Diseño y Programación de Software Multiplataforma

ACILO DE ANCIANO ESPERANZA

MANUAL TÉCNICO

Desarrollado por:

- Luis Edgardo Villalta Reinoza VR181981

INTRODUCCIÓN

La finalidad de todo manual técnico es la de proporcionar al lector las pautas de configuración y la lógica con la que se ha desarrollado una aplicación.

REQUERIMIENTOS DE DESARROLLO

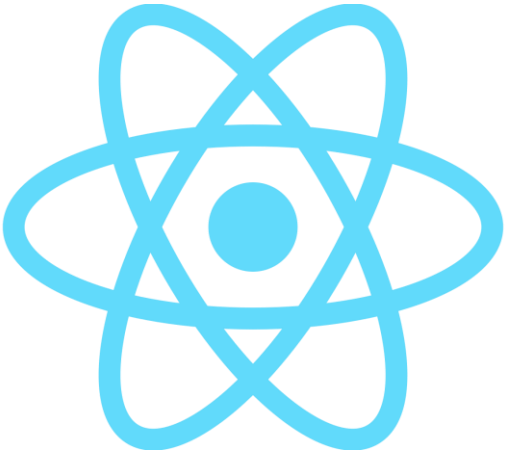
ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA.

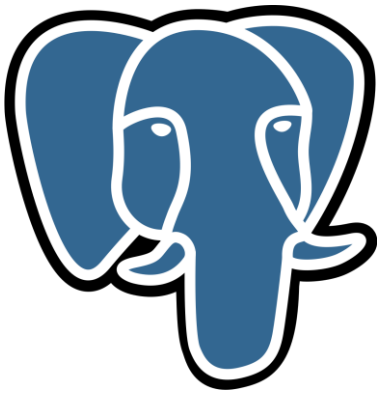
NODE JS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

REACT NATIVE



React Native, es un framework de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP al permitir que los desarrolladores usen React con las características nativas de estas plataformas.



POSTGRESQL

PostgreSQL, también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT.

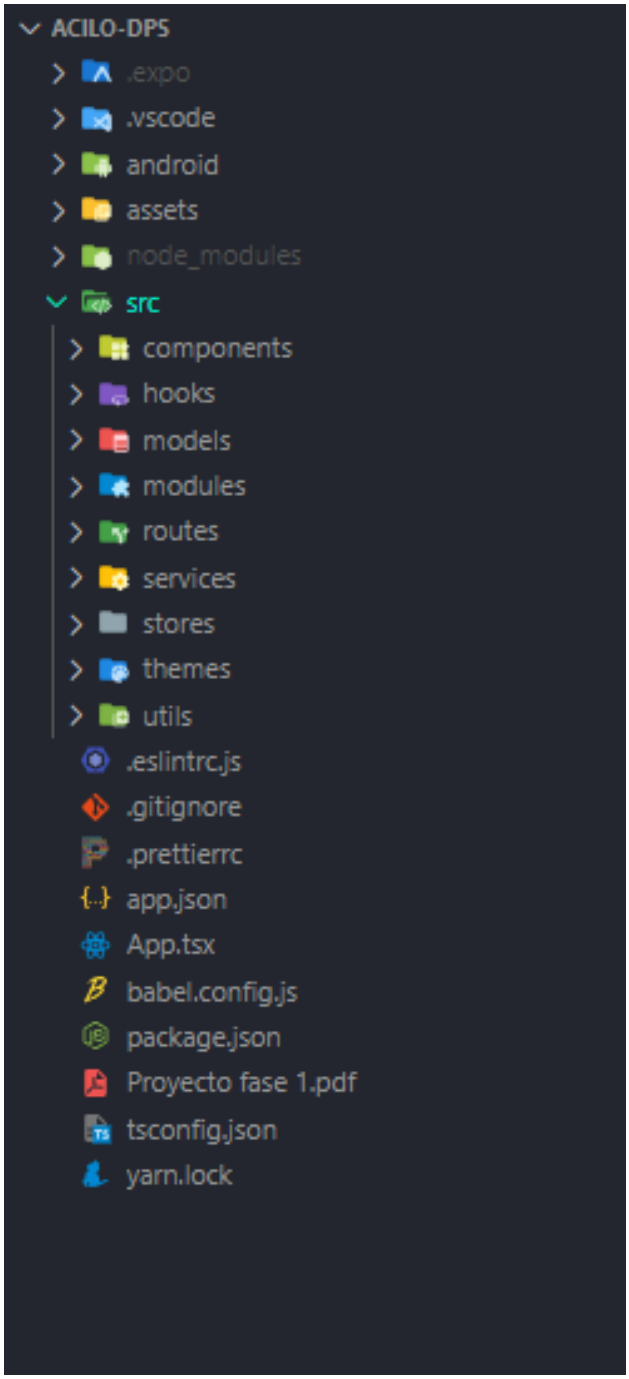
NEST JS

Nest. js es un framework de desarrollo web basado en Node. js que utiliza TypeScript para proporcionar una estructura de programación sólida y altamente escalable.

FIREBASE

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles lanzada en 2011 y adquirida por Google en 2014

ESTRUCTURA FRONTEND



El proyecto esta estructurado de la siguiente forma:

- Android: Proyecto de android studio
- Assets: Imágenes e iconos
- Src: Carpeta root del proyecto de react native
  - Components: Componentes compartidos por toda la aplicación
  - Hooks
  - Models: Representación de objetos de la base de datos
  - Módulos: Cada modulo del proyecto, cada modulo tiene las siguientes subcarpetas:
    - Components
    - Constantes
    - Hooks
    - Screens (pantallas)
    - Stores (gestión de estados)
    - Styles: Estilos
    - Validaciones
    - Types
  - Routes: rutas y navegación
  - Services: servicios y consultas a la api
  - Stores: Gestión de estados globales
  - Themes: Temas y colores
  - Utils: Funciones y utilidades generales

DEPENDENCIAS UTILIZADAS

Se listaran los paquetes de terceros utilizados dentro del proyecto:

- ## Zustand

Zustand es una solución de gestión de estados pequeña, rápida y escalable.
https://github.com/pmndrs/zustand

- ## Formik

Formik es una librería declarativa que se encarga de manejar todos los posibles escenarios al interactuar con un input, de cualquier tipo que este sea. - Generar errores descriptivos. - Procesamiento de la información. - Interacción con el usuario.
https://github.com/jaredpalmer/formik

- ## React-query

Biblioteca de administración de estado y recuperación de datos que simplifica enormemente el proceso de manejar información asíncrona en aplicaciones React.
https://github.com/tanstack/query

- ## React native paper

Paper es una colección de componentes personalizables y listos para producción para React Native, siguiendo las pautas de Material Design de Google.
https://github.com/callstack/react-native-paper

- ## React native vector icons

Íconos personalizables para React Native con soporte para NavBar/TabBar, fuente de imagen y estilo completo.
https://github.com/oblador/react-native-vector-icons

- ## Moment y momento-timezone

Librería para trabajar con con los formatos de fechas y calendarios como: día, mes, hora, minutos, segundos, am-pm, nombres del día, mes y rangos de fechas.
https://github.com/moment/moment/

- ## Localforage

Es una biblioteca de almacenamiento rápida y sencilla para JavaScript. localForage mejora la experiencia fuera de línea de su aplicación web mediante el uso de almacenamiento asincrónico.
https://github.com/localForage/localForage

- ## Lodash

Una biblioteca de utilidades de JavaScript que ofrece coherencia, modularidad, rendimiento y extras.
https://github.com/lodash/lodash

- ## @react-native-google-signin/google-signin

https://github.com/react-native-google-signin/google-signin

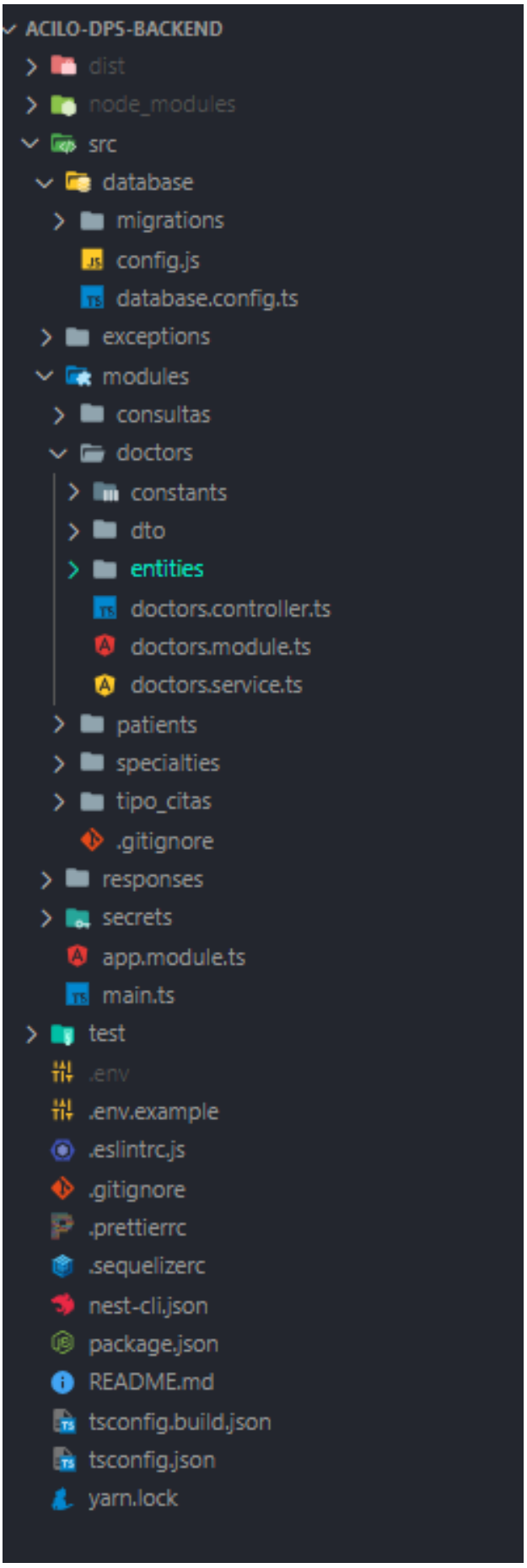
- ## Yup

Yup es un creador de esquemas para el análisis y validación de valores en tiempo de ejecución. Defina un esquema, transforme un valor para que coincida, afirme la forma de un valor existente, o ambas cosas.
https://github.com/jquense/yup

ESTRUCTURA BACKEND

El backend esta hecho utilizando el framework **NestJs,** y sigue la siguiente estructura:

- Dist: se genera automáticamente al iniciar el servidor.
- Src: estructura del proyecto, tiene la siguiente estructura interna:
    - Database: Carpeta con la configuración de la base de datos y migraciones
    - Exceptions: Mensajes de error preestablecidos para ser usado como respuesta por cada controlador.
    - Modules: NestJs sigue el esquema de módulos, cada modulo incluye:
        - Constants: Constantes utilizadas
        - Dto: Objeto de transferencia de datos (usadas en los post, put y patch)
        - Entities: entidades que son representación de un objeto de la base de datos, se utiliza el ORM Sequelize para establecer sus atributos y relaciones.
        - Archivo controlador: Con la declaración de cada endpoint y llamados al servicio.
        - Archivo de servicio: Con la conexión a la base de datos utilizando el ORM
        - Archivo Module: Archivo que encapsula la configuración y cada archivo del modulo, que luego será invocado en el archivo app.module.ts para registrarlo en el sistema.
    - Responses: Respuestas preestablecidas para ser usadas en las llamadas de la api
    - Secrets: Llaves secretas para la configuración de módulos
    - App.module.ts: Archivo que encapsula cada modulo para que puedan ser ocupados
    - Main.ts: Archivo de configuración del servidor


ACILO-DPS-BACKEND
dist
node_modules
src
database
migrations
config.js
database.config.ts
exceptions
modules
consultas
doctors
constants
dto
entities
doctors.controller.ts
doctors.module.ts
doctors.service.ts
patients
specialties
tipo_citas
.gitignore
responses
secrets
app.module.ts
main.ts
test
.env
.env.example
.eslintrc.js
.gitignore
.prettierrc
.sequelizerc
nest-cli.json
package.json
README.md
tsconfig.build.json
tsconfig.json
yarn.lock

DEPENDENCIAS UTILIZADAS

- **NestJs**

Framework de desarrollo web basado en Node. js que utiliza TypeScript para proporcionar una estructura de programación sólida y altamente escalable.
https://nestjs.com/

- **Sequelize**

ORM que permite a los usuarios llamar a funciones javascript para interactuar con SQL DB sin escribir consultas reales.
https://github.com/sequelize/sequelize

- **@nestjs/sequelize**

Documentación de integración y uso de sequelize con nestjs
https://docs.nestjs.com/techniques/database#sequelize-integration

- **Swagger**

Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful.
https://swagger.io/

- **@nestjs/swagger**

Documentación de como documentar un endpoint.
https://docs.nestjs.com/openapi/introduction