# Proyecto Wallet | Java 

## Situación inicial 📍

Somos un equipo de desarrollo y nuestro líder técnico nos comparte el pedido de un nuevo cliente que está precisando de nuestros servicios. Se trata de AlkyBank, un banco que busca innovar y mejorar la experiencia del usuario. Con este objetivo en mente, Matías, líder del Centro de Innovación de AlkyBank, quiere impulsar el desarrollo de una Wallet digital para integrar en el Homebanking de sus clientes y facilitar sus transacciones de una manera rápida, fluida y segura.

## Tu objetivo 📋

Como parte de un equipo de desarrolladores, deberán desarrollar una Wallet que le permita a los clientes del banco realizar transacciones de forma virtual y facilitar la operaciones entre personas.

## Requerimientos 🔧

Se deberá desarrollar una Wallet utilizando **Java 17 con Spring Boot**. Esta consta de autenticación, carga de saldo, gastos, balance, movimientos y envío de dinero.

Las diferentes funcionalidades las podemos agrupar en:

1. Login
2. Carga de saldo
3. Gastos
4. Balance
5. Movimientos
6. Envío de dinero
7. Inversiones
8. Documentación
9. Testing

## A tener en cuenta 📣

Te recomendamos seguir el Google Java Style Guide a la hora de codificar para mantener las buenas prácticas.

**Algunas tools que recomendamos para utilizar son 🛠️**

- ➔ Intellij cómo IDE, aunque podés usar cualquiera con el que te encuentres familiarizado.
- ➔ Postman para probar los endpoints.
- ➔ Algún visor de base de datos cómo por ejemplo DBeaver.

## 📌 Funcionalidades por sección

**Login**

- ❖ Una vez logueado permite ver cargas de saldo, balance, gastos, etc
- ❖ En caso de no tener cuenta, permitir crearle una.
- ❖ Se deberá implementar Spring Security.
- ❖ Las claves deben estar encriptadas.

**Carga de saldo**

- ❖ Permitir realizar una carga de saldo en la cuenta (depósito).
  - ➢ Las cargas deben tener un monto mayor que cero.
  - ➢ Las cargas deben tener un concepto (texto corto).
  - ➢ Las cargas deben tener una fecha.
  - ➢ Las cargas deben especificar una moneda.
  - ➢ Solo se puede editar el concepto de una carga.

**Gastos**

- ❖ Permitir cargar pagos para poder llevar control de los egresos de dinero.
  - ➢ Los pagos deben tener concepto (texto corto).
  - ➢ Los pagos deben tener un monto mayor que cero.
  - ➢ Los pagos deben tener una fecha.
  - ➢ Los pagos deben especificar una moneda.
  - ➢ Solo se puede editar el concepto de un pago.

**Balance**

- ❖ Poder obtener el balance para saber cuánto dinero tengo en la cuenta.
  - ➢ Se debe mostrar balance como la suma de todos las cargas menos la suma de todos los pagos.
  - ➢ Se debe mostrar los balances de ambas cuentas.
  - ➢ Tener en cuenta los plazos fijos que puedan estar creados. De existir deben ser mostrados.

**Movimientos**

- ❖ Listar todos los movimientos para llevar control de los ingresos y egresos de dinero.
  - ➢ Se deben listar todos los movimientos desde el más nuevo al más viejo

- Se deben paginar cada 10 movimientos.

- Opcional:
  - Se debe poder filtrar por carga/pago.
  - Se debe poder filtrar por concepto .
  - Se debe poder filtrar por moneda.

**Envío de dinero**

- Permitir enviar dinero a otro usuario para que tenga más ingresos.
  - Al enviar dinero, el dinero aparecerá como un gasto cualquiera, pero NO puede ser modificado.
  - Al enviar dinero, ese dinero aparecerá en la lista de ingresos del usuario al que se lo envíe. NO puede ser modificado.

**Inversiones**

- Permitir realizar un plazo fijo para poder invertir una suma de dinero.
  - El plazo mínimo de un plazo fijo será de 30 días.
  - Se debe generar un interés del 0.5% diario.
  - El monto del plazo fijo debe restarse del balance, ya que NO puede estar disponible para pagos o transferencias.

**Documentación**

- Se deberá documentar la API utilizando Swagger.

**Testing**

- Se deberán realizar los tests correspondientes a los diferentes endpoints utilizando JUnit y Mockito.

## 📌 Recursos y documentación

A continuación les compartimos los links de acceso al Backlog y al Repositorio Base (hacer un fork del repositorio) para que puedan comenzar a desarrollar.

Les sugerimos trabajar a través de la metodología ágil Scrum. Como bien saben, en Alkemy Academy disponen de un curso sobre **Metodologías de Trabajo** que cuenta con información clave sobre la metodología ágil Scrum.

Para ingresar, deben entrar a https://academy.alkemy.org/ con su usuario (correo electrónico) y contraseña (alkemy).

Asimismo, pueden crear un tablero gratuito con todas las tareas del proyecto en las siguientes páginas:

1. Clickup
2. Mural
3. Trello
4. Asana
5. Miro
6. iceScrum
7. taiga
8. Monday

Además, encontrarán cursos en la sección de Recursos con toda la información complementaria para poder avanzar con el proyecto:

- SKILL UP | Java
- Skill Up Metodologías de Trabajo
- Skill Up de Soft Skills

## 📤 Entregables

La Demo es una ceremonia clave en el mundo del desarrollo. Es una instancia de presentación de los avances del proyecto. Deberán realizar un **video Demo de 10 minutos** máximo presentando el proyecto.

- Scope: mostrar aplicación (funcionalidades, métodos y componentes principales), la dinámica del equipo y en los últimos minutos pueden comentar aprendizajes de esta experiencia. Participar con cámara encendida.
- Preparar slides para introducir la presentación. Recomendamos agregar: Integrantes del equipo, Mentor, Tecnología con la que trabajaron, Agenda de la presentación.

Asimismo, deberán entregar:

- Link al repositorio.
- Planilla de ceremonias Scrum. (hacer una copia de esta planilla, completarla y entregarla)

## ¡Éxitos! 🚀