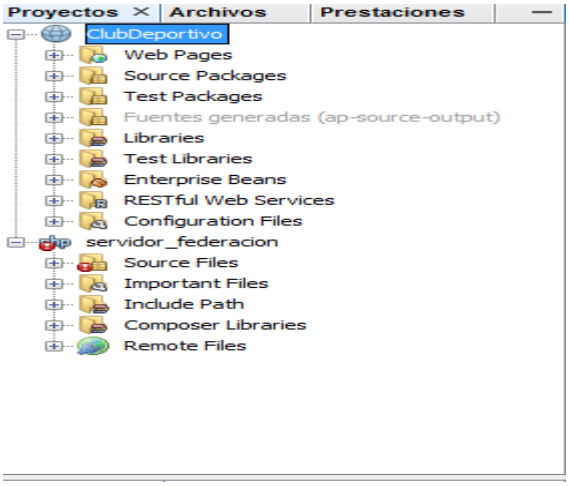


Ilustración 51 Creación proyecto

Fuente: Propia.

Tabla 15 Sprint 1 – Hoja Electrónica

SPRINT: 1 **FECHA INICIO**:02 de Feb. De 2018			02 - Feb
		Tareas Pendientes	0
REQUERIMIENTO	**TAREA**	**RESPONSABLE**	**ESTADO**
Análisis y estructura del proyecto	Instalación de MySQL.	Estalin Trujillo	TERMINADO
	Instalación de Apache.	Estalin Trujillo	TERMINADO
	Instalación de codeigniter.	Estalin Trujillo	TERMINADO
	Creación de la base de datos.	Estalin Trujillo	TERMINADO
	Creación de Proyecto.	Estalin Trujillo	TERMINADO

Fuente: Propia

Tabla 16 Sprint 1 - Pizarrón

PENDIENTE	EN PROCESO	TERMINADA
_____	_____	Instalación de MySql
_____	_____	Instalación de Apache
_____	_____	Instalación de Codeigniter
_____	_____	Creación de Base de Datos
_____	_____	Creación del Proyecto

Fuete: Propia

3.8.2 Iteración 2 Creación del módulo del club.

- **Requerimiento Nro. 2.-** Creación modulo club.
- **Tarea Nro. 6-** Creación de Interfaces de modulo club.

Módulo club, en el cual los infantes podrán revisar los clubs y también encuentra los jugadores.

Ilustración 52 Página Principal de Club

Fuente: Propia.

- **Tarea Nro. 7-** Configuración página super usuarios.

  Este usuario puede administrar los usuarios.

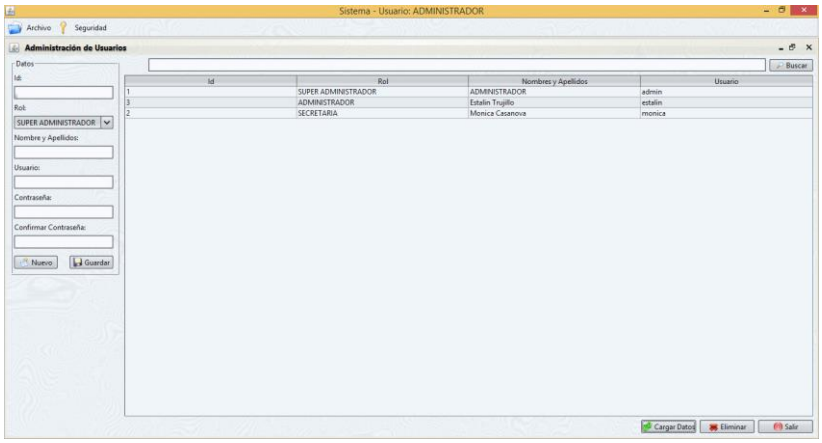


Ilustración 53 Pagina de Configuración de Clubs

Fuente: Propia

- **Tarea Nro. 8-** Configuración página parámetros.

  Muestra los clubs que participan.

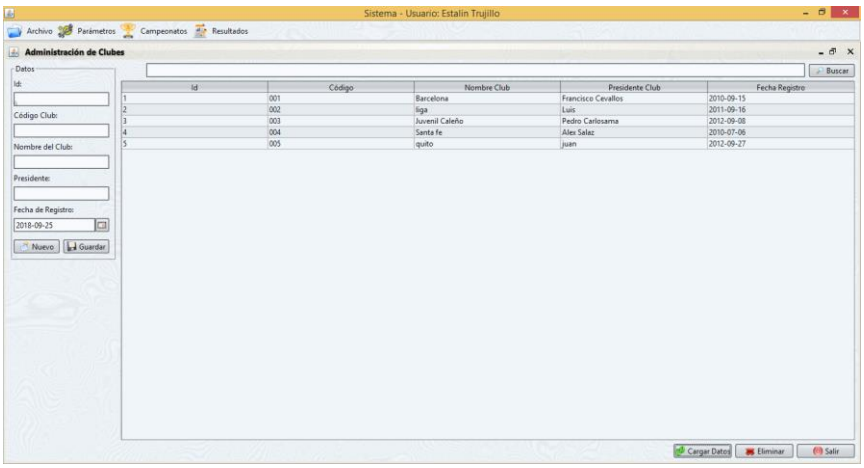


Ilustración 54 Pagina Configuración de Escuela de Futbol

Fuente: Propia.

- **Tarea Nro. 9-** Configuración página campeonato.

  Muestra lo que se encuentra establecido en campeonato.

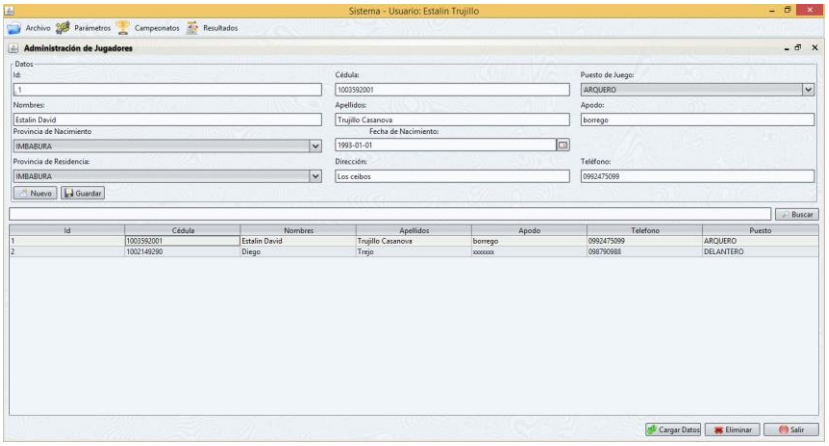


Ilustración 55 Pagina Configuración jugadores

Fuente: Propia.

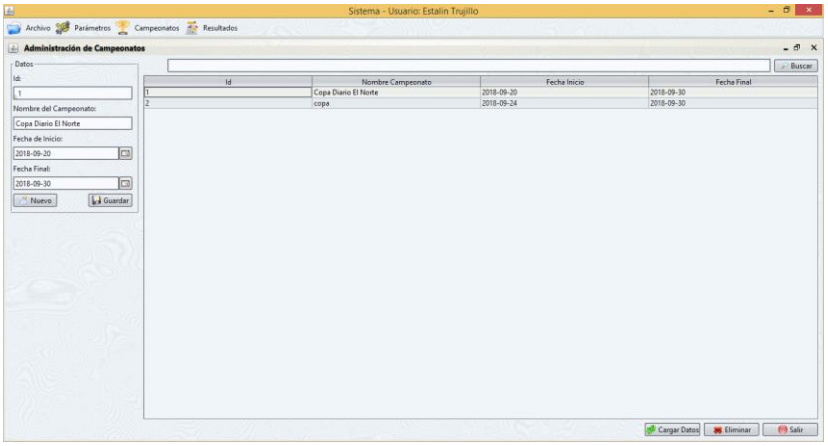


Ilustración 56 Página configuración jugadores

Fuente: Propia.

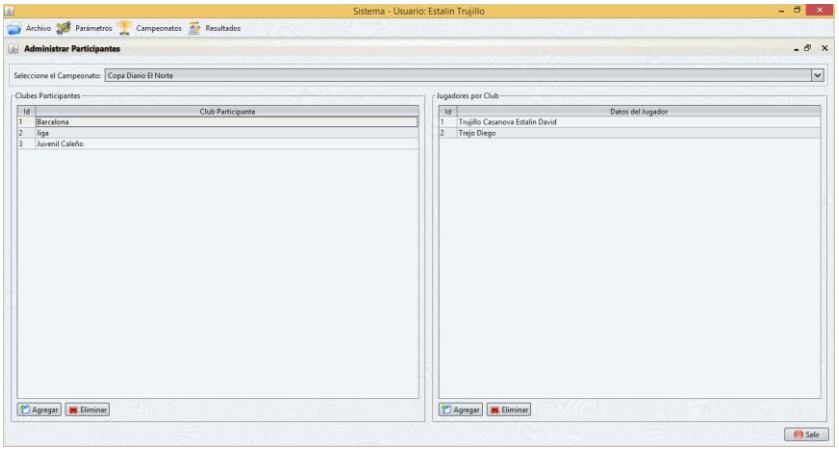


Ilustración 57 Pagina Configuración jugadores

Fuente: Propia.

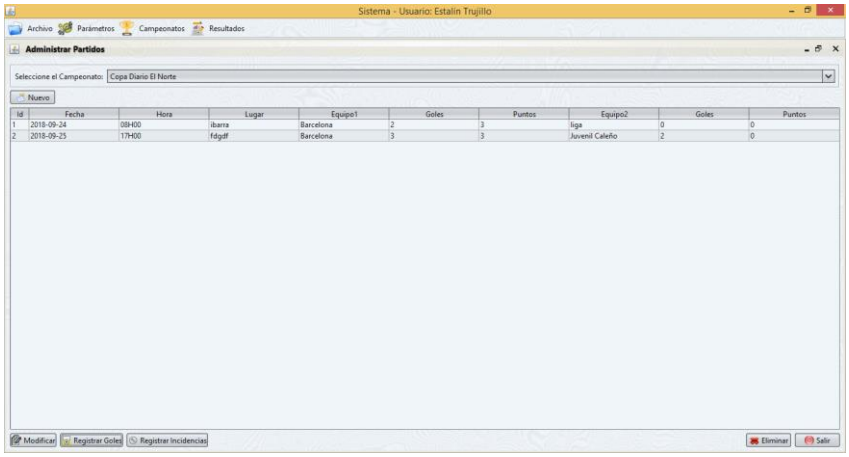


Ilustración 58 Pagina Configuración jugadores

Fuente: Propia.

- **Tarea Nro. 10-** Configuración página resultados.

  Muestra el listado de todos los campeonatos.

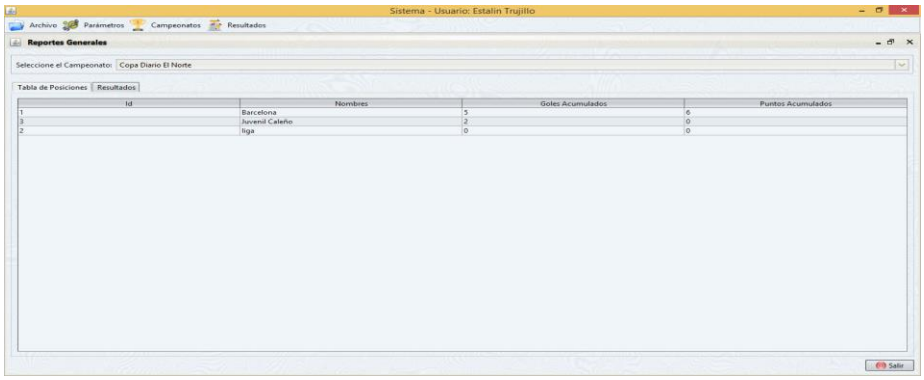


Ilustración 59 Pagina de Configuración de Campeonato

Fuente: Propia.

Tabla 17 Sprint 2 – Hoja Electrónica

SPRINT: 2 **FECHA INICIO**:18 de May. De 2018			18 - May
		Tareas Pendientes	0
REQUERIMIENTO	**TAREA**	**RESPONSABLE**	**ESTADO**
Creación módulo club.	Creación de Interfaces de modulo club.	Estalin Trujillo	TERMINADO
	Configuración página super usuarios	Estalin Trujillo	TERMINADO
	Configuración página parámetros.	Estalin Trujillo	TERMINADO
	Configuración página campeonato.	Estalin Trujillo	TERMINADO
	Configuración página resultados.	Estalin Trujillo	TERMINADO

Fuente: Propia.

Tabla 18 Sprint 2 - Pizarrón

PENDIENTE	**EN PROCESO**	**TERMINADA**
_____	_____	Creación de Interfaz modulo club
_____	_____	Configuración página super usuarios
_____	_____	Configuración página parámetros
_____	_____	Configuración página campeonato
_____	_____	Configuración página resultado

Fuente: Propia.

3.8.3 Iteración 3 Implementación en plataforma OPENSHIFT.

- **Requerimiento Nro. 3.-** Implementación en plataforma OPENSHIFT.
- **Tarea Nro. 11-** Investigación plataforma openshift.

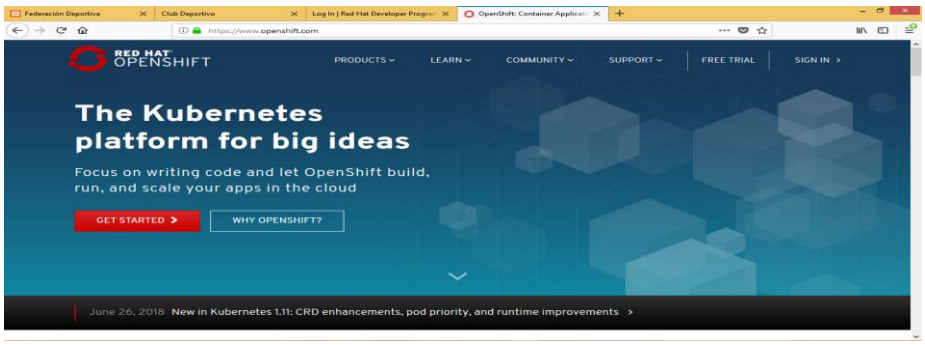


Ilustración 60 Openshift

Fuente: https://www.openshift.com/

- **Tarea Nro. 12-** Creación plataforma openshift.

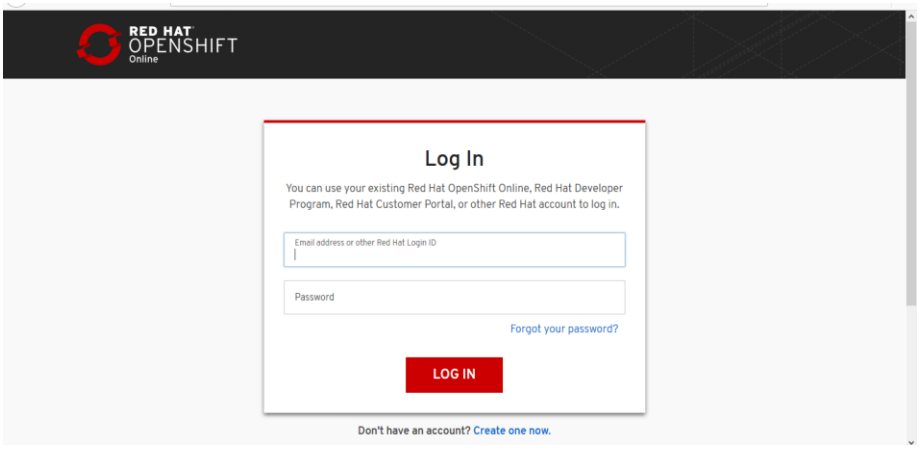


Ilustración 61 Creación de la cuenta Openshift

Fuente: https://www.openshift.com/

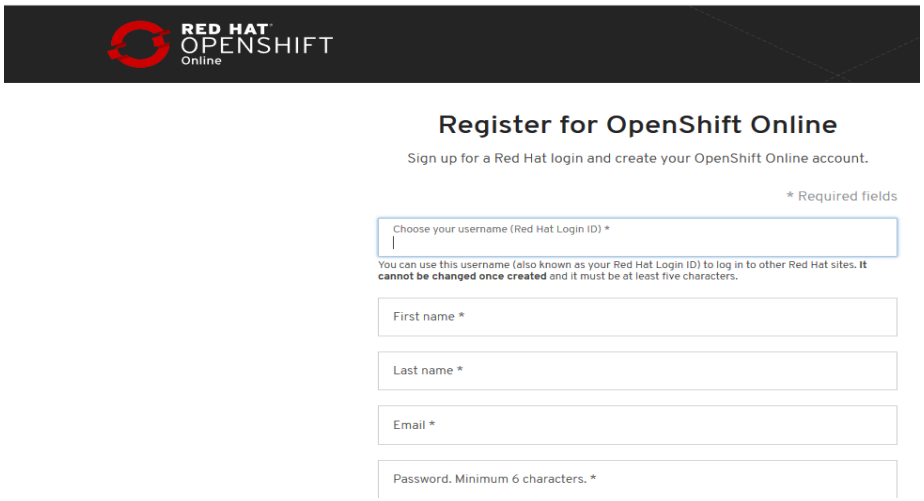


Ilustración 62 Registro de datos

Fuente: https://www.openshift.com/.

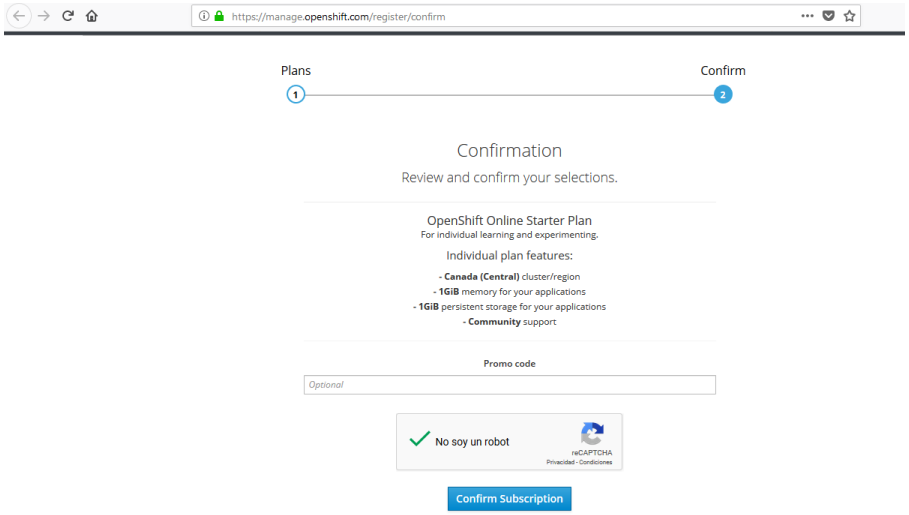


Ilustración 63 Confirmación de la Cuenta

Fuente; https://www.openshift.com/

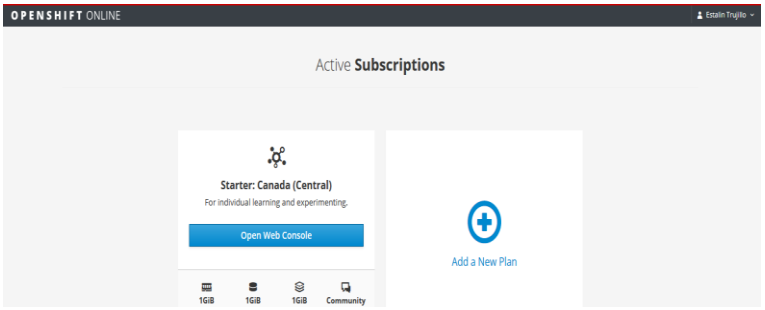


Ilustración 64 Pagina de Creación de Proyectos.

Fuente: https://www.openshift.com/

- **Tarea Nro. 13-** Implementación de aplicación en la plataforma.

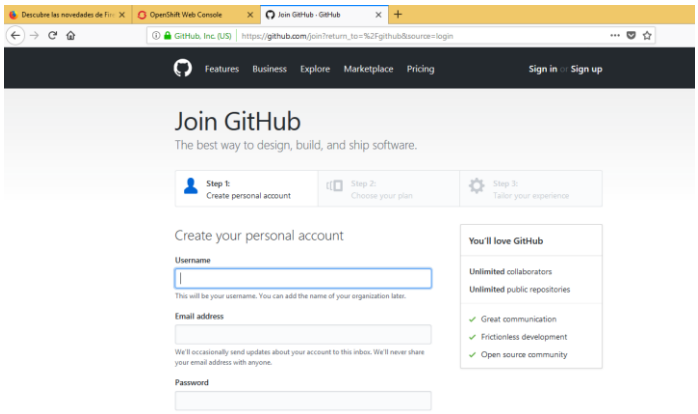


Ilustración 65 Página principal GitHub

Fuente: https://github.com

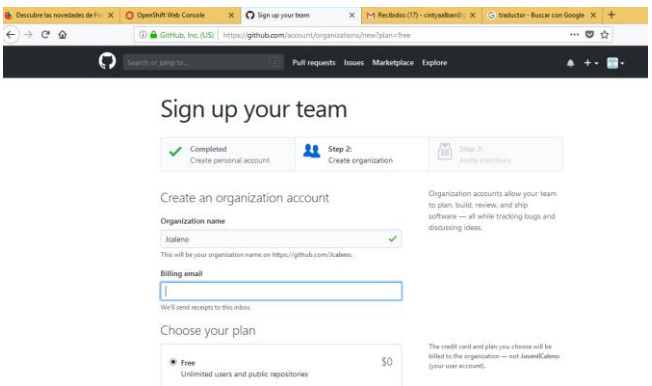


Ilustración 66 Registro de datos

Fuente: https://github.com

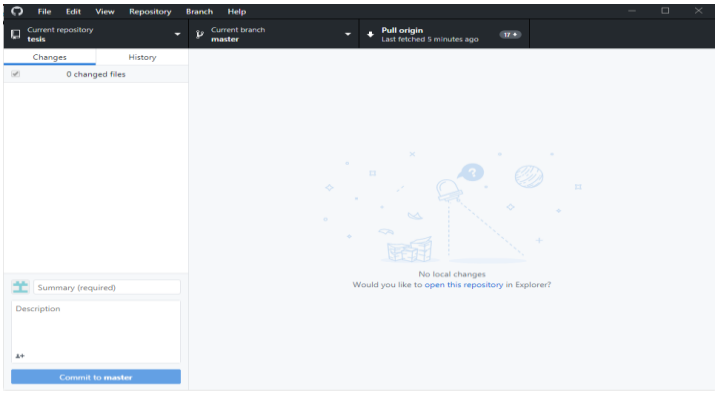


Ilustración 67 GitHub Desktop

Fuente: Propio.

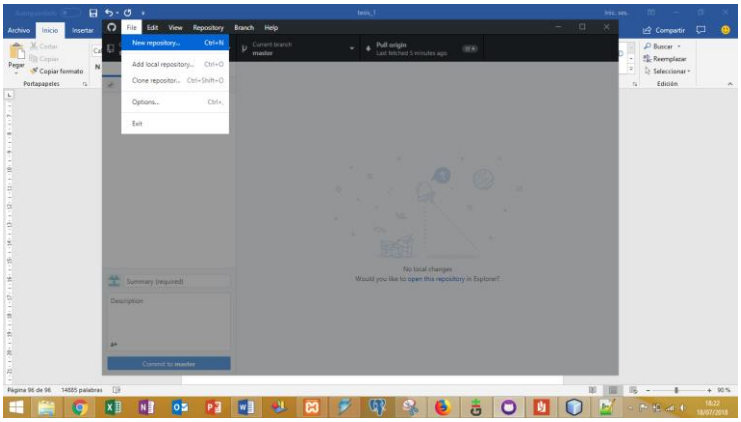


Ilustración 68 Crear Repositorio

Fuente: Propio.

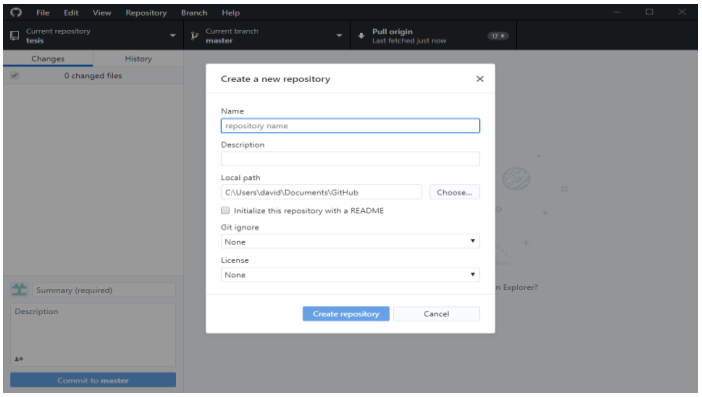


Ilustración 69 Nombre de Repositorio

Fuente: Propio.

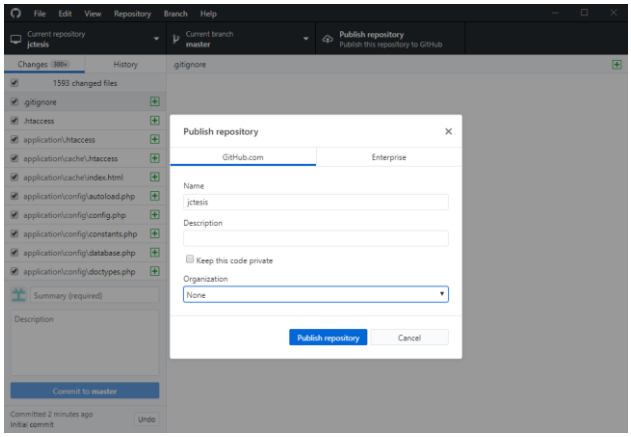


Ilustración 70 Publicar repositorio

Fuente: Propio.

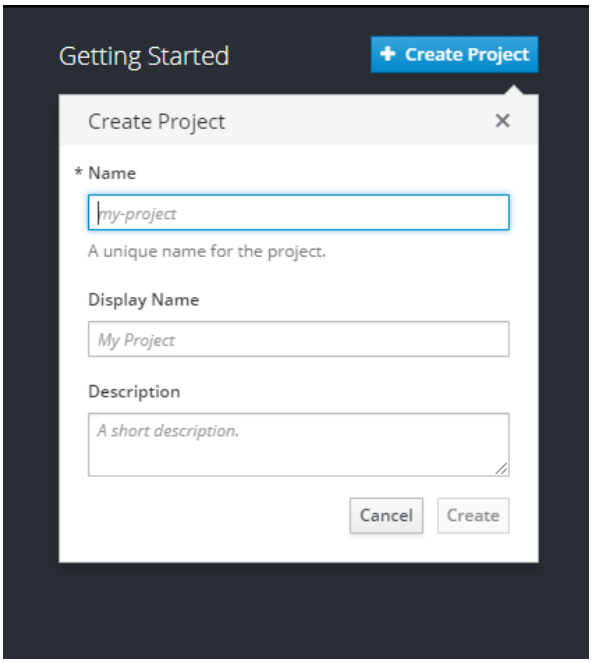


Ilustración 71 Crear proyecto Openshift

Fuente: https://www.openshift.com

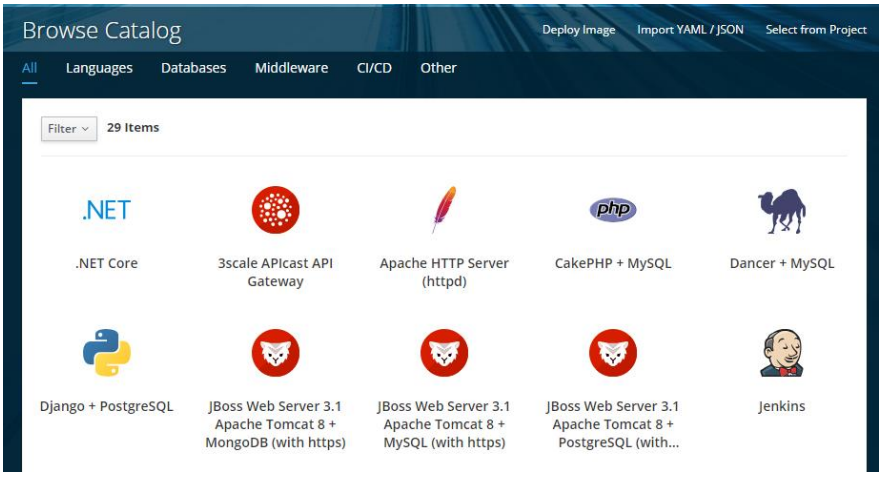


Ilustración 72 Catalogo de Programas

Fuente: https://www.openshift.com

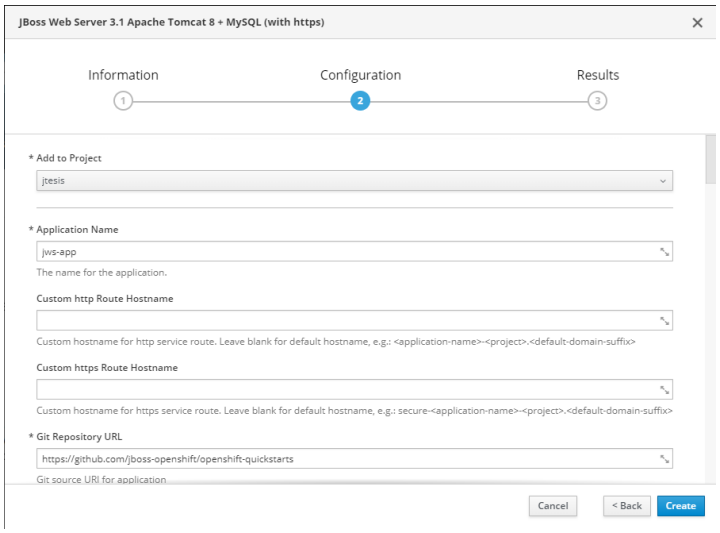


Ilustración 73 configuración entorno

Fuente: https://www.openshift.com

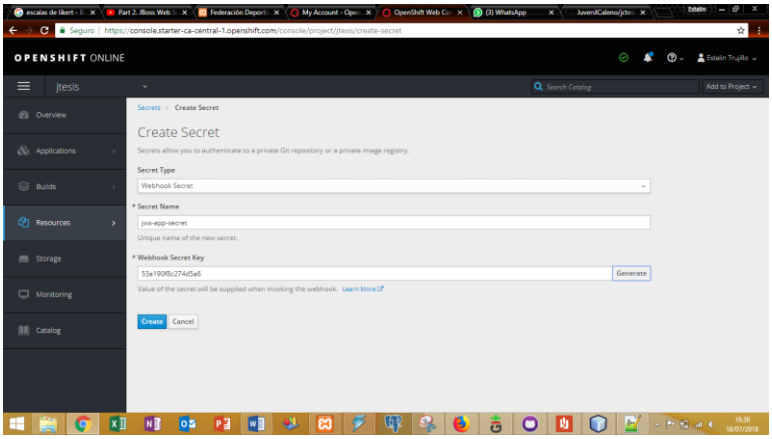


Ilustración 74 Configuración clave secreta

Fuente: https://www.openshift.com

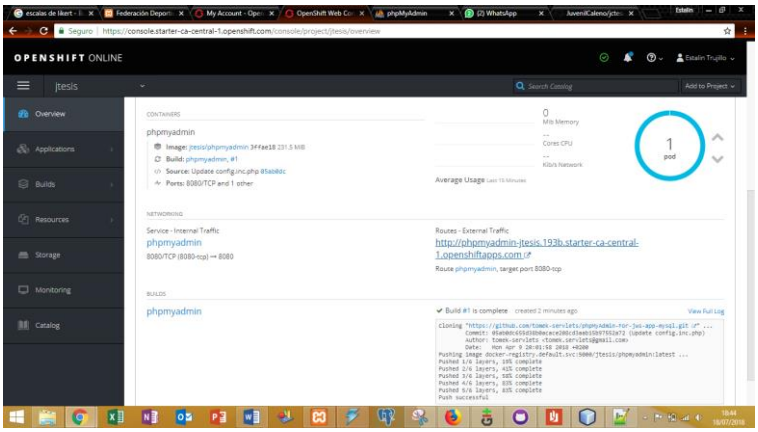


Ilustración 75 Creación phpMyAdmin

Fuente: https://www.openshift.com

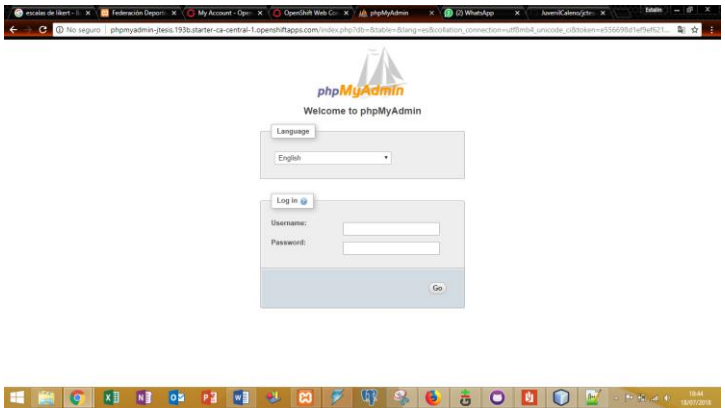


Ilustración 76 Página principal phpMyAdmin

Fuente: https://www.openshift.com

- **Tarea Nro. 14-** Pruebas en la plataforma.

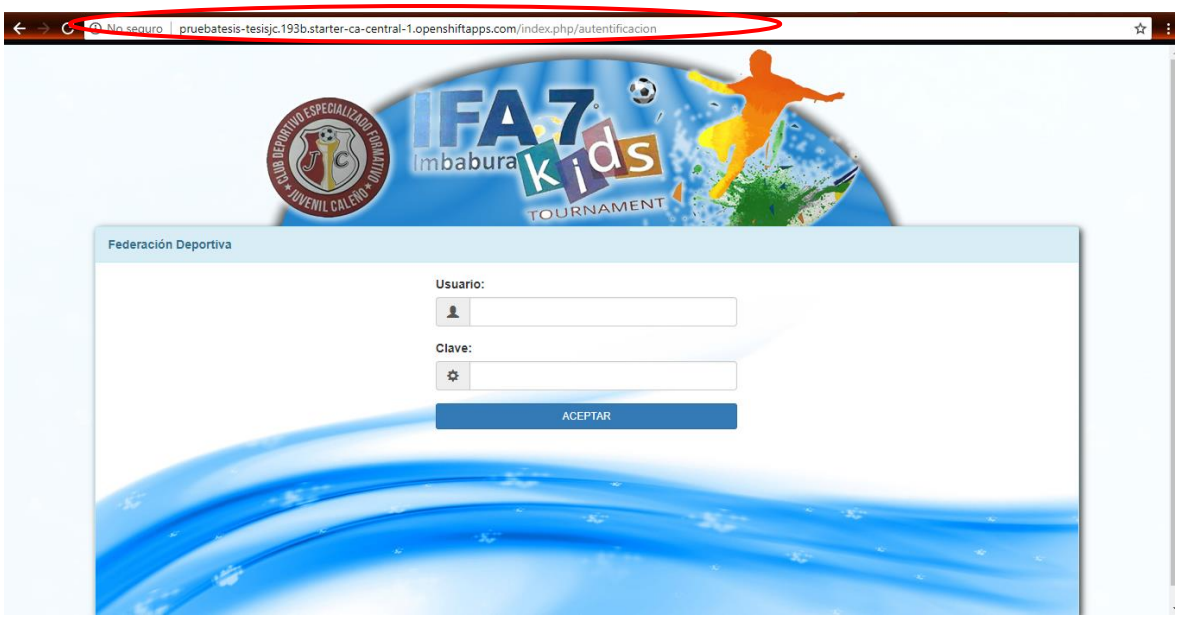


Ilustración 77 Página Principal en Openshift

Fuente: Propia.

Tabla 19 Sprint 3 – Hoja Electrónica

SPRINT: 3 **FECHA INICIO**:18 de May. De 2018			18 - May
		Tareas Pendientes	0
REQUERIMIENTO	**TAREA**	**RESPONSABLE**	**ESTADO**
Seguridad y Control,	Investigación Plataforma OPENSHIFT.	Estalin Trujillo	TERMINADO
	Creación Plataforma.	Estalin Trujillo	TERMINADO
	Implementación de aplicación en la plataforma.	Estalin Trujillo	TERMINADO
	Pruebas de la Aplicación	Estalin Trujillo	TERMINADO

Fuente: Propia

Tabla 20 Sprint 3 - Pizarrón

PENDIENTE	EN PROCESO	TERMINADA
_____	_____	Investigación Plataforma OPENSHIFT.
_____	_____	Creación Plataforma.
_____	_____	Implementación de aplicación en la plataforma.
_____	_____	Pruebas de la aplicación.

Fuente: Propia.

CONCLUSIONES Y RECOMEDACIONES

En el presente capítulo se detalla las conclusiones obtenidas tras el desarrollo de este, las recomendaciones necesarias para que el sistema funcione correctamente.

4.1 Conclusiones.

- Al realizar un sistema deportivo se debe enfocar en la ejecución del programa más que en su diseño.
- El tiempo de respuesta de Rest es mejor que el de Soap.
- Los paquetes enviados en Rest son más rápidos que los enviados en Soap.
- Se logró fundamentar las bases del proyecto, con lo cual se pudo diseñar adecuadamente los diferentes procesos involucrados en el mismo.
- Empleando software de análisis como es Jmeter, se logró obtener los datos estadísticos necesarios para comparar web services.
- Mediante el uso de web services podemos generar soluciones de mejor manera, centradas en la escalabilidad y rendimiento.
- El desarrollo de esta aplicación facilita la difusión de información al club, haciendo uso de las nuevas tecnologías.
- La elección de las herramientas MySQL y CodeIgniter fueron óptimas para desarrollar cada fase del proyecto, ya que estas herramientas se complementas de mejor manera.

4.2 Recomendaciones.

- Se recomienda realizar pruebas de concepto al momento de decidir el tipo de Web service que se va a implementar.
- Existen diferentes softwares para realizar pruebas de web services, pero Jmeter es muy recomendable, porque a parte de ser libre brinda las estadísticas necesarias para comparar servicios, hacer testing, etcétera.
- Se recomienda el uso de web services Rest para la realización del sistema, después de las pruebas realizadas.
- Se recomienda CodeIgniter como librería de apoyo para realizar web services Rest, gracias a su api Rest que facilita la conexión y envió de datos.

- El uso de formato Json agilita el envío y recepción de información en los web services ya sean Rest o Soap.
- Usar metodologías ágiles para desarrollar web service, con el fin de liberar carga en documentación y centrarse en las necesidades del cliente.

BIBLIOGRAFÍA

abc. (1990). *xyz.* iba: anaya.

Brea, O. F. (s.f.). *desarrolloweb*. Obtenido de https://desarrolloweb.com/articulos/1883.php

ecured. (s.f.). *ecured*. Obtenido de https://www.ecured.cu/Protocolo_simple_de_acceso_a_objetos_(SOAP)

Heredia, Á. &. (2011). Comparación y tendencias entre metodologías ágiles y formales. En Á. &. Heredia. Serie Científica de la Universidad de las Ciencias Informáticas.

Marset, R. N. (2006). Modelado, Diseño e Implementación de servicios web. ELP.

Matsumura, M., Brauel, B., & Shah, J. (2009). Adopción de SOA para dummies. En M. Matsumura, B. Brauel, & J. Shah, *Adopción de SOA para dummies* (pág. 3). Estados Unidos: Wiley.

Microsoft. (s.f.). *microsoft*. Obtenido de https://msdn.microsoft.com/es-es/library/cc468318(v=vs.71).aspx

MySQL. (s.f.). Obtenido de https://www.oracle.com/es/mysql/

O'Neill, M. (2003). Web services Security. En M. O'Neill, *Web services Security* (pág. 4). Osborne: McGraw-Hill.

openshift. (s.f.). *Red Hat, Inc.* Obtenido de https://www.openshift.com

postgresql. (s.f.). Obtenido de https://www.postgresql.org/

primefaces. (2009). *primefaces*. Obtenido de https://www.primefaces.org/

Ramesh Nagappan, R. S. (2003). Developing Java Web Services. En R. S. Ramesh Nagappan, *Developing Java Web Services* (pág. 22). Wiley Publishing Inc.

Sutherland, K. S. (2016). La Guía Definitiva de Scrum.

W3C. (s.f.). *W3C*. Obtenido de https://www.w3schools.com/xml/xml_soap.asp

w3c_es. (s.f.). *w3c_es*. Obtenido de http://www.w3c.es