Aplicación de seguimiento de pedidos

Nombre Estudiante
Antonio Martir Millán

Nombre Consultor
Roger Montserrat Ribas

Fecha de entrega
04 Enero 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	Seguimiento de pedidos y ofertas comerciales
Nombre del autor:	Antonio Martir Millán
Nombre del consultor:	Roger Monserrat Ribes
Fecha de entrega:	1/2017
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo (máximo 250 palabras):

Desarrollo de una aplicación para dispositivos móviles para mejorar la comunicación de una empresa con sus clientes, el cliente de la empresa podrá consultar el estado en tiempo real, recibir ofertas,…

Esta aplicación tiene sentido cuando un proveedor de productos y/o servicios está interesado en proporcionar información útil y servicios adicionales al cliente.

Por ejemplo: enviar una notificación al móvil del cliente indicándole exactamente cuándo le será entregada la mercancía, otro ejemplo sería que desde la aplicación el cliente consulte el estado actual (preparando, en ruta…), también es un medio para hacer llegar a los clientes ofertas comerciales.

Abstract (in English, 250 words or less):

Development of an application for mobile devices to enhance the communications of the enterprise with their customers, check order status or receive news.

This application is useful when a product or services provider is interested in providing important information to the customer.

For example: To send a notification to the phone of a customer with the time of delivery, the customer can also use the application to query about the current status of an order (handling, shipping…), this application allows also to provide commercial offers to customers.

Palabras clave (entre 4 y 8):

pedido estado información cliente proveedor publicidad

Índice

Tabla de contenido

1. Introducción

1.1 Contexto y justificación del Trabajo

Las empresas de venta de productos necesitan agilizar y facilitar la comunicación con sus clientes, puede utilizarse el dispositivo que los clientes llevan siempre a mano para mejorar la relación con los clientes, enviando el estado de su pedido e informándoles de ofertas comerciales.

La aplicación inicialmente enlazará con un ERP, pero se estructura de forma que puede conectarse con otros ERPs. Para ello se diseña un servicio Web (WebService) que será adaptado para cada uno de los ERPs.

La primera implantación se realizará sobre la empresa DONPIN 2002 S.L.

La empresa DONPIN 2002 S.L. proporciona bebidas y productos de alimentación a puntos de venta al por menor. La empresa desea mejorar la relación con sus clientes proporcionándoles información útil durante el transcurso de un pedido, usando una aplicación móvil personalizada. El cliente podrá consultar el estado de los pedidos, así como el histórico de pedidos realizados. En la actualidad se proporciona únicamente a los clientes una fecha aproximada de entrega. Con el nuevo sistema se mejora la comunicación con el cliente proporcionando un mejor servicio y abriendo la puerta a nuevas posibilidades, como notificaciones de ofertas comerciales.

1.2 Objetivos del Trabajo

Creación de un producto: aplicación para dispositivos móviles. Esta aplicación debe ser capaz de proporcionar información al cliente sobre el estado de sus pedidos, y ser una herramienta de consulta, además será un **nuevo medio de comunicación entre la empresa y el cliente**. Aunque las funcionalidades iniciales serán limitadas y se implementa en una única empresa, el producto se diseña para que sea fácilmente ampliable y personalizable.

1.3 Enfoque y método seguido

La solución en grandes empresas para el seguimiento de los pedidos pasa por utilizar el sistema de mensajería electrónica EDI (UN/EDIFACT), usando documentos predefinidos estandarizados: ORDER, DELFOR (delivery schedule message), DESADV(despatch advice message), sin embargo, las pequeñas empresas no disponen de un sistema capaz de procesar estos mensajes.

Las aplicaciones preventa/postventa del mercado están enfocadas sobre todo en la fuerza de ventas, esto es en el rendimiento de los vendedores y el soporte a estos, pero no en el servicio a clientes. Por otro lado, cada tienda online implementa su propio sistema de notificación de envíos (Amazon, Ebay…), integrándose únicamente con sus aplicaciones.

Proporcionar una aplicación instalable en el móvil del cliente es una solución fácil y práctica (desde el punto de vista del cliente), además en un futuro la aplicación podría ser ampliada con otras funcionalidades.

Será necesaria la implementación de una interfaz de servidor, esta interfaz de servidor será bidireccional, enviará notificaciones a los dispositivos y podrá ser consultada (consulta sobre estado de pedidos). Se implementarán las funcionalidades necesarias en el servidor para cada funcionalidad que la aplicación móvil requiera.

Se evaluarán las principales tecnologías que podrían usarse en el proyecto, decidiendo la más adecuada para este proyecto. Se tienen en cuenta las plataformas mayoritarias Android y IOS, pero se valorará que la tecnología escogida tenga la posibilidad de incluir otras plataformas fácilmente.

1.4 Planificación del Trabajo

Se utilizará Scrum para la planificación del proyecto, en cada uno de los Sprints se alcanzarán unas metas hasta conseguir el resultado, producto final. En el capítulo 2 se detalla la planificación del trabajo.

Se realizarán 3 entregas, que coinciden con cada Sprint:

1.Diseño: Prototipo de la aplicación, mientras se realiza el prototipo se prepararán las bases para el desarrollo, herramientas y servicios (backend).

2.Aplicación con funcionalidad parcial: Se realizará una entrega con funcionalidad limitada, de esta forma se podrá comprobar el correcto avance del proyecto.

3.**Aplicación final.** Entrega de la memoria y presentación del producto final.

1.5 Productos obtenidos

El producto obtenido será una aplicación para dispositivos móviles multiplataforma, que permitirá mejorar la comunicación de una empresa de distribución de productos con sus clientes finales.

1.6 Capítulos de la memoria

La memoria consta de los capítulos: planificación, estimación de costes, diseño, implementación y pruebas, guía de desarrollo de nuevas funcionalidades, comentarios sobre el desarrollo y conclusiones.

2. Planificación

2.1 Equipo y sistema de desarrollo

En el plan de implantación se utilizará desarrollo ágil usando **Scrum** como método de trabajo y seguimiento del proyecto y **XP** (eXtreme Programming) como filosofía de desarrollo. Se intentará obtener una versión operativa lo antes posible, aunque se disponga de pocas funcionalidades, para conseguir retroalimentación de la experiencia de usuario lo antes posible.

2.2 Product Owner (propietario del producto)

El Product Owner es Mao Dong Chen, responsable de sistemas informáticos de DONPIN 2002, S.L. Se compromete a realizar las tareas propias de este rol, tales como definir cómo será el producto, qué funcionalidades tendrá, control de calidad, invirtiendo el tiempo necesario en el seguimiento del producto, y reuniéndose las veces necesarias con el Scrum Master.

2.3 Scrum Master y equipo de desarrollo

La función de Scrum Master y equipo de desarrollo recae sobre una única persona, el autor de este trabajo.

El Scrum Master estará en continua comunicación con el Product Owner, informando de cualquier incidencia, validará las funcionalidades de producto requeridas por el Product Owner.

2.4 Product Backlog (ficha del producto)

En el Product Backlog se introducen las historias de usuario (funcionalidades de la aplicación) y otros requisitos (funcionalidades técnicas necesarias), si hay muchas historias pueden agruparse en temas, estas historias de usuario son muy genéricas y deben dividirse en tareas, las tareas se asignarán en los Sprints. El Product Backlog es gestionado por el Product Owner, pero en caso de modificación, el Scrum Master y el Product Owner acordarán la modificación, con las consecuencias que tenga en los Sprints siguientes.

Se proporciona el Product Backlog en un archivo a parte, ProductBacklog.xlsx.

2.5 Sprints

Se definen varias fases en el desarrollo (Sprints), fijando unos objetivos en cada Sprint. El Sprint actual tendrá el detalle de tareas a realizar, los Sprints siguientes serán menos detallados. Después de cada sprint, el equipo y el Product Owner revisan la ejecución del sprint (Sprint Review) y confecciónan de forma detallada el siguiente sprint (Sprint Planning).

2.6 Sprint Backlog

En el Sprint Backlog se detallan las tareas del Sprint actual, es gestionado por el Scrum Master, en él se anota la prioridad de cada tarea y su tiempo de implementación estimado, también las incidencias o impedimentos que se encuentran durante el desarrollo.

Se proporciona el Sprint Backlog en SprintBacklog.xlsx

2.7 Sprint 0

En el sprint 0 se define el contenido y forma del producto de forma general, se completa la versión inicial del Product Backlog y se planifica detalladamente el Sprint 1. En esta fase ya se define la tecnología y herramientas de desarrollo a utilizar y se realiza un análisis inicial de costes.

2.7 Sprint 1

En el sprint 1 se diseñará un prototipo de la aplicación, también se preparará el entorno de desarrollo. Este proyecto requiere la programación de un servicio de servidor (backend) que proporcione información en tiempo real a la aplicación móvil. Se implementará un Webservice lo liviano posible que se comunique con el ERP de DONPIN 2002 S.L. para obtener la información que será proporcionada por el servicio. Se diseñará un interfaz que permita la futura interacción con otros ERP.

2.8 Sprint 2

En el sprint 2 se implementará la aplicación usando la tecnología seleccionada.

2.9 Sprint 3

En el sprint 3 se finalizará el desarrollo de la aplicación, invirtiendo gran parte del tiempo de documentación en la creación de una video presentación del proyecto.

2.10 Plan de pruebas

Se he definido un plan de pruebas. Un documento que detalla el procedimiento a seguir en las pruebas y los distintos tipos de pruebas. Además un archivo Excel que detalla las pruebas específicas. En el anexo 8.2 se detalla el plan de pruebas.

3. Costes

3.1. Recursos humanos

Se identifica el coste en horas en el Sprint Backlog. Se anotan las estimaciones y las diferencias.

En la fase de planificación se han invertido 47 horas. Se han planificado 3 Sprints, en cada Sprint con unas 85 horas de media en cada Sprint. Lo que da un total de unas 255 horas (más 47 de preparación).

El Product Owner también invertirá horas pero no están incluidas en el cálculo.

3.2. Herramientas de hardware

Ya se disponen de herramientas de hardware (PCs) para que el equipo pueda desarrollar la aplicación. No se dispone de hardware, de momento, para la generación y publicación de la aplicación para plataformas Apple (IPhone).

Servidor de contenidos: La empresa donde se implanta la aplicación ya dispone de un servidor dedicado para atender las peticiones en tiempo real de la aplicación móvil.

3.3. Herramientas de software

Las herramientas seleccionadas son gratuitas:

- Ionic Framework. MIT license (https://opensource.org/licenses/MIT)
- Apache Cordova.(http://www.apache.org/licenses/LICENSE-2.0)
- Microsoft Visual Studio Code.
    - Fuente: MIT license (https://opensource.org/licenses/MIT)
    - Producto: https://code.visualstudio.com/license

3.4. Licencias de derechos de distribución en mercados de aplicaciones.

Cuando se distribuya la aplicación móvil por los canales oficiales provistos por las distintas plataformas se indican los precios aproximados.

- Apple Store (Apple IPhone, IPad…): 99 Euros anuales.
- Google Play Store (Dispositivos Android): 25 Euros. Cuota única.
- Windows Marketplace (Windows Phone): 75 Euros. Cuota única.

4.Diseño

4.1. Usuarios y contexto de uso

4.1.1 Usuario final de la aplicación móvil

La aplicación será usada por los clientes de una empresa distribuidora de productos, para conocer los pedidos en curso y recibir notificaciones acerca del estado de sus pedidos. También recibirá, en la página principal información personalizada proporcionada por la empresa (ofertas, campañas publicitarias…)

4.1.2 Usuarios de la empresa que proporciona el servicio

Los usuarios de la empresa que distribuye los productos no usarán la aplicación móvil, sin embargo, las acciones que realicen en el aplicativo de la empresa (ERP) tendrán consecuencias sobre los usuarios finales. Por ejemplo, cuando un operador de la aplicación indique que un camión ha salido, los clientes finales recibirán una notificación. Los usuarios de la empresa tendrán capacidad para cambiar la página principal de la aplicación, globalmente o individualmente para cada cliente.

4.1.3 Perfil de los usuarios de la aplicación móvil

Los usuarios serán muy diversos, pero se tiene en cuenta bastante la implantación que se realizará en DonPin 2002 S.L. Los usuarios tendrán, generalmente, pocos conocimientos sobre tecnología, serán vendedores de productos de alimentación, una gran parte de estos usuarios tienen como lengua materna el chino, por lo que la aplicación debe contemplar este idioma.

4. Diseño conceptual

4.2.1 Escenarios de uso. Ejemplos prácticos

Un usuario consulta la lista de pedidos pendientes de recibir para repasar si debe pedir algo más.

Un usuario recibe una notificación acerca del envío de un pedido, recibe la información de que el pedido está en camino.

Un usuario podría recibir una notificación de oferta acerca de productos que compra habitualmente.

Otros posibles escenarios que probablemente no serán implementados inicialmente:, consulta del histórico de facturas, estado financiero (pagos pendientes)…

4.2.2. Funcionalidades

- Pantalla principal dinámica, contenido controlado por la empresa
- Entrada en el sistema con selección de idioma
- Lista de pedidos pendientes de recibir
- Histórico de pedidos
- Visualizar detalle de un pedido
- Recepción de notificaciones de envío de un pedido.
- Cambio de la contraseña
- Salida del sistema

Se detallan las funcionalidades básicas, que se implementarán en el proyecto, sin embargo, la aplicación se planifica para que esta lista de funcionalidades pueda crecer a medio plazo.

4.2.3 Decisiones de diseño

Dado el perfil de los usuarios la aplicación será muy sencilla, y se evitará en la medida de lo posible el uso del teclado.

El usuario debe autentificarse en el sistema, se le proporcionará, por parte de la empresa, un nombre de usuario y una clave, por lo que será necesaria una pantalla de entrada en el sistema. La aplicación recordará al usuario, excepto si el usuario lo solicita expresamente (opción salir en el menú principal).

Inicialmente se diseñó el prototipo como un simple lista de opciones, en el producto final se usa un sistema de menú desplegable desde la izquierda, enfoque moderno que utilizan algunas aplicaciones, que tiene la ventaja de dar protagonismo a la página principal, cuyo contenido es establecido por la empresa en todo momento.

Navegación: se usa la barra superior para navegar por las pantallas de la aplicación, indicando en el título dónde se encuentra el usuario, con opción a volver a la pantalla anterior, este método es sencillo e intuitivo, asemejándose ligeramente a la navegación de los exploradores de internet.

4.3. Arquitectura

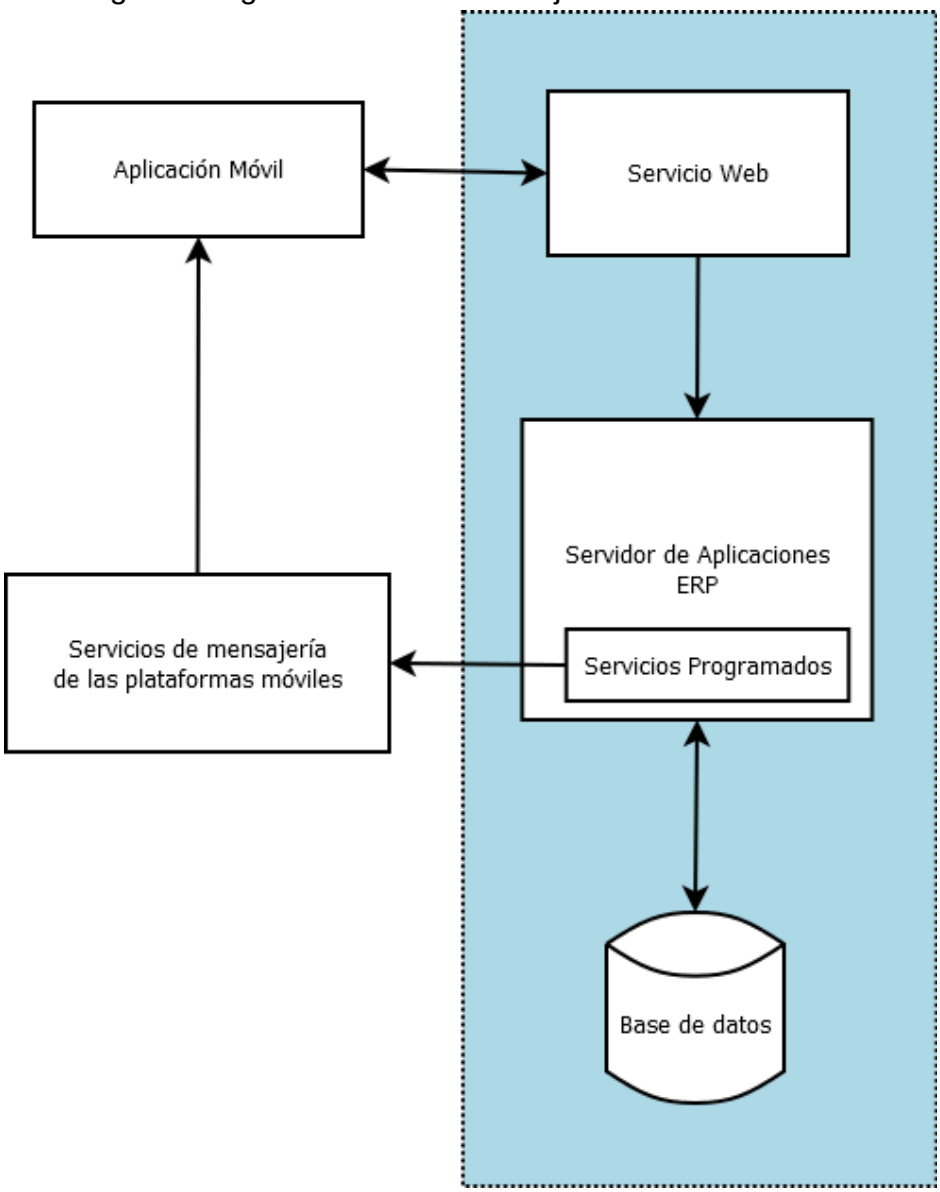
4.3.1 Flujo de datos

Los datos serán consultados por la aplicación cliente a un servicio web (API Rest), este servicio web se comunica con el servidor de aplicaciones (ERP) de la empresa. El servidor de aplicaciones es el que realmente actualiza la base de datos.

El servidor de aplicaciones tiene servicios programados, estos servicio programados se encargan de enviar los mensajes a los sistemas de mensajería de cada plataforma, como Google Cloud Messaging (https://developers.google.com/cloud-messaging/).

Esta arquitectura nos permite situar cada elemento en un lugar físico distinto, aunque en este caso, el servicio web, servidor de aplicaciones y base de datos se encuentran en el mismo lugar.

En la siguiente figura se visualiza el flujo de la información:



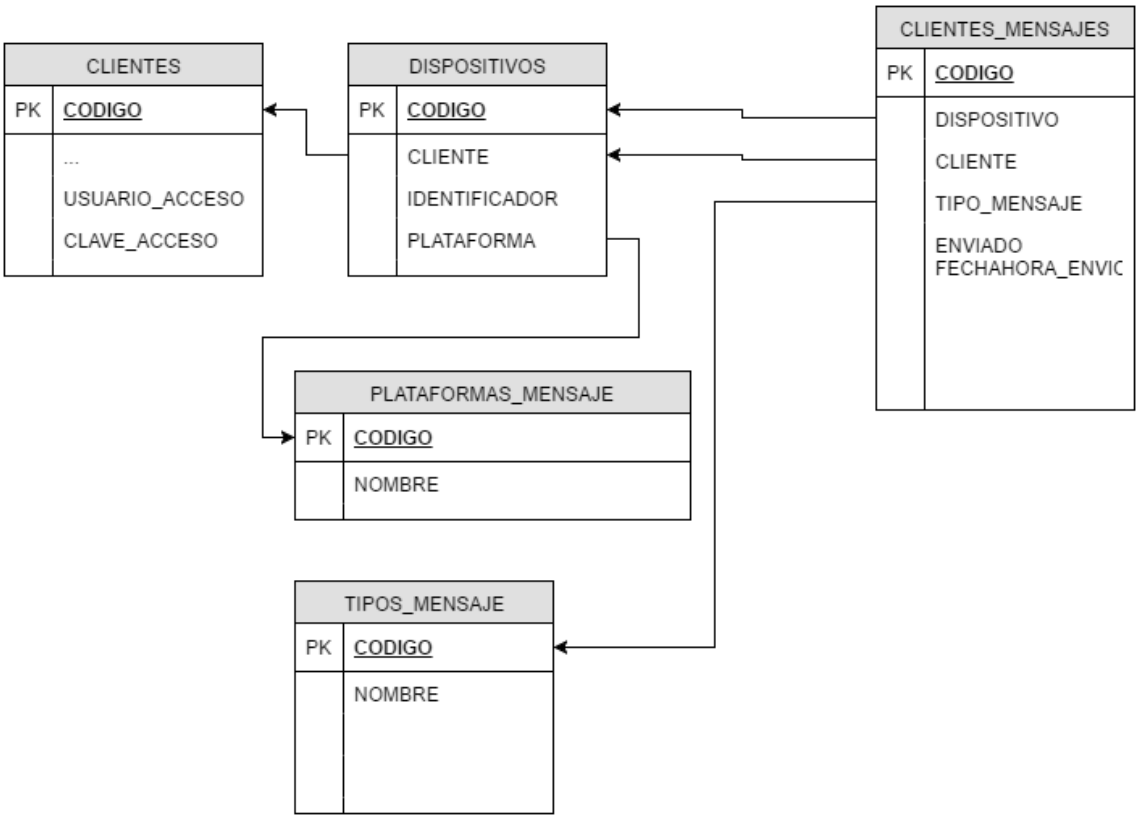
4.3.2. Cambios en el esquema de la base de datos

Se necesita un nombre de usuario y una clave en la ficha de cada cliente. La clave se almacenará mediante una codificación unidireccional tipo hash (MD5 o preferiblemente SHA1), de esta forma respetamos la privacidad de las contraseñas de los usuarios.

Se creará una tabla de dispositivos registrados, de forma que un cliente pueda registrarse con más de un dispositivo para recibir las notificaciones (padre, hijo, encargado…).

Se creará una tabla de mensajes a enviar.

Los mensajes serán procesados por un servicio específico del ERP, procesará la lista de mensajes, enviando a los dispositivos registrados, en caso de fallo de comunicación, se reintentará el envió en la siguiente iteración del servicio.



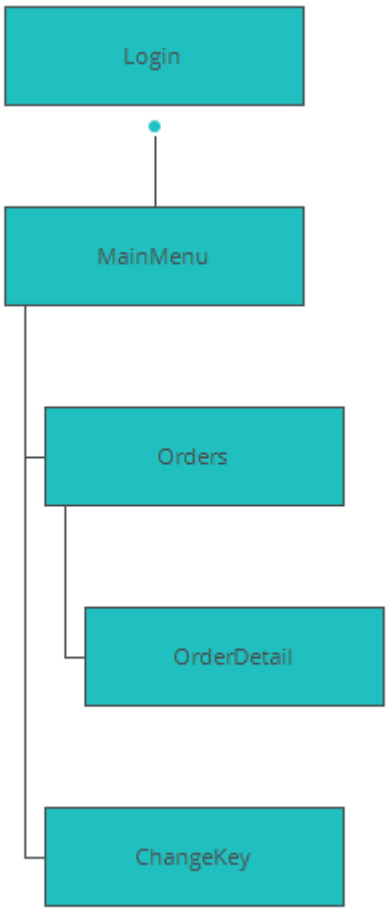
4.4.Evaluación

4.4.1. Iteraciones del diseño

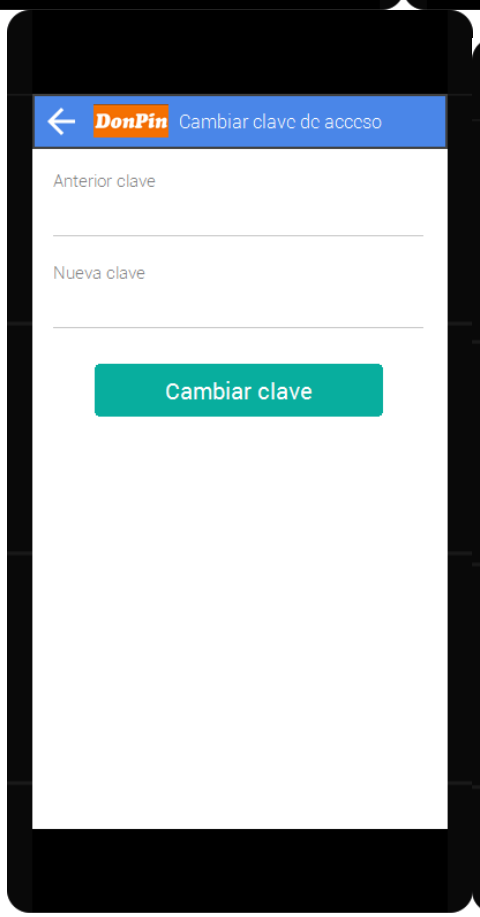
El Product Owner revisó el diseño, pero sólo se realizaron dos iteraciones. Es posible que el diseño sea revisado nuevamente en nuevas iteraciones, en concreto los detalles acerca del pedido, o la forma en que se deben presentar las líneas del pedido.

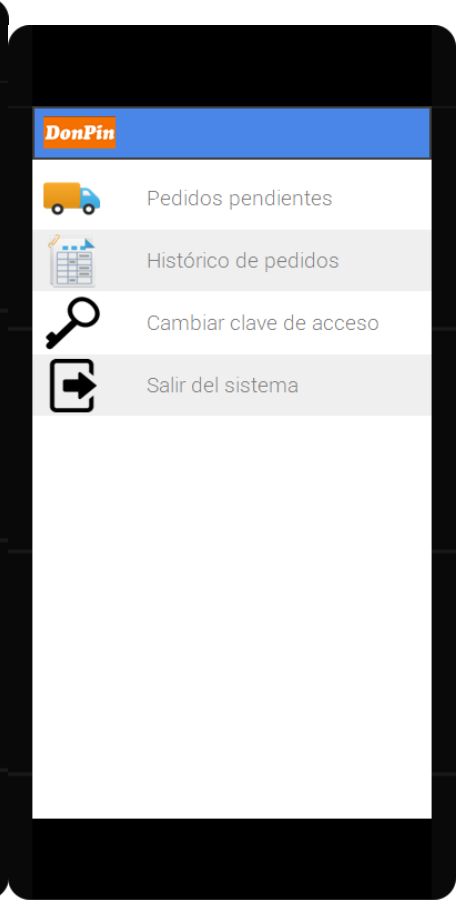
4.4.2. Mapa de pantallas del diseño

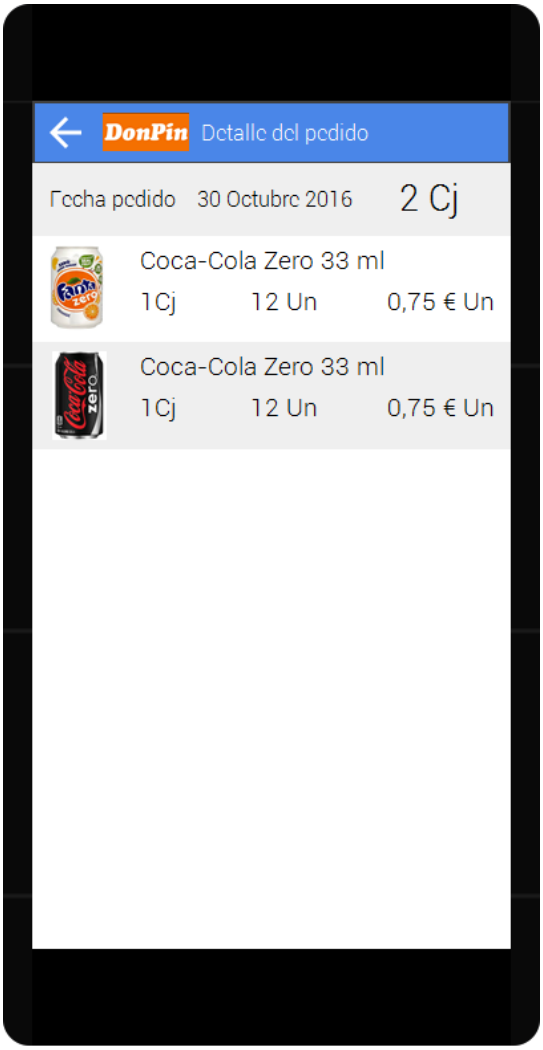
Se muestra el mapa de navegación de la aplicación.



4.4.3. Capturas de pantalla






DonPin
Dctallc dcl pcdido
Γccha pcdido 30 Octubrc 2016
2 Cj
Coca-Cola Zero 33 ml
1Cj 12 Un 0,75 € Un
Coca-Cola Zero 33 ml
1Cj 12 Un 0,75 € Un

4.4.4. Prototipo interactivo

La herramienta utilizada (JustInMind) permite interaccionar con el prototipo.

Puede accederse mediante el siguiente enlace:

https://www.justinmind.com/usernote/tests/18150236/18595754/22137393/index.html

Se detallan las interacciones principales.

- Cambio de idioma en la pantalla de entrada
- Pulsando en el botón entrar se accede al botón principal
- En el menú principal se puede acceder a los prototipos de lista de pedidos y cambio de clave
- Desde la pantalla de pedidos puede navegarse a la pantalla de detalle del pedido.

Nota: JustInMind es muy sensible al movimiento del ratón, de forma que, si se pulsa el botón del ratón, pero se mueve ligeramente, no se ejecutará la acción.

4.5. Seguimiento de la planificación durante la fase de diseño

Se han finalizado todas las tareas del Sprint 1, excepto la implementación del WebService, se estima que se finalizará en 10 horas de trabajo, se traslada la finalización del WebService al Sprint 2. Aunque el volumen de trabajo calculado fue más o menos correcto, se ha producido un desfase en el número de horas dedicadas, en el siguiente Sprint se realizará un mayor esfuerzo para cumplir las horas planificadas.

Se ha actualizado el Product Backlog con la nueva información de planificación.

5.Implementación y pruebas

5.1. Introducción

El desarrollo y las pruebas se han desarrollado simultáneamente, sin embargo, conforme se ha avanzo el desarrollo el número y frecuencia de las pruebas se ha incrementado.
Para mejorar el resultado final de la aplicación se han realizado cambios sobre el diseño inicial, estos cambios se han reflejado en el Product Backlog, Sprint Backlog y el plan de pruebas, estos cambios supondrán algo más de trabajo, pero es asumible.

5.2. Mejoras en el diseño y ampliación de funcionalidades.

5.2.1. Pantalla principal

En la pantalla principal se ha cambiado el menú principal, que ocupaba toda la pantalla, por un menú lateral desplegable, esto deja el espacio de la pantalla principal útil para promocionar los productos de la empresa. Esta pantalla se actualizará periódicamente por la empresa (oferta de Navidad etc..).

5.2.2. Detalle de producto (ficha logística)

El cliente podrá entrar en el detalle de un artículo, visualizando la imagen y datos adicionales del producto como las unidades por caja.

5.2.3. Notificaciones

Se han cambiado las estructuras de forma que un cliente puede tener más de un usuario y más de un dispositivo por usuario registrados en el sistema para recibir notificaciones. Las notificaciones serán enviadas a todos los dispositivos del cliente.

5.2.4. Lectura de código de barras

Desde la página Producto del menú principal, puede leerse un código de barras de un producto usando la cámara, y a continuación visualizar su ficha.

5.3. Tecnologías de desarrollo evaluadas

5.3.1. Desarrollo nativo

En el desarrollo nativo es necesario la programación de la aplicación en el lenguaje propio de cada plataforma:

- Java en Android
- Swift-Ojective C en IOS
- C# en Windows Phone

El problema es que debemos programar la aplicación tres veces y mantener las tres ramas de desarrollo en cada modificación, esto supone una inversión muy grande de tiempo, no sólo en la duplicidad de la programación, sino en la formación necesaria para conocer cada uno de los lenguajes y las particularidades de la plataforma, el trabajo de implantación del diseño del interfaz también se triplica.

Pero este tipo de desarrollo nos proporciona algunas ventajas, por ejemplo: la experiencia de usuario puede ser personalizada para cada plataforma, la aplicación tendrá un aspecto integrado con el sistema operativo.

5.3.2. Desarrollo híbrido (HTML JavaScript)

Se llama aplicación híbrida a aquella aplicación que no está programada con el lenguaje nativo de la plataforma (Java,Swift-Ojective C, C#), normalmente están desarrolladas con tecnologías web, que construyen una aplicación ejecutable en distintas plataformas móviles, funcionando la aplicación realmente incrustada en un navegador web, pero con acceso a recursos locales del dispositivo (cámara, GPS, archivos..).
Por ejemplo: Apache Cordova (PhoneGap).

5.3.3. Desarrollo nativo en un único lenguaje (Xamarin Forms)

Xamarin forms (https://developer.xamarin.com/guides/xamarin-forms/) es una forma de desarrollo que no entraría en la clasificación de aplicación híbrida, pero tampoco en totalmente nativo, ya que las herramientas usadas y el lenguaje pueden no coincidir con las proporcionadas por la plataforma.
Xamarin Forms nos permite programar una vez (en C#), diseñar una vez (diseñador propio de Xamarin Forms) y distribuir la aplicación en Android,IOS y Windows Phone.
Como contrapartida debemos ceñirnos a los controles visuales multiplataforma que proporciona Xamarin Forms, siendo difícil añadir nuevos (se requeriría una implementación distinta por plataforma), no tenemos la flexibilidad de diseño que proporciona HTML y CSS.
La ventaja es que la aplicación resultante usa los controles (Widgets) nativos de cada plataforma y, por tanto, se integra bien en cada plataforma.

5.4. Tecnología de desarrollo seleccionada

5.4.1. Ionic Framework versión 2

Se ha seleccionado Ionic Framework versión 2 para el desarrollo de la aplicación. Ionic 2 es un framework de desarrollo de aplicaciones híbridas, que utiliza Apache Cordova para el soporte de las distintas plataformas.
Ionic 2 tiene como base Angular2, con plantillas HTML (templates), y programación con TypeScript, siguiendo el paradigma Modelo-Vista-Controlador (Model-View-Controller), donde el modelo son los datos, la vista es el archivo HTML y el controlador es el código TypeScript, el uso del lenguaje Typescript es, en mi opinión, la mejor ventaja en relación con Ionic versión 1.

5.4.2. Motivos principales

Se ha seleccionado esta tecnología para **reducir el tiempo de desarrollo**, ya que como mínimo se ejecutará la aplicación en dos plataformas: Android y IOS, añadir una plataforma más como Windows Phone no supondrá un gran problema.
También ha influido mucho la posibilidad de **reutilizar** el código para la programación de un **portal web** en un futuro.

5.4.3. Ventajas

- Un solo código fuente y un solo diseño para las distintas plataformas. Menor tiempo de desarrollo.
- En relación con Ionic versión 1, el uso de TypeScript mejora la productividad (detección precoz de errores, se dispone de autocompletar durante la programación)
- Gran parte de la programación y diseño puede aprovecharse si se desea crear un portal web parecido a la aplicación móvil.

5.4.4. Inconvenientes

- Menor integración visual con cada plataforma, aunque Ionic 2 mejora ligeramente este aspecto, la aplicación intente emular el comportamiento y aspecto de cada plataforma (iconos, comportamiento de listas…)
- Ionic 2 se encuentra en fase Release Candidate, puede que encontremos algún problema de estabilidad durante el desarrollo.
- Complejidad general del sistema, se usan muchas tecnologías distintas para generar la aplicación: TypeScript,Html,CSS,Angular2,Ionic2 Apache Cordova.

5.5. Implementación del Backend, WebService

5.5.1. Tecnología utilizada

Para la programación del Backend se ha usado Visual Studio 2015 Community Edition, programando una aplicación ASP.Net WebAPI, programado en C#. La tecnología WebAPI proporciona un entorno multihilo de alto rendimiento para el proceso de peticiones HTTP.

5.5.2. Operaciones.

Soporta operaciones de consulta http (querys), que devuelven estructuras Json. También se encarga de proporcionar las imágenes a la aplicación cliente.

5.5.3. Ejemplos de las operaciones

Login

Comprueba los datos de autenticación de un usuario.
http://servidor:puerto/api/login?username=usuario
&password=clave&language=ES&device_id=xxxxx

Devuelve un objecto Json con la propiedad Message, OK si es correcto o el mensaje de error correspondiente.

Obtención de pedidos

```
http://servidor:puerto/api/orders?username=usuario
&password=clave&language=ES&count=10&skip=0&order_id=0
```

Puede especificarse la obtención de un pedido en concreto de un rango de pedidos, empezando por el más reciente. Devuelve un array Json con los datos de los pedidos.

Detalle de un pedido
Obtiene las líneas de un pedido
`http://servidor:puerto/api/orderdetail?username=usuario&password=clave&language=ES&order=1625364`

Devuelve un array Json con las líneas del pedido.

Detalle de un producto
Obtiene la ficha de un producto.
`http://servidor:puerto/api/product?username=usuario&password=clave&language=ES&product_id=1001`

Devuelve un array Json con una línea conteniendo los datos del producto.

Contenido de la página principal
Devuelve una array con los datos a mostrar en la página principal, textos e imágenes, el parámetro cms_code podrá utilizarse en un futuro para descargar diferentes tipos de contenido.
`http://servidor:puerto/api/cms?username=usuario&password=clave&language=ES&cms_code=1`

Salida del sistema
El dispositivo ya no recibirá notificaciones.
`http://servidor:puerto/api/unregister?username=usuario&password=clave&language=ES&device_id=xxxxx`

Imágenes
Los enlaces de las imágenes se incluyen en las propias estructuras Json devueltas por lo que no se requieren acciones especiales en la aplicación móvil.
Seguridad
En el servidor de producción se usará SSL (https) para las comunicaciones.

5.6. Instalación del entorno de desarrollo

5.6.1. Plataformas soportadas

Las herramientas de construcción y desarrollo soportan múltiples plataformas, se soportan Microsoft Windows (recomendado Windows 10), Mac OS (X El Capitan o superior) y Linux (recomendado Ubuntu).
Para la construcción de la aplicación para móviles con sistema Apple IOS es necesario un sistema Mac OS, aunque existen servicios de pago para generar estos paquetes en la nube como Adobe PhoneGap https://build.phonegap.com/

5.6.2. Ionic 2 framework (más Apache Cordova)

El proyecto está implementado con Ionic Framework versión 2, para instalar el entorno de desarrollo se deben seguir las siguientes instrucciones:

- Instalar nodejs versión 6 o superior desde: https://nodejs.org/
- Iniciar un terminal o símbolo del sistema
- Ejecutar: `npm install -g ionic cordova`

Este proceso instala la utilidad genérica nodejs junto que contiene el gestor de paquetes de software javascript npm. Mediante el gestor de paquetes se instala de forma global (-g) el paquete cordova (apache cordova) y el paquete ionic (ionic framework).

5.6.3. Sobre NodeJs

NodeJs es un entorno de ejecución (runtime) JavaScript. Una vez instalado es capaz de ejecutar aplicaciones Javascript. El lenguaje JavaScript se construyó incrustado en los navegadores Web, NodeJs no necesita un navegador web, pudiendo ejecutar aplicaciones servidor o cliente. Ionic 2 y Apache Cordova utilizan NodeJs para programar las herramientas de construcción y de gestión de paquetes (utilidad npm), el resultado final empaquetado que genera Ionic (aplicación de móvil) no necesita NodeJs para funcionar.

5.6.4. Instalación y actualización de los paquetes necesarios

Para instalar los paquetes adicionales del proyecto, nos situaremos en la carpeta y ejecutaremos:

```
npm update
```

5.6.5. Prueba de ejecución en el navegador

Nos debemos colocar en la carpeta contenedora del proyecto y ejecutar:

```
ionic serve
```

Este proceso construirá el proyecto y lanzará el navegador predeterminado ejecutando la aplicación. Una de las ventajas del uso de ionic serve es que detecta los cambios los archivos del proyecto, relanzando la aplicación cuando es necesario, sin embargo, es posible que algunos cambios no sean detectados, por ejemplo los cambios en las páginas de estilo css, que requieren refrescar manualmente la página.

5.6.6. Construcción en Android.

- Instalar el SDK de Android, puede instalarse Android Studio que instala el SDK. https://developer.android.com/studio/index.html?hl=es-419
- Instalar JAVA JDK (mínimo 8, recomendado 64bits), estableciendo la variable JAVA_HOME a la carpeta de instalación. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnes
- Ejecutar en el directorio de la aplicación:
- ionic platform add android (sólo es necesario una vez)
- ionic build android

5.6.7. Posibles problemas en la construcción

- Si se usa el java JDK de 32bits es posible que el proceso dé error de memoria, la construcción del proyecto por parte de apache cordova usa una gran cantidad de memoria RAM aunque el proyecto sea muy pequeño, se soluciona este problema instalando el JDK de 64 bits (se necesita un sistema de 64 bits).

5.7. Depurando la aplicación

5.7.1. Visual Studio Code

Visual Studio Code es un editor de código multiplataforma ligero y potente, diseñado por Microsoft.
https://code.visualstudio.com/
Tiene un soporte excelente del lenguaje TypeScript, incluyendo autocompletado, detección precoz de errores y una completa herramienta de depuración.
Visual Studio Code es OpenSource con la licencia MIT. Está programado internamente en TypeScript y C++.
https://github.com/Microsoft/vscode

5.7.2. Configuración de source maps

Para poder depurar correctamente las fuentes typescript hubo que configurar source maps, añadiendo al archivo packages.json la opción de configuración.

```
"config": { "ionic_source_map": "source-map" },
```

Esta configuración no estaba incluida por omisión, tampoco estaba correctamente documentada, es imprescindible tanto si depuramos desde el navegador como desde otro entorno, de otra forma no veríamos nuestro código, sino un código transpilado.

5.7.3. Archivo de ejecución Visual Studio Code

Se ha creado la carpeta .vscode y el archivo launch.json, con las opciones de ejecución del proyecto, este archivo se genera automáticamente con la tecla de ejecución (F5), una vez abierta la carpeta del proyecto desde el entorno Visual Studio Code.

5.7.4. Configuración de la extensión de depuración para Chrome

Se debe instalar la extensión de depuración para Chrome en Visual Studio Code. Pulsando Ctrl+P y ejecutando ext install debugger-for-chrome.

5.7.5. Configuración de Chrome para depuración

Para habilitar la depuración de código en Chrome se debe pasar el parámetro en la ejecución del navegador --remote-debugging-port=9222. Se puede crear un acceso directo que ejecute Chrome con este parámetro. Ejecutando Chrome con este parámetro y escribiendo la dirección: http://127.0.0.1:9222/json nos debe aparecer la configuración del depurador.

5.7.6. Ejecución del servidor web de Ionic

Ejecutaremos en la consola ionic serve –b

La opción –b indica que no se abra un navegador (no lo ejecutaría en modo depuración). Usaremos el navegador abierto anteriormente, con la dirección http://localhost:8100

5.7.7. Lanzar depuración desde Visual Studio Code

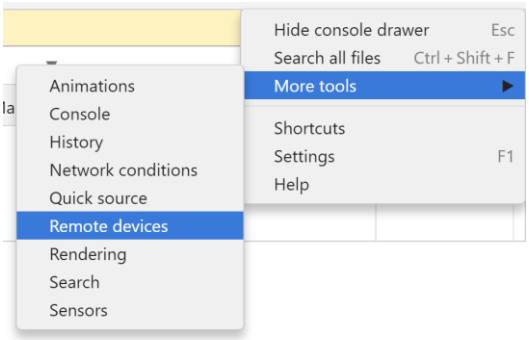
Una vez configurado, con la tecla F5, Visual Studio code se conectará al navegador Chrome, se podrán establecer puntos de interrupción en los archivos y ejecutar paso por paso, examinar variables…

5.7.8. Depuración en el dispositivo Android

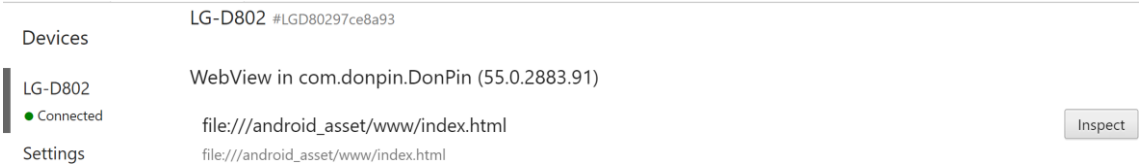
Para depurar en el dispositivo también podemos lanzar la consola de depuración de Chrome.
Primero lanzaremos la aplicación en el dispositivo, conectado por USB con el comando:

```
Ionic run android
```

Luego abriremos una instancia de Chrome, abriremos las herramientas de desarrollo (tecla F12) y seleccionamos el menú desplegable de la derecha, More tolos, Remote devices.



En la parte inferior nos aparecerá el dispositivo y las aplicaciones que se pueden depurar podremos pulsar “Inspect” para depurar la aplicación en curso.



En la consola podremos examinar el “local storage”, la consola…

5.8. Personalización de la aplicación

La aplicación es personalizable, para poder ser implantada en diferentes empresas, aunque no se ha automatizado el proceso de personalización.

- Modificar el archivo config.xml para publicar la aplicación con otro nombre, se deberán en este caso eliminar las plataformas y volverlas a crear, esto actualizará los archivos de construcción de cada plataforma: ionic platform remove android, ionic platfor remove ios, ionic platform add android…

- Deben reemplazarse los archivos splashscreen e icon de la carpeta resources, luego ejecutar ionic resources, para que genere los iconos y splashscreens de todas las plataformas.
- Las hojas de estilos principales (definen el tema de la aplicación) son app.scss y variables.scss.

5.9. Implementación de Notificaciones Push

5.9.1. Configuración inicial de la aplicación cliente

Ha sido necesario instalar el plugin de notificaciones de ionic ejecutando:

```
ionic plugin add phonegap-plugin-push --variable SENDER_ID=XXXXXXXXX
```

Se ha obtenido el Sender ID creando un proyecto de desarrollo en los servicios Firebase Cloud messaging de Google.

5.9.2. Generación de los mensajes

El ERP detecta los cambios en los pedidos (altas, asignación de camión…), insertando los datos en la tabla de mensajes.

5.9.3. Envío de mensajes

Se ha generado una aplicación que consulta la tabla de mensajes periódicamente comunicándose con el servicio Firebase Cloud Messaging (https://firebase.google.com/docs/cloud-messaging/?hl=es-419), este servicio permite el envío de mensajes tanto a dispositivos Android como IOS.

5.9.4. Recepción de mensajes

El dispositivo muestra un mensaje, permitiendo al usuario descartarlo o bien visualizar el pedido al que se refiere el mensaje.

5.10. Herramientas adicionales

5.10.1. IonicView

IonicView nos permite compartir la aplicación con otros usuarios sin necesidad de construir la aplicación para cada plataforma, el usuario debe instalarse la aplicación IonicView en su sistema (Android,IOS…), introduciendo las credenciales le aparecen las aplicaciones disponibles y puede ejecutarlas.
El desarrollador puede enviar una invitación por e-mail.
Una vez ejecutada la aplicación el usuario puede enviar sus comentarios al desarrollador.

5.10.2. IonicServe

Con la utilidad ionic serve pueden realizarse modificaciones en la aplicación y el proyecto se reconstruye automáticamente cuando se guardan los cambios permitiendo depurar en el navegador de forma rápida los cambios realizados (live reload).

5.11. Optimizaciones

5.11.1. Imágenes

La transmisión de imágenes se ha optimizado, en las listas se hace referencia a imágenes de calidad media, en el modo ficha (detalle de línea pedido) se hace referencia a imágenes de alta calidad.
La optimización de cache de carga se realiza mediante el sistema estándar de cache de navegadores de internet HTTP ETag: (https://en.wikipedia.org/wiki/HTTP_ETag).
El web service genera un ETag único por cada imagen, de forma que el navegador puede identificar cada imagen en la caché de imágenes. El WebService se encarga de enviar un ETag distinto si la imagen ha sido modificada.
Sin estas optimizaciones el rendimiento de la visualización de imágenes era muy pobre.

5.11.2. Listas con elementos virtuales

Para mejorar el rendimiento del desplazamiento de las listas (lista de pedidos) se ha intentado utilizar una característica llamada lista virtual, que proporciona Ionic 2, de modo que las celdas de la lista se reutilizan, reduciendo drásticamente el tiempo de procesado del DOM de la página, esto es parecido a como están implementadas listas nativas de las distintas plataformas. Sin embargo, esta característica tiene algunos fallos cuando se combina con las listas de desplazamiento infinito, el código está preparado y posiblemente se active en un futuro esta característica, es probable que este problema esté corregido en el lanzamiento final de Ionic 2.

5.11.3. Listas con desplazamiento infinito

En la lista de pedidos, se cargan solamente una parte de los elementos de la lista, y conforme el usuario avanza se van cargando los siguientes, de forma que la velocidad de presentación mejora mucho.

5.12. Problemas durante el desarrollo

5.12.1. Documentación imprecisa

Ionic junto con apache-cordova es una plataforma con documentación más bien escasa, cuando hay algo que no funciona, es difícil encontrar una solución, por ejemplo, el proyecto en blanco que crea ionic viene preparado con un splash screen y un icono personalizado de la aplicación, de forma que parece que sustituyendo estos iconos, se suficiente, sin embargo no es así, hay que instalar el plugin de splash screen de cordova: `cordova plugin add cordova-plugin-splashscreen`. La documentación no indica este paso (http://ionicframework.com/docs/cli/icon-splashscreen.html)
Lo mismo pasa con las funciones del teclado (mostrar/ocultar), aparentemente debería funcionar, pero no es así, en este caso no hay que instalar un plugin de cordova, sino uno específico de ionic: `cordova plugin add ionic-plugin-keyboard`. El problema es que en la documentación se indica cómo mostrar el teclado, pero no se indica que es necesario un plugin específico.

La documentación es aún más confusa cuando el comportamiento es distinto en el navegador y en el dispositivo, por ejemplo: en las peticiones de servicios http, el teclado en pantalla, el input password…

5.12.2. Combinación de funcionalidades delicada

Ionic 2 es maniático en cuanto a la combinación de diferentes funcionalidades, por ejemplo: la etiqueta <ion-content padding> no funciona junto con <infinite-scroll>, debiendo utilizar <ion-content> sin padding, esto no está documentado y puedes ser bastante difícil encontrar por qué no funciona alguna característica de ionic.

5.12.3. Documentación dispersa

La documentación está distribuida en demasiados ámbitos-tecnologías: Angular2, Ionic 2, Apache-Cordova, HTML, CSS, SASS, TypeScript (transpilación javascript) , Nodejs (npm…), se requiere un mayor conocimiento de todas las tecnologías por parte del desarrollador, para determinar si el problema está en un ámbito o en otro.

5.12.4. Fragilidad del sistema de ejecución

La aplicación es frágil en cuanto a errores de programación y diseño html, en el sentido que, un error en un archivo html (comilla de más o etiqueta no cerrada), nos da un error general (*template parse error* de Angular 2) que provoca que la aplicación no funcione en absoluto, arranca pero se ve una página en blanco. No siempre se indica el archivo y línea que provoca el error en la consola del navegador, debido a las múltiples transformaciones (platillas y transpilaciones) que se realizan en la construcción, por lo que hay que recordar qué archivos se tocan para detectar eficientemente el origen del problema. El uso de un control de versiones ha ayudado en este sentido. En mi opinión este problema tiene difícil solución porque los navegadores web tienen la premisa de ignorar los errores y mostrar lo que se pueda, pero Angular necesita que el esquema html esté perfectamente construido.

5.12.5. Cross origin resource sharing

Con la utilidad ionic serve, la llamada a servicios http suele dar un error de cross domain origin, la solución pasa por configurar un proxy o bien modificar el servicio web para que permita consultas Cross Domain Origin. https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

5.12.6. Acceso a servicios desde el dispositivo

Ejecutando la aplicación en el dispositivo tampoco funcionan por omisión los servicios web (error 404 not found), de nuevo la documentación es escasa, las soluciones encontradas son dispares, pero una de ellas funcionó correctamente:

- Ejecutar cordova plugin add cordova-plugin-whitelist
- No fue necesario modificar el archivo config.xml con el dato <allow-navigation href="http://*/*" />

5.12.7. Errores en la construcción para cada plataforma

Durante la construcción para cada plataforma (ionic build android por ejemplo) ser realizan más comprobaciones que en la simulación web (ionic serve), por lo que de vez en cuando es conveniente construir para una plataforma, es posible que nos aparezca un error del tipo la función xxx no está implementada, si en un archivo html se hace referencia a una función no implementada en código, algunos errores no son detectados por la construcción con ionic serve.

5.13. Características pendientes de implementar

Las siguientes características han quedado pendientes de implementar:

- El usuario debería poder cambiar su contraseña.
- Completar traducciones para idioma chino.

5.14. Seguimiento de la planificación durante la fase de implementación

Se han finalizado las tareas más importantes del Sprint 2, se estima que el Sprint 3 no presentará dificultad, en el Sprint 2 se ha realizado el mayor grueso de trabajo en cuanto a desarrollo de código, el Sprint 2 era el que tenía más riesgo en cuanto a toparse con impedimentos que retrasaran el proyecto.

Se ha actualizado el Product Backlog y el Sprint Backlog con la nueva información de planificación.

Se ha actualizado el Plan de pruebas para tener en cuenta las nuevas funcionalidades.

5.15. Seguimiento de la planificación en la fase de planificación final

Se ha completado el Sprint 3, añadiendo la funcionalidad de la página principal configurable por la empresa, finalizando la memoria y creando una video presentación del proyecto.

6.Guía de desarrollo de nuevas funcionalidades

6.1. Estructura del proyecto

El proyecto se estructura siguiendo el modelo planteado por Ionic 2.
La carpeta src contiene el código fuente, dentro encontramos la carpeta app, con la página de inicio de la aplicación, y la carpeta pages con las diferentes páginas.

6.2. Creación de una nueva página

Usaremos el cli de ionic para generar una nueva página, esto no es estrictamente necesario, pero nos facilita el trabajo, esta utilidad crea tres archivos, html y css para configurar la vista y un archivo typscript .ts, el controlador.
Nos situaremos en la carpeta del proyecto y ejecutaremos:

```
Ionic g page barcodeScan
```

Será necesario incluir esta nueva página en la lista de módulos de angular en el archivo app.module.ts, en la cabecera añadiremos la referencia al archivo:

```
import {BarcodeScanPage} from '../pages/barcode-scan/barcode-scan';
```

En el array de declarations y entryComponents añadiremos el nombre de la página generado, en este caso: BarcodeScanPage

Para añadir un servicio en vez de una página el proceso es parecido pero se usa provider en vez de page en el CLI, y se añade el servicio en el array de providers.

6.3. Mostrando la nueva página
Hay varias posibilidades para mostrar la nueva página, se puede realizar añadiéndola a la pila de pantallas mediante navCtrl.Push(…) o bien añadiendo un enlace en el menú principal, contemplaré este último caso.
En el archivo app.component.ts añadimos la referencia a la página en la cabecera:

```
import {BarcodeScanPage} from '../pages/barcode-scan/barcode-scan';
```

Luego añadimos una referencia la página al array del menú principal:

```
      this.pages.push({       title:       'Producto',       component:
BarcodeScanPage, action:""})
```

La propiedad action se usa para opciones que no abren ninguna página.

6.4. Accediendo a los datos del servicio

Para acceder a los datos del servicio se debe importar en la cabecera:
import { OrderService } from '../providers/order-service';

Debe añadirse la propiedad providers con el servicio al objeto @Component:

```
@Component({
  selector: 'page-barcode-scan',
  templateUrl: 'barcode-scan.html',
```

```
  providers: [OrderService]
})
```

Además el servicio debe inyectarse en el constructor:

```
constructor(…public orderService:OrderService)
```

La llamada al servicio se realiza de forma asíncrona con una promesa javascript.

```
   this.orderService.GetProduct(this,product_id,barcode)
    .then(data => {
        let productResult = data["Message"];
        if (productResult == "OK")
        {
```

Esta función de servicio en concreto nos devuelve un mensaje de error o bien “OK”, además de los datos, buscando un producto por número interno o por el código de barras.

7. Comentarios del desarrollo

7.1. Planificación

La planificación se ha realizado sin incidencias notables. El punto más importante ha sido la elección de la tecnología de desarrollo, el uso de tecnología nativa para cada dispositivo hubiera incrementado notablemente el tiempo de desarrollo, limitando el proyecto a sólo una plataforma.

7.2. Diseño

Se dedicó mucho tiempo en la fase de diseño a la programación del BackEnd, WebService, el diseño resultó demasiado básico, estos defectos intentarán corregirse en la implementación final.

7.3. Implementación

La implementación se realizó con normalidad, aunque no se implementaron todas las características, aunque sí las más importantes, la tecnología de desarrollo presenta algunos problemas que son descritos en la memoria. Se han implementado algunas nuevas funcionalidades para dar valor añadido a la aplicación, estas nuevas funcionalidades son una muestra del abanico de posibilidades que puede ofrecer en un futuro la aplicación.

7.4. Implementación final

Se completaron todas las características planificadas, a excepción del cambio de la clave del usuario, la traducción al idioma Chino está pendiente ya que no se han proporcionado todos los textos traducidos.

8. Conclusiones

8.1. Planificación

El cálculo de las horas dedicadas a cada tarea es bastante difícil, se ha tenido en cuenta la formación y experiencia del equipo de desarrollo (yo mismo) para este cálculo, aún sería más difícil si no se conocieran perfectamente las capacidades productivas del equipo de trabajo. Dentro de las horas de trabajo se tiene en cuenta la posibilidad de que surjan impedimentos que necesiten más tiempo de lo habitual para su solución.

El trabajo de planificación y documentación ocupa una buena parte del tiempo del proyecto: más del 30%.

8.2. Diseño

El tiempo invertido en el diseño del producto no ha sido suficiente, las ideas de mejora de producto que surgieron más tarde debieron ser consideradas en la fase de diseño. Este hecho ha provocado que el prototipo se haya desfasado rápidamente del aspecto real del producto.

8.3. Tecnologías

Se han requerido conocimientos en diversas tecnologías para el desarrollo de la aplicación, lenguajes HTML/CSS, TypeScript, JavaScript, C# (Asp.Net), formatos XML, Json, protocolo HTPP…

8.4. Resultado final

El producto final conseguido es fácilmente ampliable, el soporte de múltiples plataformas no es complicado debido al uso de un framework multiplataforma.

9. Glosario

- **Scrum**: nombre con el que se denomina a los marcos de desarrollo ágiles caracterizados por adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto, basando la calidad del resultado en el conocimiento tácito de las personas en equipos auto organizados, solapando las diferentes fases del desarrollo.
- **Extreme Programming** (Programación Extrema): Filosofía de desarrollo de software basada en las mejores prácticas: simplicidad, comunicación, retroalimentación (feedback), auto-documentación del código...

10. Bibliografía

1. **Métodos Ágiles y Scrum**. Editor Anaya Multimedia 2012. Autores: Alonso Álvarez García, Rafael de las Heras del Dedo, Carmen Lasa Gómez. ISBN-10: 8441531048
2. **Wikipedia**.Octubre.2016 https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)
3. **Extreme Programming Explained: Embrace Change: Embracing Change**. Editor Addison Wesley 2005. Autor: Kent Beck.
4. **Documentación de Ionic 2**. https://ionicframework.com/docs/
5. **Documentación de Apache Cordova.** https://cordova.apache.org/docs/en/latest/

11. Anexos

11.1. Product Backlog (ficha del producto)

Ver archivo **ProductBacklock.xlsx**.

11.2. Plan de pruebas

Introducción
El plan de pruebas tiene los siguientes objetivos:

1. Certificar que la aplicación cumple los requisitos exigidos por el cliente.
2. Encontrar posibles defectos de funcionamiento o presentación.
3. Asegurar que el producto está libre de errores y listo para su publicación.
4. Proporcionar un producto de software sin fallos minimizando los riesgos.

Tipos de pruebas
El plan de pruebas consta de dos grupos principales de pruebas.

Black Box Testing: Pruebas de comportamiento a nivel de usuario y cumplimiento de requerimientos funcionales.
Se realizan instalando la aplicación e interactuando con ella probando cada una de las funcionalidades requeridas. Las pruebas nos darán un resultado visual y funcional y también podremos valorar la experiencia de usuario.

White Box Testing: Pruebas programadas en dentro del propio código fuente del desarrollo.
Estas pruebas se ejecutarán de forma automática mediante el uso de ingeniería basada en tests, programación de clases y funciones de test (unit testing), usando métodos de test driven development.

Tabla de pruebas
Se proporciona un archivo excel (TablaPruebas.xlsx) con una estimación de las pruebas a realizar, dados los requisitos iniciales. El plan de pruebas no es estático, durante el desarrollo se agregarán nuevos elementos de prueba, se desglosará una prueba en varias pruebas etc.
Siempre que sea posible se automatizarán las pruebas.

Errores en las pruebas
Antes del lanzamiento de un prototipo funcional entregable, se realizarán las pruebas correspondientes de la parte ya implementada (épicas finalizadas), se realizarán las pruebas de las nuevas partes implementadas y las ya implementadas. Cuando se encuentre un fallo se introducirá el fallo en la base de datos de seguimiento de fallos (en este caso será una hoja excel). El Product Owner podrá decidir la prioridad para la solución de cada fallo.

Inicialmente se entrarán los fallos en la plantilla de pruebas (TablaPruebas.xlsx), más adelante podría configurarse un software específico de gestión de fallos tipo BugZilla.

El fallo constará de la siguiente información:
1. ¿Dónde? Pantalla-Formulario, si se encuentra específicamente en un formulario de la aplicación.
2. ¿Cómo? Método de reproducción detallado.
3. Consecuencia. Qué efectos negativos tiene sobre el funcionamiento o presentación.
4. Valoración inicial de severidad (Crítico, Grave,Mínimo)
5. Prioridad de resolución.
6. Fecha de detección
7. Estimación de tiempo de solución, opcional.
8. Fecha de solución.

Ver archivo **TablaPruebas.xlsx**

11.3. Script SQL de estructuras de datos

Se incluye el archivo **script.txt**, que contiene las tablas creadas para la gestión de permisos y mensajería en la implantación del Backend.

11.4. Aplicación paquete android instalable

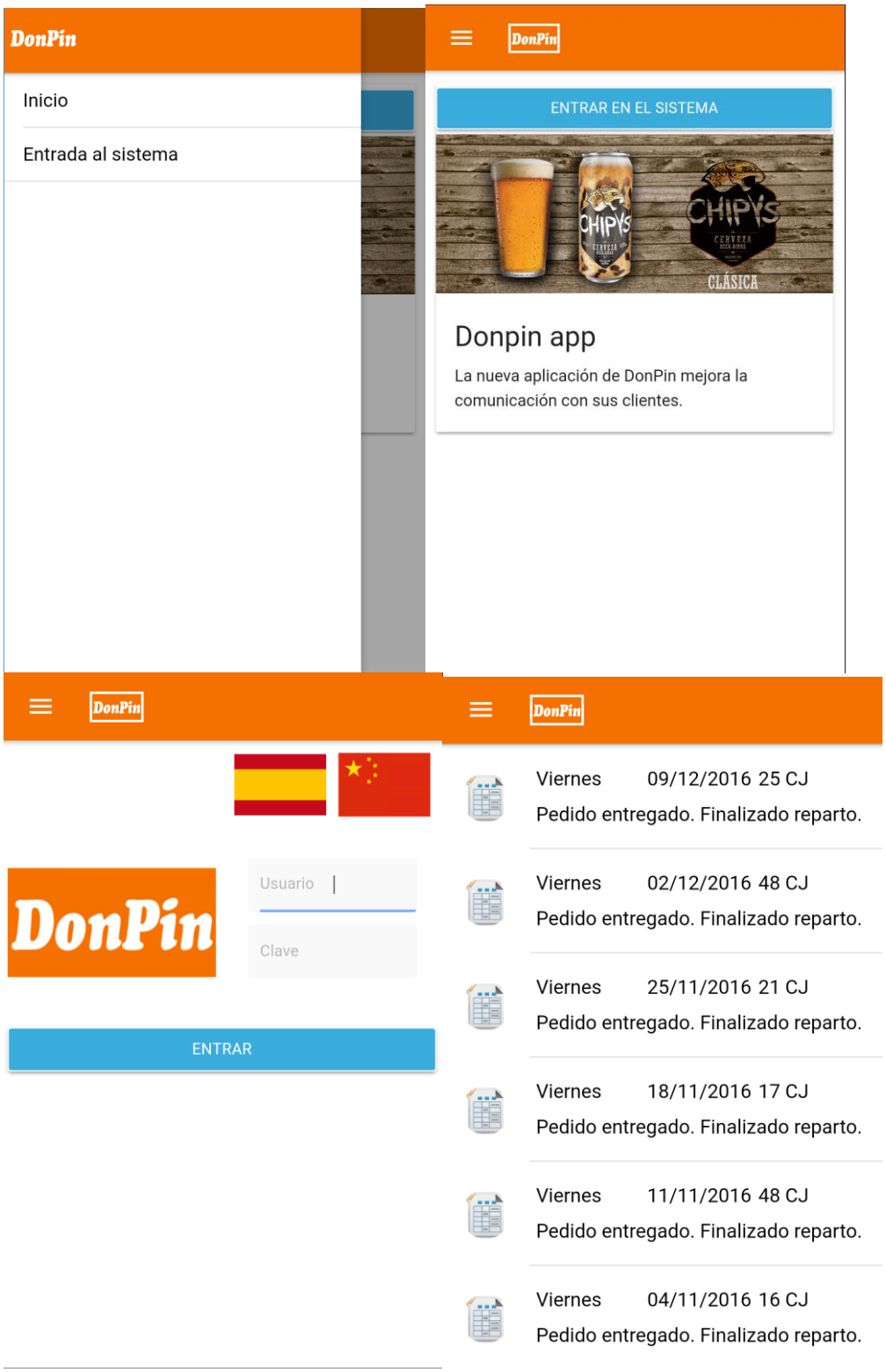
Se proporciona el **archivo proyecto.apk** instalable en una máquina con sistema Android, la versión de Android mínima soportada es la 4.4 KitKat, existen métodos para soportar versiones inferiores a costa de incrementar bastante el tamaño del paquete: https://crosswalk-project.org/documentation/getting_started.html
Será necesario activar la instalación de aplicaciones de orígenes desconocidos en la configuración de seguridad del dispositivo ya que el paquete no está firmado.

11.5. Video presentación del proyecto incluyendo demostración

Puede visualizarse la video presentación del proyecto en el siguiente enlace:
https://youtu.be/AkCtyaIdB5Y

11.6. Capturas de pantalla

Se presentan capturas de las pantallas de la aplicación.


DonPin
Inicio
Entrada al sistema
DonPin
ENTRAR EN EL SISTEMA
CHIPYS
CLÁSICA
Donpin app
La nueva aplicación de DonPin mejora la comunicación con sus clientes.
DonPin
DonPin
Usuario
Clave
ENTRAR
DonPin
Viernes 09/12/2016 25 CJ
Pedido entregado. Finalizado reparto.
Viernes 02/12/2016 48 CJ
Pedido entregado. Finalizado reparto.
Viernes 25/11/2016 21 CJ
Pedido entregado. Finalizado reparto.
Viernes 18/11/2016 17 CJ
Pedido entregado. Finalizado reparto.
Viernes 11/11/2016 48 CJ
Pedido entregado. Finalizado reparto.
Viernes 04/11/2016 16 CJ
Pedido entregado. Finalizado reparto.


DonPin
Pedido 1625364
09/12/2016 25 CJ 264.62 €
Pedido entregado. Finalizado reparto.
10 CJ 95.83 €
MAHOU 5 ESTRELLA LATA 33CL
3 CJ 43.91 €
MAHOU 5 ESTRELLA LATA 50CL
1 CJ 10.33 €
STARBUCKS FRAPPUCINO CAFE
1 CJ 9.20 €
KINDER DELICE CACAO 42GR
1 CJ 5.89 €


DonPin
Línea pedido 1625364
MAHOU 5 ESTRELLA LATA 33CL
Mahou
Cantidad
240 UN
Precio unidad
0.40 €
Unidades Caja
24