Nombre: Ninoska Burgos

programación Hibrida.

Docente: Santiago Leiva

## Instrucciones generales

Una vez realizada la lectura comprensiva del material de estudio, es fundamental la realización de este taller, el que tiene por objetivo medir la correcta internalización y aplicación de los conceptos abordados en las mencionadas unidades.

El desarrollo de este trabajo es individual, no se permitirán entregas de talleres en duplas o en grupos.

**Contenidos asociados al taller:**

1. Comunicación entre componentes
2. Utilización de los ciclos de vida de Ionic y Angular
3. Persistencia de datos local (preferencias y SQLite)
4. Diseñar y validar formularios

## Instrucciones específicas

El taller propuesto requiere articular contenidos abordados en la Unidad, a través de las siguientes acciones:



**Al final de este documento podrás encontrar la pauta de evaluación y los indicadores de logro asociados a cada actividad.**

# Entregables

1. ***Informe en Word o PDF*** *que explique con sus propias palabras los fragmentos más significativos del código de acuerdo con los criterios de evaluación. Más abajo en el documento se presenta la estructura sugerida.*
2. ***Proyecto comprimido en ZIP o enlace dentro del informe****. Puede usar plataformas como GitHub, Google Drive, OneDrive, u otras similares.* ***Es esencial eliminar la carpeta node\_modules antes de adjuntar el trabajo,*** su inclusión haría que el tamaño del proyecto sea considerablemente grande.

# Actividad 1 – Prototipo

### Instrucciones

1. **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

1. **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

1. **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.

### Estructura sugerida para el informe:

1. Estructura de la solución  
   Explique las piezas de su solución, puede ahondar en:  
   - Responsabilidades  
   - Relaciones  
   - Dependencias
2. Diseño de la interfaz gráfica
3. Validaciones
4. Persistencia de las configuraciones
5. Persistencia en SQLite
6. Reflexión final (qué problemas tuve, cómo puedo mejorar, en qué necesito ayuda)

**Información Importante**

Esta actividad será evaluada por el profesor de la asignatura, por lo que se sugiere, para su realización, considerar los Criterios de Evaluación detallados al final de este documento. El puntaje total de este taller es de 100 puntos, y el puntaje mínimo de aprobación es de 60 puntos.

# Formalidades

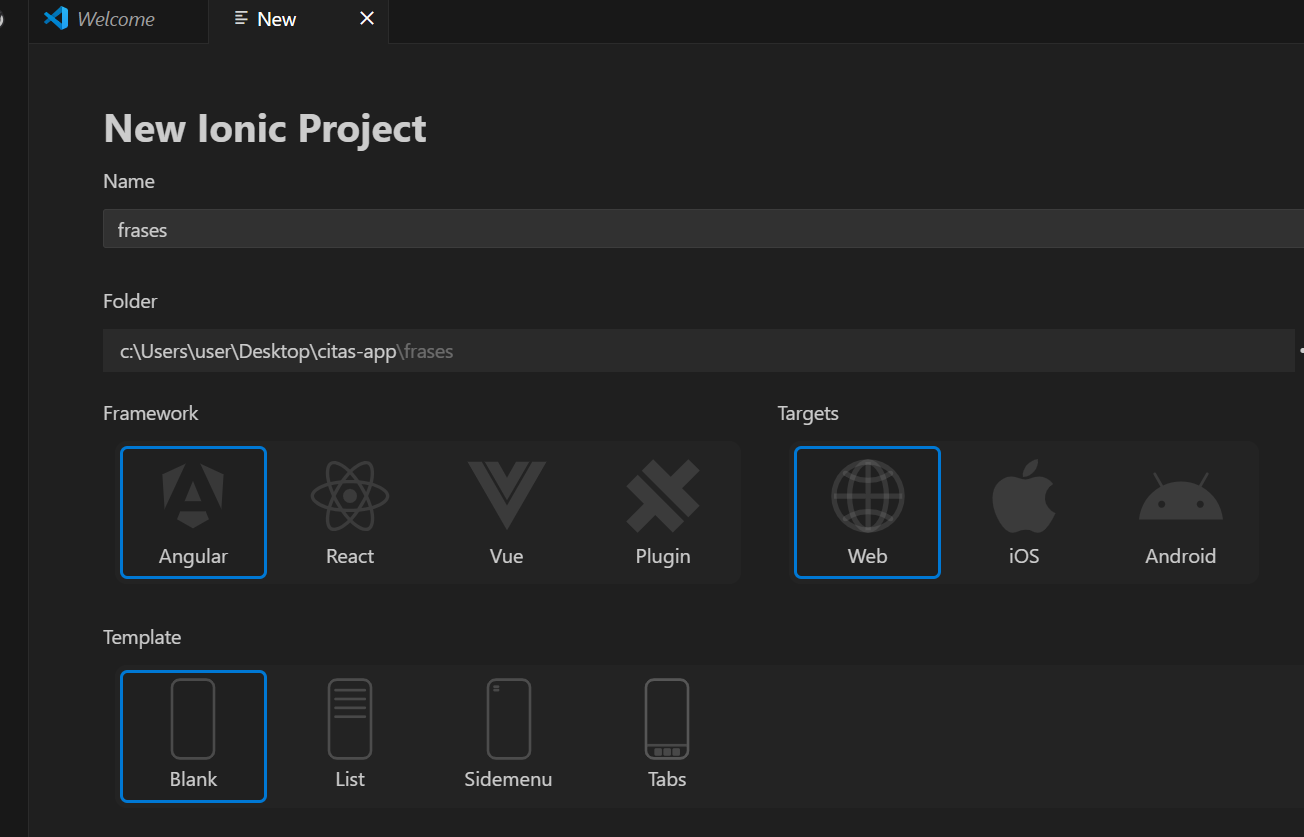
* Tipo de letra: Arial 12
* Interlineado: 1,15
* Formato - Entrega del taller en archivo Word o PDF, que lleve como nombre: SuNombre\_SuApellido.
* Subir archivo a la plataforma

# Pauta de evaluación

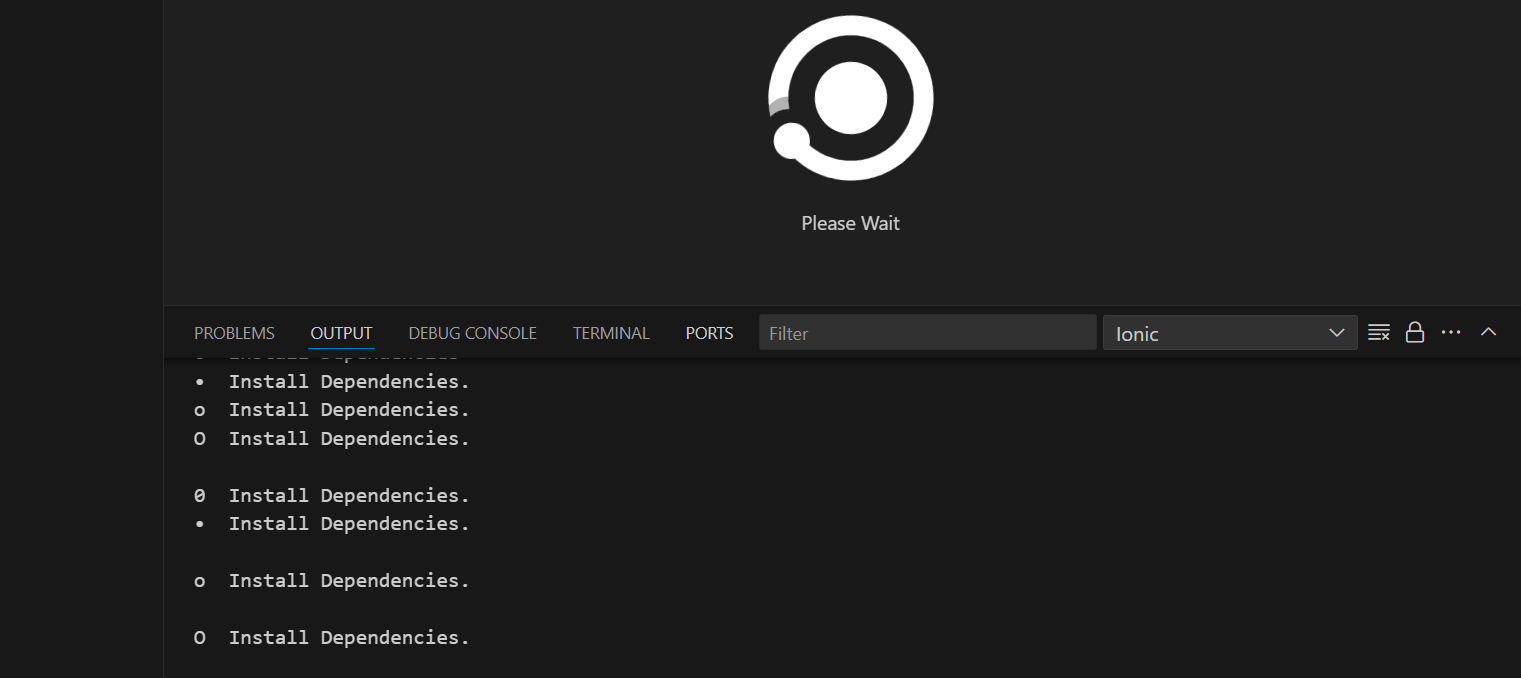
|  |  |  |
| --- | --- | --- |
| **Criterios de Evaluación** | **Indicadores de Logro** | **Puntaje** |
| Implementa una comunicación efectiva entre los diferentes componentes de la aplicación para reutilizar y dividir los requerimientos. | Divide y documenta los bloques de construcción del proyecto 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** |

Para comenzar a crear nuestro proyecto abriremos Visual Studio Code,

Luego crear nuevo proyecto seleccionamos el nombre después angular y blank:



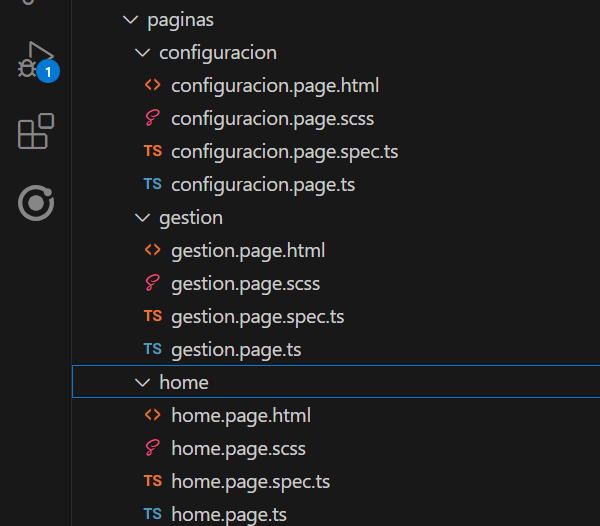
Esperamos que se seleccione las dependencias:



Luego de nuestros componentes instalados;

Crearemos las paginas en una carpeta llamada paginas como componentes serian:

Home, gestión y configuración:



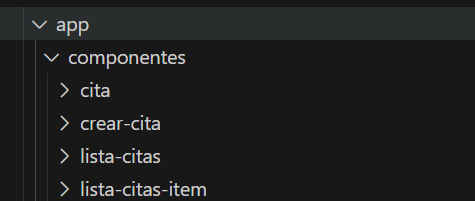
Luego los componentes

/cita

/crear-cita

/componentes/lista-citas

/componentes/lista-citas-item:



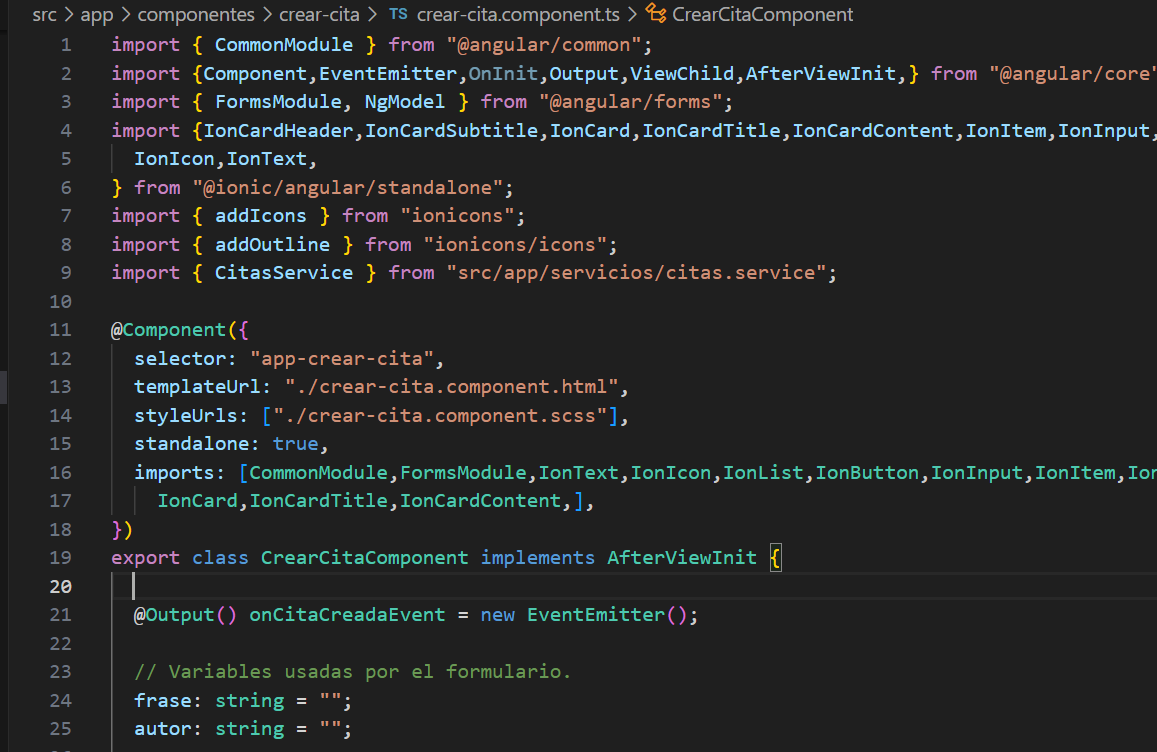
Y la configuración de las Rutas en app.routes.ts





Ahora haremos la lógica de componentes y las validaciones

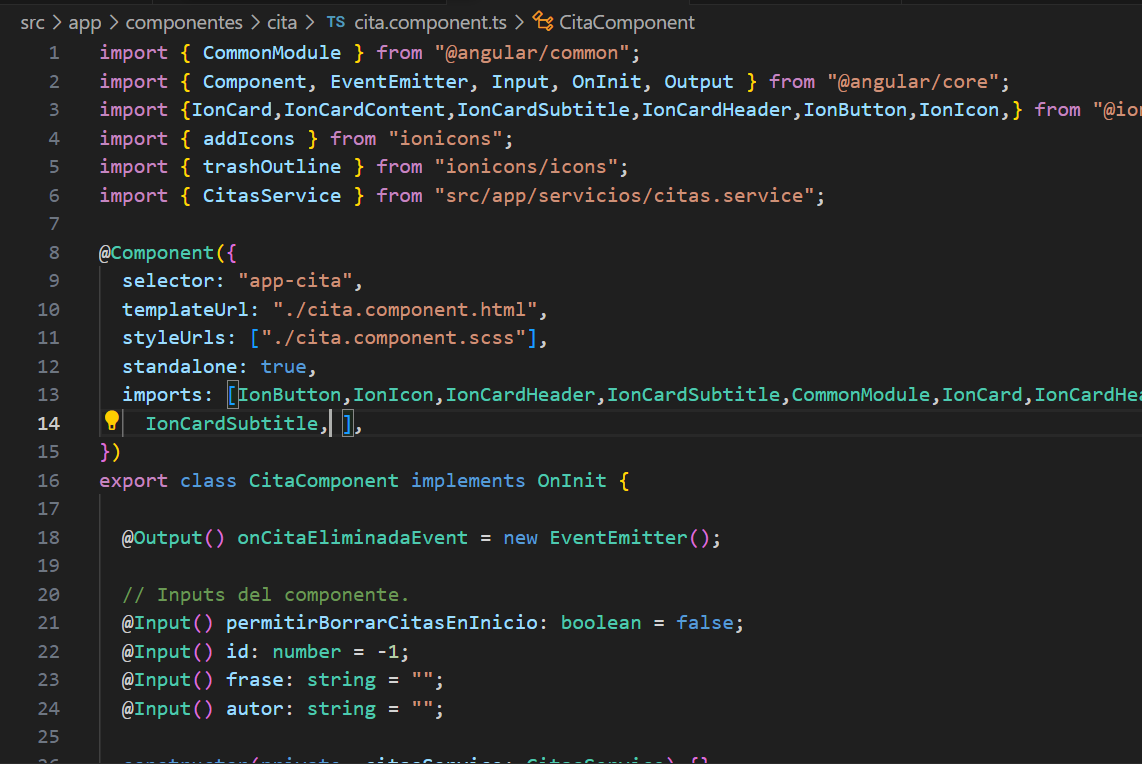
Componente crear cita:

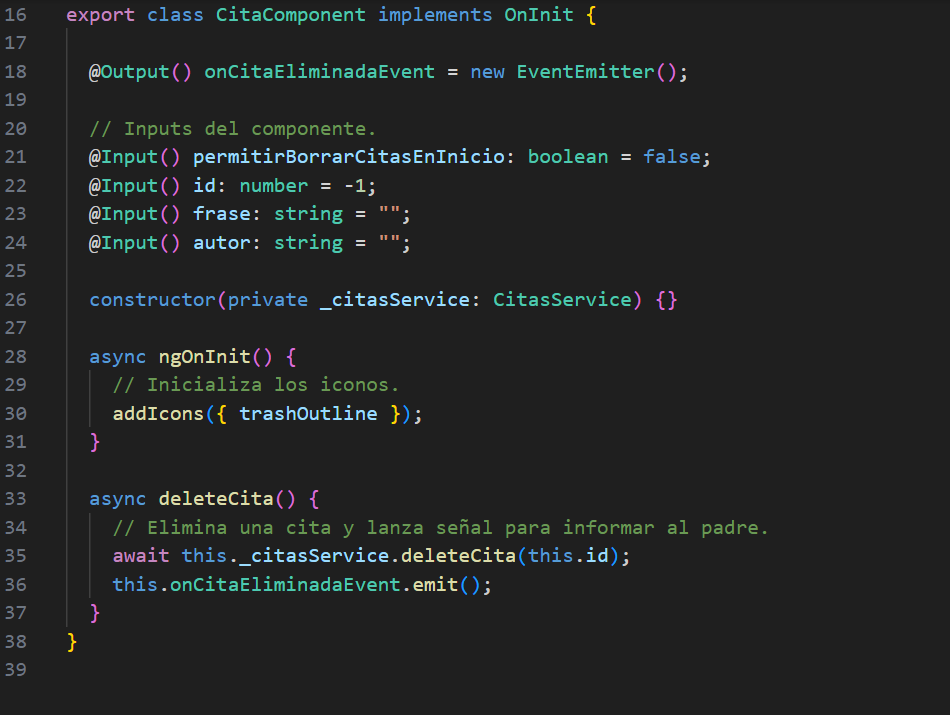


Contiene un formulario con dos campos (frase y autor) con validaciones mínimas.

En componente cita:

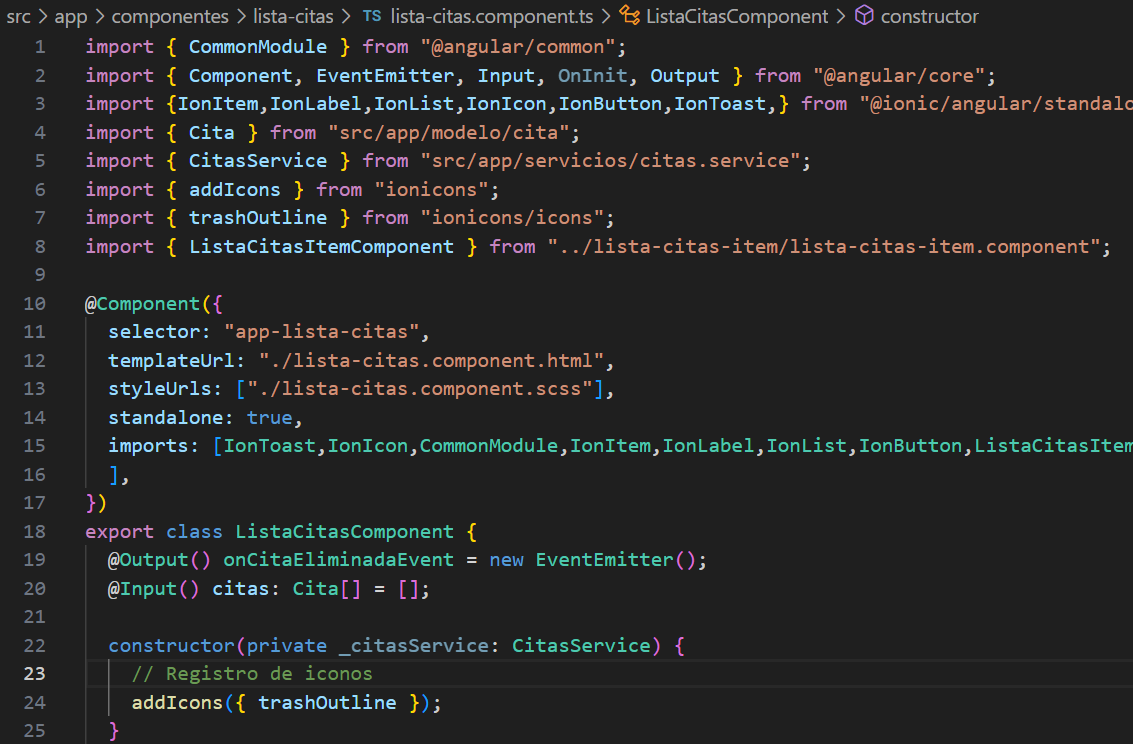
Muestra una cita individual y también colocaremos un botón de eliminar visible solo si está activado desde Configuración.





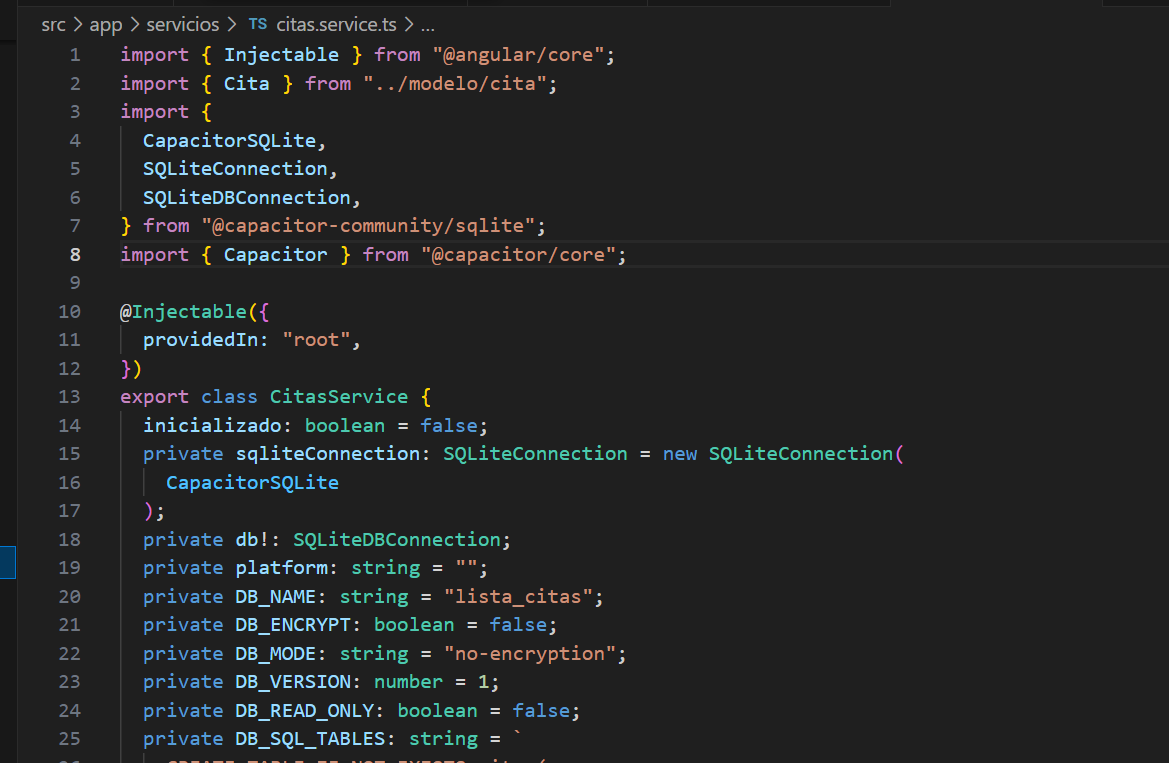
Y componente lista-citas:

Lista todas las citas usando el componente hijo lista-citas-item.



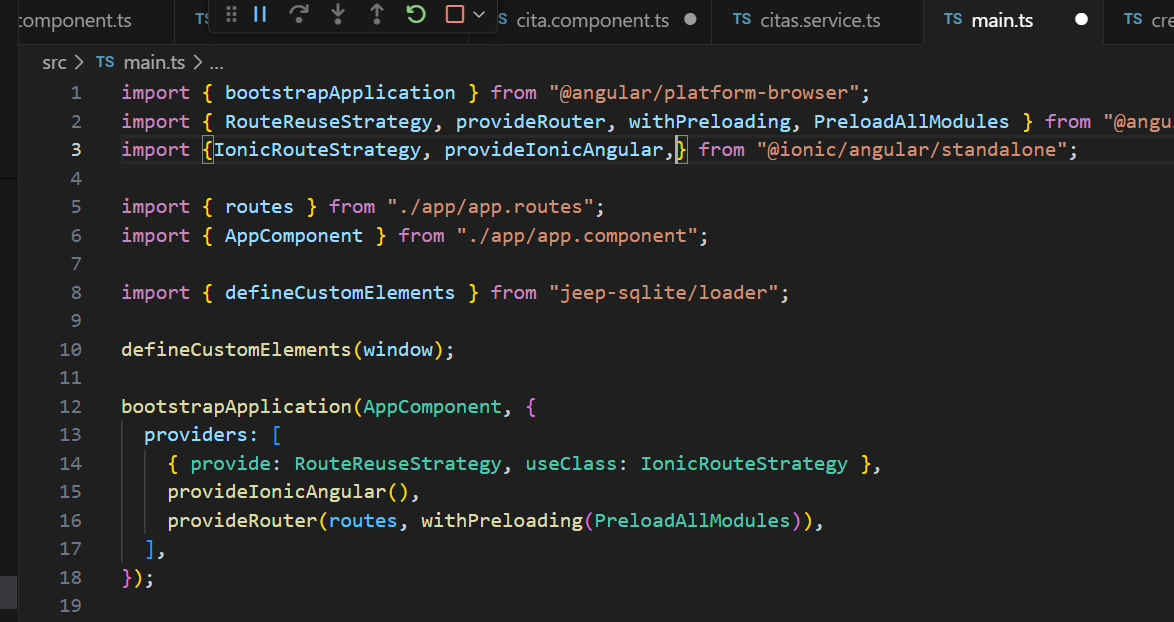
En citaService:

Manejaremos la conexión a la base de datos SQLite.



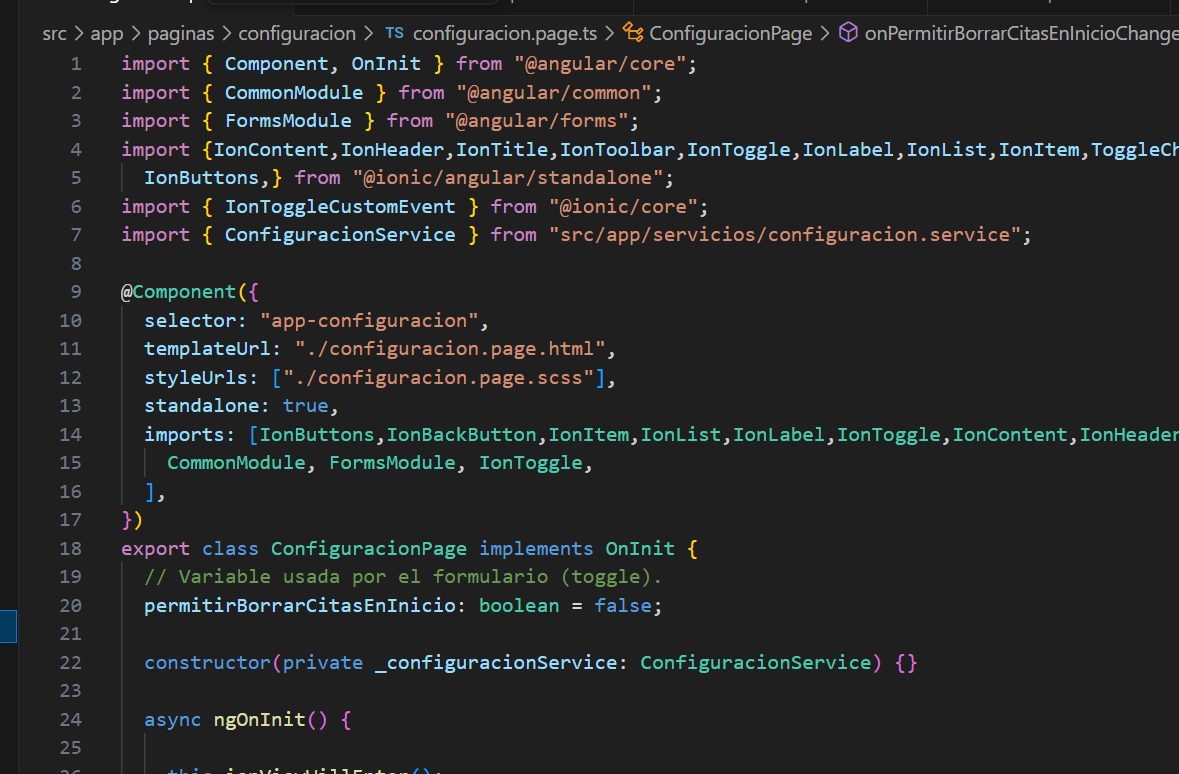
Inicialización de main.ts;

Se inicializa el componente de SQLite

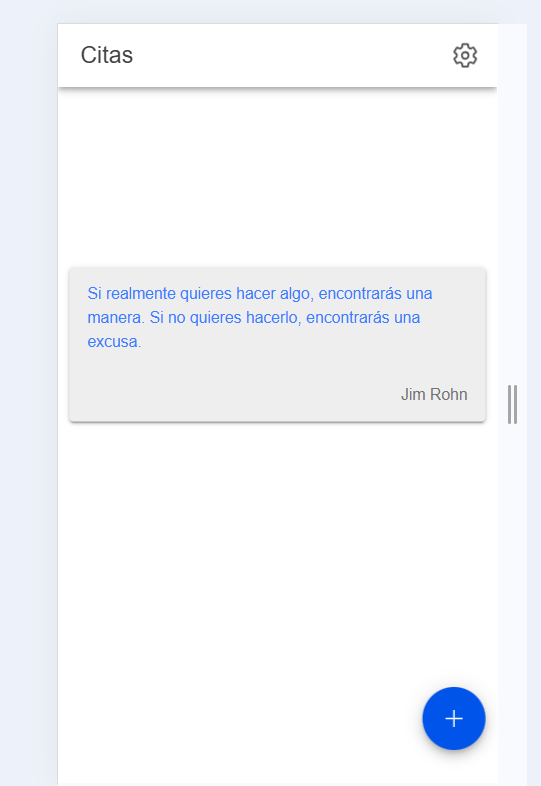


En la página configuración;

Toggle que permite activar/desactivar la eliminación de citas desde la vista principal.



En el flujo principal del usuario tendría que mostrar una cita aleatoria,



Y en la página de gestión se pueden agregar nuevas citas, también visualizar todas las citas y eliminar las citas.

