Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Escuela Superior de Ingeniería y Tecnología
Grado en Ingeniería Informática

Desarrollo de APPS en Android: Cartera virtual

Android APPS development: Virtual wallet

Adexe Sabina Pérez

La Laguna, *11* de *julio* de 2023

D. **Alejandro Pérez Nava**, con N.I.F. 43.821.179-S profesor asociado de Universidad adscrito al Departamento de Informática de la Universidad de La Laguna, como tutor

D. **Fernando Pérez Nava**, con N.I.F. 42.091.420-V profesor asociado de Universidad adscrito al Departamento de Informática de la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"Desarrollo de APPS en Android: Cartera virtual"

ha sido realizada bajo su dirección por D. **Adexe Sabina Pérez**,

con N.I.F. 79.060.750-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de Julio de 2023

Agradecimientos

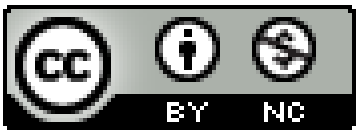
A mi tutor, por haber definido el tema de mi TFG y por haberme guiado cuando lo he necesitado. Por su paciencia y orientación, que han sido fundamentales durante todo el proceso.

A mi pareja, por su incondicional ayuda, por estar a mi lado en los mejores y peores momentos y por brindarme la confianza y el impulso necesario para lograr este objetivo.

A mis padres, ya que sin su apoyo emocional (y económico) no habría sido posible estar aquí. Han sido un pilar fundamental en mi vida y estoy muy agradecido por todo lo que han hecho por mí.

A todos mis profesores de carrera, por proporcionarme los conocimientos y bases necesarias para mi desarrollo académico.

Licencia



Resumen

Actualmente existe una gran variedad de aplicaciones en el mercado con las que podemos hacer gestiones muy diversas en nuestro día a día. Una de las más comunes y que se ha afianzado más después de la época COVID-19 es el pago con tarjeta en establecimientos comerciales.

El objetivo principal de la aplicación es realizar pagos en diferentes comercios en la moneda establecida por ellos. Para esto nuestra aplicación tendrá que ser capaz de leer un código (QR en su primera versión) generado por el propio comercio en el que deberán constar los datos necesarios para que el usuario pueda realizar el pago y visualizar su factura en el móvil.

Cuando hacemos un pago por internet nos interesa conocer datos como qué saldo tengo disponible, cuánto me he gastado, dónde lo he gastado y cuáles son los elementos que he comprado, entre otros. La mayor parte de las aplicaciones que se encuentran actualmente en el mercado no son capaces de resolver todos los puntos mencionados.

Con el desarrollo de DxWallet se consigue resolver este paradigma haciendo uso de una sola aplicación. Se trata de un entorno sencillo y usable donde el usuario puede consultar todos los datos de sus compras en una sola aplicación además de permitir el pago en establecimientos adheridos sin necesidad de utilizar tarjetas de crédito o aplicaciones del banco con tecnología insegura de NFS.

Palabras clave: Wallet, pagos, finanzas, factura, movimientos, gestión.

Abstract

We can currently see a wide variety of applications on the market with which we can carry out very diverse procedures in our daily routine. One of the most common and that has become established after the COVID-19 stage is card payment in commercial establishments.

The main objective of the application is to make payments in different businesses. For this, our application will have to be able to read a code (QR in its first version) generated by the own establishment itself, in which the necessary data must be included so that the user can make the payment and view their invoice on the mobile phone.

When we make an online payment, we are interested in knowing data such as what balance I have available, how much I have spent, where I have spent it and which were the items, among others. Most of the applications that are currently on the market aren't capable of solving all the mentioned points.

With DxWallet it is possible to resolve this paradigm using a single application. It is a simple and usable environment where the user can consult all the data of their purchases in a single application. In addition, the app allows you to pay in authorized establishments without the need to use credit cards or bank applications with insecure NFS technology.

Keywords: Wallet, payments, finances, invoice, movements, management.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Wallet y criptomonedas

Una criptomoneda (también llamado criptoactivo) es un medio digital de intercambio que utiliza criptografía fuerte para asegurar las transacciones, controlar la creación de unidades adicionales y verificar la transferencia de activos.

Una wallet o monedero digital es un elemento donde tus finanzas digitales están custodiadas. En este tipo de monederos se utiliza una moneda o criptomoneda de creación propia con una relación de valor con la moneda local del usuario. Es completamente ajeno a la cuenta bancaria ofreciendo un mayor grado de seguridad. En definitiva, las wallets nos permiten almacenar, enviar y recibir criptoactivos de una forma más segura.

1.2 Motivación

Todas las entidades bancarias tienen su propia aplicación donde poder hacer gestiones, pero a parte de ello, los comercios también están empezando a desarrollar sus propias aplicaciones para realizar los pagos en sus establecimientos así como para brindar a los usuarios una serie de ventajas como el acceso a ofertas y descuentos o la vista de folletos y revistas promocionales.

Si nos centramos en la aplicación que nos proporciona nuestra entidad bancaria para gestionar las finanzas, veremos que permiten diferentes acciones y vistas relacionadas con las transacciones, pero no permite ver qué artículos hemos pagado.

Tomando de referencia algunas de las aplicaciones más utilizadas a día de hoy para visualizar las compras realizadas y las ofertas y descuentos disponibles, nos encontramos en la situación de que podemos visualizar todos los puntos anteriormente mencionados, pero sólo del comercio en cuestión. Así mismo, si la aplicación es de un supermercado, sólo veremos lo referente a ese supermercado.

Con el desarrollo de DxWallet se fusionan ambas versiones en una sola. El usuario tendrá el control y la gestión centralizada de todas sus finanzas. Los comercios no tendrán que realizar un esfuerzo extra para ello, ya que la aplicación tiene una interfaz web que les permite generar el código para el pago de forma automática desde su propio programa TPV. El proceso es muy sencillo y se realiza de forma automatizada para que los usuarios apenas tengan que realizar acción alguna.

Capítulo 2 Antecedentes

En la era digital actual, el uso de los dispositivos móviles se ha convertido en una parte integral de nuestras vidas. Con cada vez más personas dependiendo de sus smartphones para realizar transacciones financieras y administrar su dinero, la necesidad de una aplicación móvil de wallet confiable y segura ha alcanzado proporciones significativas. Es en este contexto que surge la iniciativa de desarrollar una aplicación de wallet móvil funcional y usable que centralice todos los servicios necesarios para llevar a cabo de la forma más eficaz su labor.

El mercado de las aplicaciones móviles de wallet ha experimentado un crecimiento exponencial en los últimos años. Con el aumento de las compras en línea, las transferencias de dinero entre personas y la demanda de servicios financieros digitales, existe un espacio amplio para una solución que proporcione una experiencia de usuario intuitiva y una gestión eficiente de los activos financieros.

El análisis del mercado revela que si bien existen aplicaciones de wallet móvil disponibles, muchas de ellas presentan limitaciones significativas. Algunas carecen de una interfaz de usuario amigable, lo que dificulta la navegación y la comprensión de las funcionalidades disponibles. Otras aplicaciones carecen de características avanzadas, como integraciones con sistemas de pago populares, notificaciones en tiempo real o medidas de seguridad robustas. Además, se observa una falta de personalización y adaptabilidad en las opciones disponibles, lo que no permite a los usuarios adaptar la aplicación a sus necesidades y preferencias específicas.

En este contexto, el proyecto de desarrollo de una aplicación móvil de wallet tiene como objetivo abordar estas deficiencias y brindar una solución completa y confiable a los usuarios.

Además, se incorporarán características avanzadas que mejoren la experiencia del usuario. Esto incluirá la integración con sistemas de pago populares y seguros, notificaciones en tiempo real sobre transacciones y actualizaciones financieras, así como medidas de seguridad sólidas para proteger la información personal y financiera de los usuarios. La aplicación también se diseñará con una arquitectura flexible y escalable, lo que permitirá futuras expansiones y actualizaciones según las necesidades cambiantes del mercado y los usuarios.

El desarrollo de una aplicación móvil de wallet es una oportunidad emocionante para brindar a los usuarios una solución integral y segura para la gestión de sus finanzas personales. Con un enfoque en la usabilidad, la seguridad y las características avanzadas, este proyecto busca establecerse como una opción preferida para aquellos que buscan una aplicación móvil de wallet confiable y completa.

Figura 1: Wallet digitales VS. método de pagos físicos según los usuarios [5]

2.1. Aplicaciones existentes

Existen aplicaciones de todo tipo y que podríamos clasificar de la siguiente manera:

- **Aplicaciones de pago:** Se conectan directamente con el banco y transfieren el dinero a cuenta del comercio como si fuera una tarjeta de crédito física.

- **Aplicaciones de comercio:** Son exclusivas del local al que vayamos a comprar, suelen

ofrecernos descuentos y ventajas exclusivas por utilizar la aplicación y en su mayoría funcionan con un sistema de puntos. En la gran mayoría de ellas, a parte de utilizar la aplicación propia para conseguir el descuento, se debe utilizar además otro método para realizar el pago en dicho establecimiento.

- **Aplicaciones wallet de terceros:** Son un puente entre los dos tipos anteriores. Tienen un sistema propio de pago que es útil en la gran mayoría de los comercios y que nos permite hacer movimientos entre nuestra cuenta y las tiendas a las que vamos a comprar. Muchas acceden directamente a nuestra tarjeta de crédito mientras que otras nos permiten "comprar bonos", o "incrementar saldo" en la propia aplicación para poder utilizarlo.

Este proyecto podría posicionarse en el tercer grupo de los mencionados anteriormente aunque implementando algunas mejoras de usabilidad y seguridad que además "harían de puente" entre los otros dos grupos. Algunas aplicaciones que encontramos actualmente en el mercado son las siguientes:

2.1.1 Ten+ Móvil

Aplicación exclusiva de los servicios de transporte público de Tenerife. Para poder utilizarla hace falta incrementar saldo en su función de monedero. El funcionamiento de esta aplicación es bastante sencillo, nos registramos, compramos un bono en las diferentes estaciones de tranvía o guagua e introducimos el código que nos dan en la aplicación. Este código es una recarga del monedero.

Para hacer los pagos, una vez nos subimos al medio de transporte deseado, basta con seleccionar el bono que hayamos comprado y darle al botón de "validar" para que se nos abra la cámara y podamos leer un código QR que nos descontará el precio del viaje.

De esta aplicación nos interesa la parte del monedero, la recarga de saldo y el pago mediante código QR (retirada de saldo).

Figura 2: Captura de pantalla de la aplicación Ten+ Móvil

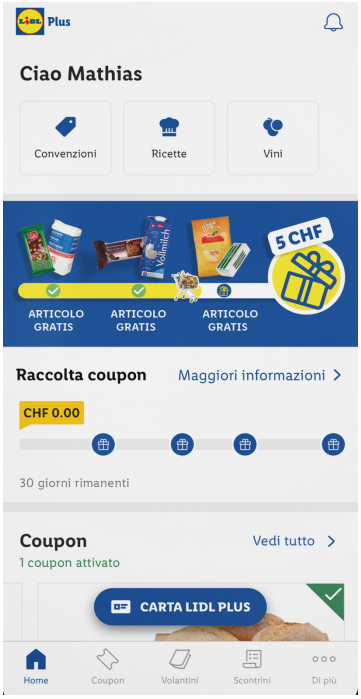


Figura 3: Captura de pantalla de la aplicación Lidl Plus

2.1.2 Lidl Plus

Aplicación de comercio exclusiva sin monedero, da ventajas a sus clientes a través de un código QR de usuario que debe ser leído en caja y sólo se utilizaba para promociones y descuentos. En una actualización posterior se permitió el pago dentro de la aplicación pero mediante el uso directo de una tarjeta de crédito.

Además incluye una funcionalidad para visualizar los tickets del establecimiento y enlazar con varias secciones de la web como son las recetas de usuarios utilizando productos lidl, compra online o las revistas y folletos promocionales.

2.1.3 PayPal

Aplicación de terceros con monedero, nos permite hacer recargas para posteriormente comprar en diferentes comercios y plataformas, pero además nos permite realizar los pagos de forma directa utilizando nuestra tarjeta de crédito sin necesidad de disponer de saldo en la cuenta de la aplicación.

Es interesante ver cómo se hace la conversión automática de divisas. Será uno de los puntos que tendremos en cuenta de esta aplicación a la hora de desarrollar la nuestra.

Otra característica de PayPal es que muestra las transacciones internas, los movimientos de entrada y salida de dinero y las compras entre otros.

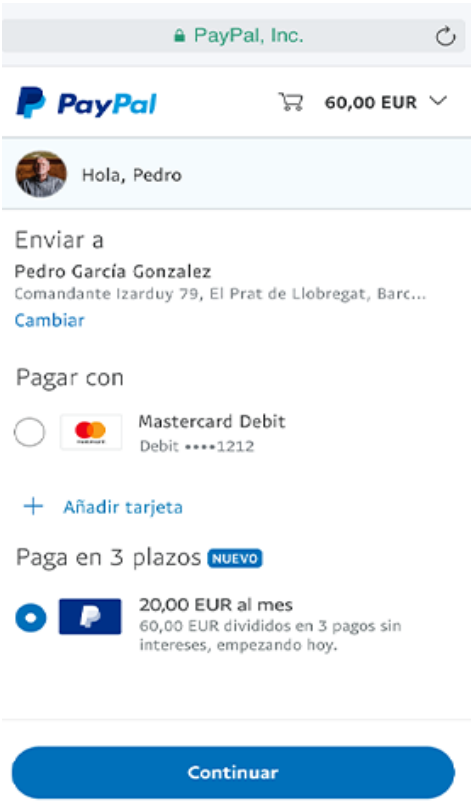


Figura 4: Captura de pantalla de la aplicación PayPal

2.1.4 Google Pay o google wallet

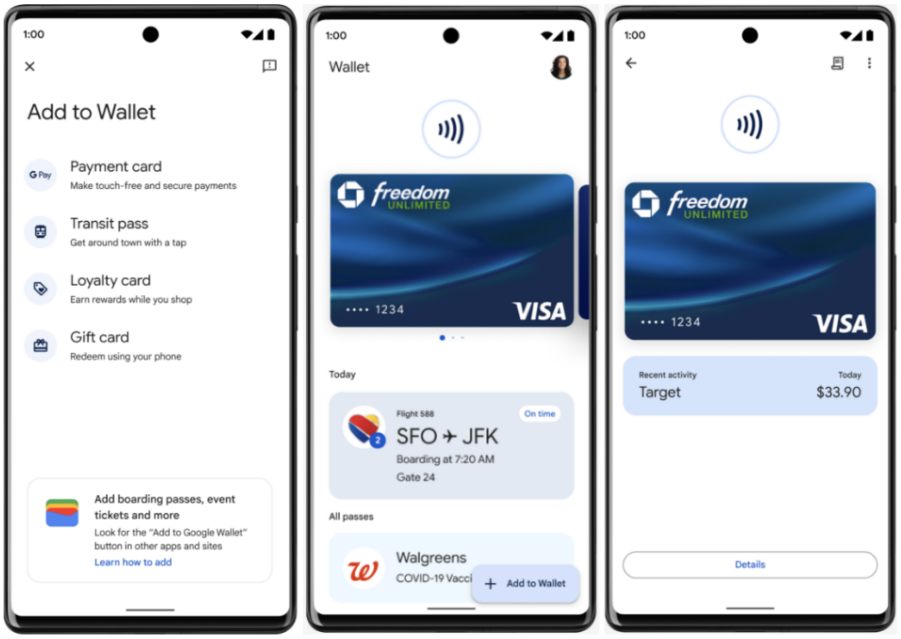


Figura 5: Captura de pantalla de la aplicación GPay o GWallet

Aplicación de terceros sin monedero, accede directamente a nuestra tarjeta y nos permite realizar pagos en diferentes comercios y plataformas de una manera sencilla.

Se sincroniza con nuestro banco y nos muestra el saldo disponible y las diferentes transacciones. Además en una de sus actualizaciones incorporó el pago entre usuarios (envío y recepción de dinero) igual que PayPal.

La conexión con los bancos para mostrar las operaciones y el saldo disponible es una de las características por las que hacemos mención de ella.

Capítulo 3 Objetivos y fases

3.1. Objetivos.

La meta final de este proyecto es la creación de una aplicación a nivel de usuario que sea fácil e intuitiva, que aporte una mejora a las aplicaciones que existen hoy en día en el mercado y que nos permitan resolver el punto más importante, que es centralizar la gestión y mostrar toda la información necesaria a través de una sola aplicación.

La base para el proyecto fue la aplicación Ten+Móvil, que utiliza un sistema de pago basado en bonos que van descontando un importe fijado a través de códigos QR dispuestos en los medios de transporte (tranvía y guaguas).

El objetivo es replicar este sistema de pago abierto a todas las empresas y comercios que deseen utilizarlo, además de añadir ciertas funcionalidades y utilidades para facilitar el uso de la aplicación, como puede ser el almacenamiento de las transacciones o facturas que hayamos realizado de una forma sencilla y localizada.

3.2. Fases.

1. Análisis de mercado y definición de objetivos:

En esta fase inicial, se lleva a cabo una investigación exhaustiva del mercado y de los competidores para identificar las oportunidades y determinar los objetivos del proyecto. Se analizan las necesidades de los usuarios potenciales, las tendencias del mercado y se definen los objetivos específicos que la aplicación de wallet debe cumplir.

2. Diseño y prototipado:

Una vez establecidos los objetivos, se procede al diseño de la interfaz de usuario (UI) y la experiencia de usuario (UX). Se crean prototipos interactivos de la aplicación móvil de wallet para validar las ideas y realizar pruebas de usabilidad. Esta fase permite refinar el diseño y la funcionalidad de la aplicación antes de pasar a la etapa de desarrollo.

3. Desarrollo de la aplicación:

En esta fase, se traducen los diseños y prototipos en código. Se desarrolla la aplicación móvil de wallet utilizando tecnologías y lenguajes de programación adecuados para la plataforma objetivo (por ejemplo, iOS, Android). Se implementan las características necesarias para permitir las transacciones, gestionar el saldo, la seguridad y la integración con sistemas de pago.

4. Pruebas y control de calidad:

Una vez desarrollada la aplicación, se llevan a cabo pruebas para identificar y corregir errores o bugs. Se realiza un control de calidad en diferentes dispositivos móviles y se comprueba que la aplicación cumpla con los requisitos funcionales y de rendimiento. Se corrigen los problemas encontrados y se optimiza la aplicación para garantizar un funcionamiento sólido y confiable.

5. Lanzamiento y distribución:

En esta fase, la aplicación móvil de wallet se prepara para su lanzamiento oficial. Se crea una estrategia de lanzamiento y se prepara la tienda de aplicaciones (App Store, Google Play, etc.) para su distribución. Se generan materiales de marketing, se establece una estrategia de promoción y se preparan las cuentas para la monetización de la aplicación.

6. Implementación y seguridad:

Una vez que la aplicación está disponible para su descarga, se realizan actualizaciones periódicas para mejorar la funcionalidad y la seguridad. Se implementan nuevas características y se corrigen errores conforme se obtiene el feedback de los usuarios. Se asegura que la aplicación cumpla con los estándares de seguridad y se aplican medidas de protección de datos personales y financieros.

7. Monitoreo y análisis:

Después del lanzamiento, se monitorea el rendimiento de la aplicación utilizando herramientas de análisis y seguimiento. Se recopilan datos sobre el uso de la aplicación, las transacciones realizadas y el comportamiento de los usuarios. Estos datos son analizados para identificar áreas de mejora y tomar decisiones estratégicas para optimizar la aplicación y su rendimiento.

8. Venta y soporte al cliente:

En esta fase final, se promueve la venta del producto a través de estrategias de marketing y publicidad. Se ofrece soporte al cliente para responder preguntas, solucionar problemas y recibir comentarios. Se realizan actualizaciones periódicas de la aplicación para mantenerla actualizada y satisfacer las necesidades cambiantes de los usuarios.

Es importante que cada fase lleve su documentación asociada para dar continuidad y soporte efectivo al proyecto.

Capítulo 4 Tecnologías

4.1 Lenguaje y entorno

La finalidad de este proyecto es la de simplificar y unificar las diferentes aplicaciones que existen en el mercado para evitar la saturación del dispositivo móvil con miles de ellas que en el fondo son exactamente iguales, por tanto, no se utilizará ninguna herramienta o tecnología innovadora en este proyecto, a parte de la idea de centralización de tecnologías en una sola.

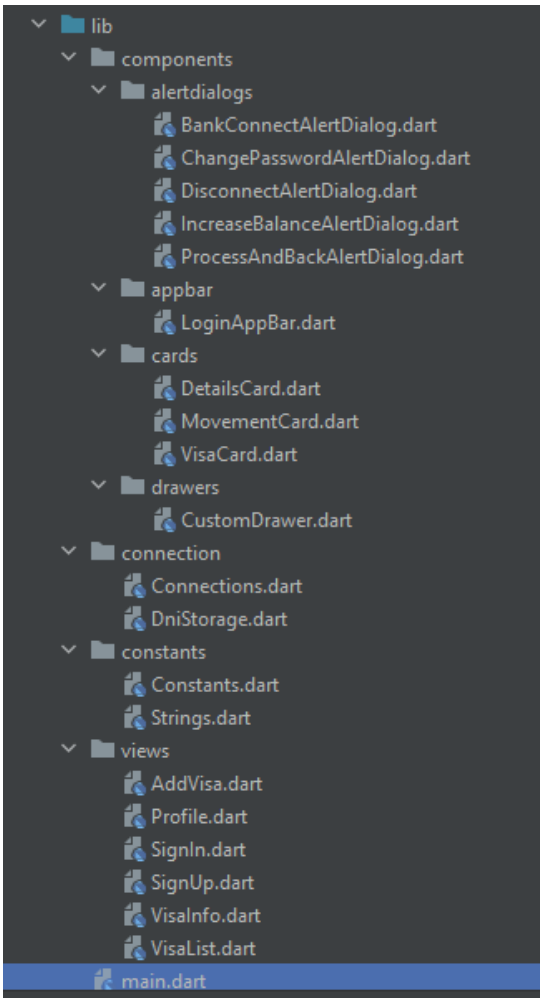


Figura 6: Estructura de archivos.dart de nuestra aplicación en Flutter

Para el diseño de nuestra aplicación inicialmente se pensó en la utilización del lenguaje Java y como herramienta de desarrollo el entorno de **Android Studio**. Posteriormente se decidió utilizar Flutter, que es un framework multiplataforma cuyo lenguaje orientado a objetos, Dart, fue desarrollado por Google y es uno de los más utilizados actualmente en el desarrollo de aplicaciones móviles para Android. La ventaja de este framework respecto a otros multiplataforma es que el código se compila a código nativo, por lo que el rendimiento es superior a otros frameworks basados en "web-views".

Este framework nos permite ordenar y visualizar todos los archivos de la aplicación de una manera sencilla y eficaz, además de disponer de gran cantidad de librerías que podemos utilizar y combinar para conseguir los resultados deseados.

En lo que concierne al almacenamiento de información se utiliza una base de datos SQL, a través de un gestor de base de datos MySQL y de endpoints php para las peticiones.

La página web de administración de comercios y de la aplicación en general, está desarrollada en tecnologías web utilizando **PHP**, **HTML**, **CSS** y **JAVASCRIPT**.

A la hora de realizar un pago, se hace uso de la tecnología de **códigos QR** con la información necesaria para comunicarnos entre la app y la tienda donde vayamos a comprar. Para ello, utilizaremos funciones y librerías ***ya existentes*****,** que sólo tendremos que adaptar a nuestra aplicación. Estas librerías pueden usarse a través del lenguaje **PHP**, lo que las hace idóneas para nuestro proyecto.

Como hosting para nuestra base de datos y para una primera publicación del entorno web utilizaremos un espacio en la nube a través del proveedor **STRATO.es**.

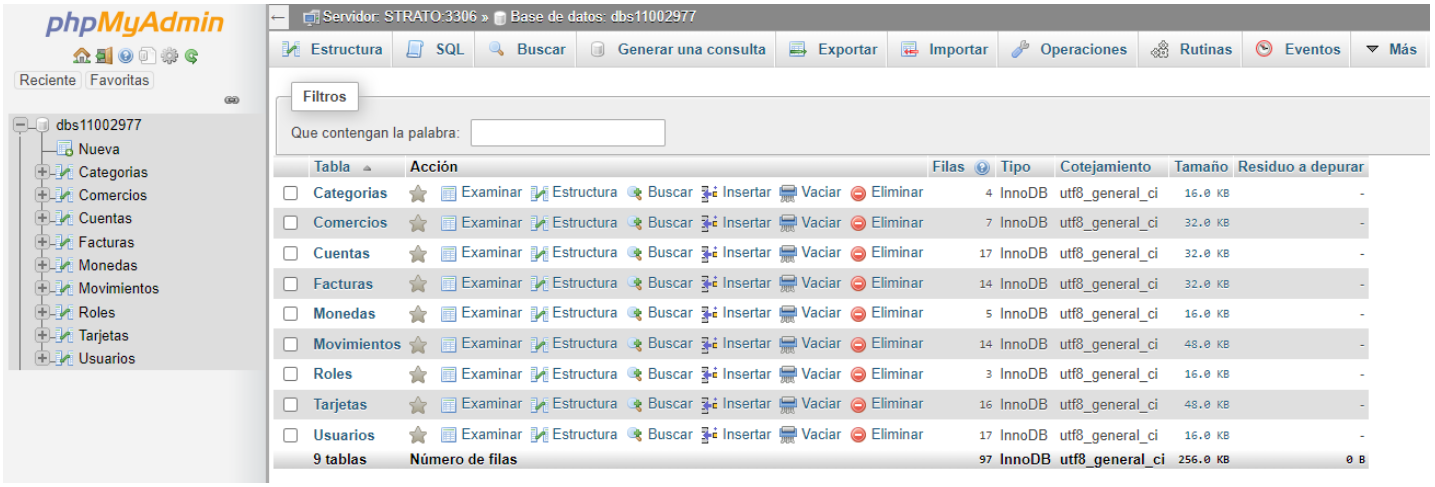


Figura 7: Vista de base de datos en phpMyAdmin

4.2 Desarrollo móvil

Dado que el desarrollo móvil se basa principalmente en código, con el fin de evitar que este documento se vuelva demasiado extenso, se ha proporcionado todo el código relacionado tanto con la aplicación de este proyecto como con la parte web del mismo en un repositorio de GitHub, el cual se encuentra referenciado en la sección [1] de la bibliografía.

4.2.1 Diseño y funcionalidad

La aplicación que utilizará el usuario debe ser sencilla e intuitiva. La función principal será la de realizar diferentes pagos en diferentes monedas virtuales, además de poder ver tanto las transacciones como los recibos de las compras realizadas.

La aplicación contará con **5 vistas** principales claramente diferenciadas, que podrán aumentar según se vea en el avance del proyecto.

1. *Acceso*

La ventana de acceso será muy sencilla. La introducción del correo del usuario y contraseña será suficiente para entrar.

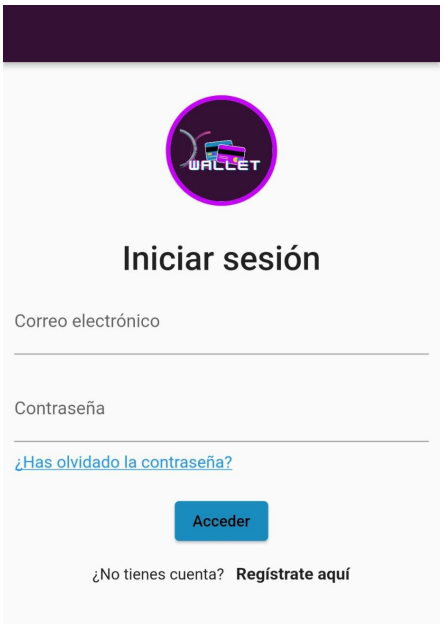


Figura 8: Captura de pantalla del login de la aplicación

2. Registro

Registrarnos en la aplicación será lo primero que tendremos que hacer antes de poder utilizar nuestra aplicación.

Para el correcto funcionamiento, y para poder comprobar la veracidad de los datos, es necesario pedir una serie de datos personales que no se darían en otros casos, como la dirección física, DNI o número de teléfono. Sin rellenar dichos campos no podremos acceder a la aplicación.

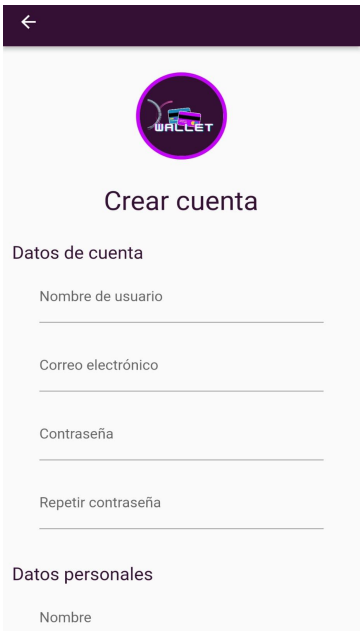


Figura 9: Captura de pantalla del registro de la aplicación

3. *Menú de tarjetas*

Este menú será la pantalla principal a la que accederemos una vez hemos iniciado sesión en nuestra aplicación. El código de esta vista lo podemos ver en el apartado 8.1 del Apéndice.

Aquí se muestra el listado de tarjetas asociadas a nuestra cuenta de usuario. Cada una de ellas tendrá su propia moneda, que a su vez será aceptada (o no) por diferentes comercios.

Tendremos la opción de crear tarjetas nuevas y la moneda que escogeremos será una de las que estén dadas de alta en la aplicación. Esto significa que las tiendas podrán tener su propia moneda siempre y cuando la registren en nuestra base de datos.

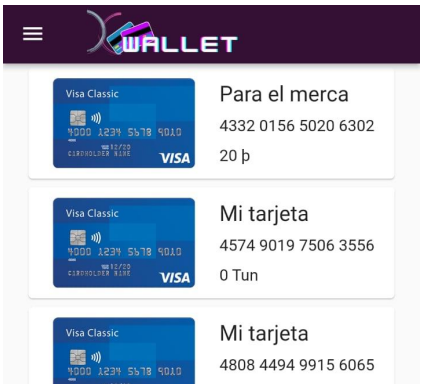


Figura 10: Captura de pantalla del listado de tarjetas del usuario

4. *Tarjeta*

La pantalla de la tarjeta nos mostrará todo lo relacionado con ella: Nombre, saldo disponible y el identificador de la tarjeta, la posibilidad de recargar saldo, ver las transacciones que hemos realizado, hacer nuevos pagos, etc.

```
/// GET LIST OF MOVEMENTS//////////////////////////////////////////////////////
Future<Map> getMovementList(String nid, String visaNumber) async {
  const bool debugThis = false; // Para entrar en el modo depuración
  final url = Uri.parse(GET_MOVEMENTS_URL);
  final response = await http.post(url, body: { 'dni': nid, 'visaNumber': visaNumber,});
  final Map <String, dynamic> resp = { "success" : "0", "body" : ""};
  if (response.statusCode == 200) { // La conexión fue exitosa
    dynamic respuesta = jsonDecode(response.body);
    if (respuesta["success"] == "1") { // Si se encontraron datos
      resp.update("success", (value) => "1");
      resp.update("body", (value) => respuesta["movementsList"]);
    } // Si no se encontraron datos
    else {resp.update("body", (value) => respuesta["message"]);}
  } // Si la respuesta no es exitosa, mostramos un mensaje de error.
  else {resp.update("body", (value) => "ERROR: ${response.statusCode}");}
  return resp;
}
```

Figura 11: Código en flutter para acceder al endpoint de transacciones realizadas

Figura 12: Captura de pantalla de la información de la tarjeta

5. Recibos

La pantalla de recibos es la penúltima de las pantallas descritas como principales. La finalidad de la misma será la de mostrar un listado con todos los recibos que se han generado al pagar con esa tarjeta en las diferentes tiendas. Se corresponde con la misma vista de la imagen anterior.

Tendremos la opción de ver la factura en línea al hacer clic en el movimiento. Este archivo será, en la mayor parte de los casos, el ticket de compra del establecimiento, en el que se detalla el importe, los artículos adquiridos, la fecha y hora de la compra, el nombre del cajero, etc.

6. Perfil de usuario

Por último no podemos olvidarnos de la ventana de perfil de usuario. En ella el usuario podrá ver y modificar sus datos personales dentro de la aplicación. Para el cambio de contraseña se requerirá que el usuario inicie sesión nuevamente tras la modificación de la misma.

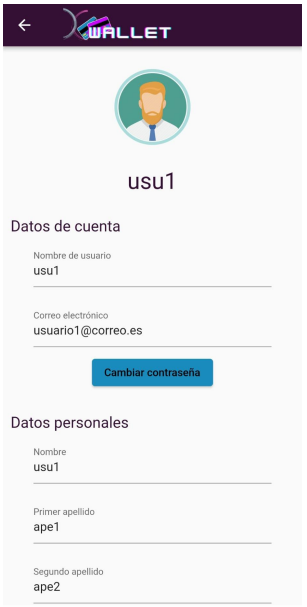


Figura 13: Captura de pantalla de la información del usuario (perfil)

7. Otras vistas

Cabe mencionar que para hacer más fácil la navegación del usuario, a través de un menú lateral desplegable éste podrá ver y dirigirse a otra parte de la aplicación con sólo un toque.

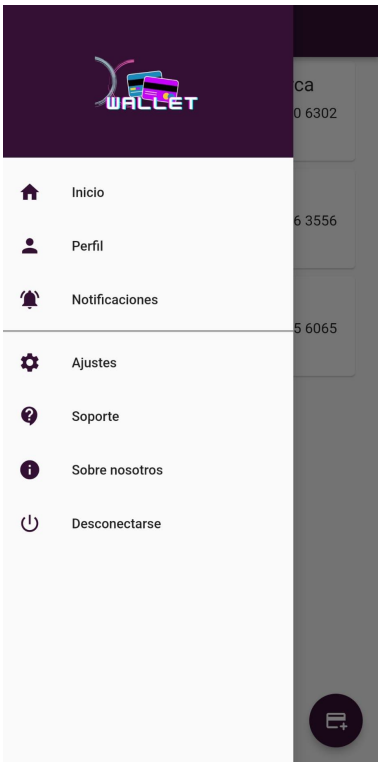


Figura 14: Captura de pantalla del menú lateral

Además existirán otras pantallas como la de creación de nuevas tarjetas, aumento de saldo, pago, notificaciones, ajustes, soporte, etc. Que, aunque también son importantes, las excluimos de esta lista para no extendernos en demasía.

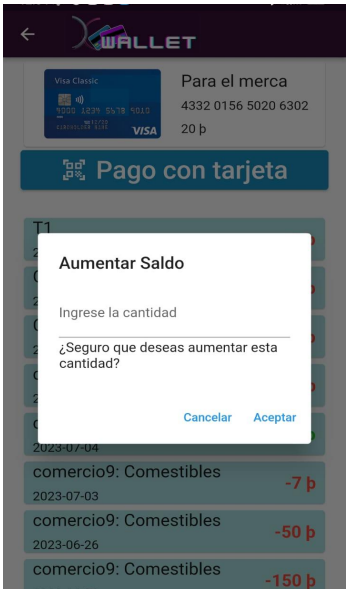


Figura 15: Captura de "Aumentar saldo"

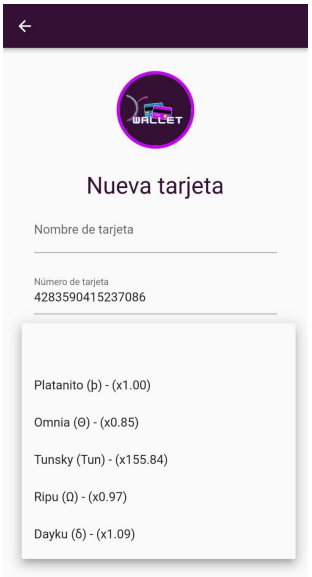


Figura 16: Captura de "Añadir tarjeta"

Para almacenar todos los datos se utilizarán diferentes bases de datos, de modo que la aplicación tendrá que estar conectada permanentemente a internet. Creemos que esto no será un problema ya que la función principal es la gestión de pagos, además de que aligeramos el tamaño y mejoramos la velocidad de la aplicación descargando únicamente lo necesario para cada pantalla que el usuario visite.

4.2.2 Gestión de operaciones (Lector de QR)

Como ya se ha mencionado, el objetivo principal de la aplicación es realizar pagos con tarjeta a diferentes comercios en la moneda establecida por ellos. Para esto nuestra aplicación tendrá que ser capaz de leer un código (preferiblemente QR) generado por el propio comercio en el que deberán constar los datos suficientes para procesar el pago y poder visualizar la factura o ticket cuando éste esté completado.

Por otro lado, si hemos comprado un artículo que no queremos tendremos que realizar una devolución. En este caso la tienda generará un nuevo código QR con el importe de la devolución y ésto creará a su vez un nuevo movimiento en la tarjeta y un recibo asociado.

En cuanto a los códigos QR, tenemos dos opciones para leerlos y hacer nuestras gestiones en la aplicación. Podemos llamar a un lector de códigos que tengamos en nuestro dispositivo o podemos, mediante alguna API, crear nosotros mismos el lector de código dentro de nuestra propia aplicación.

Si bien la primera parte es más sencilla, no lo sería para nuestros usuarios, que tendrían que descargarse, en algunos casos, otra aplicación para poder leer los QR.

Decidimos, por tanto, buscar y crear nuestro propio lector de QR dentro de la aplicación. Esto también nos aporta más seguridad ya que no tenemos que andar transfiriendo datos que podrían ser de carácter sensible a través de aplicaciones de terceros.

Flutter dispone de varias librerías para leer códigos QR, nosotros hemos utilizado una de las más populares, **flutter_barcode_scanner.**

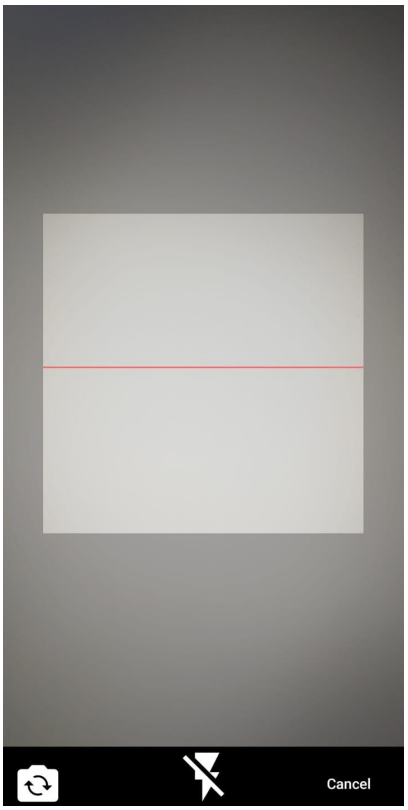


Figura 17: Captura de lector de qr

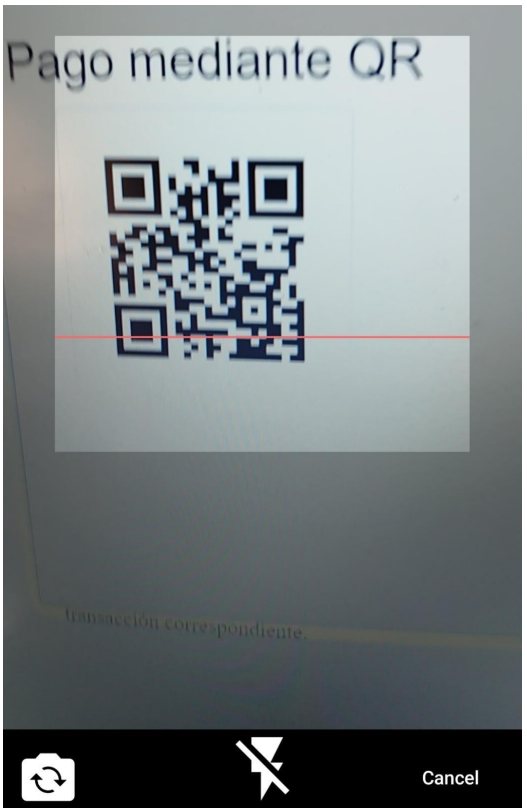


Figura 18: Captura de lector qr + qr generado

4.3 Desarrollo web

4.3.1 Diseño y funcionalidad.

En primera instancia, el diseño de la página web se propone simple y sencillo. La idea es disponer un panel de administración general en el que un responsable o administrador pueda llevar a cabo las tareas de gestión de usuarios de la aplicación, así como de la creación de monedas nuevas entre otros.

Además se desarrolla un panel de Tienda cuya intención es darle a los comercios las herramientas necesarias para poder operar en nuestra aplicación (visualización y gestión de cuentas y facturas, modificación de datos personales, etc).

Cuando una empresa se registra en nuestra web, se envía un mensaje al administrador del sitio que debe dar de alta al comercio para que éste pueda iniciar su actividad. Si dicho comercio quisiera crear una divisa nueva, tendrá que solicitarlo mediante el formulario de registro para que el administrador proceda a valorar, crear y habilitar dicha moneda a este comercio.

Los comercios por sí mismos no pueden crear divisas ni mucho menos modificar el valor actual de las mismas.

1. Panel de administración general

El administrador del sitio podrá ver el listado de comercios y de usuarios actualmente activos en la aplicación y modificar ciertos datos de cada uno, habilitar o deshabilitar a un usuario o comercio, crear nuevas monedas y asignarlas a los comercios que la pidan, etc.

Figura 19: Panel administrador - tabla monedas

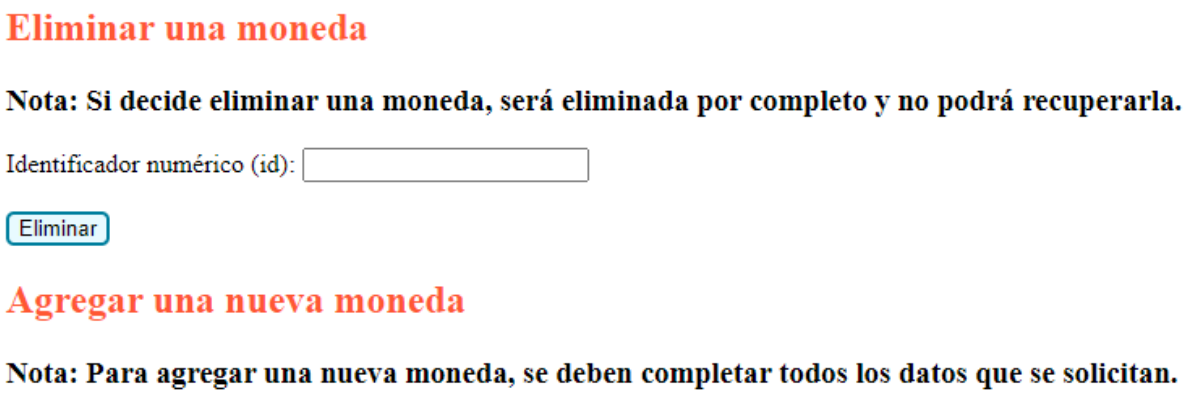


Figura 20: Panel administrador -Acciones tabla monedas

DXWallet: Tu monedero virtual

Usuarios | Comercios | Administradores | Tarjetas | Monedas | Roles

Datos de los comercios

DNI	Nombre de cuenta	Nombre de comercio	Email	Teléfono	Dirección	Cuenta activa
12345678C			ejemplo@correo.es	922000000	Mi casita de niño rico	No
99999999A	comercio9	comercio9: Comestibles	comercio9@correo.es	999999999	direccion ficticia del comercio 9	Sí
A12345678	tienda1	Tienda1	tienda1@gmail.com	911234567	Calle	Sí
B12345678	t1	T1	uncorreo@gmail.com	912345678	Calle T1	Sí
D47930273	t4	Depi	depi@lacion.com	696102555	C/ Varadero, n2	Sí
J87654203	c1	Tienda de comestibles	comestibles@gmail.com	922123456	Calle de Comercio	Sí
M9876543	t3	Taller mecánico	taller@mi.es	812384502	C/La meca	Sí
U12345678	t2	T2	t2@correo.es	822123678	C/ de U	Sí
Z12345678	c11	Comercio 11	uncorreo@gmail.com	922222222	dir	Sí

Figura 21: Panel administrador - tabla comercios

DXWallet: Tu monedero virtual

Usuarios | Comercios | Administradores | Tarjetas | Monedas | Roles

Datos de los usuarios

DNI	Nombre de usuario	Nombre completo	Email	Teléfono	Dirección	Cuenta activa
11111111A	usu1	usu1 ape1 ape2	usuario1@correo.es	111111111	direccion ficticia del usuario 1	Sí
12345678Z	g	Señor Gomez Apellido1 Apellido 2	g@g.es	612345678	C/ del Señor G, n25, Frente a ITV	No
22222222A	usu2	usu2	usuario2@correo.es	222222222	direccion ficticia del usuario 2	Sí
28462549G	a1	Alumno Ape Ape2	a@a.es	922536547	38008	No
77777777A	Pepito	Pepe De Los Palotes	p@p.es	653246852	Calle De Pepe sin N	Sí

Figura 22: Panel administrador - tabla usuarios

Eliminar un usuario

Nota: Si decide eliminar un usuario, será eliminado por completo y no podrá recuperarlo. Si solo necesita desactivarlo, mejor diríjase a la sección de deshabilitar usuario.

Dni del usuario:

Eliminar

Figura 23: Panel administrador -Acciones tabla usuarios

2. Panel de gestión del comercio

Los comercios, cuando accedan al panel administrativo, podrán ver sus datos y modificarlos, podrán ver las facturas que han generado y si están cobradas o no y podrán crear nuevas facturas. El cobro de los artículos a través de la aplicación se hace mediante códigos QR generados en esta ventana de administración y que detallaremos en el apartado siguiente.

DNI	Nombre de cuenta	Nombre de comercio	Categoría	Email	Teléfono	Dirección
99999999A	comercio9	comercio9: Comestibles	Restauración	comercio9@correo.es	999999999	direccion ficticia del comercio 9

Número de Factura	Fecha	Importe	Moneda	Ticket	Cobrado
FAC001	2023-06-27	100	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket2.PNG	No
FAC002	2023-06-26	50	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket2.PNG	No
FAC003	2023-06-25	-75	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket1.PNG	Sí
FAC004	2023-06-24	200	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket2.PNG	Sí
FAC005	2023-06-23	150	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket2.PNG	No
FAC0362	2023-07-03	23	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket1.PNG	Sí
FAC5359	2023-07-09	12	Platanito	https://sebt.es/adexe/dxwallet/appBills/Ticket2.PNG	Sí

Figura 24: Panel de gestión del comercio

4.3.2 Gestor de operaciones (generador de QR).

Los códigos QR que vamos a generar provienen principalmente de los comercios donde realizaremos los pagos. Cada comercio utiliza un programa de facturación diferente, con códigos, datos y números de factura únicos que pueden no coincidir con los de otros comercios.

Dado que deseamos que nuestra aplicación sea compatible con la mayor cantidad de comercios posible, no podemos solicitarles que modifiquen todo su sistema para adaptarse a nosotros. En cambio, somos nosotros quienes debemos adaptarnos a ellos. Por esta razón, podemos solicitarles que guarden las facturas en una ubicación accesible para nosotros una vez que sean generadas.

De este modo, cada factura generada por la empresa tendrá un código distinto en nuestra aplicación, pero estará vinculada a la factura de la transacción correspondiente, que ya contiene todos sus datos. Por lo tanto, el código QR generado por la tienda solo necesitará contener los siguientes datos:

- Número de tarjeta de la tienda de la que se genera la factura
- Importe

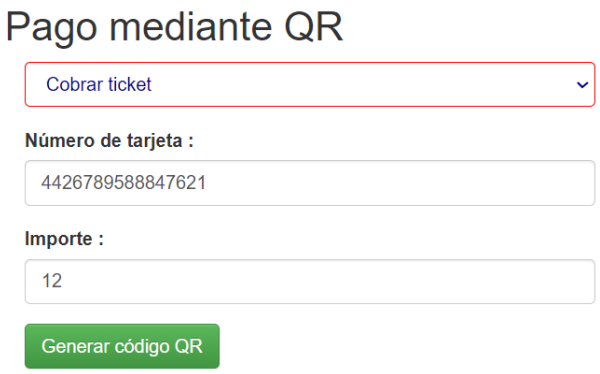


Figura 25: Formulario de generación de QR de pago

Figura 26: Ejemplo de QR de pago

Una vez realizado el pago se creará el movimiento correspondiente, se enlazará con el código de ticket que la tienda ha especificado y el usuario podrá acceder a él instantáneamente desde el apartado de movimientos.

Figura 27: Ejemplo de transacción en aplicación del cliente

Como esta parte es dependiente de cada tienda, para hacer las pruebas pertinentes se crea una ventana sencilla en HTML en la que podemos introducir el número de cuenta de la tienda y el importe para generar un QR que pueda ser leído con el móvil para hacer el pago.

4.3.3 Procedimiento de facturación

La generación de facturas la hace automáticamente cada comercio. La única modificación que tendrá que hacer será indicarnos la ruta en la que se almacena dicha factura para que nuestra aplicación pueda visualizarla. Dichas rutas estarán disponibles tanto para el cliente como para el comercio, de forma online, almacenadas en nuestro servidor.

Figura 28: Ticket de transacción

4.4 Base de datos

La base de datos permite gestionar y organizar información relacionada con transacciones financieras, usuarios y sus roles, categorías de productos o servicios ofrecidos por los comercios y detalles de facturación. Su estructura está diseñada para facilitar el almacenamiento y recuperación eficiente de los datos, así como para respaldar operaciones de consulta y generación de informes.

La elección de un sistema de gestión de bases de datos (DBMS) depende en gran medida de las necesidades y requisitos específicos de cada aplicación. En este caso, una base de datos SQL es una opción más adecuada que una base de datos NoSQL, como MongoDB.

Algunas razones por las que se decidió desarrollar la base de datos en lenguaje SQL son las siguientes:

- Estructura de datos: La estructura de los datos en esta aplicación está bastante definida y estructurada, lo que significa que una base de datos SQL es una buena opción ya que está diseñada para manejar datos estructurados y relacionales.

Por otro lado, una base de datos NoSQL es más adecuada para datos no estructurados y con menos relaciones.

- Transacciones: En esta aplicación, se realizan transacciones en las que se deben asegurar la consistencia y la integridad de los datos. Las bases de datos SQL están diseñadas para manejar transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que significa que garantizan que las transacciones sean confiables y consistentes. Aunque las bases de datos NoSQL también pueden manejar transacciones, su soporte para ellas es más limitado.
- Consultas complejas: Si se necesitan realizar consultas complejas, como consultas que involucren múltiples tablas y relaciones, una base de datos SQL es la mejor opción. Las bases de datos en lenguaje SQL permiten realizar consultas complejas de manera eficiente.

Por todo esto, la base de datos está implementada en lenguaje SQL mediante el uso de un modelo entidad-relación reflejado en la *figura 26* que se encuentra un poco más abajo.

Para la versión de pruebas hemos optado por hacer unas tablas sencillas con los campos mínimos obligatorios que necesitamos para que nuestra aplicación funcione, y poco a poco se ha ido agrandando a medida que fuimos viendo cosas que era necesario añadir.

Para el correcto funcionamiento de nuestra aplicación y de todas las partes de la misma, se distinguieron dos grandes bloques de información: La relacionada con el usuario, sus tarjetas y movimientos, y la relacionada con los comercios, sus tarjetas, las monedas utilizadas y las facturas generadas.

Es importante mencionar que inicialmente se planteó la creación de una tabla adicional que hiciera el papel de "banco" para llevar a cabo las pruebas de conexión necesarias. Sin embargo, al final se descartó esta idea y se propuso en su lugar simular que tanto la aplicación como el comercio accedieran a través de una pasarela de pago con el banco pertinente.

De este modo, el banco confirmará automáticamente las transacciones de incremento o retiro de saldo en la aplicación y no tendríamos que preocuparnos por la seguridad, ya que las pasarelas de pago actuales se encargan de toda esta gestión.

4.4.1 Acceso a la base de datos

La base de datos se encuentra en el hosting en la nube de **STRATO.es** y el acceso a ella se realiza a través del gestor de bases de datos mysql **PHPMYADMIN**. Para conectar nuestra aplicación con la base de datos, utilizamos endpoints en php que tienen los datos de conexión así como las consultas correspondientes dependiendo de la función que queramos ejecutar en nuestra aplicación. Un ejemplo del código php lo podemos encontrar en el *Apartado 8.3* del apéndice.

```php
<?php
class Conexion {
    // Configurar la conexión a la base de datos
    private $host = 'localhost';
    private $usuario = 'usuario';
    private $password = 'contraseña';
    private $baseDeDatos = 'nombre_de_la_base_de_datos';
    private $conexion;

    public function __construct() {
        $this->conexion = new mysqli($this->host, $this->usuario, $this->password, $this->baseDatos);
        if ($this->conexion->connect_error) {
            die("Error de conexión: " . $this->conexion->connect_error);
        }
        $this->conexion->set_charset("utf8");
    }

    public function getConexion() {
        return $this->conexion;
    }

    public function cerrarConexion() {
        $this->conexion->close();
    }
}
```

Figura 29: Ejemplo del fichero de conexión con la base de datos.

4.4.2 Almacenamiento de datos

El almacenamiento de todos los datos referentes a los usuarios y comercios, así como a las monedas utilizadas estará centralizado en la aplicación. Ni los usuarios ni los comercios dispondrán del control de los datos. Esto nos permitirá que las empresas más pequeñas que no tengan recursos también puedan acogerse a nuestros servicios sin necesidad de una inversión en seguridad o servidores.

El usuario, de igual manera, podrá acceder a los recibos o transacciones que haya realizado a través de cada una de las diferentes tarjetas y visionarlas de forma tanto online como offline previa descarga a su dispositivo móvil.

Para almacenar los datos de las facturas se nos plantean 2 posibilidades:

- Guardar los datos en texto plano, y cada vez que se quiera visualizar o descargar alguna factura generar la previsualización online o crear y descargar un pdf “in situ”.

- Guardar los datos en formato imagen o pdf directamente y enviar al usuario un enlace a la dirección en la que esté almacenado dicho archivo.

Se opta por la segunda opción, formato imagen, porque aunque es verdad que ocupa más espacio de almacenamiento, es más fácil para las empresas adherirse a nuestra aplicación sin tener que hacer modificaciones en su forma de facturación actual. Mediante enlaces a las

facturas permitimos que todos los tipos y modelos sean válidos y estén al alcance del usuario.

4.4.3 Esquema

Las relaciones entre las tablas quedan como se muestran a continuación:

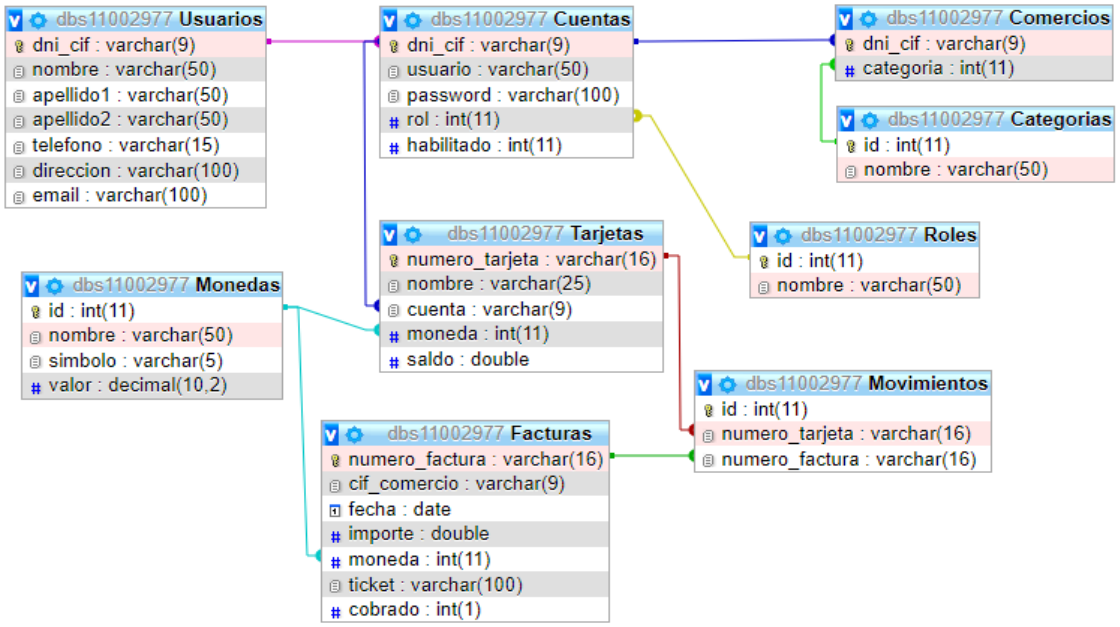


Figura 30: Esquema entidad-relación

4.5 Pasarela de pago

Para la realización de recargas o retiradas de saldo en nuestra aplicación, necesitaremos conectarnos de forma segura a nuestro banco. Para ello lo más fácil y seguro será utilizar una pasarela de pago (ya implementadas y accesibles para todo el público) que sea segura, y tratar las recargas como compras, y las retiradas como devoluciones online.

En nuestro prototipo no hacemos uso de dicha pasarela de pago, sino que damos por hecho que se realiza este proceso de forma satisfactoria para poder hacer pruebas de recargas y pagos en nuestra base de datos. En la versión final, esto se cambiará por una llamada a la pasarela de pago, que ya implementa toda la seguridad necesaria para la transacción y contactará con el banco correspondiente del usuario.

Figura 31: Logo pasarela de pago Redsys

Capítulo 5 Conclusiones y líneas futuras

5.1 Conclusiones

El mercado está lleno de aplicaciones muy similares pero exclusivas para cada marca, empresa o comercio y esto obliga a que el usuario tenga que estar constantemente buscando, descargando, actualizando o incluso aprendiendo las diferentes aplicaciones para cada tarea que deseen realizar.

Nuestra aplicación simplifica este proceso, al ser genérico y multiplataforma, con una interfaz única para todos los dispositivos y comercios, conseguimos llegar a más cantidad de usuarios y les simplificamos las tareas aprendizaje y almacenaje de otras aplicaciones que no siempre son compatibles, o que simplemente nos da pereza de descargar "una más" para un momento puntual de nuestra rutina. Con esta aplicación podremos pagar en todos los establecimientos adheridos y ver en un mismo sitio tanto el saldo disponible como los movimientos y los ítems que hemos comprado en cada uno de ellos.

5.2 Líneas futuras

Como próximos pasos a dar en el desarrollo de la aplicación, se podría implementar un sistema por puntos igual que el que existe actualmente en otras aplicaciones, dando ventajas exclusivas para el usuario, que sean dependientes y gestionadas por cada tienda o comercio en exclusiva.

Esto requerirá de una plataforma web de gestión, accesible para las empresas, en la que puedan gestionar este y otros aspectos relacionados con las transacciones. Desde ahí se podría, además, hacer gestiones más complejas referentes al saldo, ventas, stock de productos y otras funcionalidades.

Se podría plantear la creación de un "pago directo" sin necesidad de la utilización de bonos, que conecte directamente con el banco, haga la conversión y transfiera el dinero de una única vez.

Capítulo 6 Summary and Conclusions

6.1 Conclusions

The market is full of very similar applications, but exclusive to each brand, company, or store, and this forces users to constantly search, download, update, or even learn different applications for each task they want to perform.

Our application simplifies this process by being generic and cross-platform. With a unique interface for all devices and stores, we can reach a larger number of users and simplify the tasks of learning and storing other applications that are not always compatible or that we simply don't feel like downloading "one more" for a specific moment in our routine. With this application, we will be able to make payments at all affiliated businesses and view in one place both the available balance and the transactions and items we have purchased at each of them.

6.2 Future work

As next steps in the development of the application, we could implement a points system similar to the one currently existing in other applications, providing exclusive advantages for the user that are dependent on and managed by each individual store or merchant.

This will require a web-based management platform accessible to businesses, where they can handle this and other transaction-related aspects. From there, more complex tasks related to balances, sales, product stock, and other functionalities could also be managed.

We could also consider the creation of a "direct payment" option without the need to buy credit, directly connecting with the bank, performing the conversion, and transferring the money at once.

Capítulo 7 Presupuesto

En esta sección se especifican los costes que supone el proyecto, teniendo en cuenta tanto los recursos humanos como materiales.

Descripción	Coste por unidad	Unidades	Coste total
Hardware: Equipo informático y de red completo	2.000,00€	1 equipamiento pack completo	2.000,00€
Software: Flutter, Android Studio	0,00€	1 Licencia free use	0,00€
Hosting: STRATO.es	3,99€/mes	3 meses	11,97€
Desarrollador	35€/h	240h	8.400,00€
Total			**10.411,97€**

Tabla 1: *Presupuesto*

Capítulo 8 Apéndice

8.1 Código en Flutter para la vista de lista de tarjetas

(VisaList.dart)

```
class VisaList extends StatefulWidget {
  const VisaList({Key? key}) : super(key: key);

  @override
  _VisaListState createState() => _VisaListState();
}

class _VisaListState extends State<VisaList> {
  final bool debugThis = false; // Para entrar en el modo depuración
  late Future<Map> listOfVisas = Future<Map>.value({});

  @override
  void initState() {
    super.initState();
    loadVisaList();
  }

  void loadVisaList() async {
    String? loadedDNI = await DniStorage.loadDNI();
    setState(() {
      listOfVisas = getVisaList(loadedDNI!);
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: LoginAppBar(),
      drawer: CustomDrawer(),
      body: SingleChildScrollView(
        child: FutureBuilder<Map>(
          future: listOfVisas,
          builder: (BuildContext context, AsyncSnapshot<Map> snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return Center(
                child: SizedBox(
```

```
      width: 40.0,
      height: 40.0,
      child: CircularProgressIndicator(),
    ),
  );
} else if (snapshot.hasError) {
  return Text('Errores: ${snapshot.error}');
} else if (!snapshot.hasData) {
  return Text('No data available');
}else { // Retorna datos
  if (snapshot.data!["success"] == "0") {
    return Column(
      children: [
        gapH30,
        Center(
          child: Text(snapshot.data!["body"],
            style: const TextStyle(
                fontSize: SUBTITLE_SIZE,
            color: DISABLED_COLOR),
          ),
        ),
        gapH30,
        const Icon(Icons.credit_card_off_rounded,
          size: TITLE_SIZE * 4,
          color: DISABLED_COLOR,
        ),
      ],
    );
  }
  else {
    List<dynamic> visaList = snapshot.data!["body"];
    int cardCount = visaList.length;
    return Column(
      children: List.generate(cardCount, (index) {
        Map<dynamic, dynamic> visaData = visaList[index];
        return VisaCard(
          imagePath: BLUE_CREDIT_CARD,
          name: visaData["nombre"],
          number: visaData["numero_tarjeta"],
          amount:
              "${visaData["saldo"].toString()} ${visaData["moneda"]}",
          onTap: () {
```

```
                    Navigator.pushNamed(context, "VisaInfo", arguments: {"visaNumber" :
visaData["numero_tarjeta"]});
                  },
                );
              }),
            );
          }
        }
      },
    ),
  ),
  floatingActionButton: FloatingActionButton(
    backgroundColor: CUSTOM_PRIMARY_DARK,
    onPressed: () {
      // Acción al presionar el botón
      Navigator.pushNamed(context, "AddVisa");
    },
    child: const Icon(Icons.add_card_rounded),
  ),
 );
 }
}
```

8.2 Código en flutter para la obtención del listado de tarjetas

(Connections.dart)

```
Future<Map> getVisaList(String nid) async {
 const bool debugThis = false; // Para entrar en el modo depuración
 final url = Uri.parse(GET_VISAS_URL);
 final response = await http.post(url, body: {
  'dni': nid,
 });

 final Map <String, dynamic> resp = { "success" : "0", "body" : ""};

 if (response.statusCode == 200) { // La conexión fue exitosa
  dynamic respuesta = jsonDecode(response.body);
  if (respuesta["success"] == "1") { // Si se encontraron datos
   // Guardamos la respuesta del login
   resp.update("success", (value) => "1");
```

```
    resp.update("body", (value) => respuesta["visaList"]);
   }
   else { // Si no se encontraron datos
    resp.update("body", (value) => respuesta["message"]);
   }
  }
  else {
   // Si la respuesta no es exitosa, mostramos un mensaje de error.
   resp.update("body", (value) => "ERROR: ${response.statusCode}");
  }
  return resp;
}
```

8.3 Código en php para la obtención del listado de tarjetas (endpoint)

(getListOfVisas.php)

```
<?php
// Obtener los datos del formulario
$dni_cif = $_POST['dni'];

require_once 'connect.php';

// Consulta SQL para obtener la lista de visas del usuario proporcionado
$sql = "SELECT * FROM Tarjetas WHERE cuenta = '$dni_cif'";

$result['visaList'] = array();
$response = mysqli_query($conn, $sql);

if (mysqli_num_rows($response) > 0) {
        while ($row = mysqli_fetch_assoc($response)) {
                $index['numero_tarjeta'] = $row['numero_tarjeta'];
                $index['cuenta'] = $row['cuenta'];
                $index['saldo'] = $row['saldo'];
                $index['nombre'] = $row['nombre'];

                $currency = $row['moneda'];
```

```
                $sql = "SELECT simbolo FROM Monedas WHERE id = $currency";

                $response2 = mysqli_query($conn, $sql);
                if (mysqli_num_rows($response2) == 1) {
                        $row2 = mysqli_fetch_assoc($response2);
                        $index['moneda'] = $row2['simbolo'];
                        $result['success'] = "1";
                        $result['message'] = "success";
                }
                else {
                        $result['success'] = "0";
                        $result['message'] = "No hay monedas";
                }
                array_push($result['visaList'], $index);
        }

} else{
        $result['success'] = "0";
        $result['message'] = "No hay tarjetas";
}
mysqli_close($conn);
echo json_encode($result);
?>
```

8.4 Código en php para la ventana de administración de los comercios

(shopage.php)

```
<!DOCTYPE html>
<html>
<head>
  <title>DXWallet: Shop Panel</title>
  <link rel="stylesheet" type="text/css" href="/css/tablestyle.scss">
</head>
<body class="container">
<div id="sidebar">
    <ul>
<br><br><br><br>
      <li><a href="#"
onclick="window.open('https://sebt.es/adexe/facturacion/genQR/index.php','ventana','width=500,height=400'); return false;">Cobrar</a></li>
          <br><br>
      <li><a href="#" onclick="openTab('invoices')">Facturas</a></li>
          <br>
      <li><a href="#" onclick="openTab('mybcard')">Tarjeta</a></li>
          <br>
      <li><a href="#" onclick="openTab('modificar-datos')">Modificar datos</a></li>
          <br><br>
          <br><br>
          <br><br>
          <br><br>
      <li><a href="/index.php">Desconectarse</a></li>
    </ul>
  </div>
<div id="content">
  <div class="content">
  <h1>DXWallet: Tu monedero virtual</h1>

 <div>
 <h2>Datos del comercio</h2>
  <table id="shop-data">
    <tr>
```

```
    <th>DNI</th>
    <th>Nombre de cuenta</th>
          <th>Nombre de comercio</th>
          <th>Categoría</th>
          <th>Email</th>
          <th>Teléfono</th>
          <th>Dirección</th>
  </tr>
        <tbody>

 <?php
 $user = $_GET['username'];
// $user = "comercio9";
// Crear la conexión a la base de datos
require_once 'connect.php';

// Verificar la conexión con la base de datos
if (!$conn) {
   die("Error conectando con la base de datos: " . mysqli_connect_error());
}
$latecif;
$sql = "SELECT dni_cif FROM Cuentas WHERE usuario = '$user'";
if ($result = mysqli_query($conn, $sql)) {
   if (mysqli_num_rows($result) != 0) {
                  $row = mysqli_fetch_assoc($result);
                  $cif = $row['dni_cif'];
                  $latecif = $cif;
         } else {
      mysqli_close($conn);
                  header("Location: loginerror.php");
                  exit();
   }
}

$sql = "SELECT C.dni_cif AS dni, C.usuario AS usuario,
U.nombre AS nombre, U.telefono AS telefono, U.email AS email, U.direccion AS direccion,
T.nombre as categoria
FROM Usuarios AS U INNER JOIN Cuentas AS C
```

```
ON C.dni_cif=U.dni_cif INNER JOIN Comercios as M
ON C.dni_cif = M.dni_cif INNER JOIN Categorias as T
ON T.id = M.categoria WHERE C.dni_cif = '$cif'";

if ($result = mysqli_query($conn, $sql)) {
   if (mysqli_num_rows($result) != 0) {
      while ($row = mysqli_fetch_assoc($result)) {
         echo "<tr class='rowline'>
            <td>" . $row['dni'] . "</td>
                                    <td>" . $row['usuario'] . "</td>
            <td>" . $row['nombre'] . "</td>
                                    <td>" . $row['categoria'] . "</td>
                                    <td>" . $row['email'] . "</td>
                                    <td>" . $row['telefono'] . "</td>
                                    <td>" . $row['direccion'] . "</td>";
            echo "</tr>";
      }
   } else {
      echo "<tr><td colspan='6'> Ha ocurrido un error inesperado, por favor, vuelva a iniciar
sesión para intentar cargar los datos de su comercio </td></tr>";
   }
}
?>
 </tbody>
 </table>
</div>
<br><br>
</div>
 <br><br>
 <div id="new" class="tab" style="display: none;">
         <h2>Nueva compra</h2>
         <form method="post" action="https://sebt.es/adexe/facturacion/genQR/index.php"
target="ventana" name="Cobro_QR" onsubmit=mandar();>

                  <br><br>
                  <div class="form-group">
                           <input type="submit" name="submit" class="btn btn-primary"
value="Cobrar">
```

```
                </div>
            </form>
    </div>

    <div id="invoices" class="tab" style="display: none;">
    <h2>Facturas</h2>
    <table id="invoices-data">
      <tr>
        <th>Número de Factura</th>
        <th>Fecha</th>
            <th>Importe</th>
            <th>Moneda</th>
            <th>Ticket</th>
            <th>Cobrado</th>
      </tr>
          <tbody>
<?php
$sql = "SELECT * FROM Facturas WHERE cif_comercio = '$cif'";

if ($result = mysqli_query($conn, $sql)) {
        if(mysqli_num_rows($result) > 0){
                while ($row = mysqli_fetch_assoc($result)) {
                        $tmp = $row['moneda'];
                        $sql2 = "SELECT nombre FROM Monedas WHERE id = '$tmp'";
                        if ($result2 = mysqli_query($conn, $sql2)) {
                                $moneda = mysqli_fetch_assoc($result2);
                                echo "<tr class='rowline'>
                                        <td>" . $row['numero_factura'] . "</td>
                                        <td>" . $row['fecha'] . "</td>
                                        <td>" . $row['importe'] . "</td>
                                        <td>" . $moneda['nombre'] . "</td>
                                        <td>" . $row['ticket'] . "</td>";
                                        if ($row['cobrado'] == 0) {
                                                echo "<td> No </td>";
                                        } else {
                                                echo "<td> Sí </td>";
                                        }
```

```
                                        echo "</tr>";
                                }
                        }
                } else {
                        echo "<tr><td colspan='6'> No se encontraron facturas para su comercio
</td></tr>";
                }
}
?>
 </tbody>
 </table>
 </div>


 <div id="mybcard" class="tab" style="display: none;">
 <h2>Tarjeta del comercio</h2>
  <table id="bcard-data">
    <tr>
      <th>Número de tarjeta</th>
      <th>Nombre de la tarjeta</th>
            <th>Moneda asociada</th>
            <th>Saldo</th>
    </tr>
          <tbody>
<?php
$sql = "SELECT * FROM Tarjetas WHERE cuenta = '$cif'";

if ($result = mysqli_query($conn, $sql)) {
        if(mysqli_num_rows($result) > 0){
                while ($row = mysqli_fetch_assoc($result)) {
                        $tmp = $row['moneda'];
                        $sql2 = "SELECT nombre FROM Monedas WHERE id = '$tmp'";
                        if ($result2 = mysqli_query($conn, $sql2)) {
                                $moneda = mysqli_fetch_assoc($result2);
                                echo "<tr class='rowline'>
                                                <td>" . $row['numero_tarjeta'] . "</td>
                                                <td>" . $row['nombre'] . "</td>
                                                <td>" . $moneda['nombre'] . "</td>
```

```
                                        <td>" . $row['saldo'] . "</td></tr>";

                        }
                }
        } else {
                echo "<tr><td colspan='4'> No se encontraron tarjetas para su comercio
</td></tr>";
        }
}

mysqli_close($conn);
?>
 </tbody>
 </table>
 <br><br>
 </div>
        <div id="modificar-datos" class="tab" style="display: none;">
                <h2>Modificar datos</h2>
                <h3> Nota: No se puede modificar el DNI, la tarjeta ni la moneda. Si usted
marcó que quería una moneda personalizada, un administrador se pondrá en contacto con usted
para tramitarlo. Si por el contrario detectó algún error en alguno de los dos campos indicados,
<a href="#" style="color: blue; text-decoration: underline;">contacte aquí.</a></h3>
                <form action="../helper/actions/modshop.php" method="POST">
                        <br>
                        <div class="form-group" style="display: none;">
                                <label for="dni">DNI: </label>
                                <input type="text" name="dni" id="dni" required value="<?php
echo $latecif; ?>">
                        </div>
                        <div class="form-group">
                                <label for="accountName">Nombre de cuenta: </label>
                                <input type="text" name="accountName" id="accountName">
                        </div>
                        <br>
                        <div class="form-group">
                                <label for="shopName">Nombre de comercio: </label>
                                <input type="text" name="shopName" id="shopName">
                        </div>
                        <br>
```

```
                    <div class="form-group">
                        <label for="category">Categoría: </label>
                        <input type="text" name="category" id="category">
                    </div>
                    <br>
                    <div class="form-group">
                        <label for="email">Email: </label>
                        <input type="text" name="email" id="email">
                    </div>
                    <br>
                    <div class="form-group">
                        <label for="phone">Teléfono: </label>
                        <input type="text" name="phone" id="phone">
                    </div>
                    <br>
                    <div class="form-group">
                        <label for="address">Dirección: </label>
                        <input type="text" name="address" id="address">
                    </div>
                    <br>
                    <div class="form-group">
                        <label for="visaName">Nombre de tarjeta: </label>
                        <input type="text" name="visaName" id="visaName">
                    </div>
                    <br>
                    <div class="loginbtn">
                        <button type="submit" class="btn">Modificar</button>
                    </div>
                </form>
            </div>
        </div>

        <script> document.getElementById("invoices").style.display = "block";</script>
        <br><br>

        <script src="/js/popupscript.js"></script>
        <script src="/js/tablescript.js"></script>
</div>
```

```
<script>
    // Código JavaScript para resaltar el elemento de menú actual
    var menuItems = document.querySelectorAll('#sidebar ul li a');
    var currentURL = window.location.href;

    menuItems.forEach(function(item) {
      if (item.href === currentURL) {
        item.classList.add('active');
      }
    });
  </script>
</div>
</body>
</html>
```

Capítulo 9 Bibliografía

1. Alu. (s. f.). GitHub - alu0100769609/DXWallet: Aplicación en Flutter para el TFG-2023. GitHub. https://github.com/alu0100769609/DXWallet
2. Dart packages. (s. f.). Dart packages. https://pub.dev/
3. flutter_barcode_scanner | Flutter Package. (s. f.). Dart packages. https://pub.dev/packages/flutter_barcode_scanner
4. Joomla! Extensions Directory. (s. f.). https://extensions.joomla.org/extension/chrono-connectivity/
5. Aranda, A. (2022b, diciembre 19). Más de la mitad de los españoles con smartphone utiliza monederos digitales. *GetApp*. https://www.getapp.es/blog/3242/estudio-uso-monederos-digitales-espana#Ventajas-y-desventajas-de-los-monederos-digitales-frente-a-los-me%CC%81todos-tradicionales
6. Karakaya, H. (s. f.). How to Generate PDF Receipts from HTML in Java | PSPDFKit. PSPDFKit. https://pspdfkit.com/blog/2021/how-to-generate-pdf-receipts-from-html-in-java/
7. Material Symbols and Icons - Google Fonts. (s. f.). Google Fonts. https://fonts.google.com/icons
8. ¿Qué es el lenguaje de programación Dart? (2020, 30 octubre). inLab FIB. https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart
9. MySQL Tutorial - w3resource. (2022, 19 agosto). w3resource. https://www.w3resource.com/mysql/mysql-tutorials.php
10. Scribbr. (2022, 31 agosto). Formato con el Generador de Scribbr. https://www.scribbr.es/citar/generador/
11. Srckto. (s. f.). GitHub - srckto/E-Wallet-app-UI-flutter. GitHub. https://github.com/srckto/E-Wallet-app-UI-flutter
12. TenMas – Todo sobre la nueva tarjeta sin contacto. (s. f.). https://tenmas.es/
13. Zxing. (s. f.). GitHub - zxing/zxing: ZXing («Zebra Crossing») barcode scanning library for Java, Android. GitHub. https://github.com/zxing/zxin