Nombre: Francisco mesa Arellano

Asignatura; Programación Hibrida

Carrera: Técnico en informática

Actividad 1 – Prototipo

Instrucciones

En esta primera etapa, se le solicita desarrollar la estructura inicial del proyecto, dividiendo el problema en componentes más pequeños y diseñando las interfaces gráficas. La aplicación que va a construir permitirá visualizar y gestionar citas famosas dichas o escritas por distintos autores. A continuación, se detallan los puntos clave a considerar:

La primera página debe mostrar una cita aleatoria

Las citas constan de la frase y el autor que la dijo o escribió

En otra página, se debe poder gestionar las citas, permitiendo agregar y eliminar registros

En una tercera página, se debe poder configurar las opciones de la aplicación. Se debe, al menos, incluir la opción "¿Desea poder borrar citas en el inicio (frase aleatoria)?"

Observe las figuras que se muestran más abajo para la construcción de la solución

Divida el problema en:

Páginas (componentes de Angular con ruta)

Componentes

Servicios

Use el componente Router de Angular para la navegación entre páginas

Utilice @Input y @Output para la comunicación de los componentes

En esta etapa, es aceptable utilizar datos literales (hardcoded) para que los servicios provean información a los componentes. Esto facilitará la implementación inicial y se pueden incorporar fuentes de datos más dinámicas en etapas posteriores del desarrollo.

|  |  |  |
| --- | --- | --- |
| A white card with blue text  Description automatically generated  Figura 1 Sugerencia de diseño de página de inicio | A screenshot of a phone  Description automatically generated  Figura 2 Sugerencia de página de gestión de citas | A white background with black dots  Description automatically generated  Figura 3 Sugerencia de página de gestión de configuraciones |

Actividad 2 – Validaciones

Instrucciones

En esta segunda etapa, se le pide que desarrolle las validaciones para el formulario de ingreso de citas. A continuación, se detallan los puntos clave que se deben considerar:

Asegúrese de que tanto el campo para la frase como el campo del autor sean obligatorios.

Establezca una validación para que el campo de la frase tenga al menos 5 caracteres. El formulario debe indicar un error si la longitud de la frase es inferior a este requisito.

Implemente una validación que requiera que el campo del autor tenga un mínimo de 2 caracteres. El formulario debe señalar un error si la longitud del nombre del autor es menor a esta cantidad.

Actividad 3 – Persistencia

Instrucciones

En esta última etapa, se le encomienda la tarea de persistir los datos de la aplicación mediante el uso de los plugins de preferencias y SQLite. . A continuación, se detallan los puntos clave que se deben considerar:

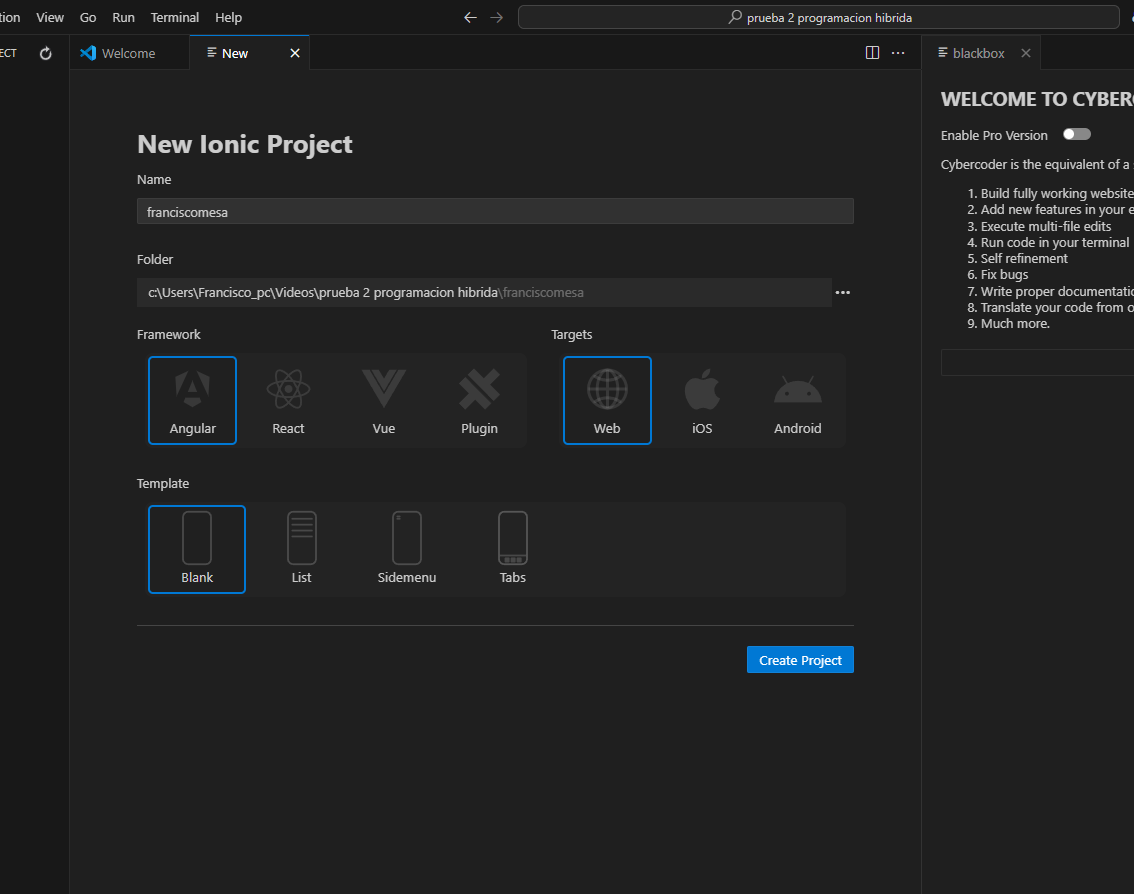
Asegure que los datos de la página de configuración sean persistentes mediante el uso del plugin Preferences. Garantice que la información configurada por el usuario, como la opción de "borrar citas en el inicio (frase aleatoria)", se mantenga después de cerrar y volver a abrir la aplicación.

Habilite la persistencia de los datos de las citas a través del plugin de SQLite. Asegure que las citas ingresadas por el usuario se almacenen de manera segura y estén disponibles incluso después de reiniciar la aplicación.

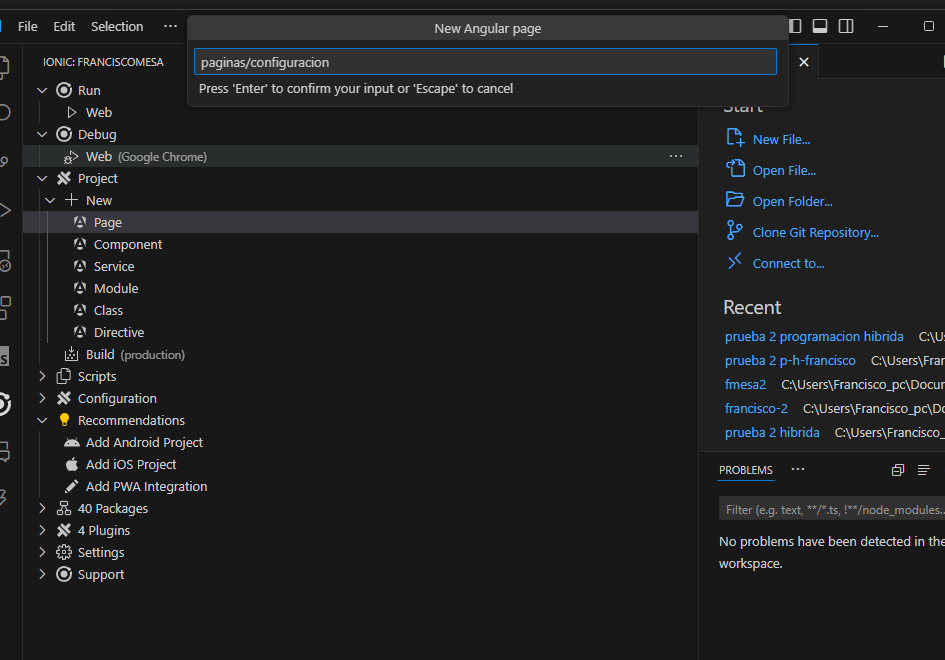
Aproveche el Ciclo de Vida de Angular o Ionic para la inicialización de los sistemas de almacenamiento y la carga de datos según sea necesario.

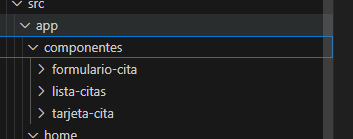
### ****Creación del Proyecto****

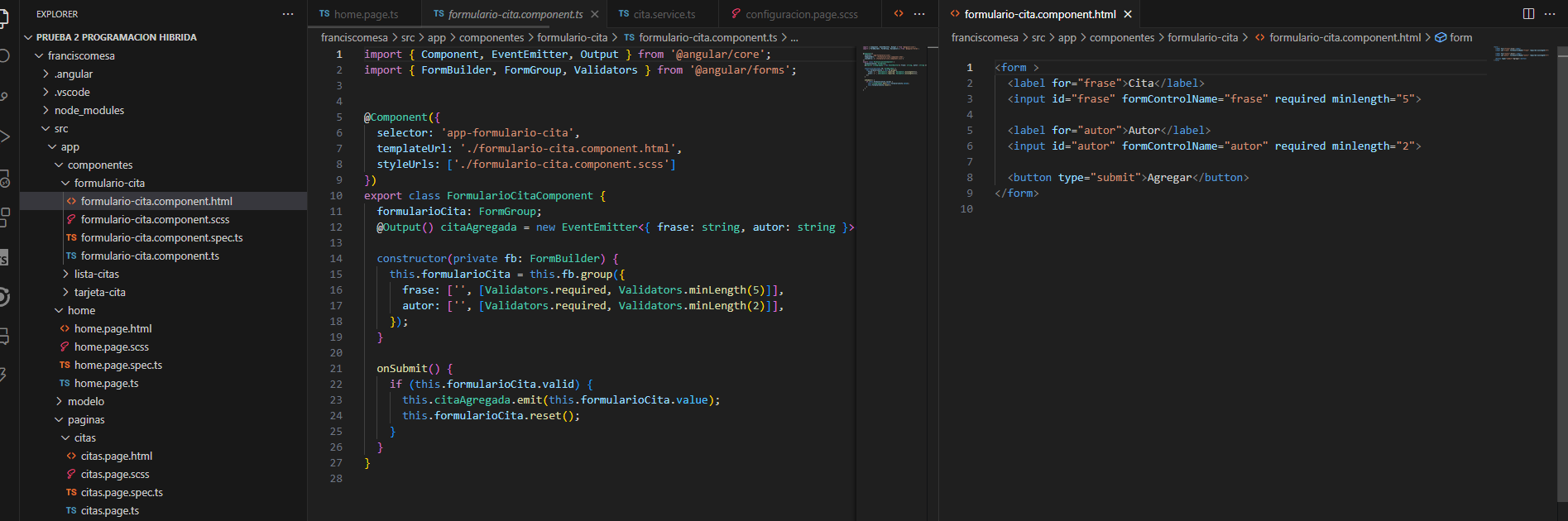
1. **Estructura Inicial:**
   * Creación de componentes como FormularioCita, ListaCitas y HomePage.
   * Clase modelo para las citas.
   * Páginas como CitasPage y ConfiguraciónPage.
2. **Servicios:**
   * Desarrollo del servicio CitaService para gestionar las citas.
3. **SQLite:**
   * Configuración del archivo sql-wasm.wasm en la carpeta assets.
   * Modificación de los archivos main.ts y app.component.ts para integrar SQLite.
4. **Validaciones:**
   * Implementación de validaciones antes mencionadas.



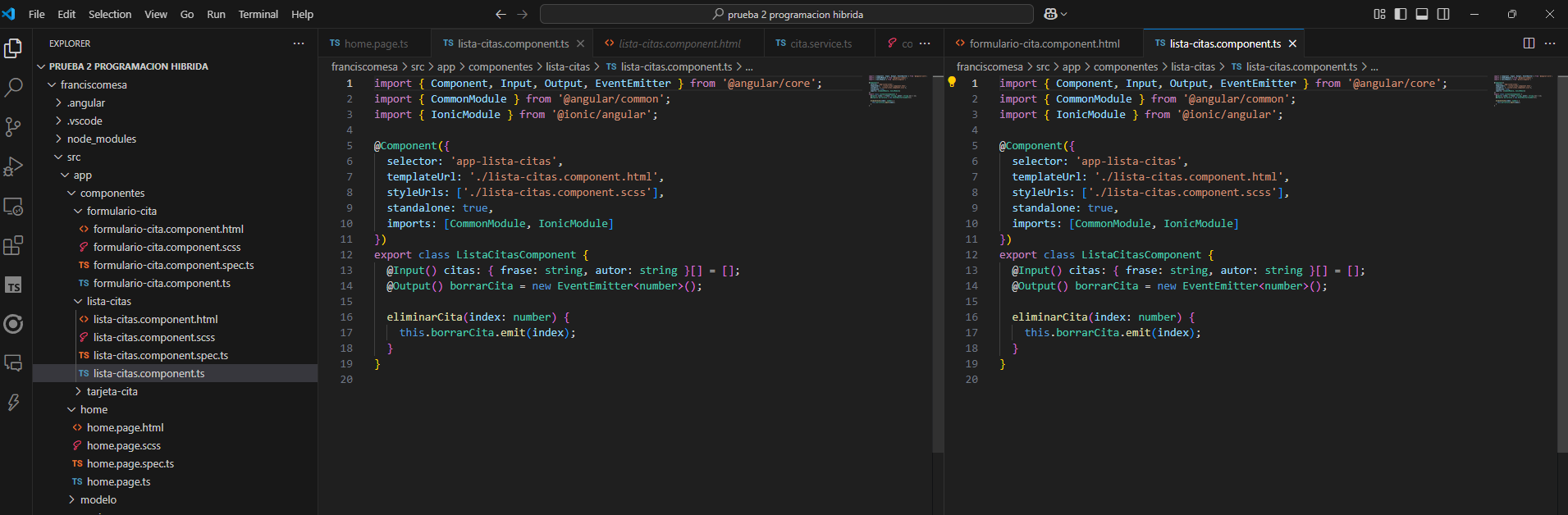
Creación de proyecto

agregando los app

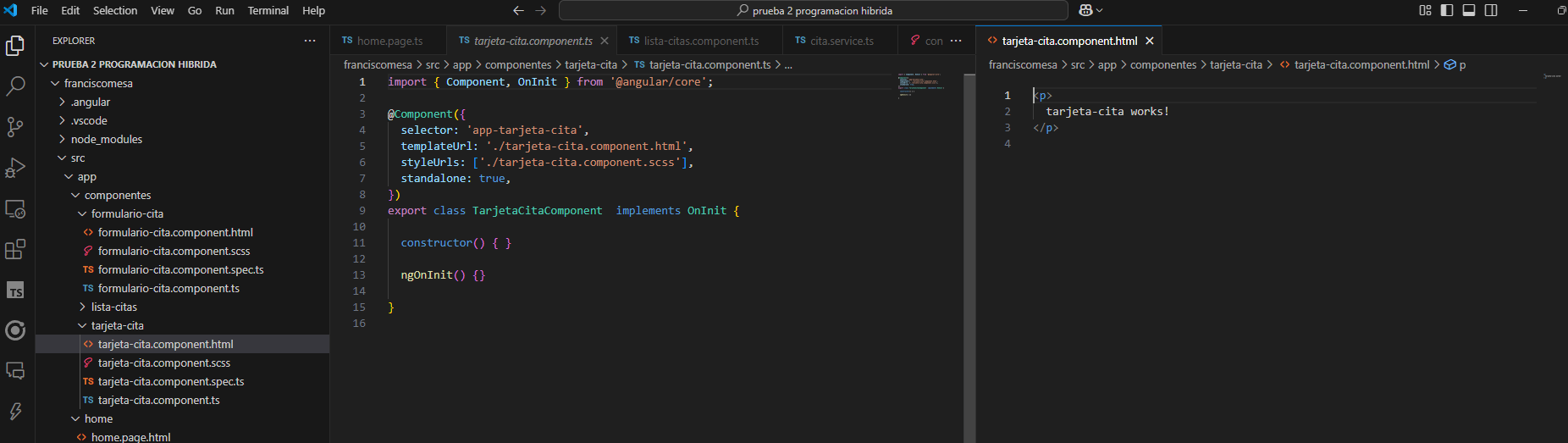
Agregado los componentes con los formularios mas sus códigos respectivos



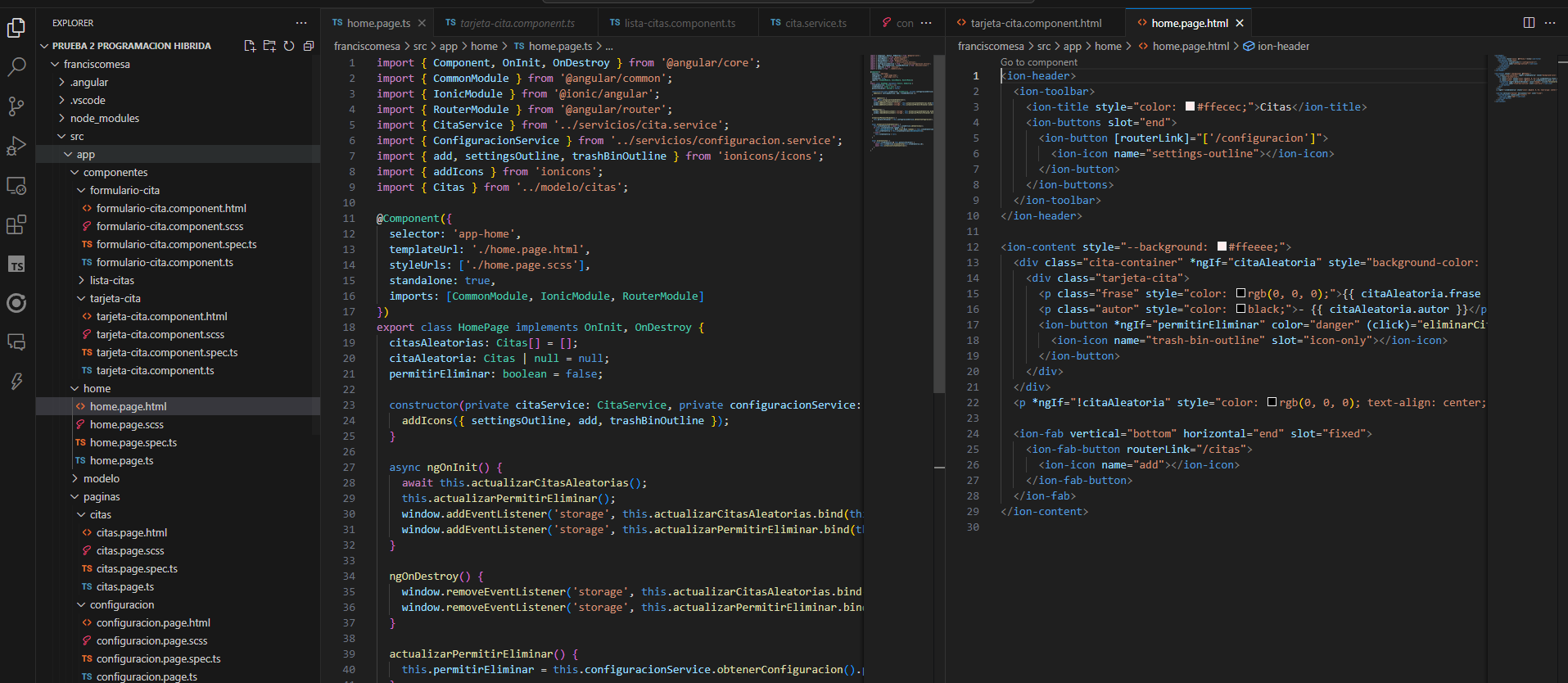
Formulario-cita



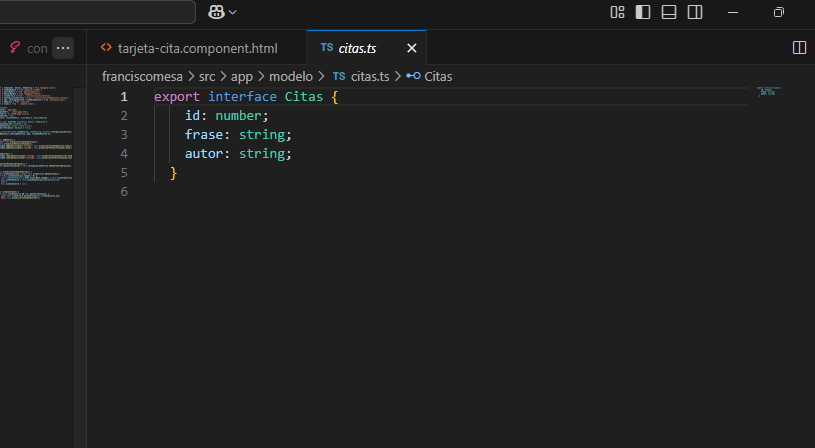
Lista-citas



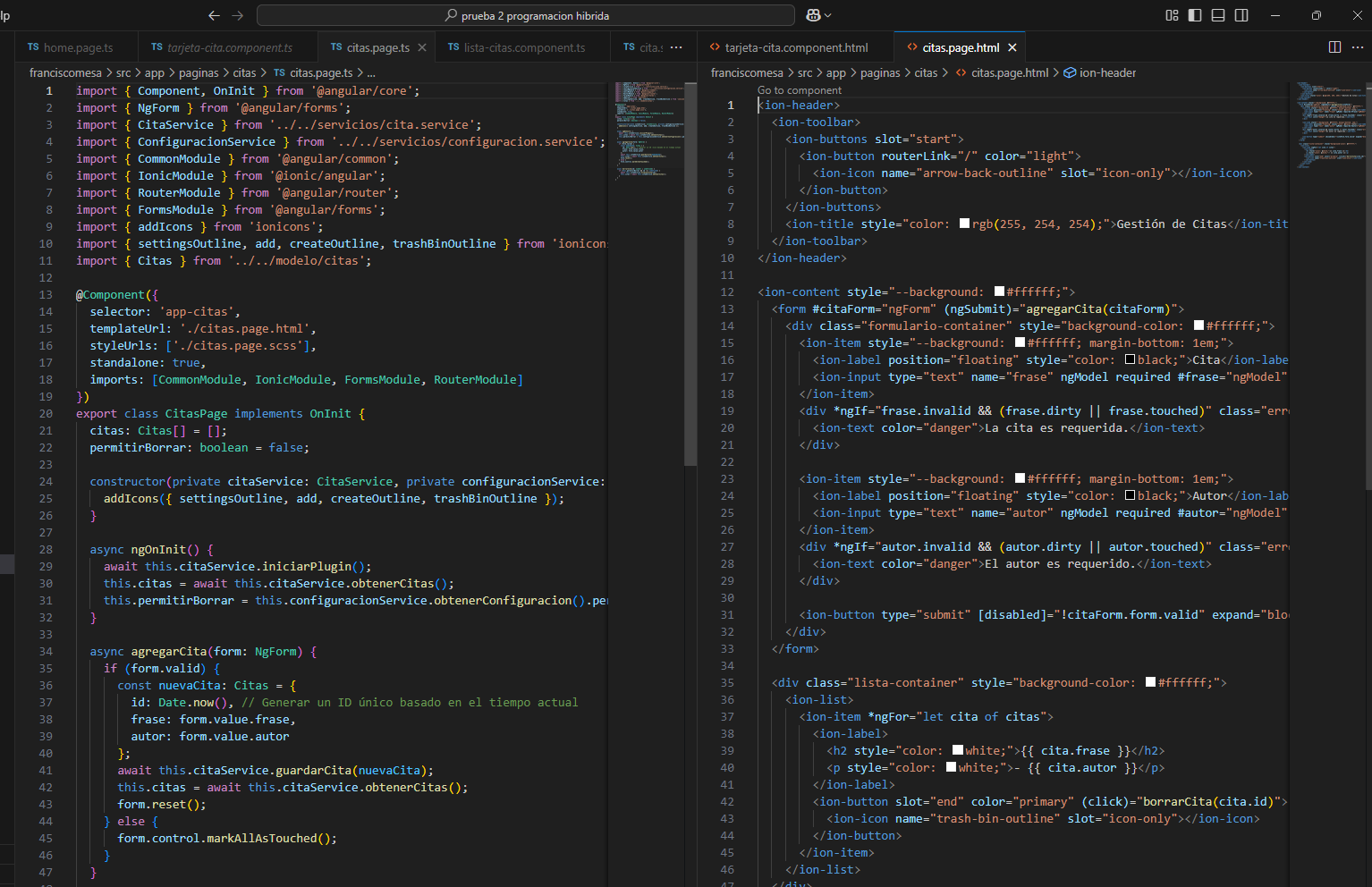
Formulario-cita



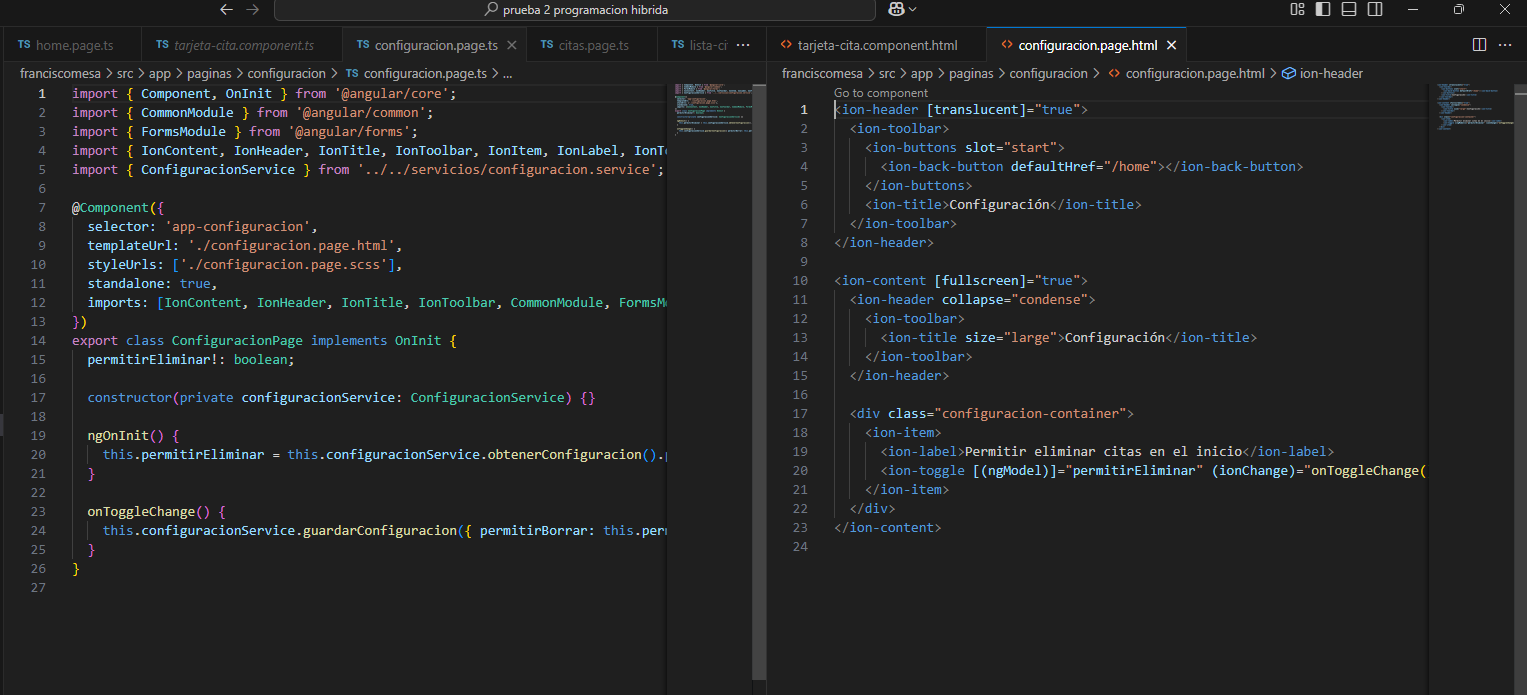
Home.page



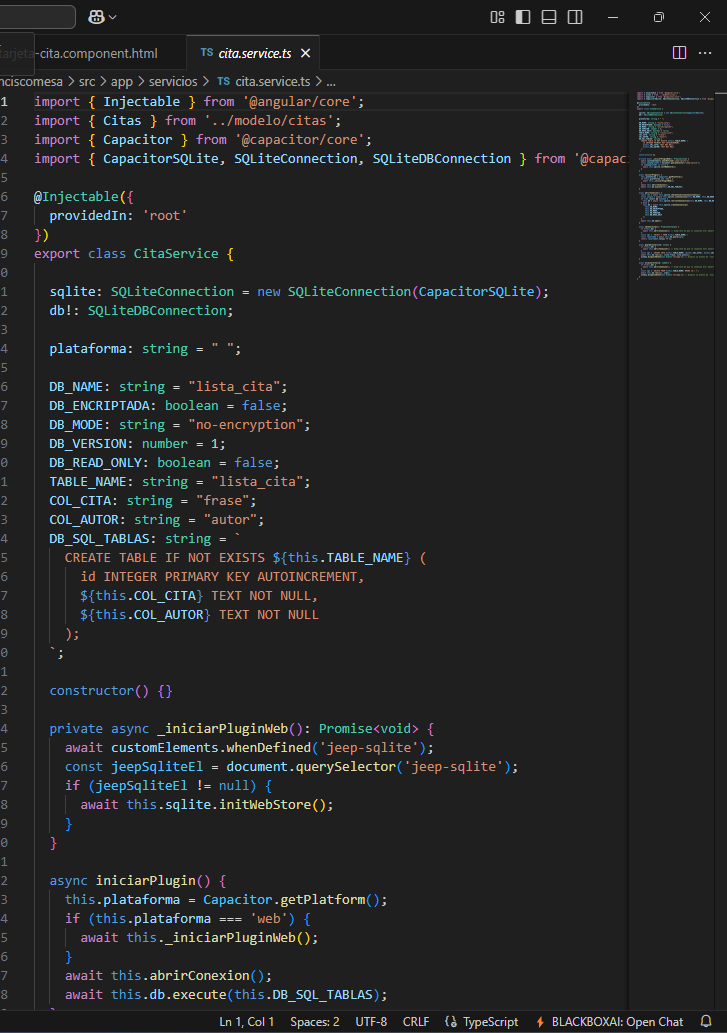
Una clase modelo



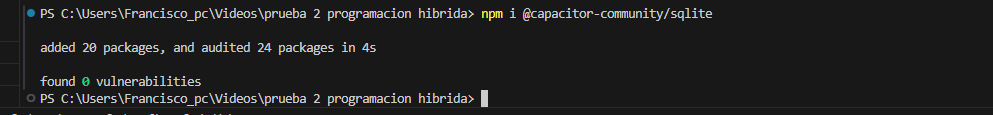
Paginas citas page



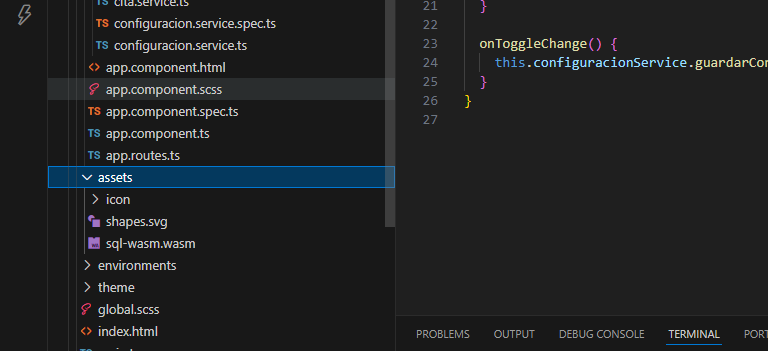
Configuración.page



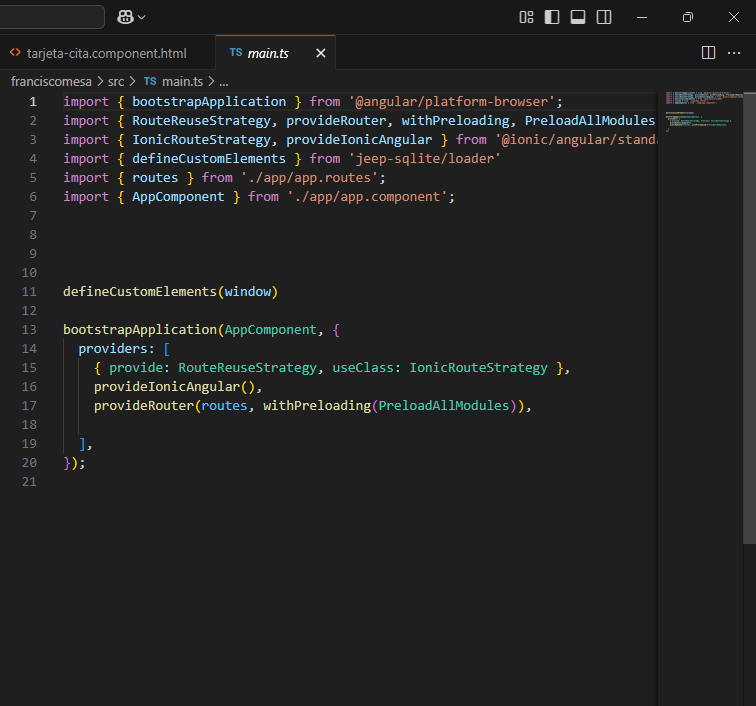
Se crean los servicios cita.cervice



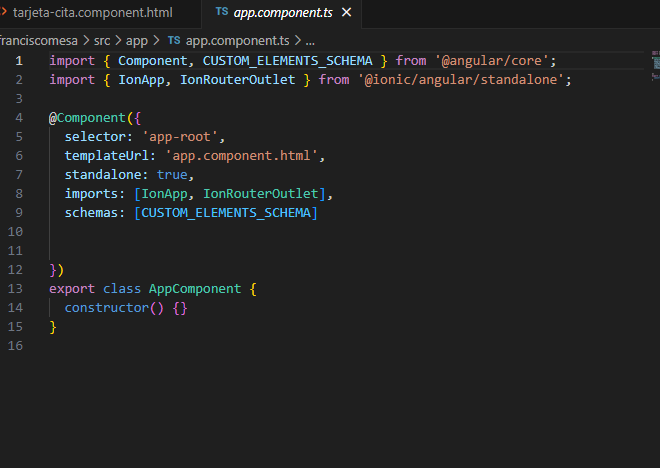
Instalación de dependencia



Copia de sql-wasm.wasm en assets

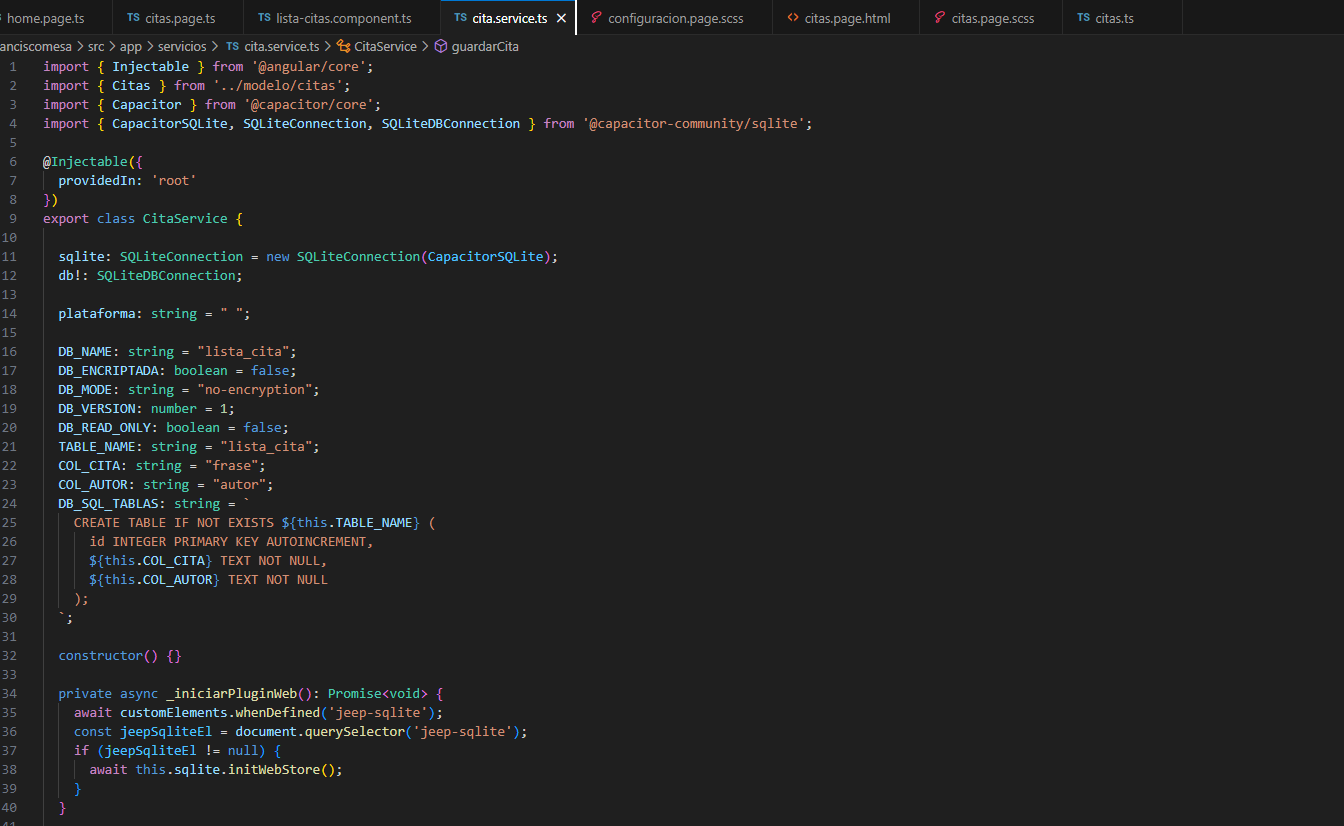


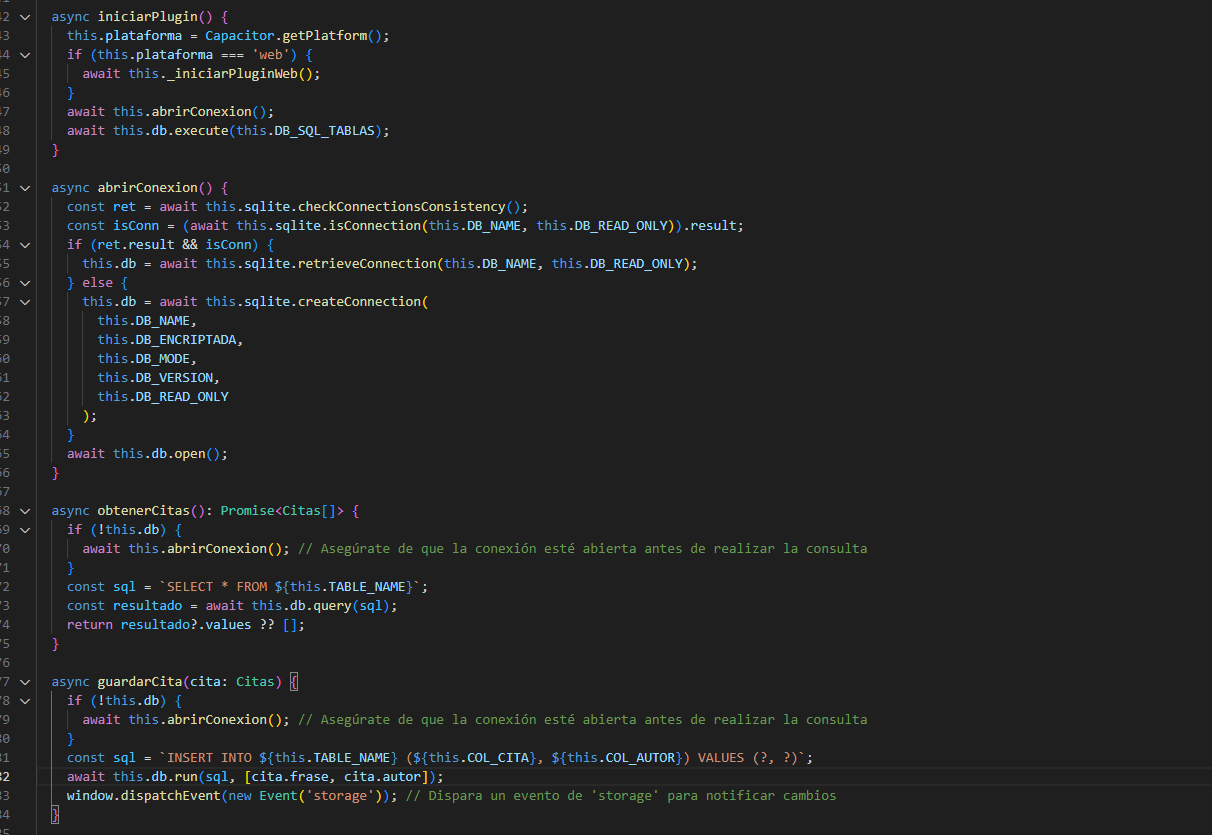
Se edita el archivo main.ts

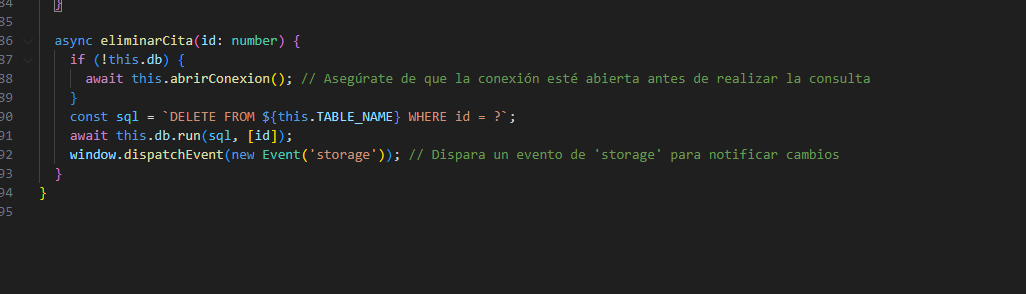


Se edita el archivo app.component.ts

Se añaden las dependencias al citas service para sqlite y la conexión en la página guarda datos al abrir y cerrar la pagina se añaden las validaciones correspondientes







1. Reflexión final (qué problemas tuve, cómo puedo mejorar, en qué necesito ayuda)

Algunos desafíos al integrar dependencias y conectar SQLite con el servicio.

Ayuda

Consultas sobre la implementación y depuración del almacenamiento con SQLite.

Pauta de evaluación

|  |  |  |
| --- | --- | --- |
| Criterios de Evaluación | Indicadores de Logro | Puntaje |
| Implementa una comunicación efectiva entre los diferentes componentes de la aplicación para reutilizar y dividir los requerimientos. | Divide y documenta los bloques de construcción del proyecto para solucionar el problema de negocio | 5 |
| Utiliza correctamente el decorador @Input para recibir información | 10 |
| Utiliza correctamente el decorador @Output para generar eventos | 10 |
| Crea y diseña adecuadamente los servicios que permitirán dar solución al problema de negocio | 5 |
| Utiliza las directivas estructurales "if" y "for" de Angular en la interfaz de usuario para construir interfaces avanzadas. | Utiliza correctamente la directiva \*ngFor para iterar sobre los elementos en las plantillas de los componentes de Angular | 5 |
| Diseña y valida formularios en la aplicación para asegurar la integridad de los datos ingresados por el usuario. | Valida correctamente los campos indicados | 10 |
| Impide realizar la acción y muestra un mensaje en la UI cuando uno o más campos son inválidos | 10 |
| Utiliza técnicas de persistencia de datos locales en el dispositivo para almacenar información relevante. | Instala y codifica el uso del plugin de Preferencias en un Service | 10 |
| Integra el Service con las configuraciones correctamente para persistir y recuperar datos | 5 |
| Instala y codifica correctamente el uso del plugin de SQLite en un Service | 10 |
| Integra correctamente el Service de citas en la página correspondiente para insertar y eliminar registros | 5 |
| Aplica el conocimiento del ciclo de vida de Ionic y Angular en una aplicación híbrida para gestionar con precisión el comportamiento de ésta. | Utiliza el ciclo de vida para inicializar el plugin de SQLite | 5 |
|  | Adjunta el código o proporciona un enlace al proyecto, siguiendo buenas prácticas como: mantener un código ordenado con una correcta indentación, la inclusión de comentarios explicativos, utilización de nombres significativos para variables y métodos, y se asegura de que la carpeta node\_modules no esté incluida. | 5 |
|  | Adjunta el informe en formato Word o PDF, asegurando mantener una ortografía impecable. El informe explica de manera detallada los fragmentos de código relevantes. Se asegura de que el código incorporado esté ordenado y sea fácilmente legible, con un resaltado de sintaxis adecuado. La estructura del informe debe ser coherente, abordando en detalle tanto los problemas que pudo solucionar como aquellos que no pudo resolver. | 5 |
| Puntaje Total | | 100 |