TPV Virtual Móvil: Solución para Transacciones Rápidas y Seguras

Adrià Martínez Muñoz

Resumen — Se presenta el desarrollo de una innovadora plataforma diseñada para el uso de un TPV en cualquier dispositivo inteligente. Esta solución ofrece el procesamiento de pagos con tarjeta de manera efectiva y rápida sin necesidad de ningún dispositivo adicional. El proyecto se enfoca en crear un entorno seguro para los usuarios, empleando técnicas de cifrado avanzadas y cumpliendo con normativas internacionales de protección de datos, juntamente con un portal útil para la gestión de las transacciones, aportando una interfaz de información enriquecedora La flexibilidad y escalabilidad de la aplicación la hacen ideal para pequeños negocios y autónomos que buscan un sistema de pago ágil y accesible.

Palabras clave — TPV, móvil, cifrado, transacción, negocio.

The development of an innovative platform designed for the use of a POS on any smart device is presented.

Abstract — The development of an innovative platform designed for the use of a POS on any smart device is introduced. This solution provides an effective and fast card payment processing without the need for any additional device. The project focuses on creating a secure environment for users, using advanced encryption techniques and complying with international data protection regulations, along with a useful portal for transaction management, providing an enriching information interface. The flexibility and scalability of the application make it ideal for small businesses and self-employed people looking for an agile and accessible payment system.

Index Terms — POS, smart device, encryption, transaction, businesses.

E-mail de contacto: adrimm666@icloud.com
Mención realizada: Ingeniería de Software
Trabajo tutorizado por: Sergio Bachiller Rubia (Àrea de Ciències de la Computació i Intel·ligència Artificial)
Curso 2024/25

1 Introducción - Contexto del trabajo

El auge de las tecnologías en los últimos años ha impulsado un cambio significativo en las formas de pago, donde más del 70% de las compras se realizan mediante tarjetas bancarias [1]. Además, plataformas como PayPal o Bizum han introducido soluciones similares. No obstante, esta propuesta busca ir un paso más allá, facilitando pagos directos entre tarjetas bancarias y smartphones. Esta innovación ofrece nuevas oportunidades para pequeños negocios e individuos que necesitan una solución de pago rápida y discreta. Esta iniciativa esta impulsada por el proveedor de pagos Square en octubre de 2013 que tuvo gran éxito en EE. UU [2]. Por este motivo, hoy en día las entidades financieras implementan estas ventajas para facilitar las transacciones comerciales entre persona y empresa.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación móvil con la capacidad de procesar pagos mediante la tecnología NFC a través de tarjetas físicas. A partir de este propósito central, se definen varios objetivos secundarios orientados al perfeccionamiento del entorno de pago. Entre los cuales destacan: la creación de una interfaz intuitiva y eficiente para el usuario, crear una arquitectura rápida para los micropagos y la implementación de mecanismos de seguridad que garanticen la protección contra la interceptación de transacciones y la pérdida de fondos acumulados. Asimismo, se contempla el diseño de estructuras de datos robustas y seguras, capaces de almacenar de manera confiable toda la información sensible y las transacciones realizadas por cada usuario. El proyecto tendrá como misión completar los objetivos descritos hasta un cierto nivel de perfeccionamiento, se prevé poder alcanzarlos en el momento de desarrollo en el que nos encontramos. En caso de encarar alguna dificultad priorizaremos la seguridad y efectividad antes los pagos, dejando de lado el perfeccionamiento de la interfaz de usuario y funcionalidades adicionales.

1.3 Alcance

El plan propuesto tiene como meta implementar un sistema capaz de gestionar y almacenar fondos para cada cliente, permitiendo que estos puedan realizar transacciones de cobro. El proyecto se espera poder realizar multiplataforma pero se estudiarà la viabilidad en diferentes sistemas operativos. Por otro lado, tal como se establece en los objetivos, los aspectos relacionados con la seguridad y la experiencia del usuario, aunque importantes, son considerados secundarios en esta fase inicial. Por ello, se espera alcanzar el objetivo principal, y posteriormente perfeccionar la aplicación con características adicionales que garanticen una solución más completa y eficiente.

2 Metodología

Para el desarrollo de este proyecto, se adoptará una metodología híbrida que combina enfoques procedimentales y ágiles. Este enfoque permite llevar a cabo un trabajo secuencial y estructurado, abordando tareas definidas, al mismo tiempo que ofrece la flexibilidad necesaria para adaptarse a cambios y mejoras continuas durante el proceso de desarrollo. Se implementará SCRUM, con revisiones semanales del progreso para identificar y solucionar posibles obstáculos de manera ágil.
Dado que el proyecto se inicia desde cero, se establecerán procesos fundamentales que sirvan como base para la aplicación. Posteriormente, se evaluarán retrospectivas al finalizar cada sprint, lo que permitirá ajustar la carga de trabajo y definir las tareas del siguiente ciclo. Además la metodología permite el trabajo paralelo de dos campos del proyecto y que estos progresen de la mano.

Este enfoque metodológico garantiza un desarrollo organizado y controlado, facilitando la resolución de inconvenientes de manera eficiente. Al mismo tiempo, brinda flexibilidad para ajustar tanto las características técnicas como funcionales del proyecto, en función de las necesidades o cambios que puedan surgir.

3 Planificación

La división de tareas en el proyecto se organiza en 6 fases, cada una con responsabilidades afines y objetivos claros que marcarán los hitos a alcanzar. En la *Figura 1* del Apéndice se puede observar el diagrama de Gantt del proyecto con la distribución de tareas en el tiempo. La subdivisión en tareas se ha realizado en base a los diferentes componentes que mayoritariamente conforman estos tipos de aplicaciones, ligado a su esfuerzo dedicado y dependencias que surgen por tareas anteriores. Las fases son las siguientes:

1. Definición de requisitos.
2. Análisis y selección técnica.
3. Diseño del sistema.
4. Programación de módulos clave.
5. Pruebas (testing).
6. Mejora continua.

Finalizando el primer *sprint*, la planificación se esta llevando a cabo como se había previsto. Las fases iniciales han concluido de forma correcta estableciendo unas bases a seguir en el desarrollo y posterior programación.
Al finalizar los próximos sprints se evaluarán los hitos alcanzados en dichas semanas y modificar, si es necesario, las tareas o el tiempo de las mismas para proseguir con el flujo de trabajo esperado.

4 Desarrollo

4.1 Definición de Requisitos

En esta sección se definen los **requisitos** esenciales y los **casos de uso** que guiarán el desarrollo de la aplicación. Éstos describen las funcionalidades clave que el sistema debe cumplir, tanto a nivel funcional como no funcional, garantizando un entorno seguro y eficiente. Los casos de uso detallados en la *Figura 2* muestran la interacción entre clientes y el sistema, destacando los escenarios principales en los que la aplicación permitirá procesar y gestionar

transacciones. *Véase en la Figura 2 del anexo.*

Requisitos Técnicos:

1. Interfaces y Funciones de la Aplicación
   - 1.1. Pantalla "Home"
     - 1.1.1. *Dashboard* de la aplicación
     - 1.1.2. Historial de transacciones
   - 1.2. Pantalla registro/Inicio de sesión
     - 1.2.1. Gestión de usuarios
   - 1.3. Pantalla del perfil
     - 1.3.1. Modificar los datos de usuario
   - 1.4. FAQS / políticas de privacidad
   - 1.5. Pantalla **Prepago**: establece la cantidad a pagar, y habilita el pago vía NFC.
     - 1.5.1. Acceso al Hardware NFC.
   - 1.6. Pantalla **InterPago:** función de leer, autenticar la tarjeta e introducir el correspondiente PIN.
   - 1.7. Pantalla **PostPago:** carga mientras el servidor valida el pago y muestra el estado de la transacción.
2. API Pasarela de Pagos
   - 2.1. Sistema de Usuarios con JWT
     - 2.1.1. **Hashing** seguro de contraseñas
   - 2.2. Validación de transacciones
     - 2.2.1. Validación de tarjeta de crédito
     - 2.2.2. **Tokenización** de tarjetas
     - 2.2.3. Encriptación de datos sensibles
     - 2.2.4. Accionador de transacc
     - 2.2.5. Confirmación y notificación
   - 2.3. Estadísticas: informes sobre las ventas periódicamente.
   - 2.4. Uso de proveedores consolidados: Stripe, PayPal o Square
   - 2.5. Cumplimiento de estándares de seguridad PCI-DSS.
3. Servidor de Entidad Bancaria Simulada
   - 3.1. Configuración para peticiones HTTPS
   - 3.2. Sistemas de *Dummy-Users*
   - 3.3. Procesamiento de pagos y envío de fondos

   Véase la Figura 3 del anexo.
4. Modelado Base de Datos
   - 4.1. Esquematización de entidades y datos
   - 4.2. Consultas de estadísticas

   Véase en la Figura 4 del anexo.
5. Encriptación de la APP
   - 5.1. Cifrado de extremo a extremo (E2EE)
   - 5.2. TLS (Transport Layer Security)
   - 5.3. Certificados SSL renovados

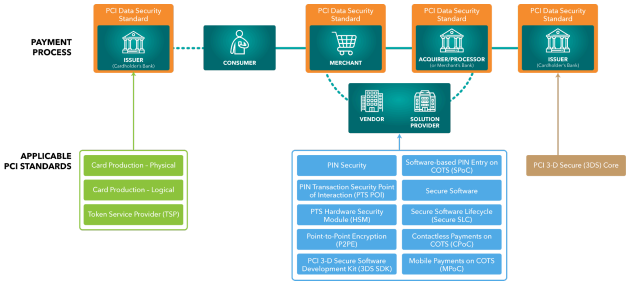

Fig. 5. Este diagrama indica los estándares de seguridad PCI aplicables. Comuníquese con las marcas de pago para obtener información sobre los programas de cumplimiento relacionados. [4]

Requisitos Operativos:

6. Licencias y Permisos Legales [5]
   - 6.1. Cumplimiento con PCI-DSS
   - 6.2. Licencia de entidad de pago
   - 6.3. Licencias por esquemas de tarjeta
   - 6.4. Cumplimiento con **PSD2**
   - 6.5. Registro ante **SEPBLAC**
   - 6.6. Cuenta Bancaria (***SandBox***)
7. Infraestructura de Servidores
   - 7.1. Hosting o despliegue en la nube

4.2 Análisis

Una vez definidos los requisitos iniciales, es fundamental abordar diversas cuestiones clave que aseguren que el sistema propuesto se ajusta plenamente a nuestras necesidades. Este análisis guiará las decisiones estratégicas, especialmente en cuanto a la **arquitectura**, **lenguaje de programación, plataforma y aspectos relevantes** :

Plataforma

Antes de comenzar el desarrollo, la elección de la plataforma sobre la cual se desplegará el TPV es crucial para facilitar la posterior programación. Además, la accesibilidad a características internas del sistema operativo y dispositivo es fundamental para cumplir los objetivos en el tiempo esperado. Si bien el objetivo principal del proyecto es ofrecer una aplicación **multiplataforma**, la primera semana de desarrollo se evaluarán los esfuerzos dedicados a funcionalidades específicas que es posible que iOS limite mayormente.
En caso de que surjan dificultades en el acceso a las funciones necesarias, se optará por limitar el despliegue a un solo sistema, **Android** que permite el acceso requerido para dichas características.

✅ Acceso completo a NFC y *Google Pay*.
❌ Diferentes versiones del SO pueden complicar el desarrollo.
Mayores retos de seguridad.

✅ Despliegue desde una única base de código.
Reducción de costes y tiempo de desarrollo.
❌ Limitaciones en el acceso a APIs o HW específicos.
Rendimiento no óptimo en comparación con el nativo.

Framework del Cliente

La previsión es el uso de un framework multiplataforma para desarrollar tanto en Android como iOS. Las opciones evaluadas son Flutter y React Native, considerando la facilidad de programación y el soporte de la comunidad.
Pero ya que la prioridad es el acceso al hardware y una fácil integración de bibliotecas y APIs externas del sistema de pago, la mejor opción es **React Native.**

✅ Código compartido entre iOS y Android.
UI fluida.
Buen rendimiento.
❌ Acceso limitado a características de hardware.
Plugins escasos o menos optimizados.

✅ Código compartido entre iOS y Android.
Amplias bibliotecas.
Fácil integración con APIs de terceros (*Stripe*).
❌ Menor acceso a hardware específico.

Arquitectura del servidor

Para las características del lado de servidor se busca una arquitectura que optimice la velocidad y la seguridad en las transacciones. *Stripe* ofrece diversas soluciones para programadores basado en APIs y SDKs que serán una fuente de información para generar una infraestructura correcta. Dicho esto, estos recursos están creados en lenguajes como Node.js, Python o Golang, por ello son evaluados para el backend de nuestra aplicación. Además, la comunicación entre módulos debe ser eficiente y segura, utilizando **HTTPS** y cifrado, asegurando un entorno fiable para el correcto tratamiento de datos sensibles [6][7]. Aquí mostramos una valoración de los diferentes Frameworks.

✅ Manejo óptimo de solicitudes I/O..
Ecosistema grande y robusto (Stripe y PayPal SDKs).
Curva de aprendizaje adecuada.
❌ Comparado con otros, no es el más seguro de base.

✅ Fácil desarrollo y mantenimiento.
Buena integración con APIs de terceros.
Bibliotecas de seguridad y manejo de datos sensibles.
❌ Mal rendimiento para alta concurrencia.

✅ Procesamiento en paralelo y alta concurrencia..
Buen soporte de seguridad y robustez.
Recursos optimizados.
❌ Más tiempo de desarrollo y complejidad.

Bajo estas demandas, **NodeJS** es el elegido para manejar el ***Backend*** de la aplicación, gracias a la capacidad de manejo de solicitudes concurrentes.
Golang, por otro lado, es el seleccionado para el **simulador de banca**. La implementación de *goroutines* para la concurrencia, y alta seguridad asegura un entorno fuerte y ágil para realizar las pruebas financieras.

Permisos Legales

El proyecto tiene un carácter financiero y por ello presenta ciertas cuestiones de ámbito legal. Para ser publicado se han de cumplir ciertos estándares legales en el desarrollo, con el fin de mantener la seguridad, el almacenamiento de fondos y de información sensible de los clientes:

- **Cumplimiento PCI-DSS**[8]: estándar de seguridad publicado por la **PCI SCC** (consejo compuesto por diferentes entidades de pago) con el propósito de mantener seguros los datos del titular de la tarjeta durante su procesamiento. Constan **12 requerimientos** obligatorios basados en el control de seguridad física, lógica y administrativa para cada entorno de pago (*Véase en la Figura 6 del anexo*).
- **Licencia de Entidad de Pago**[9]: permiso emitido por la **Autoridad Bancaria Europea (EBA)** que mantiene un registro bajo la directiva **PSD2**. Entre estas reglas, se destacan la **Autenticación Reforzada del Cliente (SCA)** que requiere autenticación multifactor para cada transacción, **Acceso de Terceros a las Cuentas (XS2A)** con el consentimiento previo, para realizar pagos o consulta de información, **transparencia** en las condiciones de los servicios de pago y **supervisión** por las autoridades competentes de cada país.
- **Licencia por Esquemas de Tarjeta**[10]: consentimiento dado por las **redes de tarjetas** bancarias como Visa o Mastercard que permite a una empresa operar como adquirente o emisor de tarjetas de pago, pudiendo así gestionar transacciones de sus clientes. No es estrictamente necesaria según la lógica del proyecto, ya que es posible colaborar con un **procesador de pagos** como *Stripe* o *Adyen*.
- **Protección de Datos**[12]: cumplimiento con Reglamento General de Protección de Datos (GDPR) de la UE que establece cómo deben manejarse los datos personales para garantizar la privacidad y protección de las personas. Como puntos claves constan el consentimiento específico del consumidor, transparencia de uso de los datos, establecer derechos de acceso como de rectificación y supresión al usuario, minimizar la cantidad de datos recopilados, juntamente a proteger la información sensible y responsabilidad y supervisión ante cualquier fallo o violación de seguridad.

Por otro lado, la compañía detrás del proyecto debe estar registrada y/o autorizada legalmente para poder practicar las funciones básicas de la aplicación.

- **Registro ante SEPBLAC**: organismo que en España supervisa y controla el blanqueo de capitales. Las empresas de servicios financieros deben registrarse y cumplir con los requisitos de la ley (**Ley 10/2010 en España**). El SEPBLAC define *"los expertos externos tienen por objeto la realización de un examen anual de las medidas de control interno en el que se detallen las medidas de control interno existentes, valore su eficacia operativa y proponga, en su caso, eventuales rectificaciones o mejoras"* [11]. Asimismo, las empresas obligadas a realizar este examen externo deben encargarlo a personas que posean la **formación académica** y la **experiencia profesional adecuadas** para asegurar la calidad y la experiencia que hagan idónea la evaluación del cumplimiento.

Seguridad

Es crucial definir cómo se almacenarán, procesarán y protegerán los datos de los usuarios, dado que las transacciones financieras tienen una naturaleza muy sensible. Por ello, el entorno debe implementar mecanismos de seguridad robustos para garantizar la integridad de las transacciones. Juntamente, para el tratamiento de los mismos datos para mantenerlos protegidos en toda comunicación entre módulos. Se definen tres aspectos clave:

- **Protección de datos**:
  - Cifrado de Base de Datos: Utilizar **AES-256** para cifrar los datos almacenados, especialmente la información sensible como los números de tarjeta y datos personales.
  - Pseudonimización: Utilizar técnicas como la **autenticación multifactor** para que los datos no sean identificables sin claves adicionales.

- **Encriptación de tarjetas y transacciones**:
  - TLS (Transport Layer Security): Asegura la transmisión segura de datos entre la app y el servidor mediante un protocolo de seguridad fiable, que cifra la comunicación en internet para prevenir intercepciones.
  - Tokenización: Emplear el uso de tokens de tarjetas para reemplazar la información sensible de la tarjeta, que no puede ser utilizada fuera del contexto de la transacción.
- **Encriptación en la Aplicación y Servidor**:
  - Cifrado de Archivos en el Servidor: Asegurar que los datos almacenados en el servidor estén cifrados con AES-256 para protección en reposo.
  - Hash-based Message Authentication Code: Aplicar claves secretas para asegurar la integridad de los datos y prevenir modificaciones no autorizadas en las transacciones.
  - Encriptación End-to-End: de cifrado que asegura que solo el remitente y el destinatario de una comunicación puedan acceder a la información.

La combinación de los diferentes lenguajes de programación y entornos aprovechará las fortalezas de cada uno para montar una infraestructura práctica, segura y competente para el TPV virtual y su funcionamiento esperado. Además, se mantendrán presentes las directrices nombradas en ámbitos de seguridad para que cada avance en el desarrollo mantenga los estándares de seguridad definidos.

4.3 Diseño del Sistema

Presentado las necesidades de nuestra aplicación y seleccionado las cuestiones primordiales de cómo se construirá, se da paso al diseño del sistema, tanto estructural como visualmente.
Lo primero a definir es la arquitectura que se levantará y mantendrá durante todo el progreso del proyecto. Detallaremos tres componentes principales:

- La interfaz de cliente.
- La API que gestiona las transacciones, sistema de usuarios y funciones estadísticas.
- El simulador de Banca que verifica dichos pagos.

Para esquematizarlo hemos elaborado **diagramas de arquitectura** que especifican la organización de los módulos clave, como el procesamiento de pagos, gestión de usuarios, comunicación con bases de datos, entre otras. La estructura se aloja alrededor de un **Servidor** en Node que trabajará como API manejando las peticiones de cada usuario en todo momento, ya sean de carácter financiero o auxiliares a la infraestructura, como el sistema de usuarios, estadísticas o manejo de eventos por parte del cliente. Este servidor estará apoyado por la **Interfaz pública** en la cual integra la lógica del servidor de una manera más intuitiva y práctica, también se añaden ciertas funciones para hacer una experiencia de usuario más atractiva. Por último, implementamos un servidor que actúa como un **Simulador Bancario**, ya que actualmente no es viable establecer una conexión con entidades bancarias reales debido a limitaciones de permisos, presupuesto y tiempo. Este simulador nos permite probar y validar la efectividad del sistema mediante casos prácticos, emulando operaciones reales, para asegurar el rendimiento y fiabilidad de la aplicación en un entorno controlado. *Véase Figura 7 en el anexo.*

Asimismo, se presentan **mockups** de las interfaces de usuario, con el objetivo de asegurar una experiencia intuitiva para el cliente final. Diseñado y prototipado con *Figma* se han llevado a cabo las siguientes interfaces, entre otras, que se observan en la *Figura 8*.

Figura 8. Interfaces del prototipo de nuestra aplicación.

Gracias a estos mockups podemos aplicar ***Prototype Testing***. Evaluaremos usuarios, se les explica acerca de la funcionalidad de nuestra aplicación y ofreceremos a que puedan probar de primera mano la usabilidad de la misma. En el siguiente paso observaremos como el usuario efectúa el flujo de trabajo establecido o, por otra parte, detecta problemas de usabilidad, se recopila toda la información necesaria para ajustar las interfaces y flujos antes de la implementación final.

Prototype testing

En las pruebas realizadas a ciertos usuarios de diferentes edades, se han encontrado ciertos patrones de error o mejora en la aplicación. Debido a la naturaleza de ser un prototipo básico de nuestro proyecto muchos usuarios no gozaban la posibilidad de tener todas las opciones clicables, como botones o desplegables entre otros, y este fue uno de los **errores** más comunes. Además, el flujo de pago se percibe claro pero no cumplió con las expectativas, ciertas pruebas mostraban desconcierto de como avanzar o volver a un estado inicial después de formalizar el pago o durante el mismo. Finalmente, los *testers* daban recomendaciones acerca de completar las pantallas con información más extensa para conocer el estado de la aplicación en todo momento, ya sean iconos, mensajes, animaciones, etc. Gracias a esta información obtenida, se proponen ciertos puntos de **mejora**:

- Como se espera de la versión final, dotar de funcionalidad a todo elemento clicable.
- Aportar **más iconos o botones** para mejorar la navegación y la ejecución de ciertas funciones.
- Mostrar **mensajes emergentes** durante procesos críticos dentro de nuestra aplicación.
- Dar **más información** en las **pantallas de pago** para conocer el estado de nuestra transferencia.

La implementación de estas mejoras permitirá desarrollar una aplicación más completa y alineada con las necesidades de los usuarios.

Durante este proceso, se está construyendo un **mapa conceptual** del desarrollo completo del proyecto. Éste funciona como una guía estructurada que servirá para verificar que los avances se ajusten a las expectativas y objetivos previamente establecidos. Con ello, se asegura un enfoque iterativo en la mejora continua de la aplicación.

4.4 Programación

Durante esta fase se implementa el **código fuente** de los módulos centrales del sistema, como el motor de procesamiento de pagos y el cifrado de datos sensibles.

Fase 0: Montaje Inicial del Proyecto

Se comienza creando los diferentes entornos en el lenguaje correspondiente, junto con una configuración inicial básica. Cada módulo se diseña para poseer una responsabilidad específica, organizando las funcionalidades mediante rutas y controladores.
Instalación de herramientas esenciales:

- **Node.js LTS y Microsoft OpenJDK 17**.
- **Android Studio**.
- **React Native CLI**.
- **Golang**.
- **MySQL y PostgreSQL** para las bases de datos.

Configuración inicial de los módulos:

- **Frontend**: Se modifica el fichero App.tsx para empezar a introducir código y se eliminan archivos innecesarios.
- **Backend**: Se configura un nodo raíz que escucha en una URL (IP:PORT) y redirige a un enrutador principal.
- **Simulador**: Similar al backend, se configura un nodo raíz que escucha en una URL (IP2:PORT2) y redirige a un enrutador principal.
- **Bases de datos**: Se ejecutan archivos SQL para crear cada entidad y sus columnas. Adicionalmente, se inicializa información inicial como datos dummy para el registro de cuentas bancarias simuladas.

Primera versión - Fecha objetivo: 15/12/2024

La primera entrega de la aplicación incluye el maquetado de las pantallas principales, el sistema de usuarios y el lector de NFCs. También se configuran las entidades de la base de datos y las necesarias en el simulador para ejecutar transacciones.
El backend sigue una arquitectura **Modelo-Vista-Controlador** (MVC), mientras que el simulador de banca utiliza un sistema de capas para la gestión, autenticación y ejecución de funciones.

1. Servidor:

Basado en el patrón MVC, el flujo del sistema se organiza de la siguiente manera:
App → Routes → Controller → Model → Sequelize

- **Modelo**: Se utiliza *Sequelize* para crear instancias de entidades en la base de datos.
- **Controlador**: Implementa la lógica de negocio.
- **Vista**: Los enrutadores de cada entidad definen las rutas de acceso y los mensajes de respuesta del servidor.

Cada entidad (*User, User Bank Account, Transaction*) sigue esta estructura Ruta/Controlador/Modelo, con excepción del Controlador de Estadísticas, que se alimenta del Modelo de Transacciones. Adicionalmente, se implementa la gestión de usuarios y la verificación mediante **Json Web Token (JWT)** en el lado del servidor.
El flujo del sistema de pagos se lleva a cabo replicando un proveedor de pagos al uso como el conocido *Stripe*, para garantizar la totalidad y eficiencia en la gestión de pagos.

2. Simulador Bancario:

Este servidor utiliza una arquitectura de capas debido a sus funciones limitadas y específicas. Sus módulos principales son:

- Autenticador de pagos: Valida la información proporcionada por Stripe, asegurando que los datos de la transacción sean correctos y verificando la disponibilidad de fondos.
- Accionador de transferencias: Ejecuta transferencias entre clientes del banco.

3. Frontend:

En la aplicación de usuario, las rutas de navegación se estructuran mediante tres Navigators:

- **RootStack Navigator**: Define las rutas principales de la aplicación, incluyendo:
  - Main: Contiene el *BottomTab Navigator*.
  - Pantallas de perfil, inicio de sesión y registro.
  - Enrutador *Payment Navigator*.
- **Bottom Tab Navigator**: Establece una barra de navegación inferior con tres funciones principales:
  - Pantalla Home.
  - Pantalla de Estadísticas.
  - Configuración de la aplicación.
- **Payment Navigator**: Gestiona las pantallas relacionadas con el proceso de pago, encargándose de:
  - Teclado numérico para configurar importe y PIN.
  - Leer tarjetas vía NFC.
  - Proporcionar respuestas al usuario sobre el éxito de los pagos.

Además, se planea reorganizar o añadir opciones de navegación en futuras versiones para mejorar la estructura de la aplicación.
Funcionalidades principales:

- Funcionalidades de usuario en producción.
- Almacenamiento local de datos de configuración y tokens.
- El accionador de pagos está diseñado para funcionar una vez se habilite la captura de información de tarjetas de crédito.

Segunda versión - Fecha objetivo: 19/01/2025

El desarrollo con fecha para la segunda versión corrige errores importantes respecto a la primera. El lector de tarjetas ya está en producción y formaliza el pago como esperado, así con un historial es posible continuar con el

desarrollo de otras funciones como el retorno de un pago o la visualización de estadísticas por parte del usuario. Además se incluyen mejoras en el diseño de la aplicación, en aspectos de seguridad y en la optimización de modelos y lógica interna del servidor.

1. Ciclo de Pago

El lector de tarjetas NFC se encuentra completamente integrado y en producción, permitiendo que las transacciones se formalicen de manera fluida y segura. Los usuarios ahora pueden iniciar un pago introduciendo la cantidad deseada, tras lo cual el pagador externo aproxima su tarjeta al lector NFC. La validación del PIN se realiza inmediatamente, tras lo cual se comunica con el servidor para completar la operación a través de un proveedor de pagos externo (3rd Party). Este flujo asegura la rapidez y confiabilidad de cada transacción.

2. Funciones Secundarias

Se han desarrollado nuevas herramientas para mejorar la experiencia del usuario y ofrecer información valiosa sobre las transacciones. Entre estas se incluyen:

- **Historial de transacciones**: Listado detallado de las operaciones pasadas.
- **Información del balance**: Visualización del balance de los últimos siete días, con desglose de ganancias diarias.
- **Gestión de transacciones**: Un menú desplegable permite realizar acciones como la devolución de pagos y observar información más detallada sobre la misma.
- **Estadísticas temporales**: Reportes gráficos y números de ingresos anuales, mensuales y semanales para un análisis más detallado.

3. Mejoras en el Diseño

El diseño de la aplicación ha sido optimizado para ofrecer una experiencia visual más intuitiva y atractiva:

- **Reestructuración de componentes**: Los elementos visuales han sido reorganizados para maximizar su reutilización y mejorar el rendimiento general.
- **Cambios estéticos**: Ajustes en la paleta de colores, diseño responsivo para adaptarse a diversos tamaños de pantalla y la incorporación de nuevos recursos gráficos.
- **Patrones de diseño**: Se aplican principios de diseño modular para garantizar que cada componente cumpla con sus responsabilidades sin sobrecargar la memoria o efectividad del dispositivo.

4. Refuerzo de la Seguridad

Dada la naturaleza crítica de las operaciones de pago, la seguridad sigue siendo una prioridad:

- **Protección de datos sensibles**: Toda la información crítica es encriptada y tokenizada antes de su almacenamiento, minimizando el riesgo de filtraciones.
- **Certificados de seguridad**: Se han añadido certificados SSL/TLS para garantizar una comunicación segura entre el frontend y el backend, blindando las transacciones contra posibles ataques.
- **Escenarios de contingencia**: Se han analizado posibles vulnerabilidades para mitigar riesgos de interrupciones o accesos no autorizados.

La segunda versión del proyecto TPV móvil se encamina hacia una solución sólida que combina funcionalidad, diseño atractivo y estándares de seguridad. Al priorizar la implementación de características críticas como la lectura NFC y las herramientas de gestión, el proyecto garantiza un producto funcional. A medida que se avanza hacia las siguientes fases, la aplicación se perfila como una solución integral que responderá con éxito a los requisitos definidos en las bases del proyecto.

4.5 Testing

Una vez programados los módulos clave, se da inicio a la fase de pruebas, un componente esencial para garantizar la estabilidad y funcionalidad de la aplicación. Este proceso incluye pruebas unitarias, de integración y de seguridad, las cuales se alinean con los sprints establecidos para mantener un desarrollo controlado y predecible. El objetivo principal es identificar y resolver errores, asegurando que cada módulo opere correctamente, tanto de forma independiente como en conjunto.

Pruebas Unitarias

Las pruebas unitarias se centran en escenarios concretos y limitados, verificando la funcionalidad de segmentos específicos del código. Utilizamos herramientas externas como Postman para ejecutar funciones específicas alojadas en URLs del servidor. En esta etapa, se crean casos de prueba basados en los formatos requeridos por las solicitudes, simulando escenarios reales para analizar cómo el sistema gestiona los datos y ejecuta las funciones previstas. Estos tests permiten detectar y corregir problemas a nivel de implementación, asegurando que cada componente responda correctamente a las solicitudes individuales.

Pruebas de Integración

Las pruebas de integración se enfocan en validar la interacción entre los diferentes módulos y elementos del sistema. Estas pruebas simulan escenarios reales en los que las funcionalidades principales, como los ciclos de pago, devoluciones y visualización de datos, son probadas bajo diversas condiciones. En este proceso, conectamos los distintos componentes del sistema, verificando tanto la navegación como la correcta transferencia y procesamiento de información. Los resultados exitosos confirman que los módulos individuales funcionan correctamente en conjunto y cumplen con las especificaciones establecidas.

Pruebas de Seguridad

Las pruebas de seguridad constituyen un aspecto crucial, especialmente para una aplicación que maneja información sensible y transacciones financieras. Durante esta fase, se simulan contextos que podrían comprometer la seguridad de los datos, incluyendo intentos de acceso no autorizado, intercepción de información sensible y vulnerabilidades en la comunicación entre el cliente y el servidor.

Aquí tienes tres casos de pruebas de seguridad comunes, prácticas y fáciles de implementar para proteger tu

aplicación frente a ataques mayoritarios:
Pruebas de Fuerza Bruta: Detectar si el sistema puede bloquear ataques automatizados destinados a adivinar credenciales (contraseñas o PINs). Un script se encarga de reproducir intentos repetitivos de inicio de sesión y flujos principales del sistema para observar si algún entorno se bloquea debido a la sobrecarga.
Pruebas de Inyección SQL: Identificar vulnerabilidades que permiten manipular bases de datos mediante inputs maliciosos.
Pruebas de Comunicación Segura: Comprobar si los datos en tránsito están protegidos contra interceptación. Con herramientas como *Wireshark* se analiza el tráfico entre cliente y servidor.
Estas pruebas cubren vulnerabilidades críticas y son una excelente base para garantizar la seguridad de la aplicación en las áreas más expuestas a ataques comunes.

El proceso de testing asegura que la aplicación no solo sea funcional, sino también segura, estable y confiable. La combinación de pruebas unitarias, de integración y de seguridad permite identificar y resolver problemas en todas las capas del desarrollo. Al abordar de manera rigurosa cada una de estas etapas, estamos construyendo una solución robusta que responde a los estándares más altos de calidad, brindando confianza tanto a los usuarios finales como a los socios estratégicos. Esto posiciona al proyecto como una herramienta innovadora y confiable en el ecosistema de pagos móviles.

4.6 Mejora Contínua

El proyecto, con sus dos versiones iniciales, ha mostrado un desarrollo adecuado, sin embargo, una estrategia de mejora continua puede garantizar que la aplicación evolucione para satisfacer las necesidades de los usuarios, mantener la competitividad en el mercado y cumplir con los estándares legales y técnicos.

Evaluación de las Versiones Anteriores

Áreas de Mejora:

- Limitación en las funcionalidades críticas como la devolución de pagos y estadísticas.
- Diseño inicial básico que podría optimizarse para una mejor experiencia de usuario.
- Seguridad básica implementada, pero sin simulación de ataques complejos o integraciones avanzadas de encriptación.
- La gestión de usuarios y transacciones, aunque funcional, podría beneficiarse de una mayor automatización y soporte para casos de uso complejos.
- La interfaz gráfica, aunque mejorada, aún puede beneficiarse de ajustes adicionales para garantizar la accesibilidad y usabilidad.
- Escalabilidad limitada en los módulos, lo que podría ser un desafío a medida que crezca la base de usuarios.

Estrategia de Mejora Continua

La integración con un proveedor de pagos como Stripe, Adyen, o PayPal es esencial para garantizar transacciones seguras, confiables y globalmente aceptadas. Además ofrece beneficios clave, como el cumplimiento inmediato con estándares de seguridad PCI DSS, acceso a herramientas avanzadas para la gestión de pagos, devoluciones y reportes, así como la facilidad de expansión a mercados internacionales. Para su implementación, es necesario configurar la API del proveedor para manejar los flujos de transacciones, realizar pruebas exhaustivas en entornos sandbox antes del lanzamiento en producción y aprovechar las herramientas de prevención de fraudes y autenticación avanzada que estos servicios ofrecen.

Optimización del Flujo de Pago

Se proponen diferentes funcionalidades auxiliares para mejorar la experiencia de usabilidad, algunas opciones son introducir opciones de pago alternativas como códigos QR, wallets digitales y transferencias inmediatas o implementar notificaciones en tiempo real para informar a los usuarios sobre el estado de sus transacciones.
Por otro lado, en cuestiones técnicas se debe analizar la latencia en el procesamiento de pagos, y optimizar los servicios del servidor para una respuesta más rápida.

Escalabilidad del Sistema

Se propone migrar a un entorno de microservicios para gestionar los módulos de forma independiente, lo que mejorará la escalabilidad y el mantenimiento del sistema. Además, la implementación de bases de datos distribuidas optimizará el rendimiento y garantizará una mayor disponibilidad. Por último, se planea incorporar herramientas de monitoreo como Prometheus o New Relic para identificar y resolver cuellos de botella de manera proactiva.

Fortalecimiento de la Seguridad

Se recomienda realizar auditorías de seguridad periódicas para identificar y mitigar vulnerabilidades de forma proactiva. Además, la incorporación de autenticación multifactor (MFA) reforzará la protección del acceso de los usuarios. Por último, el uso de algoritmos avanzados garantizará la encriptación robusta de datos sensibles tanto en tránsito como en reposo, mejorando significativamente la seguridad del sistema.

Visión Futura

Con la integración de un proveedor de pagos reconocido y la implementación de una estrategia de mejora continua, el proyecto se perfila para ofrecer un sistema robusto y escalable. El enfoque en seguridad, experiencia de usuario y escalabilidad permitirá que la aplicación no solo cumpla con los estándares del mercado, sino que también se posicione como una solución confiable e innovadora en el ecosistema de pagos móviles.

5 RESULTADOS

En base en el desarrollo descrito, la valoración de los resultados obtenidos puede ser sintetizada en los siguientes puntos:

Avances Técnicos y Funcionales

- **Hitos Iniciales**: El desarrollo demuestra la integración del lector NFC y la estructura del flujo de pagos. Se toma como referencia un TPV físico y se propone una solución lo más fehaciente a la realidad. Los logros obtenidos cumplen con los objetivos propuestos en las fases tempranas, a pesar de no tener una aplicabilidad real, ya que la integración con un proveedor como *RedSys* o *Stripe* supone un desarrollo más extenso y dificultades en temas legales que no permite abarcar el proyecto.

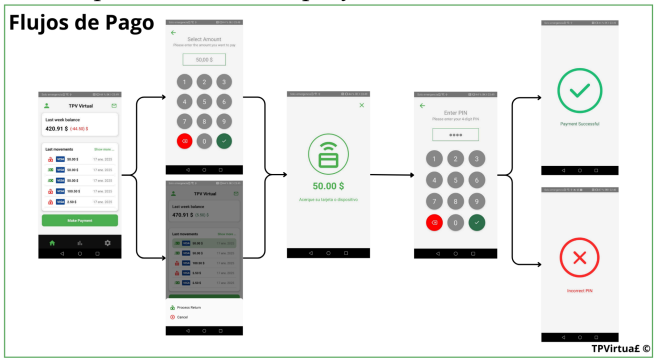

Figura 9. Flujo de pago de la aplicación.

- **Funcionalidades Esenciales**: A banda del sistema de pagos, hay otros módulos que son necesarios para nuestro entorno como la gestión de usuarios, o el *3rd Party* que realiza la lógica simulada de una entidad bancaria. Para ello, se forman otros flujos de trabajo para cumplir con las necesidades del proyecto. Un caso particular es la entidad bancaria que se replica de manera simple pero eficaz un manejador de verificaciones sobre saldos bancarios, para que en nuestra aplicación la transacción se realice.
- **Seguridad:** Debido a una cuestión de tiempo y recursos, la seguridad ha pasado a un segundo plano en el desarrollo. Se aplican directrices de seguridad en el autenticado, procesamiento de tarjetas y en datos sensibles pero la aplicación no logra cumplir con un grado de seguridad elevado dado la naturaleza financiera de la aplicación. Para obtener mejores resultados en este apartado, se debería formar un equipo dedicado a resolver y mitigar posibles intrusiones maliciosas, y así lograr una alta calidad en el producto final.

Estructura y Usabilidad

- **Interfaz Eficiente**: Las mejoras en el diseño visual han optimizado la experiencia del usuario, garantizando que las funcionalidades principales sean entendidas por cualquier usuario.
- **Estrategia Modular**: La arquitectura basada en patrones de diseño, como MVC y sistemas de capas, ha permitido una organización clara y reutilizable, facilitando tanto el mantenimiento como el escalamiento futuro del sistema.

Resultados en Pruebas y Validación

- **Pruebas Exitosas**: Las pruebas unitarias e integradas han demostrado la funcionalidad y estabilidad de los módulos clave. Además, el planteamiento de ciertas pruebas de seguridad han contribuido a identificar y mitigar riesgos potenciales, aplicando buenas prácticas de encriptación y confidencialidad de los datos.
- **Evidencia de Viabilidad**: El testing continuo y los resultados observados confirman que el proyecto puede ser implementado de manera eficiente en un entorno controlado, ofreciendo una solución sólida. A partir de este fase beta de la aplicación y con un enfoque de producción, se plantea la refactorización con microservicios que habiliten el pago con entidades financieras, la consulta legal de protección de datos y avales al *SEPBLAC* y por último, autorías de seguridad con personal experto para garantizar un sistema sin penetración por agentes maliciosos.

Valoración Final

Los resultados obtenidos reflejan un avance importante hacia el producto final de un ecosistema de pagos móviles. A pesar de las áreas en las que aún se requiere trabajo adicional, ya demuestra características necesarias de un producto robusto. La atención al detalle en diseño y estructura del sistema permiten continuar el desarrollo con una visión de negocio, y más adelante y con buenas prácticas poder salir al mercado. Estos puntos posicionan al sistema como una herramienta confiable y lista para entrar en su etapa de consolidación y establecer casos de uso del mundo real.

4 Conclusión

El proyecto de TPV móvil ha logrado alcanzar un sistema estable que permite procesar pagos de manera sencilla con las tecnologías de NFC que suponían un reto al inicio del proyecto. Con la arquitectura descrita proporciona una solución práctica para atender las necesidades de pago móvil, y junto a un diseño intuitivo la aplicación ha sido modelada para ofrecer una experiencia de usuario excepcional. No cabe añadir que los resultados de las pruebas pruebas han permitido detectar áreas clave de mejora en usabilidad y navegación, consolidando la funcionalidad y el atractivo del producto. Por otro lado, algo en lo que la aplicación aún se considera primitiva es en aspectos de seguridad. En el desarrollo se utilizan técnicas de encriptación y procediendo con buenas prácticas, pero dado el carácter financiero de la aplicación no es posible asegurar la total protección de datos o la intrusión de algún agente externo.

Como bien se valora anteriormente, es un proyecto que necesita de más recursos para llevarse a cabo como una solución real. Existen diversos factores que pueden alterar el funcionamiento normal de la aplicación o por otro lado filtrar una información privada y comprometida. Por ello, y con visión de futuro, un equipo se debería coordinar con un mismo propósito y obtener fuentes de inversión para promover y producir un producto final.En definitiva, este proyecto se perfila como una solución robusta que enriquecerá la experiencia de pago móvil. Con una visión clara y un compromiso firme, definiría como conclusión de este proyecto, la semilla de un posible gran árbol en un futuro.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas y organizaciones que han contribuido al desarrollo de este proyecto.

En primer lugar, agradezco a Sergio Bachiller Rubia por permitirme desarrollar mi idea y haber escogido mi proyecto, y sobre todo por su orientación técnica durante las etapas más tempranas del proyecto. A la universidad, por proporcionarme los conocimientos necesarios para llevar a cabo este trabajo. También extiendo mi gratitud a mis padres y mi pareja por su colaboración en fases del desarrollo y obtener valoraciones muy positivas para el desarrollo y la mejora de la aplicación.

Bibliografía

[1] F. Javier Goya Olivera y E. Maroto Almarcha. “Estudio “Hábitos de pago en Europa” realizado por Oney”. www.oney.es. Accedido el 25 de septiembre de 2024. [En línea]. Disponible: https://www.oney.es/wp-content/uploads/2021/05/Informe_resultados_encuest

[2] SquareUp. “Acepta pagos con Cash App Pay | Centro de Atención al Cliente de Square - US”. Power your entire business | Square. Accedido el 27 de septiembre de 2024. [En línea]. Disponible: https://squareup.com/help/us/es/article/7635-accept-payments-with-cash-app-pay#:~:text=Cash%20App%20 Pay%20simplifica%20las,realizar%20el%20pago%20Cash%20App

[3] Asana. “Las 12 metodologías más populares para la gestión de proyectos [2024] • Asana”. Asana. Accedido el 29 de septiembre de 2024. [En línea]. Disponible: https://asana.com/es/resources/project-management-methodologies

[4] PCI Security Standards ORG. (s.f.). The PCI Security Standards Ecosystem [Imagen]. PCI Security Standards Council – Protect Payment Data with Industry-driven Security Standards, Training, and Programs. https://www.pcisecuritystandards.org/wp-content/uploads/2024/07/standards-ecosystem.png

[5] Servicios de pago y otras medidas urgentes en materia financiera., Real Decreto-ley n.º 19/2018 (2023) (España). https://www.boe.es/buscar/act.php?id=BOE-A-2018-16036

[6] Usuario. “Node.js vs Golang: Which is Better for Backend Development? - Sling Academy”. Sling Academy. Accedido el 28 de octubre de 2024. [En línea]. Disponible: https://www.slingacademy.com/article/nodejs-vs-golang-backend-development-comparison/

[7] S. Rane. “What are the Differences Between Go, Node.js, and Python? A Comprehensive Comparison – SourceBae”. Sourcebae | Hire Top Remote Developers & Engineers in 2024. Accedido el 28 de octubre de 2024. [En línea]. Disponible: https://sourcebae.com/blog/what-are-the-differences-between-go-node-js-and-python-a-comprehensive-comparison/#:~:text=In%20the%20realm%20of%20programming,for%20its%20versatility%20and%20readability.

[8] D. Acosta. “¿Qué es PCI DSS? | PCI Hispano”. PCI Hispano | Análisis de los estándares del PCI SSC en español (PCI DSS, PCI 3DS, PCI PIN, P2PE, PCI SSF, PCI PTS, PCI TSP). Accedido el 8 de noviembre de 2024. [En línea]. Disponible: https://www.pcihispano.com/que-es-pci-dss/

[9] European Banking Authority. “Register of payment and electronic money institutions under PSD2 | European Banking Authority”. Homepage | European Banking Authority. Accedido el 8 de noviembre de 2024. [En línea]. Disponible: https://www.eba.europa.eu/risk-and-data-analysis/data/registers/payment-institutions-register

[10] Stripe. “Guía sobre tipos de métodos de pago”. https://stripe.com/es/guides/payment-methods-guide. Accedido el 8 de noviembre de 2024. [En línea]. Disponible: https://stripe.com/es/guides/payment-methods-guide

[11] SEPBLAC. “Definición y requisitos”. https://www.sepblac.es/es/expertos-externos/definicion-y-requisitos/. Accedido el 8 de noviembre de 2024. [En línea]. Disponible: https://www.sepblac.es/es/expertos-externos/definicion-y-requisitos/

[12] Europa, Unión Europea. (2016, 27 de abril). Reglamento (UE) 2016/679 n.º 2016/679, Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE. Accedido el 8 de noviembre de 2024. [En línea]. Disponible: https://eur-lex.europa.eu/ES/legal-content/summary/general-data-protection-regulation-gdpr.html

[13] Meta Open Source. (2024, 23 de octubre). Set Up Your Environment · React Native. reactnative.dev. https://reactnative.dev/docs/set-up-your-environment

[14] Esteban, D. (2023, 23 de diciembre). Autenticación en Node.js con JSON Web tokens y Express. Medium. https://medium.com/@diego.coder/autenticación-en-node-js-con-json-web-tokens-y-express-ed9d90c5b579

APÉNDICE

A1. Figura 1: Diagrama de Gantt

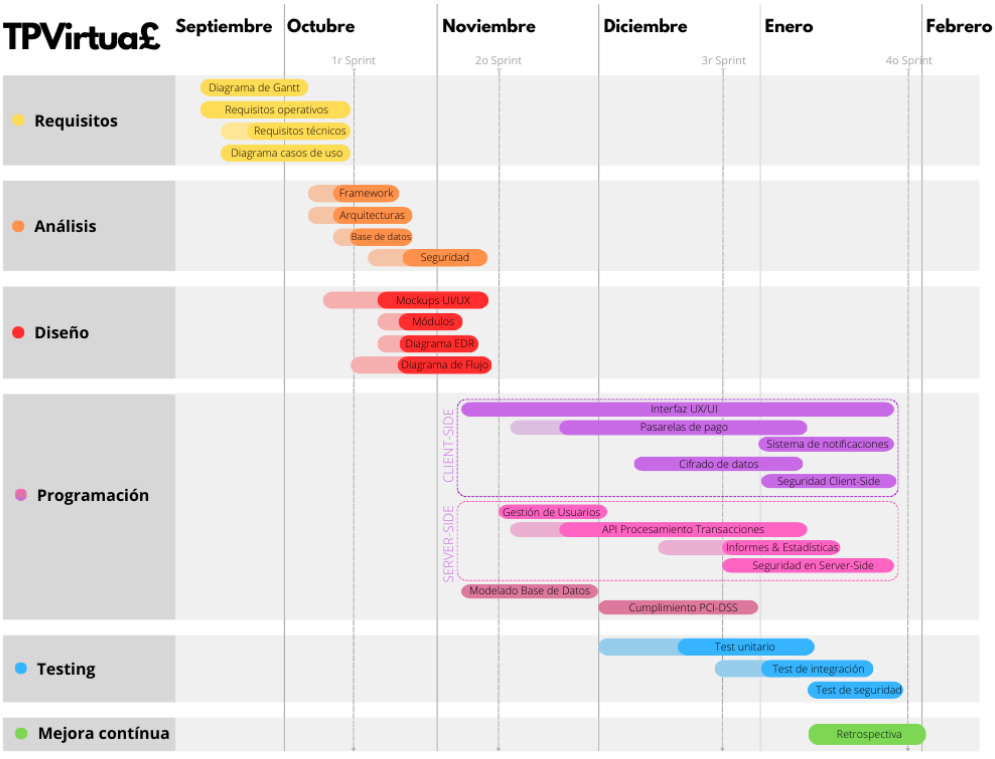

Fig. 1. Diagrama de Gantt que muestra la planificación del proyecto.

A2. Figura 2: Diagrama de casos de uso

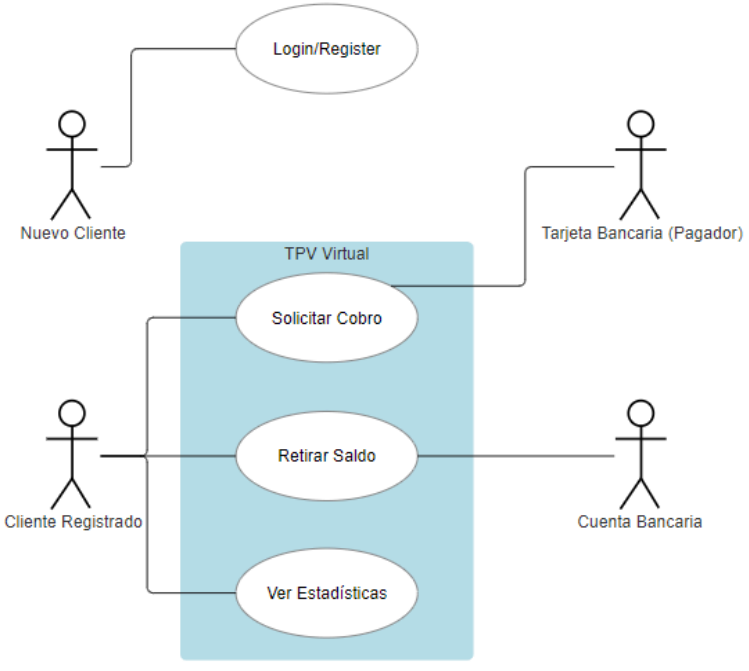

Fig. 2. Diagrama de Casos de Uso de la aplicación.

A3. Figura 3: Diagrama de secuencia

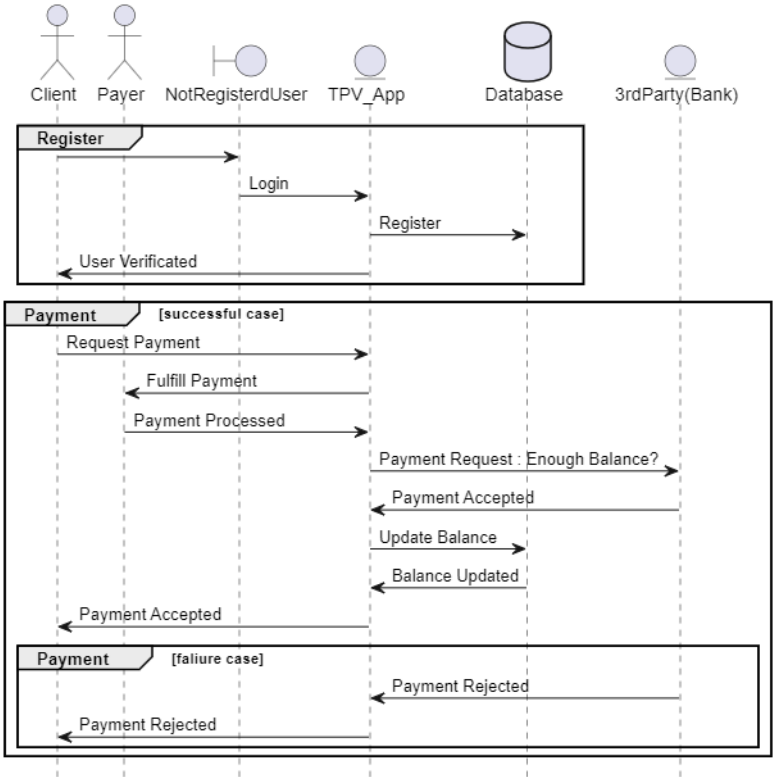

Fig. 3. Diagrama de Secuencia - Flujo de uso de los diferentes módulos.

A4. Figura 4: Diagrama Entidad - Relación de las bases de datos del sistema.

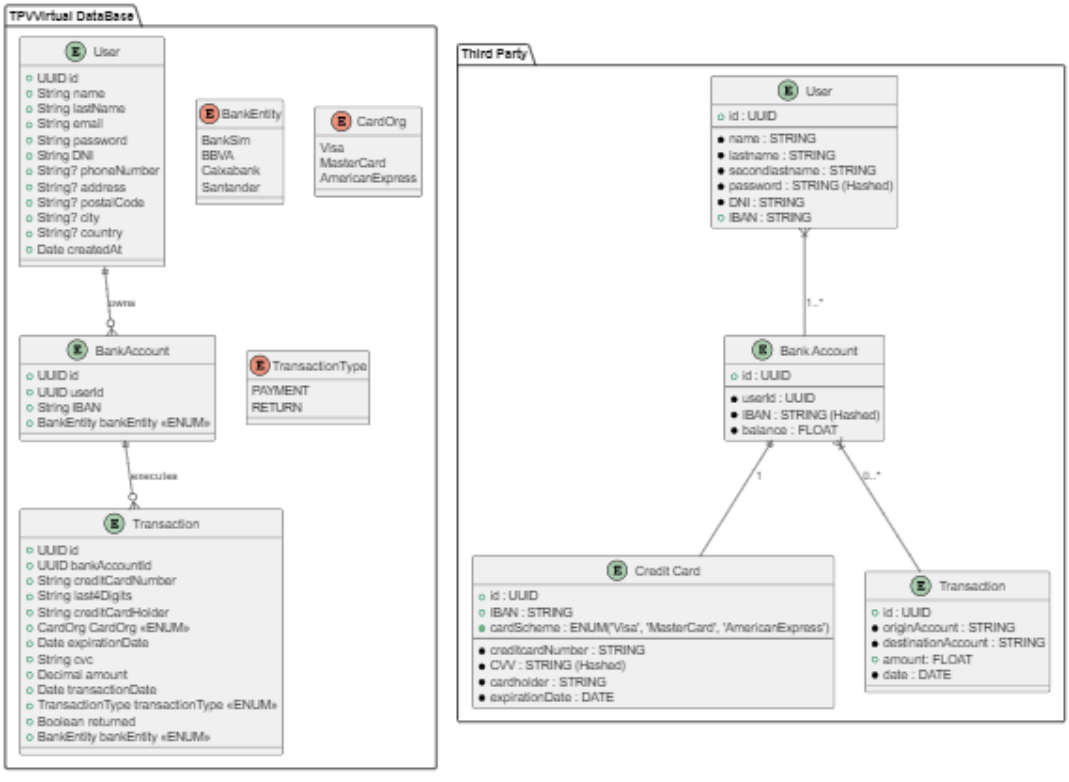

Fig. 4. Diagrama de Entidad - Relación de la Base de Datos.

A5. FIGURA 6: CONTROLES DE SEGURIDAD DEL ESTÁNDAR PCI DSS

Construya y Mantenga Redes y Sistemas Protegidos	Proteja los datos del titular de la tarjeta	Mantenga un Programa de Gestión de Vulnerabilidades	Implemente Medidas Sólidas de Control de Acceso	Monitorear y Verificar las Redes Regularmente	Mantenga una Política de Protección Informática
1. Instale y Mantenga Controles de Seguridad en la Red 2. Aplique Configuraciones Protegidas para Todos los Componentes del Sistema	3. Proteja los Datos de Cuenta Almacenados 4. Proteja los Datos del Titular de la Tarjeta con una Sólida Criptografía Durante la Transmisión a Través de Redes Públicas Abiertas	5. Proteja Todos los Sistemas y Redes de Software Malintencionado. 6. Desarrolle y Mantenga Sistemas y Softwares Protegidos	7. Restrinja el Acceso a los Componentes del Sistema y a los Datos del Titular de la Tarjeta Según las Necesidades Comerciales 8. Identifique a los Usuarios y Autentique el Acceso a los Componentes del Sistema 9. Restrinja el Acceso Físico a los Datos del Titular de la Tarjeta	10. Registre y Monitorear Todo el Acceso a los Componentes del Sistema y a los Datos del Titular de la Tarjeta. 11. Verifique la Seguridad de los Sistemas y Redes Regularmente	12. Respalde la Protección Informática con Políticas y Programas Organizacionales.

Fig. 6. Controles de seguridad del estándar PCI DSS [9].

A6. FIGURA 7: DIAGRAMA DE COMPONENTES

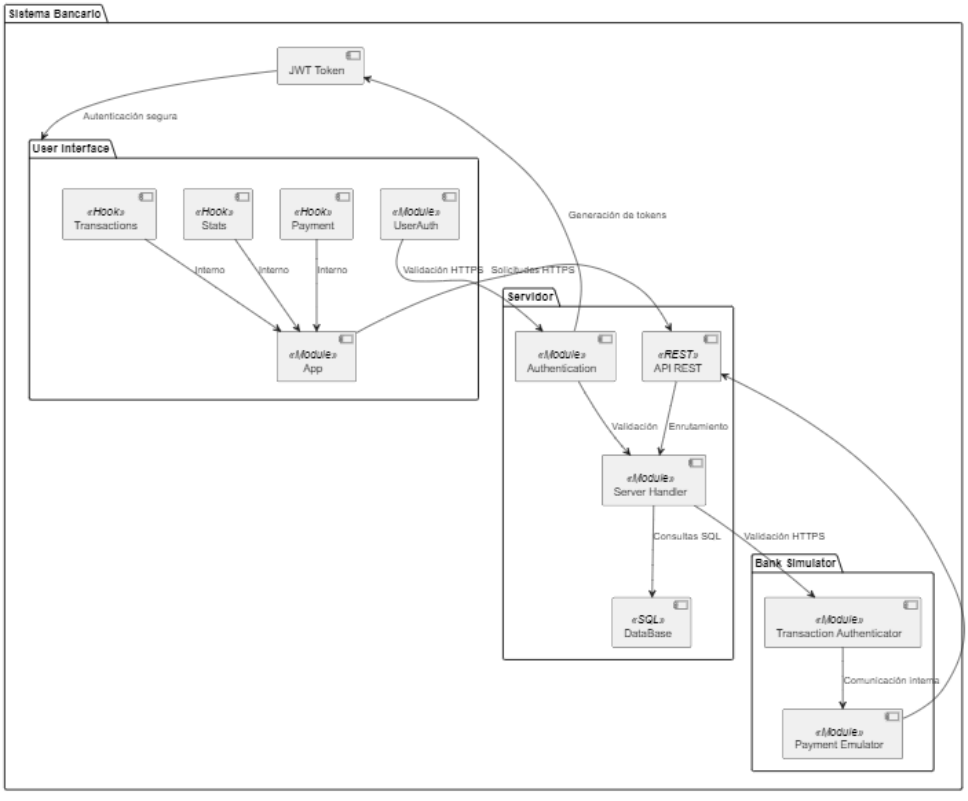

Figura 7. Diagrama de Componentes que forman parte de la app.