DESCRIPCIÓN TÉCNICA DETALLADA DE LA SOLUCIÓN

La solución para administrar Entidades y Empleados mediante los métodos CRUD y un sistema de autenticación y autorización consta de tres componentes principales: el backend, el frontend y la base de datos.

Backend: El backend de la solución está desarrollado en Node.js utilizando el framework Express.js. Se encarga de manejar las peticiones HTTP, procesar los datos y realizar las operaciones correspondientes en la base de datos. Además, implementa un sistema de autenticación y autorización basado en tokens JWT.

El backend consta de los siguientes archivos y carpetas:

- **app.js**: archivo principal del backend que configura el servidor Express.js, define las rutas y middleware necesarios para la aplicación, y establece la conexión con la base de datos MongoDB.
- **routes/**: carpeta que contiene los archivos que definen las rutas y controladores de la aplicación para cada entidad.
- **models/**: carpeta que contiene los archivos que definen los modelos de datos de la aplicación.
- **middleware/**: carpeta que contiene los archivos con los middleware personalizados que se utilizan en la aplicación, como el middleware de autenticación y autorización.
- **config/**: carpeta que contiene los archivos con la configuración de la aplicación, como la conexión a la base de datos y la generación de tokens JWT.

El backend utiliza los siguientes paquetes de Node.js:

- **Express.js**: framework para crear aplicaciones web en Node.js.
- **Mongoose**: librería para interactuar con bases de datos MongoDB desde Node.js.
- **jsonwebtoken**: paquete para la generación y verificación de tokens JWT.
- **bcrypt**: paquete para el hash y la comparación de contraseñas.

Frontend: El frontend de la solución está desarrollado en React.js y es responsable de la interfaz gráfica de usuario. Utiliza Bootstrap como framework de diseño y axios como librería para realizar peticiones HTTP al backend.

El frontend consta de los siguientes archivos y carpetas:

- **public/**: carpeta que contiene los archivos públicos de la aplicación, como el archivo HTML principal y las imágenes.
- **src/**: carpeta que contiene los archivos fuente de la aplicación React.js.
- **src/components/**: carpeta que contiene los componentes React.js de la aplicación, como el componente de la lista de entidades y el componente de formulario.
- **src/services/**: carpeta que contiene los archivos con las funciones que se encargan de realizar peticiones HTTP al backend.

El frontend utiliza los siguientes paquetes de Node.js:

- **React.js**: biblioteca para construir interfaces de usuario.
- **Bootstrap**: framework de diseño para construir interfaces de usuario responsivas.
- **axios**: librería para realizar peticiones HTTP desde el cliente.

Base de datos: La solución utiliza una base de datos MongoDB para almacenar la información de las entidades y los empleados. Se utiliza el ORM Mongoose para interactuar con la base de datos desde el backend.

La base de datos consta de las siguientes colecciones:

- **entities**: colección que contiene la información de las entidades.
- **employees**: colección que contiene la información de los empleados.
- **users**: colección que contiene la información de los usuarios.

Cada entidad tiene un conjunto de campos que incluyen un nombre, una descripción y una lista de empleados asociados. Cada empleado tiene un conjunto de campos que incluyen un nombre, y un cargo.

En resumen, la solución para administrar Entidades y Empleados mediante los métodos CRUD y un sistema de autenticación y autorización

CRITERIOS DE IMPLEMENTACIÓN DEL DISEÑO

Basándonos en la solución propuesta de la administración de Entidades y Empleados mediante los métodos CRUD y un sistema de autenticación y autorización, se pueden considerar las siguientes páginas a nivel de diseño para el aplicativo:

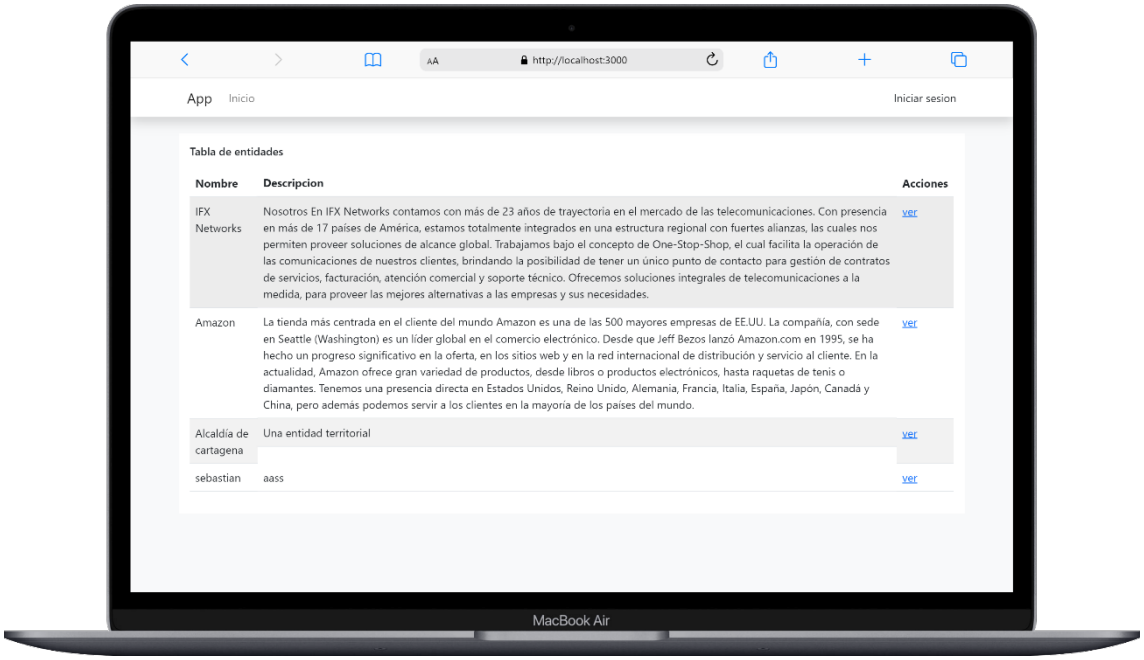
1. Página de inicio de sesión: Esta página será la primera en aparecer y permitirá al usuario ingresar sus credenciales de inicio de sesión (correo electrónico y contraseña) para acceder al sistema. También se incluirán opciones para recuperar la contraseña o registrarse si el usuario aún no tiene una cuenta.
2. Página de registro: Esta página permitirá al usuario registrarse en el sistema ingresando sus datos personales, incluyendo correo electrónico, contraseña, nombre completo y cargo dentro de la empresa.
3. Página de inicio: Después de iniciar sesión, el usuario será redirigido a la página de inicio donde se mostrará una lista de entidades registradas. Cada entidad se mostrará con su nombre y una breve descripción.
4. Página de detalle de entidad: Al hacer clic en una entidad, el usuario será redirigido a una página que mostrará más detalles sobre la entidad seleccionada, incluyendo su nombre, descripción y la lista de empleados asociados.
5. Página de creación de entidad: Esta página permitirá al usuario crear una nueva entidad proporcionando información como su nombre y descripción.

6. Página de edición de entidad: Si el usuario tiene permisos de administrador, se permitirá editar la información de una entidad, incluyendo su nombre y descripción.
7. Página de eliminación de entidad: Si el usuario tiene permisos de administrador, se permitirá eliminar una entidad seleccionada.
8. Página de detalle de empleado: Al hacer clic en un empleado dentro de una entidad, el usuario será redirigido a una página que mostrará más detalles sobre el empleado seleccionado, incluyendo su nombre, apellido, dirección y número de teléfono.
9. Página de creación de empleado: Esta página permitirá al usuario crear un nuevo empleado proporcionando información como su nombre, apellido, dirección y número de teléfono.
10. Página de edición de empleado: Si el usuario tiene permisos de administrador, se permitirá editar la información de un empleado, incluyendo su nombre, apellido, dirección y número de teléfono.
11. Página de eliminación de empleado: Si el usuario tiene permisos de administrador, se permitirá eliminar un empleado seleccionado.

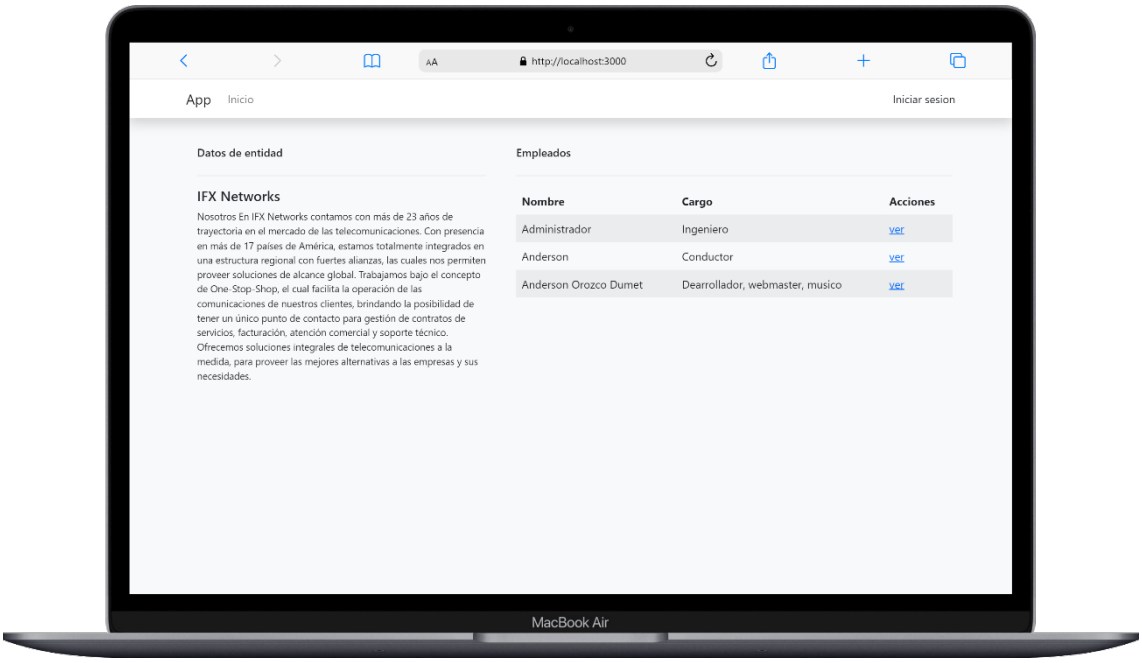
En general, las páginas estarán diseñadas para ser intuitivas y fáciles de usar para que los usuarios puedan administrar las entidades y los empleados de manera eficiente y efectiva.

DISENO DEL APLICATIVO VISTAS Y VENTANAS

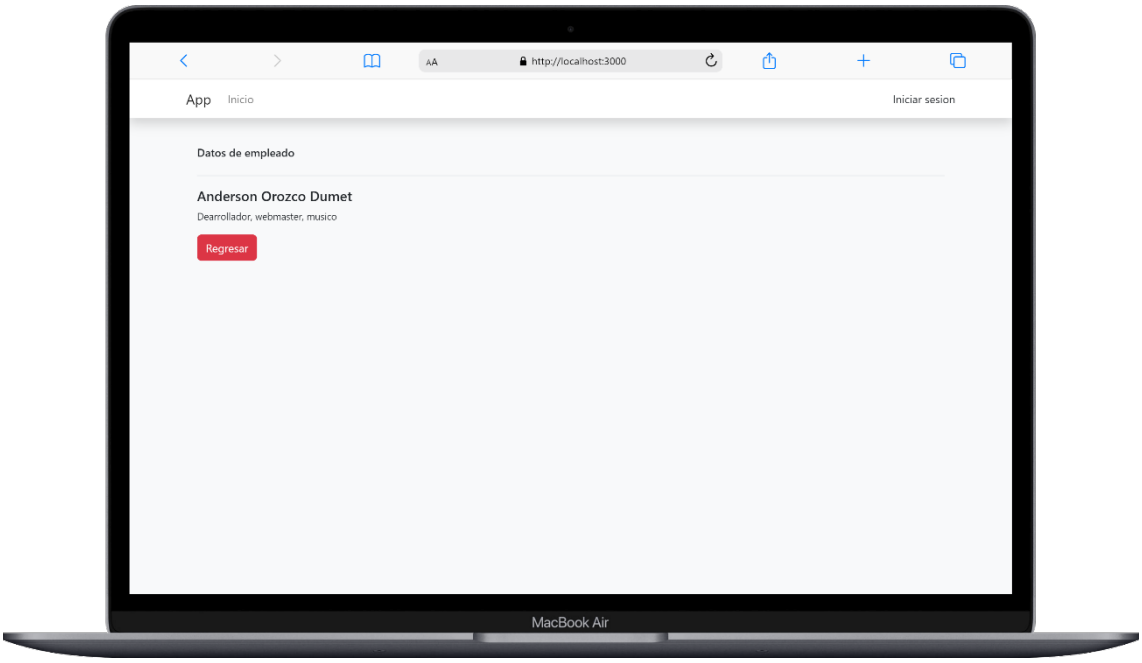
- **Home – barra de navegación, tabla de entidades**



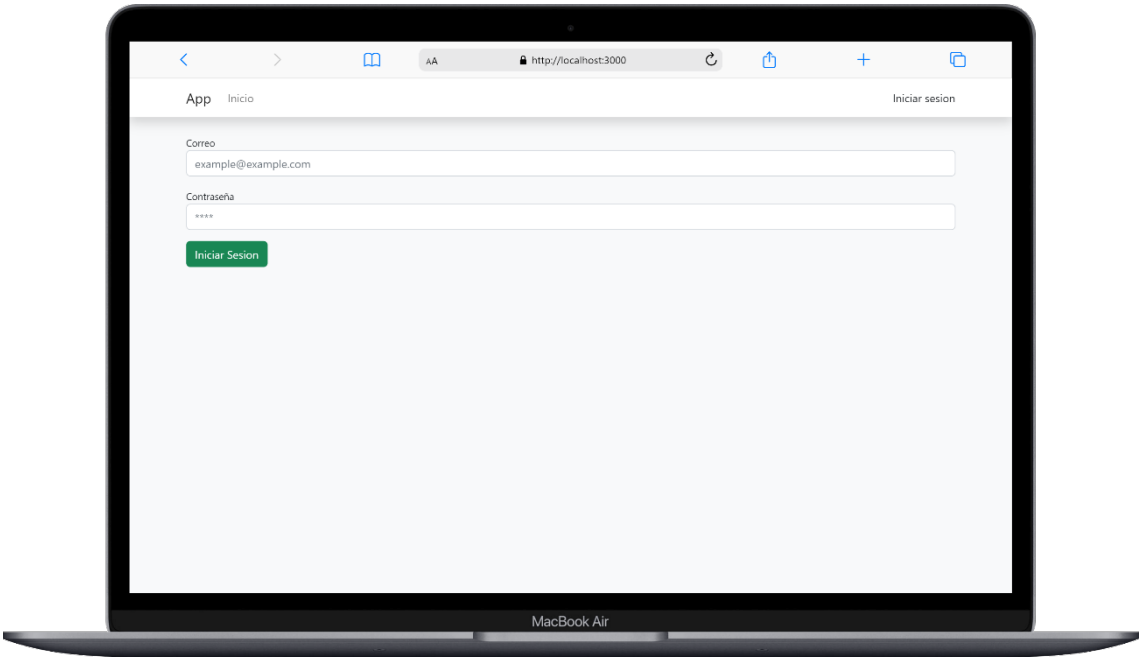
- **Datos de la entidad**



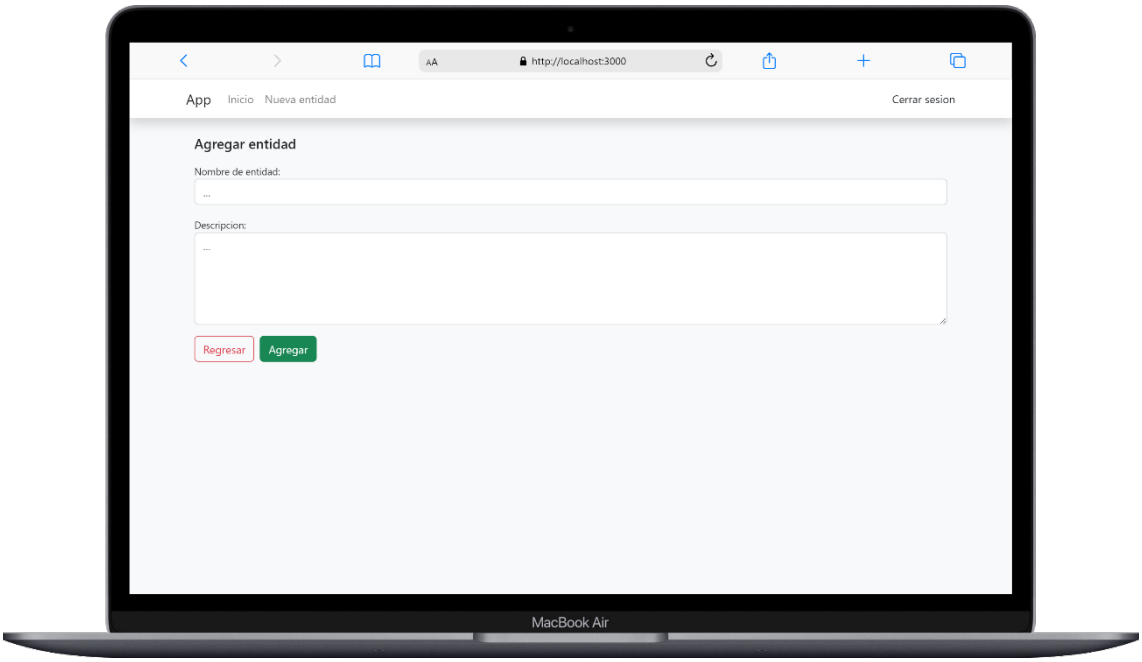
- **Datos del empleado**



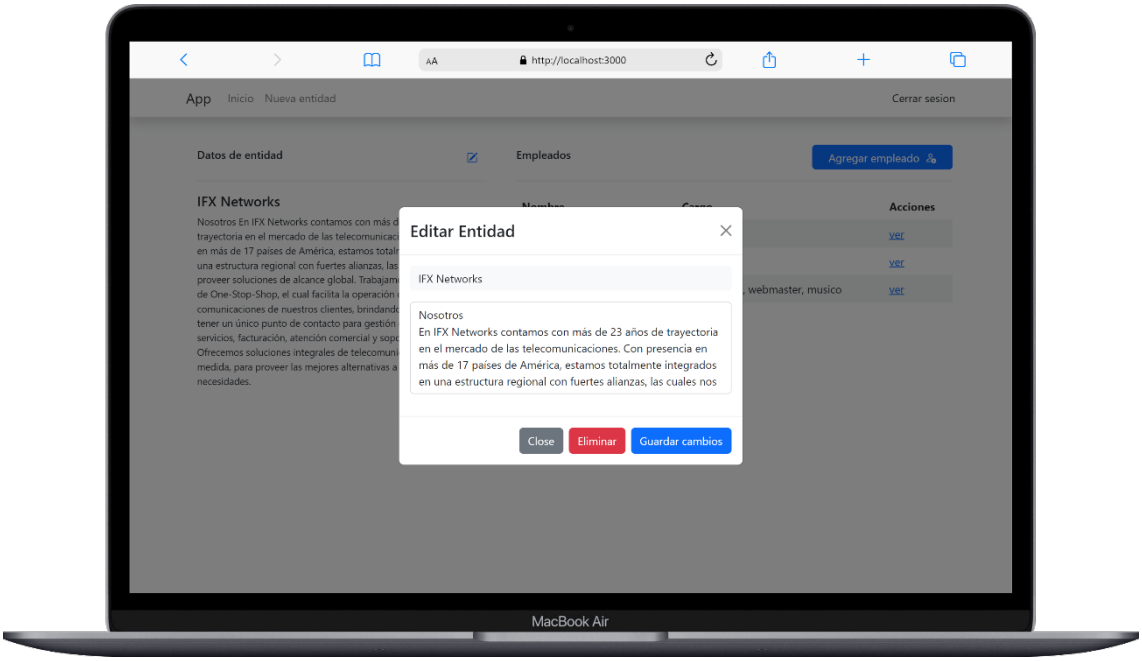
- **Inicio de sesión**



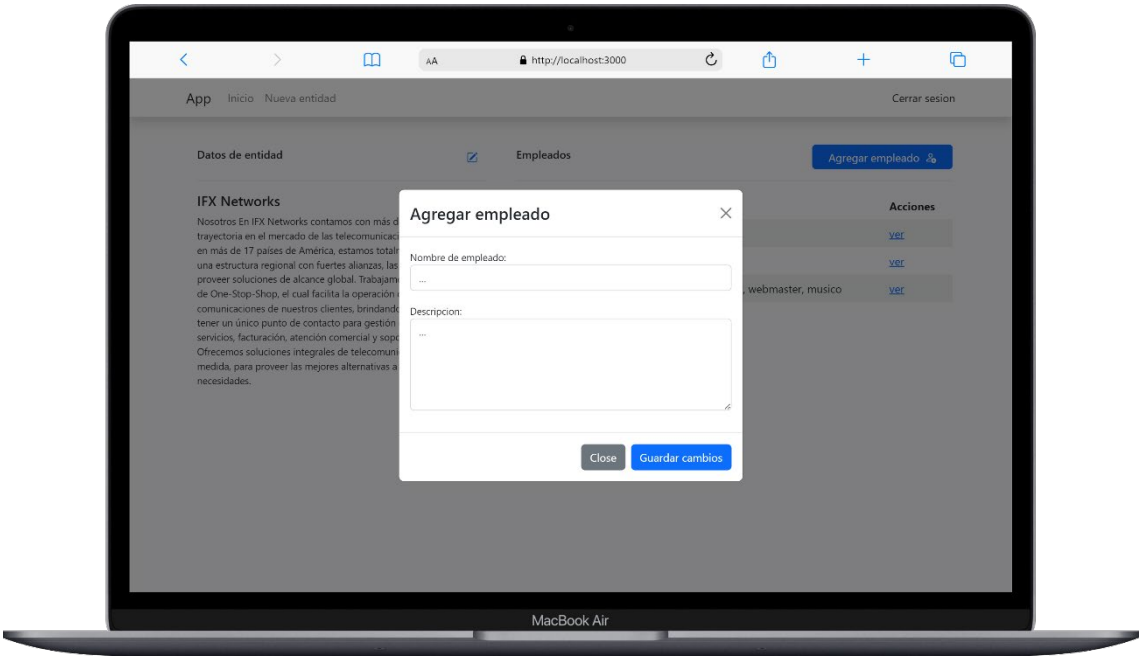
- **Vista de administrador**



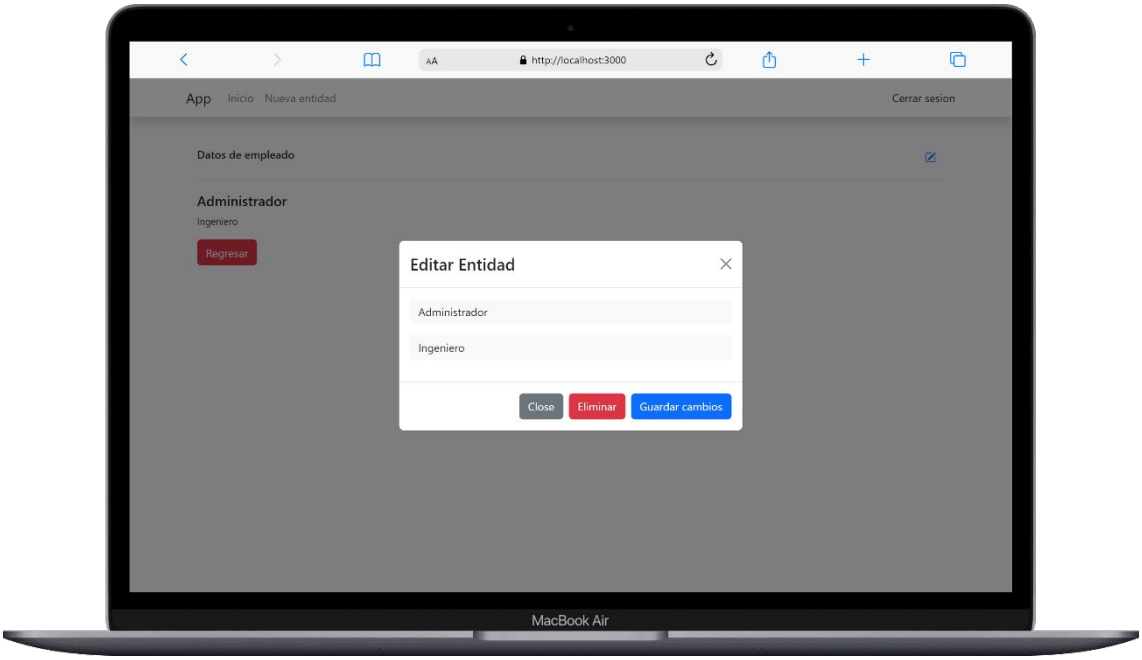
- **Vista edición de la entidad**



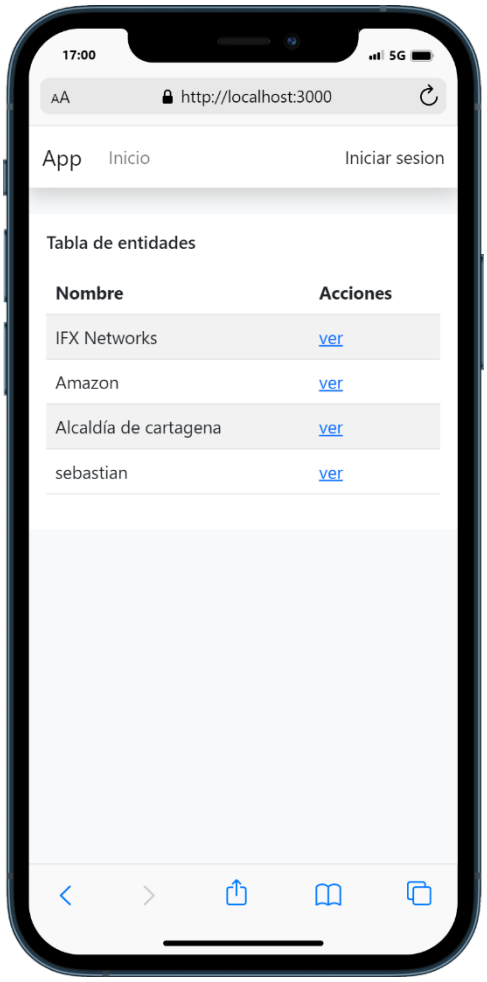
- **Vista para agregar empleados**



- **Vista de editar empleado**

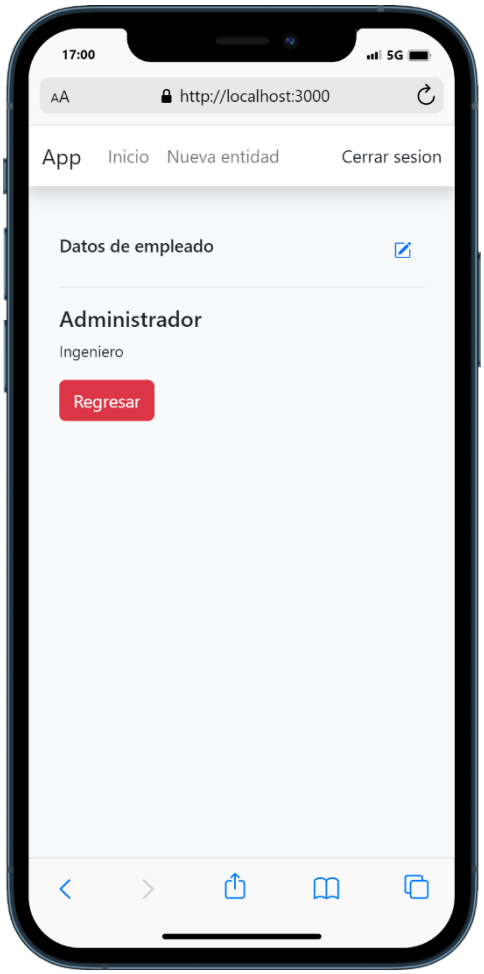


Vista responsiva




17:00
5G
AA
http://localhost:3000
App
Inicio
Nueva entidad
Cerrar sesion
Datos de entidad
Editar Entidad
IFX Networks
Nosotros
En IFX Networks contamos con más de 23 años de trayectoria en el mercado de las telecomunicaciones.
Close
Eliminar
Guardar cambios
Orozco Dumet
webmaster,
musico


17:00
5G
AA
http://localhost:3000
App
Inicio
Nueva entidad
Cerrar sesion
Datos de entidad
Agregar empleado
Nombre de empleado:
...
Descripcion:
...
Close
Guardar cambios
musico


17:00
5G
AA
http://localhost:3000
App
Inicio
Nueva entidad
Cerrar sesion
Datos de empleado
Administrador
Ingeniero
Regresar


17:00
5G
AA
http://localhost:3000
App
Inicio
Nueva entidad
Cerrar sesion
Datos de empleado
Administrador
Editar Entidad
Administrador
Ingeniero
Close
Eliminar
Guardar cambios

DOCUMENTACIÓN DEL CÓDIGO FUENTE Y LAS PRUEBAS UNITARIAS

Para la documentación del código fuente y las pruebas unitarias, se pueden utilizar herramientas como JSDoc y Mocha/Chai, respectivamente.

JSDoc es una herramienta que permite generar documentación automáticamente a partir del código fuente. Permite documentar funciones, variables y clases con un formato legible y fácil de seguir. Para utilizar JSDoc, se deben agregar comentarios especiales al código fuente y luego ejecutar la herramienta para generar la documentación en formato HTML.

Por otro lado, Mocha/Chai son herramientas para realizar pruebas unitarias en JavaScript. Mocha es un marco de pruebas que proporciona una estructura para escribir y ejecutar pruebas unitarias. Chai es una librería de aserciones que proporciona una forma de verificar los resultados de las pruebas. Juntos, Mocha/Chai permiten escribir pruebas unitarias completas y confiables para el código JavaScript.

Para documentar el código fuente con JSDoc, se pueden seguir los siguientes pasos:

1. Agregar comentarios especiales al código fuente: En el código fuente, se deben agregar comentarios especiales que indiquen la documentación para cada función, variable y clase. Estos comentarios comienzan con "/**" y terminan con "*/" y pueden incluir etiquetas especiales para indicar el tipo de datos, la descripción y otros detalles.
2. Instalar JSDoc: Se debe instalar la herramienta JSDoc mediante el administrador de paquetes npm.
3. Configurar JSDoc: Se debe configurar JSDoc para que use la plantilla de documentación adecuada y para que incluya los archivos y carpetas necesarios.
4. Generar la documentación: Se debe ejecutar la herramienta JSDoc para generar la documentación en formato HTML.

Para realizar pruebas unitarias con Mocha/Chai, se pueden seguir los siguientes pasos:

1. Instalar Mocha y Chai: Se deben instalar las herramientas Mocha y Chai mediante el administrador de paquetes npm.
2. Escribir pruebas unitarias: En un archivo de prueba separado, se deben escribir pruebas unitarias para cada función o método que se quiera probar. Estas pruebas deben incluir una serie de aserciones que verifiquen los resultados esperados.
3. Configurar Mocha: Se debe configurar Mocha para que use la plantilla de informe adecuada y para que incluya los archivos y carpetas necesarios.
4. Ejecutar las pruebas: Se debe ejecutar la herramienta Mocha para ejecutar las pruebas unitarias y generar un informe de resultados.

Con estas herramientas y procesos, se puede proporcionar una documentación clara y completa del código fuente y garantizar la calidad del software mediante pruebas unitarias automatizadas.

Paquetes NPM

Para utilizar paquetes de npm para facilitar la documentación del código fuente y la realización de pruebas unitarias. Aquí te presento algunos paquetes que pueden ser útiles:

- JSDoc: Este paquete permite generar documentación a partir de comentarios especiales en el código fuente. Para instalarlo, se puede ejecutar el siguiente comando: **npm install --save-dev jsdoc**.
- Mocha: Este paquete es un marco de pruebas que proporciona una estructura para escribir y ejecutar pruebas unitarias. Para instalarlo, se puede ejecutar el siguiente comando: **npm install --save-dev mocha**.
- Chai: Este paquete es una librería de aserciones que proporciona una forma de verificar los resultados de las pruebas. Para instalarlo, se puede ejecutar el siguiente comando: **npm install --save-dev chai**.
- Istanbul: Este paquete es una herramienta de cobertura de código que permite medir la cantidad de código que se está probando con las pruebas unitarias. Para instalarlo, se puede ejecutar el siguiente comando: **npm install --save-dev nyc**.
- Eslint: Este paquete es una herramienta de linting que permite detectar errores y problemas de estilo en el código fuente. Para instalarlo, se puede ejecutar el siguiente comando: **npm install --save-dev eslint**.

Para utilizar estos paquetes, se debe configurar adecuadamente el archivo package.json para que se ejecuten las tareas necesarias al realizar pruebas y generar documentación. Por ejemplo, se puede agregar lo siguiente al archivo package.json:

```
"scripts": {
  "test": "mocha",
  "coverage": "nyc mocha",
  "lint": "eslint .",
  "docs": "jsdoc -c jsdoc.json"
}
```

Con esta configuración, se pueden ejecutar las pruebas con el comando **npm test**, verificar la cobertura de código con el comando **npm run coverage**, realizar linting con el comando **npm run lint**, y generar la documentación con el comando **npm run docs**.

Es importante señalar que la configuración exacta puede variar dependiendo de los requisitos y preferencias del proyecto.