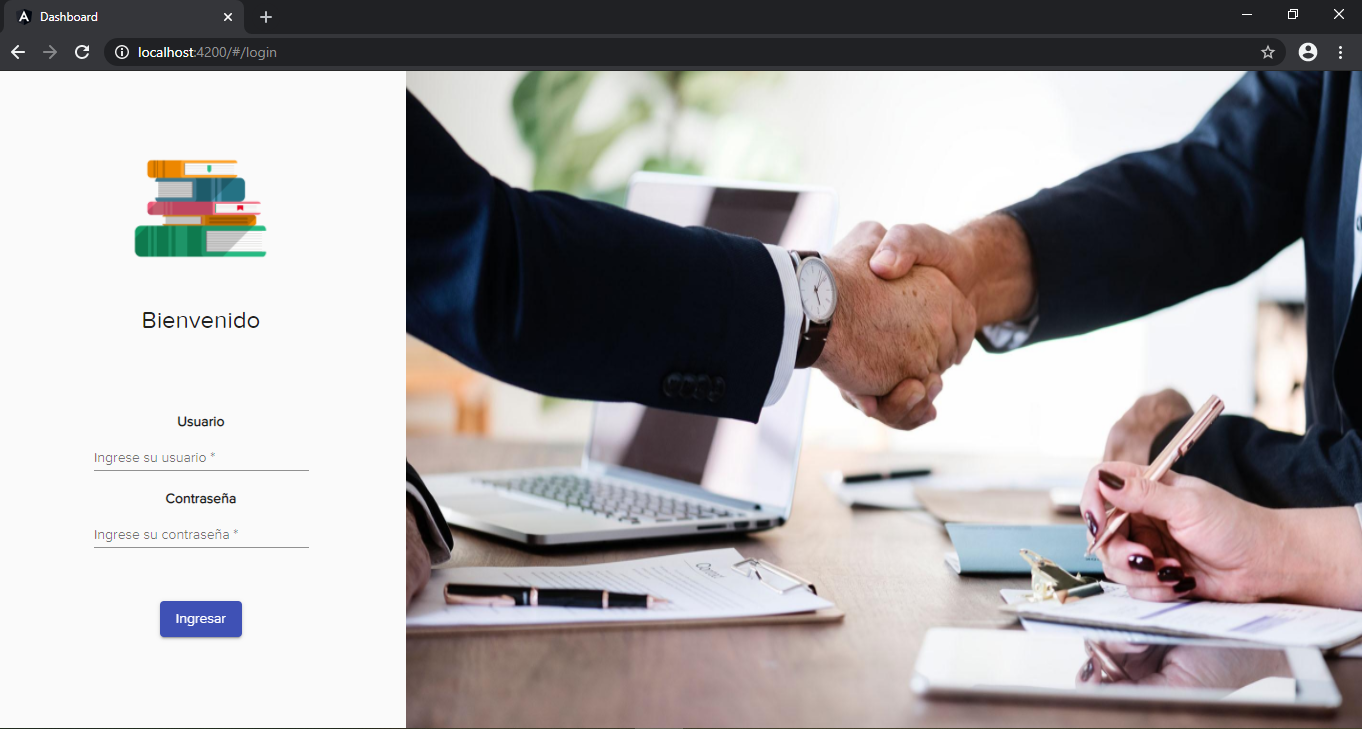
**Documentación FrontEnd (Angular 8)**

URL: <http://localhost:4200/#/login>

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

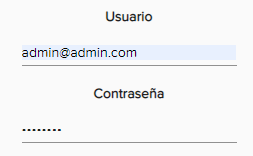
Aplicación basada en los mejores conceptos de programación Client, Maneja LaziLoading “Carga diferida”, estilos reutilizables, iconos en modo SVG para no saturar peso

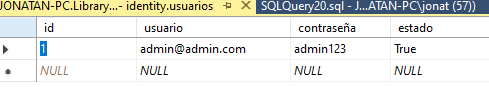


Nota “El usuario para poderse loguear es”

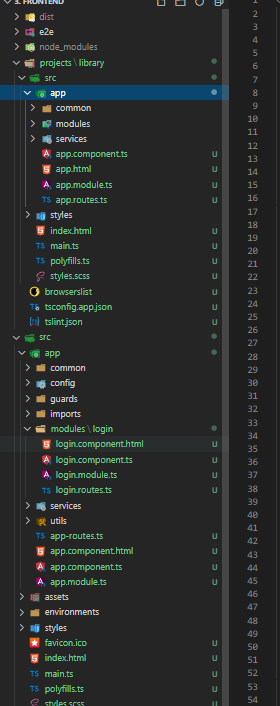
Usuario: [admin@admin.com](mailto:admin@admin.com)

Contraseña: admin123





Estructura



* Package.json

Configuracion 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.



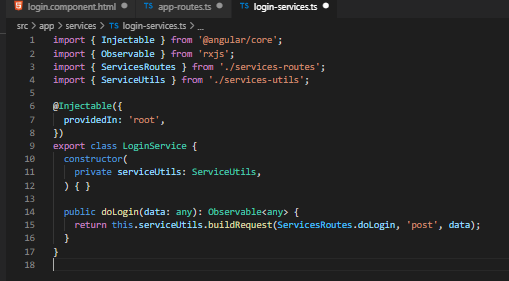
* Routing Login

Se evidencia la implementación de carga diferida



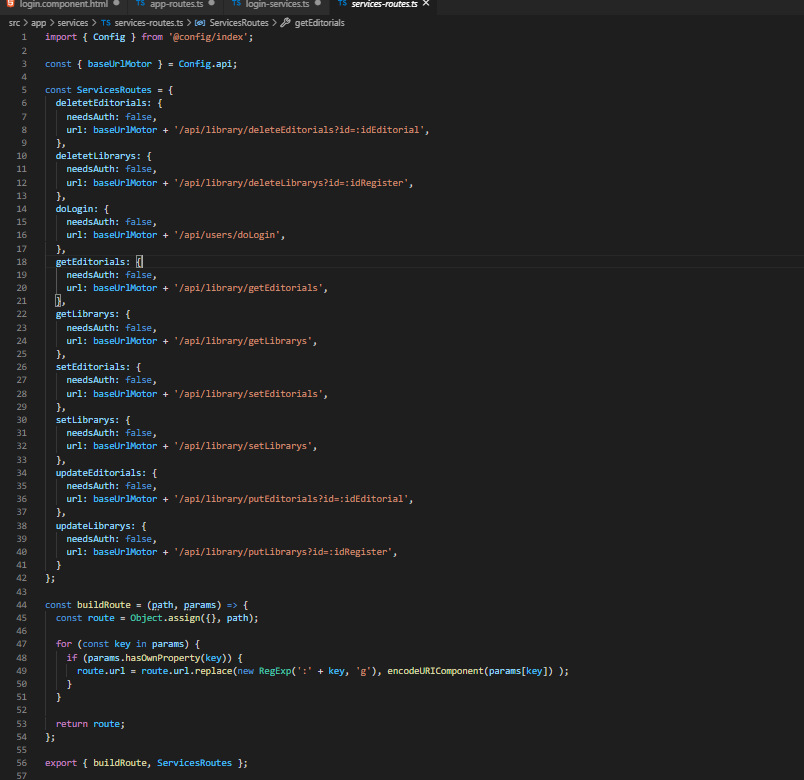
* Ejemplo servicio login-service.ts

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



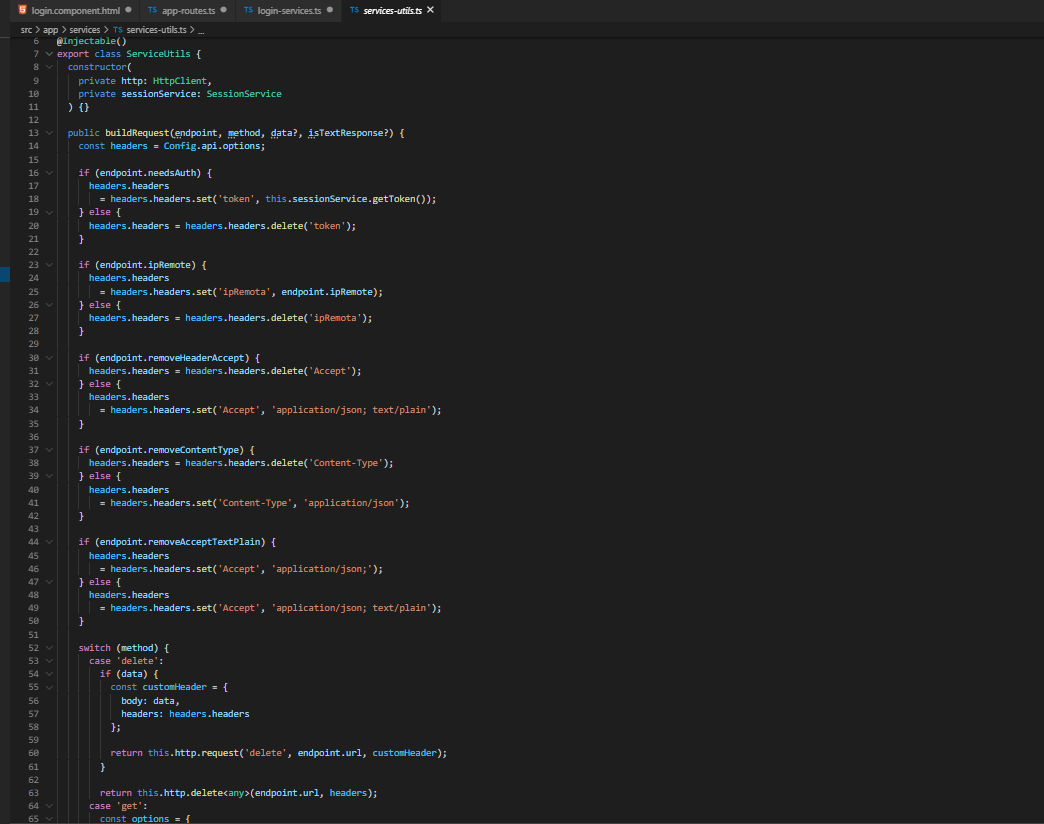
* Ejemplo service-routes.ts

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 envio de token en las cabeceras Http



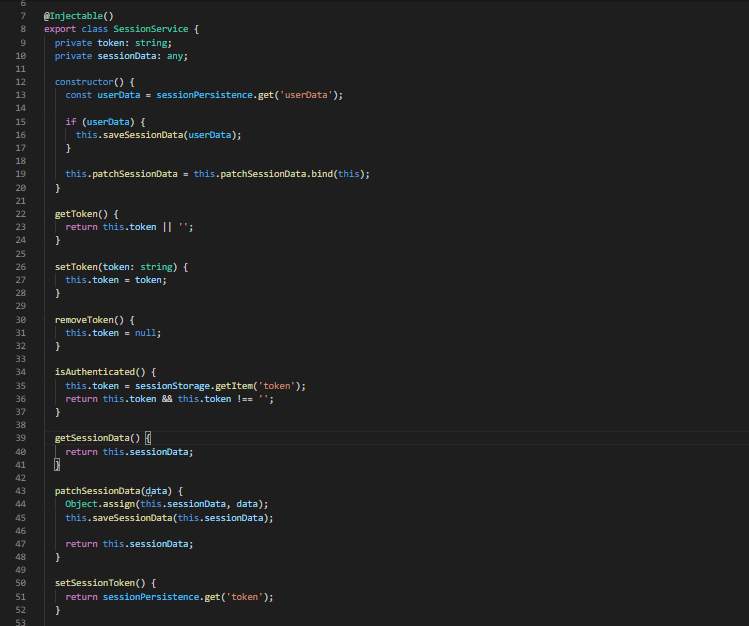
* Services-utils.ts

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



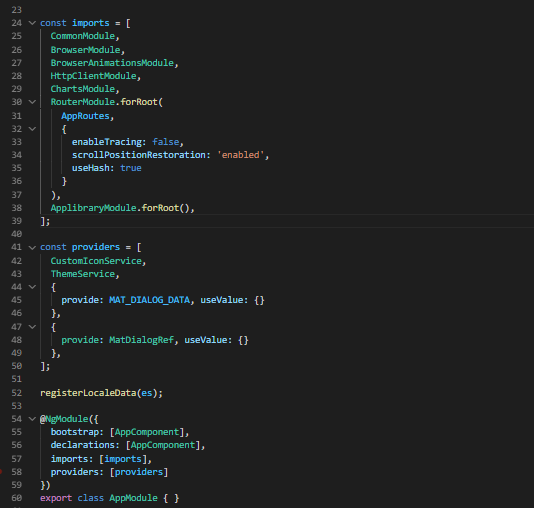
* Sesión-service.ts

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



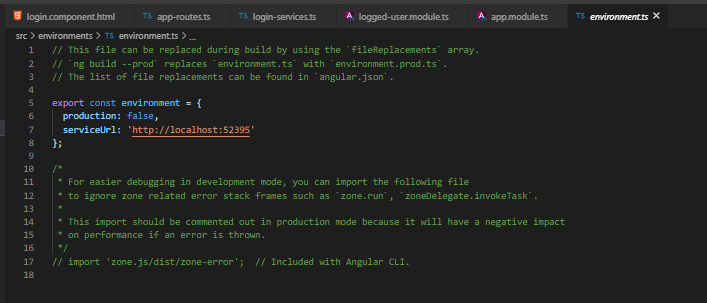
* app.module.ts

Modulo principal del proyecto el cual carga los moodulos necesarios para cargar la ventana login, por otro lado carga el modulo del proyecto de librarys llamado AppLibraryModule, detallado mas adelante.



* enviroment.ts

clase la cual tiene le ruta del api que consume



* **Assets**
* Fonts

Fuente propia cargada en el proyecto



* Iconos

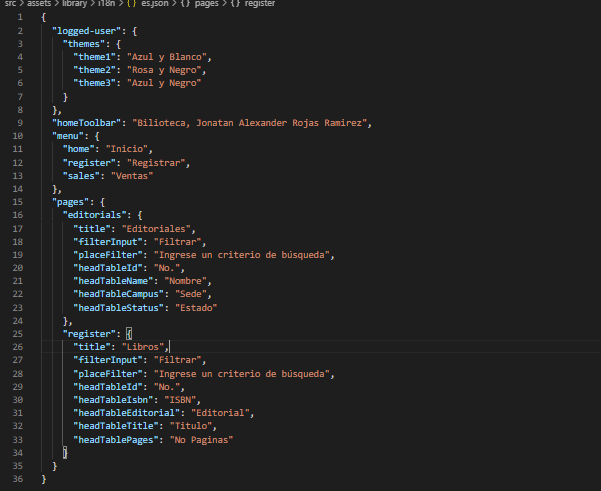


* Multilenguaje

Esta aplicación de angular soporta el multilenguaje, configurados como modo de pruebas Ingles y español

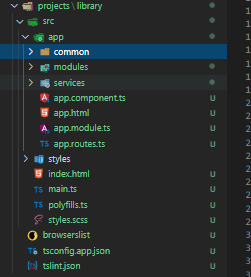


* Ejemplo json, textos en español



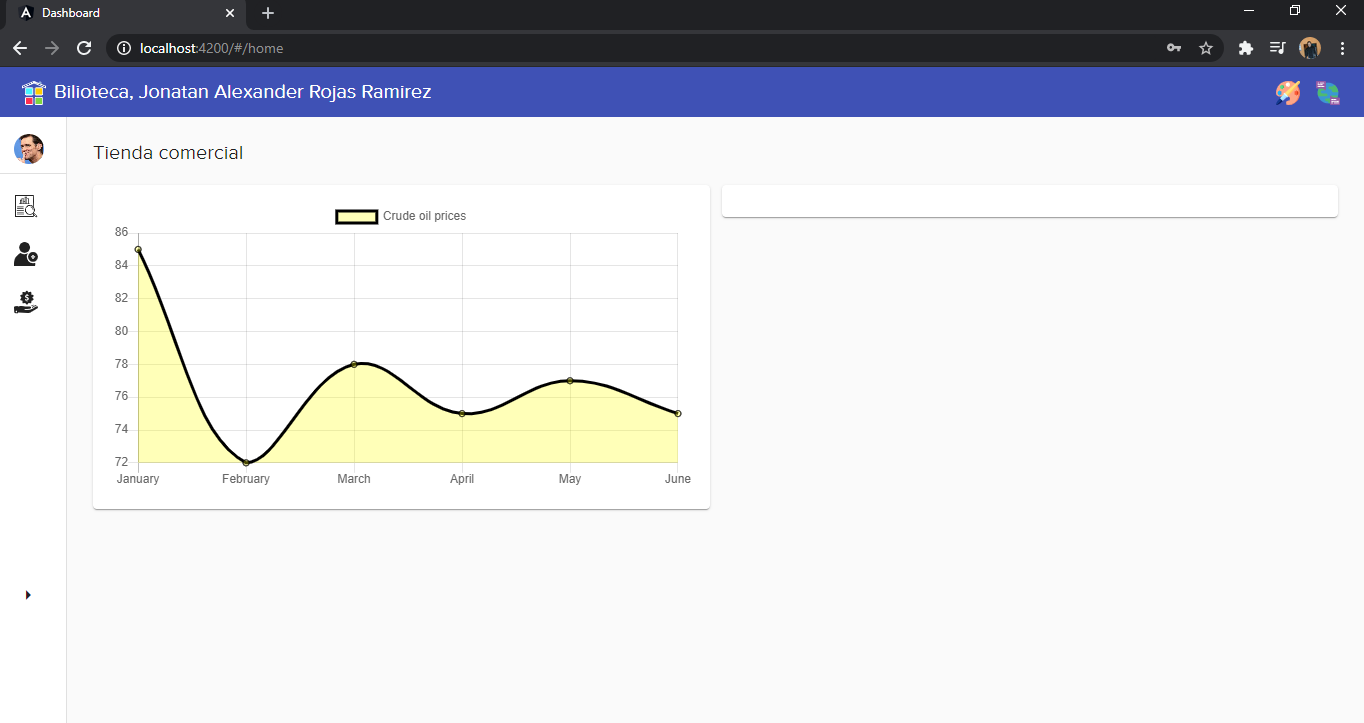
* Proyecto Librarys

Este proyecto esta dentro de la aplicación la cual se despliega luego del login, todo tratando de llevar las capas y mejor aprovechamiento de cada uno de sus componentes



* Dashboard

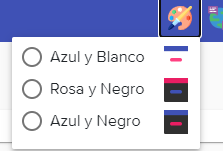
Sección que accede luego de ingresar al login, como se observa se utilizan todos los temas de Angular material para que la experiencia visual del usuario sea lo mejor posible, se destaca en esto la utilizacon de theming de angular logrando asi una experiencia visual cambiante de colores.





* Theming

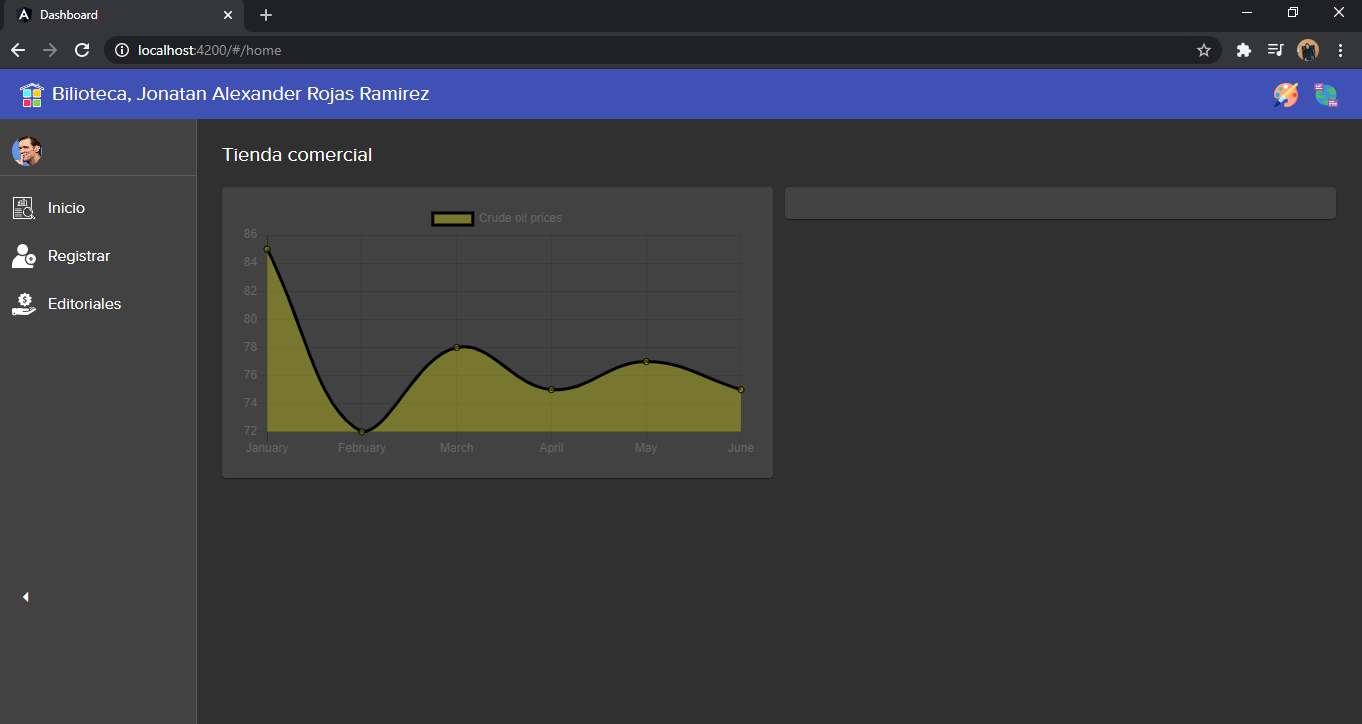
A continuación de detalla el pluss agregado a esta prueba que, a una experiencia visual agradable, con tres temas agregados al proyecto de la prueba



Como se puede ver agregue 3 temas con su correspondiente previsualización de cómo podría quedar

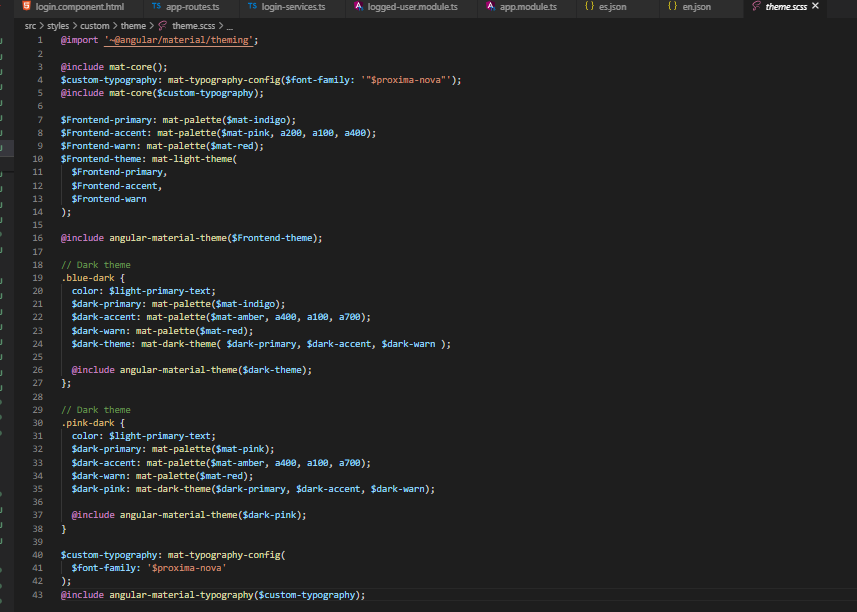
Explicacion:



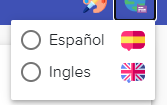


* Theme.scss

Basado en la documentación de la suite Angular material se agregan 3 temas que podrían ser más cambiantes a través de observables y etiquetas [ngClass], además de cambiar la typografia de los temas a la cargada llamada próxima-nova



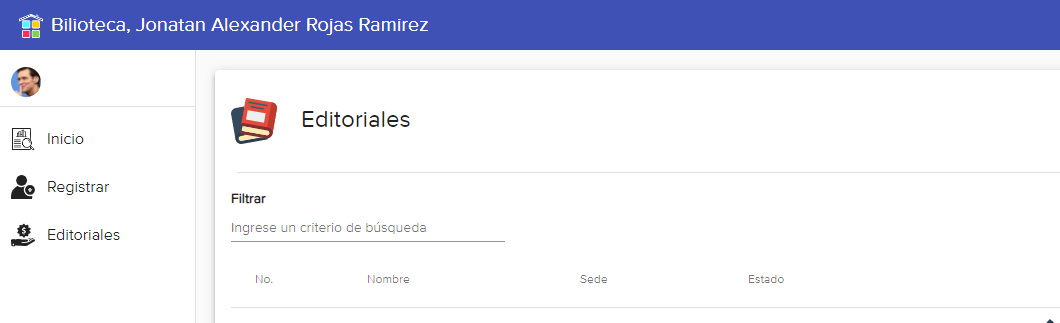
* Multilenguaje



Se implementa el modulo ngx-translate-multi-http-loader de npm con el cual el poryecto de puede traducir a varios idiomas.

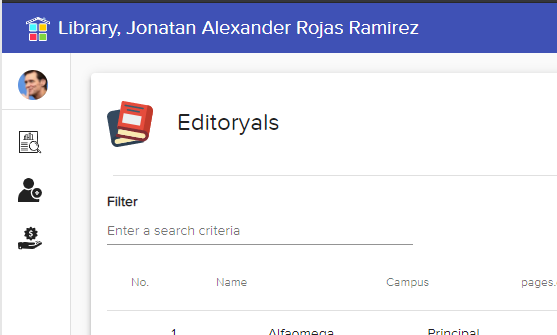
* Español

Ejemplo modulo español



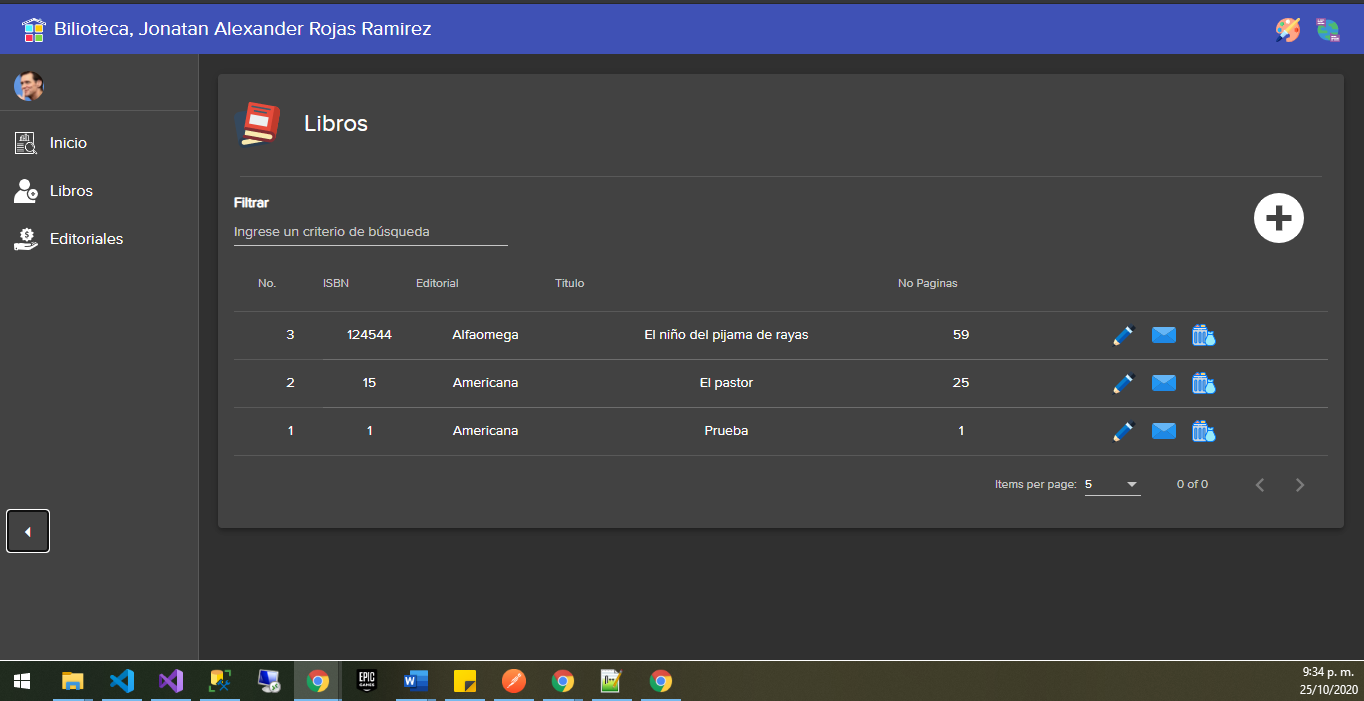
* Ingles

Ejemplo modulo ingles



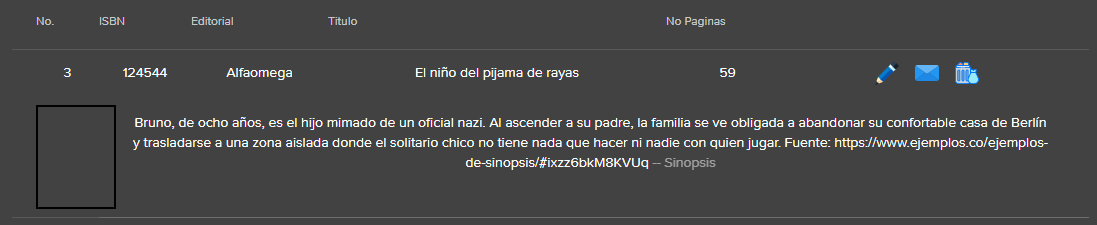
* Componente Libros

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



* Función expanded

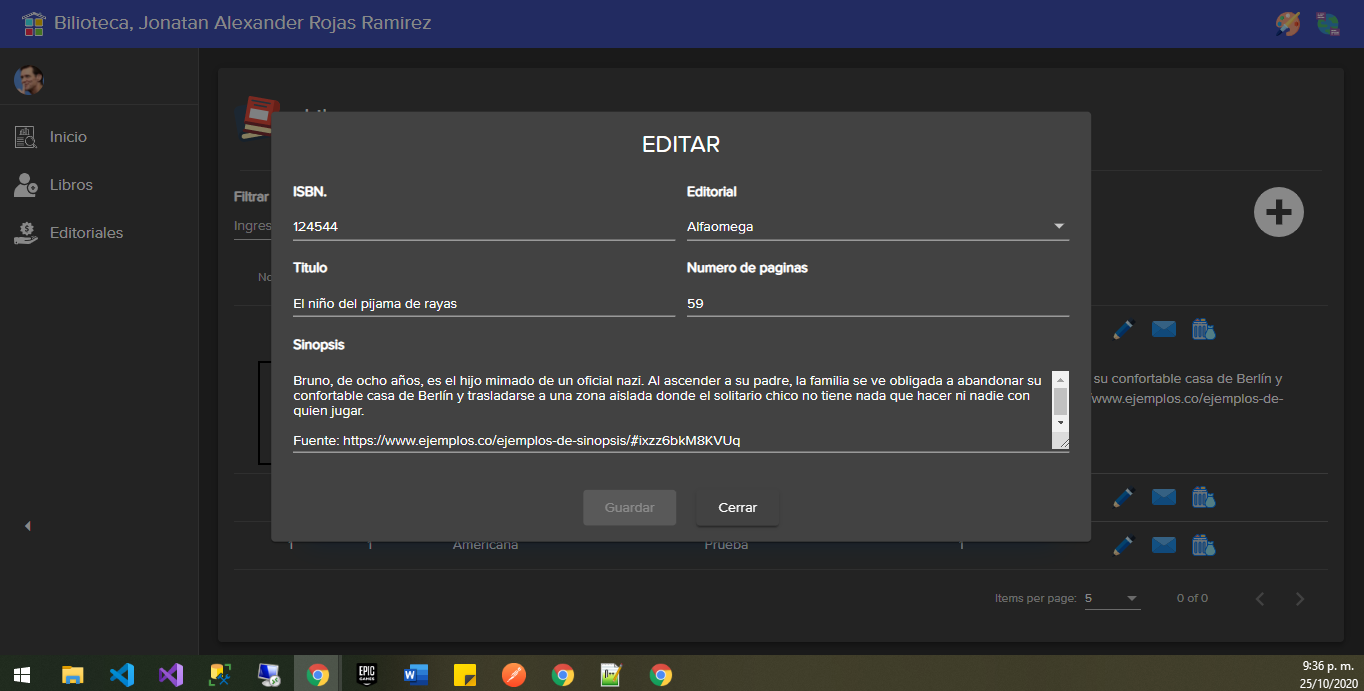
La cual se activa al dar click sobre la fila, y muestra la sinopsis del libro



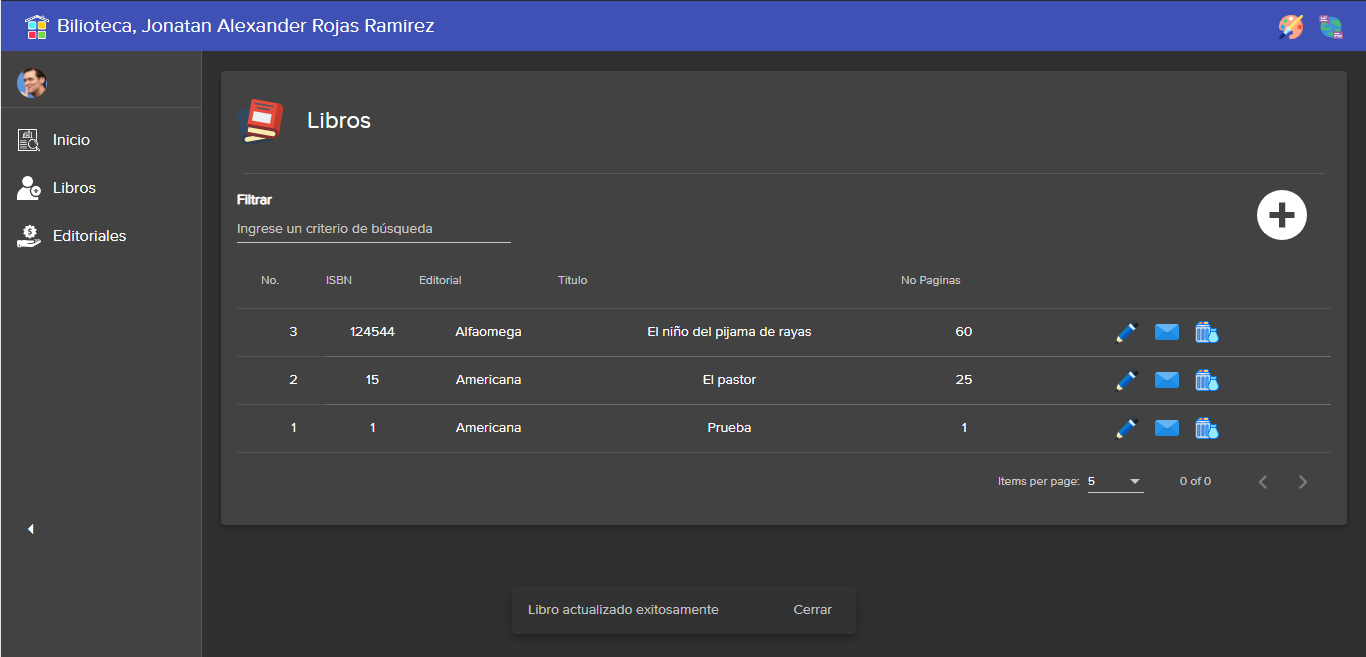
* Editar Libros

Opción para editar, aquí se implementa el concepto de padre e hijo de angular en el cual se envían datos por el decorador @Input y este a su vez me retorna al padre por el decorador @Ouput

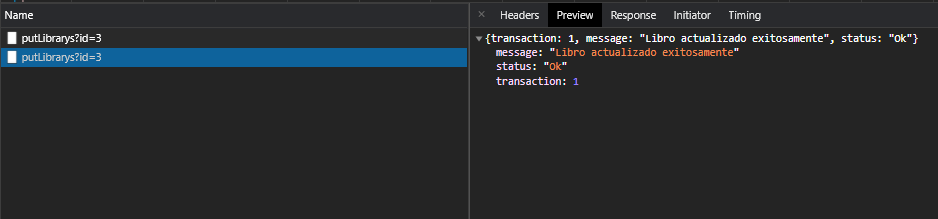
Como se puede ver el modal se implementa y se muestra una <mat-card>, ajustable a temas tambien



Validaciones de angular Forms, ReactiveForms, y selectFormModule para traer las editoriales creadas.



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



* Creación de registro

Modal de creación de registro con misma funcionalidad a modo de ejemplo

