PROYECTO FIN DE CARRERA

Aplicación para terminales Android orientada a la gestión remota, reparto, y venta de productos - aguapp

AUTOR: BRAIS VILAVEDRA FERNÁNDEZ
TUTOR: ABRAHAM RODRÍGUEZ RODRÍGUEZ
COTUTOR: KUNAL MAHTANI DARYANANI

EDIFICIO DE INFORMÁTICA Y MATEMÁTICAS
JULIO DE 2017

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

A mis padres, por hacer todo esto posible

Contenido

Introducción

A lo largo de este documento se presenta el desarrollo del proyecto fin de carrera **Aplicación Android para gestión remota y reparto de productos - aguapp**. Esta aplicación se ha desarrollado con objetivo principal de agilizar y facilitar el trabajo tanto de los repartidores, como de los administrativos de cualquier empresa de reparto y venta de productos a domicilio.

Para la presentación de la aplicación se ha tomado como ejemplo una empresa de reparto de aguas modelo, principalmente alimentada esta idea por el gran número de empresas similares que existen en la Comunidad Autónoma Canaria. Se parte del escenario en el que la empresa tiene implementado un software de gestión o administración tipo ERP.

Los sistemas de planificación de recursos empresariales ('ERP', por sus siglas en inglés, **E**nterprise **R**esource **P**lanning) son los sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios, pero carecen de interfasados con dispositivos externos, o de tenerlos, en muchos casos no cumplen las necesidades específicas de cada tipo de negocio.

En el caso de un ERP como puede ser Microsoft Dynamics NAV, que cuenta con un cliente web, algunos menús, aún simplificados pueden ser confusos para un tipo de usuario final que no tenga conocimientos administrativos o financieros. Por ejemplo, un repartidor de agua no tiene por qué saber qué tipo de impuesto debe aplicar a un determinado cliente, ni a que cuentas contables deben dirigirse los movimientos de valor. O puestos a usar más dispositivos externos, como pueden ser impresoras portátiles, el ERP no cuenta con interfaces concretas para cada tipo de impresora, como pueden ser las impresoras Zebra que cuentan con su propio lenguaje de etiquetado para impresión: ZPL (Zebra Programing Language)

De ahí nace la necesidad de crear esta aplicación, con el fin de simplificar el desempeño diario de los repartidores de aquellas empresas que opten por digitalizar sus procesos diarios.

Contexto

Para el desarrollo de esta aplicación se expusieron en primer momento los objetivos a completar, simulando una toma de requisitos por parte de un cliente, pilar básico de cualquier modelo de desarrollo de software.

Estos requisitos se tradujeron en la idea de presentar como solución una aplicación Android para la gestión remota y la venta de productos online, desde la que se puedan consultar pedidos creados en el sistema para albaranarlos (enviarlos) o facturarlos (cobrarlos) en el momento de la venta, o realizar ventas sobre la marcha, con soporte para impresión de tickets.

Objetivos

Una vez presentado como modelo de empresa el de una empresa de reparto de aguas que cuente con un sistema de gestión informatizado, se presenta una carencia evidente en el día a día, y es que los repartidores de muchas de estas empresas de reparto siguen utilizando como método de facturación, un bloc de facturas y albaranes preimpresos, en los cuales se rellena a mano la cantidad de cada producto, se firman, y sellan. En la **ilustración 1** se puede ver un ejemplo de una factura preimpresa.

Estos blocs no solo se usan de cara al cliente, sino que incluso de forma interna se utilizan para justificar las cargas y descargas del camión. Esto provoca que se desaproveche mucho papel, ya que por ejemplo para ventas de pocos artículos como pueden ser unas pocas garrafas o un pack de botellas a domicilio, la cantidad de papel que queda sin rellenar es notable. Al final del día, los repartidores entregan las ventas del día en estos blocs para que posteriormente desde administración se inserten a mano en el sistema.

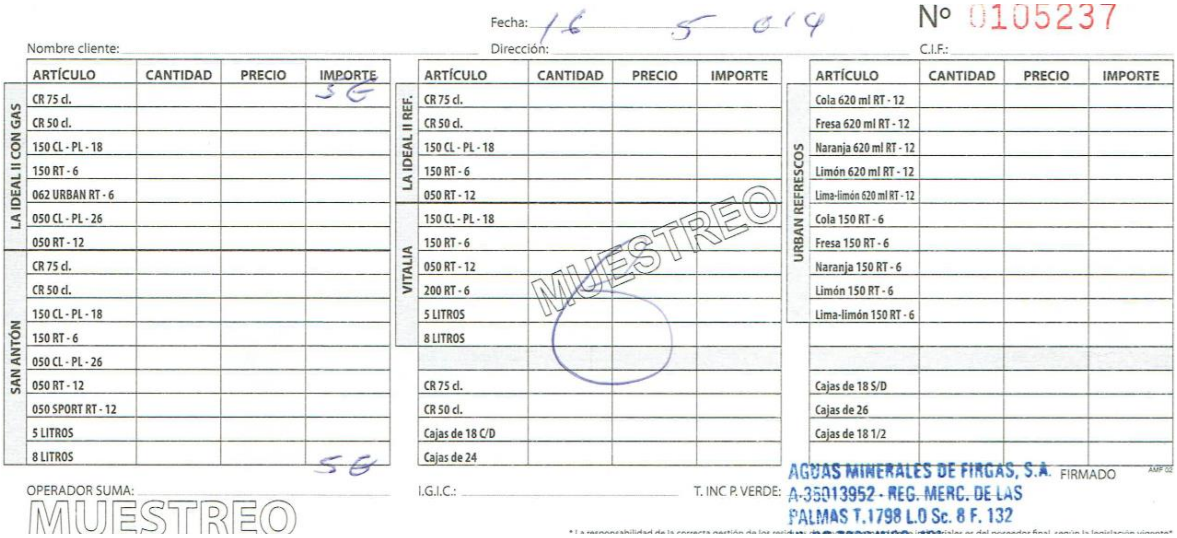

Ilustración 1 - Factura preimpresa de muestreo (descuento 100%)

Esto se puede desglosar en varios objetivos:

- La necesidad de desarrollar una solución informática que agilice la venta por parte de los repartidores, pudiendo recuperar pedidos ya creados en el sistema como parte de la preventa, o realizar ventas sobre la macha.

- La meta de reducir al máximo el consumo de papel, por un lado, por la necesidad patente general de cuidar el medio ambiente, y por otro, el intento de ahorrar recursos a la empresa. Si en lugar de blocs preimpresos se utilizasen tickets impresos sobre la marcha, la cantidad de papel se reduciría considerablemente.

- Agilizar el trabajo administrativo a la hora de introducir en el sistema las ventas al final del día de cada repartidor. Los datos de disponibilidad de productos se necesitan para planificar el día siguiente, por lo que es crítico tenerlos disponibles con la mayor brevedad posible. Esto provoca que muchos empleados administrativos tengan que trabajar, no solo horas extra, sino en horas intempestivas como la noche o la madrugada. Esto puede traducirse en más gastos para la empresa y un peor rendimiento y más estrés para los empleados, todos estos hechos no deseados para una empresa.

Solución propuesta

Con la problemática analizada de la empresa modelo, la solución que se propone debe suplir una serie de requisitos:

Contar con un sistema de usuarios, con la necesidad de que cada usuario se autentifique al inicio de cada sesión. El servicio de creación de usuarios estaría disponible desde el sistema informático de la empresa. Dicho servicio debe dar la posibilidad de diferenciar varios tipos de usuario, como pueden ser transportistas, administradores, o comerciales.

Dar soporte a la app de realizar cargas y descargas del camión, al comienzo y final del día. Extendiendo este requisito, se propone incorporar la funcionalidad de realizar transferencias entre camiones. Dar la posibilidad de cargar en la app pedidos creados en el sistema, para albararnarlos o facturarlos. Asimismo, debe dar la posibilidad de realizar ventas en el momento, creando pedios remotamente en el sistema. Para pedidos que solo hayan sido albaranados, debe dar la posibilidad de realizar su facturación y pago.

Se propone también que la aplicación sea usable por los comerciales de la empresa, pudiendo dedicarse a tareas de preventa, tales como la creación de pedidos para que los repartidores durante su jornada puedan ir cargando pedidos asignados a ellos y procesarlos al momento.

Acompañar la app de una impresora de tickets portátil que permita imprimir tickets para todas funciones de la app que así lo requieran. Se va a utilizar como modelo la impresora Zebra iMZ320, que se comunicaría por Bluetooth con la app.

Todos los procesos deben comunicarse directamente con el sistema, usando la información disponible en él y actualizándola, para contar en tiempo real en el sistema con los datos de ventas y disponibilidad.

Metodología

Como modelo de desarrollo se ha elegido el enfoque tradicional del modelo en casada. Principalmente motivado porque su visión es muy simple: el desarrollo de software se debe realizar siguiendo una secuencia de fases. Cada etapa tiene un conjunto de metas bien definidas y las actividades dentro de cada una contribuyen a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la misma. Su ventaja principal es su sencillez, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software El arquetipo del ciclo de vida abarca las siguientes actividades:

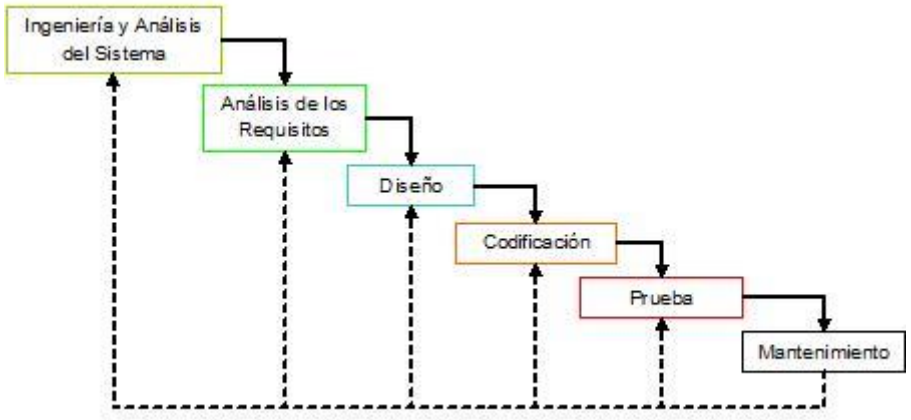

Ilustración 2 - Modelo en Cascada

En la **ilustración 2** puede verse una ilustración del modelo, que de describe a continuación.

- **Ingeniería y Análisis del Sistema**
  El trabajo comienza estableciendo los requisitos de todos los elementos del sistema.
- **Análisis de los requisitos del software**
  Se debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas para recopilar los requisitos.
- **Diseño**
  Se enfoca en cuatro atributos distintos del programa; la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz.
- **Codificación**
  El diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

- **Prueba**

  La prueba se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

- **Mantenimiento**

  El software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que se haya encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos) o a que el cliente requiera ampliaciones funcionales o del rendimiento.

Análisis

Una vez presentadas a modo de introducción las necesidades del desarrollo, en este apartado se pasa a analizar la aplicación desde un punto de vista técnico, junto con un estudio del estado del arte de soluciones similares en el mercado. También se repasará el modelo de datos de la solución junto con la descripción de su modelo de desarrollo.

Estudio de aplicaciones similares

Después de realizar una búsqueda de aplicaciones o soluciones similares a la propuesta en este documento, se han analizado sus prestaciones para plasmarlas en este apartado.

Se incluye una tabla comparativa, en la que se valoran los siguientes aspectos:

- **Diferenciado por usuarios**
  Que la app cuente con sistema de gestión de usuarios, con autenticación y personalizaciones.
- **Recuperación de pedidos creados en el sistema**
  Que haya posibilidad de cargar en la app pedidos creados desde el sistema de gestión, para procesarlos.
- **Gestión de carga del camión**
  Que la app de la opción de asignar la carga al camión, representando un movimiento de productos.
- **Posibilidad de asignar repartidores a camiones (superusuario)**
- **Modificación de pedidos enviados/facturados**
  Una vez registrados, que haya posibilidad de realizar cambios sobre el pedido.
- **Integración con sistema de gestión**
  Que haya posibilidad de conexión con un sistema externo más allá de una base de datos local
- **Seguimiento por geolocalización.**

A continuación, se presentan las más representativas

Anveo Mobile

AnveoNav, Complemento para ERP Microsoft Dynamics NAV, disponible en Google Play

Anveo Mobile es un add-on para el ERP Microsoft Dynamics NAV. Este add-on requiere a su vez de otro, Anveo Client Suite, que aporta una interfaz al usuario para modificación y diseño de la aplicación móvil. No solo es una herramienta de reparto y cobro, sino que añade otros escenarios, como una app orientada a Servicio técnico con listas to-do, tiempos de trabajo, costes, y material requerido, y también sirve como CRM[1] para Microsoft Dynamics NAV sin la necesidad de usar Microsoft Dynamics CRM, lo que de cara a costes de licenciamiento puede ser atractivo para muchas empresas. La versión disponible en Google Play es gratuita, pero es una versión de demo que funciona con datos de prueba, requiere licenciamiento y compra del producto completo para su uso.

Capturas

Es una aplicación de uso bastante sencillo, e incluso la versión final de aguapp se asemeja a esta aplicación. En las **ilustraciones 3 y 4** se puede ver la distribución de su menú principal. **La ilustración 5** corresponde con la lista de pedidos "My Orders"

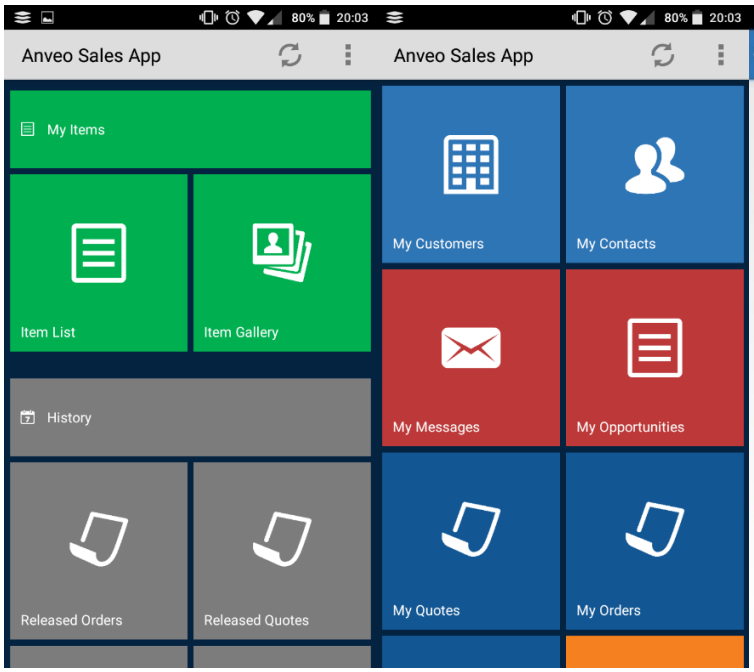

Ilustración 3 - Menú principal

Ilustración 4 - Menú principal 2

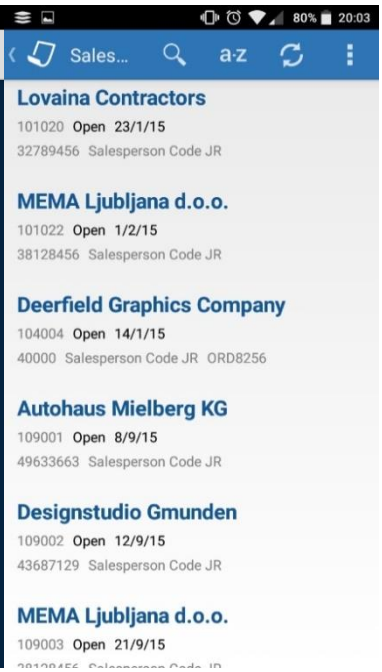

Ilustración 5 - Lista de pedidos

[1] CRM proviene de la sigla del término en inglés **C**ustomer **R**elationship **M**anagement, son sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing

Track-POD Proof of Delivery

Track-POD, disponible en Google Play

Track-POD Proof of Delivery permite que los repartidores reciban pedidos con la ruta y provee de un sistema de notificaciones en tiempo real. Se compone de la app móvil y de un backend en su propio cloud. Informa de la carga que se componme el pedido o describe el trabajo a realizar y provee de un sistema de posicionamiento con navegación a la ruta que acompaña el pedido. Permite seguimiento en tiempo real de la posición de los repartos y llamada directa al cliente desde la app. No cuenta con posibilidad de integración con sistemas externos. Es necesario registro con cuenta de pago en su sistema, con licenciamiento por cada repartidor.

Capturas

La filosofía de esta aplicación difiere de la planteada en aguapp de la forma en que está más orientada al posicionamiento, y en vez de mostrar los pedidos por separado, la muestra dentro de una ruta (**ilustración 6** muestra los puntos de ruta, la **ilustración 7** las rutas). En la **ilustración 8** se muestra un resumen del contenido del pedido y las opciones de cierre del mismo.

Ilustración 6 - Menú principal

Ilustración 7 - Opciones de la ruta

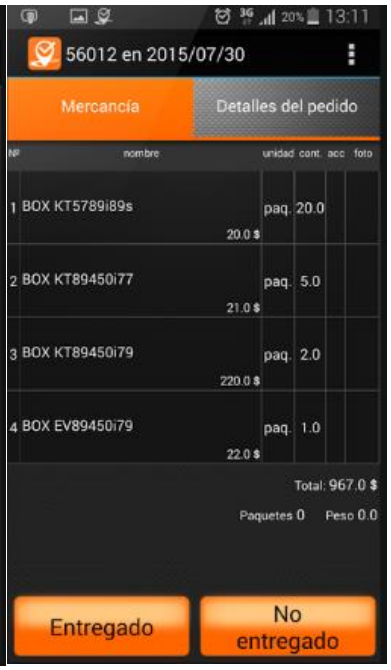

Ilustración 8 - Detalle de mercancía

Reparto & Cobranzza

StartApp Argentina, disponible en Google Play

Reparto & Cobranza es una solución móvil para que permite administrar la venta, cobro y distribución de productos. Entre otras funcionalidades, la aplicación permite los repartos asignados a repartidores, gestión de ventas por reparto, registro de pagos, devoluciones, envases prestados y cobros. La versión gratuita de Google Play es una versión de demostración, para uso continuado es necesario el contacto con la empresa para licenciamiento de su app de móvil y sistema informático, o para la integración con sistemas externos.

Capturas

Similar a Track-POD en términos de interfaz, pero carente de las opciones de posicionamiento. En la **ilustración 9** se muestra una lista de los pedidos dentro de la ruta asignada, siendo las opciones disponibles sobre el pedido las que aparecen en la **ilustración 10**.

Permite llevar un control sobre el dinero que se maneje como se puede ver en la **ilustración 11**, de cara a llevar una gestión como la que se lleva a cabo en los TPV (Terminal Punto de Venta) con apertura y cierre de caja.

Ilustración 9 - Menú principal

Ilustración 10 - Acciones sobre pedido

Ilustración 11 - Entrega

Tabla comparativa

A continuación en la **ilustración 12** se realiza una comparación entre las aplicaciones descritas en el apartado anterior junto con aguapp. Se han elegido las características que se consideran como objetivos generales, aparte de las representativas de las aplicaciones por separado.

	aguapp	Anveo Mobile	Track-POD	Reparto & Cobranza
Diferenciado por usuarios	✓	✓	✓	✓
Recuperación de pedidos creados en el sistema	✓	✓	✗	✓
Gestión de carga del camión	✓	✗	✓	✗
Posibilidad de asignar repartidores a camiones (superusuario)	✓	✗	✗	✓
Modificación de pedidos enviados/facturados	✓	✓	✗	✓
Integración con sistema de gestión	✓	✓	✗	✓
Seguimiento por geolocalización	✗	✗	✓	✗

Ilustración 12 – Tabla comparativa de apps

Requisitos

Usuarios potenciales

Como los usuarios potenciales de esta app serían los empleados. Dentro de los empleados de este tipo de empresas, el rango de edad varía entre 20 medios y puede abarcar hasta empelados cerca de la jubilación, aspecto muy a tener en cuenta a la hora del desarrollo usable de la aplicación.

Para entender la siguiente tabla, primero hay que hacer una visión global de la organización de la empresa. El primer rasgo diferenciable es el rol del empleado. Las funciones que ofrezca la app deben variar en función de las necesidades del empleado, ya que cada empleado o grupo de empleados tiene sus tareas, que no se solapan con los otros. Por lo que, por ejemplo, un repartidor no necesita tener acceso a la asignación de repartidores por camión, ya que no es una de sus tareas en la empresa.

- **Repartidor**: Desempeña las funciones de conductor o ayudante, y sus tareas se limitan a la carga, reparto y venta de productos. Visita a una serie de clientes fijos cada día de reparto, a los que consulta qué selección de productos quiere. Los nuevos clientes llaman para que se les incluya en la lista de visitas, reciben visitas de los comerciales, o se lo piden a los repartidores al verlos. Puede entregar pedidos ya recibidos en administración.
- **Administrador**: Toma las decisiones administrativas y financieras de la empresa. Hace las veces la función de comercial, y suele consultar la información de ventas y stock a menudo.
- **Comercial**: Se encarga de realizar llamadas y visitar centros de negocio en busca de nuevos clientes, aparte de recibir pedidos de clientes habituales y pasarlos al sistema.
- **Gerente**: Encargado de los recursos de la empresa. Se encarga de panificar las compras de productos en función del stock y planificar las cargas de los camiones cada mañana para suplir los pedidos. También se encarga de realizar la asignación de repartidores y ayudantes por camión.

A continuación, en la **Ilustración 13** se muestra una tabla de *dummies*, cuyo objetivo es identificar los perfiles y comportamientos de usuarios potenciales para adaptar el diseño y desarrollo acorde a sus necesidades.

Se plantea el caso en el que los administradores y gerentes de la empresa buscan agilizar las tareas diarias de la empresa, algo critico en una empresa de distribución, y en que sus repartidores están dispuestos a la innovación, con un caso concreto, un usuario de 52 años que no se le da bien el uso de móviles táctiles.

	Rol	Edad	Tipo	Motivación
Paco	Repartidor	36	Usuario normal	Encuentra engorroso el uso de blocs
John Doe	Administrador	48	Superusuario	Mejorar la comunicación con los empleados
Daniel	Comercial	29	Superusuario	Tiene que pasar tiempo extra pasando los pedidos al sistema cuando acaba la jornada
Alexis	Repartidor	34	Usuario normal	Ha visto a los repartidores de Correos y le pareció buen sistema
Maria	Repartidor	24	Usuario normal	Le gusta la idea de trabajar con el móvil
Alberto	Gerente	56	Superusuario	Mejorar el rendimiento de la empresa
David	Repartidor	52	Usuario normal	No se le dan bien los móviles, usa la app por imposición de empresa

Ilustración 13 – Tabla de 'dummies'

Requisitos de usuario

Para plasmar de una forma más visual los requerimientos de la app, en la **Ilustración 14** se incluyen diagramas de uso de la app. Define y concreta todo lo que el usuario puede realizar en el sistema, además de aportar valor al software desarrollado, ya que en él se ven reflejados los motivos de por qué alguien iba a usar el sistema.

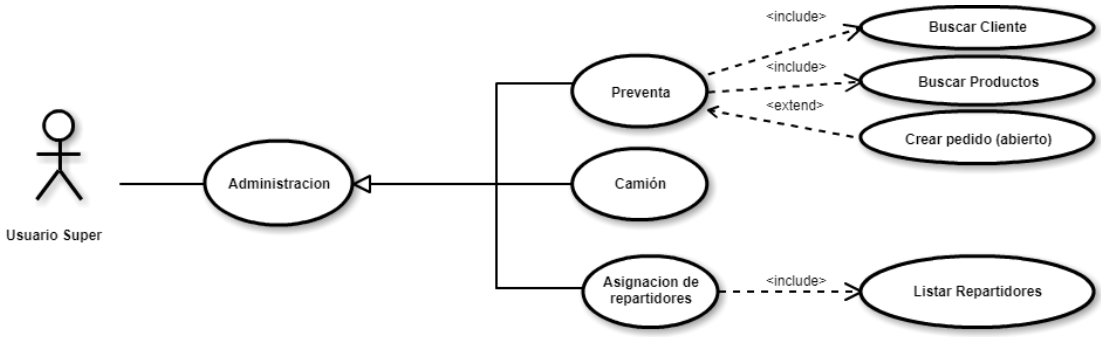

Ilustración 14.1

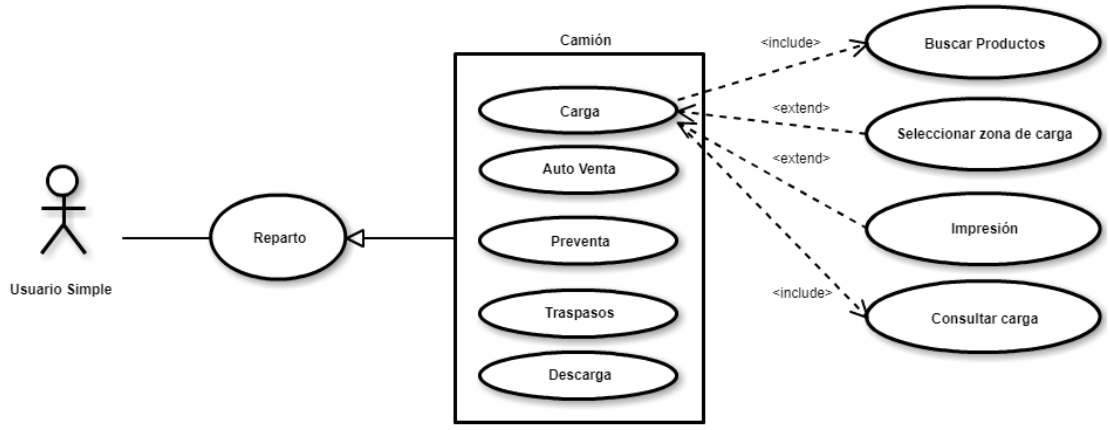

Ilustración 14.2

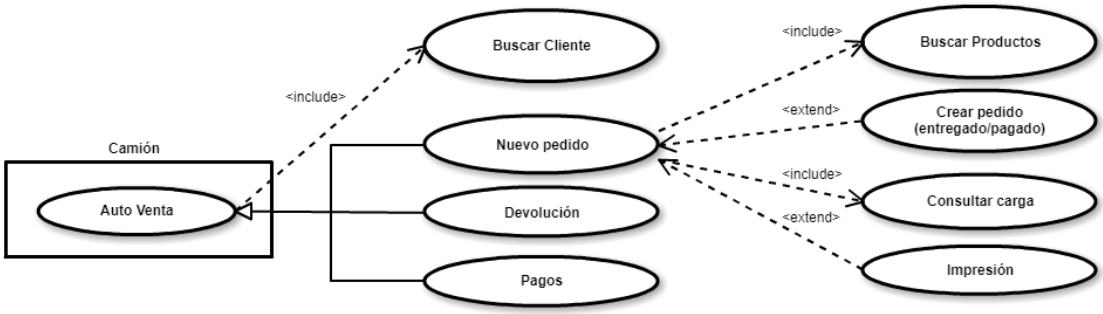

Ilustración 14.3

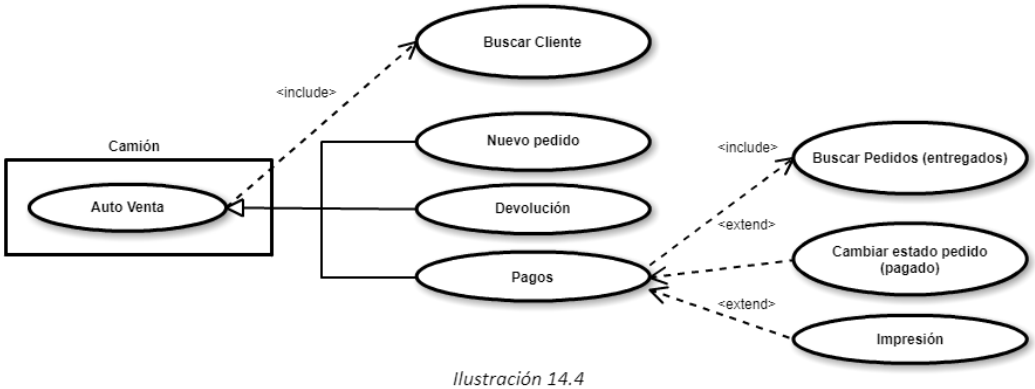

Ilustración 14.4

Ilustración 14 – Diagrama de casos de uso

La **ilustración 14.1** muestra las operaciones disponibles para un usuario super. La tarea de Administración se desglosa en 3 acciones disponibles, orientadas a la gestión y control de la venta. Dentro de la preventa, el usuario puede ir dejando pedidos abiertos, que más tarde serán recuperados y completados por los repartidores. Además, un usuario super puede desempeñar las funciones de un usuario repartidor, a través de las operaciones sobre el camión. Estas operaciones están desglosadas a continuación en la **Ilustración 14.2**. También los usuarios super pueden asignar repartidores a camiones en concreto, según diga la necesidad.

En la **ilustración 14.2** se muestra un ejemplo de caso de uso de una carga de material en el camión. Esta función dentro de las operaciones del camión da la posibilidad de listar todos los productos disponibles en el sistema, y cargar la cantidad deseada en el camión. Esta carga puede distribuirse por zonas, a modo informativo para el repartidor o el ayudante, que queda reflejado en el ticket impreso.

En el cuadro 'Camión' se pueden apreciar el resto de operaciones disponibles sobre el camión.

La **ilustración 14.3** refleja el proceso de crear un nuevo pedido a un cliente. Para ello, el repartidor debe elegir un cliente, y rellenar el pedido en base a la carga que lleve el camión. Puede ser que se realicen entregas parciales, ya por planificación o porque no queda más carga en el camión. Estas ventas se reflejan en un ticket impreso que debe ser entregado al cliente, que en algunas ocasiones puede requerir firma y copia. Estos pedidos se entregarían sin cobrar (albaranados) o cobrados (facturados).

Por último, **la ilustración 14.4** muestra el proceso de cobro de un pedido ya entregado. Para ello el repartidor, al igual que pare crear un pedido, debe introducir el cliente del que desea listar pedidos entregados, para proceder al cobro de los mismo. Este proceso genera una factura simplificada en forma de ticket impreso.

Requisitos del sistema

Planteada y analizada la problemática existente en los métodos de reparto de agua a domicilio tradicionales y realizado un desglose del tipo de usuarios potenciales que van a utilizar la app, se pueden extraer requisitos específicos de usuario para tratar de lograr una mayor aceptación de la aplicación entre un grupo de usuarios más diverso.

Analizando la tabla de dummies, se detectan como críticos los siguientes requisitos:

- La app debe actualizar los datos en el sistema en tiempo real, y de igual forma debe presentar datos actualizados.
- Los procesos derivados de la app deben automatizar al máximo cualquier tipo de tarea mecánica o repetitiva, como por ejemplo albaranar o facturar los pedidos automáticamente al entrar al sistema.
- Necesidad de una interfaz intuitiva y usable, bien guiada y con no demasiadas opciones para evitar que usuarios menos acostumbrados se pierdan en los procesos o los menús.
- Minimizar el uso de papel a la hora de notificar al cliente.
- Adaptarse a todos los tamaños de dispositivo sin perder funcionalidad
- Acometer cambios offline, que sean integrados en el sistema al volver la conectividad
- Ofrecer herramientas de consulta de ventas/carga en tiempo real

Mockups

Dentro del diseño de la app, la maqueta o mockup nos sirve para crear interfaces de usuario que den al usuario final una vista de cómo se verá el software sin necesidad de programar o desarrollar funcionalidades.

La maqueta de **aguapp** se creó una vez definidos los requisitos y necesidades de la misma para plasmar a modo de boceto las líneas generales del apartado gráfico del diseño. En esta maqueta se presentan las pantallas básicas y como fueron concebidas en su creación, aunque algunos diseños fueron finalmente modificados o reemplazados por necesidades de usabilidad y diseño.

aguapp sigue un diseño basado en pantallas independientes, a las que se le aplica scroll cuando es necesario para el desplazamiento vertical a lo largo de las pantallas.

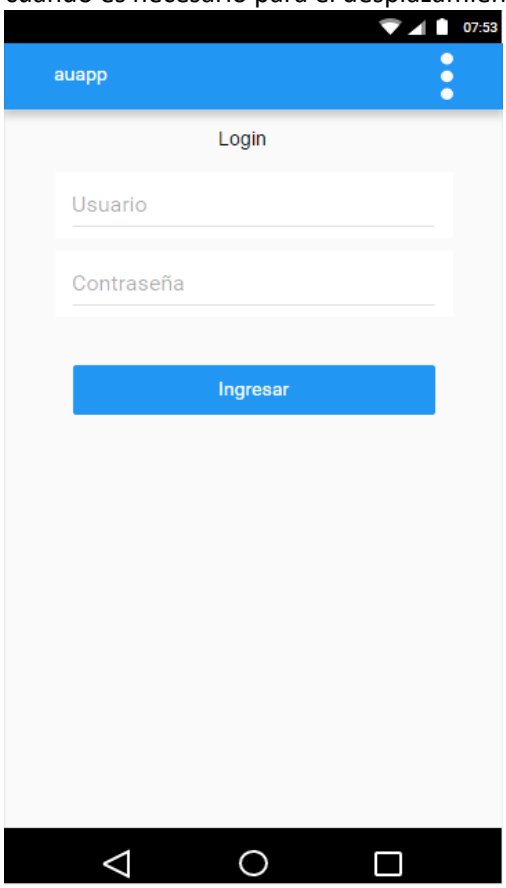

Ilustración 15 - Pantalla de Login

La pantalla de **Login (ilustración 15)**, así como todas las pantallas de aguapp, cuentan con una barra de actividades típica de Android como encabezado, en la que se muestra en todo momento una descripción o título de la página, así como un espacio de botones de opciones (los 3 puntos verticales blancos).

Para el login se presentan dos campos de texto para usuario y contraseña, y un botón de acceso. Los botones inferiores de la pantalla sobre el fondo negro corresponden a los botones del propio sistema Android, en este caso en particular de las versiones 5.x.

Desde el comienzo del diseño de la aplicación se consideró la posibilidad de incluir tipos diferenciados de usuarios. Para la maqueta se depuró más la idea, hacia una diferenciación usuario simple y super-usuario. La pantalla de **Administración (Ilustración 16)** estaría destinada para los tipos super-usuario después de realizar el login, los cuales podrían gestionar la preventa y realizar tareas como asignar conductores a camiones, así como todas las funcionalidades disponibles para el resto de tipos de usuario.

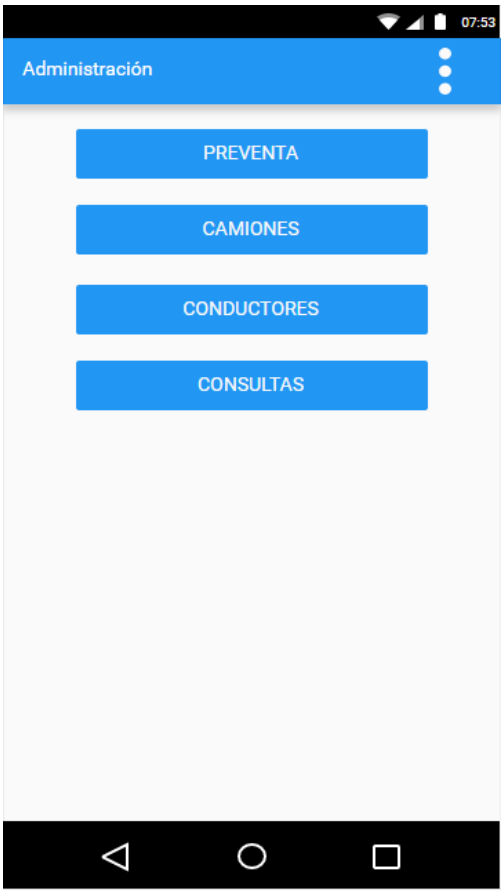

Ilustración 16 - Pantalla de superusuario

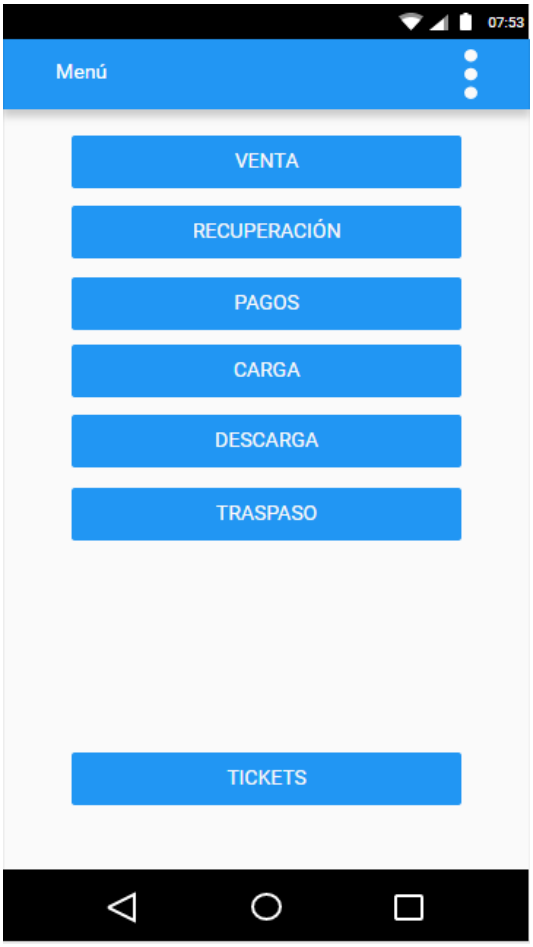

Ilustración 17 - Menú Usuario Simple

Asimismo, para los usuarios simples después del login se presentaría el **Menú (Ilustración 17*)***, desde el cual los usuarios simples (como los repartidores) tendrían disponibles aquellas funcionalidades ajustadas a ellos. En esta pantalla se listan:

- ***Venta***: Venta directa de productos.
- ***Recuperación***: Selección de pedidos asignados al repartidor, realizados desde la preventa. Estos pedidos estarían abiertos en el sistema, a la espera de ser rescatados por los repartidores y albaranados o facturados.
- ***Pagos***: Aquellos pedidos que sólo hayan sido entregados pueden ser facturados más tarde, ya sea por ejemplo porque al cliente se le asigna un crédito mensual, en el que se le factura una vez al mes por todo su consumo, o porque se ha realizado una entrega parcial.
- ***Carga***: Al comienzo de la jornada, los camiones deben cargar la mercancía en el almacén. Esta carga debe quedar registrada en el sistema, así como el resto movimientos de productos.
- ***Descarga***: Al igual que la carga, al finalizar la jornada los camiones deben volver al almacén y dejar la mercancía no vendida. Esto vuelca toda la carga de los mismos al almacen central, y al quedar registrado en el sistema por medio de la app, se cuentan con los datos de disponibilidad y ventas en tiempo real.
- ***Tickets***: Por último, se daría la opción a imprimir los últimos tickets o tickets anteriores.

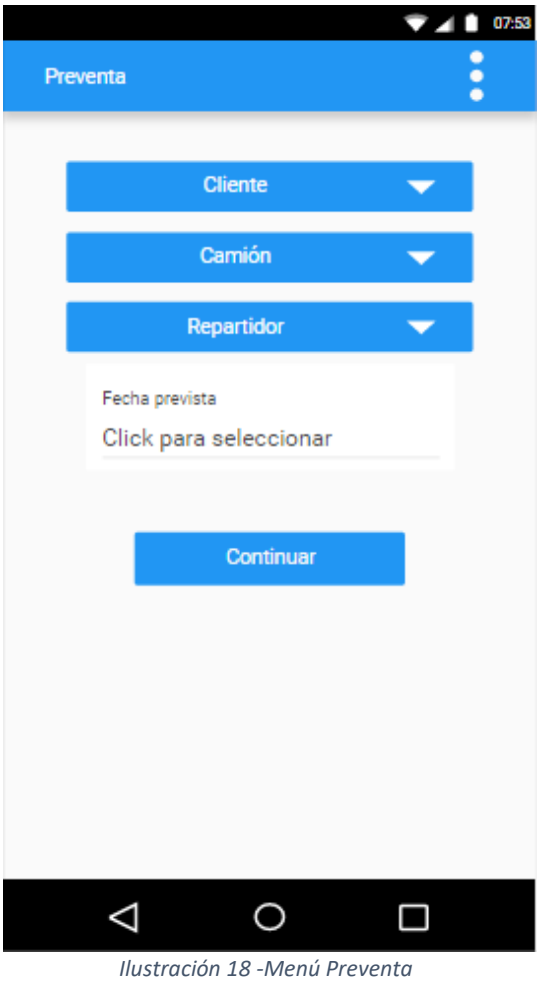

Ilustración 18 -Menú Preventa

Un usuario super podría acceder a la pantalla de **Preventa (Ilustración 18)**, en la cual podría realizar pedidos asignados a repartidores o camiones para que después los usuarios simples los recuperen a lo largo de su jornada y los completen. Esta funcionalidad sería una réplica de la funcionalidad que ofrecería el sistema de gestión informático de la empresa para la creación de pedidos.

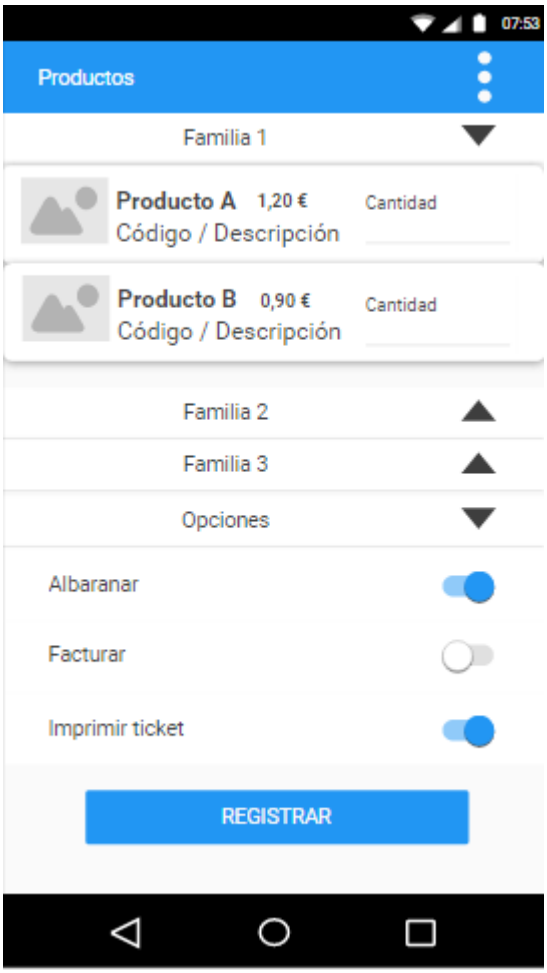

Ilustración 19 - Selección de productos

A la pantalla de **Productos (Ilustración 19)** se llegaría a través de cualquier función que requiera de una selección de productos, ya sea la carga, la venta o un traspaso.

Se presenta como una serie de menús expandibles que catalogarían las distintas familias de productos para que una vez expandidos muestren pequeñas fichas de productos, en los cuales se introduzca la cantidad deseada de cada producto. Una vez finalizada la inserción de datos, bastaría con pulsar el botón *Registrar* para validar los datos al sistema.

Esta es la parte de aguapp que más cambios ha sufrido de cara a su versión final, ya que por motivos de usabilidad se optó por mostrar tanto las familias como los productos en botones cuadrados más semejantes a los de Windows Phone.

A su vez, la inserción de datos por producto se separó a pantallas independientes para hacer más cómoda la inserción de las cantidades en otros tipos de unidad de medida, como pueden ser las cajas y los palés.

Modelo de negocio

El desarrollo de esta aplicación se presenta como un desarrollo a medida solicitado por un cliente, en concreto una empresa de reparto de aguas a domicilio, a una empresa de desarrollo de aplicaciones. No se pretende el uso público de la misma, ya que se pretende que su uso sea interno, dirigido a los trabajadores de la empresa.

Cabe destacar que la decisión de realizar el desarrollo de la aplicación en Android, recae sobre la intención del cliente de ahorrar costes, y para ello recurrir a terminales de gama media-baja Android, los cuales están muy por debajo del precio de cualquier iPhone actual. (Precio iPhone 7: 729€, Bq Aquaris U Lite: 149€, precios consultados en Phone House, marzo 2017)

Desarrollo

Alcance de la implementación

Como se indicó al inicio de este documento, se toma como modelo una empresa de reparto de aguas que cuente con un sistema informático de gestión. Para ello, la parte de este sistema se ha simulado como una base de datos MySQL que contenga una estructura de datos similar a la de este tipo de software, junto con una serie de servicios web tipo REST desarrollados con Slim Framework. En combinación, dotan a la app móvil de un origen de datos y a su vez de operaciones específicas del sistema, como asignación de números de línea en un pedido o la asignación de números de serie distintivos a dichos pedidos.

La implementación de esta aplicación se presenta a modo de prototipo, con el que se pueda apreciar el correcto planteamiento y desarrollo aguapp. El modelo de trabajo seguido ha sido en base a objetivos, simulando una toma de requerimientos firme y cerrada por parte del cliente, y buscando minimizar el tiempo de entrega y la creación de prototipos funcionales lo más rápido posible para el testeo de las funcionalidades consideradas como críticas, simulando un desarrollo ágil SCRUM.

Muchas características y funcionalidades no han sido implementadas, principalmente porque se salían de tiempo previsto para la realización del trabajo y ha sido prioritario desarrollar funcionalidades básicas o más importantes con el objetivo de conseguir un prototipo más que funcional. Consultar el apartado de **Limitaciones** para más detalle.

Versionado

Se ha usado GitLab junto con GitExtensions para llevar el control del versionado de la app, con la posibilidad de volver atrás en las distintas versiones y de crear ramas alternativas de desarrollo.

Diseño e Implementación

El patrón de arquitectura elegido a seguir fue el **Modelo-Vista-Controlador(MVC)**. Esto quiere decir que la organización del código dentro de la aplicación sigue un diseño modular, de forma que se puedan reemplazar archivos enteros siempre y cuando estos sigan respetando la relación con la vista o con el modelo que pueda haber.

Por poner un ejemplo, las clases de los archivos de código que se encargan de manejar de forma asíncrona las peticiones a los servicios fueron reemplazados en una etapa temprana del desarrollo de la aplicación por otros, sin verse afectado el resto del código.

Se ha buscado aprovechar las abstracciones de java al máximo a la hora de extender o anular funciones predefinidas con el fin de capturar el comportamiento de diversos llamadores y adaptarlos desde raíz al funcionamiento de la aplicación. De esta forma se consigue reducir el código enormemente al reutilizar buena parte de clases ya hechas y solamente cambiando las partes críticas.

Modelo-Vista-Controlador

El **MVC** es un patrón de arquitectura software que separa los **datos** y la **lógica de negocio** de una aplicación de la interfaz de usuario. Para ello el modelo propone separar tres componentes distintos que son el **modelo**, la **vista** y el **controlador (Ilustración 20)**. Por un lado se definen componentes para le representación de la información, y por otro para la interacción con el usuario.

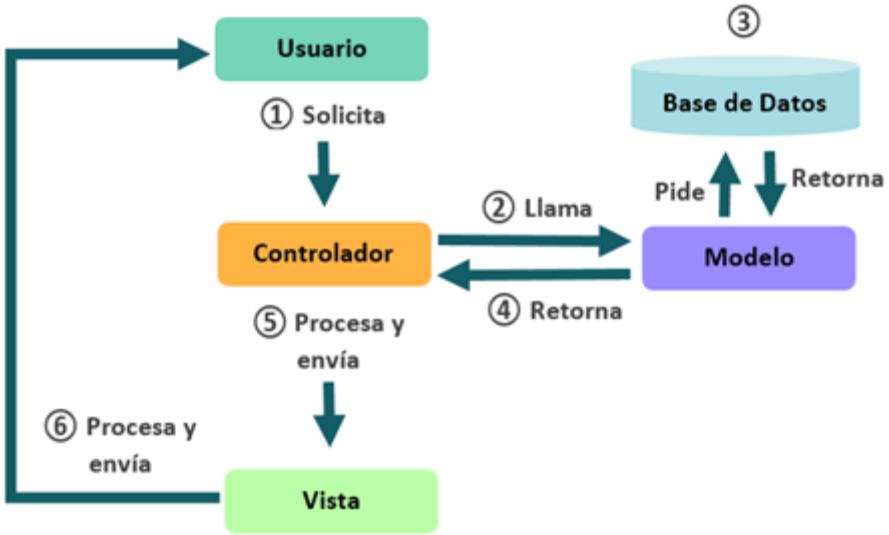

Ilustración 20 - Descripción del MVC

El **modelo** es la representación de la información con la cual opera el sistema. Gestiona los accesos a dicha información y provee de métodos de consulta, así como privilegios de acceso que se hayan especificado en los requisitos de la aplicación. El modelo envía a la vista aquella información que le es solicitada, y recibe peticiones a través del controlador. En el caso de aguapp, el papel del modelo lo juega la base de datos en combinación con una serie de servicios web.

El **controlador** es el encargado de responder a eventos y realizar peticiones al modelo cuando se hace alguna solicitud de información. En esta app el controlador son las funcionalidades de la aplicación, por ejemplo, el código de comunicación con los servicios web. Una vez se hace una petición a uno de estos servicios, cuando el dato retorna se trata de la forma que sea necesaria para dejarlo preparado para la vista. Se han implementado de forma asíncrona para que el hilo de ejecución principal, que es el que maneja la interfaz de usuario, no quede bloqueado. De esta forma se pueden cargar los datos de la aplicación de forma paralela mientras el usuario accede, y en caso de que llegue a navegar a una página y sus datos no estén disponibles, se mostrará un aviso de carga.

La **vista** presenta el modelo de una forma que el usuario pueda interactuar. Para una aplicación desarrollada en Android Studio, esta parte corresponde a los 'layouts', archivos en formato xml que describen a través de código los elementos visuales que se muestran por pantalla **Ilustración 21.** A cada vista de la aplicación le corresponde un controlador específico, el cual puede comunicarse con otros llegado el caso. De esta forma se simplifica el código dedicando un controlador para cada vista, y delegando a controladores generales los procesos reutilizables (búsqueda de clientes, listado de pedidos… cualquier operación que pueda producirse en más de una vista de la aplicación).

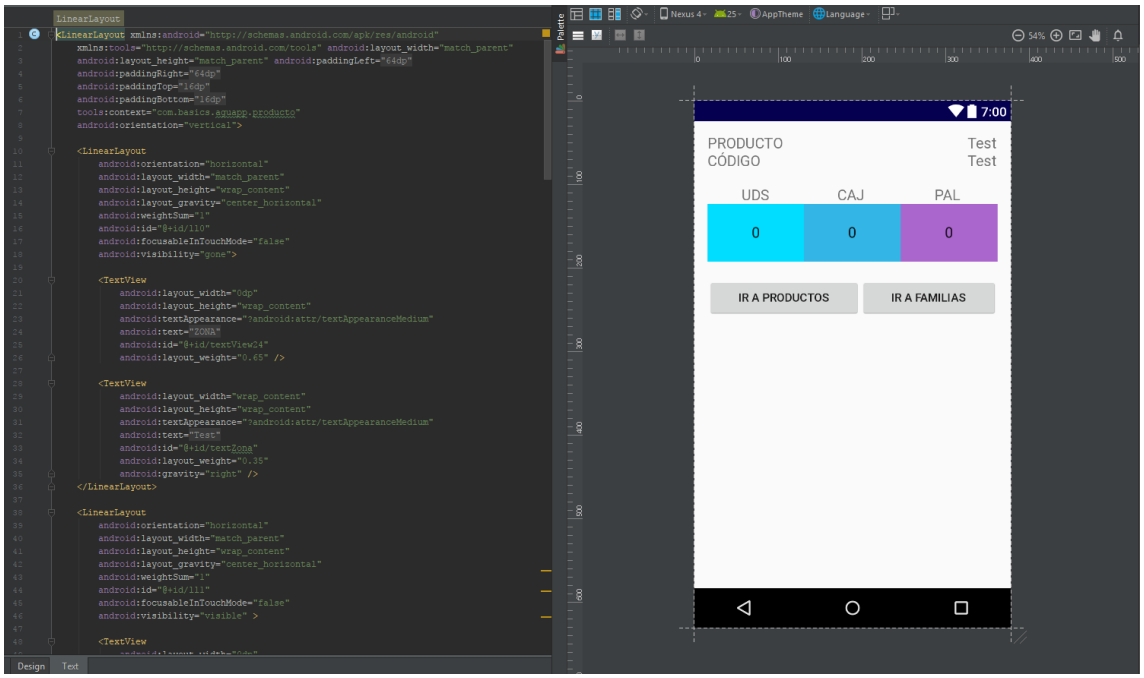

Ilustración 21 – Izquierda, código xml – Derecha, resultado

Para las aplicaciones basadas en el uso de **bases de datos** para gestionar los datos que debe manipular la aplicación, dicha gestión corresponde al **modelo**. La integración entre la **vista** y el **controlador** representa la unión entre la capa de negocio y la capa de presentación.

Descripción de directorios y archivos

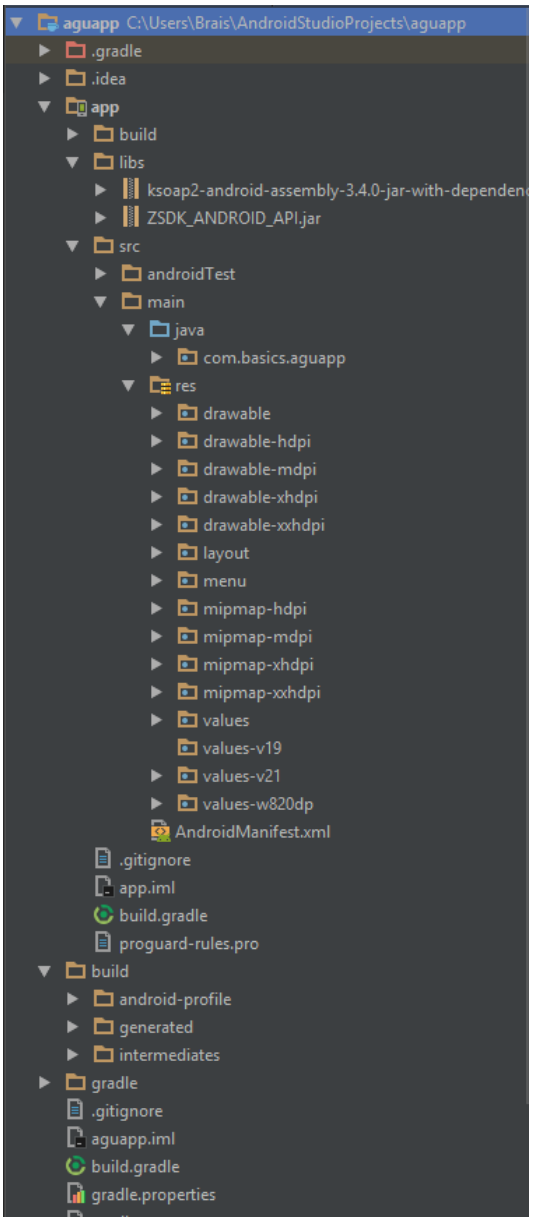

Ilustración 22 – Estructura del proyecto en Android Studio

En la **ilustración 22** se muestra el desglose de los directorios y subdirectorios que componen app. Por orden, se listan las relevantes para el proyecto:

- **app** Directorio principal, está organizado en subdirectorios
- **app > build** Archivos generados durante las fases de compilación del proyecto. Aquí es donde se general los archivos .apk. (archivo instalable de Android). En caso de ser una app que se desee publicar en PlayStore, Android Studio cuenta con funciones específicas

para subir el binario directamente al repositorio de Google, y tenerlo listo para la publicación sin subirlo manualmente.

- **app > libs** Carpeta donde se almacenan las librerías externas añadidas al proyecto. Estas pueden ser añadidas a mano o automáticamente relacionadas configurando el archivo build.gradle.
- **app > src > java** Contiene el código de la aplicación. Lugar donde se almacenan los archivos .java. No se ha desplegado el nodo por motivos de espacio, consultar el **Anexo B** para una lista completa de los elementos de este nodo.

  Los archivos de código pueden separarse en dos grupos, los encargados directamente de las vistas, y los encargados de la comunicación. Los primeros operan sobre el hilo principal de ejecución de la aplicación, mientras que los segundos operan de forma asíncrona para no bloquear las acciones del usuario en el hilo principal.
- **app > src > res** Almacena los recursos graficos relacionados con las vistas.

  Estos recursos se autogeneran para distintas densidades de pixels en pantalla (hdpi, mdpi, xdhpi..) y a su vez pueden generarse también para versiones específicas de Android.

  Se separan en:
  - **drawable** almacena la especificación en xml referente a aquellos elementos como botones y listas desplegables.
  - **layout** especifica también en xml las vistas generales de una página dentro de la aplicación.
  - **menú** donde especifica las opciones disponibles en cada página en su menú superior.
  - **mipmap** contiene elementos como por ejemplo el icono de la aplicación, o cualquier imagen que deba ser mostrada por defecto
  - **values** dentro de la cual se pueden especificar entre otros, colores y strings para reutilizarlas en distintas partes de la app.

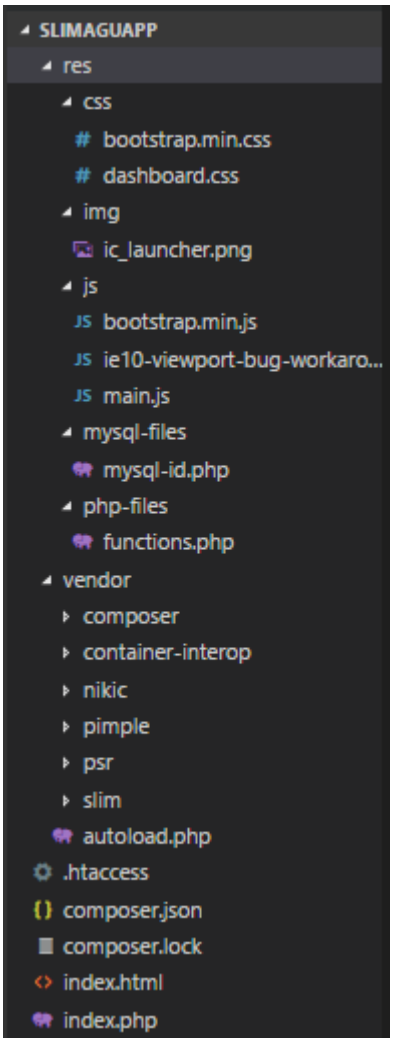

Ilustración 23 – Estructura del Slim

Al igual que se ha desglosado la estructura del proyecto de aguapp en Android Studio, ahora es el turno de los servicios web, Slim Framework. A continuación, se listan los directorios y subdirectorios que lo componen, mostrados en la **ilustración 23**.

- **res** Contiene subcarpetas ordenadas por tipo o finalidad
  - **css** acoge los ficheros de hojas de estilo. En este caso cuenta con los estilos de Bootstrap y una hoja de estilo personalizada para el panel de control. Bootstrap es una librería de estilos y funciones (HTML, CSS, JS) para desarrollo web de aplicaciones responsive.
  - **Img** contiene el icono de aguapp, utilizado como favicon y logo en el panel de control.
  - **Js** contiene los archivos de código escritos en javascript.

- **mysql-files** dentro se encuentre un archivo en php que especifica las credenciales de acceso la base de datos, y las funciones para crear la conexión.
- **php-files** contiene un archivo de funciones auxiliares en php. Una de ellas por ejemplo se encarga de codificar en UTF-8 las respuestas para evitar problemas de codificación y caracteres extraños.
- **vendor** es un directorio creado automáticamente por Composer. Composer es un gestor de dependencias php. Se pueden instalar paquetes independientemente o configurando un archivo composer.json y realizando una instalación global. Almacena los paquetes instalados adicionalmente, siendo los más relevantes el propio core de **slim**, o el directorio **nikic** que contiene un paquete para manejo de las cabeceras CORS (Cross-Origin Resource Sharing). En el archivo **autoload.php** se especifican todos los directorios y paquetes que deben cargarse, para que luego desde la aplicación principal solo haya que importar este archivo.
- **Index.html** contiene el código html del panel de control.
- **Index.php** especifica los métodos de Slim.

Modelo de la base de datos (diagrama de e-r)

Un diagrama entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. Este modelo representa a la realidad a través de un esquema gráfico empleando la terminología de Entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas Atributos.

En la **ilustración 24** se muestra un diagrama entidad-relación para facilitar la comprensión de la organización de la base de datos de la aplicación.

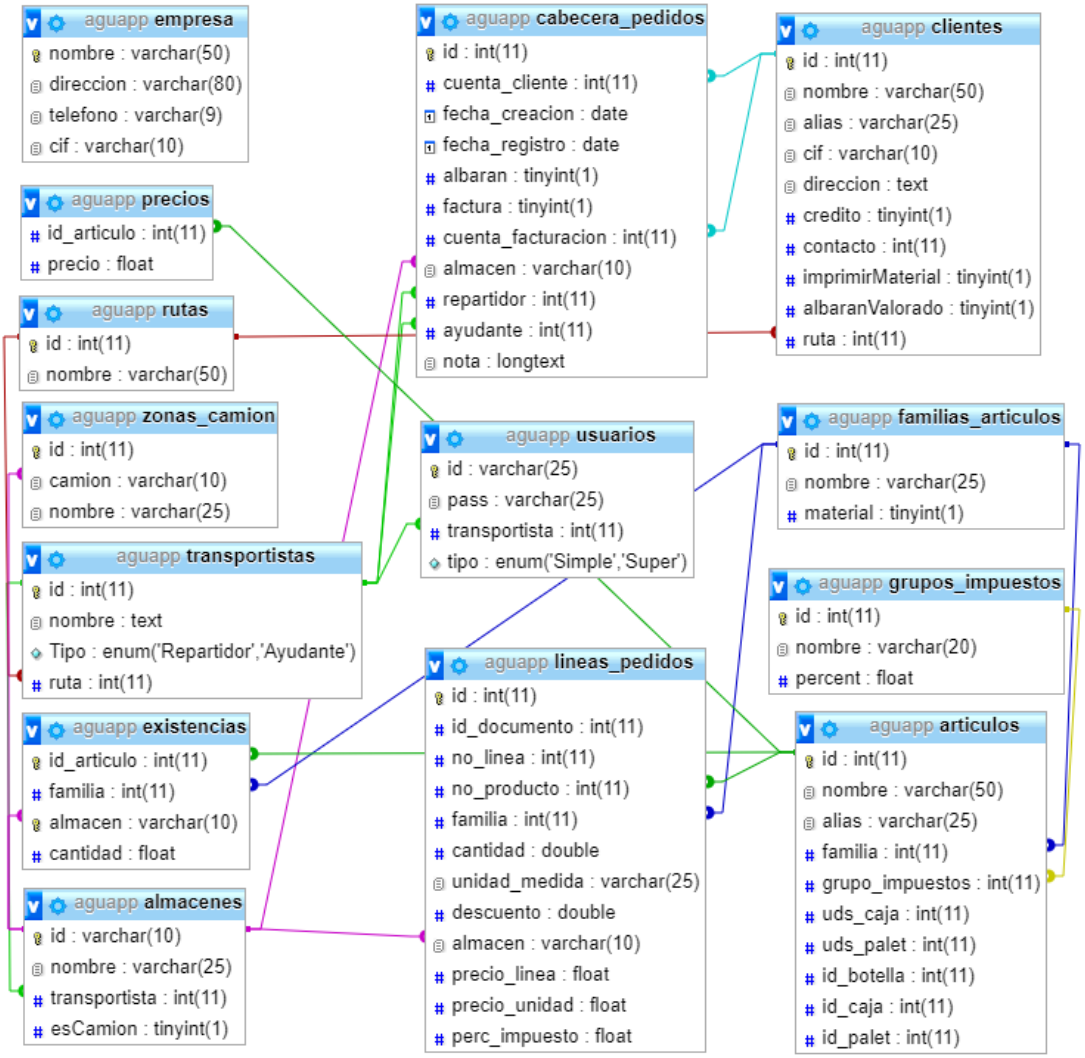

Ilustración 24 – Diagrama Entidad-Relación

Tecnologías

Una vez planteados y depurados los requisitos del sistema y del usuario, es hora de elegir qué tecnologías se van a usar en el desarrollo de la aplicación.

Lo primero que se ha planteado ha sido un control de versiones, para llevar un seguimiento de los cambios, y más importante, la posibilidad de volver a una versión anterior en caso de problemas o necesidad. Para esta parte se ha usado el repositorio GitLab junto con la herramienta GitExtensions.

En cuanto al entorno de desarrollo, valorando el requisito de que la aplicación va a ser para dispositivos Android, se ha elegido el entorno de desarrollo Android Studio. Es un entorno gratuito con gran apoyo de la comunidad y con actualizaciones constantes, siempre por delante de la versión estable publica de Android. Una de las grandes ventajas que aporta Android Studio frente a otros entornos o métodos de desarrollo, es que aporta un control total sobre todos los elementos nativos del dispositivo. Esto puede convertirse en un arma de doble filo ya que debido a la profundidad de su alcance los desarrollos pueden complicarse.

Ahora bien, se necesita un actor que haga las funciones del sistema de gestión instalado en la empresa tipo ERP. Para ello, lo que se ha elegido es una combinación entre una base de datos mysql, y un framework ligero de servicios web tipo REST, Slim Framework. Estos servicios web desarrollados en Slim se encargarían de recibir y gestionar las peticiones del controlador, y servir los datos de forma que puedan ser enviados a la vista. Además, simulan los procesos que realizaría un ERP con los datos, por lo que no solo se encarga de servir el contenido de la base de datos.

Otro de los requerimientos es que se pueden imprimir tickets-resumen de las operaciones. Contando que es una aplicación móvil, es necesario que la impresora también lo sea. Por ello se toma como modelo una impresora bluetooth Zebra iMZ320. Es una impresora térmica para facturas o etiquetas de 72mm, suficiente para las necesidades de aguapp. Como dato, pesa solo 340 gramos, lo que la hace cómoda de llevar encima. Una de las ventajas de este tipo de impresora es que no hay que reemplazar la tinta ya que carecen de ella. Las impresoras térmicas usan un tipo de papel especial sensible al calor emitido por los cabezales térmicos de la impresora. Por contraposición, este tipo de papel es algo más caro que los normales.

GitLab

Gitlab es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también un sistema de organización en base a grupos, a los cuales de se pueden adscribir distintos usuarios con permisos específicos. Se ha elegido principalmente por su sistema de versionado basado en Git. Entre sus órdenes básicas esta `git commit` (confirmación de cambios sobre le repositorio) y `git push` (subida de cambios al repositorio remoto). Estos cambios en el repositorio pueden ser revertidos a cualquier versión anterior.

También cuenta con un sistema de tareas e incidencias dentro de cada proyecto, para que sea más fácil llevar una gestión de errores o tareas pendientes.

GitExtensions

GitExtensions en una herramienta autónoma para los repositorios Git, desarrollada en Visual Studio para Windows y Linux. Provee de un entorno gráfico para gestionar los distintos repositorios tanto locales como remotos.

En la **ilustración 25** se puede ver un ejemplo de la interfaz. En la bara de tareas superior, están las operaciones básicas de git: Commit (botón verde con una v blanca) pull (copia del repositorio remoto al local) y push (subida del repositorio local al remoto).

En la parte superior se puede ver una lista de los últimos commits sobre el repositorio. A estos commits se puede añadir un mensaje descriptivo que será grabado en todos los archivos modificados. Tambien se puede ver la rama en la que se está actualmente (origin/master) el usuario del ultimo commit y cuando tiempo hace que se realizó dicho commit.

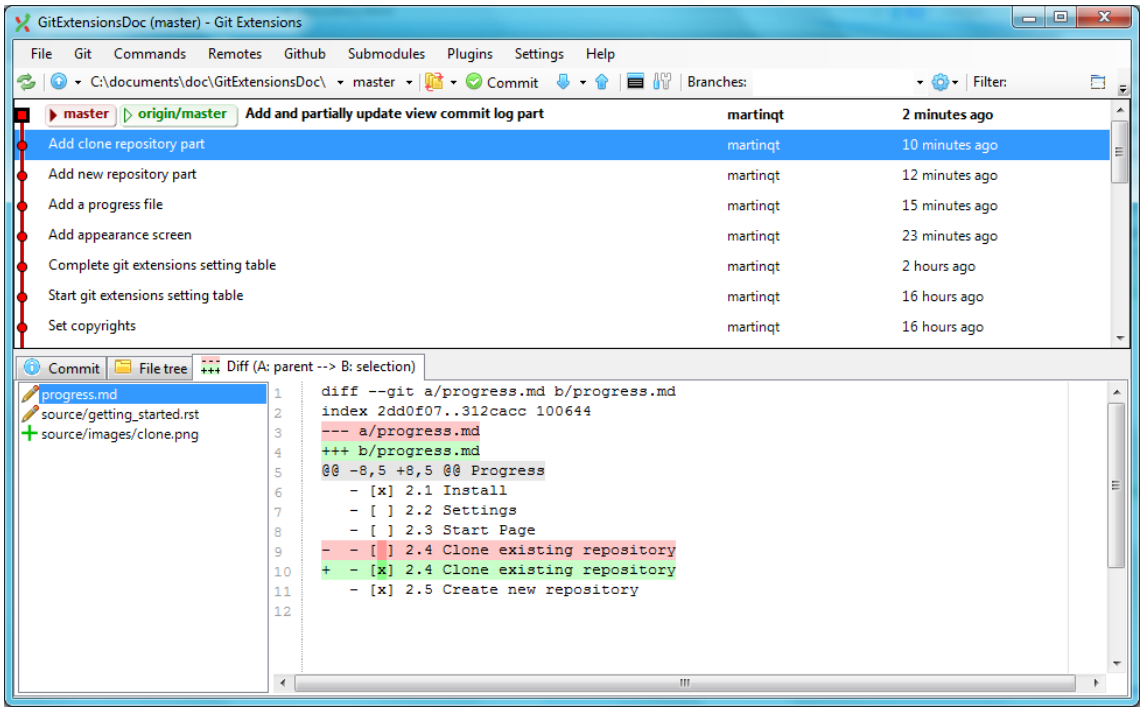

Ilustración 25 – Interfax de GitExtensions

Android Studio

El desarrollo de la aplicación se ha llevado a cabo principalmente en el entorno de desarrollo Android Studio. **Android Studio** es el entorno de desarrollo integrado oficial para la plataforma Android. Está basado en el software IntelliJ IDEA de JetBrains. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de aplicaciones Android, motivo principal por el que se ha elegido como entorno de desarrollo para aguapp.

Caracteristicas de Android Studio

Las siguientes características se proporcionan en la versión estable actual (2.3.1):

- Integración de ProGuard y funciones de firma de aplicaciones para publicación directa en PlayStore, lo cual no es necesario para aguapp.
- Renderizado en tiempo real del código xml que representa la capa de presentación.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle, gestor de dependencias. No se ha usado en aguapp ya que las librerías adicionales se han añadido a mano. (consultar apartado Librerías)
- Refactorización específica de Android y arreglos rápidos. Reorganización automática del código. Elimina comentarios sobrantes e indenta adecuadamente el código.
- Un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario, en vez de codificarlos directamente en xml.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.
- Soporte integrado para Google Cloud Platform, que permite la integración con Google Cloud Messaging y App Engine, también con Firebase (sistema de notificaciones de Google)
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones. También se puede probar directamente en un dispositivo conectado por usb.

Requisitos

- Java Development Kit (JDK) 8

Servicios web – Slim Framework

Slim es un micro framework PHP que simplifica el desarrollo y la implementación rápida de API's tipo REST y aplicaciones. Esencialmente se encarga de recibir una petición HTTP e invocar una rutina de devolución de llamada apropiada, devolviéndola en una respuesta HTTP.

REST son las siglas de Representational State Transfer. Fue definido hace una década por Roy Fielding en su tesis doctoral, y proporciona una forma sencilla de interacción entre sistemas, la mayor parte de las veces a través de un navegador web y HTTP. Esta cohesión con HTTP viene también de que Roy es uno de los principales autores de HTTP.

REST es un estilo arquitectónico, un conjunto de convenciones para aplicaciones web y servicios web, que se centra principalmente en la manipulación de recursos a través de especificaciones HTTP. Podemos decir que REST es una interfaz web estándar y simple que nos permite interactuar con servicios web de una manera muy cómoda.

Gracias a REST la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST).

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Características de Slim Framework

- Creador de rutas API REST
    - Soporta métodos HTTP standard y personalizados. Para aguapp, los métodos implementados sobre HTTP standard
    - Parámetros de ruta con comodines y condiciones. Esto nos es muy útil para evitar la duplicidad del código y dar más flexibilidad a las llamadas a la API, especialmente cuando son procesos similares en los que, por ejemplo, se requiere un parámetro de más o de menos a la hora de hacer la llamada.
    - Redirecciones de rutas, paros y saltos. En concreto en aguapp se hace uso principalmente de la redirección de rutas, ocultando la estructura del servidor a los consumidores.
- Renderizado de plantillas y vistas personalizadas.
- Mensajes Flash.
- Encriptación segura de cookies con AES-256.
- Caché HTTP.
- Logging de accesos personalizado.
- Gestión de errores.
- Configuración **muy** sencilla.

Requerimientos

- Es necesario tener instalado PHP 5.3.0 o superior.
- Servidor web tipo Apache o Ngnix.

Ejemplo

El siguiente código (**ilustración 26**) muestra una aplicación de ejemplo, en la que se devuelve un saludo si se hace una petición GET HTTP a la ruta /Slim/hello/[nombre] dentro del servidor en el que este implementado Slim.

```php
<?php
// Creación y configuración de la app Slim
$app = new \Slim\App;

// Definición de rutas y funciones
$app->get('/hello/{name}', function ($request, $response, $args) {
    return $response->write("Hello " . $args['name']);
});

// Ejecutar app
$app->run();
```

Ilustración 26 – Ejemplo de servicio Slim

Impresora Zebra iMZ320

La impresora portátil IMZ320 (**ilustración 24**) es un modelo de uso industrial, orientada a ámbito de transporte, retail (ventas), fabricación y sector hotelero. Imprime en rollo térmico de 72mm.

Ilustración 27 – Zebra iMZ320

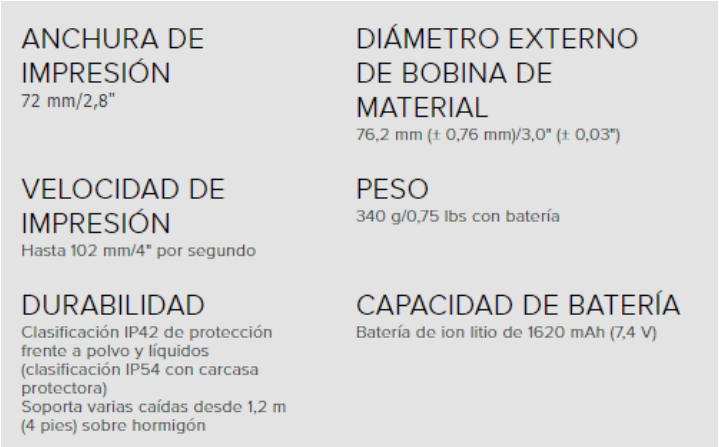

Ilustración 28 – Especificaciones

Cabe destacar su tamaño ligero y compacto, además de su peso reducido (**ilustración 27**) hace que resulte fácil llevarla durante períodos de tiempo prolongados, ya que puede acoplarse fácilmente a un cinturón o a la misma prenda.

Compatible con los sistemas operativos Apple IOS, Android y Windows para su uso con los dispositivos más recientes de smartphone y tablet, esta impresora es perfecta para gran diversidad de aplicaciones y clientes.

Para su integración con aguapp, se ha recurrido al lenguaje de programación **ZPL** (Zebra Programming Language). ZPL es un lenguaje de descripción de páginas que describe el formato de las páginas, la colocación de texto y los elementos gráficos para obtener una calidad de impresión óptima. Es un lenguaje propio de Zebra, pero a dia de hoy muchos otros fabricantes incorporan intérpretes y emuladores de ZPL.

Existen más de 170 comandos en la versión actual **ZPL** II, por lo que en este ejemplo se mostrará lo básico para imprimir una línea.

Queremos imprimir el nombre de la aplicación, “aguapp” en un ticket (**ilustración 30)**. Todas las líneas de comando empiezan con el símbolo **^.** Cada formato comienza con el comando **^XA** y debe terminar con el comando **^XZ**.

```
^XA
^LH 10,10
^FO20,10
^ADN,50,30
^FDaguapp^FS
^LH0,0
^XZ
```

Ilustración 29 – Ejemplo de código ZPL

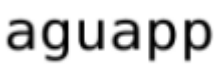

Ilustración 30 – Resultado

Además de los comandos ya comentados anteriormente, en la **ilustración 29** podemos apreciar el comando **^FO**, que indica la posición de la impresión respecto a los márgenes superior (20) e izquierdo (10). La siguiente línea que comienza con **^ADN** especifica el tamaño de la fuente. El contenido de la impresión debe contenerse entre las etiquetas **^FD** y **^FS**.

Librerías

Para la implementación de esta app se han incluido dos librerías, una de ellas de cara a aportar una conectividad futura a la aplicación, y la segunda para las comunicaciones por medio de bluetooth con la impresora portátil.

Ksoap

En principio la aplicación se planteó para implementarla directamente contra un sistema ERP Microsoft Dynamics NAV, dentro del cual se pueden desplegar muy fácilmente servicios tipo SOAP. Por ello se buscó una librería de código abierto para java con la que tratar fácilmente los objetos tipo SOAP en las llamadas a los servicios.

La elección final fue kSOAP 2, una libraría de muy poco peso, que la componen un parser XML, un serializador/deserializador, y una capa de transporte.

El **parser XML** se encarga de obtener la respuesta XML de las peticiones HTTP realizadas a los servicios web.

Para pasar los datos del XML a un objeto java se utiliza el **serializador/deserializador**, que traduce los distintos atributos y propiedades de un archivo XML a objetos nativos de java para permitir su navegabilidad.

La **capa de transporte** es la encargada de dotar de mecanismos específicas para el intercambio de mensajes entre el cliente y el servidor.

Finalmente se optó por simplificar el planteamiento y simular la parte de Microsoft Dynamics NAV, ya que el tiempo necesario para crear una implementación de NAV, configurarla y desarrollar la lógica necesaria para el funcionamiento de la aplicación se salía del tiempo disponible. Consultar el apartado de **Limitaciones.**

ZSDK Android API

Esta librería forma parte del SDK de Zebra, específicamente orientada a java y la conexión de las impresoras Zebra.

Provee de las funciones necesarias para la localización y el emparejamiento bluetooth, así como comunicación para impresión. En la **Ilustración 31** se muestra un ejemplo de uso de esta librería, en la que se anulan los métodos ‘foundPrinter’ y ‘discoveryFinished’ para capturarlos y gestionar su comportamiento, para en este caso guardar las MACs de las

impresoras, que serán más tarde utilizadas para la comunicación durante la impresión, y mostrar mensajes informativos al final del proceso.

```
@Override
public void foundPrinter(final DiscoveredPrinter discoveredPrinter) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            map = discoveredPrinter.getDiscoveryDataMap();
            adapterPrints.add((String) map.get("FRIENDLY_NAME"));
            printerMACS.add((String) map.get("MAC_ADDRESS"));
            adapterPrints.notifyDataSetChanged();
        }
    });
}

@Override
public void discoveryFinished() {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            dialog.dismiss();
            if (adapterPrints.getCount() == 1) {
                Toast.makeText(context, " Descubierto 1 dispositivo",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context, " Descubiertos " +
adapterPrints.getCount() + " dispositivos",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Ilustración 31 – Ejemplo de uso de la librería

Modelo de funcionamiento

A modo de resumen, y para reflejar de forma gráfica las tecnologías y estudiadas en este apartado, se añade a continuación un esquema en el que se relacionan estas tecnologías entre si (**ilustración 32**)

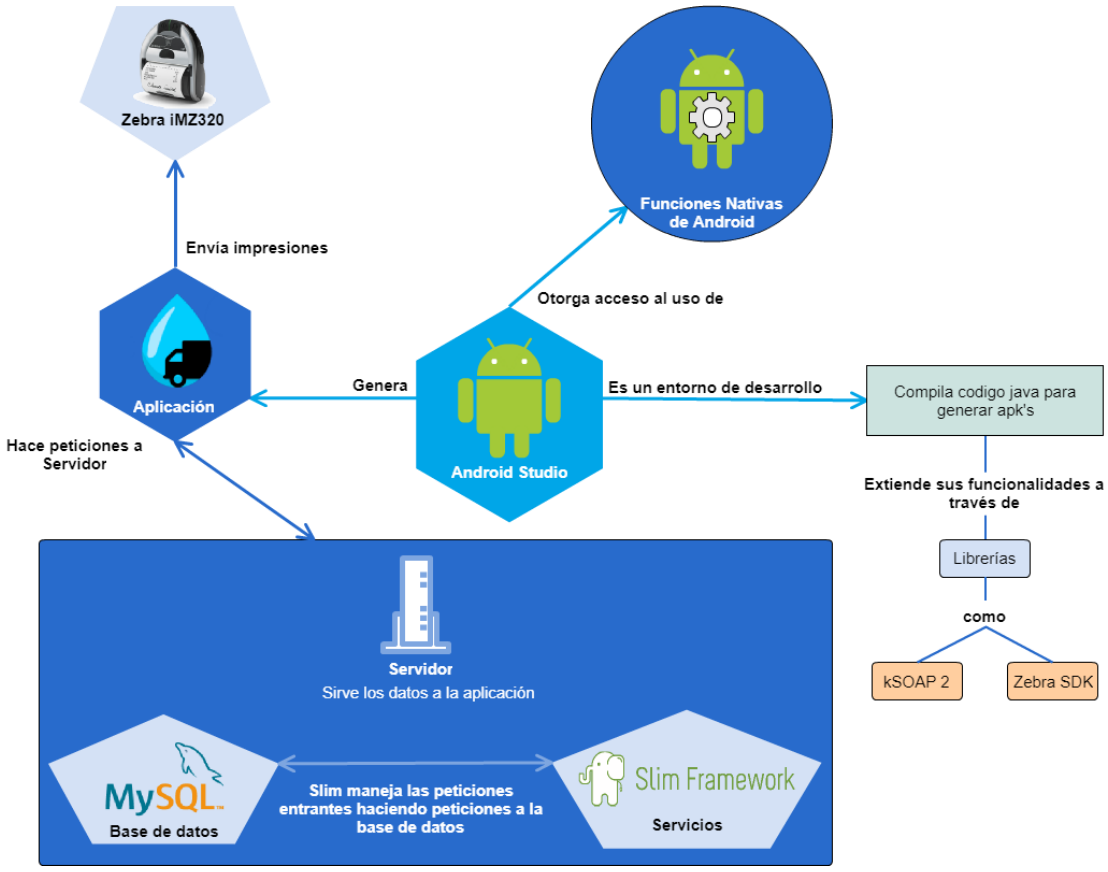

Ilustración 32 – Resumen en forma de gráfico de las tecnologías utilizadas

Seguridad

Pese a ser una aplicación orientada al uso interno dentro de una empresa, debe ser dotada de mecanismos de seguridad básicos en las comunicaciones.

Estos mecanismos comienzan aplicándose a nivel de la base de datos. En la tabla de usuarios, donde se almacenan los datos de acceso de cada uno de los usuarios de la app, la contraseña se guarda una función hash de la misma. Esto hace que las contraseñas queden codificadas de manera irreversible (la función hash genera un valor hash que puede ser generado por otro valor de entrada, de forma que a través de un valor hash no se va a tener toda la información para obtener su valor original).

Si bien la parte del servidor es una simulación de un sistema ERP, que contaría con sus propios sistemas de seguridad bien definidos, se ha implementado reescritura de url para los servicios web, de forma que no se da pista de la organización real del servidor. Los servicios de Slim también cuentan con la posibilidad de configurar CORS (Cross Origin HTTP Request), de forma que se fácil controlar el acceso a los servicios por ips especificas o rango.

Una forma efectiva de dotar de seguridad a la app para el entorno en el que se piensa utilizar sería el configurar una VPN (Virtual Private Network) junto con CORS para restringir el acceso a los servicios a la red de la VPN.

Pruebas

Para las pruebas de la aplicación se han creado una serie de datos ficticios, con el fin de simplemente ser representativos de las funcionalidades de la aplicación. Estos datos incluyen repartidores, productos, camiones y clientes. No se tienen en cuenta los procesos de aprovisionamiento de la empresa, el stock que hay en almacén para realizar la carga es el que hay disponible.

Las pruebas se han apoyado en el depurador de código de Android Studio, que permite depurar el código con la aplicación directamente ejecutándose en un dispositivo físico. Esto permite un testeo mucho más directo y ágil, en vez de probar en simuladores o interpretes web. Permite ver la pantalla de Log de la ejecución de Android en distintos niveles (verbose, error, warning). Las pruebas se han llevado a cabo a mano, ya que es interesante simular las situaciones reales que suceden en la calle para probar todas las casuísticas.

La aplicación se ha testeado en los siguientes dispositivos:

- Bq Aquaris X5 – Android 6.0.1 - Marshmallow
- Sony Xperia Neo – Android 4.2 - Jelly Bean
- Bq Aquaris M5 – Android 7.0 - Nougat

Para poner a prueba las funcionalidades de la aplicación y su robustez, se han planificado circuitos de prueba que ‘recorren’ todas las pantallas y funciones de la app. El tipo de pruebas llevado a cabo en este bloque lo podemos clasificar como pruebas de caja blanca, o pruebas estructurales. Es un tipo de pruebas muy ligado al código fuente, con el fin de examinar cada uno de los flujos de ejecución de la aplicación, asegurando que se devuelve el valor esperado a la finalización de cada uno de ellos. Cabe destacar que sometiendo a aguapp a este tipo de pruebas, pese a superarlas, no se detectarían partes incompletas de la especificación de requisitos, o requisitos faltantes.

Un punto importante en las pruebas estructurales es que cuando más crece la implementación de la aplicación, más profundas y extensas son las pruebas, lo que en un equipo de desarrollo imitado en cuanto a recursos humanos puede suponer problemas de tiempo a la larga.

Dado el gran número de flujos de ejecución y distintas variantes, se ha optado por presentar en este apartado las pruebas relativas a la **carga** del camión. Este proceso de prueba cubre desde el login en la app hasta la impresión del ticket de carga.

Asimismo, el proceso completo de prueba para una carga se desglosa a continuación:

- Login (**ilustración 33**)
  - Usuario y contraseña correctos
  - Usuario faltante
  - Contraseña faltante
  - Usuario o contraseña incorrectos
- Gestión del camión (**ilustración 34**)
  - Cambiar camión predefinido
  - Seleccionar/cambiar ayudante
- Camión (**ilustración 35**)
  - Selección de operación carga
- Carga – Selector de familias / productos (**ilustración 36**), (**ilustración 37**), (**ilustración 38**)
  - Cambio de zona de carga
  - Comprobar navegabilidad hasta ultimo nivel (producto)
- Producto (**ilustración 39**)
  - Introducir cantidad en unidades
  - Introducir cantidad en cajas
  - Introducir cantidad en palets
  - Navegar a productos
  - Navegar a familias
  - Navegar a resumen
- Resumen (**ilustración 40**), (**ilustración 41**)
  - Comprobar carga por zona
  - Registro de la carga
  - Selección de fecha
- Impresión (**ilustración 42**)
  - Comprobar consistencia de datos entre la operación en la aplicación y el resultado en el ticket

Ilustración 33 – Pantalla de login

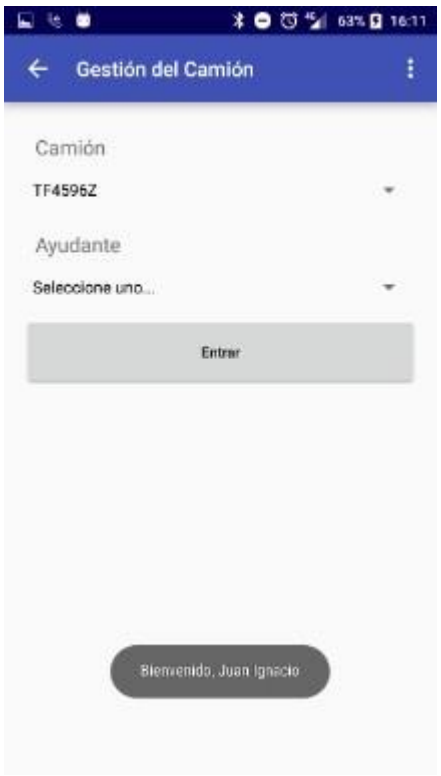

Ilustración 34 – Gestión del camión

Ilustración 35 – Operaciones sobre el camión

Ilustración 36 – Selector de familias, carga

Ilustración 37 – Selección de zona de carga

Ilustración 38 – Selector de productos

Ilustración 39 – Pantalla de producto

Ilustración 40 – Pantalla de resumen

Ilustración 41 – Pantalla de resumen, selección de fecha de carga

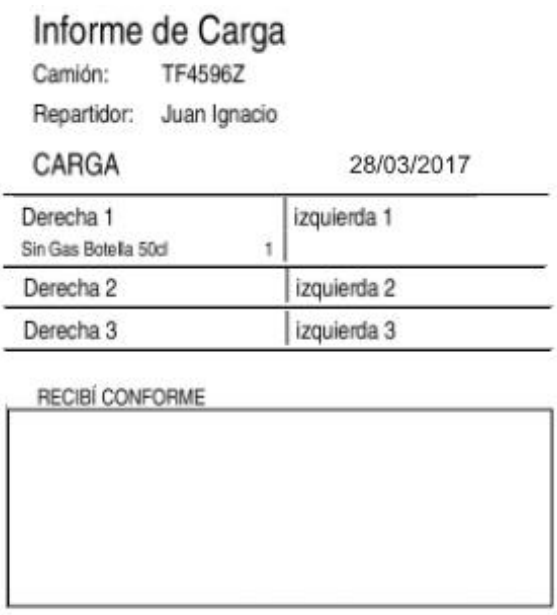

Ilustración 42 – Ticket de carga

Para apoyar las pruebas y comprobar la correctitud de los resultados de las operaciones realizadas desde la aplicación, se ha desarrollado una pequeña herramienta en HTML y Javascript con la que acceder de una forma sencilla a los datos, a modo de "Panel de Control".

Este panel de control aprovecha los mismos servicios de los que se sirve la aplicación, aparte de un método específico para listar los pedidos junto con sus líneas, de forma que no se ha desarrollado código redundante. Lista los principales módulos de la aplicación, que son los pedidos y las existencias en almacén, como se puede ver en la **ilustración 43**.

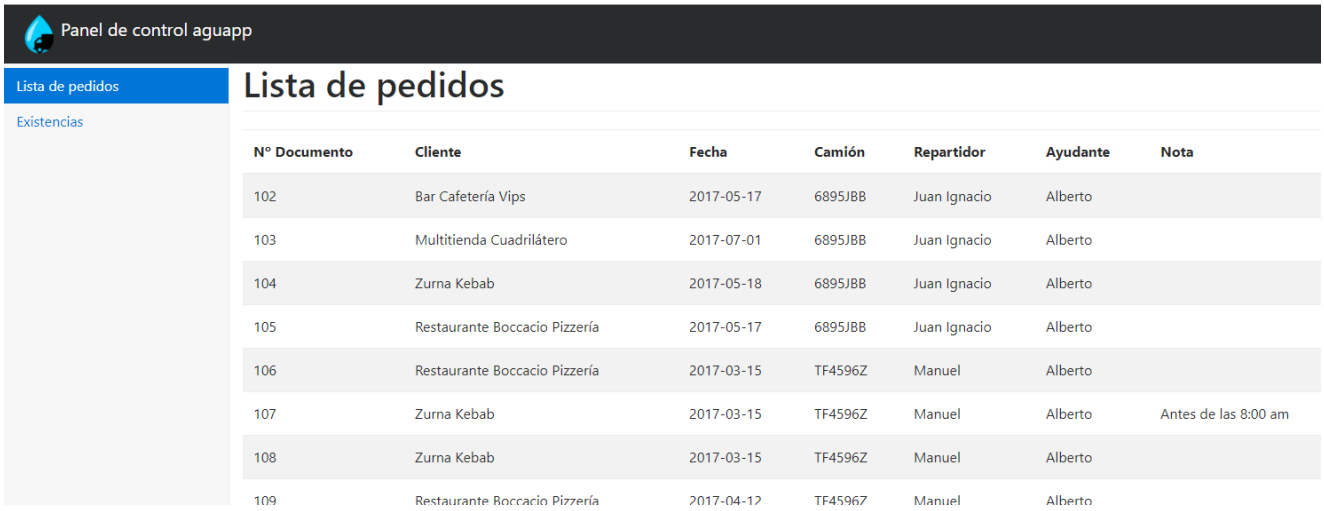

Ilustración 43.1 – Panel de control

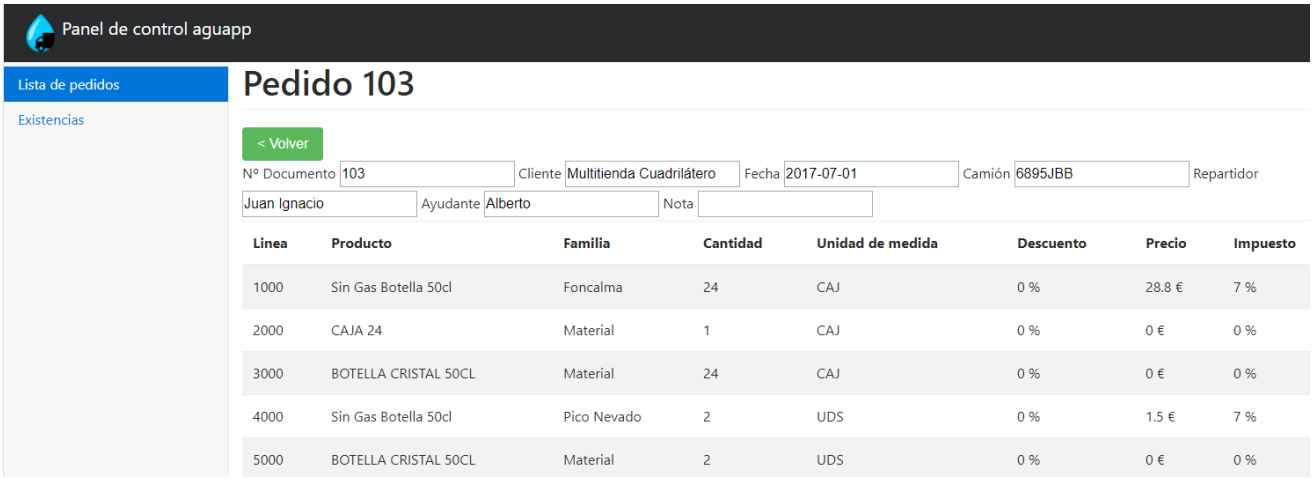

Ilustración 43.2 – Panel de control – Detalle del pedido

Despliegue

Existe una versión de aguapp desarrollada por al alumno para **Alio Soluciones**. Esta versión esta implementada directamente contra el ERP Microsoft Dynamics NAV para su uso en la empresa **Ancruzfir Distribuciones SL.**

Fue implantada en 2016, siendo usada a diario desde entonces. Cuenta con más funciones aparte de las mostradas por aguapp, gran parte de ellas solicitadas después de la estabilización de la implantación. En este caso se carece de servidor web específico ya que NAV presta todas las funcionalidades necesarias, incluidos los servicios web, no solo la base de datos y la lógica. Uno de los elementos que compone NAV son las Páginas, elementos que se encargan de mostrar al usuario la información almacenada en las tablas de la base de datos. Estas páginas pueden ser publicadas directamente a través de la interfaz de NAV como servicio web tipo SOAP o ODATA, sin necesidad de desarrollo añadido. En la **ilustración 44** se muestra un ejemplo de la especificación WSDL (Web Services Description Language) de la página de los pedidos de ventas. Se puede ver en la parte superior la especificación (contraída por espacio) de los distintos tipos de datos disponibles, y empezando por **Read** (nodo abierto) sus métodos. Los métodos básicos CRUD (create, read, update, delete) son implementados automáticamente por el sistema a la hora de la publicación.

```
+ <xsd:complexType name="Sales_Order_Line_List">
+ <xsd:complexType name="Integración_Sales_Order">
+ <xsd:complexType name="Integración_Sales_Order_List">
+ <xsd:simpleType name="Integración_Sales_Order_Fields">
+ <xsd:complexType name="Integración_Sales_Order_Filter">
- <xsd:element name="Read">
  - <xsd:complexType>
    - <xsd:sequence>
        <xsd:element minOccurs="1" maxOccurs="1" name="No" type="xsd:string" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
+ <xsd:element name="Read_Result">
+ <xsd:element name="ReadByRecId">
+ <xsd:element name="ReadByRecId_Result">
+ <xsd:element name="ReadMultiple">
+ <xsd:element name="ReadMultiple_Result">
+ <xsd:element name="IsUpdated">
+ <xsd:element name="IsUpdated_Result">
+ <xsd:element name="GetRecIdFromKey">
```

Ilustración 44 – WSDL de una página de NAV publicada como servicio

Estas funciones básicas pueden ser ampliadas en código en C/AL dentro del sistema. C/AL, o por sus siglas **C**lient/server **A**pplication **L**anguage, es un lenguaje de programación utitiizado dentro de C/SIDE (**C**lient/**S**erver **I**ntegrated **D**evelopment **E**nvironment) en Microsoft Dynamics NAV. Es un lenguaje especialmente orientado a tratamiento de bases de datos, cogiendo parte de la base de Pascal. En la **ilustración 45** puede verse un ejemplo de código.

```
IF Item.GET('31260210') THEN
    MESSAGE(STRSUBSTNO('Item name is: %1',Item.Description));

  Item.RESET;
  Item.SETRANGE("No.",FromItem,ToItem);
  Item.FINDLAST;
```

Ilustración 45 – Ejemplo de obtención y filtrado de una variable tipo Record (tabla)

Ilustración 46.1 – Muestra de menú de consulta de carga

Ilustración 46.2 – Muestra de menú de consulta de carga, detalle por familias

Ilustración 46.3 – Pantalla de camión

Cabe destacar de las **ilustraciones 46.1**, **46.2** y **46.3** las funciones no incluidas en aguapp, como son la consulta de carga por camión (función de usuarios super) y más funciones en el menú de Camión como pueden ser las roturas, el cierre parcial y la devolución a depósito.

Limitaciones

Al haber planteado el modelo de trabajo un desarrollo ágil semejante a SCRUM, conforme la aplicación fue avanzando su desarrollo, algunas de las funcionalidades que se plantearon en un principio se dejaron fuera de la implementación final, ya fuera en una etapa temprana o tardía, a beneficio de terminar de implementar otras funcionalidades más importantes, o que se salían del tiempo previsto.

La opción desechada en la etapa más temprana del desarrollo fue la de integrar la aplicación con Microsoft Dynamics NAV. El coste en tiempo y tiempo de desarrollo de crear una instancia de NAV, configurarla a nivel de datos para que sean coherentes y funcione correctamente hubiera ido en detrimento del desarrollo de la aplicación en sí, que es la verdadera 'protagonista' del desarrollo. La alternativa pasó por como ya se ha comentado desarrollar los servicios web en Slim, un Framework orientado principalmente a crear APIs con muy poco código. La implementación de estos servicios y la creación de la base de datos no se acerca al tiempo que se hubiera invertido preparando la instancia de NAV, basándome en casi 4 años de experiencia laboral con Microsoft Dynamics NAV. Además, que el licenciamiento de estos productos Microsoft pudiera haber llegado a ser un problema en caso de imprevistos.

Más adelante se valoró la posibilidad de incluir un módulo de reimpresión de documentos ya emitidos, ya que un caso probable es que, al utilizar comunicación con Bluetooth, la conexión falle y el ticket quede a medias o no llegue a salir. Para evitar este caso concreto aguapp ya cuenta con la opción de reimprimir el ultimo ticket de cada operación, pero solo el último, guardando solo el código ZPL de su impresión, sin un histórico ni persistencia de datos. Ya que todavía estaban por acabar los módulos de la preventa, se optó por terminar estos últimos ya que la correcta creación de un pedido, los pagos y las devoluciones premiaba acabarlos más que un histórico de reimpresión. Esta situación se fue repitiendo hasta considerar que es una función redundante parta aguapp, y que no muestra nada más allá de lo que ya aporta el resto del desarrollo.

Otra funcionalidad desechada fue el seguimiento por GPS. Aunque bien es verdad que ya desde un principio se planteó como un extra, hubiera estado muy interesante verla en funcionamiento, y de hecho en la versión implementada contra Microsoft Dynamics NAV se encuentra en desarrollo. El funcionamiento es sencillo, por un sistema de notificaciones, los dispositivos van devolviendo una respuesta con su notificación, la cual podrá consultarse en un

mapa integrado con la API de Google maps desde el menú de usuario super, para ver la posición de sus repartidores.

De todas maneras, esta aplicación tiene puntos que podrían y deberían mejorarse, y otros que no se han logrado resolver.

Posibles mejoras

Examinando el panorama del desarrollo de aplicaciones a día de hoy, destaca el auge de los Frameworks y entornos de desarrollo multiplataforma. Un claro ejemplo de ello es Ionic Framework, que en 5 años de vida (2013-2017) ha alcanzado ya su 3º versión. Permite desarrollar aplicaciones en código web, que más tarde pueden ser compiladas para distintas plataformas, entre las que están Android, iOS, Windows 10 y Blackberry 10. Cierto es que entre los requisitos de la aplicación está el desarrollarla en Android por ser aguapp aplicación objetivo para terminales de gama media-baja, pero puestos a mirar hacia el futuro, y trasladándonos al caso de una ampliación de la empresa por beneficios a medio-largo plazo, es posible que se deseara renovar el stock de dispositivos, o que haya repartidores que deseen utilizar el suyo propio. Además de que se podría distribuir también como aplicación de escritorio para el módulo de administración y preventa, entendiendo que estas tareas no son necesarias realizarlas siempre durante las visitas al cliente, y que tareas como la creación de pedidos abiertos en preventa o la configuración de la carga del camión para el dia siguiente, son tareas que se pueden realizar en oficina sin necesidad de un dispositivo móvil. Para llevar a cabo esta mejora sería necesario reescribir la aplicación, ya que habría que rehacer todos los elementos gráficos en xml a código HTML, y las funciones reescribirlas en TypeScript. Esta mejora en concreto tiene tal alcance que se puede plantear como un desarrollo nuevo, ya que solo se aprovecharía la parte de los servicios web.

Otra mejora interesante sería reescribir la parte de la impresión desde la aplicación, para utilizar un sistema compatible con Google Cloud Print o similares. Gracias a que Zebra recientemente (2016) renovó su SDK y Firmware de muchos modelos añadiendo, entre otras, funcionalidades Cloud, esta sería la opción optima en caso de querer acometer la mejora. Esto permitiría tener un mayor control sobre las impresiones y herramientas como logs de impresión para resolver posibles problemas de comunicación.

El implementar notificaciones se planteó eventualmente durante el desarrollo de la aplicación, pero se desestimó debido a la poca utilidad que tendría implementarlo. La única utilidad que tendría, dentro de las especificaciones de aguapp, es la de difusión de mensajes y avisos internos, ya que por ejemplo la entrada de nuevas existencias en el almacén no es un dato que tenga que manejar ni tener presente el repartidor.

Problemas no resueltos

Entre los problemas que no se han podido resolver en el desarrollo de aplicación está la operatividad fuera de línea. La aplicación tal y como se planteó, necesita de conectividad constante ya que NAV, el ERP que los servicios de aguapp simulan, trabaja con datos en tiempo real. La creación de un pedido implica una reserva de stock y una disminución del mismo, aun siendo acotado a un camión (almacén) en concreto, por lo que hay proceso de inventariado o informes que, si son lanzados en ese espacio de tiempo que la información no es consistente, darán resultados no realistas. También podría darse el caso de que se produzca un cambio de precios, o de grupo de impuestos en algún cliente o producto, y esto no sería reflejado correctamente al cliente a la hora de realizar la entrega del pedido.

Conclusiones

Las conclusiones obtenidas tras el desarrollo de esta aplicación se han basado en el grado en el que se ha resuelto el problema planteado. Se partía principalmente del punto en el que se quería eliminar medios de escritura manual en las rutas de reparto y agilizar los procesos de venta por medio del desarrollo de esta aplicación, además de la intención de maximizar el ahorro de papel por la impresión de tickets simplificados.

Estas conclusiones se basan también en gran parte en el feedback obtenido por la implantación de la versión de aguapp implementada directamente contra Microsoft Dynamics NAV, como se pudo ver en el apartado de **Despliegue,** ambas aplicaciones cuentan con las mismas funcionalidades base. En términos de usabilidad, la aplicación se ha recibido con buenas críticas, ya que no hay menús ambiguos o que causen confusión, y los elementos activables (botones, listas, etc) se han dispuesto de forma que su uso no provoque conflicto entre ellos. Asimismo, elementos como la gradilla de selección de familias y productos fueron implementados a raíz de las sugerencias recogidas en la primera fase de testeo, ya que en un primer momento se mostraba como una lista extensible por niveles, en comparación con la versión final que muestra botones autogenerados por cada familia y producto, de forma que la selección es mucho más cómoda, sobre todo para usuarios inexpertos.

En cuanto al ahorro de papel, los costes se han visto reducidos, así como el consumo de papel. Un bloc de 50 facturas con copia de carbón tipo talonario preimpresas para rellenar a mano, cuesta 67 céntimos. Esto nos da un coste de 1.34 céntimos por documento expedido, contando con que el tamaño del mismo es de 21x10.5cm y de que hay muchos campos que no se rellenan. Un recambio de rollo de papel para la impresora Zebra iMZ320 se puede conseguir por 45 céntimos si se compra en lotes de 100, y cada rollo tiene una longitud de 20 metros. Se hizo una media de la longitud de la impresión de 100 tickets, todos de operaciones distintas, principalmente recogidos a lo largo de los procesos de desarrollo y prueba, más tickets aportados por la empresa donde se implantó el desarrollo para contar con casos reales, y la media de longitud fue de 9.2 cm. En base a esta longitud media, se obtiene que es posible imprimir 217 tickets de media con un solo rollo, costando 0.20 céntimos (una quinta parte de céntimo) cada ticket. Cabe destacar el ahorro de papel, no solo de coste sino de superficie consumida, reduciendo la superficie de papel consumido por ticket de 21 x 10.5 cm a 4.5 x 9.2 cm (media). En este aspecto se considera que el desarrollo cumple exitosamente con el requerimiento.

Dejando de lado la parte del desarrollo en si, cabe destacar la cantidad abrumadora de conocimientos y experiencia adquiridos durante la realización de todo el trabajo. En esta titulación que ya llega a su fin, por motivos comprensibles apenas hay contenidos orientados a desarrollo de aplicaciones móviles, y el conocimiento con el que puede llegar un alumno al fin de la carrera sobre las aplicaciones móviles puede ser nulo, como fue mi caso propio antes de comenzar mi experiencia laboral y de comenzar este proyecto.

No solo se han adquirido conocimientos en las tecnologías presentadas en este proyecto, sino que el hecho de evaluar otros elementos o metodologías ha desembocado en un enriquecimiento personal, usando más adelante estas tecnologías que se evaluaron como alternativas o mejoras, en proyectos posteriores. En un mercado laboral en el que cada día se solicitan más desarrolladores de aplicaciones, el contar con los conocimientos necesarios para crear un sistema desde el servidor hasta una aplicación multiplataforma sin duda abre muchas puertas de cara al futuro.

Aportando ahora una valoración personal, y contando con que he tenido la oportunidad de implementar una variación de esta aplicación en un entorno de producción real, la satisfacción es enorme al comprobar que los principales escollos que presentaba el planteamiento inicial se han resuelto y con buenos resultados. Durante el proceso de desarrollo de la aplicación se han tenido que hacer frente a infinidad de contratiempos que han tenido que ser abordados desde distintas perspectivas, y buscando soluciones, a veces pasando horas de búsqueda en documentación y foros y llegando incluso a contactar con Zebra directamente por un problema inicial con la impresión. A la conclusión del desarrollo, se ha obtenido un producto que cumple con los requisitos planteado y provee de manera exitosa una alternativa a los métodos tradicionales de venta a domicilio.

En cuanto a trabajo futuro planificado para la aplicación, no hay intención personal de sacar provecho del desarrollo o de continuarlo, ya que el modelo de negocio no está orientado a publicar la aplicación a la venta, sino que se presenta como un desarrollo a medida que cubre necesidades específicas de una empresa en un escenario concreto. Esto implica que su comercialización pasaría por un proceso de desarrollo y personalización a medida por cada cliente, algo no asumible en estos momentos por el alumno.

Documentación

- Cacoo.com – *Diseñador UML online – [online] – [Accedido 18 de Julio de 2017]* https://cacoo.com/
- Moqups.com – *Diseñador de mockups online – [online] – [Accedido 18 de Julio de 2017]* https://moqups.com/
- ERP, Wikipedia – *[online] – [Accedido 18 de Julio de 2017]* https://es.wikipedia.org/wiki/Sistema_de_planificaci%C3%B3n_de_recursos_empresariales
- CRM, Wikipedia – *[online] – [Accedido 18 de Julio de 2017]* https://es.wikipedia.org/wiki/Customer_relationship_management
- Ecured.cu – *Diagrama entidad relación – [online] – [Accedido 18 de Julio de 2017]* https://www.ecured.cu/Diagrama_Entidad_Relaci%C3%B3n
- Manuales IES San Clemente – *[online] – [Accedido 18 de Julio de 2017]* https://manuais.iessanclemente.net/index.php/Introduccion_a_API_REST_y_framework_Slim_de_PHP
- proyectosagiles.org – *SCRUM – [online] – [Accedido 18 de Julio de 2017]* https://proyectosagiles.org/que-es-scrum/
- androidstudiofaqs.com – *MVC en Android – [online] – [Accedido 18 de Julio de 2017]* https://androidstudiofaqs.com/tutoriales/modelo-vista-controlador-en-android-mvc
- phonehouse.com – *Precio de terminales – [online] – [Accedido 18 de Julio de 2017]* https://www.phonehouse.es/movil/apple/iphone-7-32gb.html
- phonehouse.com – *Precio de terminales – [online] – [Accedido 18 de Julio de 2017]* https://www.phonehouse.es/movil/bq/aquaris-u-lite.html#&color=dorado
- StackOverflow.com – *[Accedido 18 de Julio de 2017]* http://www.stackoverflow.com
- developer.android.com - *Documentación Android Studio – [Accedido 18 de Julio de 2017]* https://developer.android.com/studio/intro/index.html?hl=es-419
- slimframework.com – *Documentación Slim Framework – [Accedido 18 de Julio de 2017]* https://www.slimframework.com/docs/
- GitLab.com – *[Accedido 18 de Julio de 2017]* https://gitlab.com/
- Spuceforge.org – *GitExtensions – [Accedido 18 de Julio de 2017]* https://sourceforge.net/projects/gitextensions/
- *Documentación Git Extensions – [Accedido 18 de Julio de 2017]* https://git-extensions-documentation.readthedocs.io/en/latest/git_extensions.html#links

- Zebra.com - *Especificación impresoras – [Accedido 18 de Julio de 2017]* https://www.zebra.com/es/es/products/printers/mobile/mz-series.html
- Zebra.com – *Documentación lenguaje ZPL – [Accedido 18 de Julio de 2017]* https://www.zebra.com/content/dam/zebra/manuals/en-us/software/zpl-zbi2-pm-en.pdf
- ZPL, Wikipedia – *[Accedido 18 de Julio de 2017]* https://en.wikipedia.org/wiki/Zebra_(programming_language)
- Kobjects.org – *Librería kSoap2 – [Accedido 18 de Julio de 2017]* http://kobjects.org/ksoap2/index.html
- Zebra.com – *Documentacion ZSDK – [Accedido 18 de Julio de 2017]* https://km.zebra.com/resources/sites/ZEBRA/content/live/WHITE_PAPERS/0/WH132/en_US/GettingStartedAndroidDevelopment_ZebraAndroid_Link-OS_SDK.pdf
- angrytools.com – *Diseñador de elementos Android online– [Accedido 18 de Julio de 2017]* http://angrytools.com/android/button/
- w3schools.com – *[Accedido 18 de Julio de 2017]* https://www.w3schools.com
- dev.mysql.com - *Documentación mysql – [Accedido 18 de Julio de 2017]* https://dev.mysql.com/doc/refman/5.7/en/join.html
- api.jquery.com – *Documentación jQuery – [Accedido 18 de Julio de 2017]* https://api.jquery.com/
- Bootsnip – *Snippets de código html y css para Bootstrap*https://bootsnipp.com/
- Mozilla Developer – *Documentacion CORS – [Accedido 18 de Julio de 2017]* https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
- AppDesignBook.com – *Principios de testeo de usabilidad – [Accedido 18 de Julio de 2017]* http://appdesignbook.com/es/contenidos/test-de-usabilidad-apps/
- Pruebas de Caja blanca, Wikipedia– *[Accedido 18 de Julio de 2017]* https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca
- LanceTalent.com – *Pasos para establecer el modelo de negocio de una aplicación – [Accedido 18 de Julio de 2017]* https://www.lancetalent.com/blog/modelo-de-negocio-de-una-aplicacion-movil/
- Rollos de Papel online - *[Accedido 18 de Julio de 2017]* https://rollosdepapel-online.com/
- Cartabon.com – *Material de oficina - [Accedido 18 de Julio de 2017]* https://www.cartabon.com

- Styde.net - *[Accedido 18 de Julio de 2017]* https://styde.net/que-es-composer-y-como-usarlo/
- BulkResizePhotos.com – *Redimensionador de imágenes Online - [Accedido 18 de Julio de 2017]* https://bulkresizephotos.com/

Anexos

Esta sección la componen cuatro anexos, el primero de ellos corresponde al Manual de Usuario, el segundo a una extensión de las especificaciones de los casos de uso, el tercero a una lista completa de directorios, subdirectorios y archivos del proyecto de Android Studio, y por último una lista con las versiones de las tecnologías utilizadas en el desarrollo del proyecto.

Se busca con ello extender lo expuesto a lo largo de este documento y facilitar la comprensión, prueba y puesta en funcionamiento de aguapp.

Anexo A – Manual de Usuario

Se pasa a describir los pasos necesarios para hacer **una instalación local** y probar aguapp.

Instalación

Lo primero es, si se carece de ello, la instalación del servidor web. Para evitar problemas de versionado o incompatibilidades no descubiertas, se recomienda simular el mismo entorno sobre el que se ha desarrollado la aplicación.

El servidor web que se recomienda es WampServer, versión 3.0.6:

- http://www.wampserver.com/en/

Una vez instalado, al usarlo de forma local no hace falta configuración adicional, pero se requiere que el dispositivo con aguapp y el equipo que este ejecutando el servidor tienen que estar dentro de la misma red.

El siguiente paso es copiar la carpeta "***codigo\Simaguapp***" de los recursos al directorio *C:\wamp64\www*. Si no se ha modificado los credenciales de conexión a mysql, o el puerto de entrada del servidor, no es necesario modificar ningún archivo, de lo contrario, dentro de aguapp se deberá acceder al archivo *C:\wamp64\www\Slimaguapp\res\mysql-files\mysql-id.php* y modificar los datos de conexión necesarios.

Dentro de la carpeta del proyecto de Android Studio existe un apk listo para instalar en la ruta *"**codigo\aguapp\apk\aguapp.apk**"*. Para instalarla basta con copiarla a un terminal y abrirla con un gestor de ficheros. Al no estar firmada con una cuenta de desarrollador, es necesario habilitar las instalaciones de orígenes desconocidos. Esta opción se accede por el

menú de ***Ajustes -> Seguridad (ilustración 47)*** y buscando por la opción "Orígenes Desconocidos", que habrá que habilitar.

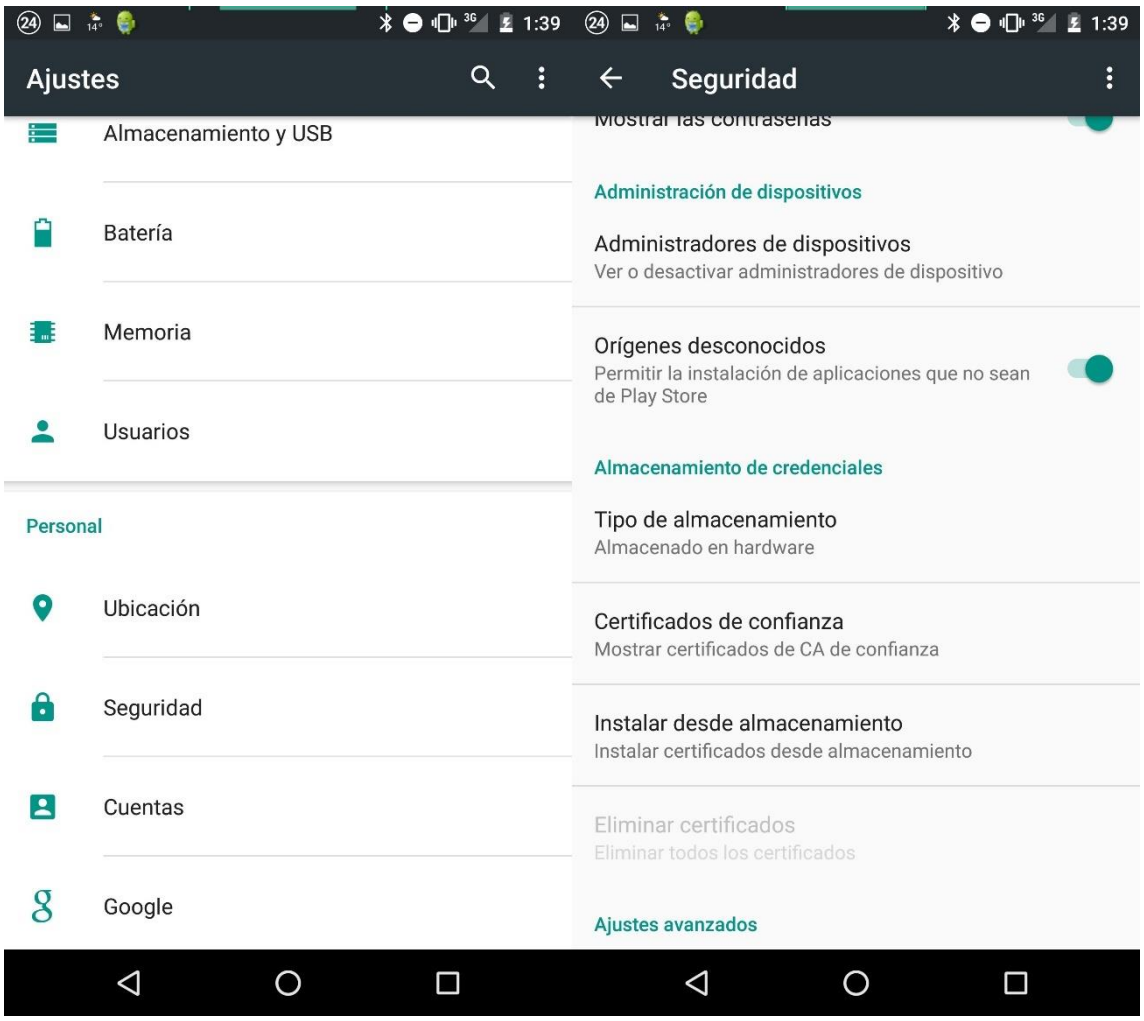

Ilustración 47 – Cómo habilitar "Orígenes Desconocidos"

Por último, es necesario cargar la base de datos. Para ello se deberá acceder a la herramienta phpmyadmin o cualquier gestor de bases de datos MySQL con el que se cuente. En caso de haber instalado Wamp sin modificar sus parámetros, se puede acceder por la ruta http://localhost/phpmyadmin con usuario "root" y contraseña vacía.

Dentro se deberá crear una base de datos con el nombre aguapp como se puede apreciar en la **ilustración 48**

Ilustración 48 – Detalle de creación de base de datos en phpmyadmin

A continuación, con la base de datos creada y seleccionada, debe proceder a importarse los datos base. El archivo que hay que cargar es "codigo/aguapp_starter.sql" de los recursos dentro de la base de datos recién creada. Con los datos cargados, solo queda configurar la app. Accediendo al menú de opciones, en la barra superior derecha de la pantalla de login, representado por un engranaje, se accede a la pantalla de configuración. En el campo de texto de "URL Servicios", como se puede ver en la **ilustración 49** es necesario introducir la ruta en la que están instalados los servicios de Slim. Debe ser una ruta local para la prueba, como la que figura en la imagen.

Ilustración 49 – Detalle de la URL de los servicios en la pantalla de configuración

El panel de control viene dentro del directorio 'Slimaguapp'. Para acceder a su consulta bastara con acceder a la url http://localhost/Slimaguapp/index.html. En caso de haber modificado la configuración de apache (rutas externas que no sean localhost, o puertos que no sean el 80) deberá modificarse la url acordemente. Como se puede ver en la **ilustración 50 y 51** permite consultar una lista de pedidos creados y las existencias por almacén o camión.

Ilustración 50 – Vista de la lista de pedidos

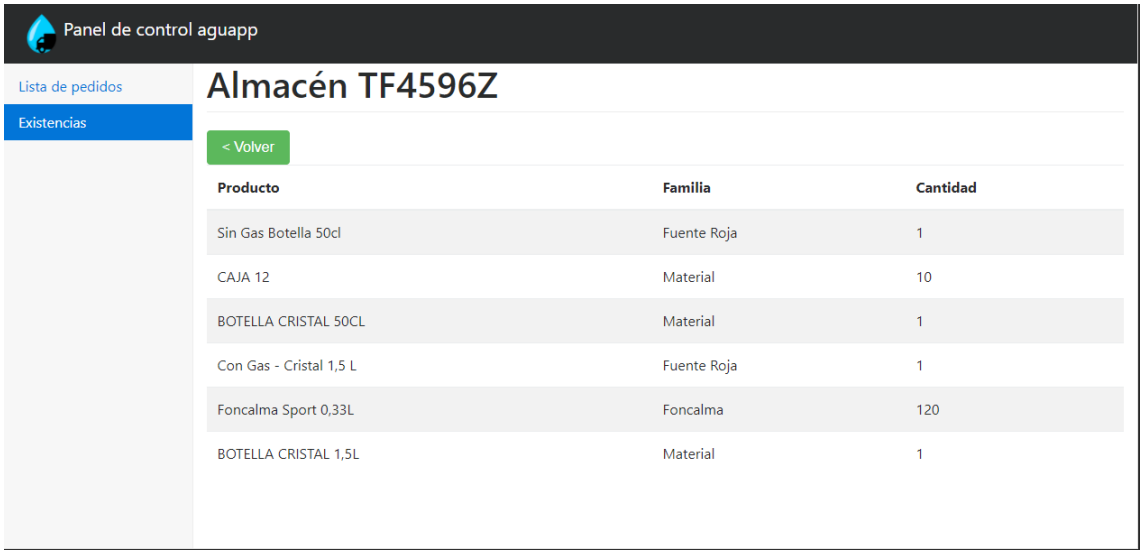

Ilustración 51 – Vista de existencias por almacén

Descripción de las vistas

En la **ilustración 52** se muestra la página inicial de la aplicación. Cabe destacar que a través del engranaje de la barra superior se accede al menú de Configuración de la aplicación. La cuenta predefinida es usuario: u, contraseña: p.

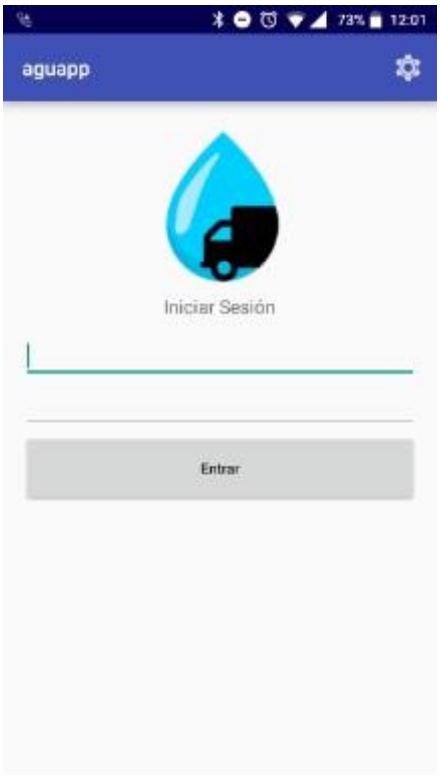

Ilustración 52 – Pantalla de login

Volviendo a la **ilustración 49**, aparte del campo de texto destinado a la URL de los servicios, está el selector de impresoras. Para realizar una búsqueda de impresoras, debe seleccionarse las dos flechas circulares de la barra superior. El proceso comenzará a buscar dispositivos e ira mostrando los encontrados en forma de lista para su selección.

Ilustración 53 – Acciones de usuario super

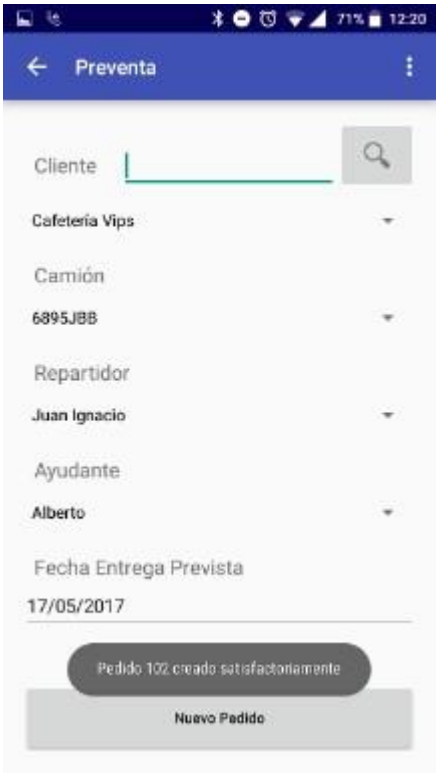

Ilustración 54 – Pantalla de Preventa

Entrando como usuario super se nos presenta el menú de acciones del superusuario, como puede verse en la **ilustración 53**. Este tipo de usuario es el único habilitado para crear pedidos de preventa, o consultar la carga de los camiones. En la **ilustración 54** se aprecia los campos a especificar en la apertura de un pedido de ventas. Debe especificarse un cliente, un camión y un repartidor, la fecha y el ayudante son adicionales.

En las **ilustraciones 55 y 56** se muestra el selector de familias y productos, lo único que se puede hacer en estas páginas, es navegar hacia delante (seleccionando una familia o producto), hacia atrás, o navegando al resumen del pedido.

Siguiendo el flujo de ejecución, pasamos a la **ilustración 57** para ver la pantalla de producto, en donde se introducen las cantidades deseadas. Esta página también tiene un acceso a la página de resumen y también a productos y familias, así como la posibilidad de marcar un producto como artículo de muestreo, lo que quiere decir que entra en el sistema con descuento del 100%.

Ilustración 55 – Selector de familias

Ilustración 56 – Selector de productos

Ilustración 57 – Pantalla de producto

Ilustración 58 – Resumen de pedido

La **ilustración 56** muestra la pantalla de resumen de pedido, en la que se muestra el cliente para quien se realiza y las cantidades deseadas. En caso de tratarse de una carga, en el menú superior, aparte de las notas, se podría especificar la zona de carga del camión, y en vez de mostrarse el cliente, se mostraría la matrícula del camión.

Ilustración 59 – Asignación de conductores a camiones

Ilustración 60 – Asignación de conductores a camiones - Detalle

Retornando al menú de usuario super en la **ilustración 53,** este tipo de usuario puede acceder a las operaciones que haría un repartidor a través del menú 'Camión' (**ilustración 61**). Este menú muestra todas las operaciones que pueden realizarse o solicitarse sobre el camión o sobre su carga. El esquema de todas es parecido, pasa por:

- Selección de la operación
- Selección de la carga que entra/sale
- Comprobación y confirmación del resumen de la operación

Por último, en este menú, se presenta la funcionalidad de la asignación de camiones. En la **ilustración 59** se puede ver la lista con el detalle del conductor, mientras que en la **ilustración 60** se enseña el menú que se despliega al seleccionar uno de los camiones de la lista.

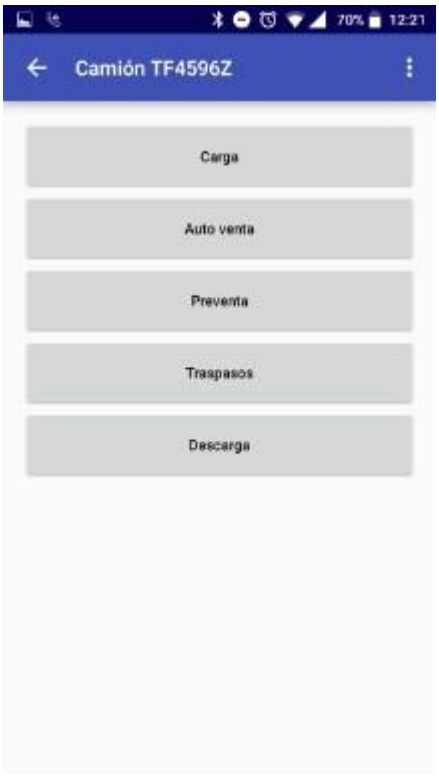

Ilustración 61 – Pantalla de operaciones del camión

Ilustración 62 – Operaciones Autoventa

Para acabar la descripción de las vistas, se presenta la **ilustración 62**, referente a las operaciones de venta sobre la marca. Desde este menú los repartidores pueden crear pedidos nuevos que no hayan sido abiertos en preventa, para acometer las ventas no planificadas del día. Del mismo modo se pueden recibir devoluciones y realizar el pago de pedidos albaranados.

Para seleccionar un cliente se puede especificar un filtro sobre su nombre o sobre su id, en caso de conocerlo.

Anexo B – Lista completa de código java de aguapp

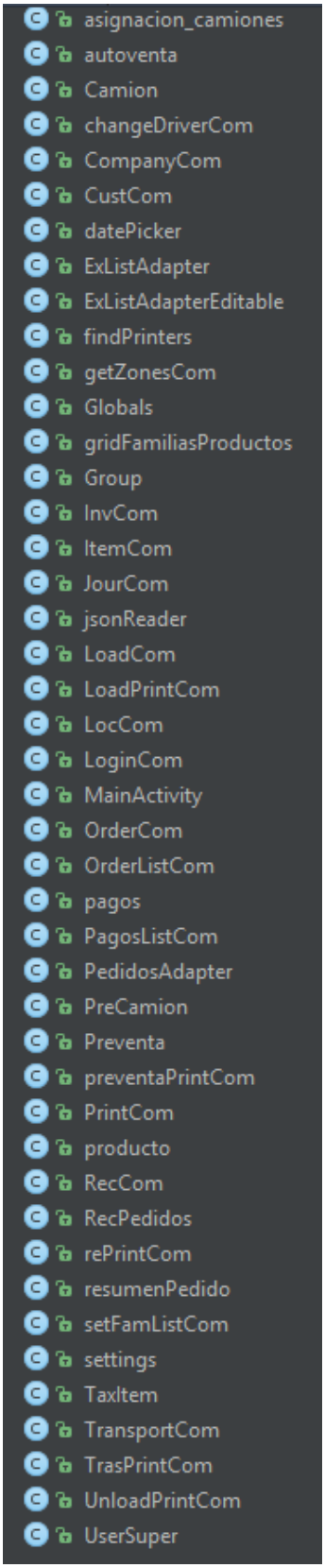

Ilustración 63 – Archivos de código java de aguapp

Agradecimientos

En primer lugar, agradecer enormemente a Abraham Rodríguez Rodríguez por haber aceptado sin vacilar ser mi tutor por la Universidad de Las Palmas de Gran Canaria y haberme acompañado hasta este momento.

También como no agradecer a mi cotutor por parte de mi empleo, Kunal Mahtani Daryanani todo el apoyo y tiempo suministrados durante el desarrollo pesado de parte de la aplicación.

Agradezco también a la Universidad de las Palmas de Gran Canaria y a todos los profesionales que en ella trabajan por darnos la posibilidad de completar nuestras metas y a todas las personas que, aunque no estén relacionadas directamente con este proyecto, hicieron de él una experiencia enriquecedora.

Mención especial a mis padres, Jose Luis y Valentina, por haber hecho posible que este momento se materialice y haberme apoyado en todo momento, y así como a toda mi familia, gracias, gracias por estar ahí.

Por último, gracias a ti Daiana, por haber sido mi mayor apoyo y mi mejor compañía.

Gracias a todos.