**Test Exploratorio – Safe Wallet**

1. Objetivo del Test Exploratorio:

El objetivo principal es identificar posibles problemas, errores o áreas de mejora en las funciones integradas en el sprint 2.

1. Áreas/funciones a Testear:

* Resumen del dinero disponible.
* Últimos movimientos de la cuenta.
* Obtener todos los usuarios.
* Tarjetas
  + Registrar. (cod 201 OK, 409 asociada a otra cuenta, 400 bad request)
  + Borrar (cod 200 OK, 404 not found)
  + Ver tarjetas disponibles. ( cod 200 vacío o tarjetas, 500 en el caso de que no encuentre).

1. Organización del Test Exploratorio:

**Sesión 1: Cuenta**

* Ver el dinero en cuenta que tiene el usuario, en el caso de no tener deberíamos ver $0.
* Verificar los movimientos de la cuenta se registren.
* Obtener todas las cuentas de los usuarios.
* Obtener cuenta de un usuario especifico

**Sesión 3: Tarjetas**

* Asociar una tarjeta a la cuenta de un usuario.
* Borrar una tarjeta especifica.
* Ver tarjetas disponibles.

**Tours:**

Tour 1: Flujo Normal de cuenta y tarjeta

**Cuenta**

* Usuario nuevo cuenta en $0.
* Recibir el premio de bienvenida cuenta en $10.000

**Tarjeta**

* Crear una nueva tarjeta satisfactoriamente.
* Eliminar una tarjeta satisfactoriamente.
* Visualizar tarjetas disponibles satisfactoriamente.

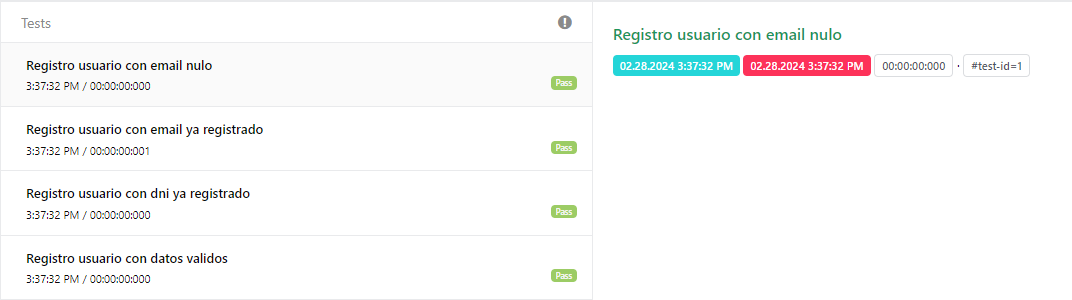
Tour 2: Escenarios de Error

* Explorar diferentes escenarios de error durante el proceso de visualización de cuenta y creación, eliminación y visualización de tarjetas.

**Escenarios posibles testeados: REGISTER**

Para el registro de usuarios en safewallet se utiliza un microservicio que se encarga de recibir los datos del usuario que se registrara. Este microservicio Guardara todos los datos del usuario y enviara solo los necesarios a dos microservicios utilizando feign

1. SafeWallet-keycloak: Quien tomara algunos datos del usuario para almacenarlos en Keycloak (Solo guarda el correo y el password).
2. SafeWallet-Transfers: Quien tomara algunos datos del usuario para crear la cuenta bancaria y almacenar solo esos datos (cbu, alias, correo, saldo, etc..).

****

1. **Registro usuario con datos validos**
   1. Microservicio: SafeWallet-UserDataFull.
   2. Clase: RegisterTest.
   3. Test: testCreateUser\_Success
   4. Resultado: Passed
2. **Registro usuario con DNI ya registrado.**
   1. Microservicio: SafeWallet-UserDataFull.
   2. Clase: RegisterTest.
   3. Test: testCreateUser\_alreadyRegister\_DNI
   4. Resultado: Passed
3. **Registro usuario con email ya registrado.**
   1. Microservicio: SafeWallet-UserDataFull.
   2. Clase: RegisterTest.
   3. Test: testCreateUser\_alreadyRegister\_EMAIL
   4. Resultado: Passed
4. **Registro usuario con email nulo.**
   1. Microservicio: SafeWallet-UserDataFull.
   2. Clase: RegisterTest.
   3. Test: testCreateUser\_alreadyRegister\_emailNull
   4. Resultado: Passed

Creación de alias y cbu aleatorio.

* 1. Microservicio: SafeWallet-UserDataFull.
  2. Test: generateCbu\_alias\_Success
  3. Resultado: Passed

Creacion de cbu y alias únicos

1. Microservicio: SafeWallet-UserDataFull.
2. Test: generateCbu\_alias\_Success\_CheckUnique
3. Resultado: Passed

Creacion de cbu con 22 digitos

1. Microservicio: SafeWallet-UserDataFull.
2. Test: generateCbu\_length
3. Resultado: Passed

* Se testea la creación de un alias extraído de un archivo txt con el formato xxx.xxx.xxx
* Se testea la creación de un cbu 22 digitos
* Se testea la creación de un cbu y alias único -> Se testearon 100 casos los cuales todos resultaron favorables.

**Escenarios posibles testeados: LOGIN**

En Safe Wallet se utiliza un adaptador keycloak desde el frontend para validar si hay un usuario en sesión, de no tener un usuario se envia al SSO de keycloak.

Una vez que el usuario este logeado se utiliza el token que se envía en un encabezado para la comunicación con los microservicios que conforman la billetera virtual. Cada microservicio es responsable de la seguridad de si mismo, traduciendo el token enviado por el Gateway para utilizarlo en el contexto de spring.

Se testearan los endpoints de Keycloack con REST ASSURE que nos permiten logearnos en nuestra aplicación.

1. **Login usuario con datos validos**
   1. Microservicio: SafeWallet-Keycloak.
   2. Test: login\_successful
   3. Resultado: Passed
2. **Login usuario con credenciales invalidas**
   1. Microservicio: SafeWallet-Keycloak.
   2. Test: login\_with\_wrong\_credentials
   3. Resultado: Passed
3. **Registro usuario con email no registrado.**
   1. Microservicio: SafeWallet-Keycloak.
   2. Test: login\_with\_invalid\_email
   3. Resultado: Passed

Consigna

Realizar un plan de pruebas en donde se mencione lo siguiente:

1. ¿Cómo escribir un caso de prueba?
   1. Se debe proporcionar el objetivo del caso de prueba, que funcionalidad o característica estamos por probar, luego deberíamos configurar el ambiente para realizar la prueba y seleccionar quien o quienes deberán comenzar el test. Luego deberíamos comenzar con el testeo de la funcionalidad y comparar el resultado que se espera obtener con el realmente obtenido.
2. ¿Cómo reportar un defecto?
   1. Para reportar un defecto debemos indicar en donde se encuentra el defecto y proporcionar toda la información para la reproducción del mismo. Ser lo mas claro posible.
3. Criterio para incluir un caso de prueba en una suite de humo.
   1. Depende del proyecto que tengamos, pero integraríamos las pruebas criticas que hacen al funcionamiento esencial del proyecto. En el caso de Safe Wallet, el login exitoso y seguro es esencial.
4. Criterio para incluir un caso de prueba en una suite de regresión.
   1. Integraríamos las pruebas de funcionalidades nuevas que afectan al software en su totalidad