



USAC

TRICENTENARIA

Universidad de San Carlos de Guatemala

INGENIERIA

CUNOC

Ing. Pedro Luis Domingo Vásquez

Teoría de Sistemas 1

Proyecto Final: Manual Técnico

Sección: A

Nombre:

Registro académico:

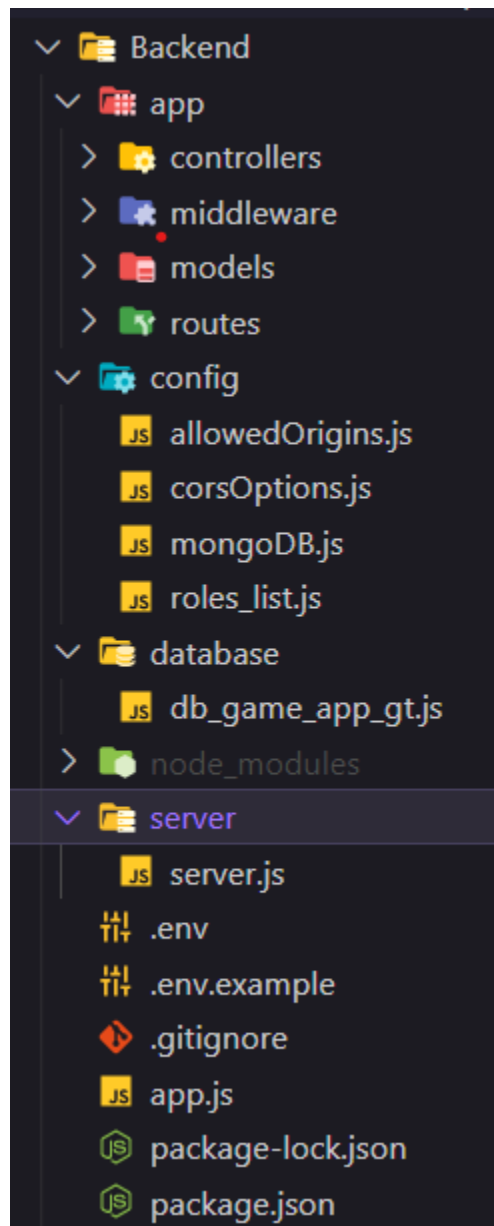
Mariano Francisco Camposeco Camposeco
Luis Nery Cifuentes Rodas
Manuel Antonio Rojas Paxtor

202030987
202030482
202030799

Quetzaltenango, 24 de mayo de 2023

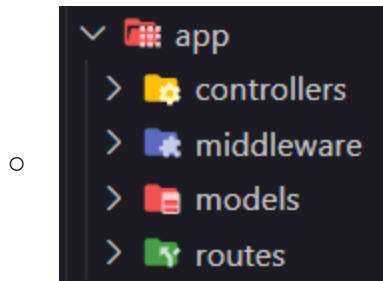
Manual Tecnico

Backend



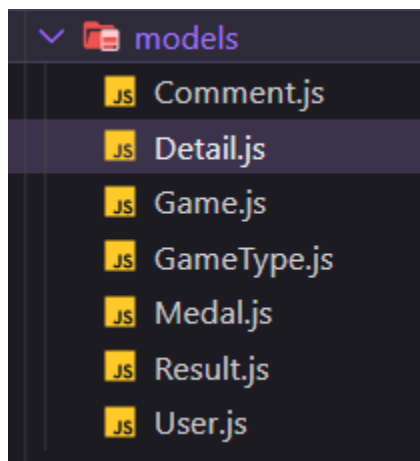
Esta es la estructura que se utilizó para el backend de la Aplicación Interactiva de Aprendizaje, tiene una estructura íntegra basada en el formato “Modelo → Controlador → Vista”. Dicha estructura maneja 4 carpetas principales en la raíz del proyecto, un archivo “app.js” el cual contiene el funcionamiento del backend y un archivo “.env” que guarda las variables para el funcionamiento del proyecto.

- **Carpeta App**



- Esta carpeta contiene 4 paquetes en su interior: “Controllers, Models, Middleware y Routes”. Cada carpeta con un objetivo en específico, albergar y separar los archivos en base a su funcionamiento.

- **Models**



- Esta carpeta guarda todos los archivos que se utilizaran para realizar las interacciones necesarios con la base de datos, generalmente todos manejan la siguiente estructura:

```

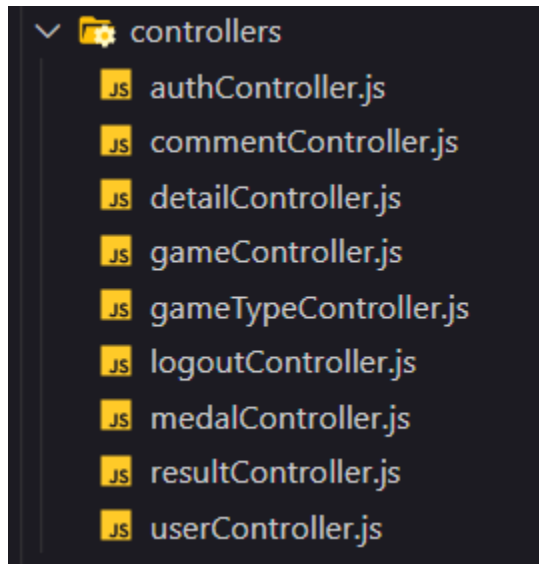
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const model = mongoose.model;

const gameSchema = new Schema({
  user: {
    type: Schema.Types.ObjectId,
    ref: "users",
    required: true,
  },
  game_type: {
    type: Schema.Types.ObjectId,
    ref: "gametypes",
    required: true,
  },
  description: {
    type: String
  },
  name: {
    type: String,
    required: true,
  },
  time: {
    type: Number
  },
  data_game: {
    details: {
      type: Schema.Types.ObjectId,
      ref: "Detail"
    }
  },
  data: Array
}, {
  versionKey: false
});

```

-
- Cada archivo posee una estructura similar, se hacen las importaciones necesarias y luego se crea una nueva instancia de Schema en la cual se define la estructura de la colección que se almacenará en la base de datos.

- Controllers



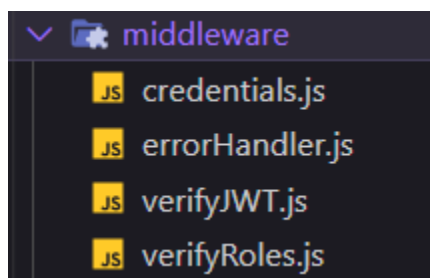
-
- Esta carpeta guarda los archivos que tendrán relación directa con las peticiones http del cliente para el servidor.

```
const Detail = require('../models/Detail');

const insertDetail = async (req, res, next) => {
  try {
    const data = req.body.data;
    const newDetail = new Detail({
      data
    });
    const insert = await newDetail.save();
    const data_game = {
      details: insert._id
    }
    res.status(200).json(data_game);
  } catch (error) {
    res.status(500).json({ error: 'Ocurrio un error interno en el servidor...' });
  }
}
```

-
- Generalmente cada archivo guarda métodos relacionados con un modelo en específico, algunos de ellos sirven como intermediarios entre la petición y un controlador final.
- Los controladores tienen como nombre la acción la que van a realizar en la base de datos.

- Middleware



- En esta carpeta se almacenan la mayor parte de de interceptores es decir métodos que se ejecutaban antes del controlador, por ejemplo los autenticadores, los autorizadores entre otros.

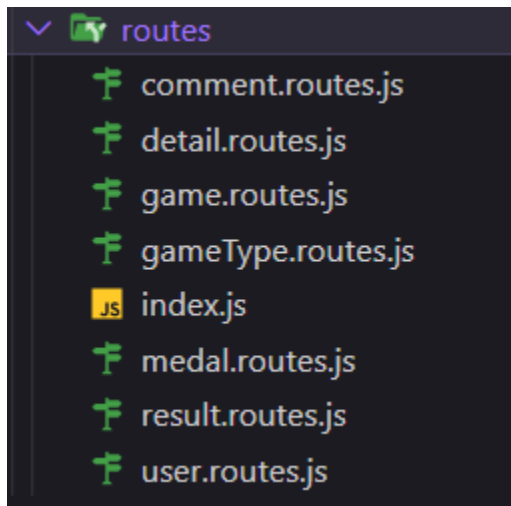
```
const jwt = require('jsonwebtoken');

const verifyJWT = (req, res, next) => {
  const authHeader = req.headers.authorization || req.headers.Authorization;
  if (!authHeader?.startsWith('Bearer ')) return res.sendStatus(401);
  const token = authHeader.split(' ')[1];
  jwt.verify(
    token,
    process.env.ACCESS_TOKEN_SECRET,
    (err, decoded) => {
      if (err) return res.sendStatus(403); //invalid token
      req.user = decoded.UserInfo.username;
      req.roles = decoded.UserInfo.roles;
      req.userId = decoded.UserInfo.id;
      next();
    }
  );
}

module.exports = verifyJWT
```

- Estos middleware poseen una estructura similar a la de un controlador pero con un parámetro de “next” el cual solo significa que continuará con otro middleware o que será procesado por un controlador.

- Routes



- En esta carpeta se guardan las rutas relacionadas con los tipos de controlador, existe un archivo “index.js” en donde se integran todas las rutas de los controladores.

```

const express = require("express");
const routes = express.Router();
const ROLES_LIST = require("../config/roles_list");
const verifyRoles = require("../middleware/verifyRoles");
const verifyJWT = require("../middleware/verifyJWT");
const {
  createNewUser,
  deleteUser,
  getAllUsers,
  getUser,
  updateUser,
} = require("../controllers/userController");
const auth = require("../controllers/authController");

routes
  .route("/")
  .get([verifyJWT, verifyRoles(ROLES_LIST.Admin)], getAllUsers)
  .post(createNewUser)
  .patch([verifyJWT, verifyRoles(ROLES_LIST.Admin)], updateUser)
  .delete([verifyJWT, verifyRoles(ROLES_LIST.Admin)], deleteUser);

routes.get("/profile", verifyJWT, verifyRoles(ROLES_LIST.Admin, ROLES_LIST.Student, ROLES_LIST.Admin), getUser);

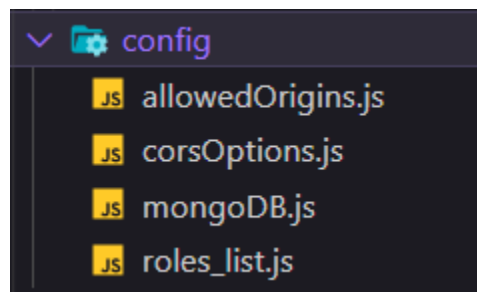
routes.route("/login").post(auth.handleLogin);
routes.route("/logout").get(auth.handleLogout);

routes.get("/refresh", auth.handleRefreshToken);
module.exports = routes;

```

- Cada ruta posee una petición http: get, post, patch o delete, dependiendo la ruta está poseerá middleware o directamente el controlador.

- Carpeta Config



-
- En esta carpeta se guardan archivos de configuración de la aplicación y sus componentes.

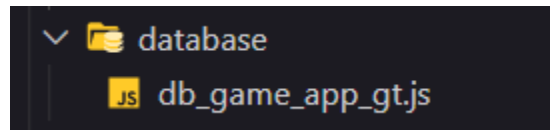
```

//requerimos mongoose
const mongoose = require('mongoose')
//funcion a exportar para llamar
const conexionBD = () => {
  //llamamos url de env
  const URLDB = process.env.URLDB;
  //ip version 4 es lo de family
  mongoose.connect(URLDB, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    family: 4
  }).then((res) => {
    console.log('Conectado');
  }).catch((ex) => {
    console.log('Error en la ruta establecida para la conexion DB');
  });
}
module.exports = { conexionBD };

```

-
- Por ejemplo este archivo: “mongoDB.js”, se configura las instrucciones necesarios para conectarse correctamente a la base de datos MongoDB.

- Carpeta Database



-
- Esta carpeta solo guarda un archivo con los datos iniciales para nuestra base de datos.

```
db.gametypes.insertMany([
  {
    name: "Memoria",
    description:
      "El juego de memoria es un clásico entretenido y desafiante que se puede jugar en una página web. El objetivo principal del juego es po
  },
  {
    name: "Ahorcado",
    description:
      "El juego del ahorcado es una variante del clásico en el que los jugadores intentan adivinar una palabra oculta. Tienen un número limit
  },
  {
    name: "Preguntas y respuestas",
    description:
      "El juego de preguntas y respuestas es una actividad interactiva en la que se plantean preguntas y los jugadores intentan responderlas
  },
  {
    name: "Descifrado",
    description:
      "El juego de Descifrado es un desafío en el que se presenta una palabra mezclada en letras al azar. El jugador debe descifrar la palabr
  },
]);
```

-
- Simplemente guarda instrucciones con scripts de mongodb.

- Archivo ".env"

```
URLDB=mongodb://localhost:27017/game_app_gt
ACCESS_TOKEN_SECRET=5439d7b0051098ac0c8fc19ba16cccd779675935c6635f1f4e965a4bbc536f547f9324f00402302fc0f909b28b807851a4fec612fcdc395ef80013e9f
REFRESH_TOKEN_SECRET=6fd2f4754e2e7ff5026307c1c4859c3425d34d1854c80736047d689764d998381751e9cab389e6ba4744d57fd79a68f1d9f978cb1716b6eae6e7f70
PUERTO=5000
HOST=http://localhost
SOCKET_PORT=5010
```

-
- Este archivo guarda variables globales que pueden ser cambiadas en cualquier momento y que simplifica tanto su acceso como su edición.

- Archivo “app.js”
 - Este archivo contiene todos los recursos que serán utilizados para el funcionamiento del backen, como las rutas, librerías, servers, entre otros, prácticamente es el unificador de todas las partes y es el responsable de lanzar la app en un puerto.

```
require("dotenv").config();
const express = require("express");
const cookieParser = require("cookie-parser");
const cors = require("cors");
const path = require("path");
const { conexionBD } = require("../config/mongoDB");
const app = express();
const body_parser = require("body-parser");
const routes = require("../app/routes/index");
const morgan = require("morgan");
const credentials = require("../app/middleware/credentials");
const corsOptions = require("../config/corsOptions");
const http = require('http');
const socket = require('../server/server');

const PUERTO = process.env.PUERTO || 5000;
const SOCKET_PORT = process.env.SOCKET_PORT || 5010;
const server = http.createServer();
const io = socket.getIo(server);
app.use(morgan('dev'));
//Hacer uso de cors y permitir los json
app.use(express.json());
app.use(body_parser.json());
app.use(credentials);
app.use(cors(corsOptions));
//middleware for cookies
app.use(cookieParser());
app.use("/api/v1.0", routes);
//ubicacion de las imagenes
app.use("/public", express.static(path.join(`${__dirname}/uploads/img`)));
//Que nuestra aplicacion escuche por el puerto
server.listen(SOCKET_PORT, () => {
  console.log("Puerto: ", SOCKET_PORT, " socket habilitado");
});
conexionBD();
app.listen(PUERTO, () => {
```

- Carpeta Server

```
const ROLES_LIST = require("../config/roles_list");

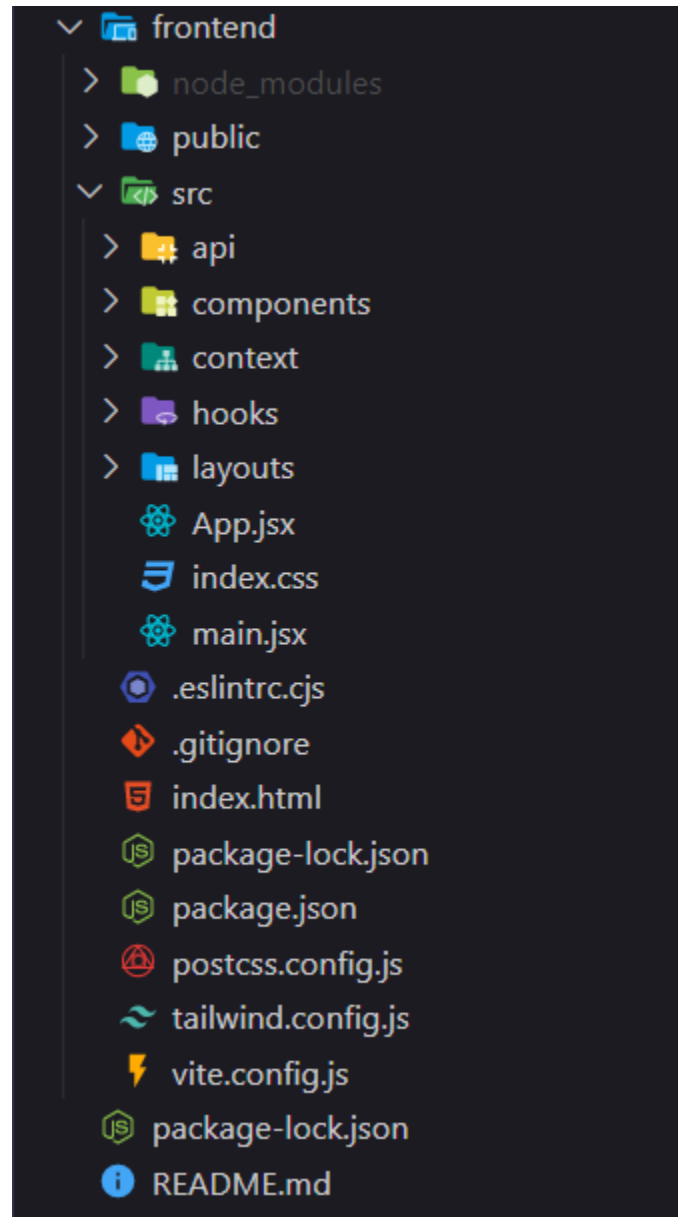
module.exports = {
  getIo: (server) => {
    const io = require('socket.io')(server, {
      cors: { origin: 'http://localhost:5173' }
    });

    let lobbyUsers = [];
    let clavelobby = '';

    io.on('connection', (socket) => {
      console.log('Conectado al socket');
      socket.on('auth', (data) => {
        const { roles } = data;
        if (roles?.includes(ROLES_LIST.Teacher)) {
          const min = 100000;
          const max = 999999;
          clavelobby = toString(Math.floor(Math.random() * (max - min + 1)) + min);
          lobbyUsers = [];
          socket.emit('Clave de Lobby', clavelobby);
        }
      })
    })
  }
}
```

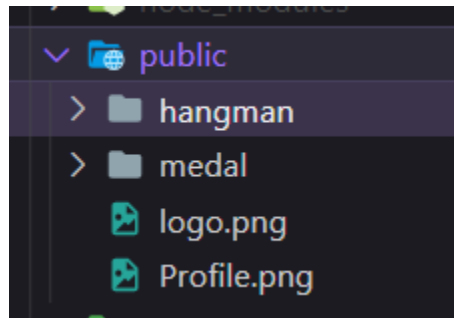
- Prácticamente posee un archivo que guarda los métodos que permiten que el juego sea colaborativo es decir que permita la interacción entre los usuarios.

Frontend



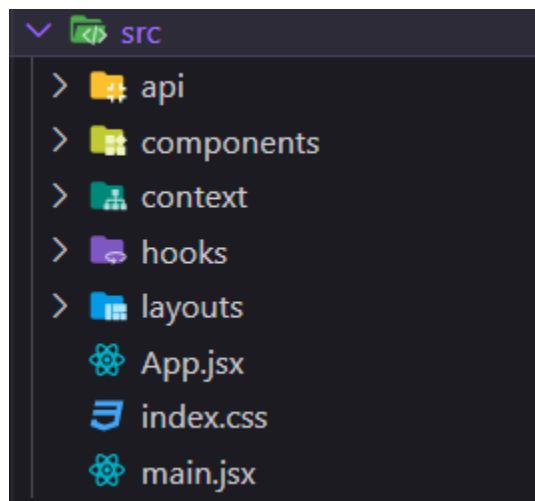
El frontend maneja paquetes los cuales están diseñados para utilizarse en conjunto como componentes y archivos que integran una funcionalidad para el frontend como un componente.

- Carpeta Public



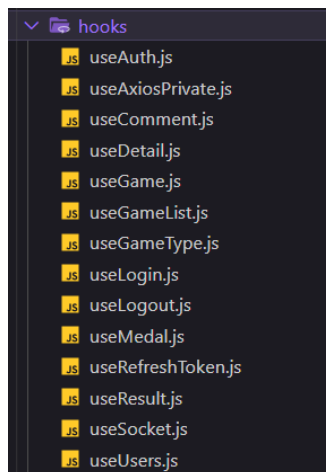
-
- Esta carpeta almacena archivos de uso público como imágenes etc.

- Carpeta SRC



-
- Esta carpeta contiene todo el funcionamiento tanto de vistas como de los juegos de la Aplicación Interactiva de Aprendizaje.
- Entre los paquetes a resaltar más importantes son:

- Hooks



- Prácticamente son archivos que manejan los estados de información para ser implementados en los componentes.

```
import { useContext, useEffect, useState } from "react";
import { io } from "socket.io-client";
import useAuth from "../useAuth";
import AuthContext from "../context/AuthProvider";

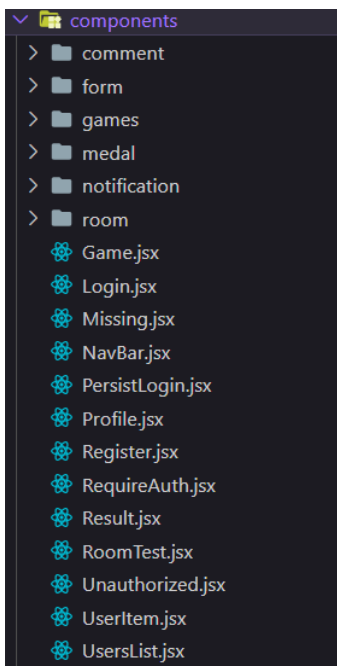
const socket = io.connect("http://localhost:5010");

const useSocket = (codeInit = 0, usernameInit = "guest") => {
  const { auth } = useContext(AuthContext);
  const [players, setPlayers] = useState([]);
  const [game, setGame] = useState({});
  const [code, setCode] = useState(codeInit);
  const [error, setError] = useState("");

  const createRoomCode = (game) => {
    return new Promise((resolve, reject) => {
      socket.emit("new_code", { roles: auth.roles, game });
      socket.on("new_game", (data) => {
        const { code } = data;
        setCode(code);
        resolve(code);
      });
    });
  };
};
```

-
- Prácticamente deberían ser archivos que manejen lógico y no contenido html para los componentes.

■ Components



-
- Prácticamente son archivos que se pueden renderizar para mostrar nueva información, asimismo con documentos que

muestran las herramientas gráficas al cliente para que pueda interactuar con el servidor.

```
const handleNext = () => {
  setCurrentWord((prevWord) => prevWord + 1);
  setCompletoWord(false);
  setSelectedLetters([]);
}

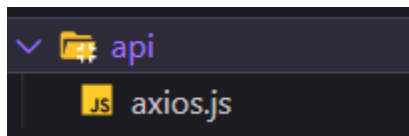
const isLetterClicked = (letter) => letter.clicked;

const wordN = wordList[currentWord];
const isLastWord = currentWord === wordList.length - 1;

return (
  <div className="w-3/4 mt-14 md:w-1/2 mx-auto">
    {currentWord === wordList.length ? (
      <div className="flex justify-center items-center">
        <p className="text-xl font-bold">El juego ha terminado</p>
      </div>
    ) : (
      <div className="card h-32 rounded-lg text-center bg-amber-500 p-4 mb-4 flex items-center justify-center">
        <h3 className="text-xl font-bold text-white">Ordena la palabra</h3>
      </div>
    )}
  </div>
);
```

- Son archivos que manejan lógica y contenido html.

■ API



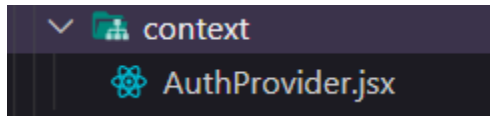
```
import axios from 'axios';
const BASE_URL = 'http://localhost:5000/api/v1.0';

export default axios.create({
  baseURL: BASE_URL
});

export const axiosPrivate = axios.create({
  baseURL: BASE_URL,
  headers: { 'Content-Type': 'application/json' },
  withCredentials: true
});
```

- Prácticamente define la estructura del componente que va a consumir la api.

■ Context



-

```
import { createContext, useState } from "react";

const AuthContext = createContext({});

export const AuthProvider = ({ children }) => {
  const [auth, setAuth] = useState({});
  const [persist, setPersist] = useState(JSON.parse(localStorage.getItem("persist")) || true);

  return (
    <AuthContext.Provider value={{ auth, setAuth, persist, setPersist }}>
      {children}
    </AuthContext.Provider>
  )
}

export default AuthContext;
```

- Es un archivo que se utiliza para compartir datos entre componentes sin necesidad de pasar props manualmente a través de varios niveles de jerarquía de componentes.

■ App.js

```
import CommentUser from "../components/comment/CommentGuest";
import Medal from "../components/medal/Medal";
import UsersList from "../components/UsersList";
import Ranking from "../components/games/Ranking";
import PruebaRanking from "../components/games/PruebaRanking";
import RoomForm from "../components/room/RoomForm";
import Profile from "../components/Profile";
import GamePublicList from "../components/games/list/GamePublicList";

export const ROLES = {
  Student: 2000,
  Teacher: 1580,
  Admin: 5002,
};

function App() {
  return (
    <Routes>
      <Route path="/" element={<Layout />} />
      { /* public routes */ }
      <Route path="/login" element={<Login />} />
      <Route path="/register" element={<Register />} />
      <Route element={<PersistLogin />} />
      <Route element={<NavBarLayout />} />
      <Route path="/" element={<RoomTest />} />
      <Route path="/games" element={<GamePublicList />} />
      <Route path="/room" element={<Room />} />
      <Route path="/enter-room" element={<RoomForm />} />
      <Route path="/room/:code" element={<Room />} />
      <Route path="/commentsUsers" element={<CommentUser />} />
      <Route path="/ranking/:game" element={<Ranking />} />
      <Route path="/pruebaRanking" element={<PruebaRanking />} />
    </Routes>
  );
}
```

- Prácticamente es un archivo que reúne los componentes que deben mostrarse para los usuarios, es un archivo unificador de vistas.