UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

APLICACIÓN PARA LA ADMINISTRACIÓN DE PRODUCTOS DEL HOGAR

AUTOR: VICENT PARÍS ANDRÉS

TUTOR: RUBÉN PÉREZ LLORENS

TRABAJO FINAL DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Índice

1. Introducción

La aplicación plantea una solución a una acción cotidiana para todos los hogares. Esta acción se basa en la experiencia propia, donde un miembro de la familia se encargaba de realizar la compra. Para ello, tenía que revisar qué productos había, cuántos y si era necesario comprar más. Otros miembros de la familia apuntaban en una hoja de papel qué productos se habían agotado. Cuando llegaba el día de hacer la compra, el encargado de hacerla preguntaba a los otros miembros si querían algún producto en concreto para comprarlo. Después revisaba qué productos, a parte de los que ya había en la lista de papel, había que comprar, provocando que en ocasiones, cuando no quedaba ninguna unidad de un producto concreto, no se diese cuenta de que este faltaba. Una vez en el supermercado, había momentos en que se terminaba algo en ese momento o alguien se acordaba de que necesitaba un producto concreto y había que llamar o enviar un mensaje al encargado de realizar la compra. También se daba la situación que incluso con la lista de papel se olvidara de comprar algún que otro producto.

Teniendo en cuenta lo anterior, me propuse crear una solución para que todos participaran de forma activa en la confección de la lista de la compra, evitando así todos estos problemas y evitando la pérdida de tiempo debido a ellos.

La solución propuesta ofrece una herramienta de administración de productos del hogar que facilita a los inquilinos una base de datos de productos (de ahora en adelante inventarios) común para todos ellos e incluso la posibilidad de crear nuevos inventarios y de poder compartirlos.

Estos inventarios, que al crearse generan un código único para cada uno, disponen de un apartado de productos con todos los datos que conllevan, así como una lista de compra que se genera automáticamente a partir de la interacción con los usuarios pertenecientes al inventario, para así tener controlado qué productos hay, qué cantidad, y cuando reponerlos. Esto permite a todos los miembros del inventario consultar qué productos tienen, cuáles se han terminado y tienen que ser repuestos.

Para ofrecer la función de compartir inventarios, se ha implementado un sistema simple de usuarios mediante id de usuario y contraseña, además de un servicio en la nube para albergar todos los datos. De este modo los datos se encuentran en alta disponibilidad.

Actualmente no existe una base de datos con todos los productos que los usuarios puedan necesitar, ya que es el propio supermercado el que debería facilitarla, por lo que de momento, los inventarios al ser creados están vacíos y son los propios usuarios los encargados de crear cada producto.

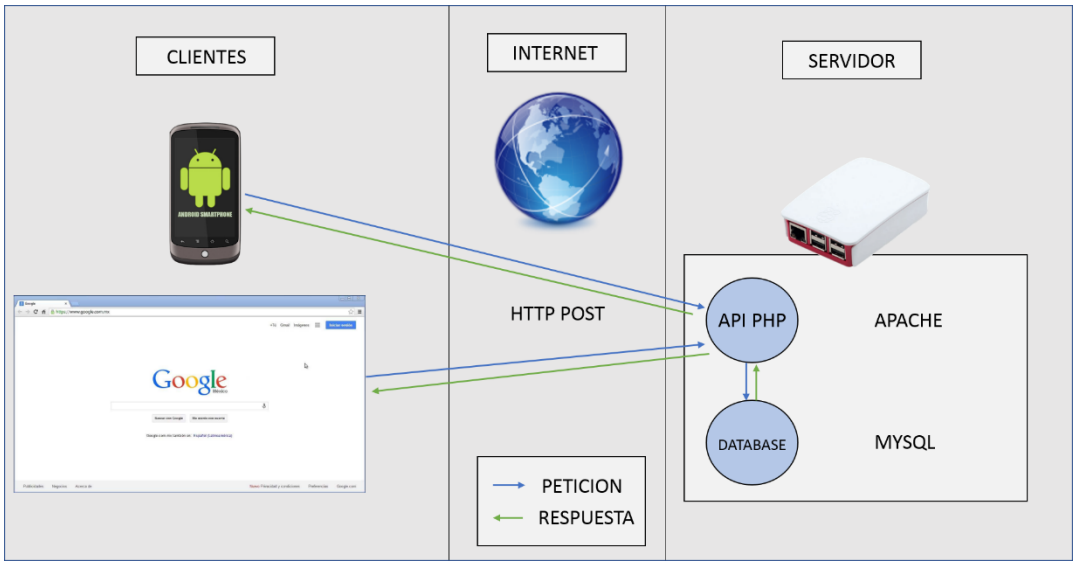
Esta solución se compone de dos clientes, una aplicación para smartphones (Android) y una aplicación Web (Chrome), además de un servidor que gestiona las peticiones de ambas aplicaciones mediante una API común y el almacenamiento de todos los datos.

En cuanto a solución como servidor hardware, se ha optado por una Raspberry Pi 3, la cual funciona con Raspbian, un Sistema operativo basado en Unix (Debian) preparado para sistemas con arquitectura ARM, y se le ha instalado LAMP (Apache, MySQL y PHP).

Como protocolo de comunicación cliente-servidor, se ha elegido HTTP. Todas las peticiones, independientemente de la función a realizar, se hacen mediante el método POST, ya que en todas ellas siempre se manda información; y con este método dicha información no es visible a través de la URL.

Este protocolo nos permite tener un servicio de forma que no se establece una conexión permanente con el servidor, la cual no hay que mantener ni actualizar, simplemente se realiza una petición y se espera su respuesta.

2. Estructura y comunicación del sistema



1 ESQUEMA DE ESTRUCTURA Y COMUNICACIÓN DEL SISTEMA

2.1. Clientes

El cliente es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador, conocido como servidor, normalmente a través de una red de telecomunicaciones.

Los clientes se comunican con el servidor mediante peticiones HTTP POST, en las cuales se envía un paquete de datos en formato JSON. Este formato de datos permite que ambos clientes se comuniquen de idéntica manera con el servidor, pudiendo así implementar una única API que resuelve las peticiones para ambos clientes de forma idéntica; así como implementar nuevos clientes (ej: Apple Iphone) con distintas tecnologías que se conecten a la API sin tener que implementar ninguna modificación o cambio en esta.

2.2. Servidor

Un servidor es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. El servidor recibe las peticiones HTTP POST con los correspondientes datos, y según un parámetro llamado "call", que se encuentra en la URL, sabe qué método debe ejecutar.

En cuanto a la API, se conecta por PDO a la base de datos MySQL. Se ha elegido PDO por su compatibilidad con múltiples bases de datos, lo que permite una migración de MySQL a otro gestor sin mayores complicaciones.

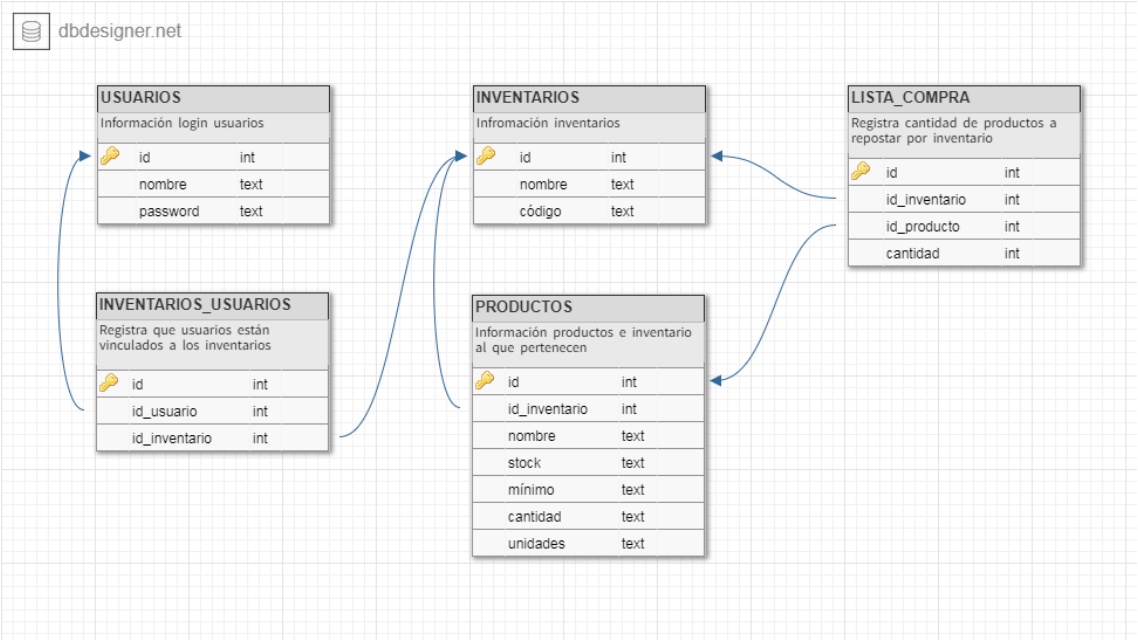
Respecto al hardware del servidor, se trata de una Raspberry Pi 3, la cual se encuentra en la red privada de mi domicilio. Con el problema de la red privada en mente, se ha tenido que buscar una solución de terceros para asignar un DNS a una IP dinámica.

Para ello se ha establecido una redirección de los puertos 80 (http) y 22 (ssh) desde el *router* hasta la Raspberry. Para que esto sea efectivo, se le ha asignado una IP estática a la Raspberry -esta es la 192.168.1.200-. Así mismo, conociendo la IP pública de mi domicilio podemos acceder a la Raspberry.

Llegado a este punto, como la IP pública la proporciona el operador de red, esta puede cambiar. Para dar solución a este problema, se ha utilizado el servicio gratuito proporcionado por www.no-ip.com. En esta web se da un servicio dinámico de DNS y proporciona un subdominio, de manera que instalando su software en la Raspberry, esta se encargara de comprobar si la IP pública cambia.

En el caso de que esta cambie, el software instalado se encarga de notificar al servidor de www.no-ip.com y actualizarle la IP. Esto nos permite asignarle un dominio fijo para que tanto los clientes como el administrador se puedan conectar siempre, sin importar la IP que esta tenga. A partir de este momento el servidor pasa ser http://pantry.sytes.net.

3. Estructura de la base de datos MySQL



2 ESQUEMA BASE DE DATOS

3.1. Tablas

- Usuarios:

Esta tabla recoge los datos de los usuarios para que estos puedan utilizar la aplicación. Para ello se registran mediante usuario y contraseña de manera que estos datos se almacenan para dar acceso al usuario desde cualquier sitio, ya sea a través de la aplicación móvil o de la página web. Para protección de los datos de los usuarios, no se almacena la contraseña como tal, sino un *hash* de esta realizado mediante la función de PHP "password_hash" y se verifica con "password_verify". Esta tabla se puede ampliar con más datos del usuario, incluyendo nuevas columnas para correo electrónico o número de teléfono para futuras actualizaciones de la aplicación.

- Inventarios:

Aquí se almacenan los datos de todos aquellos inventarios que han sido creados por los usuarios. Se establece el id como clave primaria, debido a que la naturaleza de la aplicación es la de compartir inventarios que pueden llegar a tener el mismo nombre. De esta manera surge el problema de tener dos inventarios con el mismo nombre vinculados al mismo usuario, provocando una confusión por no poderlos distinguir. Para este problema no se ha implementado una solución, pero se propone generar distintos marcadores, como colores u otros medios, para facilitar la diferenciación entre ellos. Finalmente se puede añadir una columna con el id del usuario creador para tener más información sobre el inventario.

- Inventarios usuarios:

En esta tabla se guarda un registro de qué usuarios pertenecen a qué inventarios. De manera que esta tabla es una unión de las tablas de usuarios e inventarios con una relación de muchos a muchos, así queda claro qué usuarios tienen acceso a los datos de los inventarios a los cuales están asignados. Esta tabla se puede ampliar en un futuro con la inclusión de una nueva columna para adjudicar acciones que puede realizar un usuario sobre el inventario, estableciendo así una jerarquía de permisos.

- Productos:

Es la tabla donde se guardan todas las propiedades de los productos creados por los usuarios, además del id del inventario al que pertenecen. Los datos de estos son: nombre, “stock”, cantidad, mínimo y unidades. Aquí la cantidad define la suma del producto que se añade a la compra una vez rebasado el mínimo establecido. Además, esta cantidad establecida a 0 indica que el producto no se añade a la compra automáticamente. En cuanto a las unidades, estas se pueden dejar en blanco o indicar en qué se mide este producto, como por ejemplo Botella/s 2L para el agua.

El principal problema, es que esta tabla crece mucho debido a la multiplicidad de los productos, ya que se registran productos idénticos para distintos inventarios. Se ha optado por esta solución ya que carecemos de una fuente de datos de productos, de manera que son los propios usuarios quienes registran los productos a su gusto.

- Lista de compra:

Finalmente, en esta tabla se registran los productos del inventario que necesitan ser repuestos y en qué cantidad. Así pues, se guarda un registro con el id del producto y del inventario, así como la cantidad que se ha de reponer. Con este método, podemos generar una lista de la compra para cada inventario, de manera que esta sea común para todos los usuarios pertenecientes al inventario.

4. Estructura de la API

La API es la parte fundamental del sistema, debido a que es la que contiene todos los métodos que manejan los datos. Esta es llamada mediante un HTTP POST con una URL como la siguiente: http://pantry.sytes.net/pantry.php?call=X, donde la X se sustituye por el número correspondiente a la acción que se quiera realizar (0-19). De esta manera, la API sabe qué tiene que hacer con los parámetros que se le han dado mediante POST, qué función ejecutar y qué respuesta dar. Así es como los clientes hacen peticiones a la API.

Hay hasta un total de 20 operaciones distintas; además de otras funciones que son llamadas por la anteriores 20 como las funciones de cifrado y descifrado, y como las de conexión y desconexión de la base de datos.

- 0. GET_PRODUCTOS:

Esta función, a la cual se le pasa como parámetro el id del inventario, devuelve todos los productos asociados a este con las características necesarias. Estas son el id, la cantidad, el "stock", las unidades y su nombre. La función actúa sobre la tabla de productos.

- 1. GET_INVENTARIOS:

En esta función, mediante el parámetro usuario se devuelve como respuesta el id y el nombre de los inventarios de los cuales el usuario forma parte; y actúa sobre la tabla de inventarios.

- 2. GET_LISTA_COMPRA:

Mediante esta función, con el id del inventario se devuelven el id y nombre de todos aquellos productos a reponer y en qué cantidad, actuando sobre la tabla de la lista de compra.

- 3. GET_INFO_PRODUCTO:

Con esta función, mediante el id del inventario y el del producto, se obtiene toda la información que el usuario puede modificar. Esta es, el nombre, el mínimo, la cantidad y las unidades; y actúa sobre la tabla de productos.

- 4. GET_INFO_INVENTARIO:

Esta función, mediante el id del inventario, devuelve el código único del inventario y actúa sobre la tabla de inventarios.

- 5. LOGIN:

En esta función, mediante el usuario y contraseña que le llegan como parámetros, verifica con la función de PHP *password_verify* que un usuario haya iniciado sesión correctamente, devolviendo "ACCEPTED" en caso de acceder satisfactoriamente o "REJECTED" en caso contrario. Esta actúa sobre la tabla de usuarios.

- 6. APUNTAR:

Esta función es la encargada de añadir productos a la lista de la compra. Para ello se le pasan como parámetros el id del inventario y el del producto, además de la cantidad. Esta función realiza una consulta a la base de datos para obtener la cantidad del producto que ya hay apuntada, y le suma la cantidad que se le ha pasado como parámetro. Actúa sobre la tabla de la lista de la compra.

- 7. COMPRAR:

A esta función se le pasan como parámetros el id del inventario, además del id y la cantidad de los productos a añadir al "stock". Esta función comprueba si se han comprado todos los productos, de manera que los elimina de la lista de la compra. En cambio, en el caso de que no se hayan comprado todos, la actualiza. Por ello, actúa sobre las tablas de productos y la lista de la compra.

- 8. TRASH:

Esta función se encarga de actualizar la cantidad disponible de un producto, de forma que la cantidad final siempre es menor a la inicial; es decir, actuando como una papelera. Para ello, se le pasa como argumento el id del inventario y del producto, además de la cantidad a eliminar. Si la cantidad final es igual o inferior al mínimo establecido, se añade a la lista de la compra la cantidad previamente definida por el usuario. Actuando así sobre las tablas de productos y la lista de la compra.

- 9. ELIMINAR_LISTA:

Con esta función, mediante el id del inventario y el del producto, se encarga de eliminar de la lista de la compra la cantidad que se le pasa como parámetro. Si la cantidad final es igual a 0, el producto se elimina automáticamente de la lista, actuando sobre la tabla de la lista de la compra.

- 10. ACTUALIZAR_PRODUCTO:

Esta función permite el cambio de nombre, mínimo, cantidad y unidades de un producto. El problema es que hay que identificar cuáles han sufrido cambios y cuáles no. Para ello, se ha implementado una variable, la cual según su valor le dice a la función qué debe actualizar.

Su funcionamiento es similar a los permisos en Unix, de modo que esta puede tener valores del 0 al 15. El funcionamiento es el siguiente: a la variable que inicialmente es igual a 0, se le suma 1 en caso de actualizar el nombre; 2 en caso de actualizar el mínimo; 4 en caso de actualizar la cantidad; y 8 en caso de actualizar las unidades. Las combinaciones de los valores anteriores nos dan los valores del 0 al 15, siendo 0 cuando no se ha modificado nada (el cliente no realiza petición al servidor) y 15 cuando se han modificado todos.

Para esto se pasan los parámetros id del inventario y del producto, y el de actualizar; y le siguen los parámetros que varían en función de lo que se actualiza. Estos pueden ser el nombre, el mínimo, la cantidad y las unidades. Como respuesta al cliente, se envía un mensaje confirmando la realización de la acción satisfactoriamente. Esta actúa sobre la tabla de productos.

- 11. NUEVO_PRODUCTO:

En esta función se registra un nuevo producto en el inventario. Para ello, se requiere del id del inventario en el que se va a registrar, además de todos los parámetros a excepción de las unidades, las cuales pueden quedar vacías. Si el usuario marca la casilla de añadir a la lista de la compra, se añade la cantidad introducida a la lista de la compra. Esta función actúa sobre las tablas de productos y la lista de la compra.

- 12. ELIMINAR_PRODUCTO:

Esta función se encarga de la eliminación de un producto del inventario. Para ello, primero elimina los posibles registros de este producto en la lista de compra. Por este motivo requiere del id del inventario y del producto a eliminar y actúa sobre la tabla de productos.

- 13. NUEVO_INVENTARIO:

A esta función se le pasan como parámetros el nombre del nuevo inventario y el usuario que lo crea. Se genera un código único para este inventario en el momento de su creación en la base de datos. Una vez registrado, se obtiene el id del inventario creado en la base de datos y se registra al usuario creador para que tenga acceso a este; de manera que esta función actúa sobre la tabla de inventarios.

- 14. ACTUALIZAR_INVENTARIO:

Mediante esta función se pueden realizar las modificaciones de nombre del inventario. Se le pasa el id del inventario, así como el nuevo nombre como parámetros y actúa sobre la tabla de inventarios.

- 15. ELIMINAR_INVENTARIO:

Esta función elimina el inventario. Para ello, primero elimina la lista de la compra asignada, los productos del inventario, el acceso a los usuarios y finalmente, el inventario. Siendo así, esta función requiere el id del inventario a eliminar y actúa sobre las siguientes tablas: inventarios, lista de la compra, inventarios-usuarios y productos.

- 16. UNIRSE_INVENTARIO:

Esta función es llamada para unir un usuario a un inventario. Para ello, requiere el código del inventario al que se va a unir y el nombre de usuario que se unirá. Con el código del inventario se obtiene el id, y con el id se registra el usuario para otorgarle acceso al inventario; actuando así sobre la tabla de inventarios-usuarios.

- 17. REGISTRO:

Una vez verificada la contraseña por el cliente (se comprueba que se ha introducido correctamente), se llama a esta función, a la cual se le proporciona el nombre y contraseña del nuevo usuario para registrarlo en la base de datos; donde se guardan el nombre y un *hash* de la contraseña. Actúa sobre la tabla de usuarios.

- 18. ABANDONAR_INVENTARIO:

Al pasarle el usuario y el id del inventario que se pretende abandonar como argumentos, esta función elimina el acceso del usuario al inventario, por lo que actúa sobre la tabla de inventarios-usuarios.

- 19. SUMAR:

Esta función es la herramienta mediante la cual el usuario añade nuevas cantidades a un producto sin necesidad de que este haya pasado por la lista de la compra. Para ello necesita tanto del id del inventario, como el id del producto, además de la cantidad a sumar en el "stock" de este. Actuando sobre la tabla de productos.

Para terminar, se incluyen dos componentes de la librería libre llamada "phpseclib". Estos dos componentes dotan de cifrado y descifrado RSA y AES. Respecto a este punto, se encuentra información ampliada más adelante, en el apartado de **Seguridad**.

5. Tecnologías

5.1. Cliente móvil

- Se ha desarrollado una aplicación para el sistema operativo Android. Este sistema ha sido elegido por su libertad de desarrollo y porque su creador -Google-, proporciona una herramienta gratuita llamada Android Studio; además de su amplio uso en todo el mundo. Las aplicaciones para este sistema operativo se desarrollan en el lenguaje de programación JAVA, el cual hemos estudiado durante el grado. El principal inconveniente de este sistema operativo es la fragmentación entre sus distintas versiones, por lo cual la aplicación ha sido diseñada para funcionar en aquellas versiones más recientes a la versión 4.4 (KITKAT) inclusive.

5.2. Cliente web

- Para el cliente web se ha optado por utilizar un *framework* para facilitar el diseño. Bootstrap ha sido el *framework* elegido por sus herramientas para el diseño de estilos y por ofrecer una gran variedad de plantillas de diseño basados en HTML y CSS; además de facilitar un diseño *responsive* para que este se adapte a la pantalla en la que es visualizado.

- Para dar funcionalidad a la página web se ha elegido el lenguaje JAVASCRIPT. Este lenguaje se caracteriza tanto por la implementación de mejoras en la interfaz de usuario, como por dotar a la página web de dinamismo. Esto permite generar el efecto de estar frente a una aplicación en lugar de una web.

5.3. Servidor

- Para la API se ha optado por el lenguaje PHP, sobre un servidor Apache. Este lenguaje es muy utilizado, por lo que tiene una gran cantidad de documentación disponible; además de que su curva de aprendizaje es muy baja. Por otro lado, también ofrece un acceso y administración de base de datos muy sencilla.

- Para la base de datos la elección ha sido MySQL, debido al grado de familiarización con este al haberlo estudiado detenidamente en el grado. La consistencia de los datos frente a posibles errores, ya sean por parte del propio gestor o del sistema operativo en el que corre, y su velocidad, conectividad y seguridad, lo hacen altamente apropiado para su uso en internet. Además, esta tecnología permite mover la base de datos e independizarla en otro servidor, proporcionando así movilidad e independencia.

- Como servidor hardware, la Raspberry Pi 3 -que gracias a la arquitectura ARM ofrece una buena relación en cuanto a rendimiento/consumo de energía se refiere- tiene potencia suficiente como para albergar un servidor Apache y al mismo tiempo un servidor MySQL. Esta permanece conectada 24h/día para ofrecer una alta disponibilidad, tanto del servicio para los clientes, como de los datos.

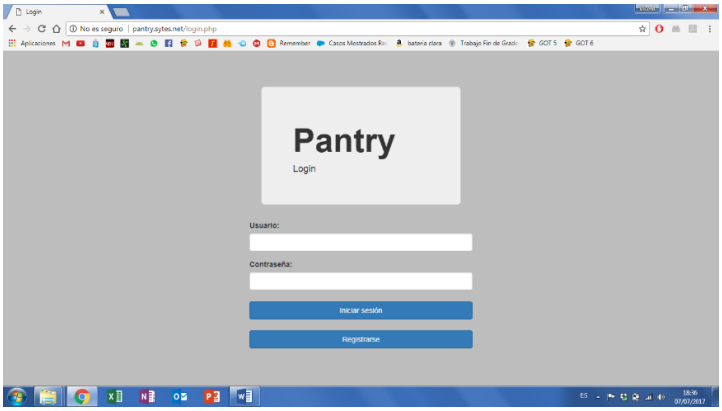
6. Estructura de la aplicación

La aplicación se compone de distintas funciones que juntas actúan como un servicio de administración. A través de los clientes, se llama a estas funciones en el servidor, las cuales manipulan los datos según qué operación ha invocado el usuario. Los clientes son una simple interfaz de la aplicación que permite al usuario interactuar con el sistema. Así pues, procedemos a la exposición de estas funciones.

Se han introducido entre paréntesis las operaciones mencionadas en el apartado **Estructura de la API**, a las cuales los clientes realizan peticiones.

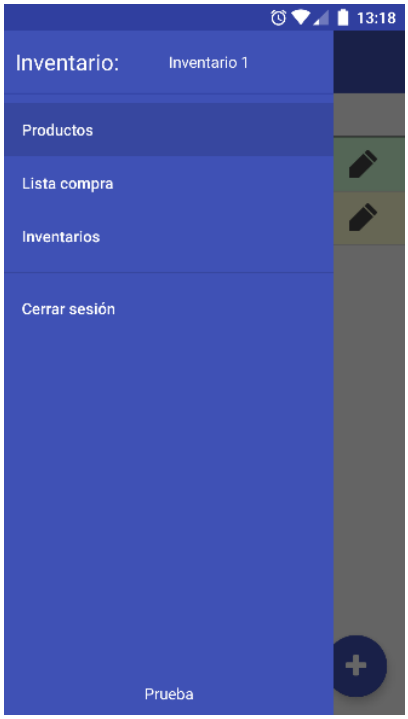
Los clientes, cuando son utilizados por primera vez, presentan una pantalla de inicio de sesión (imágenes 3 y 4) en la cual se les permite acceder a la aplicación (LOGIN) o registrarse para hacer uso de ella (REGISTRO).

3 INICIO DE SESIÓN CLIENTE ANDROID

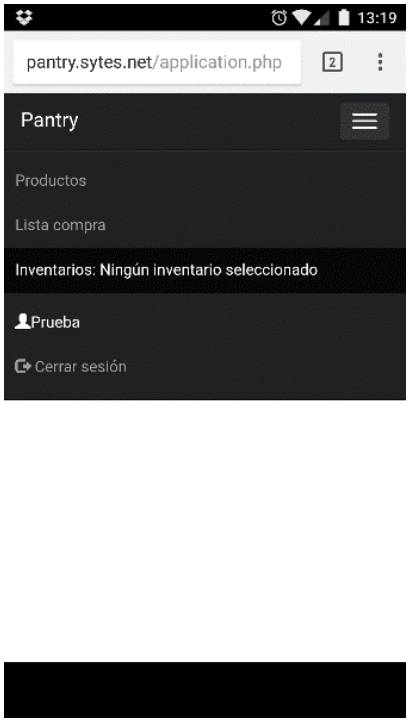


4 INICIO DE SESIÓN CLIENTE WEB

La aplicación se compone de 3 ventanas distintas que se agrupan en un menú lateral en el caso de la aplicación móvil (imagen 5) o en una barra superior en el del cliente web. Además, este último presenta en su versión para pantallas pequeñas un botón en la barra superior que al ser presionado despliega hacia abajo el menú (imagen 6).

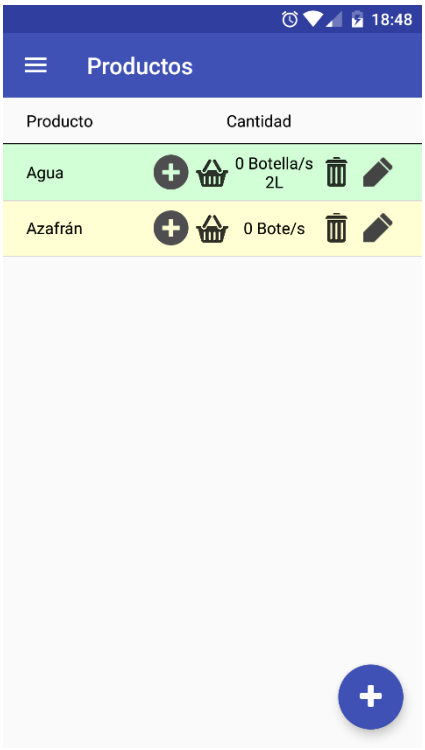


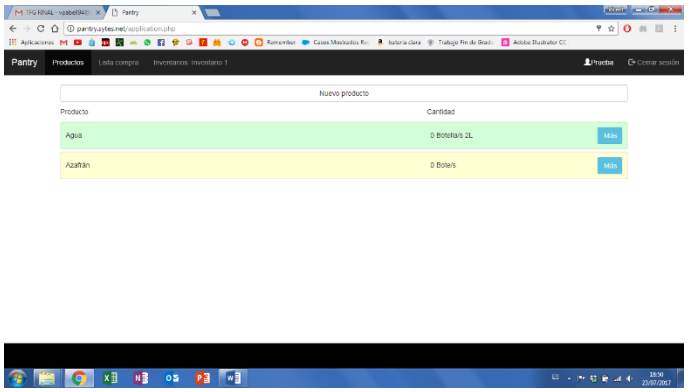
5 MENÚ APLICACIÓN ANDROID



6 MENÚ WEB VISTA MÓVIL

Una vez iniciada la sesión y haber seleccionado un inventario, tanto el cliente móvil, como el cliente web, presentan como pantalla inicial la lista de productos del inventario (imágenes 7 y 8) que se ha elegido previamente, estableciéndolo como predeterminado hasta elegir otro (uno a la vez). En dicha pantalla se presentan todos sus productos (GET_PRODUCTOS) y los diferentes botones para realizar todas las acciones disponibles. De este modo, se tiene un acceso rápido a la lista de productos del inventario para la consulta de su estado, a la vez que permite tener a mano las acciones a realizar sobre los productos

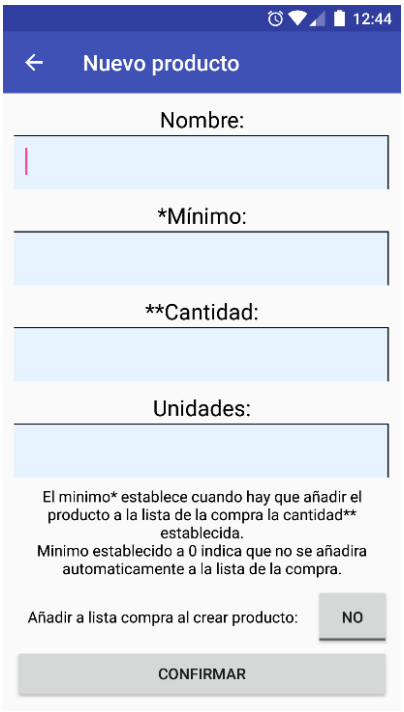


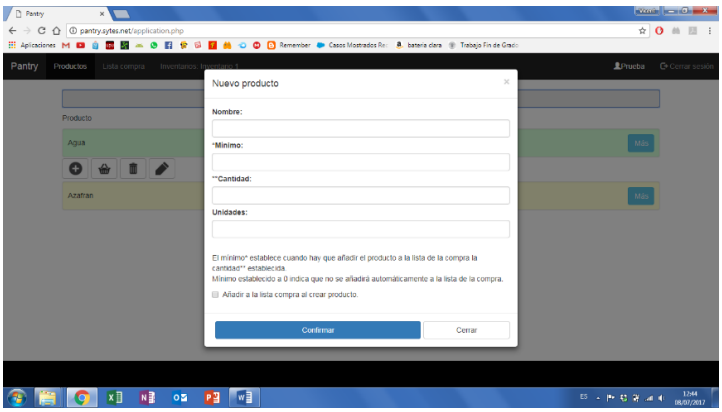


8 PANTALLA DE PRODUCTOS CLIENTE WEB

7 PANTALLA DE PRODUCTOS CLIENTE ANDROID

En esta pantalla, los productos se representan en 2 colores: verde para aquellos que se añaden a la lista de la compra automáticamente y amarillo para aquellos que no lo hacen. Además, presenta botones desde los cuales se realizan las acciones sobre los productos a los que acompañan. En la parte inferior derecha de la (imagen 7) se observa un botón que se corresponde con el de nuevo producto en la (imagen 8). Este abre una nueva pantalla (imagen 9) en el caso de la aplicación móvil y un *pop up* (imagen 10) en el caso de la web. Aquí se presenta un formulario al cual se le introducen las propiedades del producto a crear (NUEVO_PRODUCTO) y ofrece la posibilidad de añadir a la lista de la compra la cantidad establecida.





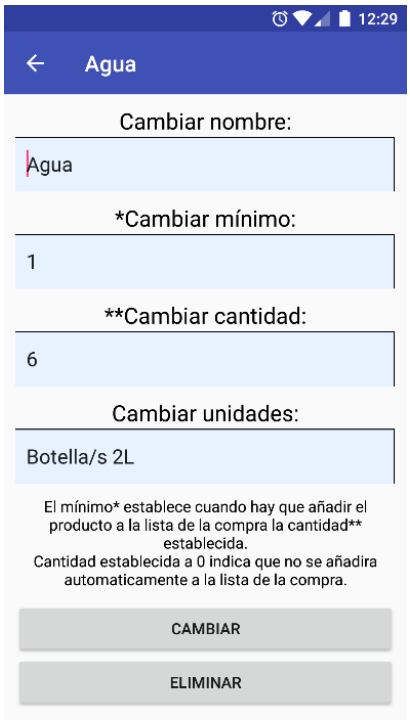
10 PANTALLA NUEVO PRODUCTO CLIENTE WEB

9 PANTALLA NUEVO PRODUCTO CLIENTE ANDROID

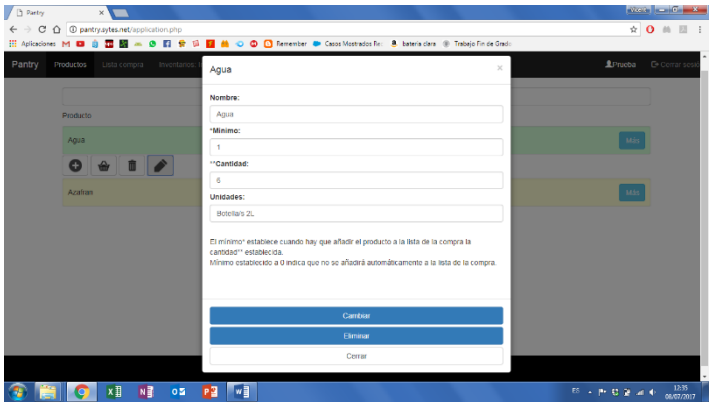
La decisión de incluir el producto -o no- en la lista de la compra recae sobre el usuario, ya que puede incluir un producto que ya se tiene y solo lo está dando de alta; o puede que el usuario quiera introducir un nuevo producto del cual no se tiene ninguna unidad y quiere que se incluya en la lista de la compra.

Los botones ⊕ 🧺 🗑 presentan un *pop up* en el cual se pone la cantidad deseada y al aceptar se realizan las acciones de añadir a productos disponibles (SUMAR), añadir a la lista de la compra (APUNTAR) o disminuir la cantidad del producto (TRASH), respectivamente. Por otro lado, el botón ✎ nos presenta una nueva pantalla en el caso de la aplicación (imagen 11), o un *pop up* en el caso de la web (imagen 12); de manera que, a diferencia de la pantalla de nuevo producto, esta muestra todas las caracteristicas del producto seleccionado, permitiendo su modificación (ACTUALIZAR_PRODUCTO).

La pantalla a la que se hace referencia presenta un botón para eliminar el producto del inventario (ELIMINAR_PRODUCTO). Esta accion tiene como consecuencia la eliminación del producto y de todas sus características; al mismo tiempo que se elimina de la lista de la compra en caso de estar apuntado.



11 PANTALLA EDICIÓN PRODUCTO CLIENTE ANDROID

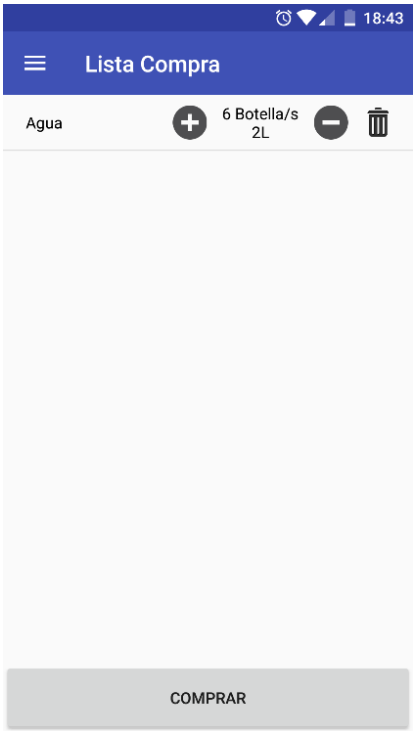


12 PANTALLA EDICIÓN PRODUCTO CLIENTE WEB

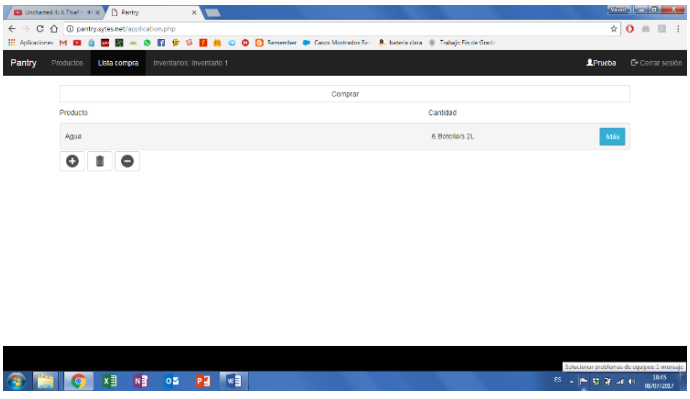
Según el orden de aparición en el menú de la aplicación, la siguiente pantalla es la de la lista de la compra (imágenes 13 y 14), en la que se encuentran aquellos productos con la cantidad que ha de ser repuesta (GET_LISTA_COMPRA). En esta pantalla se presenta un botón principal para confirmar la realización de la compra (COMPRAR). Se espera que el usuario utilice este botón cuando haya obtenido -al menos- uno de los productos de la lista, sumándolo al "stock" y eliminándolo de la lista de compra o actualizando la cantidad a reponer.

En cada producto se presentan 3 botones. Los botones suman o restan cantidad del producto en el momento de la compra, permitiendo al usuario tener la libertad de coger más o menos cantidad sin tener que navegar por la aplicación para modificarla. Este cambio de cantidad no se refleja en el servidor, por lo que no hay comunicación. Por ello, las modificaciones realizadas con estos botones se perderán al actualizar la pantalla. Esto significa que los botones no actúan sobre la lista de la compra como tal, sino que modifican la cantidad final obtenida de un producto en el momento de realizar la compra (al pulsar el botón de compra). De esta forma se evitan errores de concurrencia cuando dos usuarios realizan cambios al mismo tiempo. Cuando se pulsa el botón de comprar, los clientes recogen los productos con cantidades superiores a 0, y los suman a los productos disponibles, de modo que si se ha comprado menos cantidad de un producto de la que había originalmente, permanecerá la diferencia en la lista de la compra. Por ejemplo, si había 6 botellas de agua y se le indica a la aplicación que se han obtenido 4, en la lista de la compra permanecerán 2.

Finalmente, el botón despliega un *pop up* al cual se le introduce la cantidad que ya no se requiere, eliminando de la lista la cantidad introducida (ELIMINAR_LISTA_COMPRA). Si el producto queda con cantidad 0, se elimina automáticamente de la lista.



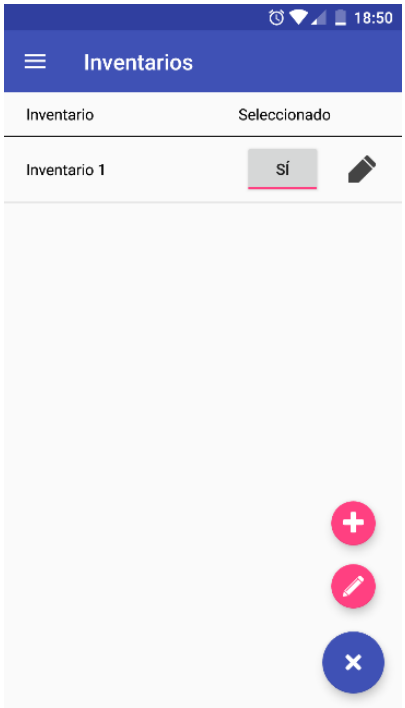
13 PANTALLA LISTA COMPRA CLIENTE ANDROID



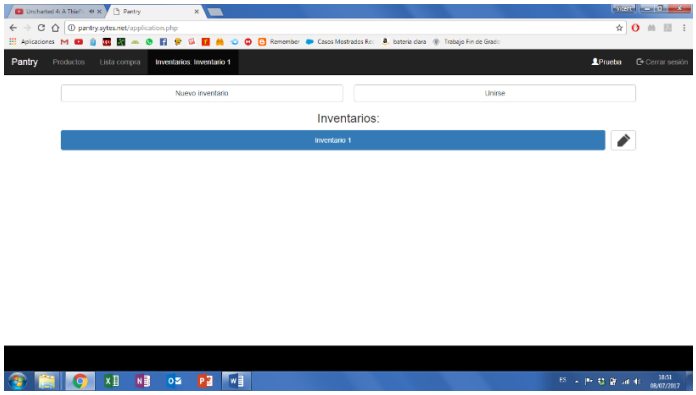
14 PANTALLA LISTA COMPRA CLIENTE WEB

Por último, tenemos la pantalla de inventarios (imágenes 15 y 16). Cuando los usuarios se registran o inician sesión por primera vez, se les presenta esta pantalla en la que aparecen los inventarios a los cuales el usuario tiene acceso (GET_INVENTARIOS). Además, hay un indicador de cuál de ellos está predeterminado, junto con la opción de cambiarlos y editarlos de manera rápida y sencilla (ACTUALIZAR_INVENTARIO). También se ofrece la posibilidad de crear un nuevo inventario (NUEVO INVENTARIO) o de unirse a uno existente (al cual todavía no se tenga acceso) mediante su código (UNIRSE_INVENTARIO).

Cuando haya almenos un inventario en su lista, deben selecionarlo para disponer de todas las funciones sobre él, ya sea tanto la lista de productos y la modificación de estos, como la lista de la compra vinculada a este inventario.

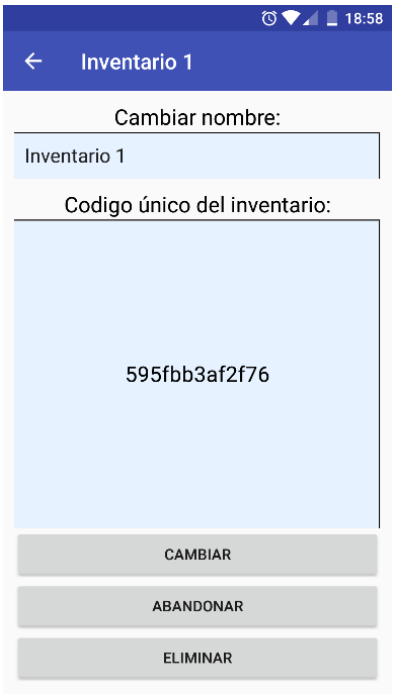


15 PANTALLA INVENTARIOS CLIENTE ANDROID

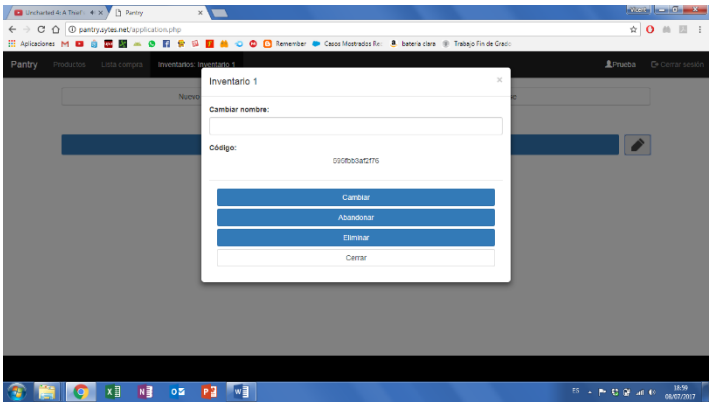


16 PANTALLA INVENTARIOS CLIENTE WEB

El botón de edición abre una nueva pantalla con la información del inventario (imágenes 17 y 18), permitiendo la modificación de su nombre y la visualización de su código, el cual es necesario para que otros usuarios puedan unirse.

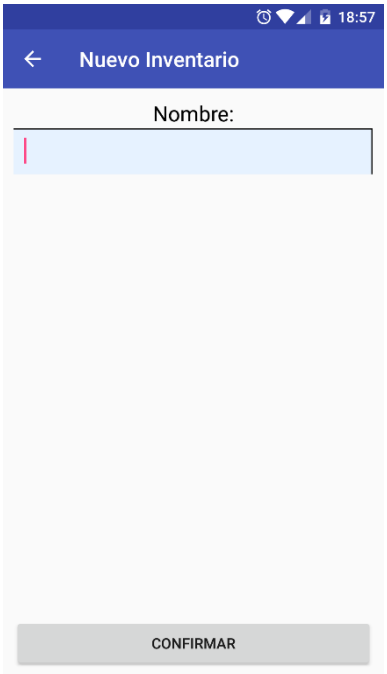


17 PANTALLA EDICIÓN INVENTARIO CLIENTE ANDROID

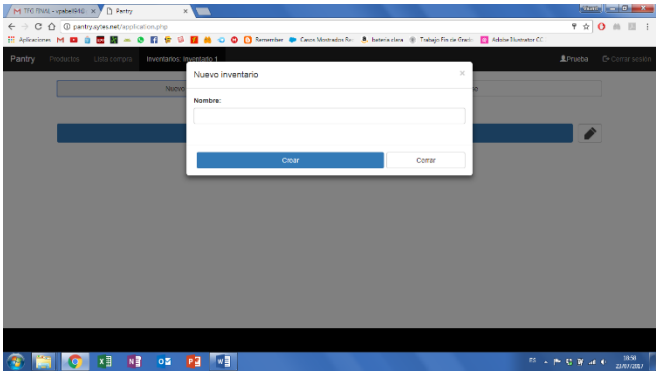


18 PANTALLA EDICIÓN INVENTARIO CLIENTE WEB

Esta pantalla también dispone de la función de abandono del inventario por parte del usuario (ABANDONAR_INVENTARIO), de modo que no afecte a otros usuarios y pudiendo siempre volver a unirse mediante el código del inventario -a excepción de si el último usuario abandona el inventario, pues este queda eliminado-. Por último, se ofrece la posibilidad de eliminar el inventario (ELIMINAR_INVENTARIO). Esto lo elimina, junto con toda su información, permanentemente de la base de datos.



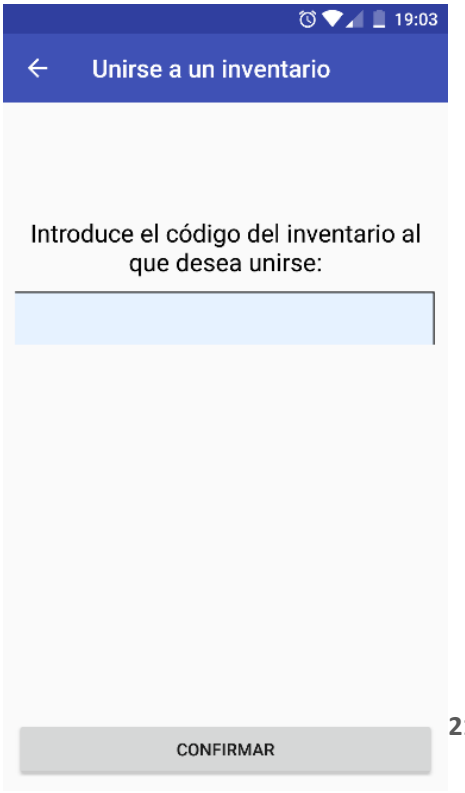
19 PANTALLA NUEVO INVENTARIO CLIENTE ANDROID



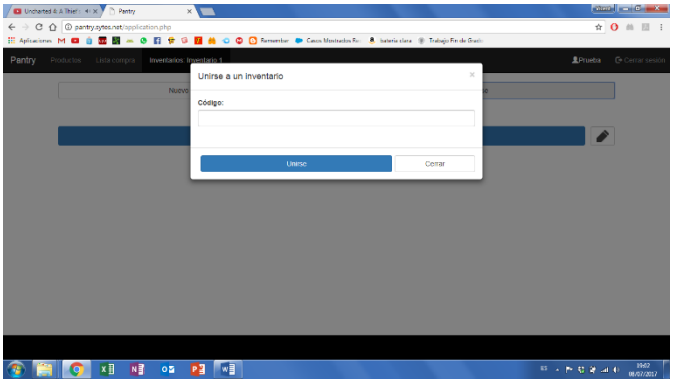
20 PANTALLA NUEVO INVENTARIO CLIENTE WEB

El botón ⊕ abre una nueva pantalla (imágenes 19 y 20), donde se pide el nombre para el nuevo usuario y presenta el botón para confirmar su creación.

Como función final, el botón de unión ✎ abre una nueva pantalla en el cliente Android (imagen 21) o un *pop up* en el cliente web (imagen 22), en la cual se le pide al usuario el código del inventario al que se quiere unir.



21 PANTALLA UNIÓN INVENTARIO CLIENTE ANDROID



22 PANTALLA UNIÓN INVENTARIO CLIENTE WEB

7. Desarrollo y gestión del tiempo

En febrero empieza el desarrollo de este proyecto, el cual finaliza la primera semana de julio. Y se emplea el resto del mes para la redacción de la memoria. Con esto en mente, se empieza el proyecto por las partes que más conocimiento se tiene (Android, MySQL) y la API (PHP) que es la parte esencial para el servicio.

7.1. Desarrollo

Se empieza con la investigación de las tecnologías más utilizadas en la actualidad, las cuáles son las más novedosas; además de aprender que ofrecen. Se empleó todo el mes de febrero para determinar qué tecnologías implementar.

Se ha estudiado qué arquitectura utilizar para la base del proyecto. Se ha investigado la tecnología de sockets que permiten mantener conexiones abiertas contra el servidor, que ofrecen la posibilidad de monitorizar los cambios en los datos en tiempo real. La desventaja de esta tecnología es el mayor consumo de recursos -ya que la conexión permanece abierta hasta que el cliente la cierra-. Además, esta tecnología varía entre lenguajes de programación, ya que para las aplicaciones web se utiliza una variante de los *sockets* que funciona sobre HTTP, por lo que se tendrían que implementar APIs distintas para los distintos clientes.

Una vez descartados los *sokets*, se ha estudiado la viabilidad de implementar un servicio RESTful. Este, aunque no permite mantener una conexión abierta entre el cliente y el servidor -ya que esta se cierra después de cada petición-, permite realizar peticiones HTTP que devuelven una respuesta a los clientes. Con esto se pierde la visualización de los inventarios en tiempo real, pero a causa del poco tiempo que los usuarios necesitan invertir en la aplicación -lo cual implica una posibilidad muy pequeña de que dos usuarios interactúen a la vez-, se ha optado por la gestión de concurrencia desde la API para que los cambios de un usuario no perjudiquen a otro.

RESTful se compone de diferentes peticiones, cada una de ellas se utiliza para implementar una función específica:

- GET: Se utiliza para acceder a un recurso, de modo que pide datos al servidor y estos los devuelve sin hacer ninguna modificación sobre estos. Esta función requiere de parámetros que se envían a través de la URI.

- POST: Envía datos para crear un recurso. Los datos se envían a través del cuerpo de la petición y no en la URI como en las peticiones GET.

- PUT: Similar a POST, pero esta se emplea para la actualización o modificación de un recurso.

- DELETE: En inglés eliminar, sirve para la eliminación de un recurso.

- PATCH: Esta sirve para la modificación parcial de un recurso, aunque su utilización es muy poco frecuente en favor a PUT.

Finalmente se ha decidido implementar un servicio basado en RESTfull, el cual utilizando exclusivamente su petición POST -que es la petición más segura-, mediante la cual se le indica a la API qué tiene que hacer; además de enviarle los datos necesarios. Así, la API devuelve una respuesta según la función realizada. De modo que según un parámetro que se ha llamado c*all* -a través de la URI-, indica que funciones de las anteriores debe realizar; además de proporcionarle los datos necesarios para llevaras a cabo.

Una vez aquí, se han investigado los lenguajes de programación para elegir en cual desarrollar la API. El primer lenguaje investigado y testeado fue Scala, es un lenguaje relativamente nuevo basado en java al cual plataformas como Twitter que ha migrado desde Python. Además, debido a que se ha utilizado Java en casi todo lo referente a programación durante la carrera, esta familiaridad puede ayudar con la curva de aprendizaje en Scala. Finalmente, se descartó por su complejidad de despliegue además de poco soporte gratuito para su desarrollo.

La siguiente opción es PHP, este lenguaje es el más utilizado para este tipo de servicios; además de no requerir ninguna licencia. PHP tiene una documentación muy amplia, es multiplataforma, cuenta con una buena seguridad -debido a que se ejecuta en el lado servidor-, y además su curva de aprendizaje es relativamente baja. Es una buena opción para empezar el desarrollo de la API.

En cuanto al cliente, se ha elegido Android por la familiaridad que ya se tiene; además del interés propio por empezar el desarrollo en esta plataforma.

Por otra parte, para el cliente web, se ha investigado directamente sobre el lenguaje Javascript -el cual ha sido introducido en la asignatura de Desarrollo Web-, con las ventajas que presenta, como generar una web dinámica que permite implementar una "aplicación" web. Este lenguaje puede ser introducido directamente sobre el fichero HTML o introducido como anexo, y es el encargado de modificar el código HTML y de implementar las peticiones para la interacción con el servidor.

Finalmente, como gestor de bases de datos, se ha elegido MySQL por el propio conocimiento que se tiene, ya que ha sido el que se ha estudiado durante toda la carrera.

Llegado este momento, se ha empezado el desarrollo basado en el modelo en espiral, debido a la novedad del proyecto. El modelo en espiral se compone de distintos bucles o iteraciones, cada uno de ellos formados por distintas actividades que generan una función de la aplicación. Así, en cada iteración, la aplicación cuenta con una nueva función al mismo tiempo que ofrece la posibilidad de empezar a utilizar el servicio una vez implementadas las funciones principales. Con este método, las pruebas por terceros pueden empezar, permitiendo las modificaciones, correcciones y mejoras en las funciones ya implementadas; además de dirigir el desarrollo del resto de funciones basándose en el *feedback* recibido.

En cuanto al servidor (Raspberry Pi) no se obtuvo hasta alcanzar la suficiente madurez del proyecto como para desplegar el servicio 24 horas todos los días.

Hasta alcanzar esta madurez, se ha utilizado un programa que facilita la puesta en marcha y detención del servicio rápidamente cada vez que se sigue el desarrollo. Este programa llamado Xampp, ofrece los servicios de Apache y MySQL en equipos Windows. De modo que no se ha instalado ninguna máquina virtual ni ha sido necesaria la instalación de un sistema operativo Unix.

Para aquellos días que el ordenador principal no está disponible, existe la posibilidad de una rápida migración a otros ordenadores mediante la instalación de Xampp en el equipo nuevo. Así, el desarrollo de la aplicación no se detiene en ningún momento permitiendo movilidad y flexibilidad; evitando la pérdida de tiempo de desarrollo.

7.2. Planificación

Con todo esto en mente, se estudia el servicio que se va a implementar para generar la base de datos e introducir algunos datos para realizar pruebas y así definir todas las funciones que se han de implementar.

Una vez la base ha sido creada y los métodos han sido definidos, se empieza el desarrollo de la API para que tenga acceso a la base de datos y poder presentar estos por peticiones POST.

Para el testeo de estas peticiones -necesario por el desconocimiento del lenguaje PHP-, se empieza el desarrollo de la aplicación Android al mismo tiempo, de modo que se implementa la comunicación cliente-servidor para testear y comprobar su correcto funcionamiento.

Con esto en marcha y funcionando, cada semana se implementan de dos a tres funciones, dependiendo de su complejidad. Se fija la media de dos funciones para que las más complejas abarquen el tiempo de aquellas poco complejas. Así se termina el desarrollo de la API y la aplicación a mediados de mayo.

En este punto se establece un período de dos semanas para el testeo del servicio, para el que se ha pedido ayuda a familiares y amigos para encontrar los posibles errores cuanto antes y poder corregirlos.

Con todos los errores solucionados a finales de mayo, se empieza con el desarrollo de la web, al cual se le asigna la duración de un mes debido a la poca experiencia sobre Javascript. Esto deriva en que la web está lista en dos semanas.

Con 2 semanas libres se opta por la implementación de la encriptación de las comunicaciones cliente-servidor. Se delimita una semana por cliente por lo que se realiza la encriptación del cliente Android, y de la API en una semana, y en otra para la encriptación de la web, en la cual se tienen problemas con la librería, ya que está preparada para funcionar en un sistema, pero se necesita que funcione en el navegador. Para ello se instala una máquina virtual con VirtualBox con Linux Ubuntu y desde esta se genera el fichero Javascript que se incluye en el navegador.

8. Seguridad

La seguridad es un aspecto muy importante cuando se trata de la manipulación de información personal de los usuarios.

8.1. Servidor

Para proteger la información que se almacena de los usuarios, se ha restringido el acceso a la base de datos a la que solo tiene acceso el administrador (root) y un usuario especialmente habilitado para nuestra API.

Se capacita a la API para acceder a la base de datos mediante el usuario habilitado para ella, de modo que debe conocer tanto el nombre como su contraseña, por lo que estos datos de usuario quedan registrados dentro de la propia API y con el fin de mantener estos datos seguros, se modifican permisos Unix. Así, se restringe el acceso a los ficheros y carpetas que conforman tanto la API como la web. Estos permisos se establecen con "chmod 740" para archivos, y "chmod 710" para carpetas.

Los valores que acompañan a "chmod" representan el nivel de permisos que tienen los distintos usuarios del sistema. Agrupados en 3 categorías, el primer número hace referencia a los permisos del creador -en este caso es "root"-, de manera que tiene los permisos de lectura, escritura y ejecución de los archivos y carpetas. El segundo número, establece los permisos para los usuarios pertenecientes al grupo asignado, esto otorga acceso de lectura a todos los ficheros, así como permiso de paso en carpetas (permiso de ejecución en carpetas implica la navegación a través de estas). Previamente, se ha cambiado el grupo al de Apache, a todo lo que hay dentro del directorio /var/www/html para que tenga acceso a estos ficheros y carpetas con los permisos descritos. También se establece esta configuración mediante el comando "chown -R root:www-data *". Y el tercer número es para los permisos del resto de usuarios (0 indica ningún permiso).

Lo anterior nos defiende contra la navegación por el sistema de archivos, pero no contra la visualización de los ficheros si se sabe la ubicación exacta. Para evitar esto, se ha creado una configuración llamada ".htaccess" a la que se le indican las extensiones de los ficheros que no se deben mostrar vía web.

Si no se aplican estos permisos ni configuraciones, los ficheros pueden ser vistos desde el navegador web como si de un explorador de archivos se tratase. De este modo se evita el acceso a los PHP dentro de la misma máquina por otros usuarios y además se evita la visualización del resto de ficheros, de manera que los visitantes se encuentran con un aviso de prohibido el acceso (imagen 23) si se intenta acceder.

23 INTENTO DE ACCESO POR NAVEGADOR A LA CARPETA /CSS

Para la administración y gestión del servidor, se ha habilitado un acceso remoto vía *Secure Shell* (ssh), al cual se le restringe el acceso a "root". Además, el usuario "pi" que viene por defecto, se elimina de "sudoers" y se le restringe el uso de super usuario mediante contraseña.

8.2. Contraseñas y encriptación

- Contraseñas:

La seguridad implementada anteriormente no es suficiente, ya que también se han de proteger las comunicaciones cliente-servidor y a los usuarios en caso de sufrir un robo de datos. Para ello se ha optado por el almacenamiento de un *hash* de la contraseña, de manera que la contraseña llega a la API y es la API la encargada de verificar si es la misma que originó el *hash* almacenado. Así, en caso de robo de datos, los usuarios quedan protegidos debido a que las contraseñas no quedan registradas en la base de datos.

- Encriptación:

Para la protección de las comunicaciones, se intenta implementar una encriptación simétrica, la cual no es viable debido a que cualquier persona con un conocimiento básico de web o Android pueden obtenerla fácilmente. Con lo cual, se implementa una encriptación asimétrica, para la que se generan un par de llaves (pública y privada), así, la API hace uso de la llave privada para desencriptar los mensajes de los clientes, los cuales han sido encriptados previamente con la llave pública.

Aquí surge un problema, la encriptación asimétrica tiene una limitación en cuanto al tamaño de los mensajes se refiere. De modo que se ha buscado una manera de reducir su tamaño para que puedan ser encriptados. Finalmente se genera una llave simétrica, con la cual se encriptan los datos y esta se envía con ellos al servidor. Para que esta llave no sea interceptada por el camino, la llave simétrica es encriptada con la llave pública antes mencionada, así la API con su llave privada, es la única capaz de desencriptar la llave simétrica y utilizarla para acceder al mensaje. Una vez la API termina la resolución de la petición del cliente, manda la respuesta de vuelta, cifrándola únicamente con la llave simétrica que el cliente ya conoce.

Con esto se ha resuelto la seguridad en las comunicaciones cliente-servidor de manera notable, aunque la generación de la llave simétrica se realiza en cada petición al servidor, y esto significa un mayor consumo de recursos del sistema.

Para la implementación de la encriptación, se utilizan 3 lenguajes distintos de manera que trabajen de manera uniforme.

1. Cliente Android:

En este cliente tanto la encriptación, como la generación de llaves es bastante sencilla, ya que el propio sistema proporciona las funciones necesarias.

2. Cliente web:

Se ha realizado una búsqueda de las mejores librerías disponibles para la encriptación tanto simétrica como asimétrica, y finalmente se implementa la librería llamada "node-forge" de código abierto a partir de la cual se han de generar los *scripts* que proporcionan tal funcionalidad. Esta librería está acompañada por una buena documentación; además de ofrecer una gran variedad de encriptaciones.

3. API:

Para la API se ha optado por la librería "phpseclib", que es una librería muy fácil de implementar y con las características que se necesitan.

9. Futuras mejoras

En este apartado se refleja la idea original que se tiene en un principio, la cual por limitaciones personales y temporales no puede ser llevada a cabo. Por lo tanto, la aplicación presentada es solo una pequeña parte de un proyecto mayor, de manera que solamente implementa algunas de sus funcionalidades.

La primera mejora y más importante, es la de poseer una base de datos de productos para que los usuarios accedan a estos y no tener que crearlos. Para ello hay que registrar producto a producto todas sus características como ingredientes, cantidad, peso, código de barras, precio, alérgenos, etc. además de una imagen de estos. El mejor modo de conseguir estos datos es conectándose mediante una API a los supermercados. Estos deben tener disponible todos los datos sobre los productos, así como las posibles actualizaciones, e incluso precios.

Otra mejora es ofrecer a los usuarios de un mecanismo de distinción de inventarios con el mismo nombre, ya sea mediante colores u otros medios.

La siguiente mejora consiste en la implementación de *sockets* como protocolo de conexión entre clientes y servidor para tener una vista en tiempo real de las operaciones (o algunas de ellas) que están haciendo unos usuarios, mientras otros ven el contenido del inventario.

También se puede mejorar la manera de indicar a la aplicación qué productos se han usado. Se podría implementar un acceso directo a un escaneo de código de barras desde el móvil o introducción de su número de identificación en la web. Para mayor comodidad, se puede desarrollar un lector de código de barras que se coloca cerca de las bolsas de basura, de esta manera los usuarios ya no requieren de sus móviles ni ordenadores para realizar esta acción que es la que más rechazo les genera.

Con lo anterior mencionado, se puede empezar a realizar compras directamente desde la aplicación. Se puede implementar distintos métodos de pagos, y recoger la compra en el supermercado elegido a la hora indicada por el usuario. Además, se puede crear un registro de cuándo y quién ha realizado la compra, y un apartado del histórico de facturas de los últimos 6 meses.

El siguiente paso es la implementación del reparto a domicilio que, mediante la implementación de un sistema de mapas, es el supermercado más cercano el que se encargue del pedido.

Se puede implementar una cola de productos que están cercanos a la fecha de caducidad y su notificación correspondiente. Para esto hay que modificar la base de datos; además de tener que cambiar de códigos de barras a códigos bidimensionales, ya que los segundos ofrecen más capacidad de almacenamiento de la información.

También se puede introducir una cola de productos que no se utilizan frecuentemente, es decir, aquellos que se adquieren para ocasiones especiales. A estos se les pueden asignar distintas prioridades; además de comentarios del usuario que los agregue a la cola. De esta manera se puede introducir una compra de productos urgentes, que se preparen y entreguen rápidamente. Y, por otro lado, otros no tan urgentes para que los usuarios los incluyan en la próxima compra.

Y finalmente, la idea original del proyecto incluye la introducción de un calendario, donde se puede programar compras periódicas, es decir, se puede establecer que se realice la compra de forma automática -por ejemplo, todos los viernes a la misma hora-, con una confirmación del propio usuario para asegurar que estará presente en el domicilio a la hora de la entrega.

Como extra, se puede introducir un apartado en los clientes de manera que, a través de estos se presentaran los nuevos productos y promociones de los supermercados.

Por último, los supermercados pueden automatizar la gestión del almacén mediante elevadores y cintas transportadoras para una mayor rapidez a la hora de reponer productos o preparación de pedidos.

10. Conclusiones

El proyecto presentado, proporciona una funcionalidad útil para todas las personas, de manera que ayuda de forma notable en aquellas situaciones en que muchos usuarios forman parte del mismo inventario.

Así mismo, la interacción con el usuario ha de ser mejorada, ya que requiere de una fuerte interacción con este para realizar las operaciones. Esta dependencia les genera rechazo, viéndose incrementado en aquellos que no compartan inventario, prefiriendo una simple aplicación de notas.

En cambio, si se implementa la aplicación tal y como se propone en el apartado de futuras mejoras, se ofrece un servicio muy bien estructurado para la administración de productos en casa, que no requiere casi de interacción por parte del usuario; además les avisa de productos que van a llegar a la fecha de caducidad, y ofrece la posibilidad de pedir productos con urgencia.

Esta nueva forma de abastecimiento de los productos del hogar, supone un ahorro de tiempo importante por parte de los usuarios que pueden emplear para realizar otras tareas.

Así mismo, este servicio supone una gran cantidad de información de estudio para supermercados, ya que las acciones de los usuarios quedan registradas, pudiendo generar otros servicios con la información obtenida.

Finalmente, esto puede suponer un cambio radical en la forma de funcionamiento de los supermercados, pudiendo llegar a convertirlos en almacenes de distribución de productos alimenticios.

11. Valoración personal

El proyecto ha supuesto un gran reto, ya que se ha tenido que desarrollar una idea, y definirla hasta tener una estructura rígida que poder implementar.

La idea elegida, basada en la experiencia personal, ofrece una solución que todavía no existe en el mercado, pero que con el avance de la tecnología y la implementación de servicios para todas las tareas habituales, acabará llegando de un modo u otro.

Llevarla a cabo ha supuesto la búsqueda de los medios necesarios para su desarrollo, aprender cómo implementarlos; además de encontrar problemas técnicos por el desconocimiento de algunos de ellos e investigar las posibles soluciones, testearlas, e implementar la más adecuada.

Por ejemplo, en Android los botones flotantes suponen una mejora en la interfaz con gran valor, pero dificultan la visión e interacción con los elementos situados debajo de ellos, por lo que se ha implementado una funcionalidad que oculta los botones cuando se desplazan los elementos de la pantalla hacia arriba y se vuelven a mostrar cuando los mueven hacia abajo.

Sin embargo, para los problemas de la interfaz, comprensión, usabilidad, facilidad de aprendizaje, se ha obtenido ayuda de terceros, que han testeado la aplicación y transmitido sus dudas. Con estas dudas, se ha modificado la interfaz para mejorarla de manera que las dudas no vuelvan a aparecer en nuevos usuarios.

Con el *feedback* por parte de los *testers*, se han introducido instrucciones de funcionamiento dentro de los clientes y explicaciones de algunas de sus características. Además, se incluyen nuevas funciones como la de SUMAR, y la casilla para dar opción de apuntar en la lista de la compra un producto al ser creado. Gracias a los *testers* se han descubierto algunos errores que perjudicaban al ciclo de vida del servicio y han sido corregidos.

Finalmente, el desarrollo de este proyecto ha servido para entender lo complejo que puede llegar a ser la implementación y estructuración de un servicio cliente-servidor. Además, de la correcta gestión del tiempo, y aprender a buscar soluciones para los problemas e implementar la más adecuada para cada uno de ellos.

12. Bibliografía

1. https://developer.android.com/index.html

En este sitio web se encuentran casi la totalidad de la documentación sobre Android; además de ejemplos de código.

2. https://es.stackoverflow.com/

StackOverflow, es el mejor sitio donde encontrar soluciones a problemas, errores e incluso búsqueda de información para hacer elecciones correctas en cuanto a implementaciones de código se refiere.

3. http://php.net/

Aquí se encuentra casi la totalidad de la documentación de PHP. También dispone de ejemplos para entender las funciones.

4. http://phpseclib.sourceforge.net/

Página oficial de la librería de encriptación utilizada para la API. Proporciona ejemplos de utilización de esta.

5. https://www.npmjs.com/package/node-forge

Librería oficial que se ha utilizado para la encriptación en Javascript. Con una extensa variedad de ejemplos para los diferentes cifrados que implementa.

6. https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-mysql-php

Tutorial de instalación paso a paso del servicio LAMP en la Raspberry Pi, así como la posterior configuración.