BACKEND

Archivo ApiNodeMySQL

Ejecute “npm i” para instalar todas las dependencias en una terminar de Visual Studio Code, NodeJs

carpeta config

- config.js
  Contiene los parámetros para la conexión con la base de datos.
  Se realiza la comunicación con la Base de datos de la API.
- env.js
  Para el JWT
- key.js
  Es la clave de cifrado que utilizara la Api para crear un jwt.
- passport.js
  Para que se puedan realizar peticiones si estas autenticado y filtrar las búsquedas utilizando el ID de los usuarios que cuenten con JWT para las posteriores peticiones.

Carpeta controllers

Se establecen los métodos de entrada de cada ruta, también contiene las funcionalidades de las peticiones, recibiendo datos obtenidos de los modelos.

- userControllers.js
  contiene las funciones que se ocupan de los usuarios como el insert y el create.

Carpeta models

Se establecen las consultas SQL con la base de datos de la API.

- rol.js: contiene la consulta para la creación del rol de usuario, que por defecto se definió que cada nuevo usuario le dará el rol de cliente.
- user.js: Contiene las consultas para el usuario, agregar.

Carpeta routes

Se definen las rutas para cada acción de la API.
En este archivo se encuentra las diferentes rutas que se utilizaran en el api para enviar y recibir peticiones por parte del usuario

- userRoutes.js
  (POST)

Carpeta utils

En esta carpeta se encuentran las configuraciones con la base de datos en Firebase, para el almacenamiento de imágenes (foto de perfil del usuario).

- async_foreach.js
- cloud_storage.js

Archivo package.json

Contiene las dependencias instaladas para las funcionalidades del proyecto

Archivo server.js

Es el archivo principal del proyecto, este contiene las configuraciones para el funcionamiento del api.

API

El archivo userRoutes.js contiene las siguientes rutas:

app.post('/api/users/create',usersController.register);

app.post('/api/users/login',usersController.login);

app.post('/api/users/roles',usersController.registerRol);

app.put('/api/users/update',upload.array('image',1),usersController.update;

Cada ruta define las operaciones que tendrán disponibles

POST: Para ingresar datos.
PUT Para modificar datos.
GET: Para obtener datos.
Ejemplo:

```
const usersController = require('../controllers/usersControllers');

//,upload
module.exports = (app, upload)=>{

    //POST para enviar datos
    app.post('/api/users/create', usersController.register);
    app.post('/api/users/login', usersController.login);
    app.post('/api/users/roles', usersController.registerRol);

    //PUT para actualizar datos
    app.put('/api/users/update', upload.array('image', 1), usersController.update);

}
```

En la carpeta userControllers se crean los métodos de entrada, con dos parámetros req y res para recibir y mandar datos de las peticiones de igual manera se hacen las validaciones de los tokens y los roles de usuario.

FRONTEND

Carpeta view

Contiene todos los archivos de tipo Fragment del proyecto, todas las vistas de la aplicación

Carpeta ui

Contiene los archivos de tipo Fragment para el recorrido del menú desplegable.

Carpeta activitysLogin

Contiene archivos de tipo Activity, estas son vistas para la nueva contraseña y el registro del usuario.

Carpeta administrador

Contiene archivos de tipo Activity, son las vistas que únicamente al administrado podrá ver.

Carpeta adapters

El archivo se usa para el recyclerView para acomodar la información de los productos en el cardview.

Carpeta api

- ApiRoutes: contiene la ruta principal y las funciones principales de las rutas que se definieron en la API en Visual Studio Code
- RetrofitClient: configuración de retrofit2

Carpeta models: Contienen los parámetros que se enviarán y recibirán del api.

- ResponseHttp
- Rol
- User

Carpeta providers: se enviará o recibirá la información de las rutas.

- UserProvider

Carpeta routers

Se declara las rutas a la que nos vamos a dirigir para cada procedimiento necesario.

- UserRoutes

Carpeta utils

Se encarga de almacenar la información cuando el usuario inicie sesión, esto para guardar o almacenar sus acciones dentro de la aplicación.

- SharedPred

Carpeta res

- Carpeta drawable

  Contiene imágenes, logos, diseños para las interfaces

- Carpeta font

  Contiene la tipografía, el tipo de letra en la aplicación.

- Carpeta layout

Contiene los layout de tipo fragmet para las interfaces de la aplicación.

Archivo builds.gradle

Contiene todas las implementaciones ocupadas en el desarrollo de la aplicación.