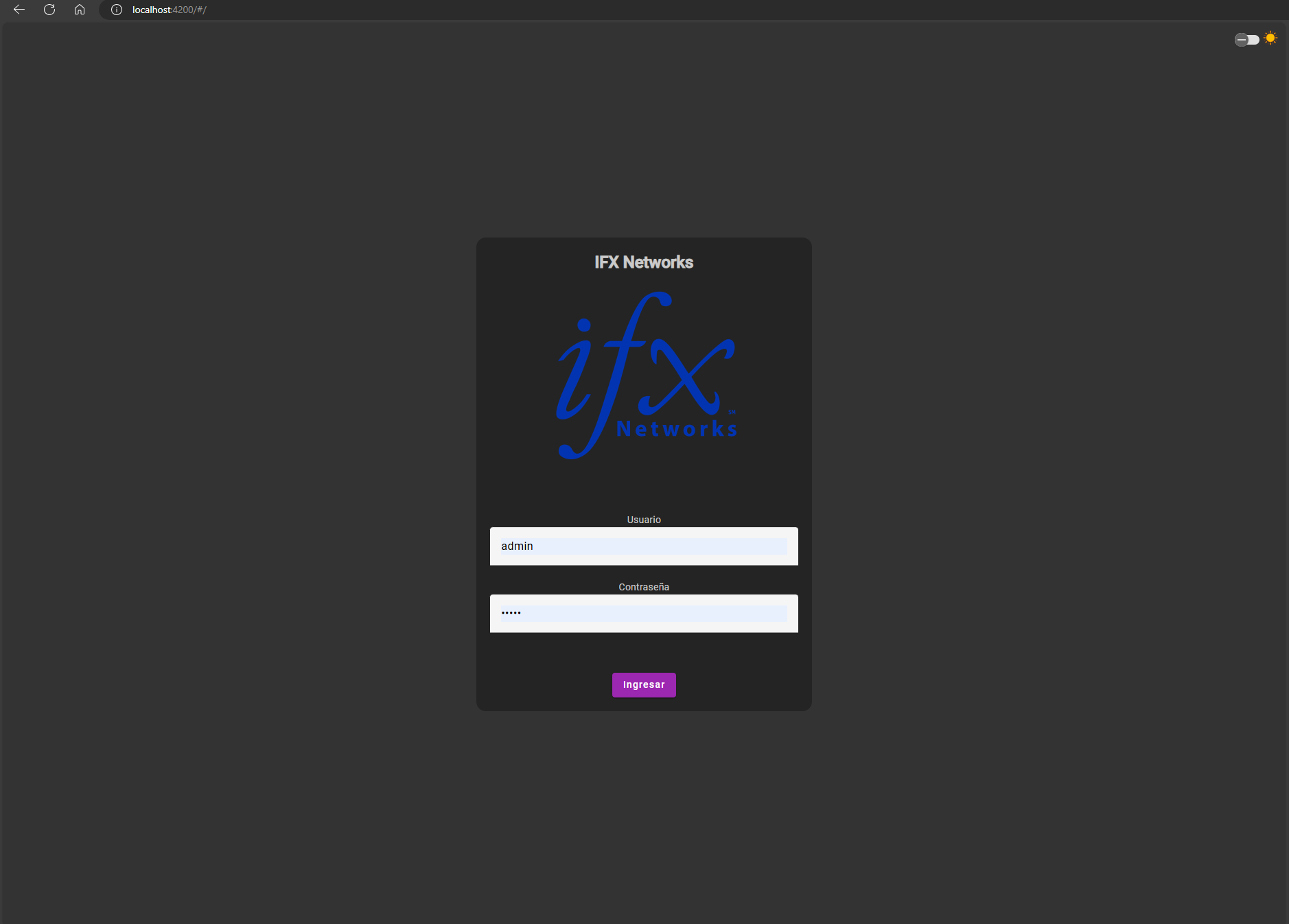
**Documentación FrontEnd (Angular 18)**

URL: http://localhost:4200/#/

Se agrega un login para el proyecto aprovechando la implementación del TokenJwr, proyecto basado en estilos .SCSS, totalmente responsive con adicionales basadas en mi experiencia en angular y su suite Angular Material, aprovechando el concepto de tematización.

Aplicación basada en los mejores conceptos de programación Client, Maneja LazyLoading “Carga diferida”, estilos reutilizables, iconos en modo SVG para no saturar peso



Nota “Se crearon 2 perfiles diferentes para poder ingresar, cada uno tiene su validacion”

Perfil: Administrador (Permite editar / eliminar / crear)

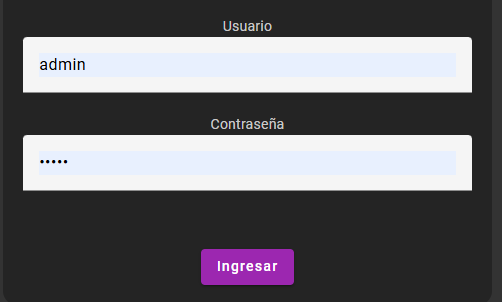
Usuario: [admin](mailto:admin@admin.com)

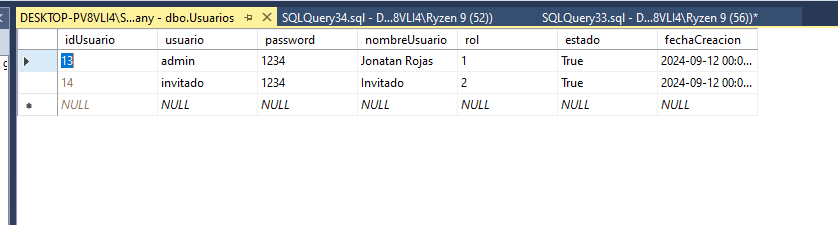
Contraseña: 1234

Perfil: Invitado (No permite editar / eliminar / crear)

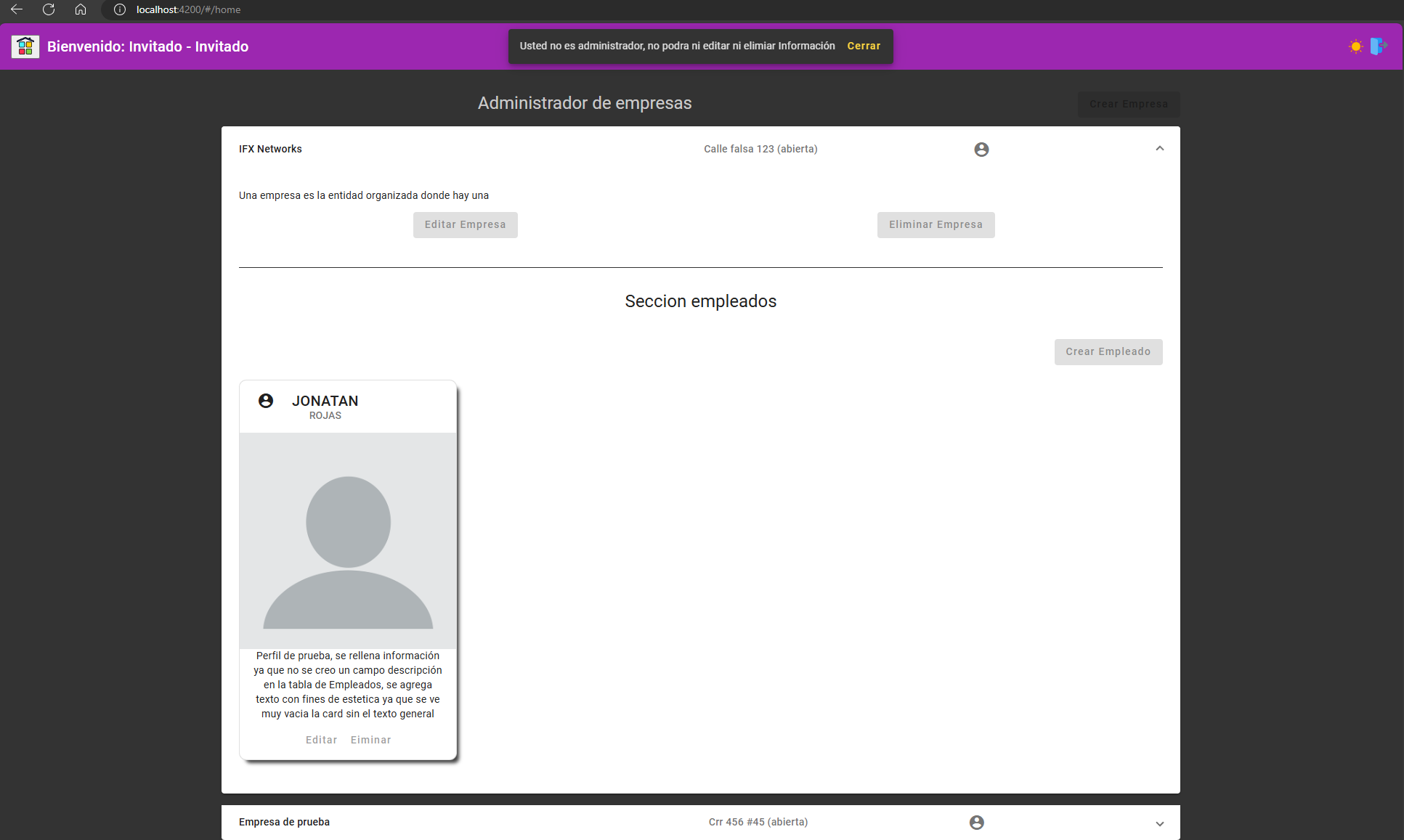
Usuario: [invitado](mailto:piloto@piloto.com)

Contraseña: 1234

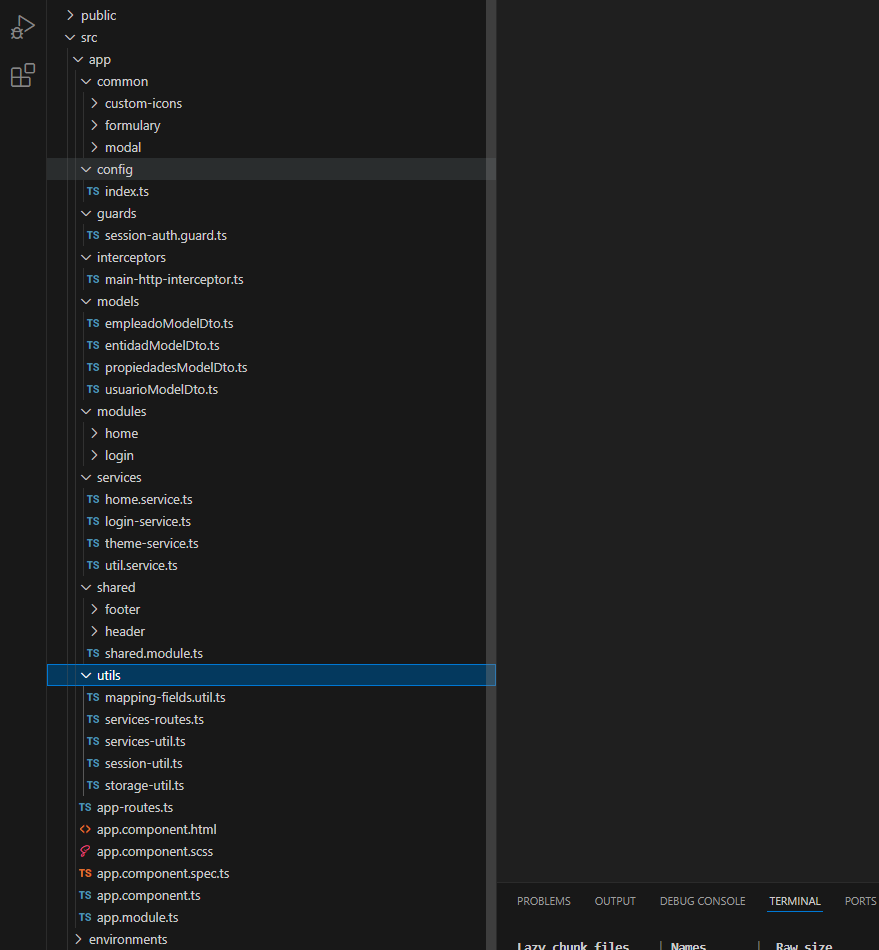




Perfil Invitado – Restringidas acciones de edición, creación o eliminación

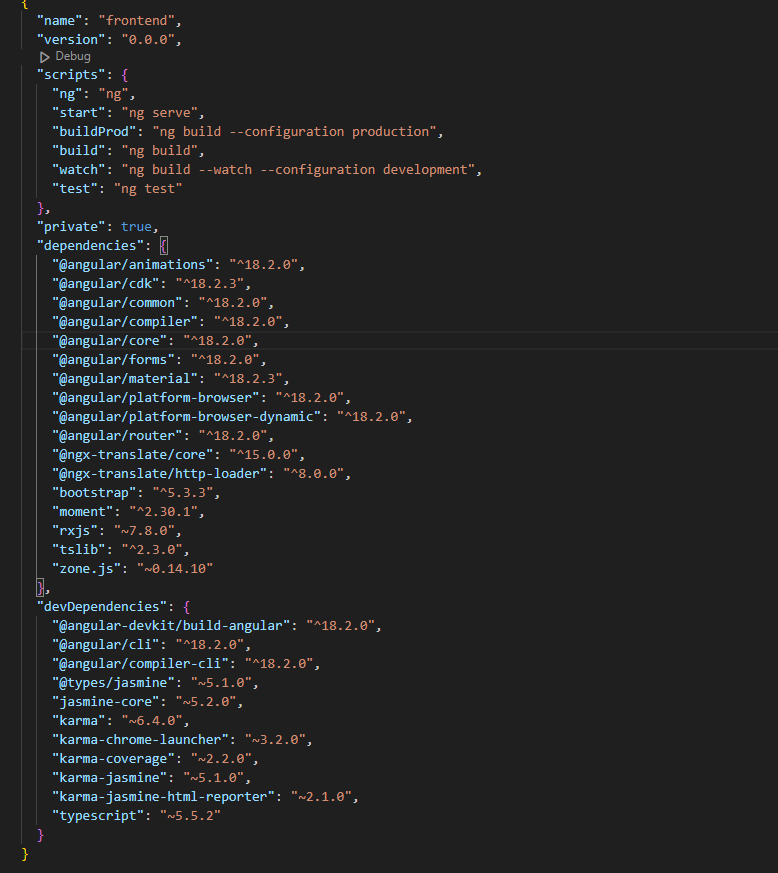


Estructura



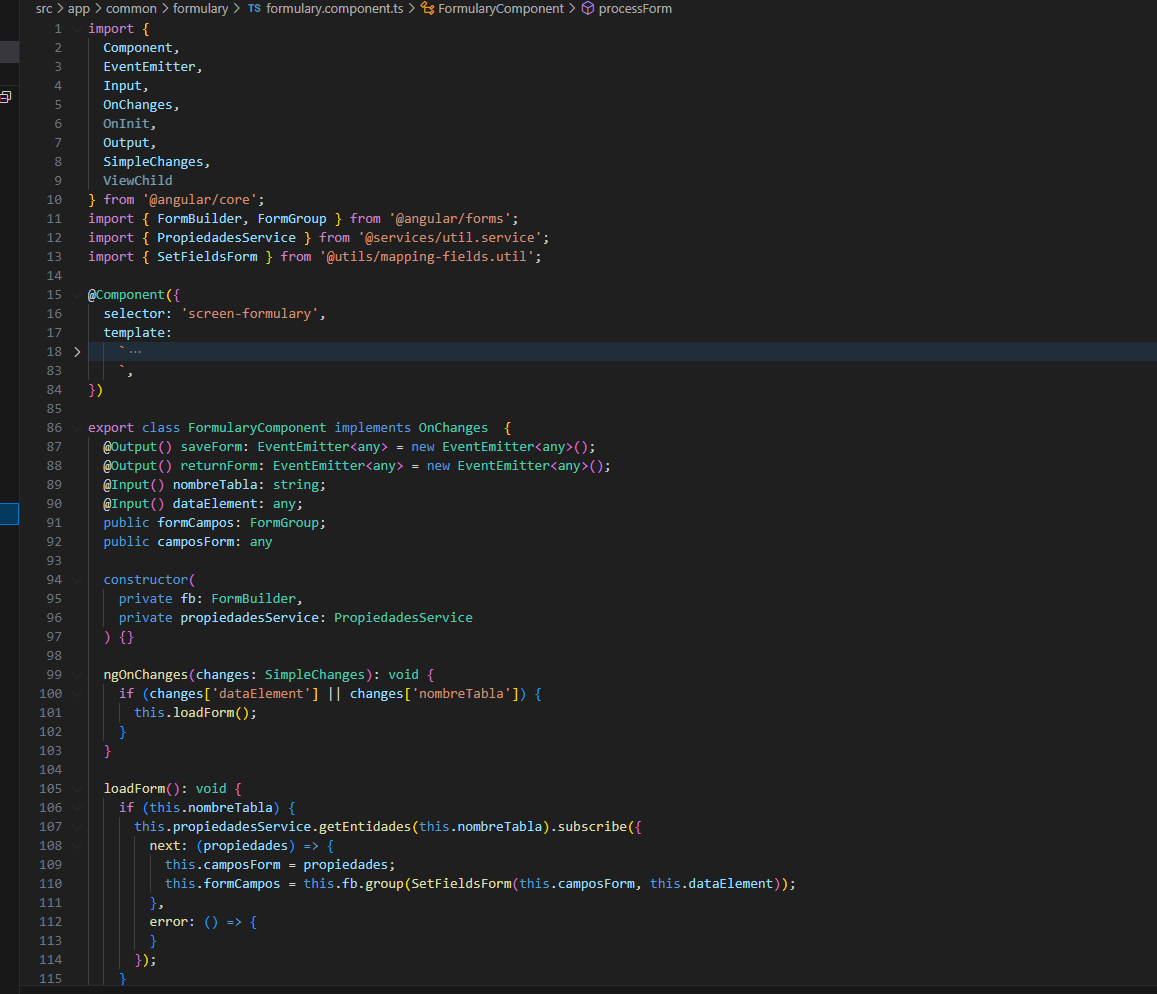
* Package.json

Configuración del proyecto, despliegues en modo producción y modo de lint con el cual se puede verificar reglas de sintaxis tanto en SCSS como en TS.



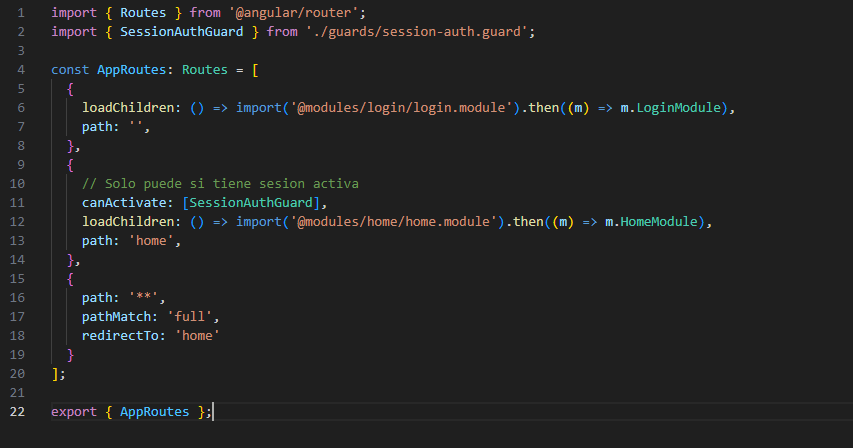
* Componente custom para cargar cualquier formulario

Es el encargado de consultar el esquema de la base de datos de una tabla que se envía como parámetro, con estos datos se puede crear un formulario reactivo con los campos, tamaños obligatoriedad, y tipo de campo



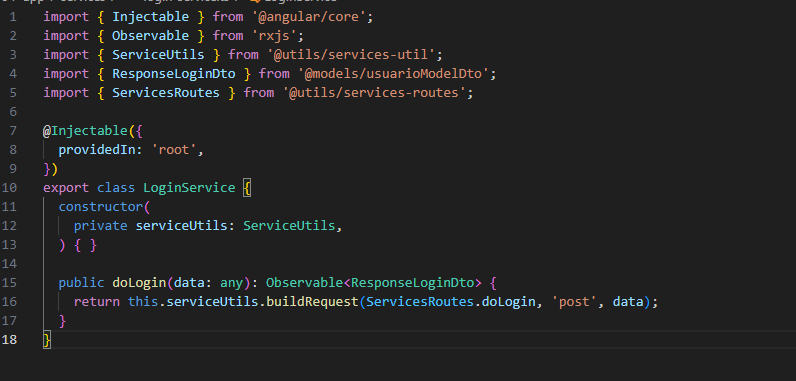
* Routing Login

Se evidencia la implementación de carga diferida



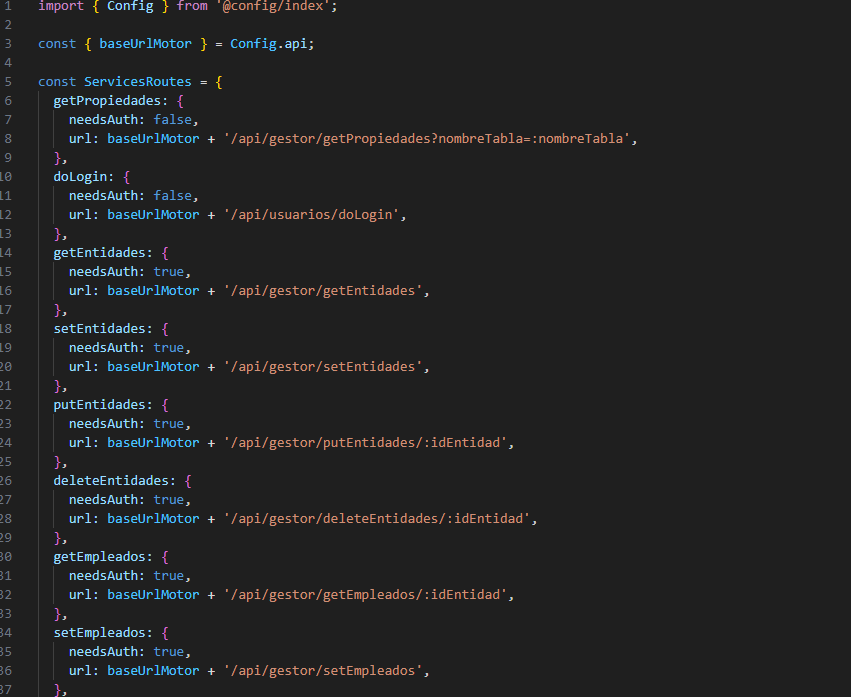
* Ejemplo servicio login-service.ts

Ejemplo de consumo de un servicio con Injectable, y Observables para el correspondiente llamado al servicio API REST .Net Core 3.1



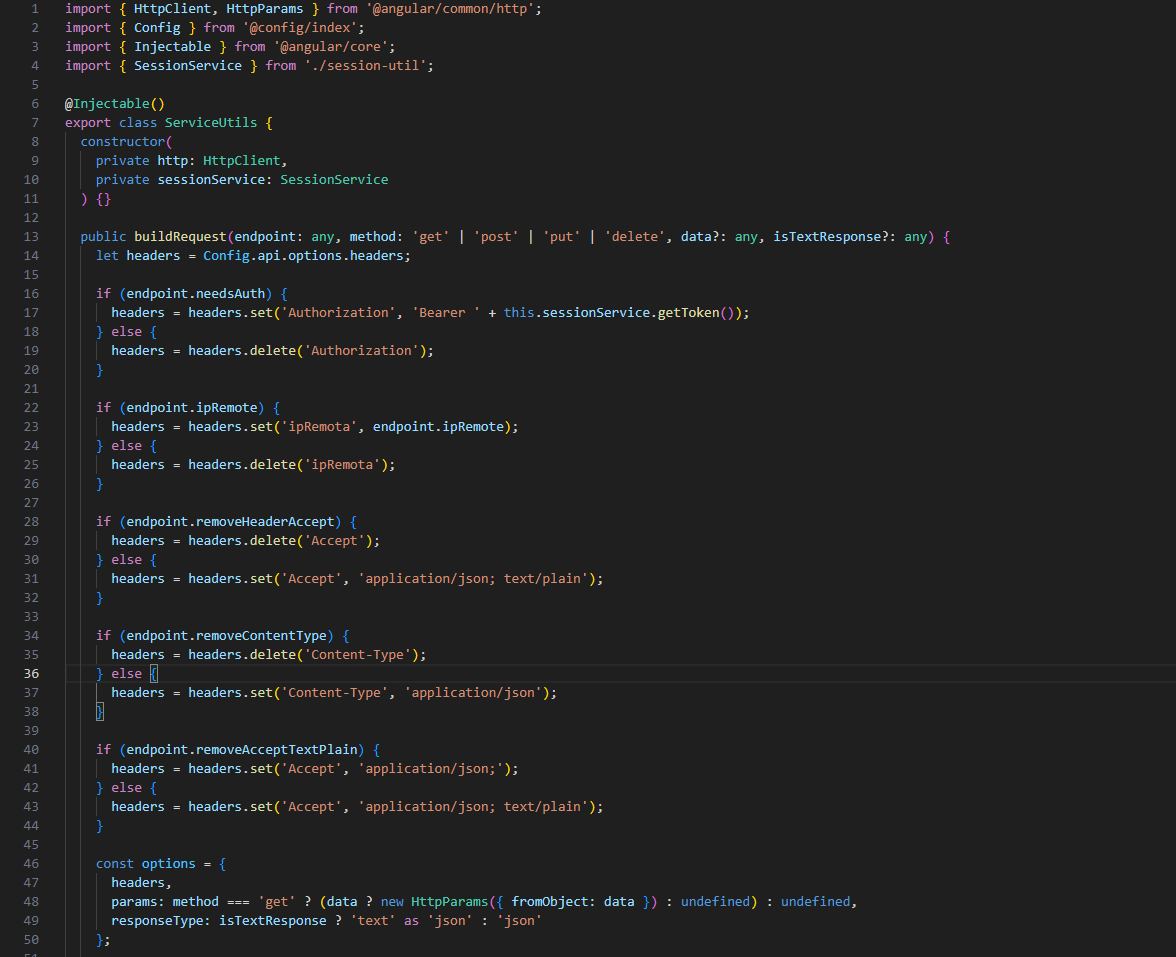
* Ejemplo service-routes.ts

Se evidencia la clase donde se almacenan todas las peticiones creadas en el API, junto con una opción reutilizable llamada needsAuth la cual es por si se requiere envió de token en las cabeceras Http



* Services-utils.ts

Utilidad reutilizable la cual arma todos los llamados al servicio, y que según su lógica organiza los arrays para peticiones GET, SET, POST, DELETE, y las cabeceras necesarias de cada uno



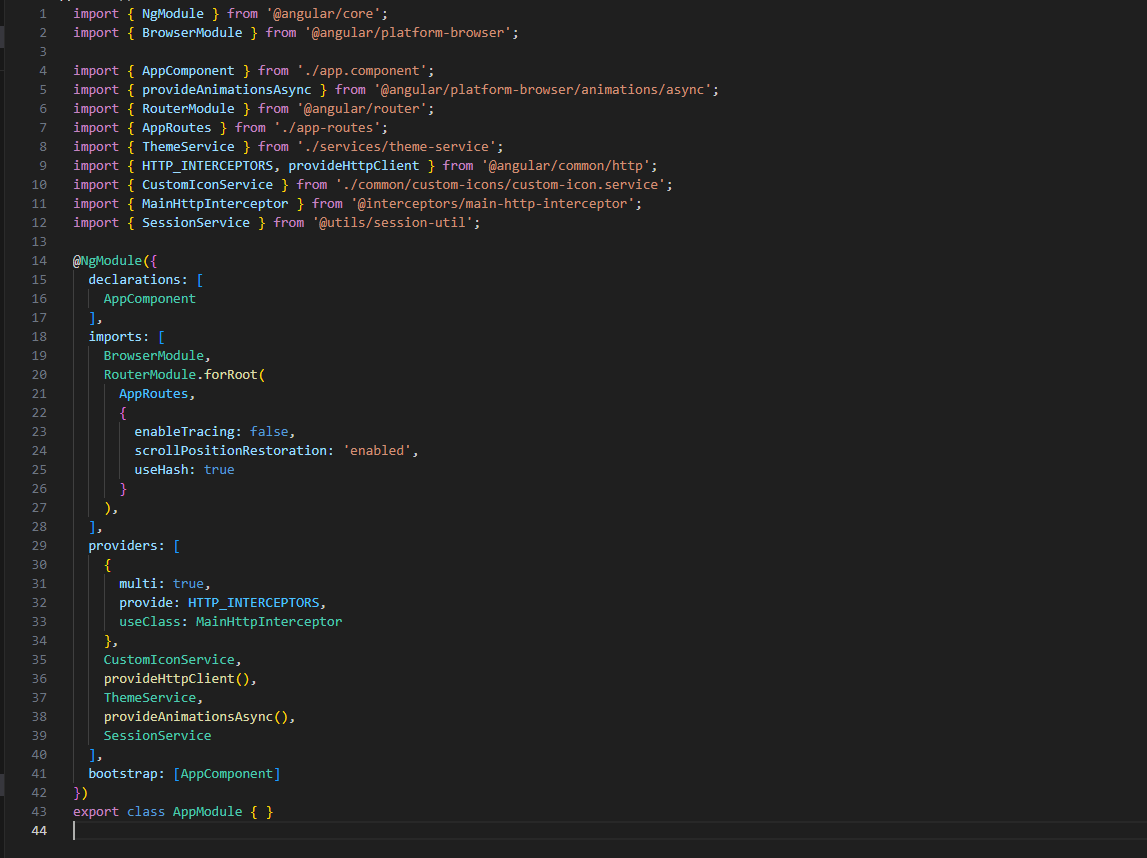
* Sesión-service.ts

Esta clase es de importancia la cual almacena a las variables local.storage y session.storage del navegador para con ello poder guardar datos globales como lo puede ser el token que iría en las cabeceras cuando se requiera



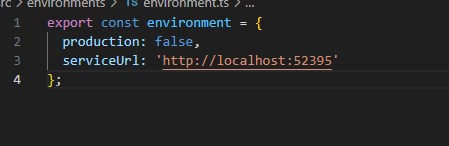
* app.module.ts

Modulo principal del proyecto el cual carga los módulos y que se utilizan en todo el proyecto, como lo puede ser el HttpClientModule que en la versión Angular 18 es provideHttpClient(), o el llamado a los iconos SVG que se usa, adicional para interceptores.

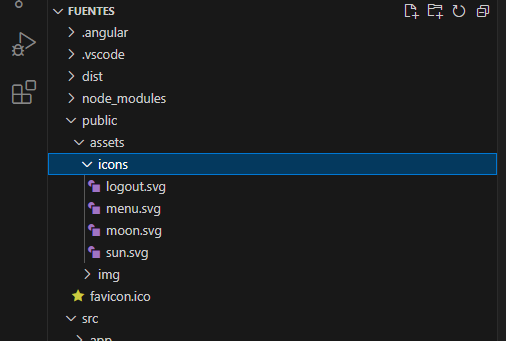


* enviroment.ts

clase la cual tiene le ruta del api que consume

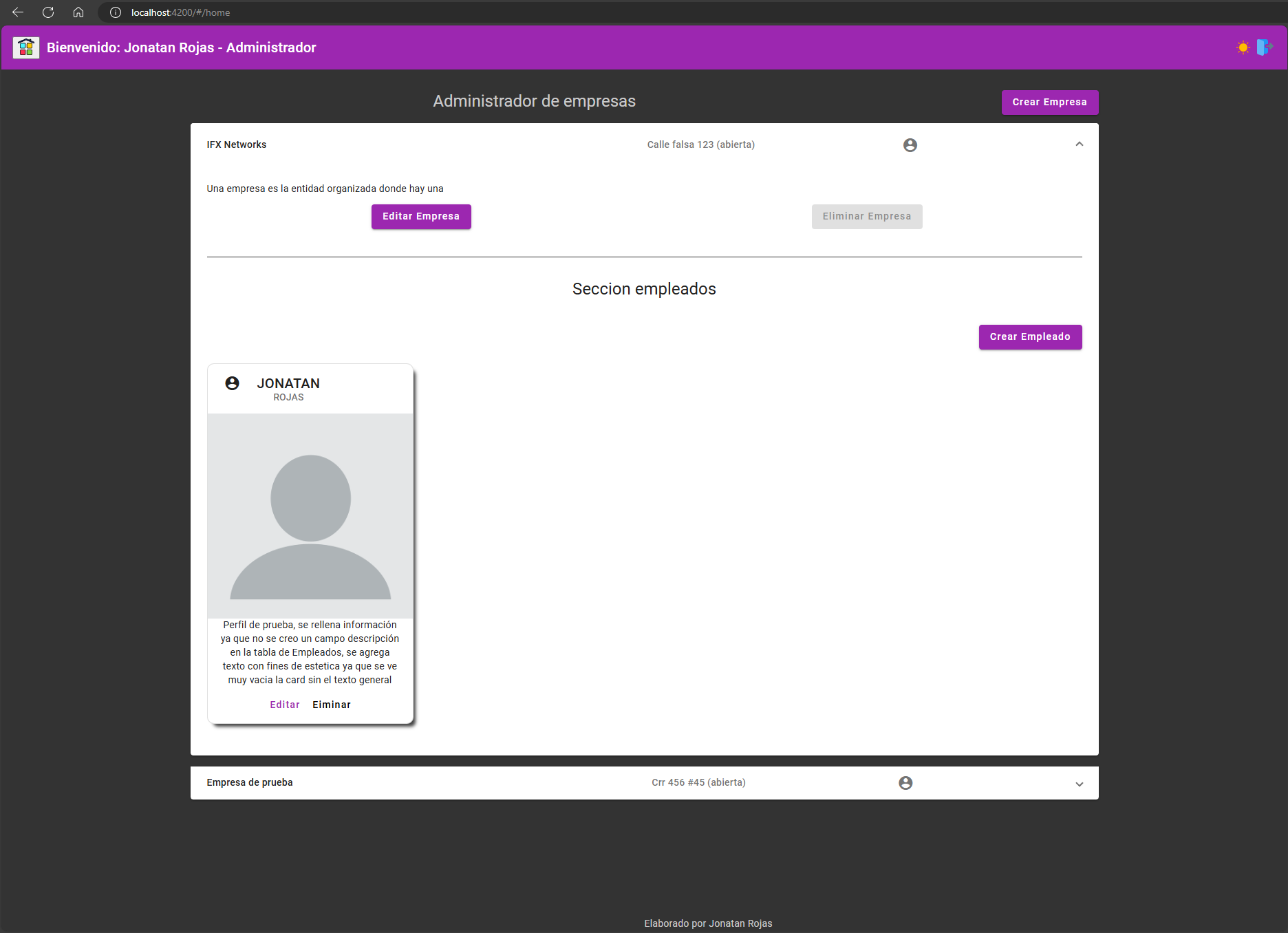


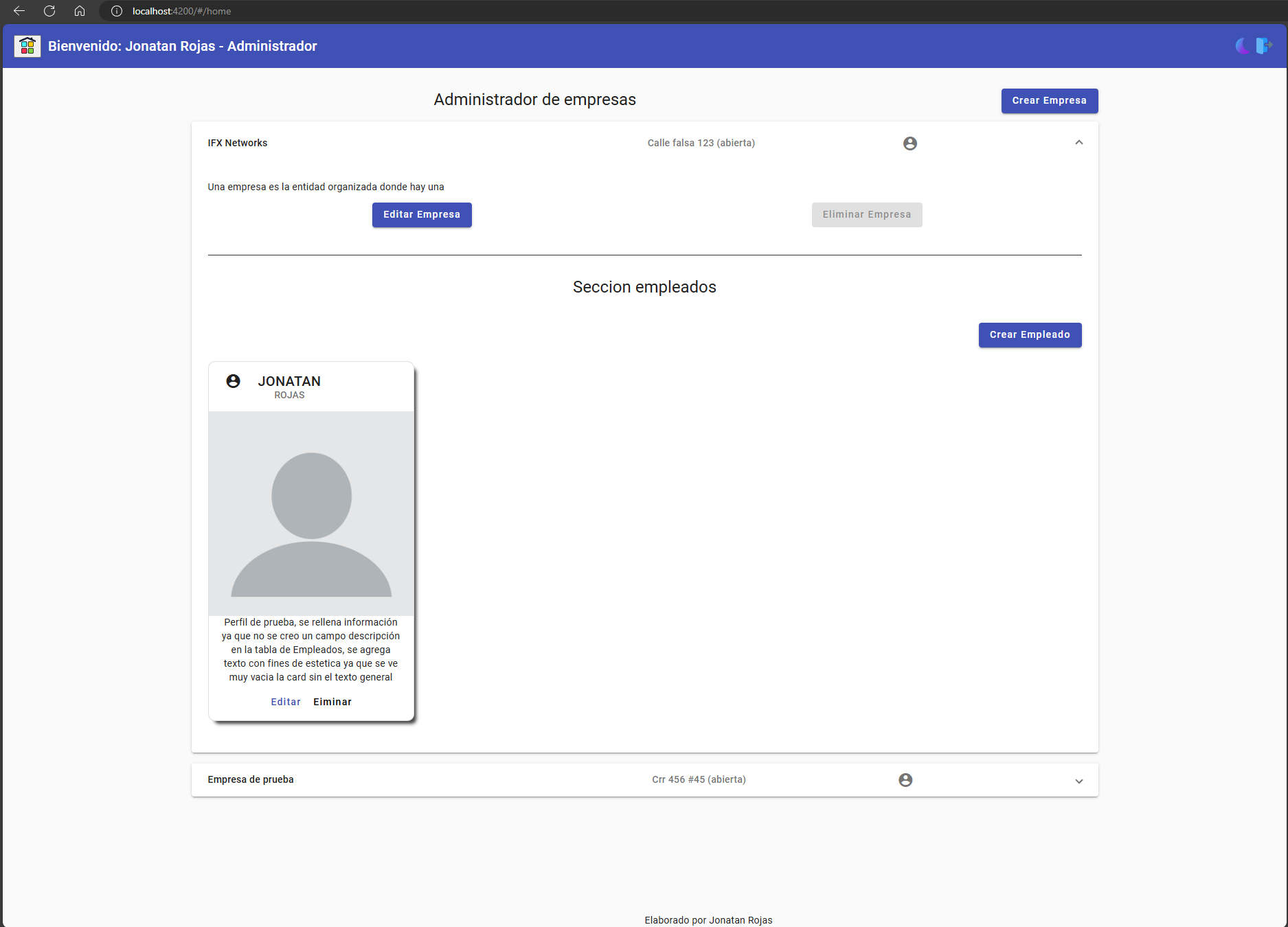
* **Assets**
* Iconos



* Dashboard o Home

Sección que accede luego de ingresar al login, como se observa se utilizan todos los temas de Angular material para que la experiencia visual del usuario sea lo mejor posible, se destaca en esto la utilizacon de theming de angular logrando así una experiencia visual cambiante de colores.





* Theming

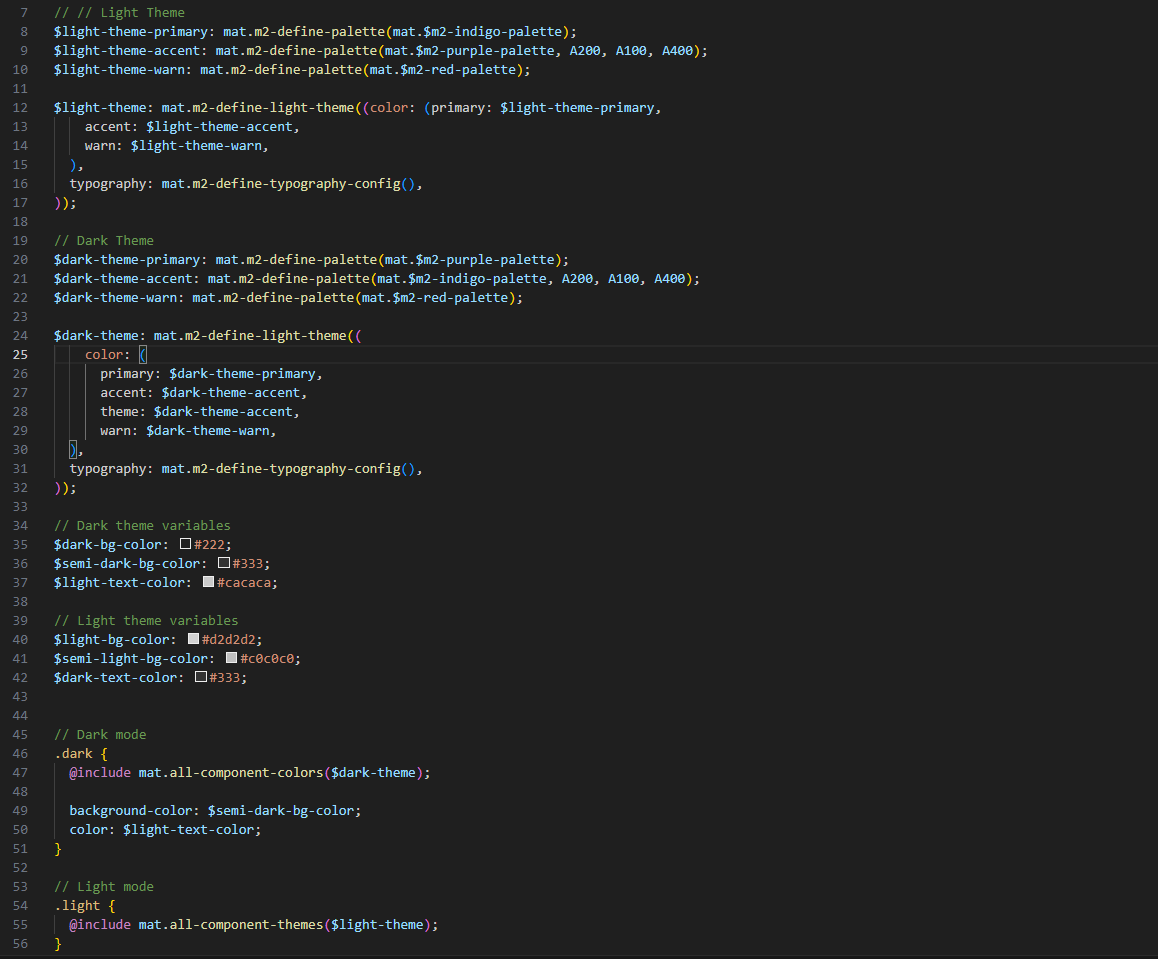
A continuación de detalla el pluss agregado a esta prueba una experiencia visual agradable, con dos temas agregados al proyecto de la prueba

Como se puede ver agregue 2 temas con su correspondiente previsualización de cómo podría quedar

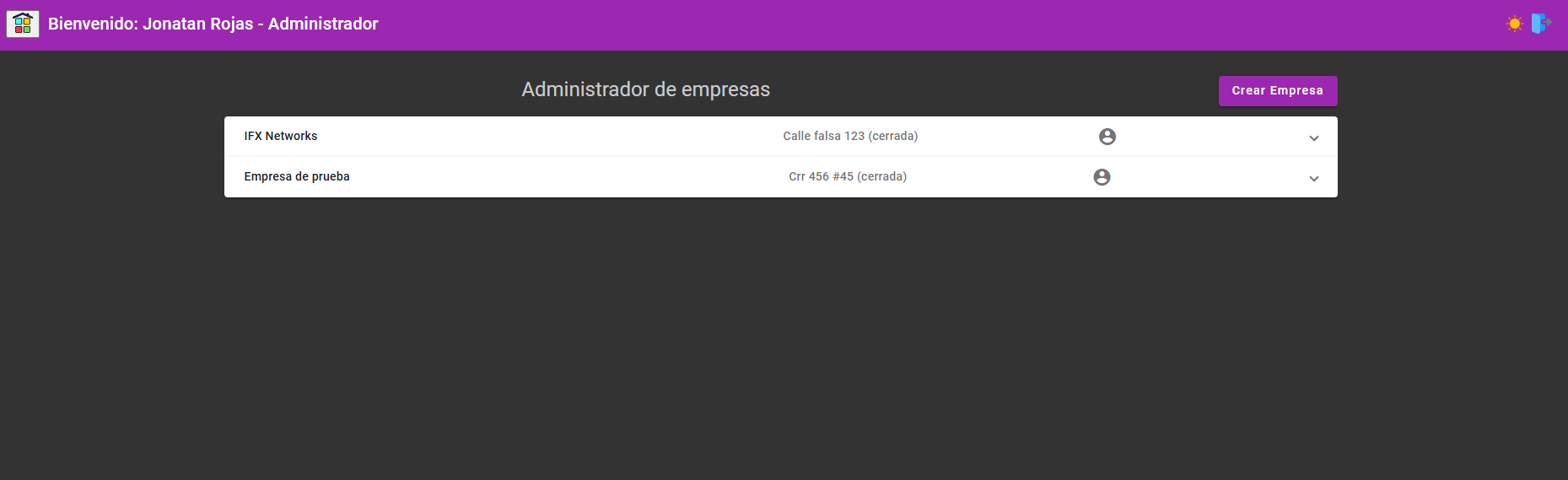
* Theme.scss

Basado en la documentación de la suite Angular material se agregan 2 temas que podrían ser más cambiantes a través de observables y etiquetas [ngClass



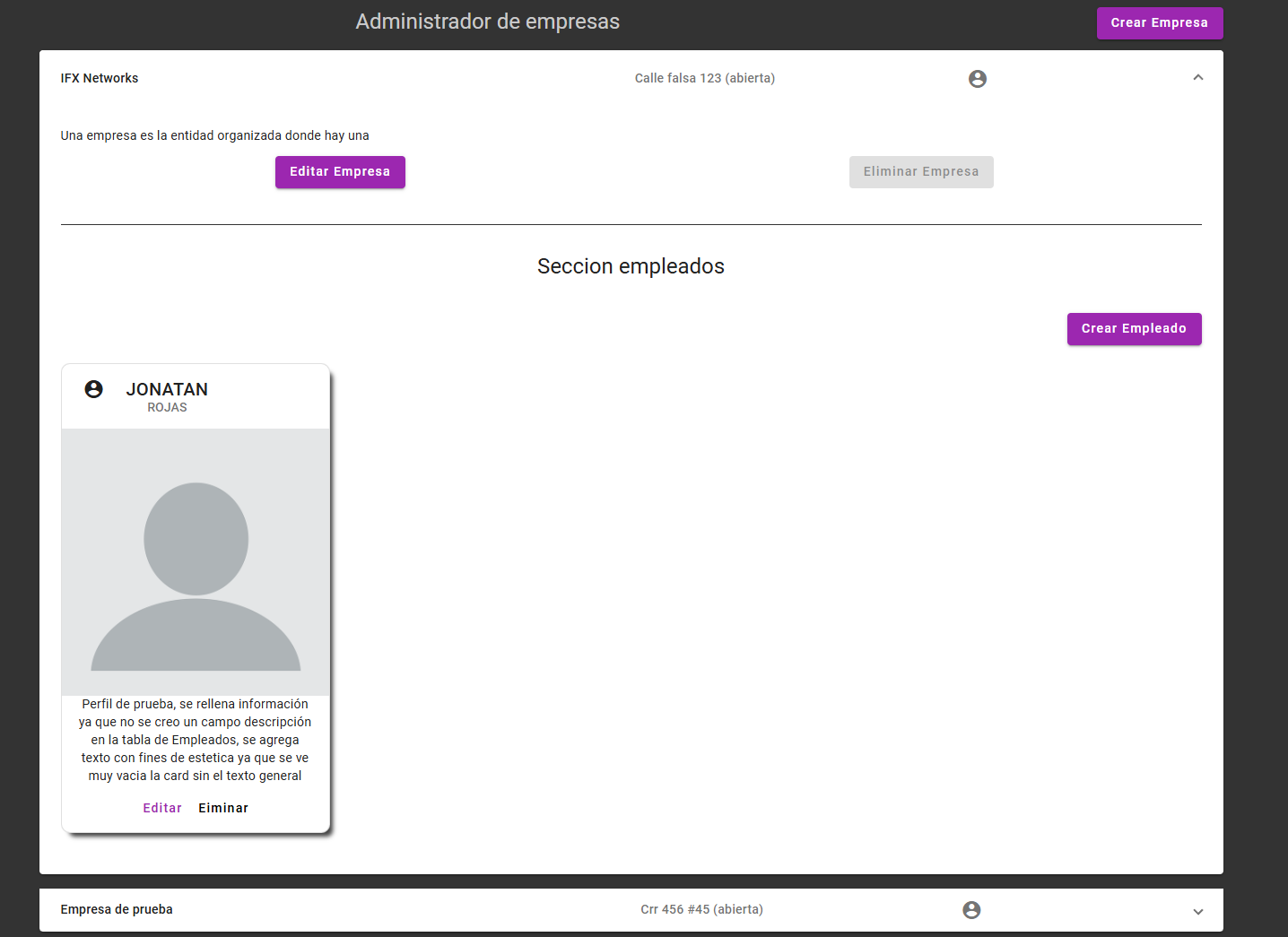
* Componente Home

Explicación componente home, en el cual se implementa en CRUD del api REST, y mejoras visuales por medio de iconos, modulo acordion, mat-icon material con complemento expanded, filtro de la tabla y paginación de la misma, modalService usado de Bootstrap.



* Función expanded

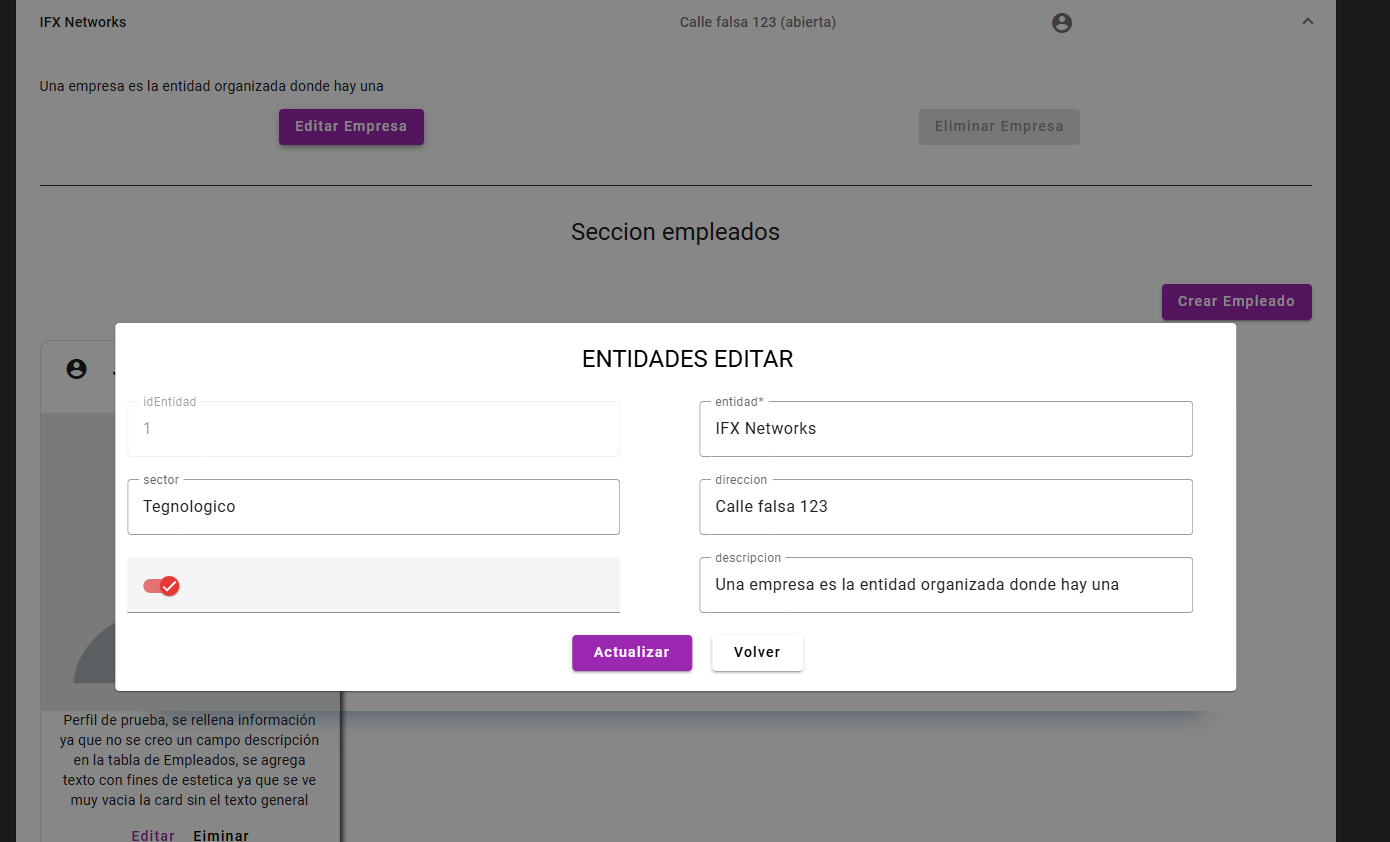
La cual se activa al dar click sobre la fila, y muestra la los detalles de la empresa seleccionada y si tiene empleados también los carga

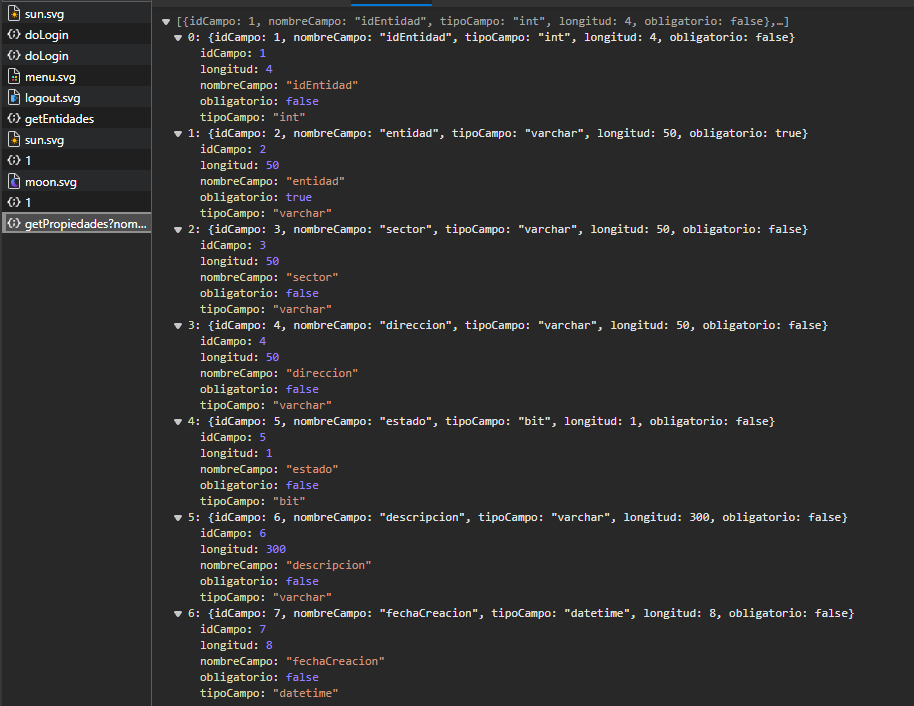


* Editar Empleados o Empresas

La opción editar o crear, se crealizo mediante un componente reuitilizable llamado formulary.component.ts, el cual por medio de @Input recibe el nombre de la tabla para poder extraer sus propiedades y los datos del elemento a editar esto nos facilita en gran medida ya que un solo componente se encarga de mapear cualquier tipo de formulario, haciendo mas optima la aplicación, y por medio de @Output recibe los datos cuando el formulario es valido

Como se puede ver el modal se llama al componente el cual mapea cualquier tipo de datos dependiente la tabla SQL





Validaciones de angular Forms, ReactiveForms, y selectFormModule para traer las empleados o entidades creadas.



SnackBar con confirmación del servicio mostrando el campo message, que responde el servicio

