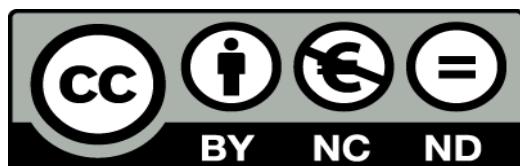




UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Desarrollo e implementación de una solución completa para empresas de mensajería/paquetería utilizando dispositivos Android y la nube de dispositivos de Digi-aplicación web
Autor/es
César Estebas Gómez
Director/es
Angel Luis Rubio García
Facultad
Facultad de Ciencias, Estudios Agroalimentarios e Informática
Titulación
Grado en Ingeniería Informática
Departamento
Curso Académico
2012-2013



Desarrollo e implementación de una solución completa para empresas de mensajería/paquetería utilizando dispositivos Android y la nube de dispositivos de Digi-aplicación web, trabajo fin de grado de César Estebas Gómez, dirigido por Angel Luis Rubio García (publicado por la Universidad de La Rioja), se difunde bajo una Licencia Creative Commons Reconocimiento-NoComercial-SinObraDerivada 3.0 Unported. Permisos que vayan más allá de lo cubierto por esta licencia pueden solicitarse a los titulares del copyright.



UNIVERSIDAD DE LA RIOJA

Facultad de Ciencias, Estudios Agroalimentarios e Informática
Departamento de Matemáticas y Computación

Trabajo Fin de Grado Grado en Ingeniería Informática

Desarrollo e implementación de una solución completa
para empresas de mensajería/paquetería utilizando
dispositivos Android y la nube de dispositivos de Digi –
Aplicación Web

Autor: César Estebas Gómez
Tutor: Ángel Luis Rubio García

Resumen

El presente trabajo consiste en la colaboración en un proyecto de desarrollo del software de gestión (demo) de una empresa de mensajería/paquetería, mediante el uso de la nube de dispositivos de Digi y hardware estándar basado en Android.

Las tareas a realizar en el proyecto son las siguientes:

- Definición y diseño de arquitectura del sistema.
- Diseño e implementación de la aplicación Android capaz de realizar las tareas más típicas en una empresa de paquetería, dotándola de comunicación bidireccional con la nube de dispositivos de Digi.
- Diseño e implementación del portal web que permita a los clientes efectuar nuevos pedidos y verificar el estado de "mis pedidos".
- Elaboración de documentación de las distintas aplicaciones y vídeo demostrativo.

Al ser un proyecto de colaboración, este se divide en dos partes. Una parte consiste en el desarrollo de la aplicación Android y la otra en el desarrollo de la aplicación web. La parte a realizar en este trabajo es la aplicación web a la que se podrá acceder desde la dirección <https://bybike-delivery.appspot.com>

Summary

The present work consists of the collaboration in a software development project (demo) of a courier / packaging company, using Digi's devices cloud and Android-based standard hardware.

The tasks to do in the project are:

- Definition and design of the system architecture.
- Design and implementation of Android application able to perform typical tasks in a delivery company, giving it bidirectional communication with Digi's cloud devices.
- Design and implementation of web portal that allows customers to make new orders and check the status of "my orders".
- Preparation of documentation for the applications and demonstrative video.

As a collaborative project, this is divided into two parts. One part consists of the Android application development and the other in the web application development. The part of the project to be made in this work is the web application which can be accessed from the address <https://bybike-delivery.appspot.com>

Índice

Resumen	1
Summary	2
Índice.....	3
Introducción.....	5
Contexto del trabajo	5
Objetivos del trabajo.....	6
Fases del trabajo	7
Calendario de trabajo.....	7
Horas planificadas para cada tarea	8
Análisis	9
Análisis de requisitos.....	9
Casos de uso	11
Tecnologías.....	15
Diseño	17
Arquitectura del sistema	17
Diseño arquitectónico	17
Patrón arquitectónico	18
Modelo de datos	18
Navegación	22
Servicios Web	23
Diseño de la interfaz.....	24
Implementación	26
Sprint 1 (27/02 – 13/03)	27
Sprint 2 (13/03 – 27/03)	31
Sprint 3 (27/03 – 10/04)	35
Sprint 4 (10/04 – 24/04).....	38

Pruebas	43
Realización del video promocional	45
Conclusiones	46
Comparativa de horas planificadas/reales.....	46
Cumplimiento de objetivos	47
Posibles mejoras y ampliaciones.....	48
Conclusión personal	48
Bibliografía	49

Introducción

El trabajo que describe esta memoria consiste en el desarrollo de una aplicación web de una empresa de paquetería en la que los clientes podrán realizar y hacer seguimiento de sus pedidos. Esta aplicación web se comunicará mediante la nube de dispositivos de Digi con los terminales Android de los trabajadores para que la administración pueda enviarles mensajes o que la aplicación les indique nuevas tareas que tienen que realizar.

Contexto del trabajo

Entre los meses de Septiembre y Diciembre de 2012 realicé las prácticas externas en la empresa Digi International. Allí se me encomendó junto con otro compañero la realización de un proyecto que consistía en desarrollar un software de gestión de una empresa ficticia de paquetería. El software debía tener dos partes comunicadas mediante la nube de dispositivos de Digi:

- Una aplicación Android para uso del trabajador.
- Una aplicación web para la administración de la empresa y para los clientes.

Decidimos que yo me encargaría de la parte Web así que en los meses de prácticas he desarrollado una aplicación web, alojada en Google App Engine, para la administración de la empresa. En ella se pueden realizar las siguientes tareas:

- Consultar la información de los pedidos y trabajadores de la empresa.
- Saber qué trabajadores están conectados, ver su localización en un mapa y enviarles mensajes.
- Obtener información de tareas o trabajadores a través de los servicios web de la aplicación.

Objetivos del trabajo

El objetivo general de este trabajo es desarrollar un sistema completo y fácil de usar que se encargue de la logística necesaria para el buen funcionamiento de una empresa de paquetería a pequeña escala.

Otros objetivos también importantes, relativos al sistema global, son:

- Comunicación bidireccional entre las dos plataformas (Web y Android), ya sea directamente entre ellas o a través de la nube de dispositivos.
- Demostración del funcionamiento del sistema completo al personal de la empresa Digi.
- Comprobación del correcto funcionamiento del sistema mediante la realización de tests, llevados a cabo por los dos alumnos y por el equipo de test de la empresa Digi.
- Realización de un vídeo promocional, donde se expliquen las principales características del sistema.

En cuanto a la aplicación Web, los objetivos son:

- Simulación completa de la experiencia que tendría un cliente al realizar un pedido a una empresa de paquetería.
- Desarrollo de los diferentes servicios web que utilizará la aplicación móvil para comunicarse con la aplicación web.
- Generación de facturas digitales al finalizar un pedido y envío de estas al correo electrónico del cliente.

Fases del trabajo

Al tratarse de un proyecto software las fases en las que se divide su desarrollo son las siguientes:

- Análisis de requisitos.
- Diseño y arquitectura.
- Implementación.
- Pruebas.

Además de estas tareas también debe planificarse tiempo para redactar la memoria del trabajo y la realización de un video promocional que aparecerá en la página de inicio de la aplicación web.

Calendario de trabajo

El trabajo se realizará entre los días 21 de enero de 2013 y 9 de mayo de 2013 y debe corresponder a unas 300 horas de trabajo.

En este periodo de tiempo, se llevarán a cabo reuniones mensuales con el tutor del proyecto y reuniones quincenales con el tutor de empresa para revisar y controlar cómo evoluciona el trabajo.

Puede verse en el siguiente calendario el tiempo que se dedicará a cada fase del proyecto.

ENERO

L	M	X	J	V	S	D
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

FEBRERO

L	M	X	J	V	S	D
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

MARZO

L	M	X	J	V	S	D
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

ABRIL						
L	M	X	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAYO						
L	M	X	J	V	S	D
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Calendario de trabajo

Leyenda:

- Análisis de requisitos
- Diseño y arquitectura
- Implementación
- Pruebas
- Realización de video promocional
- Repaso y finalización de la memoria
- Fin del trabajo

Horas planificadas para cada tarea

En la siguiente tabla se muestra la relación de horas planificadas para la realización de cada tarea incluyendo las reuniones:

TAREA	DÍAS	HORAS AL DIA	TOTAL DE HORAS
Análisis	15	3	45
Diseño	12	4	48
Implementación	40	4	160
Pruebas	5	3	15
Realización de video promocional	3	3	9
Repaso y finalización de la memoria	3	3	9
Reuniones			14
TOTAL DE HORAS DE TRABAJO			300

Análisis

Análisis de requisitos

Requisitos funcionales

Generales:

- La aplicación estará disponible en inglés y en español.

Clients:

- Los visitantes de la página podrán solicitar información sobre la empresa a través de un formulario de contacto.
- Se podrá hacer seguimiento de pedidos sin necesidad de ser cliente del sistema, simplemente se tendrá que conocer el identificador del pedido.
- Los visitantes de la página podrán registrarse en el sistema rellenando un formulario de registro y eligiendo un nombre de usuario y contraseña.
- Para acceder al área de clientes será necesario identificarse en el sistema mediante nombre de usuario y contraseña.
- Un cliente podrá realizar nuevos pedidos eligiendo entre varias opciones de entrega. Dichas opciones de entrega estarán sujetas a disponibilidad horaria, ya que el horario de entrega de la empresa es de 10:00 a 20:00.
- Un cliente podrá conocer el estado y localización de sus pedidos en curso.
- Un cliente podrá cambiar todos sus datos personales excepto el nombre de usuario.
- Un cliente podrá acceder a un listado con su histórico de pedidos y ver en detalle cada uno de ellos.
- Cuando termine un pedido, se generará una factura digital y se enviará al cliente por correo electrónico.

Administrador:

- El administrador podrá acceder a un listado con la información básica de los trabajadores. Seleccionando uno de ellos accederá a toda la información del trabajador.
- El administrador podrá localizar a los trabajadores conectados en todo momento y enviarles mensajes.
- El administrador podrá acceder a un listado con el histórico de pedidos de la empresa y ver en detalle cada uno de ellos.
- El administrador podrá ver los mensajes de solicitud de información que han enviado los visitantes de la página y responder a cada uno de ellos.

Requisitos no funcionales

Requisitos de seguridad:

- Las conexiones a la página se realizarán mediante el protocolo HTTPS.
- Las contraseñas se almacenarán codificadas en el almacén de datos.
- Los servicios web solo admitirán peticiones de tipo POST.
- Habrá filtros de acceso al área de clientes y a la de administración.
- Debe asegurarse que los mecanismos de seguridad no ralenticen notablemente la interacción con la aplicación.

Requisitos de rendimiento:

- La aplicación funcionará correctamente en los principales navegadores:
 - Google Chrome.
 - Internet Explorer.
 - Mozilla Firefox.
 - Safari.

Requisitos de disponibilidad:

- Cuando se realicen paradas en el servicio se garantizará que los datos no se perderán ni modificarán.

Requisitos de fiabilidad:

- La aplicación debe ser capaz de recuperarse de los fallos y no perder información.
- Se debe garantizar la fiabilidad en la identificación de usuarios.
- Siempre que se produzca algún error debe ser notificado al usuario para que este conozca el estado en el que se encuentra la aplicación.

Requisitos de Interfaz y Facilidad de Uso:

- La interfaz debe ser clara y fácil de usar tanto por parte de los clientes como por el administrador.
- La interfaz mantendrá el estilo de los menús, colores y contenedores de forma que se mantenga la consistencia.

Casos de uso

En la aplicación existirán tres tipos de usuarios:

- **Visitante:** Un visitante será cualquier usuario de la aplicación que no está registrado en el sistema como cliente o administrador.
- **Cliente:** Los clientes estarán registrados en el sistema y podrán realizar tareas como localizar sus pedidos en curso o realizar un nuevo pedido.
- **Administrador:** Será el encargado de la administración de la empresa. Se comunicará con los trabajadores, los localizará en el mapa y podrá acceder al histórico de pedidos de la empresa.

Teniendo en cuenta los diferentes tipos de usuarios se ha obtenido el siguiente diagrama de casos de uso:

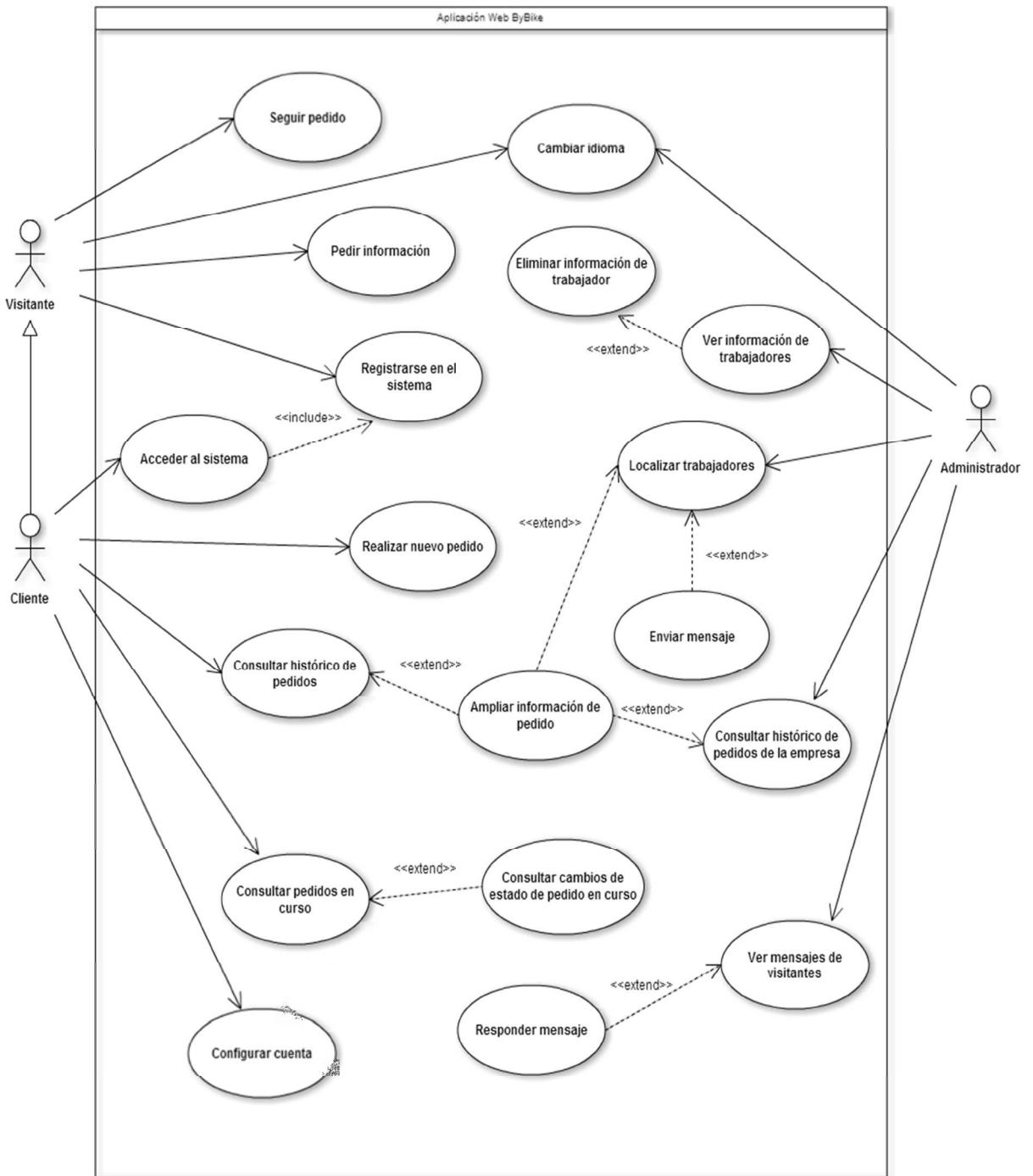


Diagrama de casos de uso

Especificación de casos de uso:

Los casos de uso más importantes de la aplicación son los siguientes:

Caso de Uso: Registrarse en el sistema
Objetivo: Un visitante se registra como cliente en la aplicación.
Actores: Visitante.
Precondiciones: No hay.
Pasos: <ol style="list-style-type: none">1. El visitante rellena el formulario de registro.2. La aplicación valida los datos.3. Se guardan los datos del nuevo cliente en la aplicación.4. El nuevo cliente accede al área de clientes. Se almacena una variable en la sesión que identifica al cliente para mantener el estado.
Extensones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y no se realiza el registro.2.2. Si el nombre de usuario ya está utilizado la aplicación muestra un mensaje de error indicándolo y no se realiza el registro.2.3. Si la contraseña introducida en el campo de introducir contraseña y la contraseña no son iguales la aplicación muestra un mensaje de error indicándolo y no se realiza el registro

Caso de Uso: Acceder al sistema
Objetivo: Un cliente accede a la aplicación.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema.
Pasos: <ol style="list-style-type: none">1. El cliente rellena el formulario de acceso con su nombre de usuario y contraseña.2. La aplicación valida los datos.3. El cliente accede al área de clientes. Se almacena una variable en la sesión que identifica al cliente para mantener el estado.
Extensones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y el cliente no accede al sistema.

Caso de Uso: Realizar nuevo pedido.
Objetivo: Un cliente realiza un pedido.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes.
Pasos: <ol style="list-style-type: none">1. El cliente rellena el formulario de nuevo pedido.2. La aplicación valida los datos.3. La aplicación asigna el pedido a un trabajador teniendo en cuenta el número de pedidos asociados al trabajador y el tiempo que tardará en realizarlos.4. La aplicación comunica al trabajador elegido que tiene un nuevo pedido.5. La aplicación comunica al cliente que el pedido se ha realizado con éxito indicando la hora máxima de entrega que dependerá del tipo de pedido.
Extensiones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y el cliente no accede al sistema.

Caso de Uso: Enviar mensaje
Objetivo: El administrador envía un mensaje a un trabajador de la empresa.
Actores: Administrador.
Precondiciones: El administrador ha accedido al área de administración. El administrador ha seleccionado un trabajador.
Pasos: <ol style="list-style-type: none">1. El administrador ha elegido un trabajador a quien enviar un mensaje.2. El administrador escribe el mensaje y hace clic en enviar.3. La aplicación envía el mensaje al trabajador a través de la nube de dispositivos de Digi.4. La aplicación comunica al administrador que el mensaje ha sido enviado con éxito.
Extensiones: <ol style="list-style-type: none">1.1. El trabajador no está conectado.1.2. El mensaje está vacío y no se envía.4.1. El mensaje no se ha enviado bien y la aplicación lo comunica mediante un mensaje en pantalla.

La especificación del resto de los casos de uso puede verse en el Anexo de Casos de Uso.

Tecnologías

Para llevar a cabo este trabajo se utilizarán las siguientes tecnologías y plataformas:

- **Etherios device cloud:** La plataforma Etherios es una plataforma operativa en la nube de máquina a máquina (M2M). Es una plataforma de servicio con *hosting* bajo demanda que no requiere de infraestructura por parte del usuario. Etherios proporciona servicios de gestión de dispositivos, mensajes de dispositivos en tiempo real y almacenamiento de datos. Hasta el 1 de Mayo de 2013 se denominaba iDigi device cloud.
- **Google App Engine:** Servicio de alojamiento web que nos permite acceder a los recursos de Google con el objetivo de crear aplicaciones que funcionen en la nube. Dispone de un almacén de datos donde podremos almacenar toda la información que nuestra aplicación necesite para funcionar. Existe un plugin mediante el cual podremos integrar Google App Engine con Eclipse.
- **Youtube:** Sitio web en el cual los usuarios pueden subir y compartir vídeos. Es el sitio web al que se subirá el vídeo promocional de la aplicación. Es un servicio gratuito, conocido a nivel mundial y que permite insertar vídeos mediante código HTML en nuestra página.
- **Eclipse IDE:** IDE basado en Java de licencia pública. Será el utilizado para desarrollar el código de la aplicación. Permite instalar plugins para ampliar la funcionalidad del IDE.
- **Git:** Software de control de versiones que se utilizará en el proyecto. El proyecto se almacenará en un repositorio localizado en el servidor de Digi. Mediante el plugin eGit podremos integrarlo en Eclipse.
- **JIRA:** Aplicación basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos. Se utilizará para realizar el seguimiento del proyecto.

- **GIMP:** Programa de edición de imágenes libre y gratuito. Se utilizará para la creación y edición de las imágenes de la aplicación.
- **Cacoo:** Herramienta online gratuita de realización de diagramas.
- **Microsoft Office Word:** Editor de texto. Utilizado para la realización de la memoria.
- **PowToon:** Herramienta de software para presentaciones de negocios en línea que permite crear videos animados libre de forma gratuita.
- **Camtasia Studio:** Editor de vídeo utilizado para montar el vídeo. Se utilizará la versión de prueba de 30 días.
- **Consola Javascript de Google Chrome:** Utilizada para depurar el código javascript de la aplicación.
- **Fiddler:** Aplicación proxy o puente para depuración de programas que utilicen el protocolo HTTP.

Además de estas tecnologías, se utilizarán los siguientes lenguajes:

- **Java:** Utilizado en el lado del servidor mediante Servlets y demás clases de la aplicación.
- **Javascript, jQuery y AJAX:** Utilizados en el lado del cliente para mostrar los elementos dinámicos de la aplicación.
- **JSP, HTML y CSS:** Utilizados en el lado del cliente para mostrar la información de la aplicación.
- **XML:** Utilizado para comunicar información a través de los servicios web de la aplicación.

Diseño

Arquitectura del sistema

Diseño arquitectónico

Para la elaboración del proyecto se ha optado por un diseño basado en capas, el cual es el más utilizado en la mayoría de proyectos de este tipo. Este diseño está basado en la independencia de cada capa respecto a las demás, haciendo que cada una de las tareas sea independiente. Las ventajas de utilizar este diseño son:

- **Robustez:** Gracias a que encapsulamos las capas, la aplicación será más robusta.
- **Facilidad para realizar cambios:** Si se detectase un fallo o se introdujeran nuevas funcionalidades en una capa, el cambio solo afectaría a esa capa.
- **Escalabilidad:** Facilidad para si se quiere en un futuro ampliar las funcionalidades de la aplicación.
- **Mantenimiento:** A la hora de revisar la aplicación será más sencillo si esta está dividida en diferentes capas.

Capas de la aplicación:

En este caso estas son las capas en que se dividirá la aplicación:



- **Persistencia:** Capa de acceso a los datos almacenados.
- **Lógica de negocio:** Lógica de la aplicación. La conforman las clases donde se concentra la funcionalidad de la aplicación y es el nexo de unión entre la capa de persistencia y la de presentación.

- **Presentación:** Interfaz de la aplicación. Es la encargada de mostrar los datos al usuario.

Patrón arquitectónico

El patrón elegido para el desarrollo de la aplicación es el patrón MVC (Modelo Vista Controlador) ya que con él se logra una división de las diferentes partes que conforman una aplicación.

El patrón se divide en tres partes que son:

- **Modelo:** Concentra las funcionalidades relacionadas con el modelo de datos.
- **Vista:** Aspecto visual/gráfico que será empleado por la aplicación. Ficheros html y jsp que mostrarán la información.
- **Controlador:** Mediador entre el Modelo y la Vista. En este caso los controladores serán los Servlets de la aplicación que recibirán peticiones, recogerán los datos del modelo y se los enviarán a la vista para que los muestre.

Modelo de datos

Debido a que Google App Engine nos proporciona su datastore como medio de almacenamiento, no será necesario utilizar una base de datos relacional en la aplicación.

Entidades JDO

Para poder operar con el datastore se utilizan entidades JDO (Java Data Objects) que es una especificación de almacenamiento de objetos en Java, la cual no se limita a lo que es el almacenamiento de datos en bases de datos relacionales únicamente, pudiendo utilizarse contra ficheros XML, documentos de OpenOffice, objetos JSON, etc.

Las entidades JDO deben ser objetos persistentes y para indicarlo utilizaremos las siguientes anotaciones para la clase y sus atributos:

- **@PersistenceCapable**

Indica que una clase puede ser recuperada y almacenada mediante el soporte de JDO.

```
import javax.jdo.annotations.IdGeneratorStrategy;  
  
@PersistenceCapable  
public class Client {
```

- **@Persistent**

Indica que queremos poder almacenar el campo de la entidad en el datastore.

```
@Persistent  
private String name;  
  
@Persistent  
private String surnames;
```

- **@PrimaryKey**

Indica que el campo va a ser la clave primaria, e irá acompañada de la anotación **@Persistent** ya que la clave también será almacenada.

```
@PrimaryKey  
@Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)  
private String nick;
```

Clase PersistenceManagerFactory

Todas las solicitudes que utilizan el datastore utilizan una instancia de la clase PersistenceManagerFactory.

Aunque una instancia de PersistenceManagerFactory tarda algún tiempo en inicializarse, solo se necesita una instancia para cada aplicación, y esta instancia se puede almacenar en una variable estática que pueden utilizar varias solicitudes y clases. Una forma sencilla de realizar este procedimiento es mediante la creación de una clase envoltorio "singleton" para la instancia estática.

La clase es la siguiente:

```
import javax.jdo.JDOHelper;
import javax.jdo.PersistenceManagerFactory;

public final class PMF {
    private static final PersistenceManagerFactory pmfInstance =
        JDOHelper.getPersistenceManagerFactory("transactions-optional");

    private PMF() {}

    public static PersistenceManagerFactory get() {
        return pmfInstance;
    }
}
```

Operaciones con el datastore

Para realizar las operaciones de lectura, almacenamiento, modificación y eliminación de los datos, crearé una serie de clases que tendrán el siguiente nombre:

<nombre del tipo de objeto>Utils.java

Estas clases utilizarán la clase PMF creada anteriormente para realizar las operaciones con el datastore.

A continuación se muestran diferentes ejemplos de operaciones que se pueden realizar:

- Lectura:

```
public static Client getClientFromDataStore(String nick) {
    Key key = KeyFactory.createKey(Client.class.getSimpleName(), nick);
    Client client = null;
    PersistenceManager pm = PMF.get().getPersistenceManager();
    try {client = pm.getObjectById(Client.class, key);
    } catch (Exception e) {
        return null;
    } finally{
        pm.close();
    }
    return client;
}
```

- Almacenamiento:

```
public static boolean addClientToDataStore(Client client) {  
    if (client == null)  
        return false;  
    PersistenceManager pm = PMF.get().getPersistenceManager();  
    pm.makePersistent(client);  
    pm.close();  
    return true;  
}
```

- Modificación:

```
public static void updateImportance(String taskId, int importance) {  
    Key key = KeyFactory.createKey(Task.class.getSimpleName(), taskId);  
  
    PersistenceManager pm = PMF.get().getPersistenceManager();  
    Task task = pm.getObjectById(Task.class, key);  
    task.setImportance(importance);  
    pm.close();  
}
```

- Eliminación:

```
public static boolean deleteWorkerFromDataStore(Worker worker) {  
    PersistenceManager pm = PMF.get().getPersistenceManager();  
    boolean deleted = false;  
    if (worker != null){  
        try{  
            pm.deletePersistent(pm.getObjectById(worker.getClass(), worker.getId()));  
            deleted = true;  
        }catch(Exception e) {  
            e.printStackTrace();  
        }finally {  
            pm.close();  
        }  
    }  
    return deleted;  
}
```

Índices

El almacén de datos de App Engine utiliza índices por cada consulta que realiza la aplicación. Estos índices se actualizan cada vez que una entidad cambia, de modo que los resultados se pueden devolver rápidamente cuando la aplicación realiza una consulta.

Para ello, el almacén de datos necesita conocer por adelantado las consultas que va a realizar la aplicación.

La información sobre los índices del datastore se guarda en el fichero datastore-indexes.xml. Se trata de un archivo XML cuyo elemento raíz es <datastore-indexes>. Contiene cero o varios elementos <datastore-index>, uno para cada índice que App Engine debe mantener.

Navegación

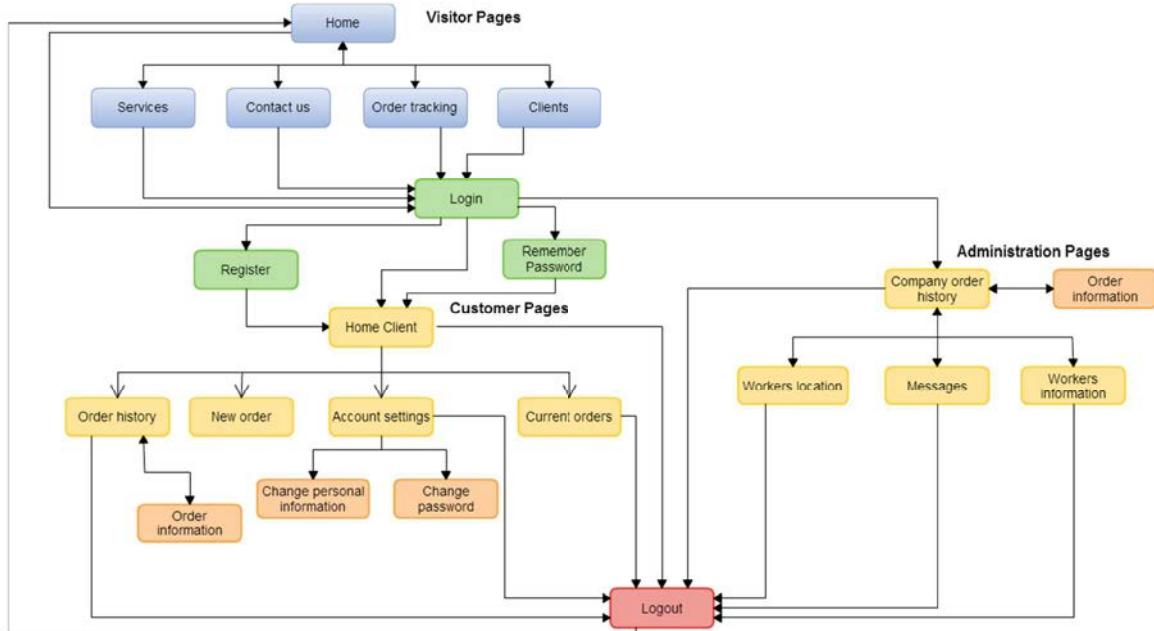
En cuanto a la navegación, la aplicación tiene tres partes diferenciadas:

- **Área de visitantes:** Área que incluye la página de inicio de la aplicación, páginas de información sobre la empresa y el área de registro y acceso al sistema.
- **Área de Clientes:** Conjunto de páginas a través de las cuales los clientes pueden realizar tareas como ver el historial de pedidos o realizar un nuevo pedido.
- **Área de administración:** Páginas a través de las cuales el administrador puede realizar tareas como ver el historial de pedidos de la empresa o localizar a los trabajadores.

Para ir del área de visitantes a las áreas de clientes y administración es necesario registrarse en el sistema o acceder con un nombre de usuario y contraseña.

Se puede volver al área de visitantes saliendo de la cuenta en la que estemos en ese momento.

No es posible acceder al sistema con dos cuentas diferentes a la vez ya sea como cliente o administrador.



Mapa de navegación del sitio web

Servicios Web

Una de las partes más importantes de cara a la comunicación entre las aplicaciones Android y la aplicación web son los servicios web. Gracias a ellos, las aplicaciones Android obtienen información de la aplicación web o la modifican. De esta forma no hace falta guardar información en la aplicación Android ya que comparte la información con la aplicación web.

Al recibir una llamada al servicio web, la aplicación analiza los parámetros recibidos y devuelve la información en formato XML.

Los servicios web de la aplicación solo admitirán peticiones POST. De esta manera conseguimos que los parámetros no se vean a simple vista y aumentamos la seguridad.

En la aplicación realizada en prácticas ya se implementaron varios servicios web para obtener la información de las tareas de cada trabajador, comunicar la desconexión del dispositivo, actualizar el estado de una tarea y registrar un nuevo trabajador. Estos servicios web se reutilizarán en la nueva aplicación modificando algunos parámetros de entrada.

Los servicios web a realizar en la aplicación permitirán realizar pedidos, hacer seguimiento de pedidos, identificarse en la aplicación, obtener los datos de los pedidos de un cliente y obtener el histórico de pedidos.

Diseño de la interfaz

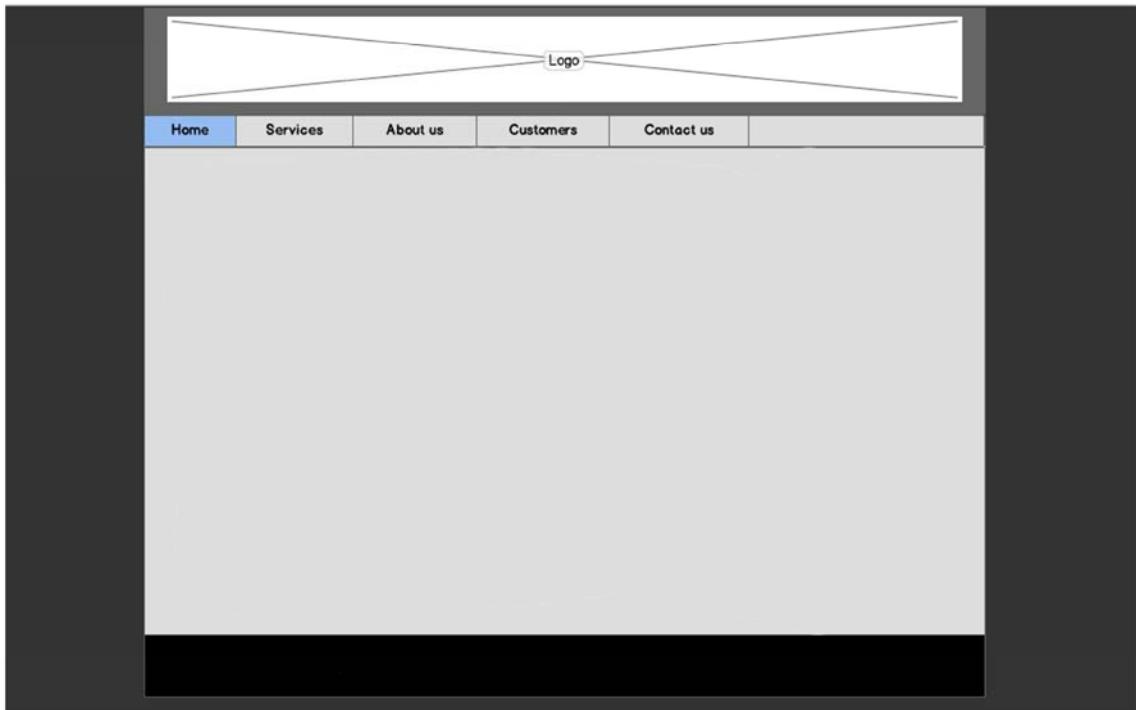
En este paso se pasará a realizar un diseño básico de la interfaz para ubicar los elementos principales de las páginas. Para ello se diseña una página base.

Se decide que la página base debe tener las siguientes partes:

- Una cabecera con el logo de la empresa.
- Un menú desde el que se accederá a otras páginas.
- La parte principal de la página que será el núcleo.
- Un pie de página.

Un primer mockup de la página se realiza con el programa Balsamiq Mockups. Todavía no se han escogido los colores o imágenes de la aplicación ya que en esta primera versión lo importante es decidir la estructura básica de las páginas.

Teniendo el mockup básico de la aplicación será más sencillo diseñar el resto de páginas de la aplicación ya que todas se basarán en este diseño.



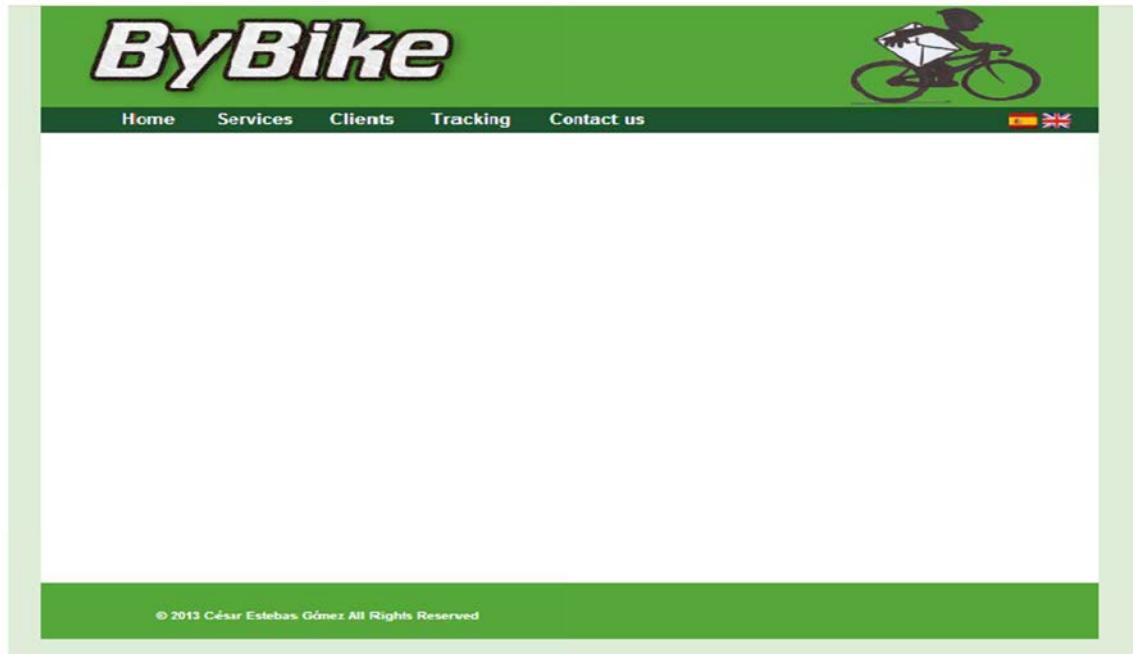
Primer Mockup de la página básica

Teniendo la página básica, se elige una gama de verdes para dar color a la página. El logo estará formado por un mensajero en bicicleta con un sobre a la derecha y el nombre de la empresa a la izquierda.

Las banderas del cambio de idioma se ubicarán a la derecha del menú.

En el pie de página aparecerán a la izquierda el Copyright y a la derecha el logo del Cloud connector de Etherios.

A partir del diseño de la página básica se irán desarrollando los diseños de las diferentes páginas de la aplicación.



Mockup final de la página básica

Implementación

A la hora de realizar la implementación se ha decidido utilizar Scrum como metodología, ya que nos permite tener versiones del sistema con funcionalidad total al final de cada sprint. Se ha dividido el trabajo en cuatro sprints de dos semanas haciendo coincidir el final de los estos con las reuniones de la empresa en las que podremos mostrar los avances conseguidos en las dos semanas.

Para gestión de esta etapa se utilizará Jira, software de gestión de proyectos utilizado en Digi, ya que nos permite utilizar metodologías ágiles y generará diagramas en los que se podrá observar la evolución del proyecto.

Sprint 1 (27/02 – 13/03)

En este primer sprint las tareas a realizar son:

- Implementar la interfaz de la página base.
- Adaptar las páginas de administración al nuevo diseño de la página.
- Implementar la página principal, la de servicios y la de contacto.
- Realizar la traducción de las páginas para que todas estén disponibles en inglés y español.

Adaptación de las páginas de administración

Tras realizar la página base, se adaptan las páginas de administración existentes manteniendo toda la funcionalidad anterior. Estas páginas son la del historial de tareas de la empresa, la de localización de los trabajadores y la página que muestra los datos de los trabajadores.

Página principal

A la hora de diseñar la página de inicio, se decide añadir un video de presentación del sistema y un slide de imágenes representativas de la empresa que cambiarán cada pocos segundos.

Página de servicios

En esta página aparece una descripción de los servicios que ofrece la empresa. En principio los servicios ofertados son:

- Madrugador: Si se realiza el pedido antes de las 10am es entregado antes de las 8pm.
- Dos horas: El pedido será entregado en un máximo de dos horas.

- Seis horas: El pedido será entregado en un máximo de seis horas.
- Día siguiente: El pedido será entregado antes de las 12:00 del día siguiente.

No se descarta que en algún momento se añadan más servicios si se considera oportuno.

Página de contacto

La página de contacto consiste en un formulario mediante el cual los visitantes de la página pueden hacer preguntas que después contestará la administración.

Los campos del formulario son los siguientes:

- Email al que se contestará.
- Nombre del interesado.
- Asunto del mensaje.
- Cuerpo del mensaje.

Para evitar que los mensajes sean demasiado largos, se limitará el número de caracteres del mensaje a 300.

Cuando se pulse el botón enviar, se comprobará que los campos son válidos y se almacenará el mensaje para que lo lea la administración. Para ello se crea la clase ContactMsg en el datastore. Si todo ha salido bien se informará al usuario mediante un mensaje en una ventana pop-up. En caso de error también se mostrara un mensaje con indicaciones para solventar el error.

Traducción de las páginas

Cuando se empezó a desarrollar la página de administración el idioma elegido fue el inglés por política de la empresa. Sin embargo, al plantear el proyecto, se decidió desarrollar la página en inglés y español ya que es interesante poder disponer de la información en varios idiomas.

Para la realización de esta tarea se barajaron dos opciones:

- La primera opción es desarrollar dos páginas, una en cada idioma. Los problemas que aparecen son que tenemos el mismo código dos veces y que si queremos añadir un nuevo idioma a la página tendremos que crear una nueva versión de la página.
- La segunda opción es realizar una página con diferentes fuentes de idioma. De esta forma los textos de la página se almacenarán en ficheros de propiedades, uno por cada idioma, y si queremos añadir un idioma nuevo solo tendremos que añadir un fichero para ese idioma.

La opción elegida es la segunda ya que no se duplica el código y nos facilita añadir nuevos idiomas en un futuro.

Para poder aplicar esta solución necesitaremos dos ficheros: bybykeMsg_en.properties (textos en inglés) y bybykeMsg_es.properties (textos en español). Como puede observarse, los dos ficheros tienen el mismo nombre cada uno seguido de un guión bajo y el Locale(código del idioma).

Mediante las etiquetas bundle de la librería fmt de JSTL podemos cargar el fichero indicando el nombre común.

```
<fmt:bundle basename="bybykeMsg">  
...  
</fmt:bundle>
```

Con la función setLocale indicaremos el locale a utilizar, por ejemplo el locale por defecto del navegador.

```
<fmt:setLocale value="default" />
```

Con la etiqueta message seleccionamos la cadena a mostrar. Un ejemplo de uso es el siguiente:

```
<fmt:message key="home.greet.tit"></fmt:message>
```

En el ejemplo se elige el idioma por defecto del navegador. Para poder cambiar de idioma y que permanezcan los cambios hay que añadir una variable a la sesión.

Se añaden dos imágenes de banderas representativas de los idiomas y al pulsarlas se cambiará el idioma. Para ello hay que crear un Servlet que recoja como parámetro el idioma a cambiar y modifique la variable de la sesión.

Fin del primer sprint

Tras las dos primeras semanas, tenemos una página con la funcionalidad de la página de administración anterior y con tres nuevas páginas para visitantes. Además se ha adoptado un sistema para la realización de la página en dos idiomas que será aplicado a lo largo de toda la implementación. El número de horas estimado para el sprint eran 36 y las horas reales han sido 35 horas y media. Como se puede observar el desfase ha sido mínimo.

Tras la reunión en la empresa se sube la versión de la página a Google App Engine y se planifica el segundo Sprint.

Sprint 2 (13/03 – 27/03)

Las tareas a realizar en el sprint son:

- Implementar las páginas de nuevo cliente, seguimiento de pedido y clientes.
- Implementar formulario de acceso.
- Implementar servicio web para hacer seguimiento de pedido.
- Implementar la página de inicio del área de clientes.
- Implementar el interfaz de la página de nuevo pedido.

Página de clientes

Esta página sirve para animar a los clientes a registrarse en la aplicación. También hay una imagen promocional de la aplicación Android para clientes.

Página de seguimiento de pedido

En esta página podremos hacer seguimiento de un pedido sin estar registrados en la página.

Al introducir el identificador del pedido que deseamos seguir aparece una tabla con los diferentes cambios de estado por los que ha pasado el pedido, la fecha y hora del cambio de estado y posibles detalles del cambio de estado.

Esta forma de hacer seguimiento permite a usuarios no registrados como clientes estar informados del estado de los pedidos en los que están interesados ya que para ello solo deben saber su identificador.

La información será muy detallada, para estar mejor informado se debe estar registrado en la aplicación como cliente y ser el propietario del pedido.

Formulario de acceso

El formulario para acceder a la aplicación aparecerá en todas las páginas del área de visitantes. Con este formulario accederán al sistema tanto el administrador como los clientes.

Al introducir el nombre de usuario y la contraseña se comprobará que son parámetros válidos. Si no lo son se muestra un mensaje de error.

Si los parámetros son válidos se genera el hash SHA1 de la cadena <usuario>:<contraseña> y se comprueba si es igual al almacenado en la aplicación para el administrador. Si es igual se accede al área de administración. Si no es la cuenta de administrador, se comprueba si los datos coinciden con los de algún cliente. Si los datos coinciden con los de algún cliente se accede al área de clientes.

En caso de no poder acceder a ninguna de las dos áreas, se mostrará un mensaje de error indicando que el nombre de usuario o la contraseña no son válidos.

Página de nuevo cliente

En esta página los visitantes podrán registrarse en el sistema como clientes. La página consiste en un formulario dos apartados.

- En el primer apartado se introducen los datos personales del nuevo cliente.
- En el segundo apartado se selecciona un nombre de usuario y una contraseña.
Debe repetirse la contraseña para validarla.

Tras introducir la información, el sistema la validará comprobando que las contraseñas son iguales y que el nombre de usuario no está siendo utilizado. Si los campos son válidos se redirecciona al cliente a la página de inicio del área de clientes.

Página de inicio del área de clientes

En esta página aparecen enlaces a las diferentes partes del área de clientes. También se ha incluido una imagen promocional de la aplicación Android de clientes.

Servicio web para seguimiento de pedido

Como parámetro de entrada en este servicio web solo se pide el código del pedido a seguir. El servicio web devuelve un XML con la información de los cambios de estado del pedido, la fecha y hora a la que se realizó el cambio y los posibles detalles.

Página de nuevo pedido

El interfaz de la página de nuevo pedido será un formulario con tres apartados:

- **Datos de recogida:** Datos de la dirección en la que se recogerá el paquete. Será obligatorio llenar los campos dirección y ciudad.
- **Datos de entrega:** Datos de la dirección en la que se entregará el paquete. Será obligatorio llenar los campos dirección y ciudad.
- **Datos del servicio:** En este apartado se indicará el tipo de servicio, el método de pago y los posibles detalles importantes para el envío.

Como ya se ha comentado, la aplicación tiene cuatro tipos de servicios que estarán disponibles en diferentes franjas horarias. Los servicios se mostrarán en las siguientes franjas:

- **Madrugador:** Disponible entre las 00:00 hasta las 10:00.
- **2 horas:** Disponible entre las 8:00 y las 18:00.
- **6 horas:** Disponible entre las 8:00 hasta las 14:00.
- **Día siguiente:** Disponible las 24 horas del día.

Dado que no todos los servicios están disponibles las 24 horas del día, solo se muestran los servicios disponibles en el momento de realizar el pedido.

Se almacenarán las direcciones utilizadas para mostrarlas como direcciones almacenadas y que no tengan que volver a rellenarse los datos en futuros pedidos.

Una vez llenado el formulario, se validan los datos. Si son válidos, estos se enviarán mediante una petición POST al servicio web de creación de pedido. Este servicio web se desarrollará en el próximo sprint.

Fin del segundo sprint

Después de las primeras cuatro semanas de implementación, ya disponemos de una funcionalidad casi total en las áreas de administración y de visitantes. Además, ya se ha comenzado con el área de clientes creando la página de inicio y la página para la realización de un nuevo pedido.

El número de horas estimado para el sprint eran 43 y las horas reales han sido 44 horas. Como se puede observar el desfase ha sido mínimo. Este se ha debido a que se ha tardado más de lo esperado en realizar la página para hacer seguimiento de pedido ya que se han tenido que crear nuevas clases JDO para almacenar los cambios de estado de los pedidos.

Tras la reunión en la empresa se sube la versión de la página a Google App Engine y se planifica el tercer Sprint.

Sprint 3 (27/03 – 10/04)

Las tareas a realizar en el sprint son:

- Diseñar e implementar el algoritmo de asignación de tareas.
- Implementar las páginas del histórico de pedidos del cliente y de pedidos actuales del cliente.
- Realizar los servicios web para el nuevo pedido, las direcciones almacenadas y el histórico de pedidos.

Algoritmo de asignación de tareas

El algoritmo planificador es el encargado de asignar las tareas a los trabajadores. Para ello se tienen en cuenta diferentes factores como el número de tareas actuales de los trabajadores, la distancia a recorrer o el tiempo necesario para realizar las tareas.

Para el cálculo de distancias se barajan dos opciones:

- La primera opción es calcular las distancias entre los diferentes puntos por los que tiene que pasar el trabajador. Para ello, cuando se generan las tareas, se almacenan las coordenadas geométricas de la dirección de origen y destino. Las coordenadas se obtienen con el API de codificación geográfica de GoogleMaps. Se barajaron otras opciones para la obtención de las coordenadas como GoogleStreetMaps pero se ha elegido Google Maps porque es más exacta.
- La segunda opción pasaría por utilizar el API DistanceMatrix de Google Maps. Las llamadas al servicio devuelven la distancia en metros entre los puntos y el tiempo que se tarda en recorrerla. El problema es que está limitada a 5000 peticiones gratuitas.

Finalmente se elige la primera opción, ya que aunque no es tan exacta, no nos limita a 5000 peticiones.

A la hora de desarrollar el algoritmo se dividen las tareas en dos grupos, las del día siguiente y el resto. Como máximo un trabajador podrá tener asignadas 6 tareas del día siguiente y 10 del resto.

Si la tarea se inicia entre las 8:00 y las 18:00 solo se tienen en cuenta los trabajadores conectados.

Pasos para asignar el trabajador a tarea de 2 horas:

1. Se tendrá en cuenta que se puedan realizar el resto de tareas de 2 horas y que tras finalizarlas se pueda hacer la tarea nueva.
2. Si esto es posible, se comprueba que se pueden hacer las tareas de 6 horas y las del servicio madrugador que tienen como hora límite las 20:00.

Pasos para asignar el trabajador a tarea de 6 horas:

1. Se tendrá en cuenta que se puedan realizar las tareas de 2 horas y el resto de tareas de 6 horas y que tras finalizarlas se pueda hacer la tarea nueva.
2. Si esto es posible, se comprueba que se pueden hacer las tareas del servicio madrugador que tienen de hora límite las 20:00.

Pasos para asignar el trabajador a tarea del servicio Madrugador:

1. Se tendrá en cuenta que se puedan realizar las tareas de 2 horas, 6 horas y el resto de tareas del servicio madrugador y que tras finalizarlas se pueda hacer la tarea nueva.

Pasos para asignar el trabajador a tarea del servicio Día Siguiente:

1. Se tendrá en cuenta que se puedan realizar el resto de tareas de tras finalizarlas se pueda hacer la tarea nueva.

Si se cumplen los requisitos se marca como válido el trabajador. Finalmente el trabajador con menos tareas asignadas será el que realice la tarea o en su defecto el que menos tarde en realizarlas.

Servicio web nuevo pedido

Este es el servicio web al que llamarán el formulario de nuevo pedido y la aplicación de clientes para realizar un pedido nuevo. Si las direcciones no habían sido utilizadas, se almacenan para futuros pedidos. El servicio web utiliza el algoritmo de asignación para seleccionar el trabajador que realizará la tarea. Si el pedido se realiza correctamente, se comunica al cliente del éxito al realizar el pedido. Si el pedido se realiza entre las 8:00 y las 18:00 se manda un mensaje al trabajador para informar de la nueva tarea. Además de este, se implementa el servicio web para las direcciones almacenadas que devuelve las direcciones almacenadas del cliente para que las muestre la aplicación móvil.

Página del histórico de pedidos

Página que muestra en una tabla los pedidos finalizados. Estos se pueden filtrar por estado o por fechas. Seleccionando los pedidos podremos ver la información ampliada de cada pedido. También se desarrolla el servicio web que devuelve todos los pedidos del cliente.

Página de pedidos en curso

Página que muestra los pedidos en realización del cliente. Al seleccionar uno vemos la información detallada del pedido y los cambios de estado por los que ha ido pasando.

Fin del tercer sprint

Tras el tercer sprint los clientes pueden registrarse, realizar pedidos y ver sus pedidos tanto actuales como finalizados.

El número de horas estimado para el sprint eran 46 y las horas reales han sido 50 horas. El desfase de 4 horas se ha debido a que en un principio se pensó utilizar el API de distancias de Google Maps pero con las pruebas de cálculo se observó que se llegaba con facilidad a las 5000 peticiones límite, por lo que se tuvo que buscar una alternativa.

Tras la reunión en la empresa se sube la versión de la página a Google App Engine y se planifica el cuarto Sprint que se prevé será el último de la fase de implementación.

Sprint 4 (10/04 - 24/04)

Las tareas a realizar en el último sprint de la implementación son las siguientes:

- Realización de las páginas de configuración de la cuenta de cliente y de mensajes de administración.
- Generación de la factura del pedido a su finalización.
- Implementar la funcionalidad del envío de correo electrónico automático y añadirla en los sitios en que se necesite.
- Implementar funcionalidad de recuerdo de contraseña.
- Mejora de la seguridad en el registro.

Envío de correo electrónico automático

Esta es la primera tarea a realizar en el sprint ya que el resto dependen de ella.

A la hora de enviar correo, GoogleAppEngine nos proporciona la posibilidad de enviar correos con una cuenta de Gmail.

Primero se ha creado la clase MailUtils.java con el método estático sendMail que será el encargado de enviar el correo. El código del método es el siguiente:

```
public static void sendMail(String msgc, String subject, String mail, String name,
                           Attachment file) {
    Properties config = new Properties();
    InputStream is = Mailutils.class.getResourceAsStream("/mail.properties");
    config.load(is);
    Properties props = new Properties();
    Session session = Session.getDefaultInstance(props, null);
    Message msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress(config.getProperty(
        "mail.smtp.user", config.getProperty("mail.smtp.name"))));
    msg.addRecipient(Message.RecipientType.TO, new InternetAddress(
        mail, name));
    msg.setSubject(subject);
    msg.setText(msgc);

    if(file!=null){
        Multipart mp = new MimeMultipart();
        MimeBodyPart mbp2 = new MimeBodyPart();
        byte[] content = file.getContent();
        if (content != null) {
            mbp2.setContent(content, file.getMimeType());
            mbp2.setFileName(file.getName());
            mp.addBodyPart(mbp2);
        }
        msg.setContent(mp);
    }
    Transport.send(msg);
}
```

Los parámetros que recibe el método son el cuerpo del mensaje, el asunto, la dirección a la que se envía el mensaje, el nombre del destinatario y los posibles adjuntos.

Los datos del emisor se almacenan en un fichero de propiedades. No es necesario almacenar la contraseña del emisor, pero si lo es que la dirección pertenezca a un administrador de la aplicación de GoogleAppEngine.

A la hora de crear el texto de los mensajes este se crea al igual que el resto de textos de la aplicación en inglés y en español. De esta forma, se tiene en cuenta el idioma a la hora de enviar el mensaje.

Uno de los usos del envío de mensajes es a la hora de crear un pedido. En este mensaje se envía el identificador del pedido para que se pueda hacer seguimiento.

Página de mensajes de administración

En esta página se mostrarán los mensajes que los visitantes dejan a la administración pidiendo información sobre el servicio. Sólo aparecerán los mensajes no contestados, para que no se acumulen.

Al seleccionar el mensaje se muestra el cuerpo del mensaje recibido, el asunto y el nombre. Con estos datos el administrador podrá contestar al mensaje con la mayor exactitud posible.

Una vez escrito el mensaje, antes de enviarlo se comprueba que los datos son válidos. El cuerpo del mensaje enviado se compone del mensaje recibido entre comillas y la respuesta debajo.

Una vez enviado el mensaje se refrescan los mensajes mostrados habiendo desaparecido de la lista el respondido. Estos mensajes no se borran, se puede acceder a ellos desde la administración del datastore de GoogleAppEngine. Así en un futuro se podrá hacer minería de datos con los mensajes de los visitantes para conocer los intereses de los posibles clientes.

Generación de factura

Para la generación de las facturas se decide utilizar la librería iText ya que es una de las que más posibilidades ofrecen y permite generar documentos PDF.

El documento será muy simple. Aparecerá el logo en la parte superior, los datos de recogida y de entrega junto con la hora de entrega y el servicio en el centro y la firma en la parte inferior.

El problema que surge es que a la hora de generar el fichero, iText utiliza la clase FileInputStream, clase que GoogleAppEngine no permite utilizar en sus aplicaciones para que no se almacenen los datos en ficheros sino en su datastore. Debido a esto, no se enviará el fichero como adjunto de correo electrónico.

Para poder transmitir la factura, se utiliza un servlet que comprueba si el pedido ha terminado y genera la factura. Una vez generada la factura, devuelve los bytes del fichero PDF generado. De esta forma no hace falta crear el fichero de forma física.

El código del servlet que devuelve la factura es el siguiente:

```
public class GetInvoiceServlet extends HttpServlet {  
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) {  
        String idTask = req.getParameter("idorder");  
        try {  
            Task t = Taskutils.getTaskFromDatastore(idTask);  
            if (t != null) {  
                if(t.getStatus()==5){  
                    resp.setContentType("application/pdf");  
                    String nomPdf = idTask + ".pdf";  
                    resp.setHeader("Content-Disposition", "attachment; filename=\"" + nomPdf + "\"");  
                    Document doc = new Document();  
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();  
                    PdfWriter.getInstance(doc, baos);  
                    doc.open();  
                    PDFUtils pdfu = new PDFUtils();  
                    byte[] data = pdfu.generatePdf(baos, doc, t);  
                    OutputStream os = resp.getOutputStream();  
                    os.write(data);  
                } else{  
                    resp.sendRedirect("/");  
                }  
            } else{  
                resp.sendRedirect("/");  
            }  
        } catch (BadElementException e) {  
            e.printStackTrace();  
        } catch (MalformedURLException e) {  
            e.printStackTrace();  
        } catch (DocumentException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) {  
        doGet(req, resp);  
    }  
}
```

La URL del servlet es: /getInvoice?idOrder=<IDPedido>

Esta URL se añade al cuerpo del mensaje enviado al terminar un pedido. En caso de terminar el pedido en incidencia, no se envía la factura sino un mensaje informando del problema.

Página de configuración

La página de configuración está dividida en tres partes:

- Cambio de datos personales.
- Cambio de idioma.
- Cambio de contraseña.

En los datos personales aparecerán los campos con los datos actuales y podrán cambiarse. Para guardar los cambios todos los datos deben ser válidos. Lo mismo ocurre con el idioma.

Para cambiar la contraseña, debe repetirse la nueva contraseña. Una vez cambiada, se informará mediante un correo electrónico del cambio.

Debido a que los clientes tienen un idioma que eligen ellos, se han quitado las banderas de selección de idioma de todas las páginas del área de clientes.

Recuerdo de contraseña

Se ha decidido añadir a la página la funcionalidad para que los clientes puedan recuperar su contraseña. Debido a que no se guarda la contraseña sino un hash no podremos recordársela, por lo que le daremos la posibilidad de cambiarla

Para ello deben indicar su correo electrónico y su nombre de usuario y se enviará por correo un enlace para cambiar la contraseña.

Con una clase en el datastore se mantiene el registro de los cambios de contraseña pedidos. Una vez realizado el cambio de contraseña, se borra este del registro. Si no se borrar, correríamos el peligro de que se ocupase demasiado espacio con peticiones de cambio de contraseña.

Una vez cambiada la contraseña se informa del cambio mediante un correo.

Mejora de la seguridad en el registro

La última tarea a realizar es mejorar la seguridad en el registro de los usuarios. Tal como se realiza ahora, no se comprueba si el correo electrónico es en realidad el de la persona que se está registrando.

Aprovechando el envío de correos, el registro se divide en dos partes. La primera parte corresponde a lo que teníamos hasta ahora, la comprobación de los datos del formulario. Una vez comprobados, se envía un correo con un enlace para activar la cuenta. De esta forma si la cuenta no está activa, no se puede acceder al sistema.

Para el control de los enlaces se crea una clase en el datastore que almacena las cuentas que no han sido activadas. Una vez activada la cuenta se borra el objeto relacionado con la activación de la cuenta.

El servlet de activación de la cuenta es el siguiente:

```
public class ActivateClientServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse resp) {  
        doPost(req, resp);  
    }  
    public void doPost(HttpServletRequest req, HttpServletResponse resp) {  
        String id = req.getParameter("id");  
        if(id!=null){  
            ActivationEnt a = ActivationEntutils.getActivationEntFromDataStore(id);  
            try {  
                if(a!=null){  
                    Client c = Clientutils.getClientFromDataStore(a.getClient());  
                    if(c!=null){  
                        if(!c.isActive()){  
                            Clientutils.updateActive(c.getNick(), true);  
                            if(ActivationEntutils.deleteActivationEntFromDataStore(a)){  
                                HttpSession session = req.getSession(true);  
                                session.setAttribute("client", c.getNick());  
                                req.getSession().removeAttribute("lang");  
                                req.getSession().setAttribute("lang", c.getLang());  
                                RequestDispatcher rd = req.getRequestDispatcher("/activateClient.jsp");  
                                rd.forward(req, resp);  
                            } else{  
                                resp.sendRedirect("/");  
                            }  
                        } else{  
                            resp.sendRedirect("/");  
                        }  
                    } else{  
                        resp.sendRedirect("/");  
                    }  
                } else{  
                    resp.sendRedirect("/");  
                }  
            } catch (IOException e) {  
                e.printStackTrace();  
            } catch (ServletException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

Como puede observarse en el código, una vez borrada la instancia de ActivationEnt, se envía al cliente al área de clientes.

Fin del cuarto sprint

Al finalizar el cuarto sprint, la aplicación tiene una funcionalidad completa en todas sus áreas.

El número de horas estimado para el sprint eran 35 y las horas reales han sido 30 horas. El desfase de 5 horas se ha debido a que no se contaba con las facilidades que proporciona GoogleAppEngine y se sobreestimó la realización de esta tarea.

El 24 de Abril se realizó una pequeña demostración de la aplicación a miembros de la oficina de Digi en Logroño y a tres visitantes de la sede de Minnetonka.

Pruebas

Se han realizado pruebas en los siguientes navegadores:

- Google Chrome.
- Mozilla Firefox.
- Internet Explorer.
- Safari.

Las pruebas realizadas han sido las siguientes:

Aspecto visual:

- Comprobar el correcto visionado de todas las páginas de la aplicación.
- Comprobar que las imágenes y el vídeo se ven correctamente en todos los navegadores.

Control de errores:

- Los formularios filtran bien los campos.
- Los filtros de acceso a las áreas de cliente y administración funcionan correctamente.
- Si se producen errores, estos se comunican al usuario por pantalla para que conozca en todo momento el estado de la sesión.

Pruebas del algoritmo:

- Comprobar que el algoritmo de asignación de tareas es eficiente. Para ello se crean varias tareas de diferentes tipos habiendo 3 dispositivos conectados.
- Comprobar que si no se puede crear la tarea el cliente lo sepa para que pueda crearla en otro momento.

Programación:

- Comprobar la correcta traducción de todos los textos de la página. Hacer lo mismo con el envío de mensajes.
- El formulario de nuevo pedido muestra solamente los servicios disponibles en el momento de realizar el pedido.
- Si un cliente utiliza una dirección nueva, comprobar que esta se almacena y se muestra cuando se realiza un nuevo pedido.
- Los enlaces para cambiar la contraseña y activar una cuenta solo podrán utilizarse una vez.
- Los filtros de los históricos de pedidos de la aplicación y de los clientes funcionan correctamente.
- Las consultas de inserción, modificación y borrado de datos del datastore funcionan correctamente.
- Las conexiones con la nube de dispositivos se realizan correctamente.
- Los servicios web funcionan correctamente y no admiten peticiones GET.

Pruebas de usabilidad:

- Usuarios ajenos al proyecto, tanto con niveles muy básicos de informática como con niveles altos, pueden utilizar la aplicación. Pruebas realizadas con familiares y trabajadores de Digi.
- La aplicación web también es accesible desde dispositivos móviles y tablets.

Comunicación con la aplicación Android

- Los mensajes enviados a la aplicación de los trabajadores son recibidos correctamente.

Realización del video promocional

La idea de realizar un video promocional de la aplicación surgió de la posibilidad de promocionar el producto a posibles compradores. Desde el principio se quiso recalcar que el producto que vendemos no son los servicios de paquetería, sino las aplicaciones que utiliza la empresa.

A la hora de realizar el video se decidió que este fuese animado, ya que de esta forma podríamos transmitir el mensaje sin necesidad de utilizar cámaras. Para ello buscamos diferentes aplicaciones que nos permitieran realizar un video animado de forma sencilla y encontramos PowToon, una herramienta basada en Flash con una gran biblioteca de efecto e imágenes prediseñadas a las que podemos añadir imágenes propias.

Antes de realizar el video se esquematizaron las ideas principales a transmitir y se plasmaron en un storyboard. Teniendo el storyboard se redactó el guión del narrador. Tras un par de reuniones entre los miembros del equipo con el tutor y el jefe de la empresa se empezó la realización del video.

Una vez creado el vídeo, se añadieron los efectos, la música y la voz en off del narrador con el programa Camtasia Studio. Este programa permite tener varias pistas de audio y video simultáneamente y permite quitar ruido ambiental de las grabaciones de audio. La lectura del guión del narrador fue realizada por el trabajador de Digi Héctor Palacios.

En principio el video solo tiene versión en español, pero en un futuro se espera que esté traducido al inglés debido al ámbito internacional de la empresa.

Una vez terminado el vídeo se subió a Youtube y aparece en la página de inicio de la aplicación web para que los visitantes de la página puedan verlo.

Conclusiones

En este apartado se plasmarán las reflexiones e impresiones sobre el proyecto una vez terminado.

Comparativa de horas planificadas/reales

En la siguiente tabla se puede ver la comparativa entre las horas planificadas y las reales:

TAREA	HORAS PLANIFICADAS	HORAS REALES
Análisis	45	50
Diseño	48	45
Implementación	160	159.5
Pruebas	15	10
Realización de video promocional	9	10
Repaso y finalización de la memoria	9	13
Reuniones	14	15
TOTAL DE HORAS	300	302.5

Como puede observarse, el desfase horario no ha sido muy grande. Las mayores diferencias se encuentran en la fase de análisis, fase que se alargó 5 horas debido a que al ser un proyecto doble aparte del análisis de cada aplicación, hay que hacer un análisis del proyecto completo. También fueron necesarias menos horas en la fase de pruebas, ya que al realizar el proyecto se comprobaba el correcto funcionamiento de los módulos implementados.

A pesar de los desfases en varios apartados, siempre se cumplieron las fechas de comienzo y finalización de las tareas.

En el siguiente gráfico que nos proporciona Jira, podemos ver como se han ido creando (puntos rojos) y cerrando (puntos verdes) las tareas de implementación:

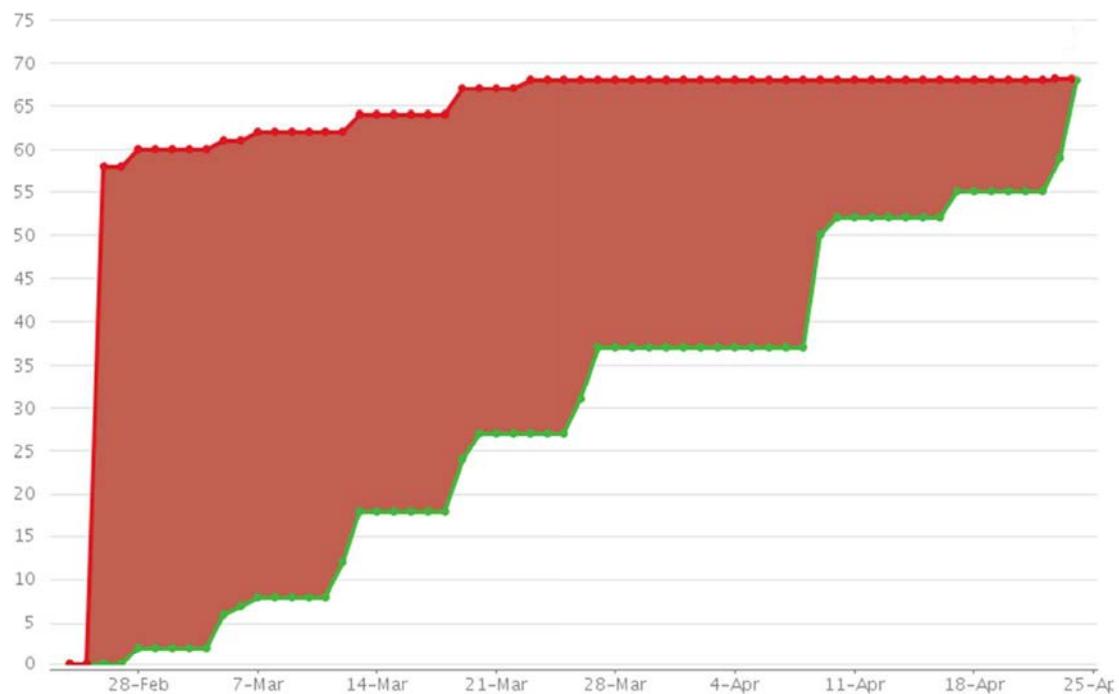


Gráfico resolución de tareas

Cumplimiento de objetivos

En cuanto a objetivos, el principal que era desarrollar un sistema completo y fácil de usar que se encargue de la logística necesaria para el buen funcionamiento de una empresa de paquetería a pequeña escala se ha cumplido con éxito.

La aplicación web simula de forma completa la experiencia que puede tener un cliente al realizar un pedido a una empresa de paquetería.

Mediante los servicios web, las aplicaciones Android pueden comunicarse con la aplicación web para obtener datos sobre pedidos o clientes.

Por último, se han realizado pruebas en diferentes navegadores y con diferentes dispositivos y se ha hecho un video promocional de la aplicación.

Posibles mejoras y ampliaciones

Aunque es una aplicación bastante completa, pueden realizarse varias mejoras como subir la aplicación a un servidor propio, de manera que no tengamos que depender de Google.

Una ampliación posible es la implementación de la pasarela de pago de los pedidos. No se ha implementado antes ya que la aplicación es una demo y no era uno de los objetivos del proyecto.

Por último se podría añadir un mecanismo de envío de SMS a los clientes.

Conclusión personal

La realización de este proyecto me ha servido para aplicar gran parte de los conocimientos adquiridos a lo largo de cuatro años de carrera. El hecho de haber realizado el proyecto junto con mi compañero en la empresa Digi International, me ha permitido hacerme una idea de cómo es el trabajo en equipo dentro de una empresa.

La asistencia a reuniones y demostraciones tanto en español a los empleados de Digi de Logroño y al tutor del proyecto, como en inglés a empleados de las oficinas de Digi de Minnetonka, me han servido para coger soltura a la hora de hablar en público y mostrar mi trabajo.

El trabajo me ha servido también para conocer la plataforma AppEngine de Google que me ha causado muy buena impresión, ya que permite subir aplicaciones web de forma gratuita y facilita el datastore para almacenar los datos de la aplicación.

Como conclusión destaco que he quedado muy satisfecho con cómo se ha desarrollado el proyecto y el resultado final.

Bibliografía

Para realizar el proyecto se ha utilizado información obtenida en las siguientes fuentes:

Bibliografía escrita:

- ROCHE, K., DOUGLAS, J., *Beginning Java™ Google App Engine*. New York, Apress, 2009.
- SANDERSON, D., *Programming Google App Engine*. Sebastopol, O'Reilly Media, 2010.
- MELONI J. C., *Programación HTML5, CSS3 y Javascript*. Madrid, Anaya, 2012.
- GOODMAN, D., MORRISON, M., NOVITSKI, P., RAYL, T. G., *Javascript® Bible 7th Edition*. Indianapolis, Wiley Publishing, 2010.
- Apuntes de la asignatura Programación de aplicaciones Web, 2012.
- Apuntes de la asignatura Desarrollo de Interfaces para Usuarios, 2013.

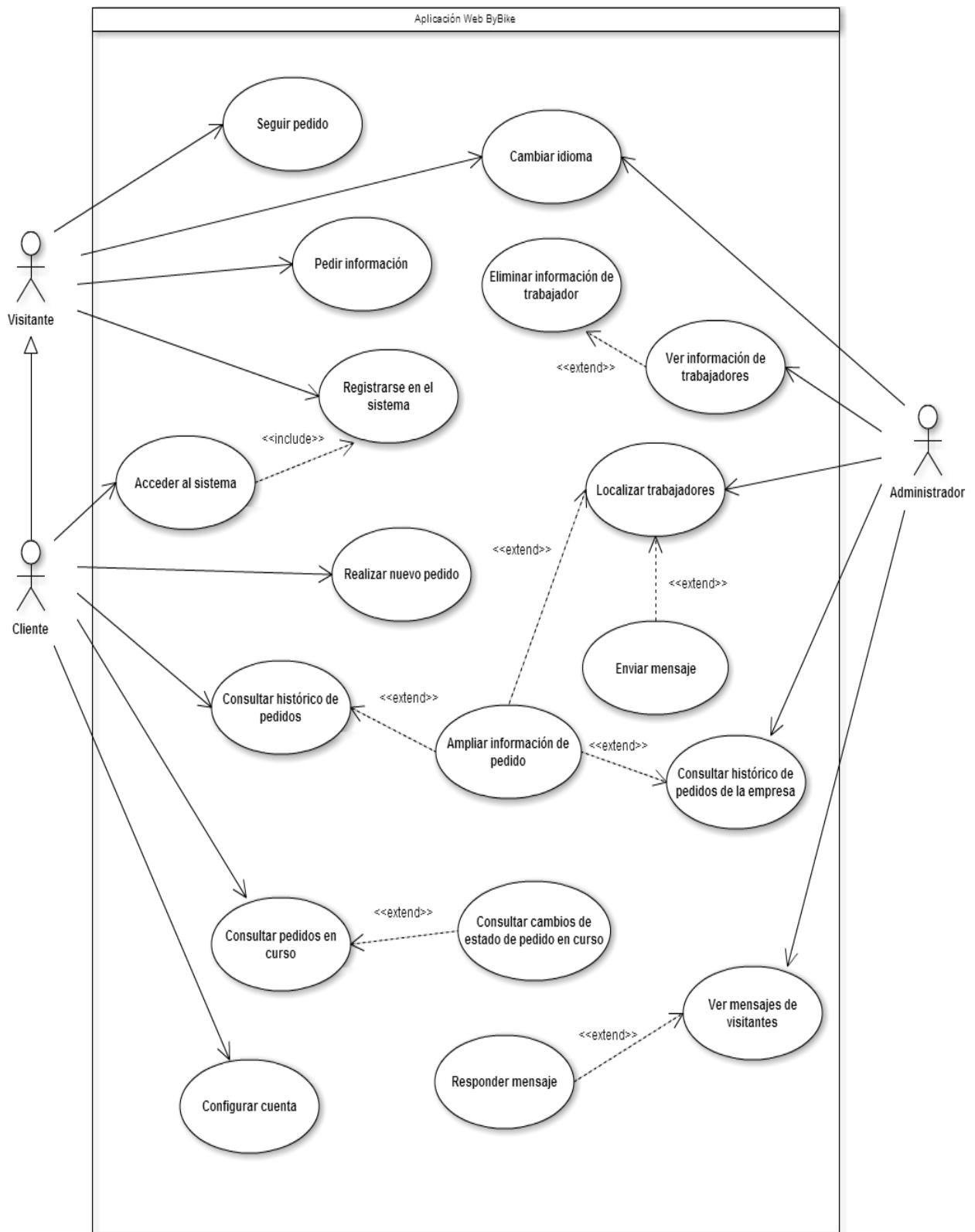
Páginas web:

- <https://developers.google.com/appengine/?hl=es>
- <https://developers.google.com/maps/documentation/distancematrix/?hl=es>
- <https://developers.google.com/maps/documentation/geocoding/?hl=es>
- <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=datastoreJD>
- <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cloudcomputing>
- <http://wowslider.com/es/>
- <http://dev7studios.com/nivo-slider/>

- <http://www.desarrolloweb.com/articulos/1513.php>
- <http://www.openstreetmap.es/>
- <http://html-color-codes.info/codigos-de-colores-hexadecimales/>
- <http://www.ideaschicago.com/como-elegir-los-colores-para-un-sitio-web/>
- <http://comercioelectronicoues01.blogspot.com.es/2012/08/google-app-engine.html>

Anexo Casos de Uso

Diagrama



Especificación de los casos de uso

Caso de Uso: Pedir información.
Objetivo: Pedir información sobre la empresa.
Actores: Visitante.
Precondiciones: No hay.
Pasos: <ol style="list-style-type: none">1. El visitante accede a la página de contacto.2. Rellena el formulario con los datos requeridos.3. Cuando el visitante hace clic en el botón enviar el mensaje es almacenado y se mostrará al administrador para que pueda responder.
Extensiones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje por pantalla indicándolo y no se envía el mensaje.

Caso de Uso: Registrarse en el sistema.
Objetivo: Un visitante se registra como cliente en la aplicación.
Actores: Visitante.
Precondiciones: No hay.
Pasos: <ol style="list-style-type: none">1. El visitante rellena el formulario de registro.2. La aplicación valida los datos.3. Se guardan los datos del nuevo cliente en la aplicación.4. El nuevo cliente accede al área de clientes. Se almacena una variable en la sesión que identifica al cliente para mantener el estado.
Extensiones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y no se realiza el registro.2.2. Si el nombre de usuario ya está utilizado la aplicación muestra un mensaje de error indicándolo y no se realiza el registro.2.3. Si la contraseña introducida en el campo de introducir contraseña y la contraseña no son iguales la aplicación muestra un mensaje de error indicándolo y no se realiza el registro

Caso de Uso: Seguir pedido.
Objetivo: Hacer seguimiento de un pedido.
Actores: Visitante.
Precondiciones: No hay.
Pasos: <ol style="list-style-type: none">1. El visitante introduce el identificador del pedido a seguir.2. La aplicación valida el identificador del pedido a seguir.3. La aplicación muestra una tabla con los cambios de estado que ha sufrido el pedido, la hora a la que se ha producido y posibles detalles del cambio de estado.
Extensiones: <ol style="list-style-type: none">1.1. Si el identificador no es válido el sistema mostrará un mensaje por pantalla para informar al usuario.

Caso de Uso: Acceder al sistema.
Objetivo: Un cliente accede a la aplicación.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema.
Pasos: <ol style="list-style-type: none">1. El cliente rellena el formulario de acceso con su nombre de usuario y contraseña.2. La aplicación valida los datos.3. El cliente accede al área de clientes. Se almacena una variable en la sesión que identifica al cliente para mantener el estado.
Extensiones: <ol style="list-style-type: none">2.1. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y el cliente no accede al sistema.

Caso de Uso: Consultar pedidos en curso.
Objetivo: Un cliente consulta la información de sus pedidos en curso.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes.
Pasos: <ol style="list-style-type: none">1. El cliente accede a la página de pedidos en curso.2. La aplicación muestra los pedidos en curso del cliente en un menú lateral.
Extensiones: <ol style="list-style-type: none">2.1. Si no hay pedidos en curso la aplicación se lo comunica al cliente mostrando un mensaje por pantalla.2.2. El cliente selecciona uno de los pedidos y se inicia el caso de uso consultar cambios de estado de pedido en curso.

Caso de Uso: Consultar cambios de estado de pedido en curso.
Objetivo: Un cliente consulta los cambios de estado de un pedido en curso.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes. El cliente se encuentra en la página de pedidos en curso.
Pasos: <ol style="list-style-type: none">1. El cliente ha elegido uno de los pedidos de la página de pedidos en curso.2. La aplicación muestra una tabla con los cambios de estado que ha sufrido el pedido, la hora a la que se ha producido y posibles detalles del cambio de estado.
Extensiones:

Caso de Uso: Consultar histórico de pedidos.
Objetivo: Un cliente consulta su histórico de pedidos.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes.
Pasos: <ol style="list-style-type: none">1. El cliente accede a la página del histórico de pedidos.2. La aplicación muestra en una tabla los datos principales de los pedidos del cliente.
Extensiones: <ol style="list-style-type: none">2.1. Si no hay pedidos la aplicación se lo comunica al cliente mostrando un mensaje por pantalla.2.2. El cliente filtra los pedidos por estado o fecha.2.3. El cliente selecciona uno de los pedidos y se inicia el caso de uso ampliar información de pedido.

Caso de Uso: Realizar nuevo pedido.
Objetivo: Un cliente realiza un pedido.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes.
Pasos: <ol style="list-style-type: none">1. El cliente rellena el formulario de nuevo pedido.2. La aplicación valida los datos.3. La aplicación asigna el pedido a un trabajador teniendo en cuenta el número de pedidos asociados al trabajador y el tiempo que tardará en realizarlos.4. La aplicación comunica al trabajador elegido que tiene un nuevo pedido.5. La aplicación comunica al cliente que el pedido se ha realizado con éxito indicando la hora máxima de entrega que dependerá del tipo de pedido.
Extensiones: <ol style="list-style-type: none">2.2. Si algún dato no es válido o no se han llenado campos requeridos la aplicación muestra un mensaje de error indicándolo y el cliente no accede al sistema.

Caso de Uso: Configurar cuenta.
Objetivo: Un cliente cambia la configuración de su cuenta.
Actores: Cliente.
Precondiciones: El cliente está registrado en el sistema y ha accedido al área de clientes.
Pasos: <ol style="list-style-type: none">1. El cliente accede a la página de configuración.2. Desde la página de configuración puede cambiar sus datos personales y su contraseña.3. La aplicación guarda los cambios.4. La aplicación comunica al cliente que los datos han sido cambiados con éxito.
Extensiones: <ol style="list-style-type: none">2.1. Fallo al intentar cambiar la contraseña.

Caso de Uso: Consultar histórico de pedidos de la empresa.

Objetivo: El administrador consulta el histórico de pedidos de la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El administrador accede a la página del histórico de pedidos de la empresa.
2. La aplicación muestra en una tabla los datos principales de los pedidos de la empresa.

Extensiones:

- 2.1. Si no hay pedidos la aplicación se lo comunica al administrador mostrando un mensaje por pantalla.
- 2.2. El administrador filtra los pedidos por estado o trabajador.
- 2.3. El administrador selecciona uno de los pedidos y se inicia el caso de uso ampliar información de pedido.

Caso de Uso: Localizar trabajadores.

Objetivo: El administrador localiza a los trabajadores de la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El administrador accede a la página de localización de trabajadores.
2. La aplicación muestra un mapa en el que aparecen los trabajadores conectados, su tarea actual y las tareas pendientes.

Extensiones:

- 2.1. El administrador selecciona uno de los pedidos del trabajador y se inicia el caso de uso ampliar información de pedido.
- 2.2. El administrador selecciona un trabajador para enviarle un mensaje y se inicia el caso de uso enviar mensaje.

Caso de Uso: Enviar mensaje.

Objetivo: El administrador envía un mensaje a un trabajador de la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

El administrador ha seleccionado un trabajador.

Pasos:

1. El administrador ha elegido un trabajador a quien enviar un mensaje.
2. El administrador escribe el mensaje y hace clic en enviar.
3. La aplicación envía el mensaje al trabajador a través de Etherios.
4. La aplicación comunica al administrador que el mensaje ha sido enviado con éxito.

Extensiones:

- 1.1. El trabajador no está conectado.
- 2.1. El mensaje está vacío y no se envía.
- 4.1. El mensaje no se ha enviado bien y la aplicación lo comunica mediante un mensaje en pantalla.

Caso de Uso: Ampliar información de pedido.

Objetivo: Ver la información ampliada de un pedido.

Actores: Cliente o Administrador.

Precondiciones: El usuario debe haber accedido al sistema.

Pasos:

1. El cliente o el administrador deciden ampliar la información que ven de un pedido.
2. La aplicación muestra los detalles del pedido.

Extensiones:

- 2.1. Al hacer clic en el botón de volver atrás se vuelve a la página desde la que se accedió a los detalles del pedido.

Caso de Uso: Ver información de trabajadores.

Objetivo: Ver la información de los trabajadores de la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El administrador accede a la página de información de trabajadores.
2. El sistema muestra la información de los trabajadores en una lista.

Extensiones:

- 2.1. El administrador selecciona un trabajador y pulsa el botón eliminar información. De esta manera se inicia el caso de uso eliminar información de trabajador.

Caso de Uso: Eliminar información de trabajador.

Objetivo: Eliminar la información de un trabajador de la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El trabajador ha seleccionado el trabajador del que eliminar la información y hace clic en el botón eliminar información.
2. La aplicación comunica que la información del trabajador ha sido eliminada con éxito.

Extensiones:

- 2.1. La información del trabajador no puede ser eliminada porque está conectado o tiene tareas pendientes y se comunica al administrador que no se puede eliminar la información.

Caso de Uso: Ver mensajes de visitantes.

Objetivo: Ver los mensajes que los visitantes han enviado a la empresa.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El administrador accede a la página de mensajes.
2. El sistema muestra una lista con los mensajes que los visitantes han enviado a la empresa.

Extensiones:

- 2.1. Si no hay mensajes se muestra un mensaje por pantalla para indicarlo.
- 2.2. El administrador selecciona un mensaje y pulsa el botón responder mensaje. De esta manera se inicia el caso de uso responder mensaje.

Caso de Uso: Responder mensaje.

Objetivo: Responder un mensaje que ha enviado un visitante.

Actores: Administrador.

Precondiciones: El administrador ha accedido al área de administración.

Pasos:

1. El administrador escribe la respuesta al mensaje del visitante.
2. La aplicación genera un email con la respuesta y lo envía al correo que indicó el visitante.
3. La aplicación cambia el estado del mensaje a contestado para que no siga apareciendo.
4. La aplicación comunica al administrador que el mensaje ha sido contestado.

Extensiones:

- 1.1. Si el mensaje está vacío no se envía el mensaje.
- 4.1. No se puede enviar el mensaje. La aplicación genera un mensaje de error y lo muestra por pantalla.

Caso de Uso: Cambiar idioma.

Objetivo: Cambiar el idioma de la aplicación.

Actores: Visitante, Cliente o Administrador.

Precondiciones: No hay.

Pasos:

1. Un visitante, un cliente o el administrador decide cambiar el idioma de la aplicación.
2. Selecciona la bandera del idioma al que quiere cambiar.
3. Se modifica el idioma de la página almacenando en la sesión una variable que indique el idioma elegido para que se mantenga durante el resto de la navegación.

Extensiones: