Integrantes:

JOSE CARREÑO CASTILLO

rENATO ESPINOZA CARRANZA

aNDRÉS HUAMAN OLIDEN

VICTOR ROJAS BARBOZA

TB02 Arquitectura de Software

2015

**ÍNDICE DEL DOCUMENTO**

**Consideraciones………………………………………………………………………………………………. 1**

1. **Requerimientos funcionales ……………………………………………………………………… 2**
2. **Backlog ………………………………………………………………………………………………………. 5**
3. **Definición de tecnologías a usar …………………………………………………………………. 7**
4. **Análisis costo y beneficio ……………………………………………………………………………. 8**
5. **Escenario de estilo arquitectónico …………………………………………………………….. 12**
6. **Enfoque Domain Driven Design (DDD) ………………………………………………………. 13**
7. **Enfoque Atributte Driven Design (ADD) ……………………………………………………. 21**
8. **Diagrama arquitectónico modular…………………………………………………………….. 26**
9. **Quality Attribute Workshop (QAW)…………………………………………………………..**
10. **Refinamiento de escenarios de atributos de calidad…………………………………**

**CONSIDERACIONES**

Este trabajo presenta el caso de la empresa QWERTY, especializada en el desarrollo de soluciones de software, el cual plantea la realización de un proyecto para la Municipalidad Metropolitana de Lima. El objetivo de dicho proyecto consiste en la construcción de una plataforma para gestionar el tráfico vehicular en Lima y que se la utilice como punto de encuentro para las instituciones involucradas. El nombre esta plataforma es “QWERTY” al igual que la empresa. Se asume que la parte inicial del proyecto, es decir los aspectos administrativos, ya se encuentran gestionados, por lo que se empezará el trabajo a partir del refinamiento de requerimientos funcionales.

1. **REQUERIMIENTOS FUNCIONALES**

Se han establecido y refinado los siguientes requerimientos funcionales luego de las reuniones con los stakeholders:

* RF01: El sistema debe permitir al usuario tipo pasajero registrar una cuenta para tener acceso.
* RF02: El sistema debe permitir al usuario elegir el tipo de cuenta que usará en la aplicación al momento de registro(Persona natural o empresa, )
* RF03: El sistema debe validar al usuario tipo empresa en el momento de registrarse en la aplicación a través de un código de seguridad que le otorgó la Municipalidad de Lima.
* RF04: El sistema debe permitir al usuario acceder a todos los servicios brindados por la aplicación con un solo inicio de sesión.
* RF05: El sistema debe permitir al usuario empresa o propietario registrar uno o más vehículos en su cuenta.
* RF06: El sistema debe comparar los datos ingresados al momento de registro con la base de datos de la SUNARP para confirmar la autenticidad de los mismos.
* RF07: Al momento de registro el sistema debe permitir al usuario propietario registrar un vehículo bajo los siguientes criterios:
  + Código GPS.
  + Número de placa.
* RF08: El sistema debe permitir al usuario propietario especificar el tipo de vehículo registrado.
* RF09: El sistema debe almacenar los registros de los ingresos de venta de las empresas.
* RF10: El sistema debe permitir al usuario pasajero consultar la ubicación de una unidad de transporte disponible mediante GPS, así como también su capacidad disponible en ese momento.
* RF11: El sistema QWERTY debe permitir al sistema del vehículo poder actualizar la capacidad disponible en cada paradero.
* RF12: El sistema debe obtener información de la localización de cada vehículo cada 5 segundos mediante el sistema GPS.
* RF13: El sistema debe permitir al usuario conductor de taxi reportar la cantidad de pasajeros, el punto de origen y el punto de destino de cada viaje.
* RF14: El sistema debe permitir al usuario empresa recibir un pago por cada servicio realizado.
* RF15: El sistema debe permitir al usuario pasajero realizar un pago por un servicio a través de tarjeta de crédito o móvil.
* RF16: El sistema debe permitir a los usuarios consultar reportes de tráfico de usuarios de la aplicación Waze.
* RF17: El sistema debe permitir al usuario pasajero calificar al encargado de un servicio brindado.
* RF18: El sistema debe permitir al usuario pasajero reportar infracciones a la conducción mediante fotos o videos.
* RF19: El sistema debe permitir al usuario conductor de empresa reportar un abuso por parte del personal PNP, SAT y SUNAT.
* RF20: El sistema debe permitir al usuario pasajero definir un contacto de emergencia.
* RF21: El sistema debe permitir al Contacto de Emergencia hacer seguimiento de la ruta del usuario mediante el sistema GPS.
* RF22: El sistema debe permitir al usuario empresa realizar búsquedas de información en el sistema para efectos de marketing.
* RF23: El sistema debe permitir al usuario editar la información de su cuenta.
* RF24: El sistema QWERTY debe permitir a la SUNAT, PNP, SUNARP poder consumir un servicio para consultar información de reclamos y reportes de infracciones.
* RF25: El sistema debe permitir al usuario empresa poder elegir una ruta de un listado de rutas válidas.
* RF26: El sistema debe permitir al usuario poder registrar una flota de vehículos.
* RF27: El sistema debe permitir al usuario poder asignar un vehículo a una flota determinada.

1. **USER STORY BACKLOG**

Los requerimientos funcionales se han agrupado en un Backlog de user stories

* US01: Como Usuario debo poder registrar una cuenta en el sistema para poder acceder a los servicios de la aplicación.
* US02: Como Usuario debo tener un inicio de sesión único para poder acceder a los servicios brindados por el sistema.
* US03: Como Usuario Natural debo poder inscribirme como un Usuario Propietario para acceder a las funcionalidades del sistema correspondientes al sistema vehicular.
* US04: Como Usuario Natural de poder inscribirme como Usuario Pasajero para acceder a las funcionalidades del sistema que corresponden al servicio de transporte público.
* US05: Como Usuario Propietario debo poder inscribir un vehículo en el sistema para que pueda ser detectado por los Usuarios Pasajeros.
* US06: Como Sistema QWERTY debo poder comparar los datos de una cuenta, al momento de registro, con la base de datos de la SUNARP para confirmar la autenticidad de los mismos.
* US07: Como Usuario propietario de vehículo particular o empresa debo poder registrar uno o más vehículos usando el código del GPS y el número de placa obligatoriamente para que la posición del vehículo pueda ser rastreada.
* US08: Como Usuario propietario de vehículo particular o empresa debo poder especificar el tipo de mis vehículos para verificar qué tipo de funciones puedo desempeñar de acuerdo a este.
* US09: Como Usuario propietario de vehículo particular o empresa debo poder pagar impuestos para simplificar el proceso de recaudación.
* US10: Como usuario pasajero debo poder consultar la ubicación de la unidad de transporte requerida, así como también su capacidad disponible para poder llegar pronto a mi centro de labores.
* US11: Como usuario conductor de empresa debo poder actualizar la información de la capacidad disponible de mi vehículo en cada paradero.
* US12: Como usuario sistema QWERTY debo poder obtener la información de la localización de cada vehículo cada 5 segundos a través del sistema GPS.
* US13: Como usuario conductor de empresa debo poder reportar la cantidad de pasajeros, el punto de origen y el punto de destino de un viaje cada vez que realice algún tipo de servicio.
* US14: Como usuario empresa debo poder recibir el pago por el servicio realizado para realizar un balance de los ingresos.
* US15: Como usuario pasajero debo poder realizar un pago a la empresa a través de tarjetas de crédito o móvil para poder hacer uso del servicio.
* US16: Como sistema QWERTY debo poder recopilar información de tráfico de usuarios Waze para gestionar el tráfico vehicular.
* US17: Como usuario pasajero debo poder calificar el servicio brindado por transporte público y reportar infracciones a la conducción y estacionamiento mediante fotos o videos para que sean aplicadas las multas respectivas a los infractores.
* US18: Como usuario propietario de vehículo y conductor empresa debo poder reportar abusos por parte del personal PNP, SAT y SUNAT para agilizar el proceso de denuncia.
* US19: Como usuario pasajero, propietario de vehículo y conductor de empresa debo poder definir un contacto de emergencia para que éste pueda hacer seguimiento en línea de la ruta seguida.
* US20: Como usuario consultor debo poder realizar búsquedas de información en el sistema para efectos de marketing.
* US20: Como usuario PNP, debo visualizar todas las alertas registradas en el sistema para poder atender las emergencias de manera eficaz.
* US21: Como usuario SUNAT debo visualizar los cobros realizados por los usuarios empresa para poder facilitar la fiscalización.

1. **DEFINICIÓN DE TECNOLOGIAS A USAR**

* **Lenguaje de programación:**

**Java**: Java es un lenguaje de programación de alto nivel desarrollado por Sun MicroSystem. Fue diseñado inicialmente para operar dispositivos de comunicaciones. Java es un lenguaje orientado a objetos que corrigió los errores comunes de los primeros lenguajes orientado a objetos como el C++. Java, al ser un lenguaje de propósito general, puede ser empleado también en el desarrollo de aplicaciones web. Si bien java no es tan popular como ruby o phyton en cuanto a la construcción de aplicaciones web, el desarrollo de la tecnologías comoSpring y Play framework han impulsado su popularidad debido a que los programadores de Java no tenían la necesidad de aprender un nuevo lenguaje para crear sus aplicaciones web. Play Framework le ha dado un nuevo impulso a Java en el desarrollo de aplicaciones web ya que tiene muchas de las ventajas de los lenguajes orientados netamente al desarrollo web.

* **Servidor de aplicaciones:**

**Weblogic:** Es un servidor de aplicaciones Java EE , así mismo puede ser empleado como contenedor de aplicaciones web HTTP, este servidor puede ejecutarse en diferentes plataformas como Unix, Linux, Microsoft Windows y otras plataformas. Weblogic puede soportar gestores de base de datos como Oracle, SQL server, DB2, entre otros. Es compatible con los estándares de Java Enterprise Edition. Además, Weblogic permite la interoperabilidad con frameworks como .Net. Por otro lado, permite la integración con tecnologías de mensajería y conectividad: JMS, CORBA, IBM WebSphere MQ.

* **Framework de Backend:**

**SPRING-DATA-PROJECT:** Provee un sofisticado soporte para las tecnologías de acceso de datos tradicionales. Simplifica significativamente la implementación del acceso a la capa de datos. Así mismo, permite independizarse de los frameworks de persistencia como JDBC, Hibernate, Mybatis, etc. Este framework ha madurado con el pasar de los años por ello las últimas versiones son robustas. El proyecto Spring Data ha desarrollado módulos para el soporte de manejo de base de datos NoSQL.

* **Servicios Web:**

**REST:** Es un estilo de software arquitectural basada en directivas y buenas prácticas para la creación de servicios web escalables. Rest es un conjunto coordinado de directrices aplicadas al diseño de componentes distribuidos de sistemas de hypermedia que conducen hacia una arquitectura mantenible y de alto rendimiento. Rest ha ganado aceptación al ser considerado una alternativa para el uso de servicios web como Soap y Wsdl.

**SOAP:** Es un protocolo estándar que define cómo dos objetos pueden comunicarse por medio de intercambio de datos XML. Este protocolo proporciona una implementación estándar de integridad de y privacidad de datos. Además, proporciona fiabilidad en un sistema de integración de mensajería instantánea.

* **Base de Datos:**

**Oracle:** Es un sistema de gestión de base de datos objeto-relacional. Este sistema se destaca por el soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Ha sido diseñada para que las organizaciones puedan controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir los costes y los riesgos asociados a la pérdida de información.

1. **ANALISIS DE COSTO Y BENEFICIO**

Con el objetivo de analizar los costos y beneficios de todos los componentes que se deberán evaluar en el desarrollo del proyecto, primero debemos tomar conciencia del alcance real del mismo, para esto, detallaremos diferentes datos estadísticos acerca de todos los involucrados dentro del proyecto.

En primer lugar, Lima cuenta con una población aproximadamente 9 millones de habitantes. Además cuenta con 30 mil taxis que circulan diariamente en toda la ciudad.

En segundo lugar, en los últimos 8 años, el número de autos que circular por Lima se han duplicado alcanzando 1 millón 200 mil vehículos en toda la ciudad y de 2 millones 200 mil vehículos en todo el país.

En tercer lugar, según la encuesta ONG Lima Cómo Vamos, se señaló que 8 de cada 10 limeños usan el transporte público y que solo un 7.6% de los habitantes emplea auto propio.

Esto quiere decir que debemos contar con un sistema que cuente con las tecnologías adecuadas y que pueda darse abasto para procesar toda la información necesaria para poder cumplir con todas las necesidades de manera eficiente.

**Análisis de costo y beneficio: Oracle vs SqlServer**

Tomando en cuenta que el sistema debe manejar grandes cantidades de datos se ha optado por un sistema robusto que fue desarrollado para el sector empresarial en el cual la confiabilidad es un factor importante a tener en cuenta. Oracle ha sido diseñado para ser un gestor de base de datos amigable para el DBA. A continuación presentamos una comparación de costos entre los motores de base de datos SqlServer y Oracle. Si bien las ventajas que posee Oracle superan 10 veces a otros motores de base de datos en cuanto a características y funcionalidades para resolver problemas complejos del manejo de datos, también ofrece ventajas en cuanto a la relación costo-beneficio. Es así que con un costo superior en 3 veces que SQL server podemos tener un gestor de base de datos superior a muchos otros.



*Fig. 1: Comparación de costos de Oracle DataBase y SQL Server 2005*

**Análisis de costo y beneficio : weblogic vs Jboss**

Si bien inicialmente Weblogic server es más costoso que Jboss debido a su costo de licencia, luego, al hacer un comparativo de costo de cinco años, podemos concluir que:

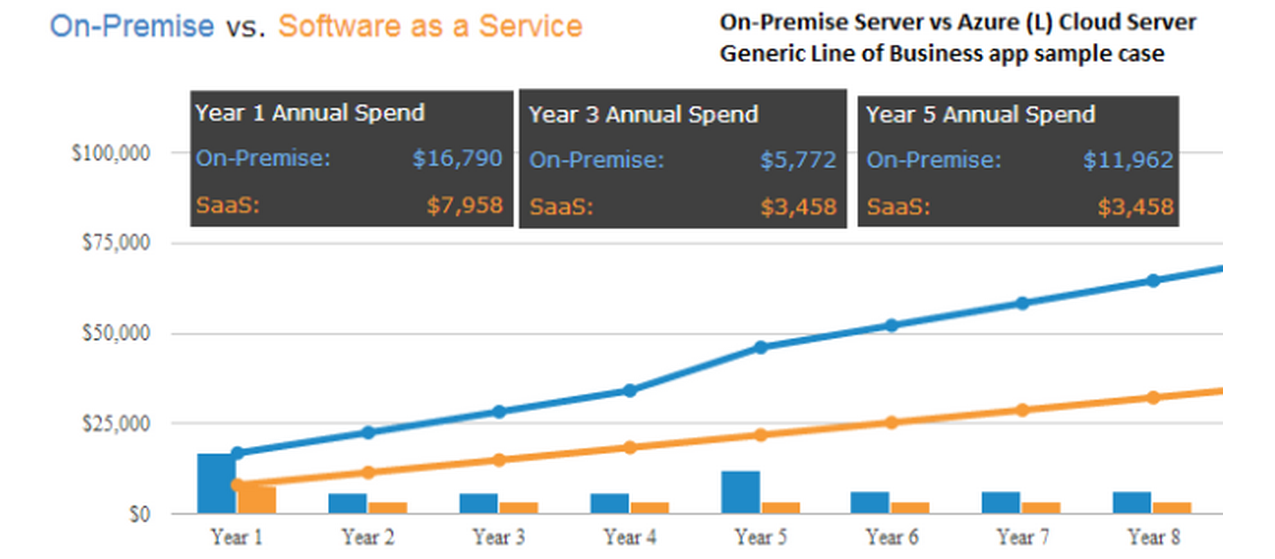
*Fig. 2: Comparación de costos en 5 años entre WebLogic y JBoss*

Jboss es 35% más costo que Weblogic a pesar de su bajo costo de licencia.

Después de los dos años de adquisición se hace menos costoso el uso de Weblogic, lo cual es una ventaja que continua con el paso de los años. Otros aspectos como el rendimiento, valor en el tiempo y configuración de la infraestructura, puede tener un impacto significativo en el rendimiento de la inversión global del negocio en cuanto a la implementación del servidor de aplicaciones.

**Análisis de costo y beneficio: Servidor local vs Servidor remoto**

Existe una tendencia marcaren el uso de la nube en cuanto al uso de servidores de aplicaciones y web que indica que los servidores, especialmente web, optan por el uso de servidores remotos. Los gastos en que se incurren al decidir por un servidor local es una de las desventajas más importantes. Por ello, para el proyecto en cuestión se ha optado por el uso del desarrollo en la nube. En la siguiente imagen, se pude observar un cuadro comparativo de costos en que se incurren en el uso de ambas opciones.



*Fig. 3: Comparación de costos entre un servidor local y un servidor en la nube*

Algunas ventajas del desarrollo en la nube:

* Los gastos iniciales son considerablemente menores.
* No se requieren técnicos expertos, ya que el data center cuenta con técnicos capacitados.
* La opción de poder elegir y configurar los recursos de hardware en el momento que se decida brinda una sostenibilidad al proyecto, es decir, poder crecer en recursos en la misma medida que la aplicación lo requiera.

**Análisis de costo y beneficio: Spring data**

Spring data ofrece menor costo de implementación a diferencia de Hibernate pues simplifica en gran medida la construcción de la capa de acceso a datos. Esto se debe a que SpringData JPA es una especificación mientras que Hibernate es una implementación particular de esa implementación. SpringData proporciona soporte para JDBC, JPA, JDO, Mybatis y el mismo Hibernate, unificando la configuración y creando una jerarquía de excepciones común para todos ellos.

1. **ESCENARIO DE ESTILO ARQUITECTÓNICO**

El estilo arquitectónico escogido para este proyecto es la arquitectura orientada a servicios (SOA). Este estilo consiste en que los componentes de una aplicación proveen servicios hacia otros componentes a través de protocolos de comunicación (contratos). El principio base de este estilo es la independencia de cualquier proveedor de un producto o tecnología, lo cual permite que los empleados y socios comerciales se adapten fácilmente a las tecnologías de información cambiantes.

Como su nombre lo indica, SOA utiliza como componente principal a los servicios. Un servicio es un componente de software distribuido que expone funcionalidad de alto valor para el negocio. Las principales características de un servicio son:

* Contrato estandarizado: Entre proveedor y consumidor.
* Bajo acoplamiento: Uso de mensajes y orquestación.
* Abstracción: El servicio funciona como una caja negra.
* Reusabilidad: El servicio es accesible a través de un directorio.
* Autonomía: El servicio no debería tener dependencias con otro servicio externo.

¿Por qué SOA?

La Municipalidad Metropolitana de Lima necesita que la información esté ampliamente distribuida. Para esto se estructurará la lógica de negocio en servicios para que puedan ser accedidos desde cualquier parte.

El negocio principal de la organización no gira alrededor de un alto volumen de transacciones sincrónicas en tiempo real.

Uno de los objetivos organizacionales es alinear el negocio con la funcionalidad de la tecnología de información. Para esto SOA elimina la redundancia, es decir que permite que el consumidor del servicio seleccione otros proveedores alternativos.

La organización es grande y opera con una red heterogénea que consume servicios de diferentes proveedores y entrega funcionalidad a diferentes consumidores lo cual evita la amenaza de la dependencia de un solo proveedor.

A continuación el diagrama del estilo arquitectónico propuesto para el proyecto.



*Fig. 4: Diagrama de estilo arquitectónico SOA propuesto*

1. **ENFOQUE DOMAIN DRIVEN DESIGN (DDD)**

DDD es una colección de principios que ayudará a los desarrolladores a construir los objetos del sistema. Un modelo de dominio es una abstracción del software que encapsula lógica de negocio compleja y relaciona la abstracción con la realidad. A continuación se presentarán los modelos de dominio relevantes encontrados.

* Registro de Usuario

Se han definido dos tipos de usuario al momento del registro: natural y empresa. El usuario natural accederá al módulo de registro a través de una aplicación móvil o un sitio web. Luego deberá ingresar sus datos personales (nombre, dirección, correo electrónico). El sistema guardará dicha información y mostrará un mensaje de confirmación. El registro del usuario empresa tiene una mayor complejidad, pues se debe verificar la información de la empresa y los premisos de circulación en las diferentes rutas vehiculares. La verificación se realiza de manera asíncrona y, una vez terminada, el sistema agrega las rutas a una base de datos y las relaciona a la cuenta del usuario empresa.

Asíncrono

Síncrono

**Municipalidad**

Usuario

**LB**

Consume servicio para validar y rutas y permisos

*Fig. 5: Modelo de dominio registro de usuario*

* Editar usuario

Los usuarios podrán editar información de su cuenta como datos personales o datos de la empresa. Al mismo tiempo, un usuario natural podrá designar a otros como contactos de emergencia. La función del contacto de emergencias se explica más adelante en el dominio de consulta de rutas.

Actualiza un objeto usuario del repositorio de usuarios

**Repositorio de usuarios**

**LB**

Usuario

*Fig. 6: Modelo de dominio editar usuario*

* Vehículo

Los usuarios tendrán que registrar los vehículos que posean. Se guardará la placa del vehículo y el código GPS del mismo. Se verificarán los datos y el tipo de vehículo a través del consumo de la SUNARP de manera asíncrona. Los usuario tipo empresa podrán, adicionalmente, crear flotas de vehículos y asignar vehículos de tipo bus a estas flotas. Además, a una flota se le podrá fijar una ruta de circulación fija.

**LB**

**SUNARP**

Usuario

Síncrono

Asíncrono

Consume servicio para validar datos de vehículo

*Fig. 7: Modelo de dominio vehículo*

* Consulta de rutas

Los usuarios tipo pasajero podrán suscribirse a las rutas que deseen. Esto les permitirá ver la posición y el detalle de los vehículos que circula por las rutas seleccionadas. Así mismo, un usuario natural puede ser designado como contacto de emergencia, por lo que podrá visualizar en todo momento la ruta que sigue el usuario que lo eligió. Para lograr esto, el sistema de los vehículos debe actualizar su posición y la cantidad de pasajeros cada 5 segundos y enviar esta información al sistema QWERTY. Si el tipo de vehículo es taxi, además se deberá actualizar el punto de destino cada vez que sea necesario. El sistema QWERTY luego se encargará de “anunciar” la nueva información y “empujar” la data actualizada a todas las cuentas suscritas. Por otro lado, se consumirá un servicio de Waze para mostrar reportes del tráfico en la ciudad.

Envía datos de la localización y capacidad del vehículo

Sistema vehículo

Publisher

Sistema GPS

Usuario suscrito

Suscriber

Envía información del tráfico

Waze web service

Infraestructura de propagación de cambios

*Fig. 8: Modelo de dominio de consulta de rutas*

* Asignación de rutas

Los usuarios empresa que posean vehículos de tipo bus deberán asignar sus rutas (obtenidas del repositorio de rutas) previamente y los vehículos tendrán que seguir dicha ruta. En cambio, los usuarios conductores de taxi deberán actualizar su ruta (punto de partida y llegada) cada vez que brinden sus servicios a un nuevo cliente.

Usuario conductor de Taxi

Repositorio De Detalle de vehículo

Envía datos de ruta y pasajeros

Actualiza Rutas

Consulta Rutas

Repositorio de Rutas

Usuario Empresa

*Fig. 9: Modelo de dominio de asignación de rutas*

* Calificación y denuncias

Los usuarios podrán calificar el servicio brindado por las unidades de transporte público y reportar infracciones y abusos de autoridad. También podrán adjuntar fotos y videos como pruebas que sustenten su denuncia. Toda esta información será almacenada en un repositorio de reportes. Las instituciones interesadas podrán acceder a la información de los reportes a través de un servicio web.

Califica y guarda reportes

Usuario

**LB**

Genera reportes

Repositorio de reportes

Consume el servicio de reportes

Instituciones

Obtiene reportes

*Fig. 10: Modelo de dominio de calificación y denuncias*

* Pagos y cobranzas

Al momento de ingresar a un vehículo que brinde servicios de transporte público, los usuarios deberán registrar el pago a través de los sistemas externos VISA y MasterCard. Estas entidades a su vez, implementarán un servicio de reporte de pagos y cobranzas que podrá ser consumido por el sistema QWERTY. QWERTY, al mismo tiempo, expondrá un servicio que brinda reportes de pago y ventas y serán consumidos por SUNAT.

Usuario realiza pago

Sistemas externos

VISA

MasterCard

QWERTY consume servicio

SUNAT consume servicio

Sistema externo

SUNAT

*Fig. 10: Modelo de dominio de pagos y cobranzas*

1. **ENFOQUE ATTRIBUTE DRIVEN DESIGN (ADD)**

El enfoque ADD es un método utilizado para definir una arquitectura de software basado en los atributos de calidad del software. Para la realización del proceso ADD son necesarias tres materias primas: los requerimientos funcionales, las restricciones de diseño y los requerimientos de atributos de calidad. Las dos primeras materias primas se han visto en este documento en los puntos 1, 2 y 3. Por lo tanto, como primer paso se verán los escenarios de atributos de calidad encontrados ordenados en tablas.

* Registro de usuario

*Availability*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Usuario realizando un registro de cuenta. |
| Fuente del estímulo | Cliente que será un Nuevo usuario |
| Entorno | Sistema bajo condiciones de sobrecarga. |
| Artefacto | Validar Usuario Empresa |
| Respuesta | Los usuarios no pierden la información del registro y el error es trasparente para el usuario. |
| Métrica | El nodo secundario toma el control y se reestablece la conexión en 5 segundos. |

*Performance*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Usuario realizando un registro de cuenta. |
| Fuente del estímulo | Cliente que será un nuevo usuario. |
| Entorno | Sistema bajo condiciones normales de uso. |
| Artefacto | Módulo de Registro de Usuario. |
| Respuesta | El usuario es registrado en el sistema satisfactoriamente y se muestra un mensaje de éxito al cliente. |
| Métrica | El usuario se registra en el sistema en un tiempo promedio de 3 segundos. |

* Contacto de emergencia

*Usabilidad*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Usuario realiza búsqueda del contacto de emergencia. |
| Fuente del estímulo | Usuario registrado como contacto de emergencia. |
| Entorno | Sistema bajo condiciones normales de uso. |
| Artefacto | Seguimiento de Rutas |
| Respuesta | Se muestra la ubicación y la ruta que está siguiendo el pasajero. |
| Métrica | El usuario visualiza la ruta del pasajero en un tiempo promedio de 8 segundos. |

* Editar usuario

*Modificabilidad*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Actualización en el módulo de usuario |
| Fuente del estímulo | Se quiere agregar una nueva funcionalidad de perfil al usuario |
| Entorno | Sistema bajo condiciones normales de uso |
| Artefacto | Módulo de registro de usuario y editar usuario |
| Respuesta | El sistema es capaz de someterse a un cambio sin afectar a otros módulos para mantener la integración |
| Métrica | El costo de del cambio es el menor posible |

*Usabilidad*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | El usuario quiere modificar los datos de su perfil |
| Fuente del estímulo | Usuario Final |
| Entorno | Sistema en condiciones normales |
| Artefacto | Editar datos de usuario |
| Respuesta | El usuario efectúa la operación de manera eficaz lo realiza de manera intuitiva |
| Métrica | El usuario se demora un promedio de dos minutos en realizar todos los pasos para editar los datos de su cuenta |

* Registro de vehículo

*Testeabilidad Registro de Vehículo*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Ejecución de pruebas unitarias |
| Fuente del estímulo | Se desean realizar pruebas unitarias al sistema para evaluar su calidad c |
| Entorno | El sistema se encuentra en mantenimiento |
| Artefacto | Módulo de registro de vehículo |
| Respuesta | Captura de los resultado de las pruebas unitarias |
| Métrica | El porcentaje del código cubierto en las pruebas de cobertura |

* Calificación y denuncias

*Disponibilidad*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | El sistema falla al enviar un reporte de infracción y denuncia |
| Fuente del estímulo | Usuario que envía el reporte |
| Entorno | Sistema bajo condiciones normales de uso |
| Artefacto | Reportar infracción |
| Respuesta | Se muestra un mensaje de error y los datos descritos en el reporte no se pierden |
| Métrica | El sistema identifica un nodo secundario y se recupera a los 5 segundos |

*Performance*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Una institución quiere hacer una consulta de las demandas que se han sido imputadas. |
| Fuente del estímulo | Se quiere consumir el servicio de consultar los reportes de infracción |
| Entorno | Sistema bajo condiciones normales de uso |
| Artefacto | Servicio de Consultas de Reportes |
| Respuesta | La petición es procesada y los reportes son adquiridos por la institución |
| Métrica | Tiempo promedio de latencia de unos 3 segundos |

* Consulta de rutas

*Modificabilidad*

|  |  |
| --- | --- |
| Elemento | Descripción |
| Estímulo | Se quiere agregar una funcionalidad al usuario de poder consultar el tiempo promedio de llegada de un bus a un determinado paradero |
| Fuente del estímulo | Los desarrolladores consideran relevante la implementación de dicha característica |
| Entorno | Sistema en mantenimiento |
| Artefacto | Core del Sistema |
| Respuesta | Los cambios son añadidos al sistema de manera eficiente |
| Métrica | Se evalúa el costo económico, de tiempo y de esfuerzo en implementar dicha característica |

El siguiente paso del enfoque ADD es elegir los candidatos a drivers de arquitectura. Para ello, se ha elaborado una tabla que refleja la importancia de las materias primas para los stakeholders vs el impacto que estas producirían en la arquitectura del software.

Importancia para los stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
|  | High | Medium | Low |
| High | Consulta de rutas  Registro de usuario  Registro de rutas  RF04 Inicio de sesión | Editar usuario |  |
| Medium | Contacto de emergencia  Pagos y cobranzas | Calificación y denuncias |  |
| Low | Registro de vehículos |  |  |

Impacto en la arquitectura

El primer candidato a driver de arquitectura y core del sistema es la consulta de rutas. El hecho de actualizar la posición de todos los vehículos registrados en el sistema (más de un millón) cada 5 segundos causa un impacto muy alto en la arquitectura del software. Se deberá tener en cuenta la implementación de balanceadores de carga (concurrencia) y otro hardware que asegure un alto rendimiento de este módulo. El login e inicio de sesión también supondrán una alta carga para el sistema. Recordemos que Lima es una ciudad con 9 millones de habitantes, por lo que el sistema debe estar preparado para soportar el intento de acceso por varios usuarios al mismo tiempo. En ese sentido, el registro de rutas por parte de los usuarios conductores de taxi es candidato a driver, ya que se realizan miles de viajes nuevos en taxis diariamente y el sistema deberá soportarlo.

1. **DIAGRAMA ARQUITECTÓNICO MODULAR**

A partir de los enfoques DDD y ADD se ha elaborado el siguiente diagrama de arquitectura tentativo. Se ha usado una arquitectura orientada a servicios cómo patrón principal de arquitectura y para el despliegue se ha elegido el patrón N-tier. La funcionalidad core del sistema -consulta de rutas- utiliza una infraestructura de distribución publisher – suscriber. Además para mejorar la concurrencia se implementará el patrón half - sync / half – async en la mayoría de los servicios de validación de datos como SUNARP o Municipalidad de Lima.

Los datos procesados y almacenados persisten en los repositorios

Servicios del sistema

Aplicación

Editar Usuario

Pagos y Cobranzas

Calificación y denuncias

Log In

Registrar Usuario

Core - Ubicación

Registro de Vehículo

Controlador

Vistas

Cliente Web

Controlador

Vistas

Cliente Movil

Presentación

Proveedor de servicios

Dominios

En los dominios residen las reglas y la lógica de negocio del sistema

Infraestructura

Repositorios de dominios

Servicios Externos

Repositorio de Detalle de Vehículo

Repositorio de Usuarios

Repositorio de Reportes

Repositorio de Rutas

Repositorio de Calificación y denuncias

Waze Service

Algunas funcionalidades de los dominios consumen el servicio de Waze

*Fig. 11: Diagrama arquitectónico modular propuesto*

1. **Quality Attribute Workshop.**

El QAW provee un método para identificar los atributos de calidad críticos de la arquitectura de software de un sistema, tales como la disponibilidad, desempeño, seguridad, interoperabilidad y modificabilidad, que son extraídos de las metas de negocio.

El QAW consiste en una serie de reuniones entre los stakeholders del proyecto, durante las cuales se generaran los escenarios de atributos de calidad requeridos, luego serán priorizados, y por último refinados. El proceso de refinamiento de los escenarios le permite a los stakeholders comunicarse entre sí, de esta manera se exponen supuestos que pueden no haber sido mencionados con anterioridad en la obtención de requerimientos.

El beneficio que este método proporciona es determinar las cualidades correctas y necesarias para el sistema antes de que este sea desarrollado. La detección de estos requisitos y el hecho de que sean logrados en la primera versión del sistema ahorran dinero y evitan una futura reestructuración del software. El QAW provee una configuración estructurada y eficiente para comunicarse con los stakeholders y favorece al análisis y el testeo a lo largo de la vida del sistema.

Este proceso consiste de los siguientes pasos:

1. **Presentación e introducción del QAW:** En este paso se describe la motivación del QAW y se explica cada uno de los pasos a realizar en las reuniones de stakeholders.
2. **Presentación de negocio:** Un representante de la comunidad de stakeholders presenta el negocio y los drivers programáticos para el sistema.
3. **Presentación del plan arquitectural:** Un representante técnico explica los planes arquitecturales del sistema. Asimismo, se exponen los requisitos técinos y limitaciones que afectarán a la arquitectura del software.
4. **Identificación de los drivers arquitecturales:** Se compartirá la lista de drivers arquitectónicos clave y se le pedirá a los stakeholders las aclaraciones, adiciones, supresiones y correciones necesarias.
5. **Lluvia de ideas para los escenarios:** Cada stakeholder expondrá un escenario que representa una necesidad que tiene que ser satisfecha por el sistema. Lo requerido es que exista por lo menos un escenario para cada driver arquitectural.
6. **Consolidación de los escenarios:** Se mezclan escenarios con contenidos similares para evitar que los votos de priorización se diluyan
7. **Priorización de los escenarios:** Se efectúan dos rondas de votación en los cuales los stakeholders decidirán cuáles son los escenarios más importantes.
8. **Refinamiento de los escenarios:** Los escenarios escogidos en el paso anterior serán refinados agregándoles los siguientes puntos:
   1. Las metas del negocio que son afectadas por esos escenarios.
   2. Los atributos de calidad relevantes asociados a esos escenarios.

**Refinamiento de escenarios del proyecto.**

|  |  |
| --- | --- |
| Escenario de refinamiento: Registro de usuario - Disponibilidad | |
| Escenario | Un futuro usuario desea registrarse en la aplicación durante un estado de sobrecarga en el sistema. Uno de los nodos de éste falla y el secundario toma el control y reestablece la conexión en 5 segundos, permitiendo que el usuario termine el registro satisfactoriamente. |
| Metas de negocio | Sistema con alta disponibilidad – Disponible 24/7 |
| Atributos de calidad relevantes | Disponibilidad |
| Estímulo | Futuro usuario realizando un registro de cuenta. |
| Fuente del estímulo | Futuro usuario de la aplicación. |
| Entorno | Sistema bajo condiciones de sobrecarga. |
| Artefacto | Validar Usuario Empresa |
| Respuesta | El usuario no pierde la información del registro y el error es transparente. |
| Métrica | 5 segundos. |
| Preguntas | ¿Qué ocurriría si el sistema falla justo al momento de confirmar el registro? |
| Consideraciones | En caso de que ocurra una falla en el momento de confirmación del registro se debería mostrar un mensaje de error pidiendo al usuario que intente de nuevo en la misma pantalla sin perder los datos del registro. |

**Tácticas a implementar:**

* Redundancia pasiva: Se configurará el servidor de aplicaciones de WebLogic usando la opción High Availability con un clúster activo pasivo.
* Spare: Se configurará un nodo auxiliar que estará desactivado hasta que el nodo principal deje de funcionar.
* Manejo de excepciones: Se implementará una clase utilitaria de manejo de excepciones para tener información detallada de los errores o fallas que ocurran en el sistema.

|  |  |
| --- | --- |
| Escenario de refinamiento: Consulta de Rutas – Performance | |
| Escenario | Un usuario requiere consultar la ubicación de un vehículo y su respectiva ruta mediante la aplicación y que este actualice la posición en mínimo de 5 segundos. |
| Metas de negocio | Sistema responsive |
| Atributos de calidad relevantes | Performance |
| Estímulo | El usuario desea consultar la posición y la ruta de los vehículos en la aplicación |
| Fuente del estímulo | Usuario Final |
| Entorno | Sistema bajo condiciones de sobrecarga. |
| Artefacto | Módulo de Consulta de Rutas |
| Respuesta | El sistema deberá actualizar la información de la función en un máximo de 5 segundos. |
| Métrica | El tiempo de respuesta de la aplicación es un máximo de 5 segundos. |
| Preguntas | ¿Qué sucede si el usuario desea revisar la información de las rutas y la posición de los buses en hora pico? |
| Consideraciones | En el caso de que el sistema se encuentre muy sobrecargado se deberá controlar la frecuencia de muestreo. Consiste en un bajo monitoreo de las solicitudes encoladas del sistema. Otra opción sería controlar las solicitudes mediante el uso de colas para mejorar el rendimiento de los recursos del sistema. |

**Tácticas a implementar:**

* **Control de demanda de recursos - *Bound queue sizes***: Para implementar esta táctica se usará el servicio de colas distribuidas de WebLogic para controlar la máxima cantidad de solicitudes que pueden ser procesadas de manera concurrente para lograr un uso eficiente de los recursos del sistema.

|  |  |
| --- | --- |
| Escenario de refinamiento: Contacto de emergencia - Usabilidad | |
| Escenario | Cuando un contacto de emergencia solicita la ubicación del usuario se muestra su ubicación y la ruta que está siguiendo en 10 segundos. |
| Metas de negocio | Sistema intuitivo y usable. |
| Atributos de calidad relevantes | Usabilidad |
| Estímulo | Contacto de emergencia realiza búsqueda del usuario. |
| Fuente del estímulo | Usuario registrado como contacto de emergencia. |
| Entorno | Sistema bajo condiciones normales de uso. |
| Artefacto | Seguimiento de Rutas |
| Respuesta | Se muestra la ubicación y la ruta que está siguiendo el pasajero. |
| Métrica | 10 segundos. |
| Preguntas | ¿Se mostrará el desplazamiento del Usuario en tiempo real o mediante estados cada cierto tiempo? ¿La funcionalidad funcionará en background? |
| Consideraciones | Se debería notificar al Usuario cuando estén haciendo seguimiento de su ruta. |

**Tácticas a implementar:**

* **Tiempo de diseño:** Este tipo de táctica consiste en decisiones de diseño que permitan al usuario interactuar con el sistema. Para este caso se implementara el patrón MVC con el framework de Spring para poder separar la interfaz de usuario de los otros módulos de la aplicación como la lógica de negocio y el acceso a datos. Así, el usuario solo podrá disponer del módulo que le corresponde.

|  |  |  |
| --- | --- | --- |
| Escenario de refinamiento: Registro de usuario - Performance | | |
| Escenario | Cuando un usuario desea registrarse en la aplicación y el sistema realiza la operación en un tiempo promedio de 2 segundos. | |
| Metas de negocio | Sistema responsivo. | |
| Atributos de calidad relevantes | Performance | |
| Estímulo | El cliente desea realizar un registro de cuenta en la aplicación. | |
| Fuente del estímulo | Cliente que será un nuevo usuario. | |
| Entorno | Sistema bajo condiciones normales de uso. | |
| Artefacto | Módulo de Registro de Usuario. | |
| Respuesta | El usuario es registrado en el sistema satisfactoriamente y se muestra un mensaje de éxito al cliente. | |
| Métrica | 2 segundos. | |
| Preguntas | ¿Cómo verifica el usuario de que su registro ha sido exitoso? ¿Es posible que un usuario pueda registrar más de una cuenta a su nombre? | |
| Consideraciones | Se permitirá un solo registro de cuenta por DNI de usuario. | |
| Escenario de refinamiento: Consulta de rutas - Modificabilidad | | |
| Escenario | | Se quiere agregar una funcionalidad al usuario para que este pueda consultar el tiempo promedio de llegada de un bus a un determinado paradero y que la implementación se realice con el menor costo posible. |
| Metas de negocio | | Sistema consistente y modificable. |
| Atributos de calidad relevantes | | Modificabilidad |
| Estímulo | | Se quiere agregar una funcionalidad al usuario para que este pueda consultar el tiempo promedio de llegada de un bus a un determinado paradero. |
| Fuente del estímulo | | Los desarrolladores consideran relevante la implementación de dicha característica |
| Entorno | | Sistema en mantenimiento |
| Artefacto | | Core del Sistema |
| Respuesta | | Los cambios son añadidos al sistema de manera eficiente |
| Métrica | | Se evalúa el costo económico, de tiempo y de esfuerzo en implementar dicha característica |
| Preguntas | | ¿A cuántos módulos afectará este cambio? ¿Cuál es el menor costo posible para realizar el cambio? |
| Consideraciones | | Se debería, en lo posible, realizar los cambios sin afectar a los diferentes módulos para reducir el costo de la implementación de la nueva funcionalidad. |

**Tácticas a implementar:**

* Estas tácticas tienen como objetivo controlar el tiempo y el costo de cambios en implementación, pruebas y ejecución. Una de las maneras en la que se pueden lograr estos objetivos sería manteniendo una coherencia semántica. Esto se refiere a la cohesión y el acoplamiento entre los componentes del software de la aplicación. La idea es asegurar un que un componente con responsabilidades comunes no tenga que depender demasiado de otros componentes, así el costo del cambio se reduce ya que sólo se afecta a un módulo. Spring permite implementar este tipo de tácticas mediante el uso del patrón MVC y además mediante el uso de inversión de control (IoS), esto quiere decir que se crea un contenedor que se encarga de crear dependencia a los componentes. Esto hace más sencillo la modificación de un componente específico sin la necesidad de realizar cambios en varios componentes que estén relacionados, esto permite minimizar el impacto y reducir el costo.

|  |  |
| --- | --- |
| Escenario de refinamiento: Editar usuario - Modificabilidad | |
| Escenario | Cuando requiere realizar un cambio o añadir una funcionalidad al sistema en el módulo del usuario y el costo de dicho cambio es el menor posible. |
| Metas de negocio | Sistema consistente y modificable |
| Atributos de calidad relevantes | Modificabilidad |
| Estímulo | Actualización de funcionalidades en el módulo de usuario. |
| Fuente del estímulo | Se requiere agregar una nueva funcionalidad de perfil al usuario. |
| Entorno | Sistema en mantenimiento. |
| Artefacto | Módulo de registro de usuario y editar usuario |
| Respuesta | El sistema es capaz de someterse al cambio sin afectar a los otros módulos manteniendo la integración. |
| Métrica | El costo del cambio es el menor posible. |
| Preguntas | ¿A cuántos módulos afectará este cambio? ¿Cuál es el menor costo posible para realizar el cambio? |
| Consideraciones | Se debería, en lo posible, realizar los cambios sin afectar a los diferentes módulos para reducir el costo de la implementación de la nueva funcionalidad. |

|  |  |
| --- | --- |
| Escenario de refinamiento: Editar Usuario - Usabilidad | |
| Escenario | El usuario quiere modificar la información de su perfil mientras el sistema está en condiciones normales de uso y logra acceder a dicha funcionalidad y realizar la tarea en 2 minutos. |
| Metas de negocio | Sistema intuitivo y usable. |
| Atributos de calidad relevantes | Usabilidad |
| Estímulo | El usuario quiere modificar los datos de su perfil. |
| Fuente del estímulo | Usuario Final |
| Entorno | Sistema bajo condiciones normales de uso. |
| Artefacto | Editar datos de usuario |
| Respuesta | El usuario efectúa la operación de manera eficaz e intuitiva. |
| Métrica | 2 minutos. |
| Preguntas | ¿Cuáles datos pueden ser modificados por el usuario? ¿El usuario puede tener más de un vehículo registrado a su cuenta? ¿El usuario puede eliminar su cuenta? |
| Consideraciones | No todos los campos de datos son modificables por el usuario. |

|  |  |
| --- | --- |
| Escenario de refinamiento: Registro de vehículo - Testabilidad | |
| Escenario | El equipo de QA desea realizar pruebas unitarias al módulo de Registro de vehículo para evaluar la calidad mientras el sistema está en estado de mantenimiento. El porcentaje de código cubierto en las pruebas de fue de 90%. |
| Metas de negocio | Sistema testeable. |
| Atributos de calidad relevantes | Testabilidad. |
| Estímulo | Ejecución de pruebas unitarias. |
| Fuente del estímulo | Equipo de QA desea realizar pruebas unitarias al sistema. |
| Entorno | Sistema en mantenimiento |
| Artefacto | Módulo de registro de vehículo |
| Respuesta | Captura de los resultados de las pruebas unitarias. |
| Métrica | 90% de código cubierto. |
| Preguntas | ¿Qué funcionalidades no fueron cubiertas por las pruebas unitarias? ¿Qué tipos de pruebas se realizaron? |
| Consideraciones | El porcentaje de cobertura debe ser el mayor posible y se deberían de realizar diferentes tipos de pruebas (Caja blanca, negra y gris). |

**CONSOLIDACION DE ESCENARIOS**

**Atributo de Calidad Disponibilidad:**

1. E01: Un futuro usuario desea registrarse en la aplicación durante un estado de sobrecarga en el sistema. Uno de los nodos de este falla y el secundario toma el control y reestablece la conexión en 5 segundos, permitiendo que el usuario termine el registro satisfactoriamente.

**Atributo de Calidad Performance:**

1. E04: Cuando un usuario desea registrarse en la aplicación y el sistema realiza la operación en un tiempo promedio de 2 segundos.
2. E02: Un usuario requiere consultar la ubicación de un vehículo y su respectiva ruta mediante la aplicación y que este actualice la posición en mínimo de 5 segundos.

**Atributo de Calidad Usabilidad:**

1. E03: Cuando un contacto de emergencia solicita la ubicación del usuario se muestra su ubicación y la ruta que está siguiendo en 10 segundos.
2. E07: El usuario quiere modificar la información de su perfil mientras el sistema está en condiciones normales de uso y logra acceder a dicha funcionalidad y realizar la tarea en 2 minutos.

**Atributo de Calidad Modificabilidad:**

1. E06: Cuando requiere realizar un cambio o añadir una funcionalidad al sistema en el módulo del usuario y el costo de dicho cambio es el menor posible.
2. E05: Se quiere agregar una funcionalidad al usuario para que este pueda consultar el tiempo promedio de llegada de un bus a un determinado paradero y que la implementación se realice con el menor costo posible.

**Atributo de Calidad Testeabilidad:**

1. E08: El equipo de QA desea realizar pruebas unitarias al módulo de Registro de vehículo para evaluar la calidad mientras el sistema está en estado de mantenimiento. El porcentaje de código cubierto en las pruebas de fue de 90%.

**PRIORIZACION DE ESCENARIOS**

|  |  |  |
| --- | --- | --- |
| #Escenario | Descripcion | Votos |
| 1 | Registro de Usuario (Disponibilidad) | 2 |
| 2 | Registro de Usuario (Performance) | 0 |
| 3 | Consulta de Rutas (Performance) | 3 |
| 4 | Conacto de Emergencia (Usabilidad) | 1 |
| 5 | Editar Usuario (Usabilidad) | 0 |
| 6 | Editar Usuario (Modificabilidad) | 0 |
| 7 | Consulta de Rutas (Modificabilidad) | 1 |
| 8 | Registro de Vehículo (Testeabilidad) | 0 |

**Bibliografía:**

<http://noticias.terra.com.pe/peru/lima-la-ciudad-de-9-millones-y-con-mas-taxis-que-nueva-york,f0daab3fd02a0410VgnVCM3000009acceb0aRCRD.html> (Lima la ciudad de 9 millones y con más taxis que Nueva York) (Consulta: 28/03/2015)

<http://www.limacomovamos.org/cm/wp-content/uploads/2012/09/InformeLimaComoVamos2011-13MovilidadyTransporte.pdf> (Informe Lima Como Vamos 2011 Movilidad y Transporte) (Consulta: 5/04/2015)

<http://zapthink.com/2004/02/16/when-not-to-use-an-soa.html> (When not to use SOA) (Consulta: 10/04/2015)

<http://www.javajazzup.com/issue2/page28.shtml> (Service Oriented Architecture) (Consulta: 10/04/2015)

<http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.226.685> (SOA: To do or not to do) (Consulta: 11/04/2015)

<http://abdennour-insat.blogspot.com/2012/05/1-intoduction-to-soa.html> (Introduction to SOA) (Consulta: 11/04/2015)

<http://www.petrikainulainen.net/programming/spring-framework/spring-data-jpa-tutorial-part-nine-conclusions/> (Spring Data JPA Tutorial Part Nine: Conclusion) (Consulta: 13/04/2015)

<https://www.opengroup.org/soa/source-book/soa_refarch/services.htm> (SOA Reference Architecture Technical Standar: Service Layer) (Consulta: 15/04/2015)

<http://betanews.com/2013/11/04/comparing-cloud-vs-on-premise-six-hidden-costs-people-always-forget-about/>

(Comparing cloud vs on-premise? Six hidden costs people always forget about) (Consulta: 15/04/2015)

# <http://stackoverflow.com/questions/12790853/how-does-spring-data-jpa-differ-from-hibernate-for-large-projects>

# ([How does Spring Data JPA differ from Hibernate for large projects](http://stackoverflow.com/questions/12790853/how-does-spring-data-jpa-differ-from-hibernate-for-large-projects))(Consulta: 15/04/2015)

# <http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/\>(REST vs. SOAP – The Right WebService)(Consulta: 15/04/2015)

### <https://blogs.oracle.com/GeorgeTrujillo/entry/mysql_versus_oracle_features_functionality>(MySQL versus Oracle Features/Functionality)(Consulta: 15/04/2015)

# <http://www.developerfusion.com/article/9794/domain-driven-design-a-step-by-step-guide-part-1/>

# <http://sa.inceptum.eu/sites/sa.inceptum.eu/files/Content/Quality%20Attribute%20Generic%20Scenarios-2.pdf>

# <https://upc.blackboard.com/bbcswebdav/pid-2869125-dt-content-rid-10322254_1/courses/SI253-1501-SW71/2006_005_001_14795.pdf>

# <https://msdn.microsoft.com/en-us/magazine/dd419654.aspx>

# <http://www-public.int-evry.fr/~gibson/Teaching/CSC7322/ReadingMaterial/Evans03.pdf>