



UNIVERSIDAD NACIONAL DE TUCUMÁN

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

DEPARTAMENTO ASD DE ELECTRICIDAD, ELECTRÓNICA Y COMPUTACIÓN

TRABAJO DE GRADUACIÓN

INFORME FINAL

Web App embebida en dispositivos móviles para la gestión de registros sobre la contaminación de afluentes y ríos.

Autores

Paula Fabiana SOTO (CX05-0967-4) - Ing. en Computación

Sergio Daniel VALLEJO (CX05-0392-4) - Ing. en Computación

Tutores

Ing. Maximiliano Augusto ODSTRCIL

7 de Marzo de 2013

Agradecimientos

Agradecemos a nuestras familias que nos apoyaron y ayudaron durante el transcurso de la carrera.

Agradecemos a la Universidad en su conjunto, pública y gratuita, que nos formó académicamente.

Gracias a nuestro tutor Maximiliano Odstrcil por aceptar la dirección de nuestro proyecto y por su buena disposición.

Por último, queremos agradecer a nuestros compañeros y amigos por los momentos de estudio, logros y festejos compartidos durante la carrera, que nos motivaron para seguir adelante.

Índice general

Introducción	6
Objetivos	6
Android	7
Desarrollo de Software	8
Análisis y Diseño Orientado a Objetos	9
Lenguaje de Modelado Unificado (UML)	10
Algunas definiciones	11
Ciclo de vida Iterativo Evolutivo	12
Beneficios	13
Proceso Unificado (UP)	13
Características	13
Disciplina de Requisitos	16
Introducción	16
Identificación de usuarios participantes	16
Educción y extracción de requisitos.	17
Planificación y Realización de Entrevistas. Estudio de Documentación . . .	17
Entrevistas. Planificación y Realización	17
Estudio de la Documentación	18
Estimación del proyecto	18
Puntos de Función y COCOMO	18
Especificación de Requisitos de Software	24
1. Introducción	24

<i>ÍNDICE GENERAL</i>	3
2. Descripción general	25
3. Requisitos específicos	26
Disciplina de Análisis	33
Vista de Casos de Uso	33
Diagrama de Contexto	34
Diagramas de Casos de Uso	35
Diagrama de Casos de Uso general	35
Diagrama de Casos de Uso de gestión de usuarios	35
Diagrama de Casos de Uso de gestión de tienda	36
Diagramas de Actividad	36
Autenticar	36
Registrar	37
Crear Tienda	38
Comprar Producto	39
Diagrama de Clases del Dominio	40
Disciplina de Diseño	41
Descripción Textual de Casos de Uso	41
Caso de Uso UC1: Autenticar	41
Caso de Uso UC2: Registrar	43
Caso de Uso UC3: Crear Tienda	45
Caso de Uso UC4: Editar Tienda	47
Caso de Uso UC5: Editar Perfil	48
Caso de Uso UC6: Crear Producto	50
Caso de Uso UC7: Editar Producto	51
Caso de Uso UC8: Eliminar Producto	52
Caso de Uso UC9: Buscar Producto	54
Caso de Uso UC10: Comprar Producto	55
Caso de Uso UC11: Listar compras realizadas	56
Caso de Uso UC12: Listar ventas realizadas	58

Caso de Uso UC13: Contactar Usuario	59
Diagrama de Clases de Diseño	61
Diagramas de Secuencia	62
Autenticar	62
Registrar	62
Crear Tienda	63
Agregar Producto	63
Comprar Producto	64
Interfaz de Usuario	65
Disciplina de Implementación	78
Arquitectura de la aplicación	78
Diagrama de Despliegue	79
Elección del Lenguaje	79
Arquitectura de Android	80
PhoneGap	81
Ventajas de PhoneGap	82
Desventaja de phoneGap	83
Web services	83
Razones para crear servicios Web	84
REST	84
Características de seguridad	86
Canal de comunicación cifrado	86
ACL para cada objeto	87
Backbone.js	88
Modelo Vista Presentador (MVP)	90
jQuery	91
jQuery Mobile	91
Herramientas de desarrollo	92

Disciplina de Pruebas	94
Test de Unidades	94
Introducción	94
Pruebas de Caja Blanca	94
Test de Módulos	94
Introducción	94
Pruebas de Caja Negra	95
Pruebas de Estrés	95
Test de Integración	95
Introducción	95
Pruebas de Integración	95
Test de Aceptación	96
Introducción	96
Prueba Alfa	96
Prueba Beta	96
Conclusiones	97

Introducción

La tecnología de los dispositivos móviles ha avanzado rápidamente en los últimos años, llegando a ser actualmente auténticas computadoras de bolsillo. La gran demanda por este tipo de dispositivos genera un gran interés por parte de empresas que desean crear aplicaciones para un mercado en pleno auge, buscando aprovechar no sólo la gran cantidad de usuarios de estas plataformas, sino también la posibilidad de ofrecer funcionalidades y capacidades imposibles para un sitio web convencional.

Muchas empresas comenzaron a desarrollar aplicaciones para teléfonos celulares y tablets como un complemento de sus sistemas existentes, haciendo que sea necesario que se desarrollen tecnologías para la comunicación entre la aplicación principal y los clientes que sean seguras y eficientes.

La plataforma que actualmente posee una mayor cantidad de usuarios y mayor crecimiento es Android, debido a que se trata de un Sistema Operativo abierto que cualquier fabricante puede adaptar e instalar en sus dispositivos, que está en constante evolución, y que aporta gran cantidad de servicios y aplicaciones.

Objetivos

Los principales objetivos del sistema a desarrollar son permitir que usuarios puedan utilizar funcionalidades del sitio web de comercio electrónico existente desde cualquier dispositivo móvil con sistema operativo Android que servirá como complemento de la aplicación web, permitiendo aumentar la cantidad de usuarios y brindando facilidades para que puedan realizar operaciones desde cualquier lugar en el que se encuentren por más que no tengan acceso a una computadora con Internet y aumentar enormemente el volumen

de ventas de las tiendas. Estas funcionalidades abarcan compra, venta y comunicación entre usuarios.

Otro de los objetivos es implementar un método de conexión con la aplicación web actual permitiendo operar con gran eficiencia y bajo costo, implementando una comunicación a través de web services.

El sistema debe ser seguro, escalable, de fácil mantenimiento y muy simple de usar utilizando sólo interfaces táctiles. Se deben implementar mecanismos de seguridad que impidan la visualización o alteración indebida de los datos sensibles de cada usuario.

Nuestros objetivos personales son aplicar los conocimientos adquiridos a lo largo de la carrera, el aprendizaje del trabajo en equipo, la coordinación en el desarrollo y sumar experiencia y conocimientos.

Android

Android es un sistema operativo Open Source, para dispositivos móviles basado en Linux. La mayor parte de su código fue publicado por Google bajo la licencia Apache. Actualmente se lo puede encontrar instalado en teléfonos celulares, tablet PCs y televisores.

Las aplicaciones de Android normalmente están escritas en el lenguaje de programación orientado a objetos Java, aunque es posible utilizar frameworks para desarrollar aplicaciones en otros lenguajes, como HTML5+CSS3+Javascript o Python. Estas aplicaciones se ejecutan sobre una máquina virtual Dalvik, que está optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando en el sistema operativo subyacente el soporte de aislamiento de procesos, gestión de memoria e hilos.

Incluye una tienda de aplicaciones llamada Google Play, desarrollada por Google. Esta aplicación se encuentra instalada en la mayoría de los dispositivos Android y permite a los usuarios navegar y descargar aplicaciones publicadas por los desarrolladores. Google retribuye a los desarrolladores el 70 % del precio de las aplicaciones.

Por otra parte, los usuarios pueden instalar aplicaciones desde otras tiendas virtuales (tales como Amazon Appstore o GetJar) o directamente en el dispositivo si se dispone del

archivo APK de la aplicación.

La compañía de investigación Canals estimó en el segundo trimestre del año 2009 que Android tenía una cuota de mercado del 2,8 % de smartphones en todo el mundo. En el cuarto trimestre de 2010 esta cifra había aumentado al 33 % del mercado, convirtiéndose en la plataforma para smartphones más vendida. En el tercer trimestre de 2011 se estima que más de la mitad (52,5 %) del mercado de smartphones pertenece a Android. En el tercer trimestre de 2012 Android tenía una cuota del 75 % del mercado mundial de smartphones según la firma de investigación IDC. En la figura 1 puede verse la evolución de las ventas de teléfonos celulares según su sistema operativo.

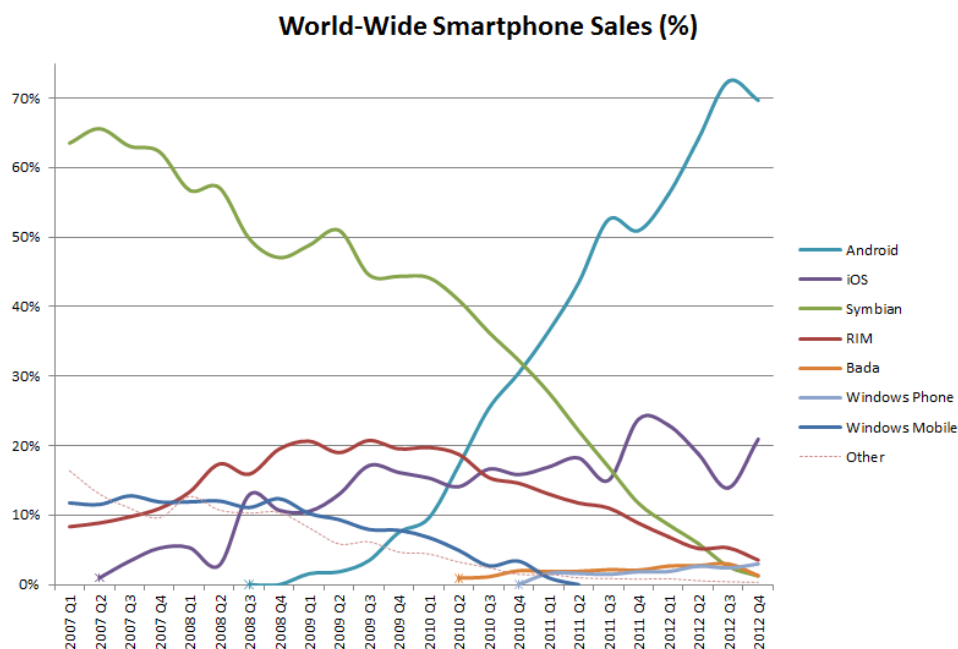


Figura 1: Ventas de smartphones según su sistema operativo, a nivel mundial

La cuota de Android en el mercado varía según la ubicación. En julio de 2012, la cuota de mercado de Android en los Estados Unidos fue de 52 %, y se eleva al 90 % en China.

Desarrollo de Software

Para resolver un problema, se sigue un Proceso de Resolución:

1. Identificar el problema.

2. Definir y Representar el problema.
3. Explorar las posibles estrategias.
4. Aplicar y mejorar las estrategias.
5. Mirar atrás y Evaluar los efectos de la actividad realizada.

El proceso de resolución de problemas software abarca:

1. Análisis y especificación de requisitos (qué)
2. Diseño del sistema Software (cómo)
3. Codificación (realización del cómo)
4. Pruebas
5. Instalación (la solución debe ser usada)

Análisis y Diseño Orientado a Objetos

El análisis se enfoca en la investigación de los problemas y los requisitos, más que en la solución. Es un término utilizado, por ejemplo, en el análisis de requisitos (una investigación de los requisitos) o análisis orientado a objetos (una investigación de los objetos de dominio).

Durante el análisis orientado a objetos, se trata de encontrar y describir los objetos o conceptos en el dominio del problema.

El diseño tiende a una solución conceptual (de software y hardware) que cumpla los requisitos, en lugar de su implementación. Los diseños pueden ser implementados, y la implementación (como ser el código) expresa el diseño realizado, verdadero y completo. El término es muy utilizado, como en el diseño orientado a objetos o el diseño de bases de datos.

Durante el diseño orientado a objetos, se procura definir objetos de software, y cómo ellos colaboran para satisfacer los requisitos.

Lenguaje de Modelado Unificado (UML)

UML es un lenguaje estándar de diagramación. Puede ser usado para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

Hay tres maneras de aplicar UML:

- *Como boceto*: Diagramas informales e incompletos (frecuentemente dibujados en pizarras blancas) creados para explorar partes dificultosas del espacio del problema o de la solución, aprovechando el poder de los lenguajes visuales.
- *Como plano*: Diagramas de diseño relativamente detallados, usados para:
 - Ingeniería inversa para visualizar y entender mejor el código existente en diagramas UML.
 - Generación de código. (ingeniería hacia adelante). Antes de programar, algunos diagramas detallados pueden proveer una guía para la generación de código (ej. Java), manualmente o automáticamente con una herramienta.
- *Como lenguaje de programación*: Especificación ejecutable completa, de un sistema software en UML. El código ejecutable es automáticamente generado.

En el presente proyecto, en general se procuró usar UML para la diagramación de bocetos, con un enfoque ágil, para la comunicación entre los miembros del equipo, y la resolución de distintos problemas, a lo largo de las disciplinas del proceso. Sin embargo, algunos diagramas se realizaron con bastante detalle, utilizando herramientas CASE de UML para la documentación de casos relevantes y para la posterior generación de código.

Una misma notación puede ser usada para tres perspectivas y tipos de modelos:

- *Perspectiva conceptual*: Los diagramas son interpretados como cosas descriptas en una situación del mundo real o dominio de interés.
- *Perspectiva de especificación (software)*: Los diagramas describen abstracciones o componentes de software con especificaciones e interfaces, pero no obligadas a una implementación particular.

- *Perspectiva de implementación (software)* Los diagramas describen implementaciones de software en una tecnología particular.

Las tres perspectivas se aplicaron a lo largo del proceso de desarrollo del sistema.

Algunas definiciones

- *Análisis Orientado a Objetos (AOO)*: El AOO es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema.
- *Diseño Orientado a Objetos (DOO)*: El DOO es un método de diseño que abarca el proceso de descomposición orientada a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico del sistema que se diseña.
- *Programación Orientada a Objetos (POO)*: La POO es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia.
- *Proceso Software*: colección de actividades que comienza con la identificación de una necesidad y concluye con el retiro del software que satisface dicha necesidad.
- *Ciclo de vida*: la transformación que el producto software sufre a lo largo de su vida, desde que nace (o se detecta una necesidad) hasta que muere (o se retira el sistema).
- *Estados*: resultado de las transformaciones que sufre el producto software a lo largo de su ciclo de vida y representan en esencia el producto mismo.
- *Modelo de ciclo de vida*: es la representación descriptiva y prescriptiva del ciclo de vida que indica el orden cronológico en el que deben llevar a cabo las actividades del

proceso software. Un ciclo de vida debe determinar el orden de las fases del proceso software y establecer los criterios de transición para pasar de una fase a la siguiente.

- *Disciplina:* Una colección de actividades relacionadas con un área de interés importante dentro del proyecto global.

Ciclo de vida Iterativo Evolutivo

“Deberías usar un desarrollo iterativo sólo en los proyectos que quieras que tengan éxito” - Martin Fowler

El desarrollo iterativo evolutivo, en contraste con el ciclo de vida de cascada o secuencial, consiste en la programación y pruebas tempranas de un sistema parcial, en ciclos repetitivos. También normalmente supone que el desarrollo se inicia antes de que todos los requisitos se definan en detalle; el feedback se utiliza para aclarar y mejorar las características cambiantes.

Cuenta con pasos de desarrollo cortos y rápidos, feedback, y adaptación para aclarar los requisitos y el diseño, sin tanta especulación antes de la programación, como en método de cascada. La investigación demuestra que los métodos iterativos se asocian con mayores tasas de éxito y de productividad, y bajos niveles de defectos.

En este ciclo de vida, el desarrollo está organizado en una serie de mini-proyectos cortos y de longitud fija (timeboxing), llamados iteraciones; el resultado de cada uno es un sistema parcial probado, testeado e integrado. Cada iteración tiene sus propias actividades de análisis de requisitos, diseño, implementación y pruebas.

El ciclo de vida iterativo se basa en la ampliación y refinamiento sucesivos de un sistema a través de múltiples iteraciones, con retroalimentación cíclica y adaptación, como conductores principales para converger al sistema adecuado. Como el sistema crece gradualmente con el tiempo, iteración por iteración, se lo llama desarrollo iterativo e incremental. Y debido a que la retroalimentación y la adaptación producen una evolución en las especificaciones y el diseño, también es conocido como desarrollo iterativo y evolutivo.

La salida de una iteración no es un prototipo experimental o desechable, y el desarrollo iterativo no es prototipado. Por el contrario, la salida es un subconjunto, en calidad de

producción, del sistema final.

Beneficios

- Menos fracasos de proyectos, mejor productividad y tasas de defectos reducidas; demostrado por investigación de métodos iterativos evolutivos.
- Mitigación temprana, más que tardía, de altos riesgos (técnicos, requisitos, objetivos, usabilidad, etc.)
- Progreso visible y temprano.
- Feedback, compromiso de usuario, y adaptación tempranos, llevando a un sistema refinado que satisface más estrechamente las necesidades reales de los clientes.

Según nuestra propia experiencia, una ventaja adicional es que nos permitió refinar además nuestras propias actividades, mejorando la productividad, estimación, diagramación y calidad del desarrollo en cada iteración.

Siguiendo recomendaciones de metodologías ágiles, trabajamos con ciclo de vida iterativo evolutivo.

Proceso Unificado (UP)

Un proceso de desarrollo de software describe un enfoque para la construcción, despliegue, y posiblemente mantenimiento del software.

El Proceso Unificado nació como un popular proceso de desarrollo iterativo para la construcción de sistemas orientados a objetos. Es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

Características

- *Iterativo e Incremental*: Es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada

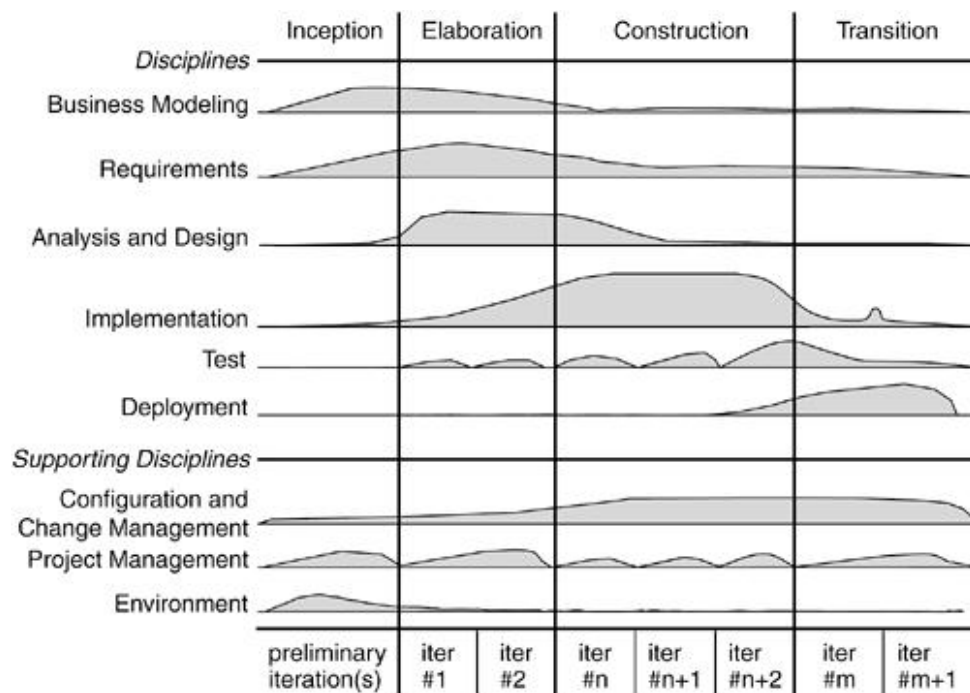


Figura 2: Disciplinas y fases del Proceso Unificado

una de estas fases es a su vez dividida en una serie de iteraciones. Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en una serie de disciplinas. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- *Dirigido por los casos de uso:* Los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.
- *Centrado en la arquitectura:* Asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.
- *Enfocado en los riesgos:* Requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada

iteración, en especial los de la fase de Elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

- *UML*: Hace uso extensivo de diagramas UML para sus artefactos.

Se trabajó sumando al proyecto, para cada iteración, un porcentaje (30 % aprox.) del total de casos de uso previstos. A medida que se iba iterando se desarrollaban las disciplinas para estos nuevos casos de uso y se refinaban los casos de uso agregados en iteraciones anteriores, gracias a la experiencia adquirida y con la ayuda del feedback del usuario final.

Disciplina de Requisitos

Introducción

Esta especificación tiene como objetivo analizar y documentar las necesidades funcionales que deberán ser soportadas por el sistema a desarrollar. Para ello, se identificarán los requisitos que ha de satisfacer el nuevo sistema mediante entrevistas, el estudio de los problemas de las unidades afectadas y sus necesidades actuales. Además de identificar los requisitos se deberán establecer las prioridades, lo cual proporciona un punto de referencia para validar el sistema final que compruebe que se ajusta a las necesidades del usuario.

Identificación de usuarios participantes

Los objetivos de esta tarea son identificar a los responsables de cada una de las unidades y a los principales usuarios implicados. Para ello se consideran los siguientes aspectos:

- Incorporación de usuarios al equipo de proyecto.
- Conocimiento de los usuarios de las funciones a automatizar.
- Repercusión del nuevo sistema sobre las actividades actuales de los usuarios.
- Implicaciones legales del nuevo sistema.

Se identificaron los siguientes usuarios:

- *Grupo de Administradores*: Formado por los administradores de la aplicación web actual.

- *Grupo de Usuarios Compradores:* Formado por usuarios que realizan operaciones de compra.
- *Grupo de Usuarios Vendedores:* Formado por usuarios que tienen una tienda con productos en venta.
- *Grupo de Usuarios No Registrados:* Formado por usuarios que pueden realizar operaciones de búsqueda y listado de productos sin estar registrados en la aplicación.

Es de destacar la necesidad de una participación activa de los usuarios del futuro sistema en las actividades de desarrollo del mismo, con objeto de conseguir la máxima adecuación del sistema a sus necesidades y facilitar el conocimiento paulatino de dicho sistema, permitiendo una rápida implantación.

Educción y extracción de requisitos.

Planificación y Realización de Entrevistas. Estudio de Documentación

Esta tarea tiene como finalidad capturar los requisitos de usuarios para el desarrollo del sistema.

Para el análisis de requisitos se usaron distintas técnicas de educación de requisitos. Entre ellas el estudio de la documentación provista por parte de administradores de la aplicación web y de los Responsables; entrevistas abiertas y estructuradas, análisis de la aplicación web.

Entrevistas. Planificación y Realización

La entrevista consiste en una interacción sistemática con un usuario para educir los conocimientos de éste.

Se hicieron entrevistas abiertas y estructuradas.

El proceso consiste en la planificación de las entrevistas, en donde se incluye fecha, hora, lugar, duración estimada y guión de la entrevista; luego se llevan a cabo las entrevistas y se las documenta identificando los requisitos con sus prioridades.

A partir de las entrevistas realizadas con los administradores de la aplicación web, se identifican los requisitos que debe cumplir el sistema y se establecerá una prioridad para los mismos, de acuerdo a las necesidades de los usuarios y a los objetivos a cubrir por la aplicación para Android.

Estudio de la Documentación

El estudio de la documentación consiste en la extracción de requisitos de los documentos e impresiones que forman parte del sistema actual.

Una vez que se obtiene toda la documentación relevante al sistema que se va a desarrollar, se lee cuidadosamente la información contenida en esos documentos y luego con la técnica de análisis estructural de textos, se extraen conceptos fundamentales del dominio buscando estructuras preestablecidas.

Estimación del proyecto

Puntos de Función y COCOMO

Introducción

Como alternativa se propuso en este trabajo la utilización combinada de dos métodos (Puntos de Función y COCOMO) tendientes a proporcionar una estimación más precisa. Primero se utilizará el método de Puntos de Función para determinar las líneas de código del proyecto software, la cual mantiene una distorsión.

En segundo lugar se aplicará el método COCOMO partiendo de la información producida por el anterior (líneas de código) para llegar a una estimación precisa de las horas hombre a aplicar y fundamentalmente a la estimación de la duración total del proyecto, dado por sus características intrínsecas independientemente de los recursos a emplear.

Aplicación del Método

Este método calcula los puntos de función de un sistema descomponiendo al mismo en cinco funciones principales (entradas, salidas, consultas, ficheros internos y externos),

asignándoles valores de acuerdo a su complejidad y en función de la cantidad de cada uno de ellos se llega a determinar, mediante su sumatoria, los puntos de función. A continuación se detallan los parámetros a considerar en el cálculo del método:

- *Datos lógicos internos (ILF)*: Corresponde al almacén de datos identificados por un nombre en un diagrama de flujo de datos. Un ejemplo de este parámetro serían los archivos maestros.
- *Datos de Interfase externos (EIF)*: Similar a los ILF, pero estos almacenes de datos son mantenidos por otra aplicación. Por ejemplo un sitio Web que muestra datos fuera de la empresa mantenido por otro sistema interno a la empresa.
- *Entrada externas (EI)*: Son datos o información de control que se introducen en la aplicación desde fuera de sus límites. Estos datos mantienen los datos lógicos internos. Un ejemplo de este parámetro son las pantallas de entradas en las que aparecen botones como agregar, modificar y quitar.
- *Salidas externas (EO)*: Son datos o información de control que salen de los límites de la aplicación. Esta salida debe ser considerada única si tiene un formato único o si el diseño requiere un proceso lógico distinto. Un ejemplo de este parámetro son los informes, cada informe producido se cuenta como una salida.
- *Consultas externas (EQ)*: Representan los requisitos de información a la aplicación, no actualiza ILF y no contienen datos derivados. Por ejemplo se tienen las búsquedas inmediatas de datos.

Descripción de la Técnica

Para el cálculo de la complejidad, se utilizaron los siguientes parámetros:

- *Data Element Type (DET)*: Campo único identificable por el usuario. Hace referencia a las claves únicas de los almacenes de datos.
- *Record Element Type (RET)*: Subgrupo de datos inidentificables por el usuario.
- *File Type Referenced (FTR)*: ILF leído o mantenido.

Definición de parámetros básicos externos del sistema

Para el cálculo de la complejidad en un ILF se accede a una tabla de doble entrada de acuerdo a la cantidad de DET y RET que presenta.

Ejecución del método en el proyecto actual

Se identificaron 4 ILF.

No se identificó ningún EIF.

ILF	RET	DET	Complejidad	Valor
Usuario	1	3	Baja	7
Vendedor	1	6	Baja	7
Producto	1	6	Baja	7
Orden	1	7	Baja	7

Baja: $4 * 7 = 28$

Total: 28

EI	FTR	DET	Complejidad	Valor
Alta Usuario	1	9	Baja	3
Modificar Usuario	1	9	Baja	3
Alta Tienda	1	4	Baja	3
Modificar Tienda	1	4	Baja	3
Alta Producto	1	6	Baja	3
Baja Producto	1	6	Baja	3
Modificar Producto	1	6	Baja	3
Alta orden	1	7	Baja	3

Baja: $8 * 3 = 24$

Total: 24

EO	FTR	DET	Complejidad	Valor
Listado de Productos	1	7	Baja	4
Listado de Órdenes	2	9	Media	5

Baja: $1 * 4 = 4$

Media: $1 * 5 = 5$

Total: 9

EQ	FTR	DET	Complejidad	Valor
Inicio de sesión	1	2	Baja	3
Consulta de tienda	2	9	Media	4
Consulta de producto	2	9	Media	4

Baja: $1 * 3 = 3$

Media: $2 * 4 = 8$

Total: 11

Resumen

Parámetro	Complejidad	Cantidad	Peso	Total
ILF	Baja	4	7	28
EI	Baja	8	3	24
EO	Baja	1	4	4
	Media	1	5	5
EQ	Baja	1	3	3
	Media	2	4	8

$$PF = ILF + EI + EO + EQ$$

$$PF = 28 + 24 + 9 + 11 = 72$$

$$PF = 72$$

Ajuste de los Puntos de Función

Característica	Descripción del grado de influencia	Valor
Comunicación de datos	Más de una PC front-end, pero la aplicación soporta un solo tipo de protocolo de comunicaciones	4
Funciones distribuidas	El proceso distribuido y la transferencia de datos son on-line en ambas direcciones	3
Rendimiento	Rendimiento y requisitos de diseño han sido definidos y revisados pero no requieren ninguna acción especial	1
Configuraciones fuertemente utilizadas	Existen algunas restricciones de seguridad o tiempo	2
Frecuencia de transacciones	No se anticipa ningún periodo de transacción pico	0
Entrada on-line de datos	Más del 30 % de las transacciones son entradas de datos interactivas	5
Diseño para la eficiencia del usuario final	Más de 6 funciones que incrementan la eficiencia del usuario final, y se establecieron requisitos de eficiencia del usuario que obligan a diseñar tareas que tienen en cuenta factores humanos	4
Actualización on-line	Actualización de 4 o más ILF. El volumen de actualización es bajo y la recuperación fácil	2
Procesos complejos	No hay procesos complejos	0
Reusabilidad	Se utiliza código reusable dentro de la aplicación	1
Facilidad de instalación	No se realizaron consideraciones ni se requirieron desarrollos especiales para la instalación por parte del usuario	0
Facilidad de operación	No se definieron por parte del usuario necesidades especiales de operación o respaldo distintas de las normales	0
Instalación en distintos lugares	Se necesita diseñar la aplicación para ser usada en múltiples lugares pero funcionará bajo entornos idénticos de hardware y software	1
Facilidad de cambio	No existe ninguna especificación por parte de los usuarios en este sentido	0

Grado de Influencia: $TDI = 24$

Factor de ajuste: $VAF = 0,65 + 0,01 * TDI = 0,89$

Puntos de función ajustados: $PFA = PF * AF = 72 * 0,89 = 64,08$

Estimación de las líneas de código necesarias

QSM SLOC/FP Data JavaScript: 54 LOC por PF

LOC= $54 * 64,08 = 3460,32$

Aplicación del método COCOMO

La estimación del tiempo de desarrollo y la cantidad necesaria de personas participantes se realizó utilizando el método intermedio del modelo orgánico.

RELY	1
DATA	1.08
CPLX	0.7
TIME	1
STOR	1
VIRT	0.87
TURN	0.87
ACAP	1.19
AEXP	1.13
PCAP	1
VEXP	1.1
LEXP	1.07
MODP	1
TOOL	0.83
SCED	1.04

Factor de Ajuste: $A = 0,782$

$$MM = A3,2(KLOC)^{1,05} = 9,21 \text{ meses hombre}$$

$$TDEV = 2,5(MM)^{0,38} = 5,81 \text{ meses}$$

$$\text{Número de personas} = 9,21/5,81 = 1,58 \approx 2 \text{ personas}$$

Especificación de Requisitos de Software

1. Introducción

Este documento es una Especificación de Requisitos Software de la Aplicación de comercio electrónico para teléfonos celulares con sistema operativo Android “DroidCommerce”. Esta documentación es fruto de las entrevistas, estudio de la documentación y del funcionamiento de la aplicación web actual, así como del análisis llevado a cabo por el equipo de desarrollo.

El objetivo de la especificación es definir en forma clara, precisa, completa y verificable todas las funcionalidades y restricciones del sistema que se desea construir.

Esta documentación está sujeta a revisiones por el grupo de administradores que se recogerán por medio de sucesivas versiones del documento, hasta alcanzar su aprobación por parte de los responsables de la aplicación web actual. Una vez aprobado, servirá de base al equipo de desarrollo para la construcción del nuevo sistema.

Esta especificación se ha realizado de acuerdo al estándar “IEEE Recommended Practice for software Requirements Specifications(IEEE/ANSI 830-1993)”.

Objetivos y alcance del sistema

Los principales objetivos del sistema a desarrollar son permitir que usuarios puedan utilizar funcionalidades del sitio web existente desde cualquier dispositivo móvil con sistema operativo Android que servirá como complemento de la aplicación web, permitiendo aumentar la cantidad de usuarios y brindando facilidades para que puedan realizar operaciones desde cualquier lugar en el que se encuentren por más que no tengan acceso a un computador con Internet y aumentar enormemente el volumen de ventas de las tiendas. Estas funcionalidades abarcan compra, venta y comunicación entre usuarios.

Otro de los objetivos es implementar un método de conexión con la aplicación web actual permitiendo operar con gran eficiencia y bajo costo.

El sistema debe ser seguro, escalable, de fácil mantenimiento y muy simple de usar utilizando sólo interfaces táctiles.

El desarrollo lo llevarán a cabo Paula Soto y Sergio Vallejo, con la opción a ser responsables del posterior mantenimiento del mismo.

Definiciones, Acrónimos y Abreviaturas

Definiciones:

- *Aplicación web*: una aplicación web es un sitio web en donde el contenido de todas o algunas de sus páginas se determina en el momento mismo de la solicitud, gracias a la ejecución de un intérprete en el servidor que traduce el código fuente en información visible en un navegador Web, de manera dinámica y con acceso en línea a los datos.
- *Régimen 24x7* disponibilidad los 7 días a la semana, 24 horas por día, 100 por ciento de disponibilidad.

Acrónimos:

- *HTML*: HyperText Markup Language.
- *HTTPS*: Hypertext Transfer Protocol Secure

Abreviaturas:

- *IEEE*: Institute of Electrical and Electronics Engineers.
- *UP*: Unified Process.
- *MCVS*: Modelo de Ciclo de Vida del Sistema.

2. Descripción general

Esta sección nos presenta una descripción general del sistema con el fin de conocer las funciones que debe soportar, los datos asociados, las restricciones impuestas y cualquier

otro factor que pueda influir en la construcción del mismo.

Las funciones que debe realizar el sistema se pueden agrupar de la siguiente manera:

- *Administración de usuarios:* debe permitir gestionar los usuarios teniendo en cuenta de si tratan de compradores o vendedores de productos en la aplicación.

Debe permitir la búsqueda y listado de productos de cualquier categoría.

Los usuarios podrán también consultar un listado de operaciones compras o ventas realizadas.

Para que un usuario comprador pueda realizar una operación de compra o un usuario vendedor una venta deberán estar previamente registrados en la aplicación.

Los usuarios deben poder también configurar opciones del perfil como el nombre, email, dirección, código postal, ciudad, provincia y país.

- *Administración de productos:* debe permitir gestionar los productos que el usuario vendedor pone a la venta y sus configuraciones como fotografía, nombre de producto, categoría, descripción, precio de producto, precio de envío.
- *Consultas de compras y ventas realizadas:* debe permitir obtener listados de las operaciones de compra y ventas realizadas por los usuarios, y se debe permitir la consulta de las mismas.

3. Requisitos específicos

Requisitos Funcionales

A. Gestion de cuentas de usuarios

Introducción: El sistema permite introducir información sobre usuarios compradores (Nombre y apellido, email, dirección, código postal, ciudad, provincia, país, etc) y editar los ya existentes.

Entrada: @nombreUsuario + Nombre + Apellido + Email + Dirección + Código-Postal + Ciudad + Provincia + País

Proceso: Comprobar si se trata de un usuario nuevo, dándolo de alta o actualizarlo si ya existe.

Se muestra en pantalla la ficha de datos de la persona.

No pueden existir dos usuarios con el mismo número de Identificación.

Salida: Datos de usuarios actualizados y mensajes de lo que está ocurriendo.

B. Gestion de Tiendas

Introducción: El sistema permite introducir información sobre las Tiendas (Nombre, Descripción, Moneda, Logo, etc), editar los ya existentes y borrarlos.

Entrada: @nombreUsuario + NombreTienda + Descripción + Moneda + Logo

Proceso: Comprobar si se trata de una tienda nueva dándola de alta en el sistema o actualizándola si ya existe.

No pueden existir dos tiendas vinculadas a una misma cuenta de usuario.

Salida: Datos de la tienda actualizados y mensajes de lo que está ocurriendo.

C. Gestión de productos

Introducción: El sistema permite introducir información sobre Productos (Nombre, Descripción, foto, precio de producto, precio de envío, etc), editar los ya existentes y borrarlos.

Entrada: @CódigoIdentificación + Nombre + Descripción + PrecioProducto + PrecioEnvío + Foto + Categoría

Proceso: Se elimina el producto.

Salida: Datos del producto actualizados y mensajes de lo que está ocurriendo.

D. Búsqueda de productos

Introducción: El sistema debe permitir la búsqueda de productos por el nombre del producto.

Entrada: CadenaBúsqueda

Proceso: Buscar las correspondencias de productos usando la cadena de búsqueda. Mostrar los resultados, aunque no se haya encontrado nada.

Salida: NombreProducto + Imagen + PrecioProducto + PrecioEnvio

E. Comprar Productos a Vendedor

Introducción: El sistema debe permitir elegir un producto del vendedor (puede ser previo paso por la pantalla de búsqueda) y asignar la cantidad del producto a comprar. Una vez finalizado, se genera la compra para ser informada al vendedor posteriormente. Se crea la obligación por parte del comprador, que debe haber iniciado la sesión previamente.

Entrada: Fecha + IdComprador + IdProducto + Cantidad

Proceso: generar la compra/venta del producto incluyendo la cantidad especificada. Generar un aviso en el vendedor. En todo momento se deben mostrar mensajes de lo ocurrido, especialmente antes de generar, avisando que se obliga a la compra.

Salida: Datos de la nueva compra. Mensajes de lo que está ocurriendo. Notificación enviada al vendedor.

F. Listado de compras

Introducción: Permite listar las compras realizadas de un determinado usuario comprador.

Entrada:

Proceso: Listar las compras realizadas

Salida: NombreProducto + PrecioProducto + PrecioEnvio + PrecioTotal + Cantidad + FechaCompra + CódigoCompra + UsuarioVendedor

G. Listado de ventas

Introducción: Permite listar las ventas realizadas de un determinado usuario vendedor.

Entrada:

Proceso: Listar las ventas realizadas

Salida: NombreProducto + PrecioProducto + PrecioEnvio + PrecioTotal + Cantidad + FechaVenta + CódigoCompra + UsuarioComprador

Suposiciones y Dependencias:

Suposiciones:

Se asume que los requisitos en este documento son estables una vez que sean aprobados por los responsables de la aplicación web. Cualquier petición de cambios en la especificación debe ser aprobada por todas las partes intervinientes y será gestionada por el equipo de desarrollo.

Dependencias:

El sistema tiene dependencia con el actual sitio web con el que compartirá los datos.

Requisitos de Usuario y Tecnológicos:**Requisitos de usuario:**

Los usuarios serán los que descarguen la aplicación de la tienda de aplicaciones de Android.

Requisitos tecnológicos:

La aplicación actuará como cliente de la aplicación web que ofrecerá Web services para el acceso a los datos.

El cliente requiere un dispositivo móvil, con pantalla táctil superior a las 3 pulgadas y Sistema Operativo Android, versión mayor o igual a 2.1.

Requisitos de Interfaces Externas:**Interfaces de usuario:**

Las interfaces de la aplicación deben ser intuitivas, fáciles de usar, amigables y de respuesta rápida. La interfaz de usuario debe ser orientada al uso táctil.

Interfaces Hardware:

Dispositivo móvil con pantalla táctil de tamaño superior a las 3 pulgadas.

Interfaces Software:

Sistema Operativo Android, versión mayor o igual a 2.1.

Requisitos de Rendimiento:

El Tiempo de respuesta de la aplicación de cada función solicitada por el usuario no debe ser superior a los 10 segundos en una velocidad efectiva de conexión con el servidor a través de 3G.

Requisitos de Desarrollo:

El ciclo de vida será Prototipado Evolutivo, debiendo orientarse hacia el desarrollo de un sistema flexible que permita incorporar de manera sencilla cambios y nuevas funcionalidades.

Restricciones de Diseño:**Ajuste a estándares:**

No se ha definido

Seguridad:

La seguridad de los datos será establecida por la utilización de cifrado mediante el uso de HTTPS en la transmisión de datos.

Se implementará también ACL para cada objeto y de esa manera evitar accesos o modificaciones que no correspondan.

Política de respaldo:

No se ha definido

Base de Datos:

La base de datos que se utilizará estará en el servidor web que funciona actualmente con la aplicación web y accederá a la misma utilizando la tecnología de web services REST.

Política de Borrado:

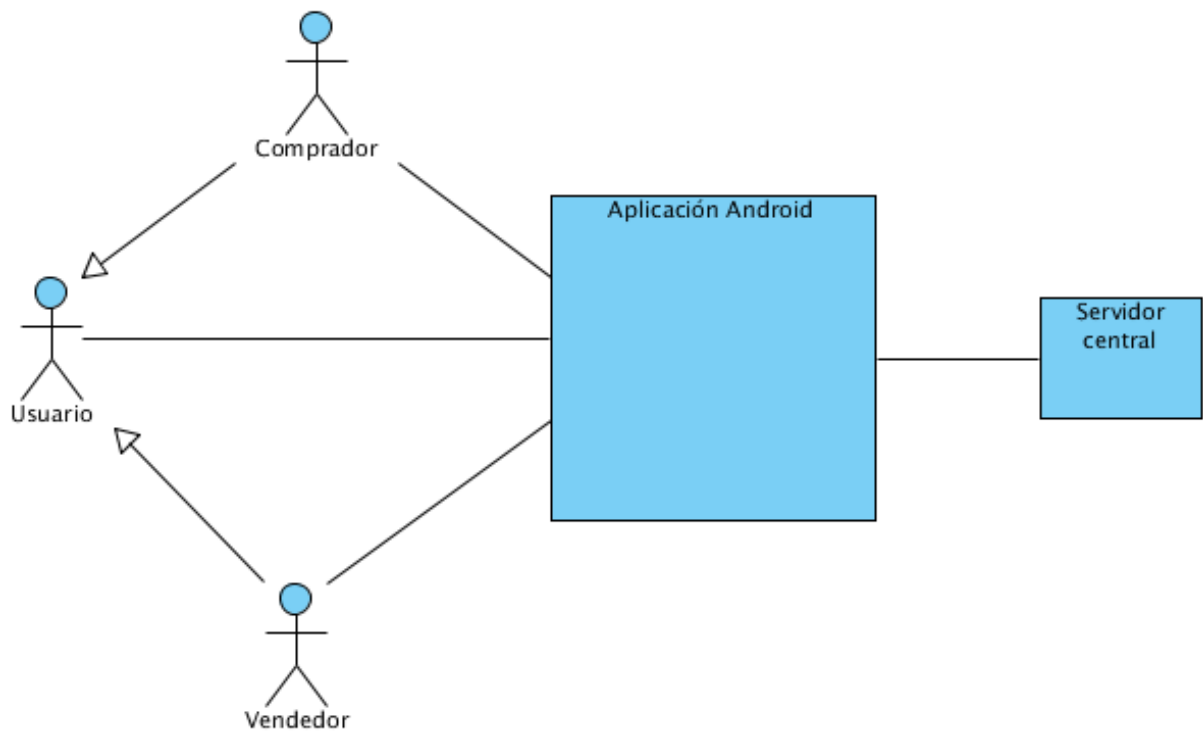
No se ha definido

Disciplina de Análisis

Vista de Casos de Uso

La vista de casos de uso captura el comportamiento de un sistema, subsistema, clase o componente, como lo ve un usuario externo. Particiona la funcionalidad del sistema en transacciones significativas para los actores (usuarios idealizados) de un sistema. Las piezas de funcionalidad interactiva son llamadas “casos de uso”. Un caso de uso describe una interacción entre actores como una secuencia de mensajes entre el sistema y uno o más actores. El término *actor* incluye a personas, como también otros sistemas de computadora o procesos.

Diagrama de Contexto



Diagramas de Casos de Uso

Diagrama de Casos de Uso general

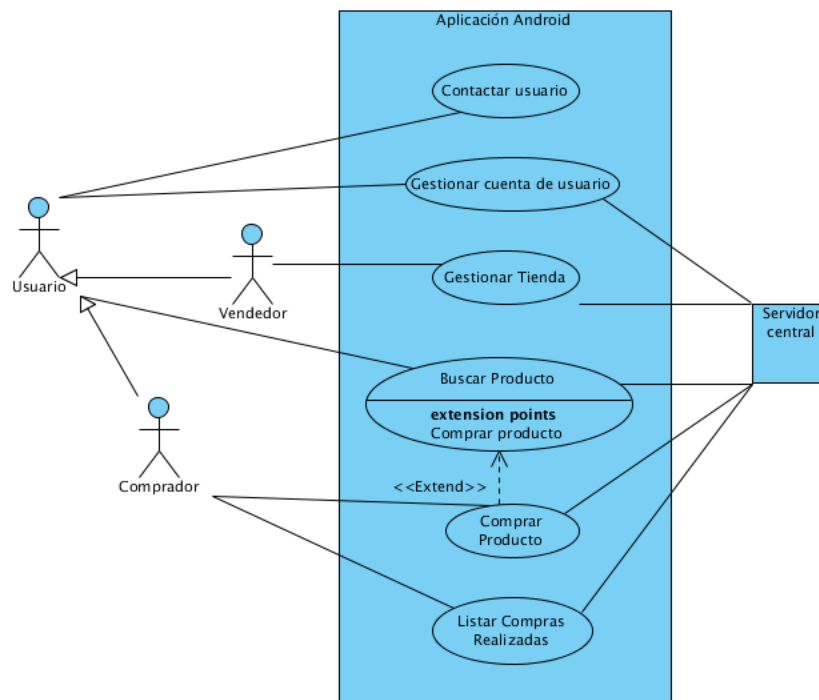


Diagrama de Casos de Uso de gestión de usuarios

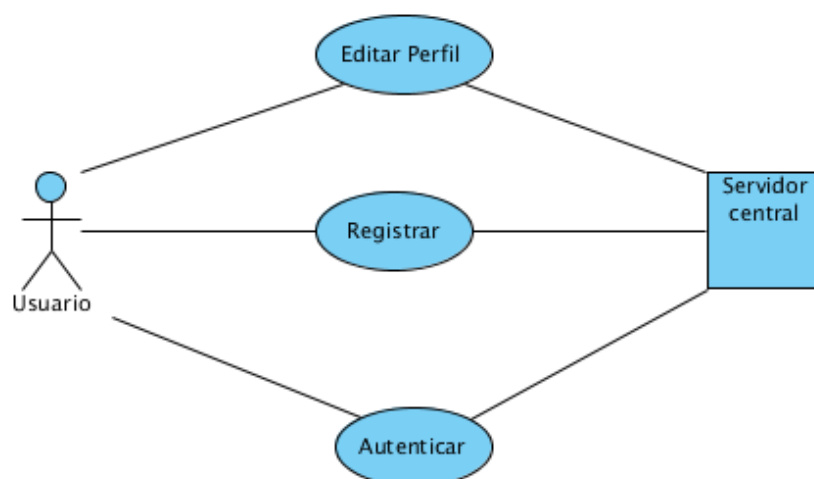
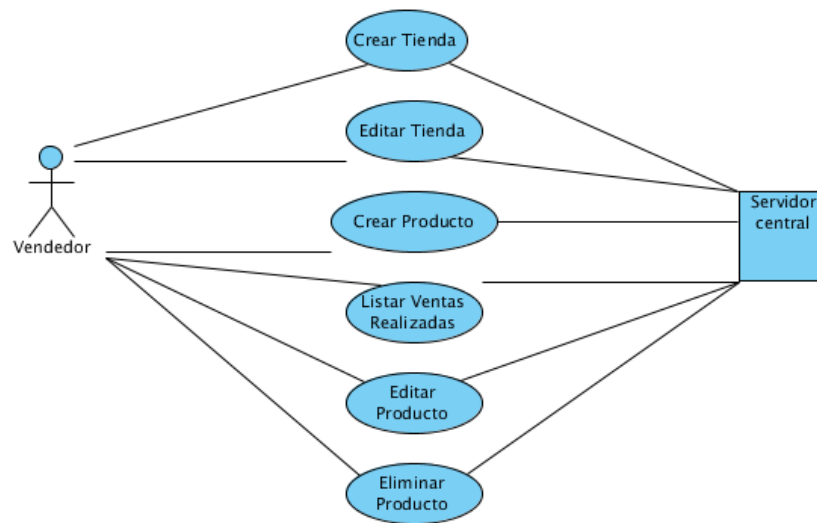
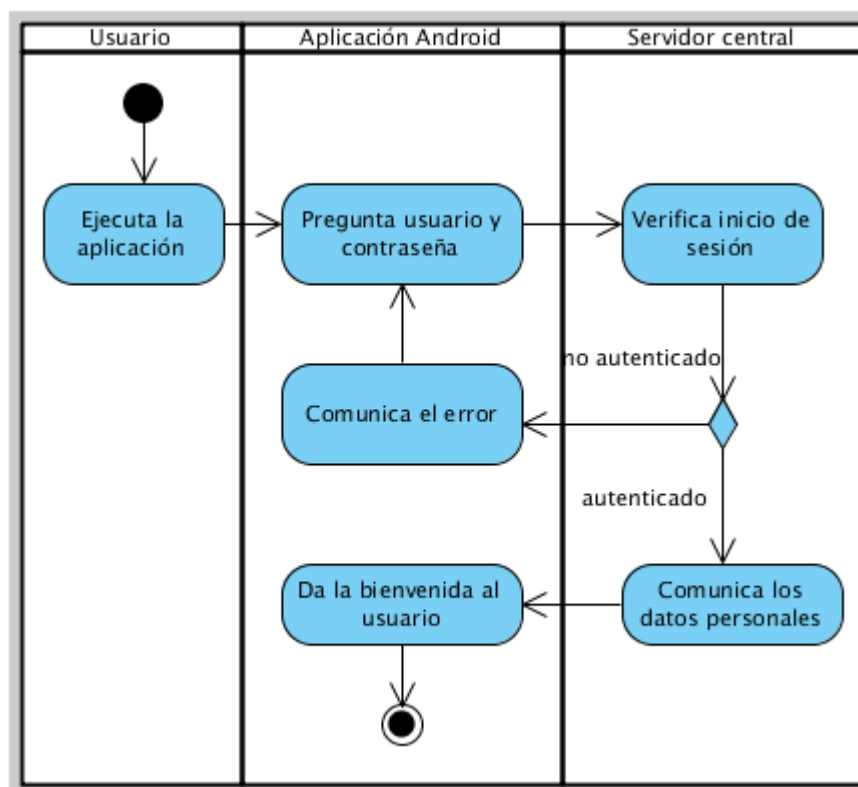


Diagrama de Casos de Uso de gestión de tienda

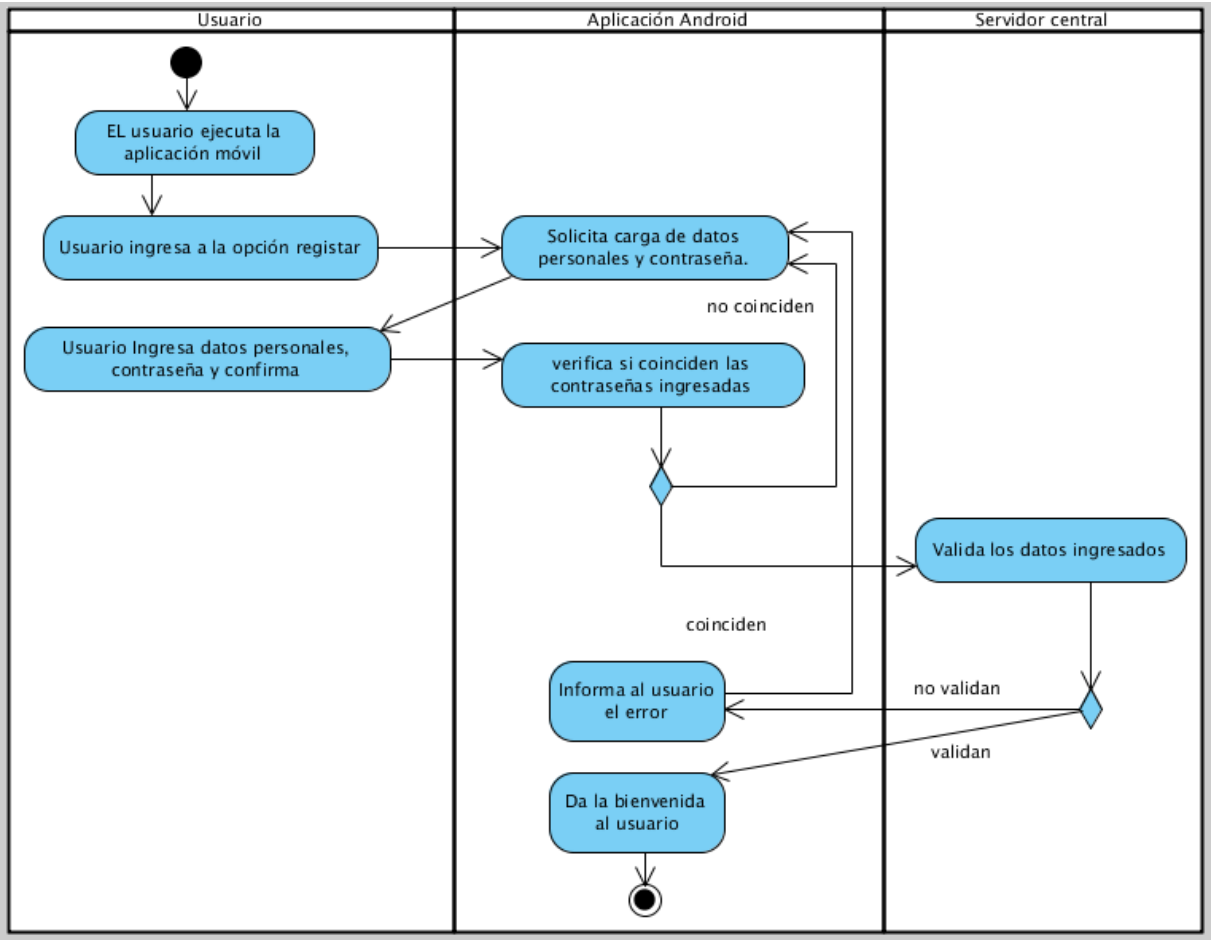


Diagramas de Actividad

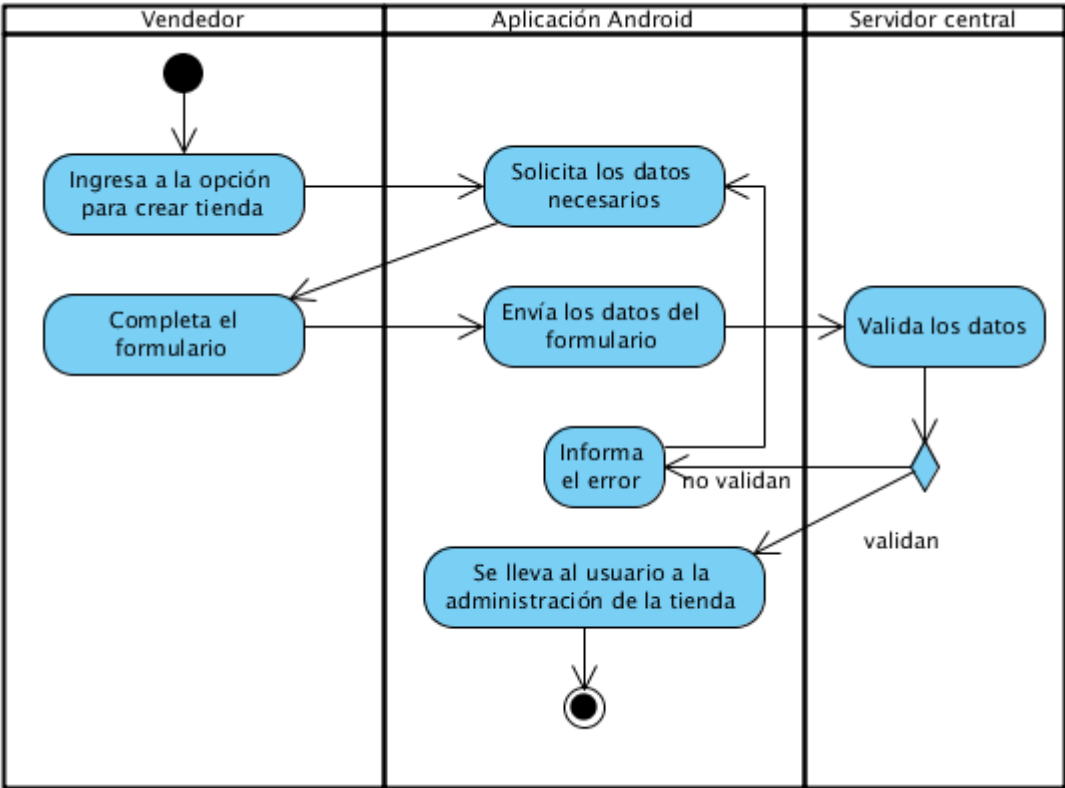
Autenticar



Registrar



Crear Tienda



Comprar Producto

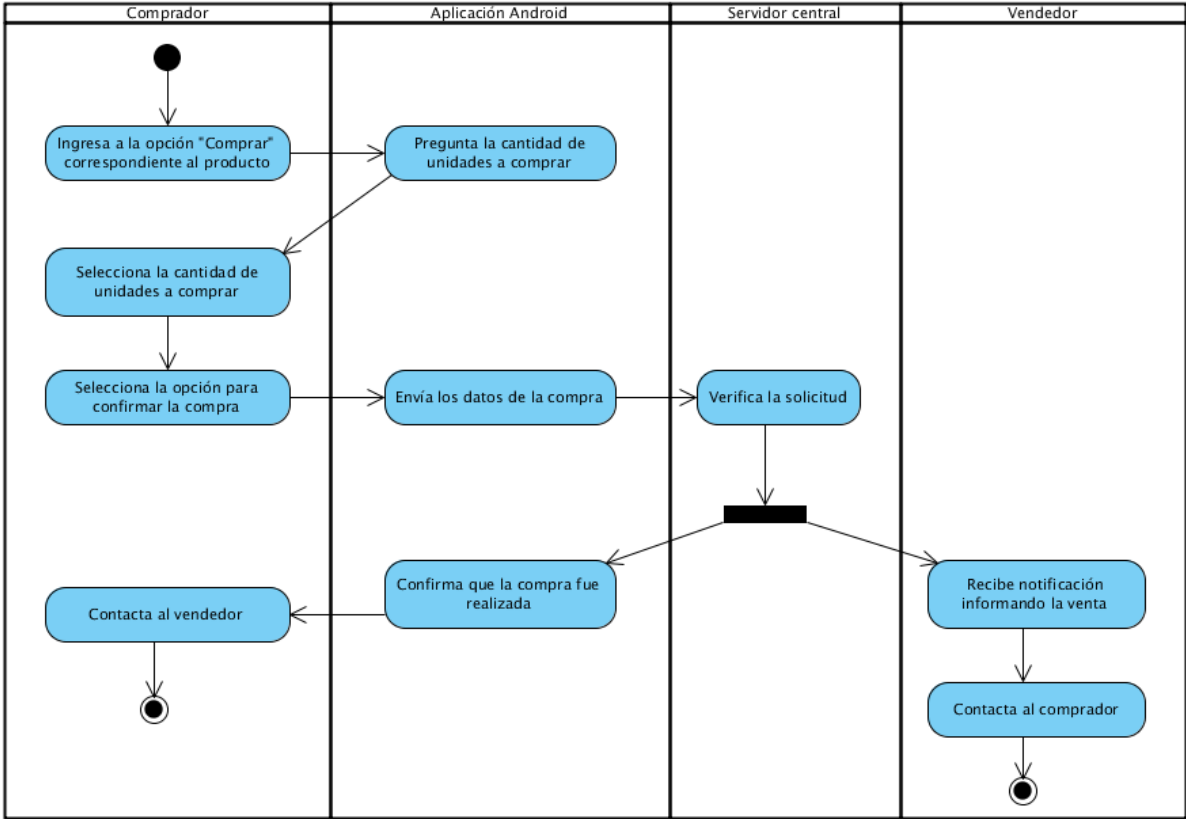
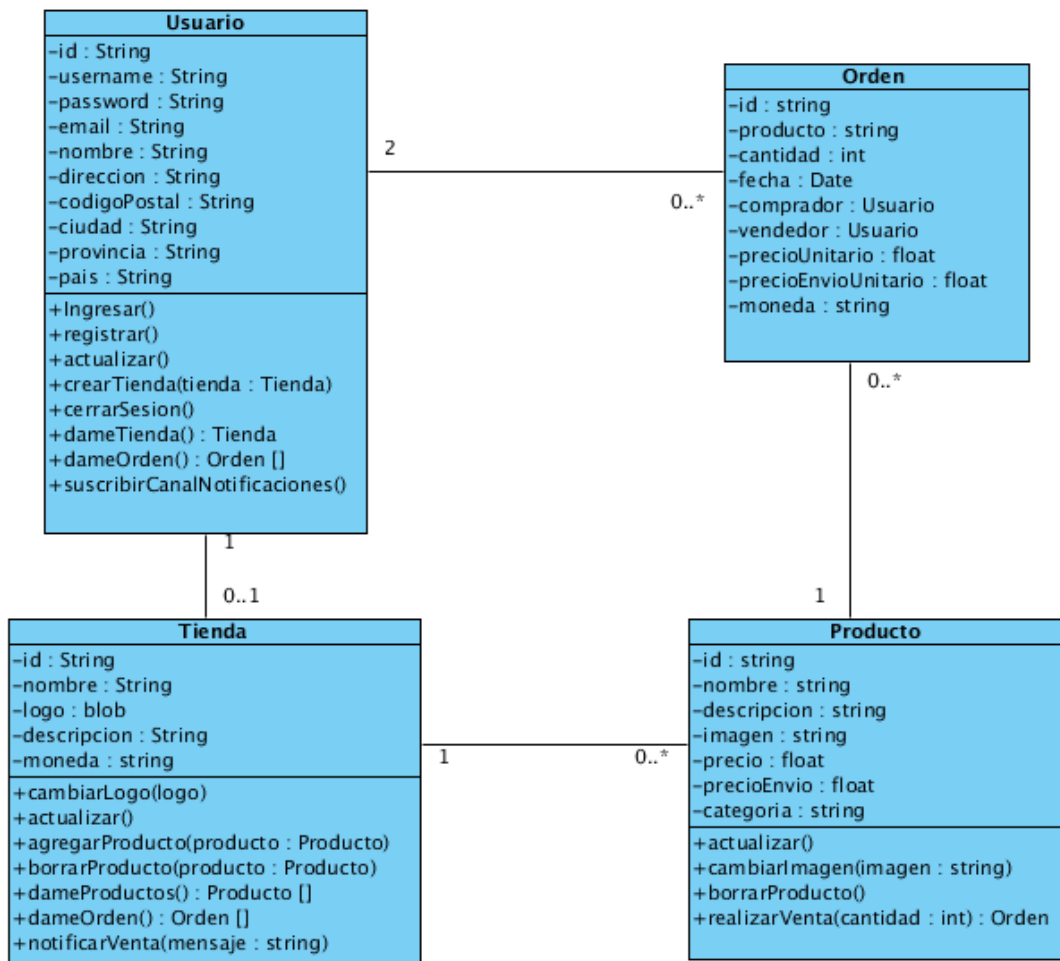


Diagrama de Clases del Dominio



Disciplina de Diseño

Por el Principio de Pareto¹, se hicieron los diagramas relevantes. Como se procura tener homogeneidad en la implementación de las clases, basta un solo diagrama de cada tipo para una clase, para describir también el comportamiento de las otras clases.

Descripción Textual de Casos de Uso

Caso de Uso UC1: Autenticar

Resumen:

Este caso de uso permite a los usuarios autenticarse con el nombre de usuario y contraseña de manera que el sistema les permita realizar operaciones.

Actores:

Usuario (primario). Servidor (en adelante S secundario)

Personal Involucrado y Metas:

Usuario: quiere que el sistema lo reconozca como tal, así pueda realizar las transacciones con la aplicación de un modo seguro y personalizado.

¹Cuando se habla de los costes de desarrollo de software enunciarse de la siguiente manera: “El 80 % del esfuerzo de desarrollo (en tiempo y recursos) produce el 20 % del código, mientras que el 80 % restante es producido con tan sólo un 20 % del esfuerzo”

Servidor: requiere identificar confiablemente a sus usuarios de manera de satisfacer sus intereses en cuanto a seguridad, accesos a su cuenta personal y datos privados.

Precondiciones:

El usuario está registrado

Poscondiciones:

Se identifica y autentica al usuario. Se conocen sus datos personales y opciones de personalización.

Escenario Principal:

1. El usuario ejecuta la aplicación móvil (en adelante AM) en su teléfono celular.
2. AM solicita al usuario su identificador y contraseña.
3. El usuario ingresa su identificador y contraseña.
4. AM solicita a S la validación del usuario.
5. S valida al usuario y comunica sus datos personales.
6. AM da la bienvenida al usuario.

Flujos Alternativos:**A1: El sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

A2: Nombre de usuario inexistente

La secuencia A2 comienza en el punto 4 del escenario principal.

5. S comunica que el nombre de usuario es inexistente.

El escenario vuelve al punto 2.

A3: Nombre de usuario existe pero contraseña inválida

La secuencia A2 comienza en el punto 4 del escenario principal.

5. S comunica que la contraseña es inválida.

El escenario vuelve al punto 2.

Requisitos de Interfaz de Usuario para todos los casos de uso:

Teléfono celular con SO Android y pantalla táctil

Requisitos No-Funcionales para todos los casos de uso:

Tiempo de respuesta: la interfaz debe responder dentro de un tiempo máximo de 10 segundos en una velocidad efectiva de conexión con el servidor a través de 3G.

Disponibilidad: debe poder accederse en un régimen 24x7.

Caso de Uso UC2: Registrar**Resumen:**

Este caso de uso permite a los usuarios registrarse como usuarios de la aplicación, permitiendo introducir sus datos y preferencias personales

Actores:

Usuario (primario). Servidor (en adelante S secundario)

Personal Involucrado y Metas:

Usuario: quiere transformarse en un usuario del sistema, así pueda realizar las transacciones con la aplicación de un modo seguro y personalizado.

Servidor: quiere registrar la mayor cantidad de usuarios posibles y que el proceso sea lo más rápido y seguro posible.

Precondiciones:

El usuario no está registrado en la aplicación.

Poscondiciones:

Se registra al usuario como usuario de la aplicación. El usuario puede realizar operaciones en la aplicación.

Escenario Principal:

1. El usuario ejecuta la AM en su teléfono celular y decide registrarse.
2. AM muestra un formulario de carga donde ingresa sus datos personales y su nombre de usuario y contraseña.
3. AM solicita a S el registro del usuario.
4. S registra al usuario y lo informa a AM.
5. AM da la bienvenida al usuario

Flujos Alternativos:**A1:El sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 3 del escenario principal.

4. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 3.

A2: Nombre de usuario existente

La secuencia A2 comienza en el punto 3 del escenario principal.

4. S comunica que el nombre de usuario es existente.

El escenario vuelve al punto 2.

A3: Contraseña inválida o no coincide con la confirmación

La secuencia A2 comienza en el punto 3 del escenario principal.

4. S comunica que la contraseña es inválida.

El escenario vuelve al punto 2.

A4: dirección de correo electrónico existente

La secuencia A2 comienza en el punto 3 del escenario principal.

4. S comunica que la dirección de correo electrónico es existente.

El escenario vuelve al punto 2.

Caso de Uso UC3: Crear Tienda**Resumen:**

Este caso de uso permite al vendedor crear una tienda de manera que el sistema le permita realizar transacciones y operaciones sobre sus productos.

Actores:

Vendedor (primario). S (secundario)

Personal Involucrado y Metas:

Vendedor: quiere que el sistema lo reconozca como tal, así pueda realizar las transacciones a través del sitio, y la administración de sus productos de un modo seguro y personalizado.

Servidor: requiere identificar confiablemente a sus usuarios vendedores de manera de satisfacer sus intereses en cuanto a seguridad, productos publicados y datos privados.

Precondiciones:

Los usuarios deben estar autenticados en AM.

Poscondiciones:

Se registra una nueva tienda en el sistema con los datos básicos necesarios para realizar ventas.

Escenario Principal:

1. El vendedor selecciona la opción para crear tienda.
2. AM solicita al vendedor a través de un formulario los datos básicos requeridos para la creación de la tienda.
3. El vendedor completa el formulario y presiona un botón para finalizar.
4. AM envía de manera segura los datos a S para que sean validados.
5. S valida los datos, realiza la creación la tienda y envía una confirmación.
6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:

A1:el sistema encuentra algún fallo para comunicarse con S

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC4: Editar Tienda

Resumen:

Este caso de uso permite al vendedor editar los datos de su tienda.

Actores:

Vendedor (primario). S (secundario)

Personal Involucrado y Metas:

Vendedor: quiere editar los datos de su tienda.

Servidor: requiere mantener actualizados los datos del vendedor.

Precondiciones:

El vendedor debe estar autenticado en AM.

Poscondiciones:

Se registran los cambios de los datos de la tienda en el servidor.

Escenario Principal:

1. El vendedor selecciona la opción para editar tienda.
2. AM solicita al vendedor a través de un formulario los datos de la tienda que pueden ser modificados o mantenidos.

3. El vendedor completa el formulario y presiona un botón para finalizar.
4. AM envía de manera segura los datos a S para que sean validados.
5. S valida los datos, realiza la actualización y envía una confirmación.
6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:**A1:El sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC5: Editar Perfil**Resumen:**

Este caso de uso permite al usuario editar los datos de su perfil.

Actores:

Usuario(primario). S (secundario)

Personal Involucrado y Metas:

Usuario: quiere editar los datos de su perfil.

Servidor: requiere mantener actualizados los datos del usuario.

Precondiciones:

El usuario debe estar autenticado en AM.

Poscondiciones:

Se registran los cambios de los datos de perfil en el servidor.

Escenario Principal:

1. El usuario selecciona la opción para editar perfil.
2. AM solicita al usuario a través de un formulario los datos del perfil de usuario que pueden ser modificados o mantenidos.
3. El usuario completa el formulario y presiona un botón para finalizar.
4. AM envía de manera segura los datos a S para que sean validados.
5. S valida los datos, realiza la actualización y envía una confirmación.
6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC6: Crear Producto

Resumen:

Este caso de uso permite al vendedor crear un producto de manera que el sistema le permita ofrecerlo a la venta.

Actores:

Vendedor (primario). S (secundario)

Personal Involucrado y Metas:

Vendedor: quiere ofrecer sus productos a través del sistema.

Servidor: requiere obtener la información de los productos de manera que sea posible ofrecérselo a los compradores.

Precondiciones:

El vendedor debe estar autenticados en AM.

Poscondiciones:

Se crea un nuevo producto en el sistema.

Escenario Principal:

1. El vendedor selecciona la opción para crear producto.
2. AM solicita al vendedor a través de un formulario los datos básicos requeridos para la creación del producto.
3. El vendedor completa el formulario y presiona un botón para finalizar.
4. AM envía de manera segura los datos a S para que sean validados.
5. S valida los datos, realiza la creación del producto y envía una confirmación.

6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al vendedor el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC7: Editar Producto**Resumen:**

Este caso de uso permite al vendedor editar los datos de uno de sus productos.

Actores:

Vendedor (primario). S (secundario)

Personal Involucrado y Metas:

Vendedor: quiere ofrecer sus productos a través del sistema.

Servidor: requiere obtener la información de los productos de manera que sea posible ofrecérselo a los compradores.

Precondiciones:

El vendedor debe estar autenticados en AM.

Poscondiciones:

Se registran los cambios de los datos del producto en el servidor.

Escenario Principal:

1. El vendedor selecciona la opción para editar producto.
2. AM solicita al vendedor a través de un formulario los datos del producto que pueden ser modificados o mantenidos.
3. El vendedor completa el formulario y presiona un botón para finalizar.
4. AM envía de manera segura los datos a S para que sean validados.
5. S valida los datos, realiza la actualización y envía una confirmación.
6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al vendedor el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC8: Eliminar Producto**Resumen:**

Este caso de uso permite al vendedor eliminar un producto de la tienda.

Actores:

Vendedor (primario). S (secundario)

Personal Involucrado y Metas:

Vendedor: quiere eliminar un producto de su tienda.

Servidor: elimina al producto.

Precondiciones:

El vendedor debe estar autenticados en AM.

Poscondiciones:

Se elimina el producto de la tienda.

Escenario Principal:

1. El vendedor selecciona la opción para eliminar el producto.
2. AM solicita al vendedor que confirme que quiere eliminar el producto.
3. El vendedor presiona un botón para confirmar.
4. AM envía de manera segura la solicitud a S para que sean validados.
5. S valida la solicitud, realiza el cambio de estado en el producto y envía una confirmación.
6. AM informa al usuario que la operación se realizó exitosamente.

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al vendedor el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC9: Buscar Producto

Resumen:

Este caso de uso permite a un usuario realizar una búsqueda de productos.

Actores:

Usuario (primario). S (secundario)

Personal Involucrado y Metas:

Usuario: quiere obtener los productos que coincidan con un determinado criterio de búsqueda

Servidor: quiere ofrecer al usuario los productos que mejor se ajustan a su búsqueda.

Poscondiciones:

Se obtiene un listado de productos que cumplen con la condición de búsqueda.

Escenario Principal:

1. El usuario ingresa el texto que se buscará.
2. El usuario presiona un botón para comenzar la búsqueda.
3. AM envía la solicitud de búsqueda a S
4. S verifica los productos que cumplen con el criterio buscado
5. S retorna el listado de productos
6. AM muestra el listado al usuario

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 3 del escenario principal.

5. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 3.

Caso de Uso UC10: Comprar Producto**Resumen:**

Este caso de uso permite a un comprador realizar la compra de un producto.

Actores:

Comprador (primario). Vendedor y S (secundarios)

Personal Involucrado y Metas:

Comprador: quiere realizar la compra del producto en el que está interesado.

Vendedor: quiere que la venta se realice sin inconvenientes y de manera sencilla.

Servidor: quiere que la operación de compra se realice de manera segura y todos los datos relevantes sean almacenados correctamente.

Precondiciones:

Los usuarios deben estar autenticados en AM.

Poscondiciones:

Se registra la compra

Escenario Principal:

1. El comprador selecciona la opción para comprar el producto.
2. El comprador selecciona la cantidad que desea comprar
3. El comprador presiona un botón para confirmar la compra
4. AM envía la solicitud de compra a S
5. S verifica la solicitud y registra la compra
6. AM informa al comprador que la compra se realizó correctamente
7. S envía una notificación al vendedor informándole la compra
8. El vendedor contacta al comprador para que acuerden las condiciones de la compra y la entrega de los productos.

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 4 del escenario principal.

5. AM informa al comprador el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 4.

Caso de Uso UC11: Listar compras realizadas**Resumen:**

Este caso de uso permite a un comprador listar la información de todas las compras que realizó.

Actores:

Comprador (primario). S (secundarios)

Personal Involucrado y Metas:

Comprador: quiere visualizar de manera completa la información de las compras realizadas.

Servidor: quiere que el comprador pueda ver la información relacionada con sus operaciones de manera segura.

Precondiciones:

El comprador debe estar autenticado en AM.

Poscondiciones:

Se muestra un listado de compras realizadas

Escenario Principal:

1. El comprador selecciona la opción para listar las compras realizadas.
2. AM envía la solicitud a S
3. S envía los datos necesarios para generar el listado a AM
4. AM muestra el listado al comprador

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 2 del escenario principal.

3. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 2.

Caso de Uso UC12: Listar ventas realizadas**Resumen:**

Este caso de uso permite a un vendedor listar la información de todas las ventas que realizó.

Actores:

Vendedor (primario). S (secundarios)

Personal Involucrado y Metas:

Vendedor: quiere visualizar de manera completa la información de las ventas realizadas.

Servidor: quiere que el vendedor pueda ver la información relacionada con sus operaciones de manera segura.

Precondiciones:

El vendedor debe estar autenticado en AM.

Poscondiciones:

Se muestra un listado de compras realizadas

Escenario Principal:

1. El vendedor selecciona la opción para listar las ventas realizadas
2. AM envía la solicitud a S
3. S envía los datos necesarios para generar el listado a AM
4. AM muestra el listado al vendedor

Flujos Alternativos:**A1:el sistema encuentra algún fallo para comunicarse con S**

La secuencia A1 comienza en el punto 2 del escenario principal.

3. AM informa al usuario el problema de conexión a través de un mensaje por la pantalla.

El escenario vuelve al punto 2.

Caso de Uso UC13: Contactar Usuario**Resumen:**

Este caso de uso permite a un usuario ponerse en contacto con otro.

Actores:

Usuario remitente (primario). Usuario destinatario (secundario)

Personal Involucrado y Metas:

Usuario remitente: quiere poder comunicarse con otro usuario de la aplicación.

Usuario destinatario: quiere poder recibir las consultas que otros usuarios puedan realizarle.

Precondiciones:

El usuario remitente cuenta con una aplicación de correo electrónico y una cuenta configurada en su teléfono celular.

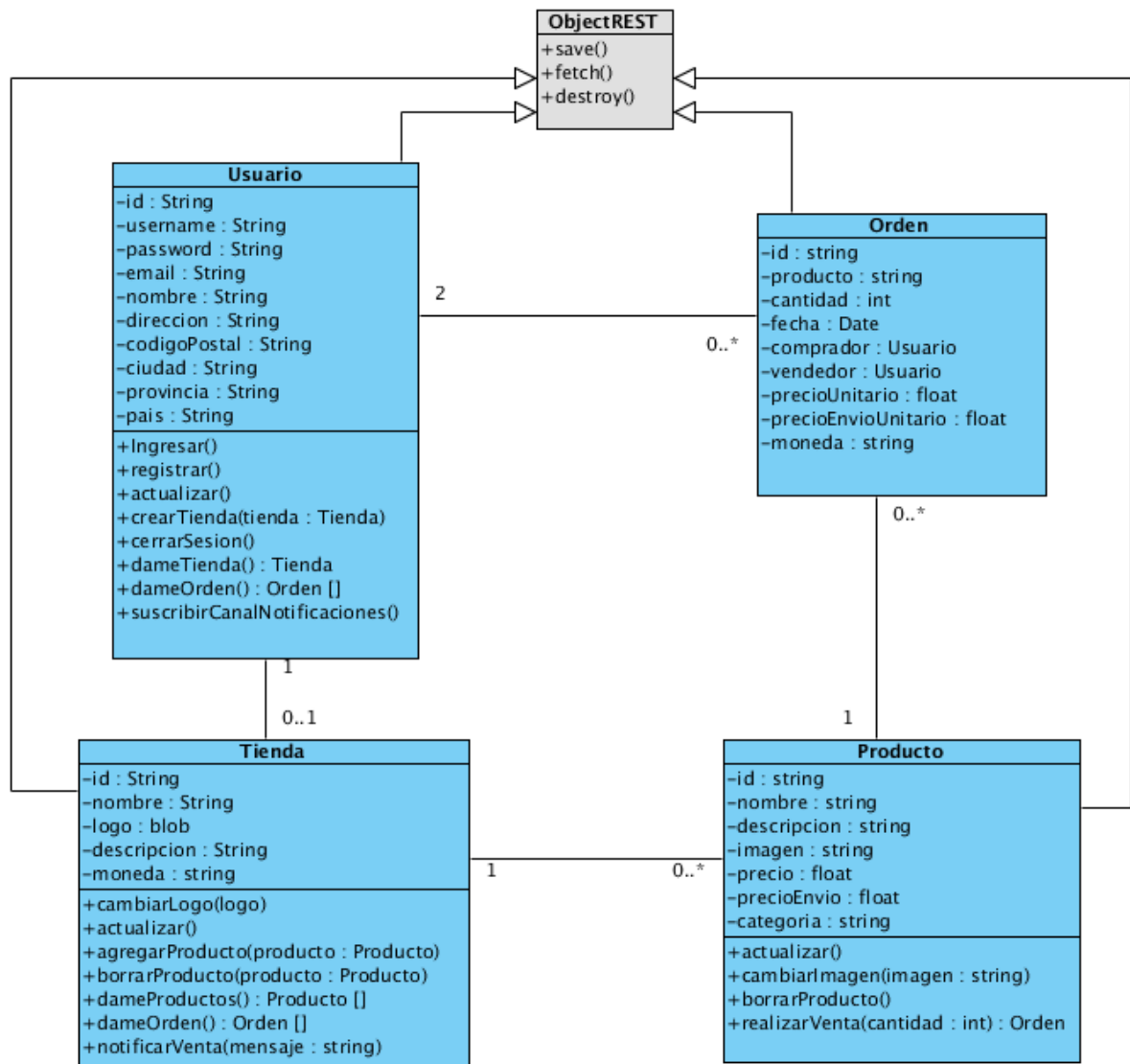
Poscondiciones:

Se envía el mensaje al usuario destinatario a través de e-mail.

Escenario Principal:

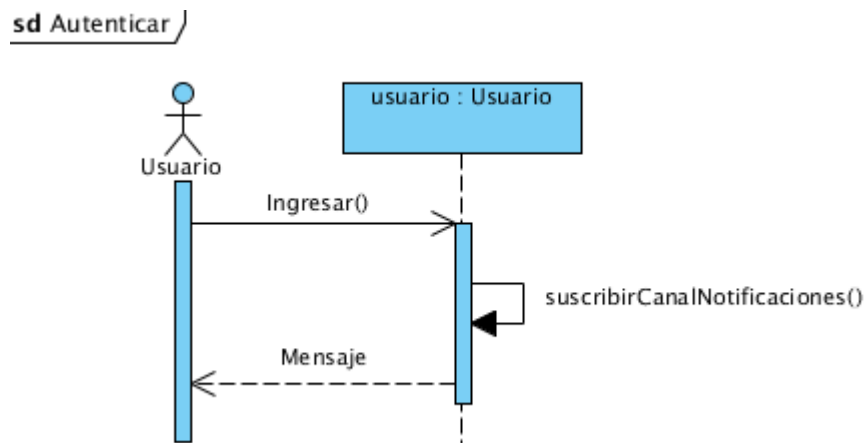
1. El Usuario remitente selecciona la opción para contactar con el Usuario destinatario.
2. AM redirige al usuario remitente a una aplicación de gestión de correo electrónico instalada en el dispositivo, y completa la dirección destino con el e-mail del Usuario destinatario.
3. El Usuario remitente completa el mensaje y realiza el envío

Diagrama de Clases de Diseño

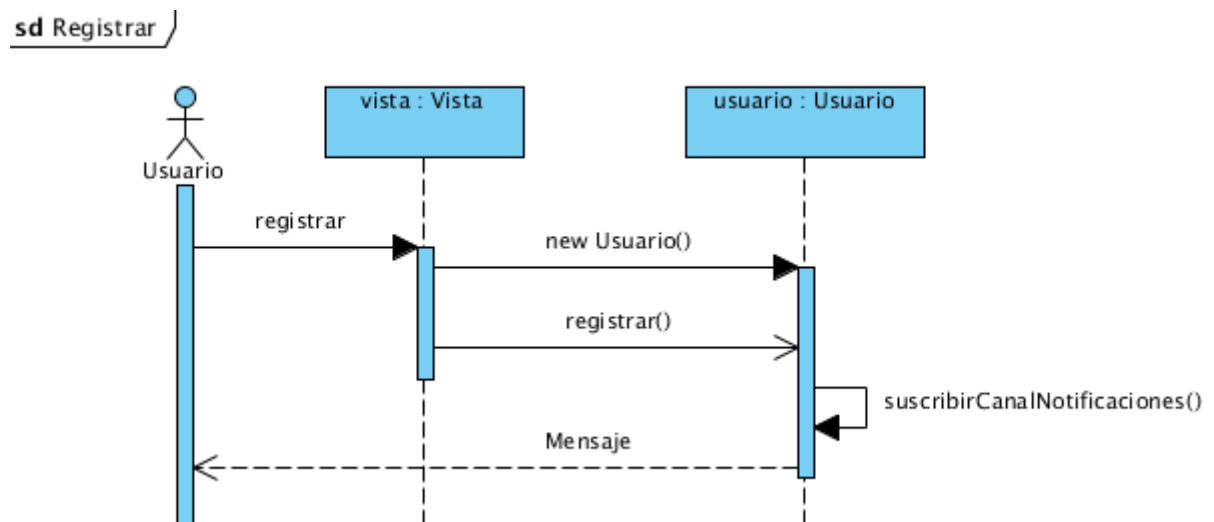


Diagramas de Secuencia

Autenticar

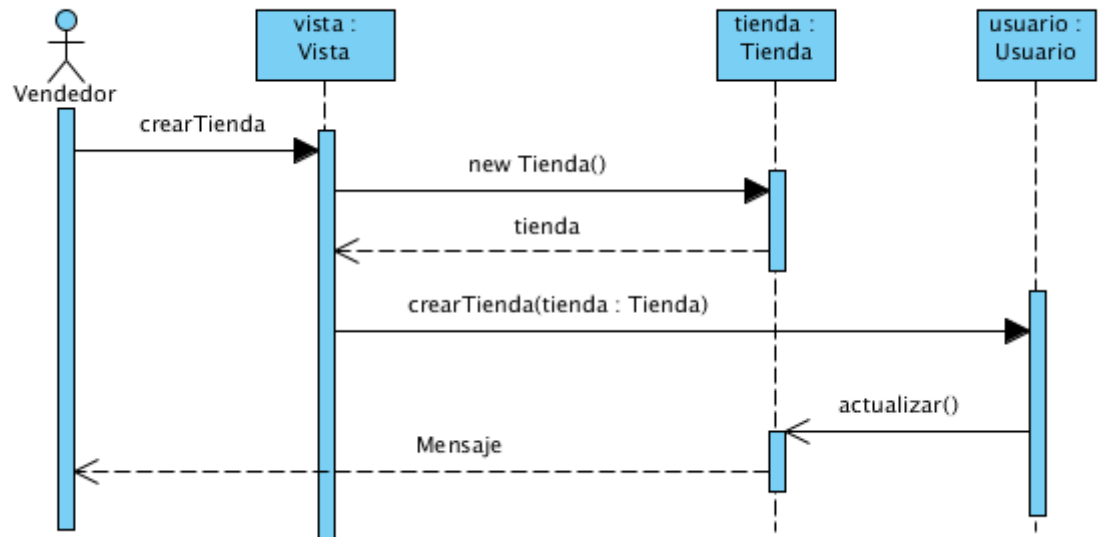


Registrar



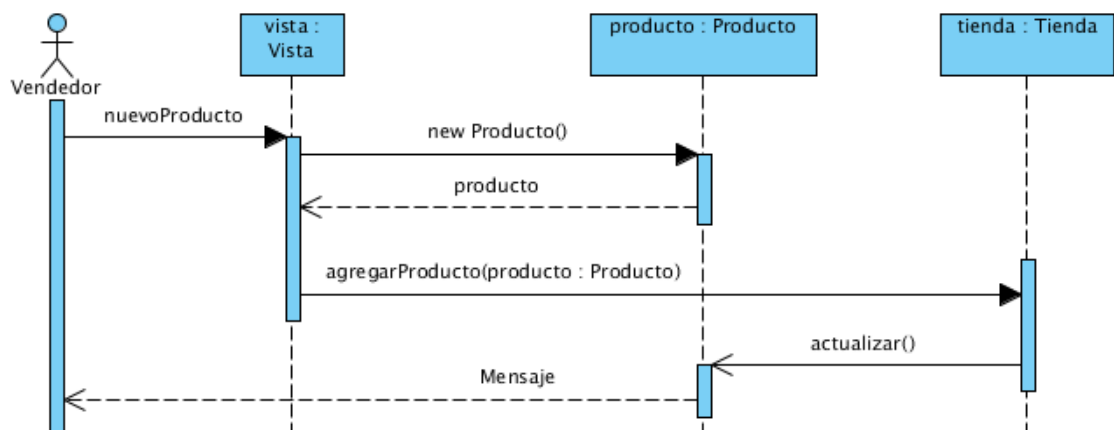
Crear Tienda

sd Crear Tienda



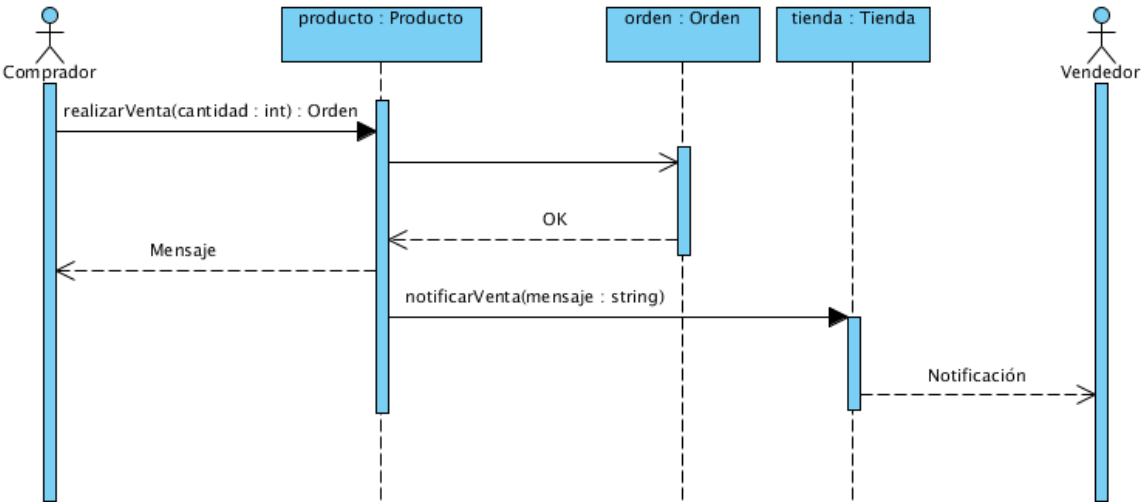
Agregar Producto

sd Agregar Productos



Comprar Producto

sd Comprar Producto



Interfaz de Usuario



Figura 3: Pantalla inicial

3G 9:11

DroidCommerce

Usuario

Contraseña

Ingresar

También podés registrarte:

+ Registrar

Home Tienda Opciones

Figura 4: Pantalla de login

3G 9:11

DroidCommerce

* Nombre de Usuario

* Contraseña

* Confirmar contraseña

* Email

Registrar

Home Tienda Opciones

Figura 5: Pantalla de registro



The screenshot displays the 'DroidCommerce' app interface for creating a new store. At the top, a black status bar shows '3G', a battery icon, and the time '9:13'. Below this, a yellow header bar contains the app name 'DroidCommerce'. The main form area has a light yellow background and includes the following elements: a red asterisk followed by the label 'Nombre' above a text input field; the label 'Moneda:' above three currency buttons ('US\$', 'AR\$', and '€'); the label 'Descripción' above a larger text input field; and the label 'Logo' above two buttons: '+ Agregar' and '✕ Borrar L...'. At the bottom, a yellow navigation bar features three icons and labels: a house icon for 'Home', a grid icon for 'Tienda', and a gear icon for 'Opciones'.


Figura 6: Pantalla de creación de tienda, parte superior



Figura 7: Pantalla de creación de tienda, parte inferior



Figura 8: Pantalla de administración de la tienda



3G 9:19

DroidCommerce

* Nombre

Categoría

Libros, revistas y cómics

Descripción

* Precio

* Precio de Envío

Home Tienda Opciones

Figura 9: Pantalla de creación de producto, parte superior



Figura 10: Pantalla de creación de producto, parte inferior



Figura 11: Pantalla de administración de productos



Figura 12: Pantalla de tienda



Figura 13: Pantalla de resultados de la búsqueda de productos



Figura 14: Pantalla de producto



Figura 15: Pantalla de listado de ventas

Disciplina de Implementación

Arquitectura de la aplicación

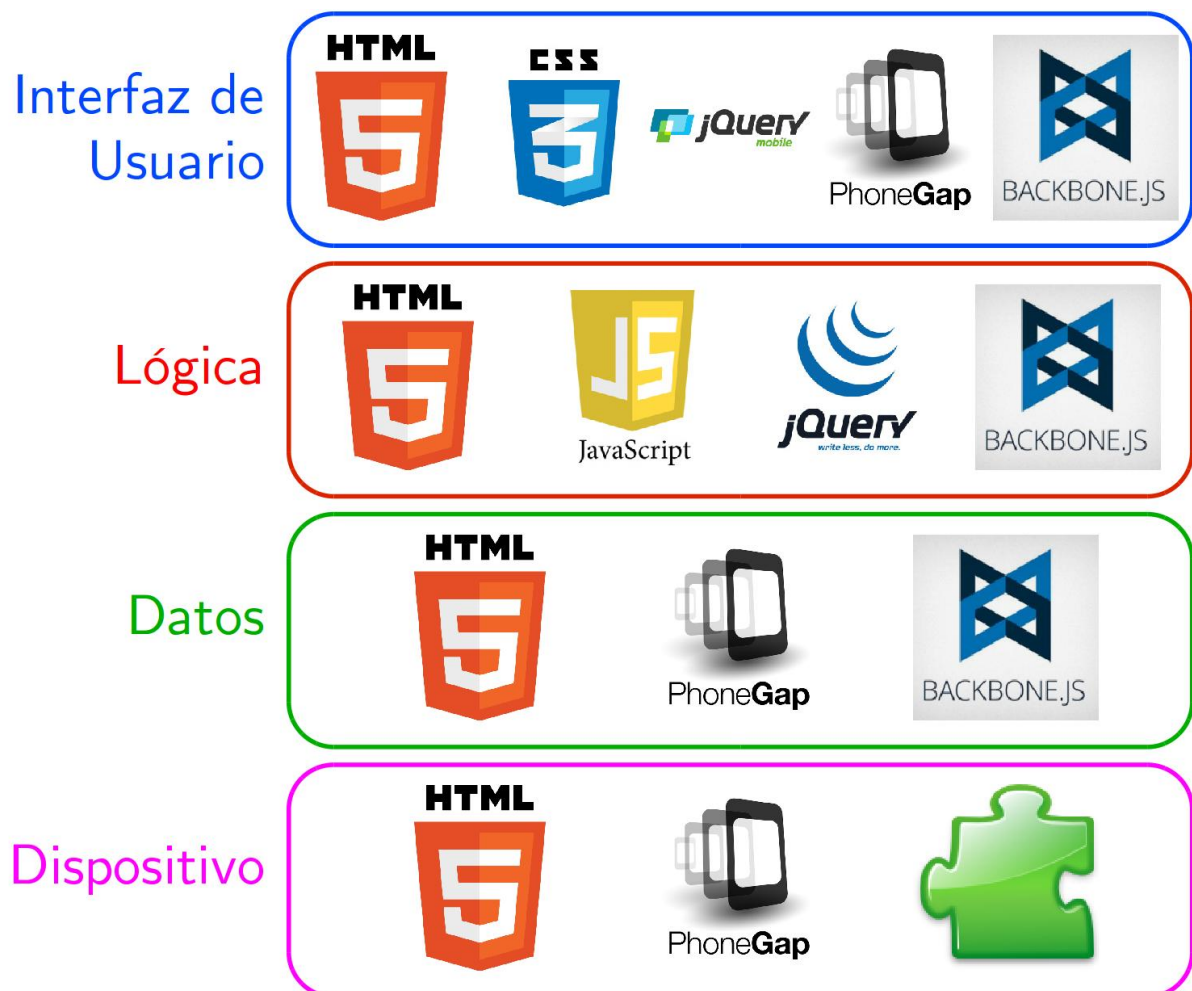
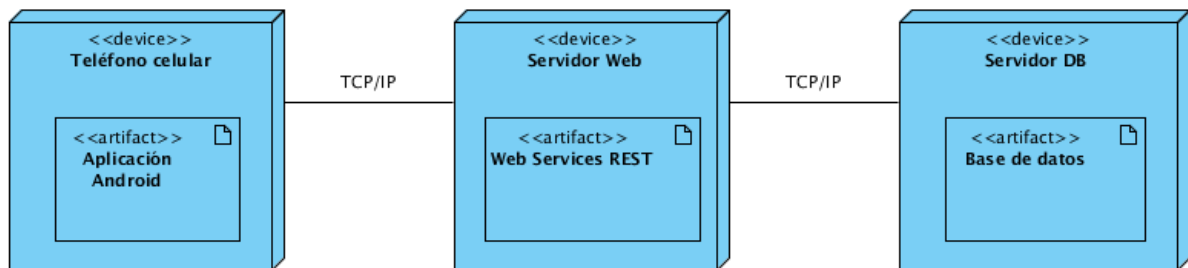


Figura 16: Arquitectura de la aplicación, mostrando los frameworks utilizados

Diagrama de Despliegue



Elección del Lenguaje

Independientemente del paradigma de ingeniería de software, el lenguaje de programación tendrá impacto en la planificación, el análisis, el diseño, la codificación, la prueba y el mantenimiento de un proyecto. Para la construcción de la aplicación se eligió la utilización de los lenguajes web HTML5, CSS3 y JavaScript.

La elección de estos lenguajes para la construcción de la aplicación se debe a las siguientes ventajas que ofrecen:

- *Mayor portabilidad:* Al ser tecnologías estándares y soportadas por la mayoría de los teléfonos celulares modernos, es posible que una misma aplicación sea muy fácilmente adaptable a varias plataformas móviles.
- *Soporte futuro:* Todas las plataformas móviles están trabajando para mejorar el soporte que ofrecen a las tecnologías web, ofreciendo una mejor experiencia al usuario.
- *Aprovechamiento de conocimiento de desarrollo de aplicaciones web:* Desarrollando aplicaciones móviles en HTML5, CSS3 y Javascript es posible aplicar el conocimiento en el desarrollo de aplicaciones web desarrolladas para navegadores en equipos de escritorio.

Arquitectura de Android

Los componentes principales del sistema operativo Android, que pueden verse en la figura 17, son:

- *Aplicaciones:* PRUEBA GLOSARIO PARA SO Sistema Operativo las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. La mayoría de las aplicaciones están escritas en lenguaje de programación Java, aunque existen TCP/IP (Internet Protocol Suite)s para desarrollar aplicaciones en otros lenguajes como HTML, CSS y Javascript o Python.
- *Marco de trabajo de aplicaciones:* los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- *Bibliotecas:* Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.
- *Runtime de Android:* Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para minimizar el consumo de memoria. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".

- *Núcleo Linux*: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.



Figura 17: Arquitectura de Android

PhoneGap

PhoneGap es un **TCP/IP (Internet Protocol Suite)** de código abierto que actúa como un intermediario entre las aplicaciones web y los dispositivos móviles. Permite crear aplicaciones móviles instalables utilizando tecnología web: **Javascript**, **HTML5** y **CSS3**.

Las aplicaciones resultantes no son totalmente nativas, ni puramente basado en la web. La desventaja de que una aplicación sea totalmente nativa es que sólo se podrá utilizar para la plataforma para la que fue realizada, es decir si se hace una aplicación

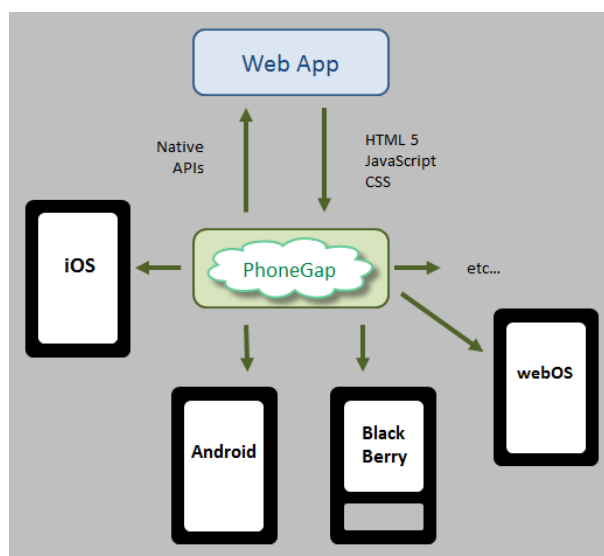


Figura 18: Arquitectura de PhoneGap

para Android luego no se podrá reutilizar el código para hacer la misma aplicación para iOS.

Con PhoneGap se puede reutilizar el código de una aplicación para crear el paquete instalable de cualquiera de las 7 plataformas móviles soportadas: iOS, Android, Blackberry, Windows Phone, WebOS de Palm, Symbian y Bada.

PhoneGap permite acceder a funciones nativas como el acelerómetro, cámara, brújula, contactos, archivos, ubicación geográfica, almacenamiento y notificaciones.

PhoneGap para Android esta dividido en dos partes:

- *Librerías nativas (phonegap.jar)*: Agrega acceso JavaScript para APIs nativas.
- *Archivos javascript (phonegap.js)*: Contenedores JavaScript para llamados de APIs nativas.

Ventajas de PhoneGap

- Soporta 7 plataformas móviles: iOS, Android, Blackberry, Windows Phone, WebOS de Palm, Symbian y Bada.
- Acceso a características nativas de cada plataforma a través de su API, a las que una aplicación web visitada desde el navegador no podría acceder, como acceso a la

cámara de fotos, acelerómetro, notificaciones, etc.

- Permite ejecutar a través de JavaScript plugins escritos en código nativo.
- Permite distribuir aplicaciones realizadas utilizando HTML5 y JavaScript a través de las tiendas de aplicaciones oficiales de cada plataforma.

Desventaja de phoneGap

- Normalmente las aplicaciones realizadas con PhoneGap tienen un menor rendimiento en tareas que requieren alta capacidad de procesamiento, sobre todo en versiones antiguas de las plataformas sobre las que se usa.
- Se pierde la posibilidad de acceder a algunas características nativas, como los diferentes elementos de interfaz de usuario propios de cada plataforma, aunque estos pueden imitarse mediante el uso de CSS.

Web services

Un servicio web (en inglés, Web service) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Las ventajas de los servicios web son:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

Razones para crear servicios Web

La principal razón para usar servicios Web es que se pueden utilizar con HTTP sobre **TCP/IP (Internet Protocol Suite)** en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls -que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.

Otra razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada. Se pueden desarrollar servicios web como parte de una aplicación web, permitiendo acceder a los mismos datos que esta.

REST

REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. REST describe cualquier interfaz web simple que utiliza **TCP/IP (Internet Protocol Suite)** (o **TCP/IP (Internet Protocol Suite)**) y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web **TCP/IP (Internet Protocol Suite)**.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

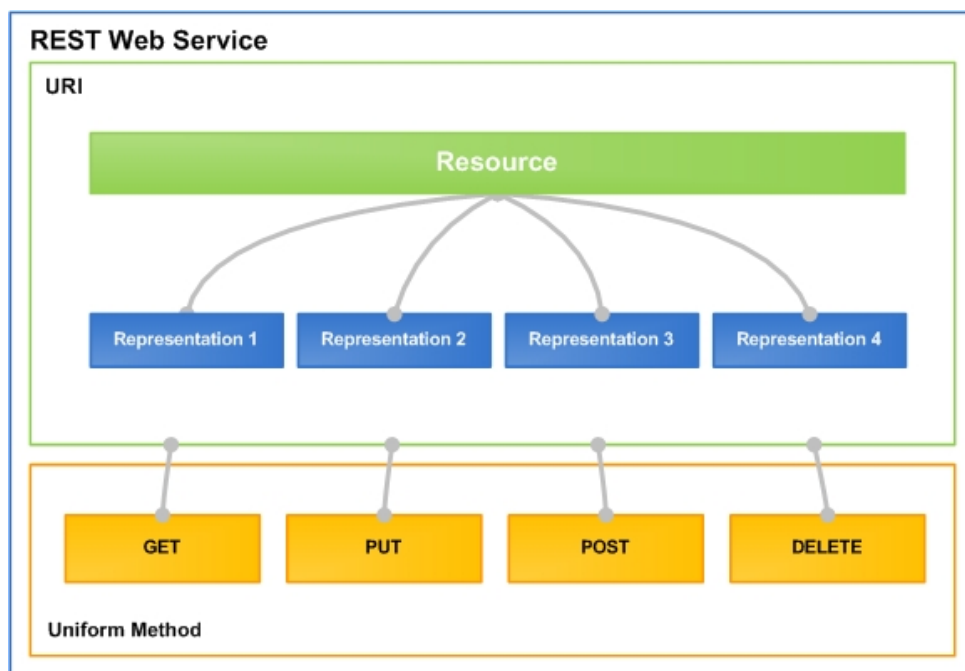


Figura 19: Web Services REST

- *Un protocolo cliente/servidor sin estado:* cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- *Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información:* HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su **TCP/IP (Internet Protocol Suite)**.
- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente **HTML**, XML o JSON. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin

requerir el uso de registros u otra infraestructura adicional.

Recursos

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso).

Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

La petición puede ser transmitida por cualquier número de conectores (por ejemplo clientes, servidores, cachés, túneles, etc.) pero cada uno lo hace sin "ver más allá" de su propia petición. Así, una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, proxys, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento **HTML**, **TCP/IP (Internet Protocol Suite)** o **TCP/IP (Internet Protocol Suite)**, aunque también puede ser una imagen o cualquier otro contenido.

Características de seguridad

Canal de comunicación cifrado

En todos los casos en los que se realiza transferencia de datos confidenciales del usuario, como sus credenciales de acceso, datos personales u operaciones realizadas, es necesario asegurar que tanto las solicitudes y las respuestas se envían a través de un canal de comunicación cifrado.

La utilización de protocolos de comunicación inseguros, como HTTP, pueden hacer que la comunicación pueda ser interceptada utilizando ataques **TCP/IP (Internet Protocol Suite)**, permitiendo que el atacante pueda ver todos los datos intercambiados e incluso manipularlos o generar solicitudes falsas.

Para mitigar este riesgo, las comunicaciones de la aplicación se realizan utilizando el protocolo de comunicación segura **HTTPS**. Este está basado en HTTP, pero utiliza un cifrado basado en **TCP/IP (Internet Protocol Suite)** para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto) más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar, como se muestra en la figura 20.

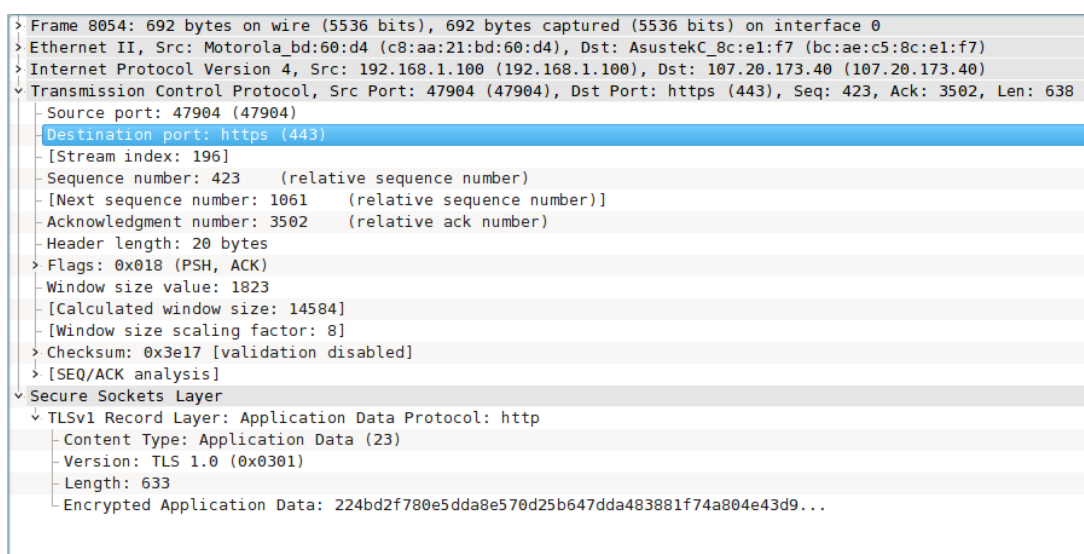


Figura 20: Captura de la solicitud cifrada como lo vería un atacante utilizando Wireshark

ACL para cada objeto

Como ya se habló anteriormente, los Web services pueden ser públicamente accedidos a través de Internet, por lo que es imprescindible contar con un mecanismo que impida la realización de consultas o modificaciones no autorizadas. En la aplicación es necesario que los datos estén definidos correctamente: sólo un vendedor puede realizar la modificación de los datos de una tienda y de sus productos, sólo puede modificar los datos personales el usuario al que pertenece la cuenta activa y las órdenes no pueden modificarse una vez creadas y sólo pueden ser vistas por el usuario comprador o el vendedor.

Para asegurar que los datos sólo son accesibles por usuarios que están autorizados a

leerlos o modificarlos se implementó un **TCP/IP (Internet Protocol Suite)** para cada objeto. Este dato se almacena cuando el objeto es creado, y está representado con un listado en formato **TCP/IP (Internet Protocol Suite)** incluyendo los permisos de los usuarios sobre ese objeto. En el siguiente ejemplo se muestra el caso de un objeto que es visible públicamente pero sólo el usuario con id `idUsuario` puede realizar modificaciones:

```
{
  "idUsuario":
    {
      "read":true,
      "write":true
    },
  "*":
    {
      "read":true
    }
}
```

Esta medida de seguridad debe ser controlada en el servidor web que recibe las peticiones, debido a que controlarlo en el cliente no soluciona el problema de las modificaciones no autorizadas. Una solicitud a un objeto sin usar las credenciales de un usuario autorizado debe ser denegada por el servidor, como se muestra en la figura 21. Es importante también que el mensaje retornado por el servidor, en caso de que el objeto no tenga permisos de visualización por el usuario actual, no revele evidencia de la existencia del objeto.

Backbone.js

Backbone.js es un **TCP/IP (Internet Protocol Suite)** para Javascript con un interfaz RESTful por **TCP/IP (Internet Protocol Suite)**, basada en el paradigma de diseño de aplicaciones Modelo Vista Presentador (MVP). Está diseñado para desarrollar aplicaciones de una única página y para mantener las diferentes partes de las aplicaciones web

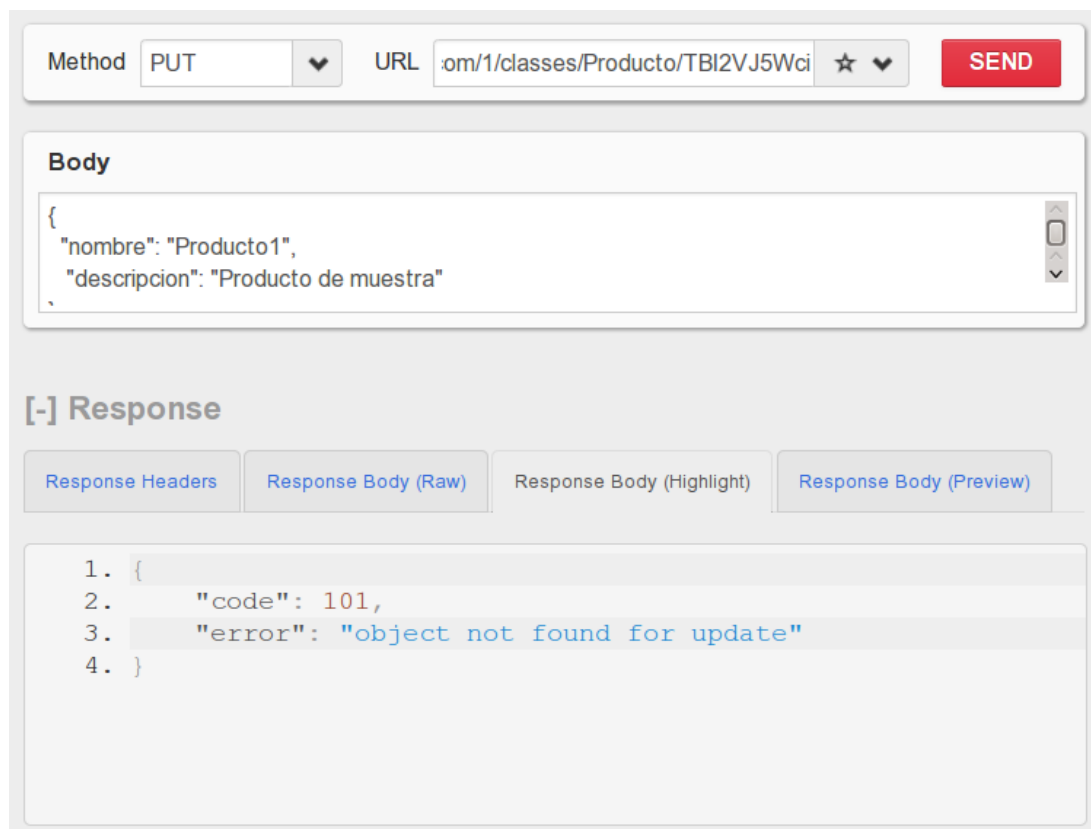


Figura 21: Solicitud de modificación rechazada a un objeto existente utilizando REST Client

(p.e. múltiples clientes y un servidor) sincronizadas.

Backbone.js posee cuatro clases principales:

- Model
- View
- Router
- Collection

Modelo Vista Presentador (MVP)

El patrón Modelo-Vista-Presentador (MVP) surge como una variación del patrón Modelo-Vista-Controlador (MVC).

Los componentes básicos de este patrón son:

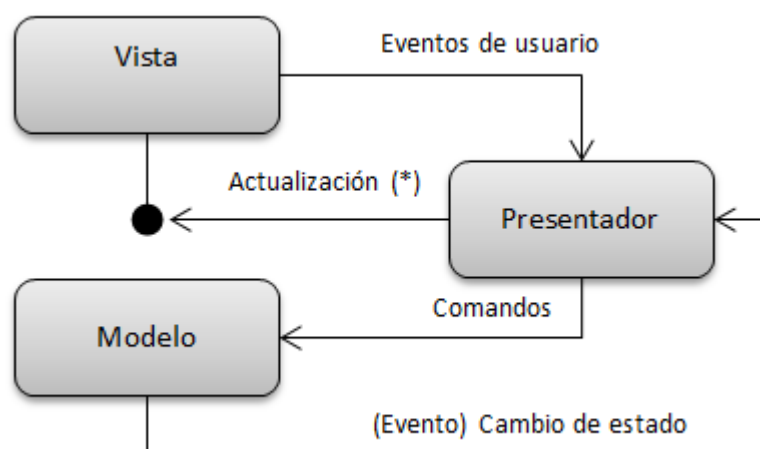


Figura 22: Componentes MVP

- *Modelo*: El modelo es normalmente los datos de la aplicación y la lógica para recuperar y conservar los datos. A menudo, se trata de un modelo de dominio que puede basarse en una base de datos o los resultados de los servicios web. En algunos casos, que el modelo de dominio corresponde perfectamente a lo que se ve en la pantalla, pero en otros casos ha de ser adaptada, agregados o extendido para ser utilizable.

- *Vista:* La vista es típicamente un control de usuario o formulario que combina varios en una interfaz de usuario. El usuario puede interactuar con los controles en la vista
- *Presentador:* El presentador tiene toda la lógica de la vista y es responsable de sincronizar el modelo y la vista. Cuando la vista notifica el presentador que el usuario ha hecho algo (por ejemplo, hacer clic en un botón), el presentador a continuación, actualizar el modelo y sincronizar los cambios entre el modelo y la vista.

jQuery

jQuery es una biblioteca de **Javascript** que permite simplificar la manera de interactuar con los documentos **HTML**, manipular el árbol **TCP/IP (Internet Protocol Suite)**, manejar eventos, desarrollar animaciones y agregar interacción con la técnica **TCP/IP (Internet Protocol Suite)** a páginas web.

Es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y cerrados.

Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y líneas de código.

jQuery Mobile

jQuery Mobile es un **TCP/IP (Internet Protocol Suite)** para Javascript utilizado en el desarrollo de la interfaz de usuario para aplicaciones web adaptadas a dispositivos móviles. Incluye elementos de UI y soporte a eventos relacionados con el uso de pantallas táctiles.

Las principales características de jQuery Mobile son:

- Es compatible con otros frameworks que utilizamos para el desarrollo de la aplicación móvil, tales como Phonegap o Backbone.js.
- Es compatible con las principales plataformas móviles, así como todos los navegadores de escritorio principales, incluyendo Android.

- Construido sobre jQuery.
- Permite la creación de temas personalizados mediante el agregado de estilos CSS. Proporciona además una base de temas que permite a los desarrolladores personalizar las combinaciones de colores y determinados aspectos de las características de interfaz de usuario.
- Tiene mínimas dependencias, aumentando la velocidad y reduciendo el consumo de memoria.
- Adaptación automática del diseño al tamaño de la pantalla.

Herramientas de desarrollo

- *Eclipse*: Es un entorno de desarrollo integrado (IDE) de código abierto multiplataforma.
- *Visual Paradigm para UML*: Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

- *L^AT_EX*: Es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.

L^AT_EX facilita el uso del lenguaje de composición tipográfica. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con L^AT_EX es comparable a la de una editorial científica de primera línea.

- *ShareLaTeX*: Es un editor online (en tiempo real) de L^AT_EX, que puede ser utilizado por varios usuarios a la vez.

- *Dropbox y Google Drive:* Son servicios de alojamiento de archivos multiplataforma en la nube, operado por las compañías Dropbox y Google.

Permiten a los usuarios almacenar y sincronizar archivos en línea y entre computadoras y compartir archivos y carpetas con otros.

Disciplina de Pruebas

Test de Unidades

Introducción

El Test de Unidades consiste en realizar pruebas de las unidades individuales de código. En esta fase se realizan las pruebas de caja blanca.

Pruebas de Caja Blanca

Es un tipo de método de prueba que permite detectar errores internos del código de cada módulo.

Con estas pruebas se pueden garantizar que se ejercitan por lo menos una vez todos los caminos independientes de cada módulo, que las decisiones lógicas se evalúan en sus dos variantes (verdadera y falsa), que se ejecutan todos los bucles en sus límites operacionales y que se ejercitan las estructuras internas de datos para asegurar su validez.

Test de Módulos

Introducción

El Test de Módulos consiste en realizar pruebas de los módulos funcionales del sistema. En esta fase se realizan las pruebas de caja negra y las pruebas de estrés.

Pruebas de Caja Negra

En este método de prueba se ve a cada módulo como una caja negra y se generan conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa, observando las salidas.

Con estas pruebas se pueden detectar funciones incorrectas o ausentes, errores de interfaz, errores de rendimiento, etc.

Pruebas de Estrés

Esta prueba se centra en realizar el análisis de valores límites, y en condiciones límites, ya que se ha demostrado que los errores tienden a darse más en los límites del campo de entrada y sometidos a condiciones límites.

Test de Integración

Introducción

El Test de Integración consiste en realizar pruebas de la estructura modular del programa y su interacción a través de la prueba de integración.

Pruebas de Integración

En este tipo de prueba los errores surgen al integrar los módulos. En esta fase se pueden detectar errores como por ejemplo que las subfuncion, es cuando se combinan pueden no producir la función principal, un módulo puede tener un efecto adverso e inadvertido sobre otro, etc.

El objetivo es tomar los módulos probados y construir una estructura de programa que esté de acuerdo con lo que dicta la especificación C.

Existen dos tipos de integración:

Integración descendente: En este tipo se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando con el módulo de control inicial.

Integración ascendente: En este tipo se integran los módulos atómicos primero y luego se continúa con el nivel inmediato superior.

En el desarrollo de este sistema se utilizó la integración descendente.

Test de Aceptación

Introducción

El Test de Aceptación consiste en realizar la prueba del software para validar si funciona de acuerdo con las expectativas razonables del cliente. En esta fase se llevan a cabo las pruebas Alfa y Beta.

Prueba Alfa

Esta prueba es conducida por el cliente en el lugar de desarrollo. Se usa el software de forma natural (previa capacitación), con el encargado de desarrollo mirando “por encima del hombro” del usuario y registrando errores y problemas de uso. Se lleva a cabo en un entorno controlado.

Prueba Beta

Esta prueba se lleva a cabo en uno o más lugares de clientes, por los usuarios finales de software. El encargado de desarrollo no está presente. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba o informa a intervalos regulares al equipo de desarrollo. Se lleva a cabo en un entorno no controlado.

Los procedimientos de prueba se diseñaron para asegurar que se satisfacen todos los requisitos funcionales y que se alcanzan todos los requisitos de rendimiento.

Conclusiones

El desarrollo de nuestro proyecto final nos permitió poner en práctica temas que aprendimos en distintas asignaturas durante el transcurso de nuestra carrera, el aprendizaje y la experiencia de implementar nuevas tecnologías, enfrentándonos a problemas reales de diseño e integración que nos forzaron a tomar decisiones a fin de encontrar soluciones eficientes a los mismos. Además, nos dio la posibilidad de aprender a trabajar con herramientas con los cuales no estábamos familiarizados.

Con la puesta en funcionamiento de este proyecto damos valor agregado a la aplicación web existente cumpliendo con los objetivos planteados inicialmente, obteniendo un producto seguro, escalable, de fácil mantenimiento y simple de usar.

Se logró implementar a través de Web services una comunicación eficiente con la base de datos, con mecanismos de seguridad que impiden la visualización o alteración indebida de los datos sensibles de los usuarios.

Vimos la importancia del uso de un desarrollo modular para la realización de proyectos flexibles, escalables, más legibles y manejables en aplicaciones actuales.

También podemos decir que este sistema queda abierto a la incorporación de futuras actualizaciones para satisfacer las necesidades de los usuarios cuando los procesos así lo requieran.

Con respecto a la estimación del tiempo necesario para el desarrollo realizada al inicio del proyecto, cuyo resultado fue de alrededor de 1500 horas hombre, al finalizar el desarrollo del mismo estimamos haber dedicado alrededor de 1300 horas hombre quedando dentro del margen de error del método utilizado.

Glosario

CSS

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML.. [79](#), [80](#), [81](#)

HTML

HTML, siglas de HyperText Markup Language («lenguaje de marcado de hipertexto»), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (`<i>`,`<i>`). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo, JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.. [79](#), [80](#), [81](#), [85](#), [86](#), [91](#)

HTTPS

Hypertext Transfer Protocol Secure (o Protocolo seguro de transferencia de hipertexto), más conocido por sus siglas HTTPS, es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hiper Texto, es decir, es la versión segura de HTTP.. [86](#)

Java

Java es un lenguaje de programación publicado en el 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. El lenguaje deriva mucho de su sintaxis de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede correr en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora. Java es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir del 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.. 80

Javascript

JavaScript es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web aunque existe una forma de JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.. 79, 80, 81, 91

Linux

Linux es un núcleo libre de sistema operativo basado en Unix. Es uno de los principales ejemplos de software libre. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo.. 80

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.. 80

Sistema Operativo

Un sistemaasdasdad operativo (SO, frecuentemente OS, del inglés Operating System) es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes.. 80

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 91

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 91

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 87

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 86

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 86

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 85

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 84

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 84, 86, 87, 88

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 84, 86

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 84

TCP/IP (Internet Protocol Suite)

Es un conjunto de protocolos de comunicación que se utiliza en Internet y en otras redes similares. Sus componentes más importantes son TCP e IP, y se encuentran en representados en un modelo de capas, que van desde la capa de enlace hasta la capa de aplicación, pasando por la capa de Internet y la de transporte.. 80, 81, 88, 91

Bibliografía

- [1] Craig Larman. Applying UML and Patterns second edition.
- [2] Maximiliano Odstreil. Apuntes de Clase Ingeniería de Software I. 2009 .
- [3] Maximiliano Odstreil. Apuntes de Clase Ingeniería de Software II. 2010.
- [4] Documentación oficial de PhoneGap. (<http://phonegap.com/>)
- [5] Thomas Myer. Beginning PhoneGap
- [6] Documentación oficial de Backbone.js. (<http://backbonejs.org/>)
- [7] Documentación oficial de jQuery Mobile. (<http://jquerymobile.com/>)
- [8] Maximiliano Firtman. jQuery Mobile: Up and Running.
- [9] Roger S. Pressman. Software Engineering sixth edition.
- [10] Wikipedia en inglés (<http://en.wikipedia.org/>).
- [11] Wikipedia en castellano (<http://en.wikipedia.org/>).
- [12] Wikilibros: Manual de \LaTeX .
- [13] Gabriel Valiente Feruglio. Composición de textos científicos con \LaTeX . 1999.
- [14] átopos. \LaTeX para Humanidades. 28 de Noviembre de 2005.