**Desarrollo prueba leal**

Para el desarrollo de la prueba se creó una aplicación angular que consume los servicios dados. Se utilizó el framework Materialize el cual permite tener una interfaz amigable y responsive.

De acuerdo con lo anterior, el proyecto angular cuenta con 3 servicios diferentes:

**Login.service:** Este servicio se encarga de consumir el endpoint = <https://pruebatecnica.puntosleal.com/api/user/login> con el cual se obtiene un token que luego será guardado como una variable en el navegador. Este token es necesario para consumir los demás endpoints. Por otro lado, este servicio también se encarga de validar si hay o no un usuario logueado.

**Transacciones.service:** Este servicio se encarga de consumir 2 diferentes endpoints.

-https://pruebatecnica.puntosleal.com/api/user/my/transactions

-https://pruebatecnica.puntosleal.com/api/user/my/transactions?startDate=2018-01-01&endDate=2018-11-04

El primero obtiene todas las transacciones históricas del usuario. El segundo hace un filtro por fecha.

**Token-generator.service:** Este servicio implementa HttpInterceptor y se encarga de setear el header Authorization con el token obtenido al momento de hacer el login.

Así mismo, se crearon 3 componentes diferentes

**Login**: Componente encargado de mostrar el formulario de login y llamar el servicio “Login”.

**Transacciones:** Este componente tiene la responsabilidad de mostrar las transacciones que son dadas por el servicio que lleva su mismo nombre. También implementa un filtro que se ejecuta mediante un pipe (del cual se hablará después).

**NavBar:** Este componente es el encargado de tener el botón “Salir” el cual permite hacer logout.

Tanto Login, como transacciones cuentan con una ruta para la navegación entre estos 2 componentes.

Por otro lado, se crearon 2 guards.

**Auth.guard:** Valida si no está logueado el usuario, si es el caso, siempre redirecciona a la ruta de login.

**No-auth.guard:** Valida si está logueado el usuario, si es el caso, no se puede acceder a la ruta de login

También se crearon 2 modelos que permiten aprovechar más el tipado de typescript.

**Transacción:** Modela el objeto transacción que será visualizado en el componente transacciones.

**Usuario:** Modela el objeto de usuario que contiene la información necesaria para hacer Loguin.

Se creó también un pipe.

**Filter.pipe:** Este pipe permite filtrar las transacciones que tienen un valor menor a....

Por ultimo se implementaron pruebas unitarias a los servicios “Transacciones” y “Login”

En el primero se prueba que se consuma los endpoints correctamente.

En login se prueba que se consuma el servicio correctamente y que valide si un usuario está logueado o no.