Universidad de las Fuerzas Armadas ESPE
Departamento de Ciencias de la Computación
Desarrollo de Aplicaciones Móviles

Tarea 2.2 Crear Api

Integrantes:

Gonzaga Javier
Munarco Jhon
León William
Valdiviezo Darwin

Docente:

Ing. Doris Karina Chicaiza Angamarca

PREGRADO S-II OCT 24 - MAR 25

17/12/2024

Índice

1. Introducción

El presente informe tiene como finalidad el desarrollo de una API REST utilizando Node.js y MongoDB. Las APIs (Interfaces de Programación de Aplicaciones) son herramientas fundamentales en el desarrollo de software moderno, ya que permiten la comunicación entre sistemas diferentes. En este trabajo, se demostrará el proceso de configuración, desarrollo y prueba de una API REST básica que gestiona información de contactos, simulando un escenario real de aplicación móvil.

2. Objetivo General

Desarrollar una API REST básica para la gestión de contactos utilizando Node.js como servidor, Express como framework y MongoDB como base de datos, para proporcionar operaciones CRUD (Crear, Leer, Actualizar y Eliminar).

3. Desarrollo

3.1 Conceptos Claves generales

API REST:

Una API (Application Programming Interface) es un intermediario entre un cliente y un servidor. En este caso, usamos REST (Representational State Transfer), que organiza las solicitudes HTTP para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar).

MongoDB:

Base de datos NoSQL que almacena los datos en formato JSON. Es ideal para aplicaciones con datos dinámicos y sin estructura fija.

Node.js y Express:

Node.js: Permite ejecutar JavaScript en el servidor.

Express.js: Framework que facilita la creación de servidores y APIs en Node.js.

Flutter:

Framework desarrollado por Google para construir aplicaciones móviles, web y desktop. Utiliza el lenguaje Dart y permite desarrollar interfaces modernas.

CRUD:

Las operaciones CRUD son la base del desarrollo de aplicaciones:

C: Crear (Agregar datos)

R: Leer (Mostrar datos)

U: Actualizar (Modificar datos)

D: Eliminar (Borrar datos)

3.2 Herramientas de Desarrollo

Visual Studio Code

Editor de código utilizado para el desarrollo de la API y la app móvil.

Node.js y npm

Node se utiliza para ejecutar el servidor, y npm para gestionar las dependencias del proyecto.

Express.js

Framework de Node.js que facilita el manejo de rutas y solicitudes HTTP.

MongoDB

Almacenamiento de datos en formato JSON. Puedes usar MongoDB local o en la nube con MongoDB Atlas.

Flutter

SDK para desarrollar la aplicación móvil.

Postman

Herramienta para probar la API localmente antes de conectarla con Flutter.

MongoDB Compass

Herramienta visual para administrar la base de datos MongoDB.

3.3 Proceso:

3.3.1 Configuración del entorno

API

Instalación de Node.js:

Descarga Node.js desde nodejs.org e instálalo. Verifica con:

```
node -v
npm -v
```

Configuración de MongoDB:

Instalar MongoDB localmente o usar MongoDB Atlas.

Cómo es local iniciamos MongoDB con:

```
mongod
```

Configuración de Flutter:

Descarga Flutter SDK desde flutter.dev.

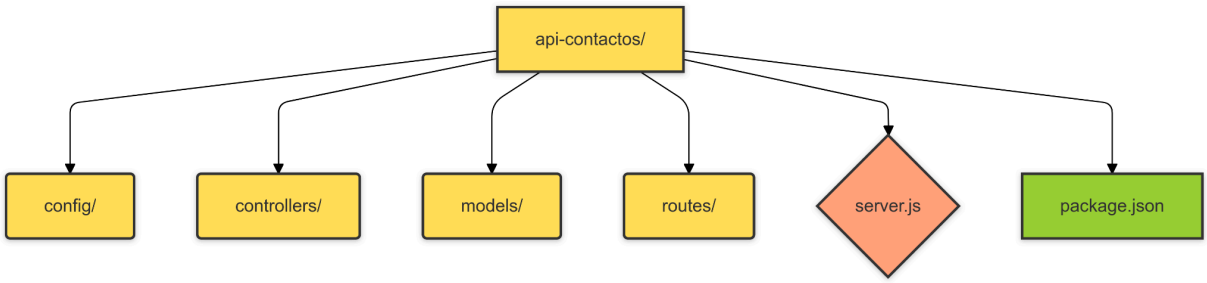
Configura el PATH y verifica con:

```
flutter doctor
```

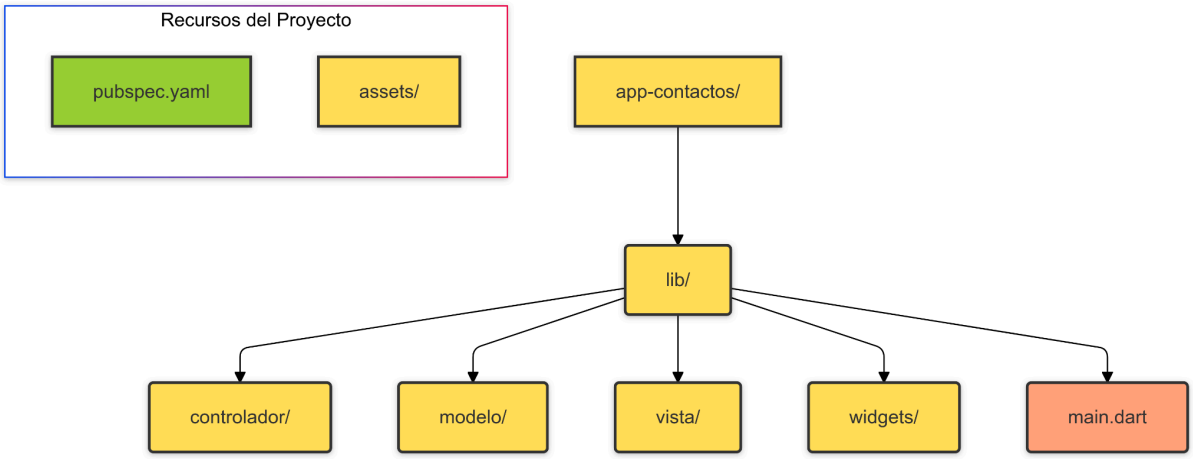
3.3.2 Diseño de la estructura del proyecto

Backend (API con Node.js y MongoDB)

La API tendrá la siguiente estructura:



El Sistema tendrá la siguiente estructura:



3.3.3 Implementación de la aplicación

3.3.3.1 Consumo de la API

Inicializar el proyecto:

Creamos el proyecto y configuramos las dependencias.

Se conecta MongoDB y se define el modelo de datos para contactos (nombre, apellido y teléfono).

Implementación de Rutas CRUD:

GET: Obtener la lista de contactos.

POST: Agregar un nuevo contacto.

PUT: Actualizar un contacto existente.

DELETE: Eliminar un contacto por su ID.

Pruebas de la API:

Usar Postman para verificar que cada ruta funcione correctamente.

3.3.3.2 Operaciones CRUD

La app móvil realiza las siguientes operaciones conectándose a la API:

Lectura de Datos (GET):

La aplicación realiza una solicitud HTTP GET para mostrar los contactos en una lista.

Agregar y Actualizar Contactos:

Se usa un formulario modal (DialogoActualizar) que permite ingresar o modificar los datos del contacto.

Eliminar Contactos:

Se confirma con un diálogo si el usuario desea borrar el contacto antes de enviarlo a la API mediante una solicitud DELETE.

Actualización en Tiempo Real:

Después de realizar cada operación, se refresca automáticamente la lista de contactos.

3.3.3.3 Almacenamiento local

Se usó MongoDB junto con Compass para el almacenamiento NOSQL tipo json de nuestros contactos en nuestro computador.

3.3.4 Diseño de la interfaz

3.3.4.1 Topologia

Pantalla Principal:

- Muestra la lista de contactos en tarjetas con botones de editar y eliminar.
- Botón flotante para agregar nuevos contactos.

Formulario Modal:

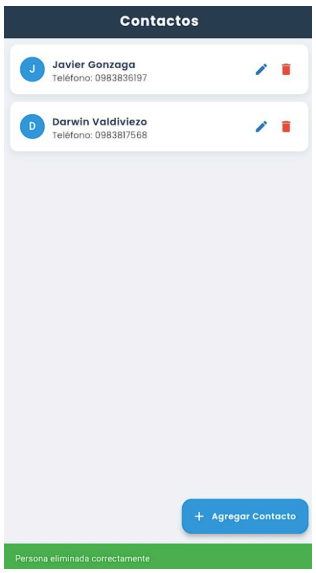
- Se utiliza para agregar o editar un contacto con los campos nombre, apellido y teléfono.
- Los datos se envían a la API.

Confirmación de Eliminación:

- Al borrar un contacto, aparece un diálogo confirmando la acción.

3.3.5 Capturas de pantallas

Figura 1: Pantalla de Bienvenida

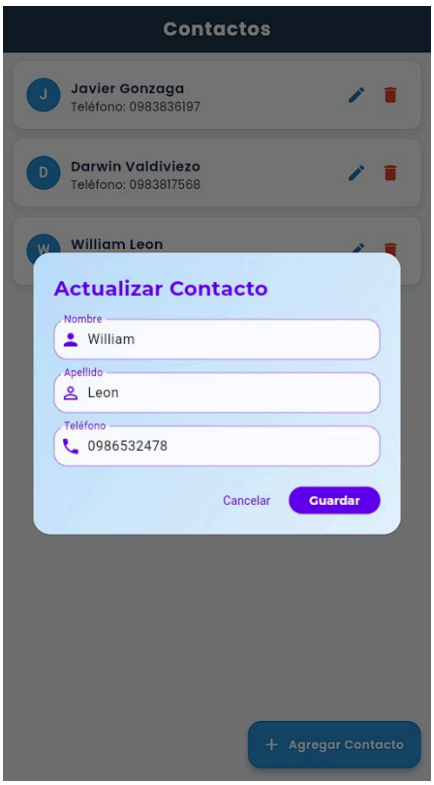


Nota 1: La pantalla presenta un diseño moderno y un botón de ingreso.

Figura 2: Agregar contactos

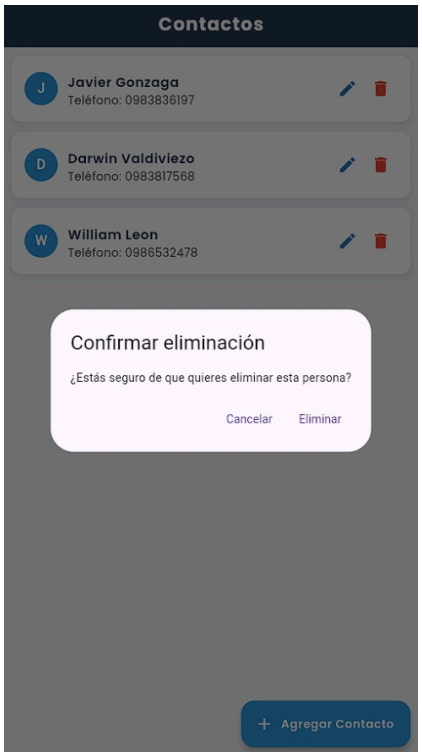
Nota 2: Nos muestra los campos de ingresos de un nuevo contacto.

Figura 3: Actualizar Contacto



Nota 3: Se muestran los campos del contacto a editar

Figura 4: Eliminar contacto



Nota 4: Se visualiza la confirmación de eliminación del contacto.

4. Conclusión

Mediante el uso de Node.js y MongoDB, se desarrolló una API REST funcional, la cual permitió gestionar información de contactos mediante operaciones CRUD.

Mediante la implementación del framework Express y el uso de herramientas como Postman, se facilitó el desarrollo, prueba y validación del funcionamiento de la API.

5. Recomendaciones

Para el desarrollo de APIs REST se recomienda emplear Node.js con Express, por su flexibilidad y facilidad de implementación.

En la gestión de bases de datos no relacionales se sugiere el uso de MongoDB junto con Mongoose, ya que facilita el manejo de documentos y la validación de datos.

Se recomienda el uso de Postman para realizar pruebas exhaustivas de las rutas y validar el correcto funcionamiento de las operaciones CRUD.

6. Enlace Github

https://github.com/DarwinValdiviezo/Tarea2.2Moviles.git

7. Referencias

- Google Developers. (n.d.). *Flutter - Build apps for any screen*. Recuperado el 5 de mayo de 2024, de https://flutter.dev
- Node.js Foundation. (2024). *Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine.* Recuperado de https://nodejs.org
- Express.js. (2024). *Express: Fast, unopinionated, minimalist web framework for Node.js.* Recuperado de https://expressjs.com
- MongoDB Inc. (2024). *MongoDB: The Developer Data Platform.* Recuperado de https://www.mongodb.com
- Dart Team. (2024). *HTTP Package.* Recuperado de https://pub.dev/packages/http
- Microsoft. (2024). *Visual Studio Code: Code editing. Redefined.* Recuperado de https://code.visualstudio.com