**MANUAL TÉCNICO: FrontEnd**

**FECHA CREACION: 25/05/2020**

**AUTOR (ES) CREACION**

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE** | **GERENCIA** | **EMPRESA** | **FECHA(DD/MM/YYYY)** |
| **Juan Manuel Valderrama** | **Medios de Pago** | **Accenture** | **25/05/2020** |
|  |  |  |  |
|  |  |  |  |

**REVISOR (ES) DE DOCUMENTO**

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE** | **GERENCIA** | **EMPRESA** | **FECHA(DD/MM/YYYY)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**APROBADOR (ES) DE DOCUMENTO**

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE** | **GERENCIA** | **EMPRESA** | **FECHA(DD/MM/YYYY)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**HISTORIA DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NOMBRE** | **VERSIÓN** | **DESCRIPCION DEL CAMBIO** | **PMO** | **FECHA(DD/MM/YYYY)** |
| **Documento Inicial** | **1.0** | **Creación de documento** | **31568** | **25/05/2020** |
| **Cambio estructura de carpetas** | **2.0** | **Se realiza actualización a la estructura de carpetas y nuevas funcionalidades** | **31568** | **20/09/2020** |
|  |  |  |  |  |

**TABLA DE CONTENIDO**

[1. Descripción del módulo 6](#_Toc44064021)

[2. Generalidades técnicas del proyecto 6](#_Toc44064022)

[3. Herramientas necesarias para ejecutar el proyecto 7](#_Toc44064023)

[4. Estructura general del proyecto, archivos de configuración y librerías 7](#_Toc44064024)

[5. Detalle por componente 16](#_Toc44064025)

[6. Consumo de servicios 25](#_Toc44064026)

[7. Implementación pruebas unitarias 27](#_Toc44064027)

**IMÁGENES**

Imagen 1. Estructura del proyecto frontend. 8

Imagen 2. Estructura archivo Package.json 9

Imagen 3. Estructura archivo Angular.json 11

Imagen 4. Estructura carpeta assets 13

Imagen 5. Estructura archivo environment.ts. 14

Imagen 6. Estructura archivo environment.dev.ts. 14

Imagen 7. Estructura archivo environment.cert.ts. 15

Imagen 8. Estructura archivo environment.prod.ts. 15

Imagen 9. Estructura carpeta config. 16

Imagen 10. Estructura carpeta guards. 16

Imagen 11. Estructura carpeta interceptors. 17

Imagen 12. Estructura carpeta providers. 17

Imagen 13. Estructura carpeta services. 18

Imagen 14. Estructura carpeta layout. 18

Imagen 15. Estructura carpeta modules. 19

Imagen 16. Estructura carpeta boton-bancolombia. 20

Imagen 17. Estructura carpeta code-qr. 20

Imagen 18. Estructura carpeta home. 21

Imagen 19. Estructura carpeta query-logs. 21

Imagen 20. Estructura carpeta query-trx. 21

Imagen 21. Estructura carpeta shared. 22

Imagen 22. Estructura carpeta animatios. 22

Imagen 23. Estructura carpeta components. 23

Imagen 24. Estructura carpeta constants. 23

Imagen 25. Estructura carpeta directives. 24

Imagen 26. Estructura carpeta model. 25

Imagen 27. Estructura carpeta utils. 26

Imagen 28. Prueba unitaria login component. 32

Imagen 29. Configuración karma.config.js. 33

Imagen 30. Resultado pruebas unitarias. 34

**TABLAS**

Tabla 1. Login.service.ts 26

Tabla 2. Transactions.service.ts 27

Tabla 4. Logs.service.ts 27

Tabla 5. Logs.service.ts, Insertar log 29

Tabla 6. Code-qr.service.ts, consulta qr 30

Tabla 7. Code-qr.service.ts, Editar delegado. 30

# Descripción del módulo

En este artículo se definirán las especificaciones técnicas requeridas y necesarias para el funcionamiento de la aplicación cliente frontend para los usuarios internos del Banco. Al igual se detalla los componentes realizados en este aplicativo y cada una de sus funciones.

# Generalidades técnicas del proyecto

La tecnología implementada para el desarrollo del frontend es Angular, un framework que facilita la creación y programación de aplicaciones web de una sola página, asegurando los desarrollos con rapidez y a la vez que posibilita modificaciones y actualizaciones.

Este framework es modular y escalable adaptándose a las necesidades, basado en el estándar de componentes web con un conjunto de interfaz de programación de aplicaciones (API).

El desarrollo del proyecto se realizó bajo la versión 8 del framework, lineamiento especificado por el Banco.

Adicionalmente a esto, entre las principales librerías implementadas en el proyecto se tienen las siguientes con sus respectivas versiones:

*"@angular/animations"*: "~8.2.14",

*"@angular/cdk"*: "~8.2.3",

*"@angular/common"*: "~8.2.14",

*"@angular/compiler"*: "~8.2.14",

*"@angular/core"*: "~8.2.14",

*"@angular/forms"*: "~8.2.14",

*"@angular/material"*: "~8.2.3",

*"@angular/platform-browser"*: "~8.2.14",

*"@angular/platform-browser-dynamic"*: "~8.2.14",

*"@angular/router"*: "~8.2.14",

*"adal-angular4"*: "^4.0.12",

*"hammerjs"*: "^2.0.8",

*"material-design-icons"*: "^3.0.1",

*"ngx-cookie-service"*: "^10.0.1",

*"rxjs"*: "~6.4.0",

*"rxjs-compat"*: "^6.5.4",

*"saturn-datepicker"*: "^8.0.5",

*"tslib"*: "^1.10.0",

*"zone.js"*: "~0.9.1"

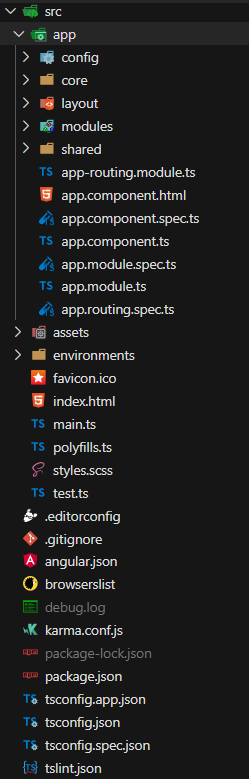
# Herramientas necesarias para ejecutar el proyecto

Las herramientas necesarias para la ejecución y correcto funcionamiento de la aplicación de Frontend son las siguientes:

* **NodeJs**: es un entorno de tiempo de ejecución de JavaScript, está diseñado para crear aplicaciones de red escalables y de alta mantenibilidad. La versión implementada para este proyecto es 12.13.1.
* **Npm**: es el administrador de paquetes para la plataforma Node para el lenguaje JavaScript, su función es gestionar las dependencias del proyecto de forma inteligente. La versión que se implemento es la 6.4.1.
* **Angular CLI**: es la herramienta de línea de comandos estándar para crear, depurar y publicar aplicaciones Angular. La versión implementada para el proyecto es 8.3.20.
* **Angular Material**: es una librería de estilos basada en la guía de diseño de Material Design, realizado por el equipo de Angular para integrarse perfectamente con Angular; permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web de acuerdo con las necesidades del cliente.

# Estructura general del proyecto, archivos de configuración y librerías

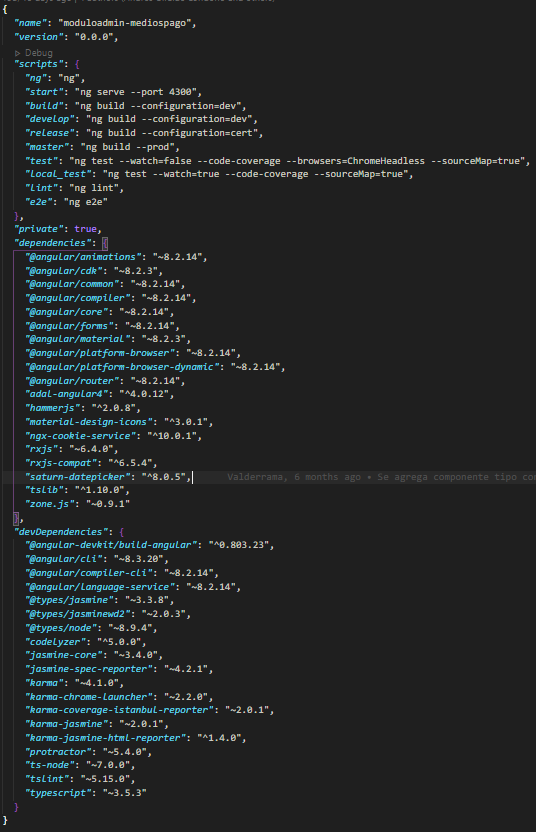
A continuación, se detalla la estructura del proyecto y su composición para la correcta ejecución, detallada por módulos y componentes generales desarrollados dentro del módulo administrativo. En la siguiente imagen se visualiza la estructura que conforma la aplicación de frontend para el módulo administrativo:



*Imagen 1. Estructura del proyecto frontend.*

Entre los archivos descriptos en la anterior imagen se puede destacar los archivos de configuración como lo son:

* **Package.json**: En este archivo podremos encontrar las librerías implementadas para el proyecto con sus respectivas versiones.



*Imagen 2. Estructura archivo Package.json*

En la anterior imagen se puede observar el detalle de la configuración de algunos elementos importantes como:

* + **Scripts**: son la serie de comandos que ejecutaremos para iniciar y ejecutar el proyecto (*npm start*). *build, develop, release y máster* sirven para compilar y construir nuestro proyecto con sus respectivas configuraciones dependiendo del ambiente en el que se desee desplegar, esto va en relación del archivo *angular.json* donde se detalla la construcción de la aplicación de acuerdo a estos parámetros.
  + **Dependencias**: Aquí se encuentran las librerías y componentes necesarios para construir la aplicación, en este espacio se puede verificar la versión de *Angular* que se está implementando y otros complementos como *Adal* que se utiliza para realizar el inicio de sesión con directorio activo de *Windows*.
  + **Angular.json**: La función de este archivo es servir de guía para *Angular CLI*, declarando una serie de atributos que permiten a esta herramienta realizar operaciones sobre la aplicación.



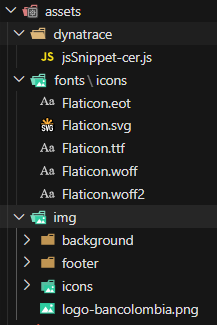
*Imagen 3. Estructura archivo Angular.json*

En la anterior imagen se puede observar el detalle de la configuración de algunos elementos importantes como:

* **$schema:** permite a *Angular* *CLI* respetar y asegurar la especificación de Angular *Workspace*.
* **Versión:** número entero que declara cual es la versión actual de la especificación de Angular *Workspace*.
* **NewProjectRoot**: corresponde al *path* donde las nuevas aplicaciones o librerías serán creadas.
* **Projects:** propiedad que declara la configuración para cualquier proyecto en el *Workspace*.

Cada proyecto es configurado con los siguientes atributos:

* **Root**: propiedad que define cual es el directorio principal con todos los archivos de los proyectos.
* **Sourceroot**: Propiedad que especifica el directorio donde se encontrará el código de fuente escrito.
* **ProjectType**: Propiedad que especifica el tipo de proyecto.
* **Prefix**: Propiedad que declara el prefijo utilizado al momento de crear nuestros componentes y directivas.
* **Schematics**: Propiedad que permite definir la configuración del módulo de a nivel de proyecto.
* **Architect**: Cada proyecto permite definir una serie de acciones como el *building*, *serve*, *testing*, *linting*, entre otros. Estas tareas están basadas en *builders* predeterminados definidos dentro del módulo de *Angular CLI*.
* **Build**: Aquí se detallan cada una de las configuraciones necesarias para construir la aplicación como lo son desarrollo, certificación y producción, que se mencionaron el archivo package.json.
* **Assets:** En esta carpeta se especifican todos los archivos y/o recursos que son usados en el módulo administrativo, como lo son imágenes, iconos y fuentes.

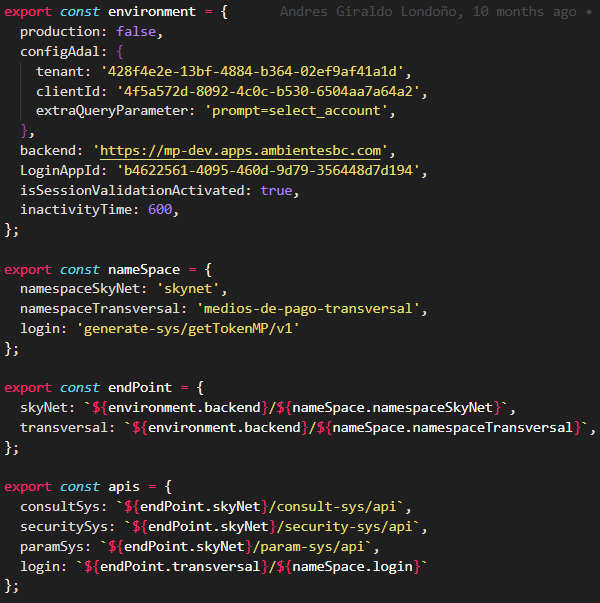


*Imagen 4. Estructura carpeta assets*

* **Enviroment.ts:** En este archivo se especifican características propias para la ejecución de la aplicación para cada uno de los diferentes entornos y ambientes; como lo son desarrollo, certificación y producción. Estas características son utilizadas por el archivo Angular.json para la construcción del proyecto.



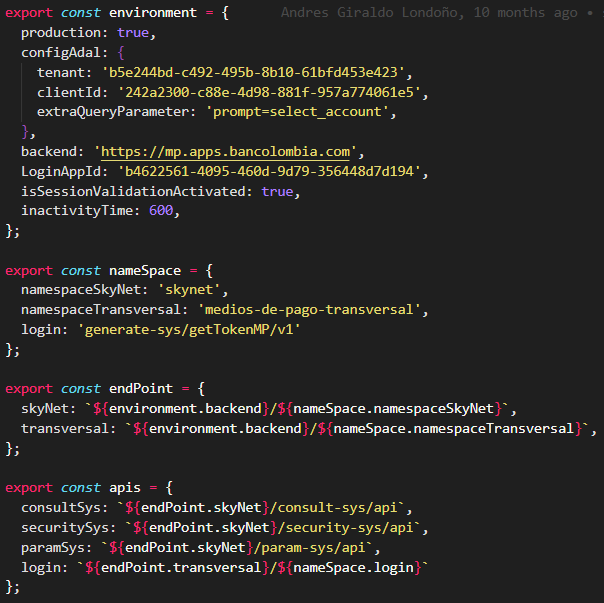
*Imagen 5. Estructura archivo environment.ts.*



*Imagen 6. Estructura archivo environment.dev.ts.*



*Imagen 7. Estructura archivo environment.cert.ts.*

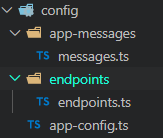


*Imagen 8. Estructura archivo environment.prod.ts.*

# Detalle por componente

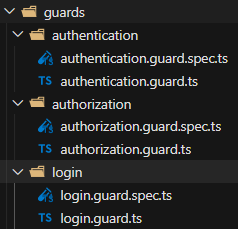
En el anterior punto se detalló la estructura general del proyecto y los archivos de configuración, en este apartado se definirán cada una de las carpetas y sus componentes.

* **Config:** en esta carpeta se encuentran todas las configuraciones globales de la aplicación como lo son mensajes de bienvenida, mensajes de error y *endpoints* para el consumo de servicios.



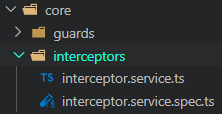
*Imagen 9. Estructura carpeta config.*

* **Core:** en esta carpeta se encuentran las subcarpetas *Guards, Interceptors, Providers y Services.*
  + **Guards:** aquí encuentra todo lo relacionado con validaciones de seguridad, es decir la protección de las rutas de la aplicación para que dependiendo del rol le sea permitido o no la visualización de la información.



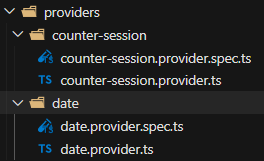
*Imagen 10. Estructura carpeta guards.*

* + **Interceptors:** aquí se encuentra el servicio que proporciona a todos los *endpoints* el token de la aplicación.



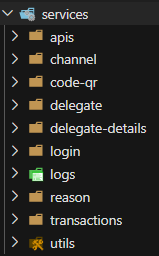
*Imagen 11. Estructura carpeta interceptors.*

* + **Providers:** en esta carpeta se encuentran los servicios que proporcionan la visualización de la fecha y el *counter sesión*.



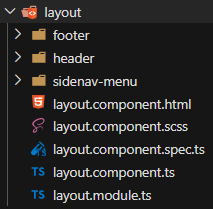
*Imagen 12. Estructura carpeta providers.*

* + **Services:** en estacarpeta se encuentran los servicios que proporcionan la información de *Apis*, canales, códigos QR, delegados, detalle delegados, botón Bancolombia, acciones de usuario, consulta transaccional transversal y el ingreso a la aplicación.



*Imagen 13. Estructura carpeta services.*

* **Layout:** En esta carpeta se encuentran las pantallas donde se construye el footer, header y el menú de la aplicación, además del componente principal Layout.

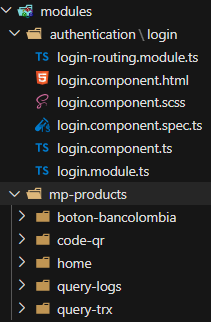


*Imagen 14. Estructura carpeta Layout.*

* **Modules:** en esta carpeta se encuentran las pantallas con las cuales el usuario realiza las operaciones del módulo administrativo. Ingreso al módulo administrativo, consulta transaccional transversal y consulta acciones de usuario. Esta carpeta se conforma de las siguientes subcarpetas: *mp-products, bt, qr, shared*.
  + **Authentication:** esta carpeta contiene el componente login con sus respectivos archivos de configuración: *.html, .scss, .spec* y *module* para por de ejecutar correctamente el inicio de sesión.
  + **mp-products:** esta carpeta contiene las carpetas *botón-bancolombia, code-qr, home*, query-logs y query-trx.

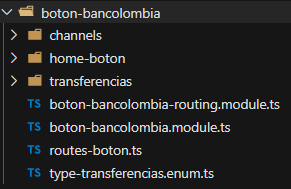
Todos los componentes tienen definidos los tipos de archivos que se describen a continuación:

* + - **html:** los archivos con esta extensión detallan la estructura y maquetación del componente.
    - **scss:** aquí se construye los estilos del componente, los estilos aquí descritos solo aplican para el componente.
    - **ts:** es elcontrolador del componente**.**
    - **spec:** archivo donde se realizan las pruebas unitarias del componente.
    - **module.ts:** aquí se realiza la declaración del componente y la importación de los módulos necesarios para el correcto funcionamiento del componente.
    - **routing.ts:** en este archivo se definen las rutas del componente.



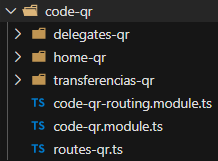
*Imagen 15. Estructura carpeta modules.*

* + **botón-bancolombia:** hace referencia a los componentes y funcionalidades que pertenecen a botón Bancolombia yaquí se encuentran los canales de vinculación, el home de botón y el componente transferencias*;* con sus respectivos archivos *module.ts* y *routing.ts.*



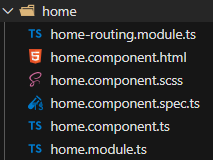
*Imagen 16. Estructura carpeta botón-bancolombia.*

* + **code-qr:** hace referencia a los componentes y funcionalidades de código QR y aquí se encuentran las funcionalidades de delegados, el *home* y las *consultas de transferencias;* con sus respectivos archivos *module.ts* y *routing.ts.*



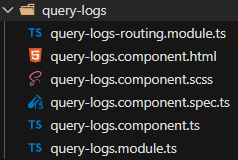
*Imagen 17. Estructura carpeta code-qr.*

* + **home:** hace referencia al home principal de la aplicación, donde se visualizan todas las opciones y funcionalidades del sistema. *;* con sus respectivos archivos *module.ts* y *routing.ts.*



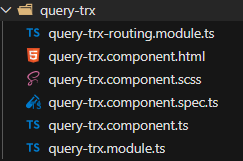
*Imagen 18. Estructura carpeta home.*

* + **query-logs:** hace referencia al componente de consulta de acciones de usuario, componente que contiene la funcionalidad para visualizar la trazabilidad que realiza un usuario dentro del sistema.



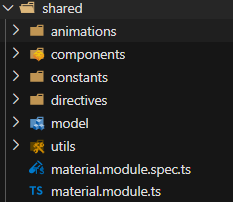
*Imagen 19. Estructura carpeta query-logs*

* + **query-trx:** hace referencia al componente de consulta transaccional transversal, componente que contiene la funcionalidad para visualizar las transacciones realizadas por los diferentes productos de medios de pago. (Botón Bancolombia y Código QR).



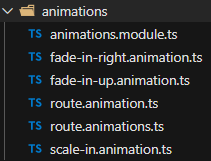
*Imagen 20. Estructura carpeta query-trx*

* + **shared:** en esta carpeta se encuentran los recursos que son compartidos en todo el módulo administrativo. Esta carpeta se conforma de las siguientes subcarpetas: *components*, *constants*, *data*, *directives*, *model* y *utils.* Además, dentro de este directorio se encuentran los módulos de *angular material* y *shared.*



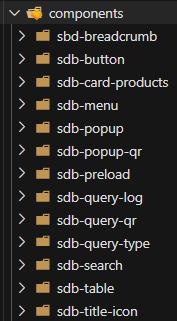
*Imagen 21. Estructura carpeta shared.*

* + **animations:** aquí se encuentran las animaciones que se realizan sobre los elementos html de la aplicación.



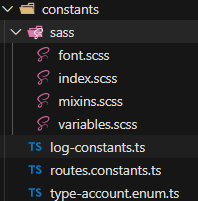
*Imagen 22. Estructura carpeta animations.*

* + **components:** aquí se encuentran todos los componentes que pueden ser usados en cualquier parte de proyecto haciendo referencia al selector del *component.ts*.



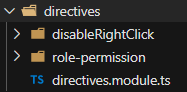
*Imagen 23. Estructura carpeta components.*

* + **constants:** esta carpeta está compuesta por todas las configuraciones *scss y constantes* que utiliza el módulo administrativo.



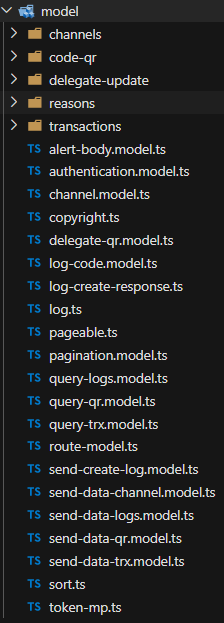
*Imagen 24. Estructura carpeta constants.*

* + **directives:** aquí se encuentran las funciones que son invocadas cuando el DOM (*Document Object Model*) es compilado por el framework de Angular. La finalidad de una directiva es modificar o crear un comportamiento totalmente nuevo. Ejemplo: deshabilitar el clic derecho u obtener los roles del usuario logueado.



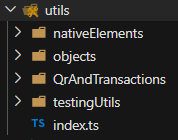
*Imagen 25. Estructura carpeta directives.*

* + **model:** en esta carpeta se encuentra la estructura de los datos que son manejados dentro del módulo administrativo, se usa con el fin de proteger la data de asignaciones indebidas.



*Imagen 26. Estructura carpeta model.*

* + **Utils:** en este directorio se encuentran las funciones útiles que pueden ser usadas en cualquier parte del módulo administrativo.



*Imagen 27. Estructura carpeta utils.*

# Consumo de servicios

A continuación, se describen todos los servicios utilizados en el módulo administrativo para la visualización de la información y el correcto funcionamiento de este.

|  |  |  |
| --- | --- | --- |
| **Nombre** | Login | |
| **Ruta de ubicación** | \src\app\core\services\login | |
| **Tipo de petición** | POST | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| Login.service.ts | Servicio que realiza petición al backend para realizar el proceso de login. | **Desarrollo**  https://mp-dev.apps.ambientesbc.com/medios-de-pago-transversal/generate-sys/getTokenMP/v1  **Certificación**  https://mp-qa.apps.ambientesbc.com/medios-de-pago-transversal/generate-sys/getTokenMP/v1  **Producción**  https://mp-apps.apps.ambientesbc.com/medios-de-pago-transversal/generate-sys/getTokenMP/v1 |

***Tabla 1****. Login.service.ts.*

|  |  |  |
| --- | --- | --- |
| **Nombre** | Consulta transaccional | |
| **Ruta de ubicación** | \src\app\core\services\transactions | |
| **Tipo de petición** | GET | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| Transactions.service.ts | Servicio que realiza petición al back para realizar consulta de información de transacciones. | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/ skynet/consult-sys/api/  transactions?paginationSize=10&  paginationKey=0  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/ skynet/consult-sys/api/  transactions?paginationSize=10&  paginationKey=0  **Producción:**  https://mp.apps.bancolombia.com bancolombia.com/ skynet/consult-sys/api/  transactions?paginationSize=10&  paginationKey=0 |

***Tabla 2****. Transactions.service.ts.*

|  |  |  |
| --- | --- | --- |
| **Nombre** | Consulta acciones de usuario | |
| **Ruta de ubicación** | \src\app\core\services\logs | |
| **Tipo de petición** | GET | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| Logs.service.ts | Servicio que realiza petición al back para realizar consulta de información de acciones de usuario. | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/security-sys/api/logs-list?size=10  &page=0&createStar=2020-06-27%2000:00:00%20AM&  createEnd=2020-09-5%2011:59:59%  20PM&frstConsult=true  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/skynet/security-sys/api/logs-list?size=10  &page=0&createStar=2020-06-27%2000:00:00%20AM&  createEnd=2020-09-5%2011:59:59%  20PM&frstConsult=true  **Producción:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/security-sys/api/logs-list?size=10  &page=0&createStar=2020-06-27%2000:00:00%20AM&  createEnd=2020-09-5%2011:59:59%  20PM&frstConsult=true |
|  | Servicio que realiza petición al back para realizar consulta de códigos de transacción de acciones de usuario. | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/security-sys/api/  code/all-codes  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/ skynet/security-sys/api/  code/all-codes  **Producción:**  https://mp.apps.bancolombia.com bancolombia.com/skynet/security-sys/api/  code/all-codes |

***Tabla 3****. Logs.service.ts.*

|  |  |  |
| --- | --- | --- |
| **Nombre** | Registro acciones de usuario | |
| **Ruta de ubicación** | \src\app\core\services\logs | |
| **Tipo de petición** | POST | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| Logs.service.ts | Servicio que realiza petición al back para insertar el registro de acción de usuario al iniciar o cerrar sesión. | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/security-sys/api/create-log  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/skynet/security-sys/api/create-log  **Producción:**  https://mp.apps.bancolombia.com bancolombia.com/ security-sys/api/create-log |

***Tabla 4****. Logs.service.ts, Insertar log*

|  |  |  |
| --- | --- | --- |
| **Nombre** | Consulta códigos QR | |
| **Ruta de ubicación** | \src\app\core\services\code-qr | |
| **Tipo de petición** | GET | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| code-qr.service.ts | Servicio que realiza petición al back consultar los códigos QR. | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/consult-sys/api/qr/list?paginationSize=10&  paginationKey=0&firstConsult=true  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/skynet/consult-sys/api/qr/list?paginationSize=10&  paginationKey=0&firstConsult=true  **Producción:**  https://mp.apps.bancolombia.com bancolombia.com/skynet/consult-sys/api/qr/list?paginationSize=10&  paginationKey=0&firstConsult=true |

***Tabla 5****.* *Code-qr.service.ts, consulta qr*

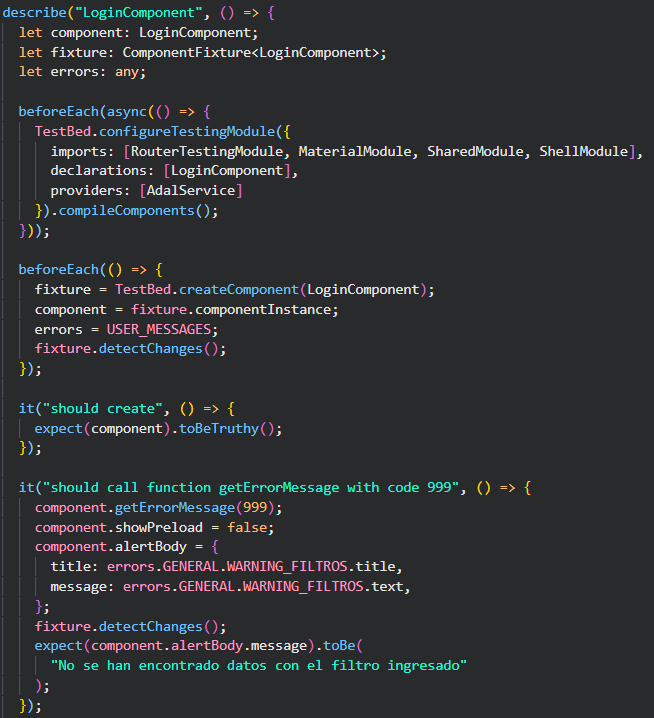
|  |  |  |
| --- | --- | --- |
| **Nombre** | Editar delegado | |
| **Ruta de ubicación** | \src\app\core\services\code-qr | |
| **Tipo de petición** | POST | |
| **Nombre del servicio** | **Descripción del servicio** | **Endpoint** |
| code-qr.service.ts | Servicio que realiza petición al back para editar un delegado | **Desarrollo:**  https://mp.dev.bancolombia.com bancolombia.com/skynet/param-sys/api/qr/update-delegate  **Certificación:**  https://mp.qa.bancolombia.com bancolombia.com/skynet/consult-sys/api/qr/update-delegate  **Producción:**  https://mp.apps.bancolombia.com bancolombia.com/skynet/consult-sys/api/qr/update-delegate |

***Tabla 6****.* *Code-qr.service.ts, Editar delegado.*

# Implementación pruebas unitarias

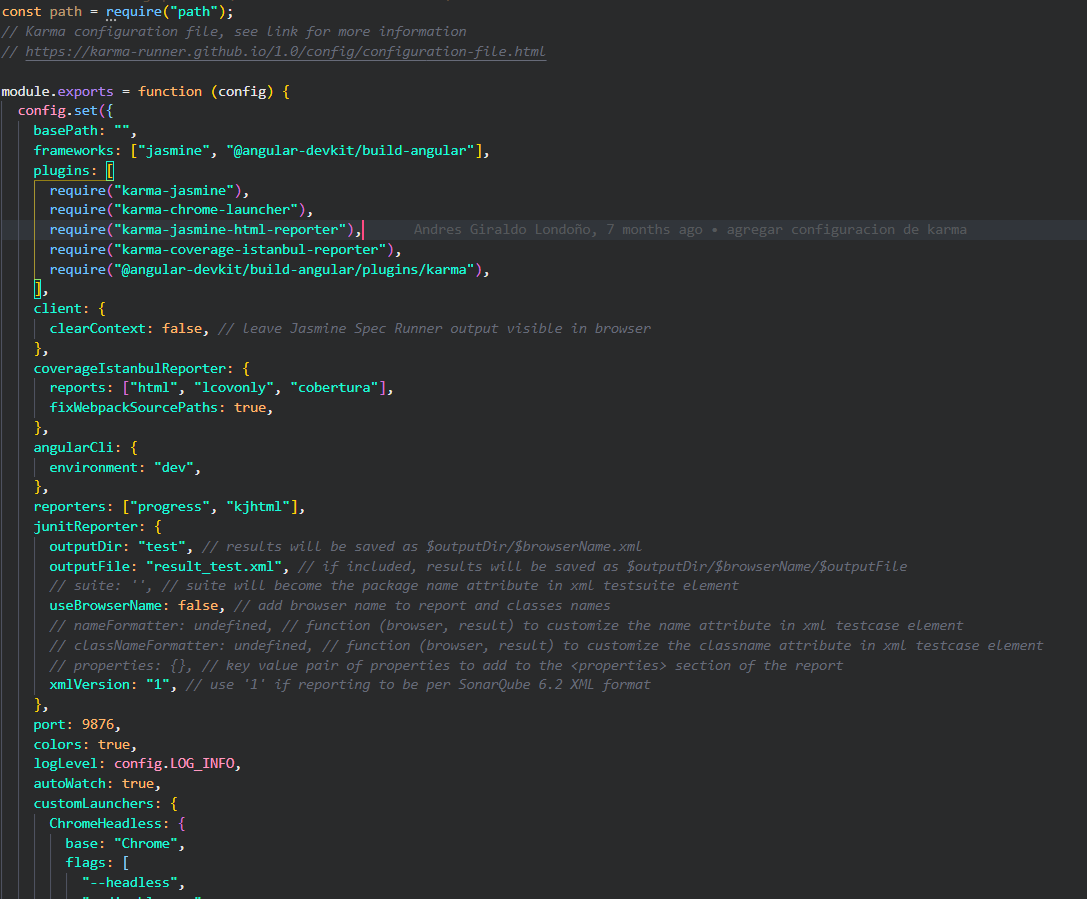
Para las pruebas unitarias se hace uso de *JasmineJs*; lenguaje con el cual se realizan las pruebas de los componentes y servicios desarrollados para el módulo administrativo. Además de *JasmineJs* se utiliza *Karma* que es una herramienta encargada de ejecutar las pruebas unitarias en el navegador y simular las peticiones desde la aplicación hacia los distintos servicios expuestos en nube.

En la siguiente imagen se visualiza un ejemplo de una prueba unitaria, aplicada al componente *login.component*.



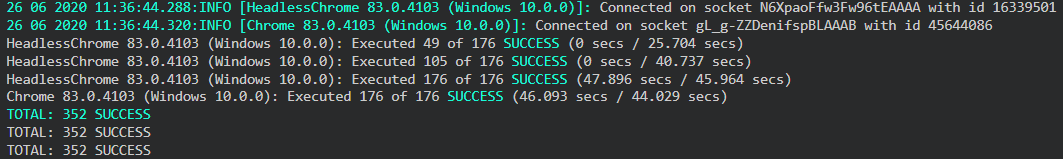
*Imagen 28. Prueba unitaria login component.*

Para la configuración del entorno de pruebas se debe tener en cuenta el archivo *karma.conf.js,* donde se puede definir variables como el navegador, el número de ejecutores, entre otros. En la siguiente imagen se puede visualizar gran parte de configuración del archivo *karma.config.js* del módulo administrativo.



*Imagen 29. Configuración karma.config.js.*

Para la ejecución de las pruebas unitarias dentro del módulo administrativo se recomienda utilizar el siguiente comando: ***npm run local\_test*:** este servirá para probar cada uno de los métodos y servicios implementados que tengan una prueba unitaria y además indicará la cobertura y funcionamiento del código. Al ejecutar este comando se mostrará el resumen de cuantas pruebas se ejecutaron y cuantas se llevaron a cabo satisfactoriamente, como se indica la siguiente imagen.

 *Imagen 30. Resultado pruebas unitarias.*