**SISTEMA TRANSMILENIO**

**MARIA CAMILA BUITRAGO**

**OMAR GUILLERMO SISA**

**DE:**

**INTEGRACIÓN DE APLICACIONES CON FRAMEWORKS DE PERSISTENCIA**

**PRESENTADO A:**

**ING. Jairo Alberto Botero Fagua**

**UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA**

**ESPECIALIZACIÓN EN BASES DE DATOS**

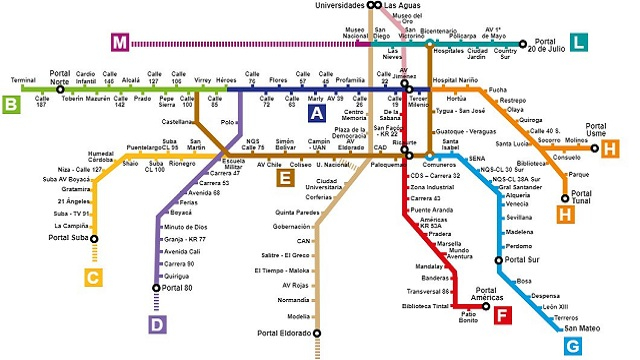
**TUNJA**

**2019**

**SISTEMA TRANSMILENIO**

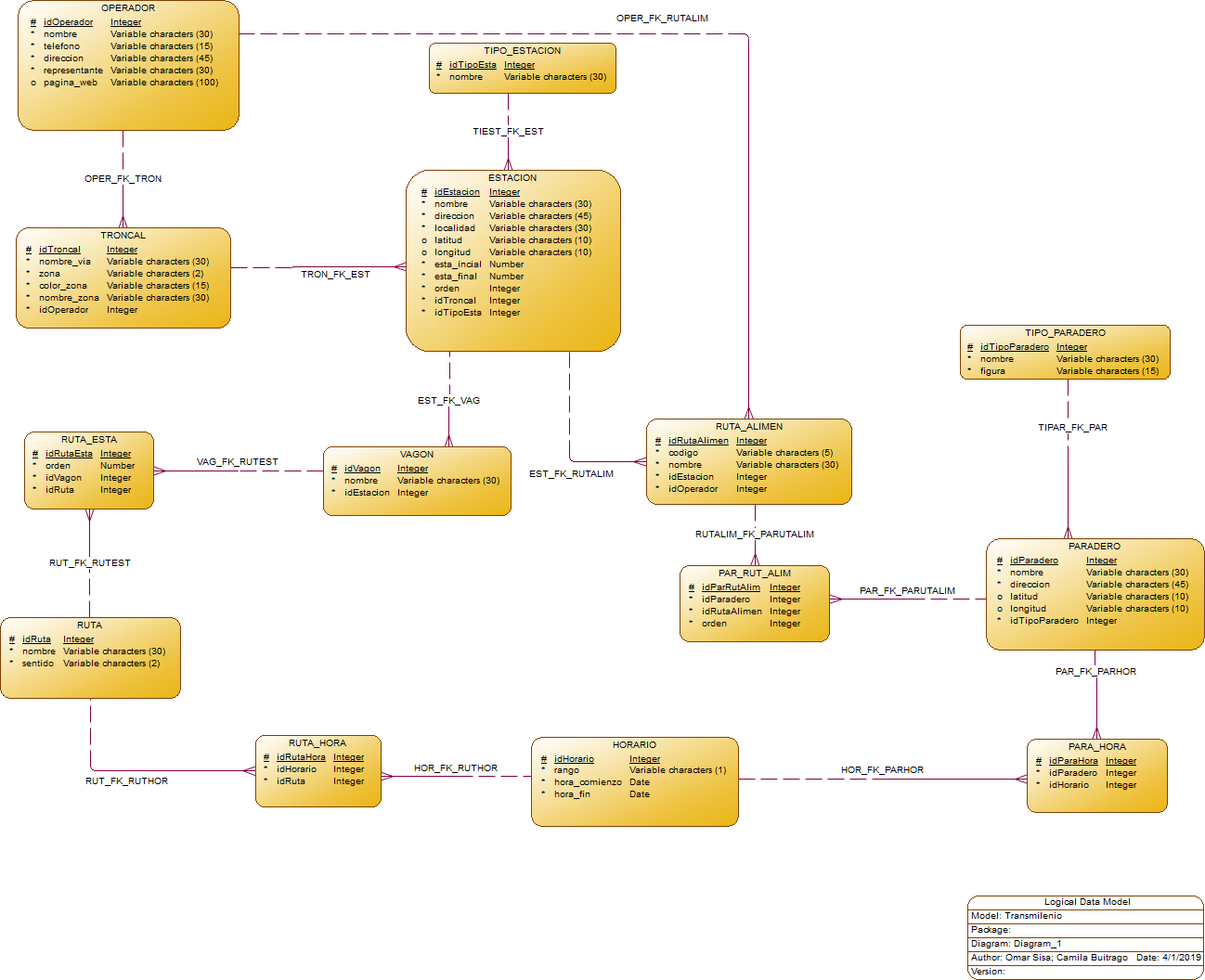
A continuación, se describe el proceso detallado para la construcción del sistema web sobre el Transmilenio.

De forma inicial se consultaron las fuentes oficiales del Transmilenio para realizar el respectivo análisis sobre el modelo entidad relación:



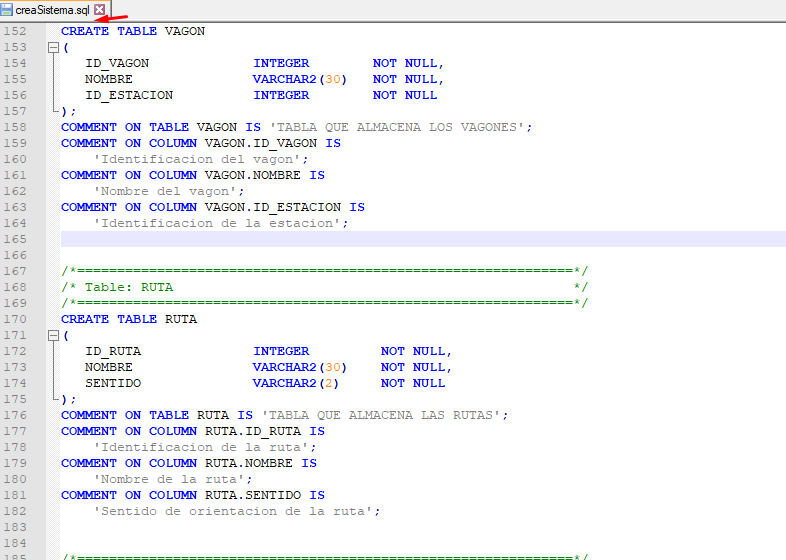
Figura, diagrama del sistema de Transmilenio.

En la figura anterior se puede evidenciar con mayor claridad como funciona el sistema de Transmilenio en el cual se encuentran varios componentes que interactúan entre si como: Portales, Rutas, estaciones, tipos de estación, Operadores, entre otros.

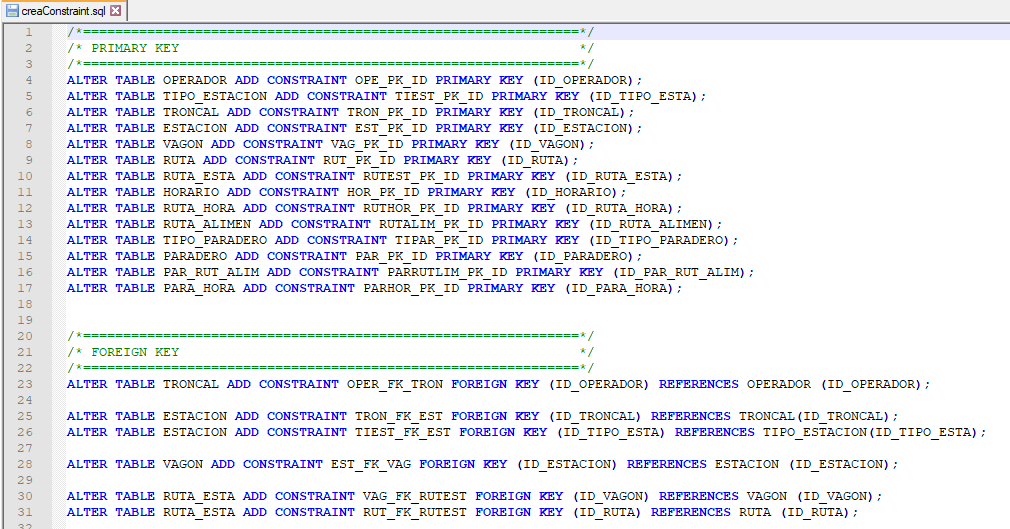


Figura, modelo entidad relación de la base de datos.

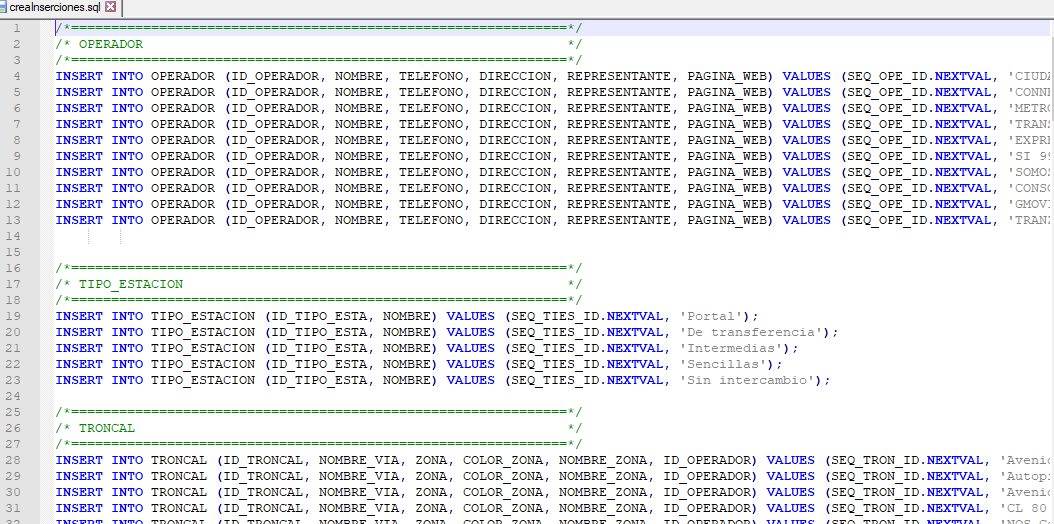
Luego de analizar y realizar el modelo entidad relación es necesario crear los scripts de la base de datos los cuales se encuentran dentro de la carpeta docs/scripts.



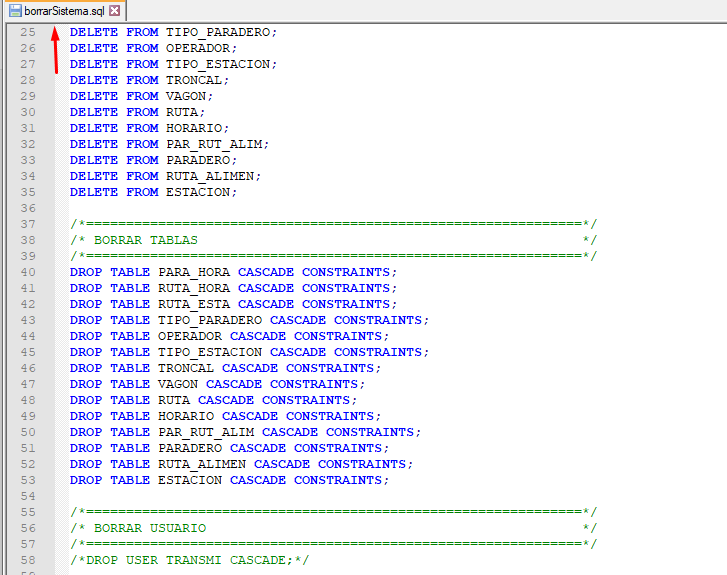
Figura, archivo creaSistema.sql el cual es necesario para crear todas las tablas del modelo.



Figura, archivo creaConstraint contiene los foreign key los primary key los check para limitar las entradas al sistema con la lógica de negocio.



Figura, archivo creaInserciones.sql para poblar la base de datos.



Figura, archivo borrarSistema.sql en el cual inicialmente se borran los check luego la información que hay en cada una de las tablas, y luego borra las tablas.

Para cada uno de los archivos anteriores fue necesario analizar el orden de ejecución de las instrucciones para respetar la integridad referencial.

Posteriormente fue necesario seleccionar las herramientas necesarias para construcción del proyecto:

**Lenguajes:** Java –TypeScript

**Frameworks:** Hiberante - Angular

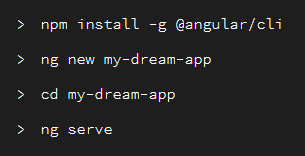
**Base de datos:** Oracle 11

**Editor:** Visual Studio Code

**Control de versiones:** Git –GitHub <https://github.com/camila95/transmilenio-ebd>

En el último enlace de git se puede ver el código fuente del sistema.

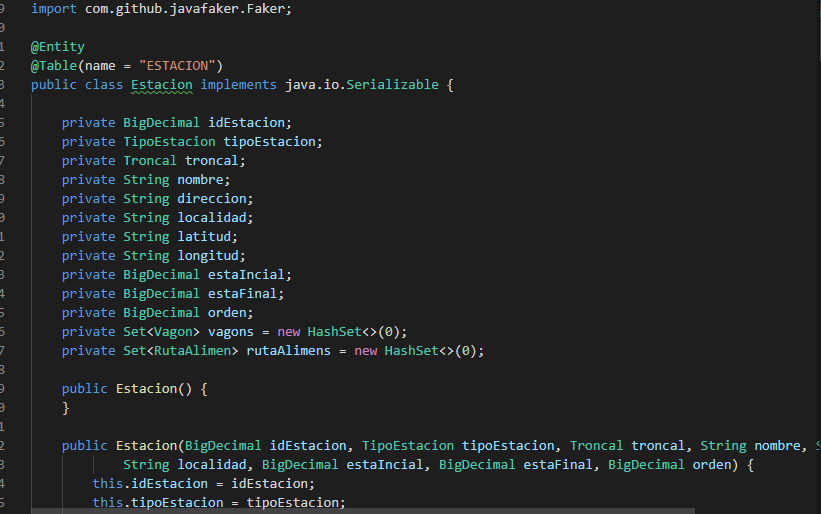
Para manejo de angular es necesario instalar una herramienta llamada angularCli la cual nos facilita el proceso de crear el proyecto



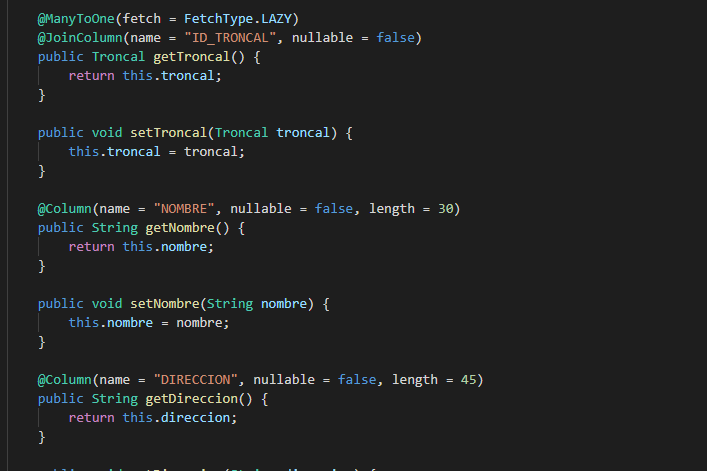
Figura, instalar angular cli.

En el primer comando funciona para instalar angular cli, el segundo es para crear la estructura de carpetas de angular en donde “my-dream-app” es el nombre del proyect, luego ingresa a la carpeta que se creó y ejecuta un servidor en el puerto 4200 para ver los cambios del sistema. Para instalar los anteriores comandos es necesario tener instalado nodejs desde la página oficial.

Luego es necesario crear el backend y el frontend como se muestra a continuación:

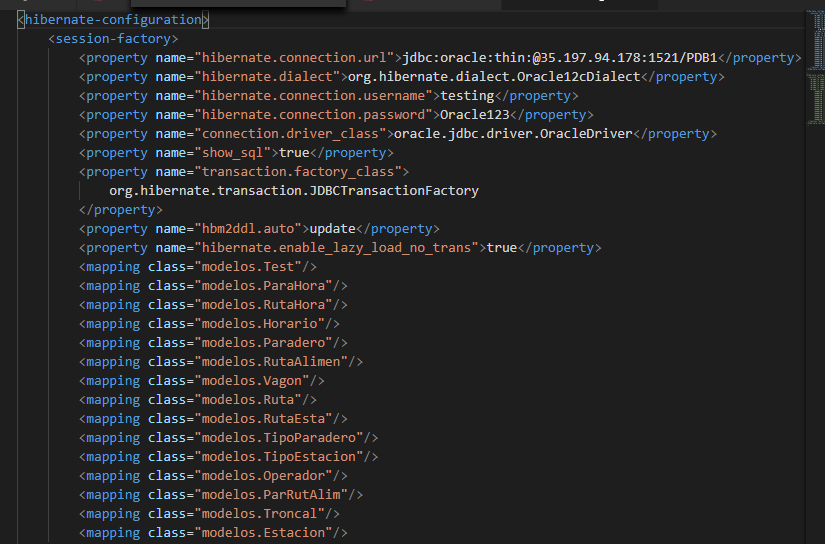


Figura, pojo de estación en hibernate utilizando JPA



Figura, ejemplo de relaciones manyToOne utilizando jpa.

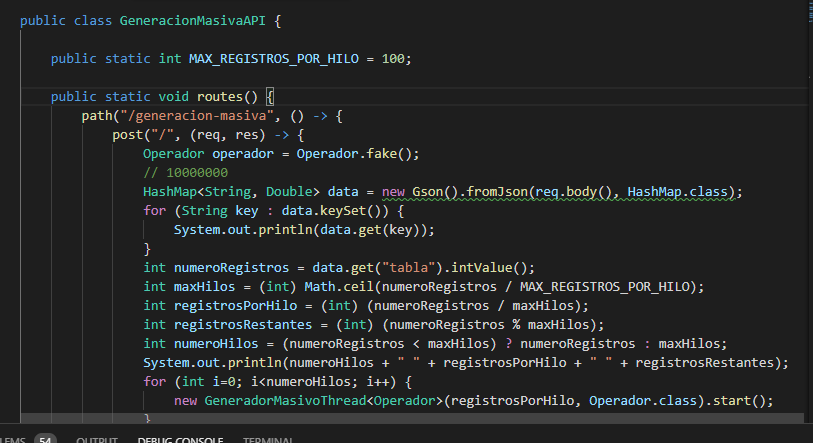
El proceso anterior de crear las anteriores clases, es necesario crear una clase por cada tabla del modelo entidad relación con sus respectivos atributos, constructores, getters y setters.



Figura, archivo de configuración de hibernate.

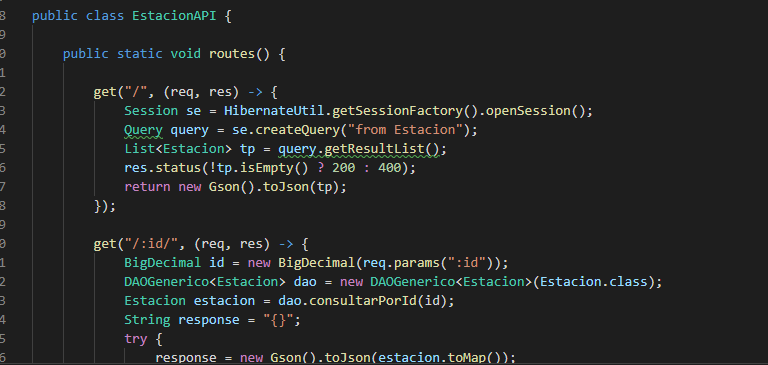


Figura, testApi.java para ejecutar el main e iniciar en el puerto 8444

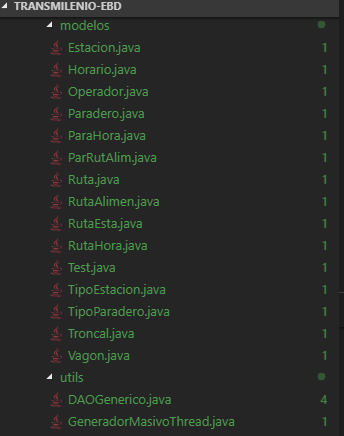


Figura, api de generación masiva.

Para generar los datos utilizando el método fake para generar datos.



Figura, estación API para traer los datos de la tabla estaciones.



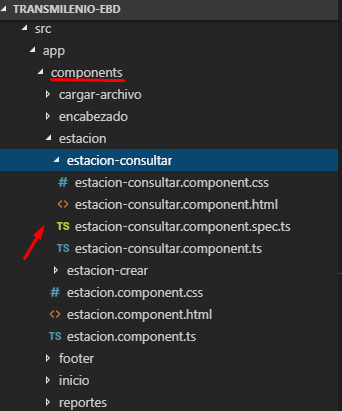
Figura, estructura de carpetas

En la figura anterior se muestra la evidencia para crear los POJOS de cada una de las tablas de la base de datos, esto se realiza para tener comunicación con la base de datos. Los anteriores archivos se pueden encontrar dentro de la carpeta Backend.

A continuación, es construir la parte del frontend o la parte visual, para eso es necesario crear un componente por cada parte de la pantalla que cambia, por ejemplo: un componente para el encabezado otro para el pie de pagina, para crear troncales, consultar etc. Para crear cada uno de los siguientes componentes es necesario desde un cdm ejecutar el siguiente comando Utilizando angularCLI



Figura, comando para crear componentes.

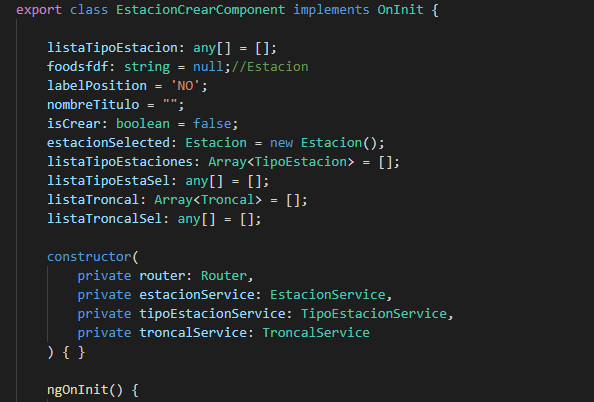


Figura, estructura de carpeta de angular

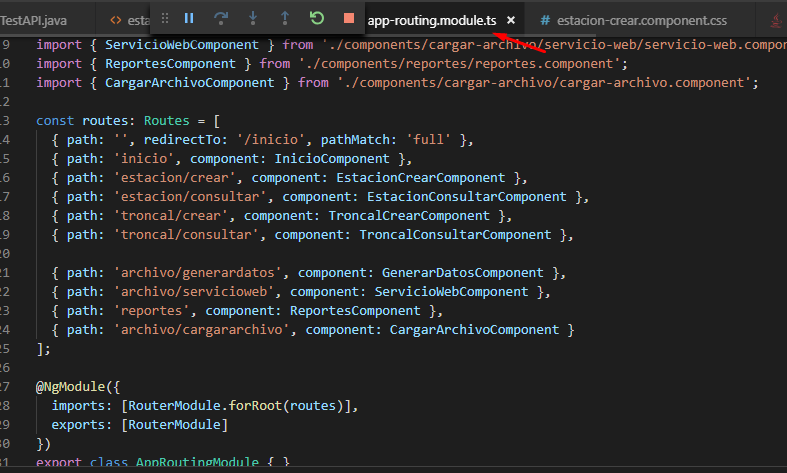
En la imagen anterior se evidencia el proceso de crear componentes y sus archivos.



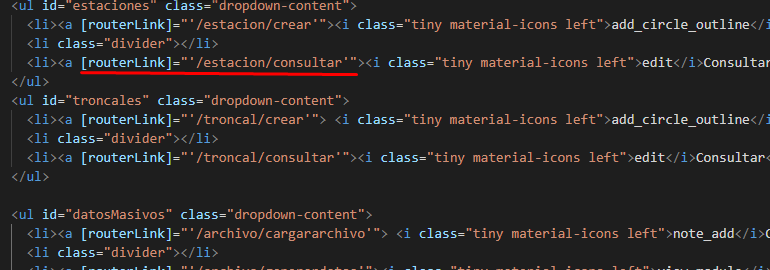
Figura, corresponde al archivo estación-consultar.html corresponde a la parte visual del proyecto con los campos respectivos. Es necesario una pantalla por cada formulario o funcionalidad.



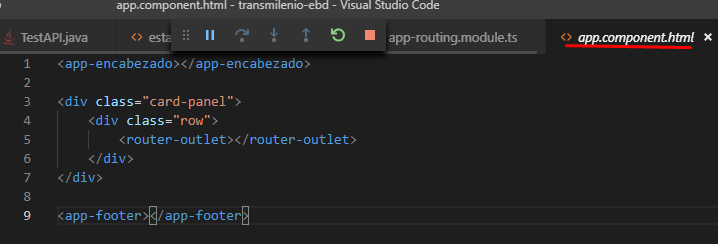
Figura, estacioncrear.component.ts corresponde a un archivo TypeScript para la manejar las variables que representan los que están en la base de datos, para ser mostrados en pantalla.



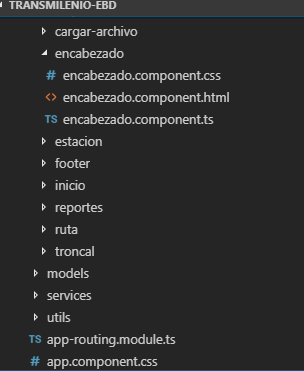
Figura, archivo app-routing.module.ts corresponde al manejo de rutas es necesario crear una ruta por cada pantalla creada para posteriormente ser llamado en el html como se muestra a continuación.



Figura, llamar un link con el atributo RouterLink.



Figura, corresponde a la pantalla principal llamando los componentes necesarios, cada componente representa un trozo de la pantalla en la interfaz.



Figura, estructura de carpetas evidencia de todos los componentes creados para cada una de las pantallas y los archivos de configuración, en la carpeta frontend.

**Nota:** En la carpeta adjunta se encuentran varias subcapetas explicación de esto acontinuacion:

Backend: corresponde al código fuente en Hibernate para comunicación con la base de datos.

Docs: se adjuntan scripts para la base de datos, presentación, y modelo de la base de datos.

Frontend: corresponde al código fuente en angular para la interfaz grafica.

A continuación, se muestra la evidencia con los pantallazos del proyecto

**Referencias:**

<https://www.transmilenio.gov.co/>

<https://angular.io/docs>

<http://hibernate.org/>

<https://www.oracle.com/index.html>