PROYECTO 4 FAMILIAS

4 Familias: Página web que permite gestionar documentos pdf, donde los usuarios pueden loguearse y consultar y descargarse los pdfs pertenecientes a su familia. Proyecto realizado con tecnología javascript, Node.js y Express para Backend, Postman y React para Frontend .

Proyecto realizado por: Juan Garrido Troche.

Contenido

[4 FAMILIAS 2](#_Toc139380350)

[Título 2](#_Toc139380351)

[Descripción 2](#_Toc139380352)

[Usuarios anónimos 2](#_Toc139380353)

[Usuarios logueados como usuarios 2](#_Toc139380354)

[Usuario logueado como administrador 2](#_Toc139380355)

[BASE DE DATOS 3](#_Toc139380356)

[INTRODUCCIÓN 5](#_Toc139380357)

[DEPENDENCIAS NECESARIAS PARA SAAWIT 6](#_Toc139380358)

[RUTAS / ENDPOINTS 8](#_Toc139380359)

# 4 FAMILIAS

El proyecto web consistirá en la muestra de una página principal con un input que pida una contraseña. Dependiendo de la contraseña introducida le llevará a una de las 4 familias que le mostrará otra página única con los pdfs de esa familia, con posibilidad de lectura y descarga y un formulario (Ayúdanos a mejorar) por si el usuario quiere aportar información acerca del árbol genealógico mostrado en el pdf.

Existirá un administrador (5º password) que le mostrará el contenido de las 4 familias). En la página principal mostrará las 4 familias que al seleccionar una de ellas, mostrará los pdfs con posibilidad de sustituirlo por otro fichero, eliminarlo o enviar uno nuevo que subirá el administrador.

## Título

4 Familias (4F).

## Descripción

En “4 Familias”, implementaremos una API que permita gestionar pdfs por parte del administrador, donde los usuarios puedan loguearse y vero o descargarlos pdfs pertenecientes a su familia.

## Usuarios anónimos

* Visualizar la página de login.

## Usuarios logueados como usuarios

* Visualizar la página de login.
* Visualizar la página con todos los pdfs pertenecientes a la familia.
* Descargar los pdfs pertenecientes a la familia.

## Usuario logueado como administrador

* Visualizar la página de login.
* Visualizar todas las familias.
* Visualizar la página con todos los pdfs pertenecientes a la familia donde cada pdf:
  + Puede eliminar cualquier pdf.
  + Puede sustituir cualquier pdf.
  + Puede subir un nuevo pdf.
* Descargar los pdfs pertenecientes a la familia.

# BASE DE DATOS

El cliente ha solicitado que no haya Base de datos como tal si no guardar los datos en un fichero. Hemos elegido almacenar los datos en un fichero JSON ya que es un formato muy extendido para el intercambio de datos, con los siguientes atributos y valores:

[

  {

    "id": "nD-uOe-J5S-TI",

    "lineage": "Anca",

    "pdf": [],

    "password": "$2b$10$2q16ou.scLZBnv60aXKDAOad9oOa0b1EGkqiZwvu0It1n4D5bf4Fq",

    "active": true,

    "url": "anca",

    "role": "user",

    "logo": "oak.svg"

  },

  {

    "id": "rj-6E8-SUC-4S",

    "lineage": "Cabalar",

    "pdf": ["cabalar.pdf"],

    "password": "$2b$10$rbuyjIC.djFDHtiDJs3Oh.9gIZ3fOPSHz8J8bVyEc/SYcxzrWH6e2",

    "active": true,

    "url": "cabalar",

    "role": "user",

    "logo": "hazelnut.svg"

  },

  {

    "id": "lt-Ghi-uSI-Th",

    "lineage": "Seco",

    "pdf": ["seco.pdf", "seco-2.pdf"],

    "password": "$2b$10$J1BRBz0tMuacYlog37OFkOxTiqUi6AKI3YF3/Bo/RnLANKOo2OsUu",

    "active": true,

    "url": "seco",

    "role": "user",

    "logo": "oak"

  },

  {

    "id": "Ec-Fyl-M07-qV",

    "lineage": "Díaz",

    "pdf": ["diaz.pdf"],

    "password": "$2b$10$7TPYsxzbGl0eFjI4q/asqOZPn0XuxKckK8MmyqKSsP0ejVM2RLg8e",

    "active": true,

    "url": "diaz",

    "role": "user",

    "logo": "oak"

  },

  {

    "id": "fs-VvW-X6l-hI",

    "lineage": "seco-admin",

    "pdf": [],

    "password": "$2b$10$EDnH.l30yXSTaQ35Y8h9ke.1zoIRi9HoJfuK3B8EeqRtd/Y.DQTcC",

    "active": true,

    "url": "administrator",

    "role": "admin",

    "logo": "oak"

  }

]

# BACKEND

Antes de instalar alguna dependencia crearemos nuestro fichero package.json con la siguiente línea de comandos, situada en la carpeta de nuestro proyecto:

* + - npm init
    - npm init -y (si queremos crear package.json con valores por defecto)

En nuestro proyecto queda del siguiente modo:

{

  "name": "4F",

  "version": "1.0.0",

  "description": "API que permite gestionar ficheros pdf",

  "main": "server.js",

  "scripts": {

    "dev": "nodemon server.js"

  },

  "repository": {

    "type": "git",

    "url": "git+https://github.com/JuanGarridoTroche/4f.git"

  },

  "author": "Juan Garrido Troche",

  "license": "ISC",

  "bugs": {

    "url": "https://github.com/JuanGarridoTroche/4f/issues"

  },

  "homepage": "https://github.com/JuanGarridoTroche/4f#readme",

}

# DEPENDENCIAS NECESARIAS PARA 4F

Las dependencias son aquellas aplicaciones o bibliotecas requeridas por otro programa para poder funcionar correctamente. Por ello, se dice que determinado programa “depende” de tal aplicación o biblioteca (library). Para instalarlos, emplearemos un sistema de gestión de paquetes de Node llamado “npm” (Node Package Management).

Existen 3 tipos de dependencias:

* Dependencias CORE: Aquellas que ya vienen instaladas y se pueden utilizar solo llamando al módulo, como, por ejemplo, path o fs.
* Dependencias: Aquellas necesarias que instalaremos para que nuestra API funcione de manera correcta. Ejemplo: mysql2, dotenv, bcrypt, etc.
* Dependencias para desarrolladores(-D): Aquellas dependencias que nos harán la programación más fácil como control de sintaxis (eslint), formateo del código para que se vea ordenado (prettier) o reinicio de nuestro servidor cada vez que guardamos un cambio (nodemon). Este tipo de dependencias se instalan añadiendo un -D en la línea de comandos. En cuanto finalice nuestro proyecto y lo pongamos en producción se podrían desinstalar (npm uninstall mi\_DevDependence).

A continuación, vamos a indicar que dependencias son las que vamos a utilizar en nuestro proyecto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NOMBRE | INSTALACIÓN | LLAMADA | TIPO | USO |
| express | npm i express | const express = require("express") | Dependencia | Nos permite levantar un servidor. Además nos permite leer los datos enviados desde el body en formato raw -JSON. |
| dotenv | npm i dotenv | require("dotenv").config(); | Dependencia | Nos permite leer los valores de las variables del archivo oculto .env |
| bcrypt | npm i bcrypt |  | Dependencia | Permite encriptar y desencriptar datos. |
| nodemon | npm i nodemon -D | **X** | Dev | Monitoriza los cambios en el código fuente que se está desarrollando y automáticamente reinicia el servidor que está corriendo sin tener que reiniciarlo. |
| jsonwebtoken | npm i jsonwebtoken | const jwt = require("jsonwebtoken") | Dependencia | Nos crea un token de seguridad estándar para transmitir de forma segura en internet, por medio del formato JSON. |
| fileUpload | npm i express-fileupload | const fileupload = require("express-fileupload") | Dependencia | Decodifica (o deserializa) los datos enviados desde el body con el formato “form-data”. |
| fs | Npm i fs | const fs = require("fs/promises") | Dependencia | Gestiona el CRUD de ficheros. |
| path | No hay que instalar | const path = require("path") | Core | Gestiona las rutas de nuestros ficheros. |
| joi | Npm i @hapi/joi | const joi = require("@hapi/joi") | Dependencia | Sirve para validar todo tipo de datos introducidos por el usuario. |
|  |  |  |  |  |
| nodemailer | npm i nodemailer | const nodemailer = require("nodemailer") | Dependencia | Nos permite enviar correos a través de una configuración sencilla. |
| morgan | npm i morgan | const morgan = require("morgan") | Dev | Monitorización de datos en tiempo real, visto desde el terminal. |
|  |  |  |  |  |
|  |  |  |  |  |

# MIDDLEWARES

Un middleware es una función que se puede ejecutar antes o después del manejo de una ruta. Esta función tiene acceso al objeto Request, Response y la función next().

Antes de empezar a programar nuestra API debemos establecer los middlewares necesarios para que nuestras rutas o endpoints funcionen:

|  |  |  |
| --- | --- | --- |
| MÉTODO | MIDDLEWARE | USO |
|  | const **app** = express() | Crea el servidor |
| USE | app.use('/static', express.static(UPLOADS\_DIR)) | Acceso a la carpeta de "UPLOADS\_DIR" que tenemos en el fichero .env. Desde fuera se llamará static |
| USE | app.use(morgan("dev")) | Monitorización de datos en tiempo real, visto desde el terminal. |
| USE | app.use(express.json()) | Deserializa el body con formato JSON |
| USE | app.use(fileupload()) | Deserializa el body con formato form-data (Método CRUD para ficheros). |
| USE | app.use(cors()) | Cross-Origin of Resource Sharing: Permite conectar el backend con el frontend. Un user-agent obtiene permiso para acceder a recursos de este servidor. |
|  |  |  |

# RUTAS / ENDPOINTS

Un endpoint es una pasarela que conecta los procesos del servidor de la aplicación con una interfaz (API, Application Programming Interface), es decir, la ruta que hay que añadir a nuestra URL, a la que se envían peticiones.

Antes de empezar a programar nuestra API debemos establecer los endpoints necesarios para que nuestra aplicación sea funcional:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MÉTODO | RUTA | FUNCIÓN | USO | PERMISOS |
| ENDPOINTS USUARIOS | | | | |
| POST | /login | login.js | Login de usuario. | Todos los usuarios |
| GET | /loggedProfile | readLoggedProfile.js | Muestra todos los datos del usuario logueado. | isAuth |
| GET | /familyNames | getFamilyNames.js | Obtener el nombre de todas las familias. | isAuth |
| POST | /form/sendForm | sendForm.js | Enviar sugerencia a través del formulario | isAuth |
| GET | /:url | showLineage.js | Acceso a los pdfs de la familia. | Todos los usuarios |
| GET | /data/:url | getFamilyDataByUrl.js | Obtener los datos de la familia a través de la url | isAuth |
|  |  |  |  |  |
| ENDPOINTS PDFS | | | | |
| POST | /createPDF/:url | createPDF.js | Sube un pdf como administrador | isAuth |
| PUT | /pdf | updatePDF.js | Actualiza un pdf como administrador | isAuth |
| DELETE | /pdf | deletePDF.js | Elimina un pdf como administrador | isAuth |
|  |  |  |  |  |

# FRONTEND

Para el uso de React utilizaremos el instalador de VITE:

$ npm create vite@latest

? **Project name:** » vite-projectfrontend

√ **Project name:** ... frontend

? **Select a framework:** » - Use arrow-keys. Return to submit.

> Vanilla

Vue

React

Preact

Lit

Svelte

Others

√ **Select a framework:** » Vanilla

? **Select a variant:** » - Use arrow-keys. Return to submit.

> JavaScript

TypeScript

√ **Select a variant:** » JavaScript

Scaffolding project in D:\Programación\PROYECTOS\xpaccount\frontend...

Done. Now run:

cd frontend

npm install

npm run dev

En nuestro caso Elegiremos a las preguntas las siguientes respuestas:

# PAGES

<Login />

|  |  |
| --- | --- |
| **COMPONENTE** | <Login /> |
| **Ruta** | /users/login |
| **Atributos** |  |
| **Ubicación** | pages/Login.js |
| **Uso** | Carga la página de Login |

# COMPONENTES

<Header />

|  |  |
| --- | --- |
| **COMPONENTE** | <Header /> |
| **Ruta** | /users/login |
| **Atributos** |  |
| **Ubicación** | pages/Login.js |
| **Uso** | Carga la página de Login |