Plataforma web

Manual técnico de SIGN TO ALL

Presentado por: - Dayana Aranda Penagos,
- Niyireth Lorena Cortes Ballesteros
- Cesar Leonardo Figueroa Cruz
- Danilo Arturo Varon Silva
- Luz Adriana Yara Tique

Contenido

1. Objetivos

Explicación de los principales objetivos

- Dar a conocer toda la información necesaria a los administradores que llevaran a cabo el control de la plataforma digital SIGN TO ALL.
- Representar la estructura técnica del prototipo.

2. Requerimientos mínimos de Hardware

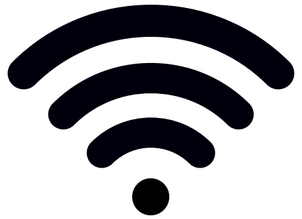
Especificación

Computador(usuario)

Procesador: 2 Núcleos - 2 GHz ✓ Memoria RAM: mínimo: 4 Gigabytes (GB) ✓ Disco Duro: 128GB.

Servidor

Procesador: 1 v CPU ✓ Memoria RAM: 512 MB ✓ 20 GB SSD.



Conexión a internet

Para usar Sign To All es necesario que cuentes con una conexión a internet para poder ingresar al sistema.

3. Requerimientos mínimos de Software

Especificación

Dependencias

Node >= v16.14.0

Npm >= v8.3.1

Angular CLI >= v13.1.4

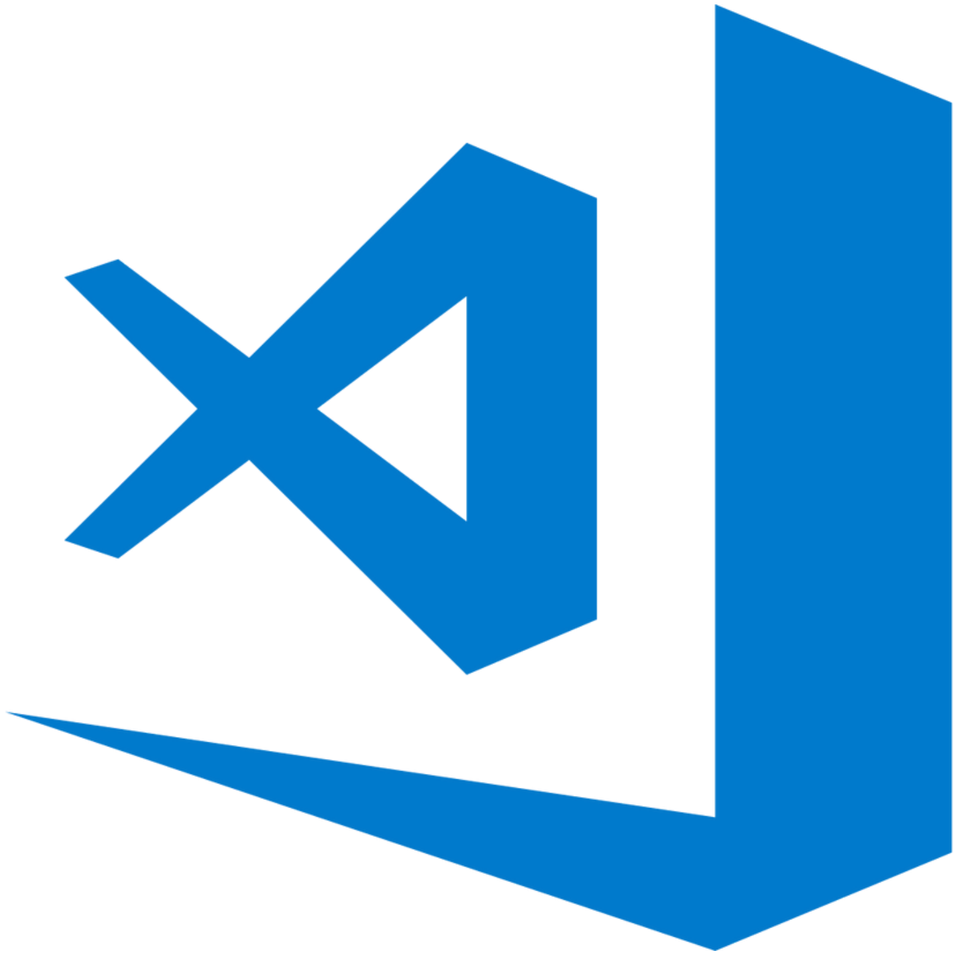
Navegador

Cualquier navegador basado en chromium o firefox.

4. Herramientas usadas para el desarrollo

Especificación

4.1 Visual Studio Code



Visual Studio Code es un editor de código abierto desarrollado por Microsoft, potable, ligero y rápido. Este editor de texto es gratuito además permite trabajar con varios lenguajes de programación diferentes, también posee varias funcionalidades al instalar sus diversas extensiones las cuales proporcionan gran facilidad al usuario a la hora de programar, y ayudan a obtener una mejor visualización en el código,

4.2 HTML

Html no es un lenguaje de programación, este es un lenguaje de marcado de hipertexto (HyperText Markup Language), este lenguaje se escribe con elementos los cuales a su vez están construidos por etiquetas, contenido y atributos.

Este puede ser trabajado desde cualquier editor básico para textos, a la vez es multiplataforma, esto quiere decir que será visualizado desde cualquier navegador en cualquier sistema operativo, podemos adicionar que en este lenguaje de marcado no hay distinción entre mayúsculas y minúsculas.

4.3 CSS

Por sus siglas CSS (Cascading Style Sheets) significan hojas de estilo de cascada, las cuales parten de un concepto simple el cual es aplicar estilos (colores, formas, márgenes, entre otros) a uno o varios documentos (usualmente HTML) de forma masiva.

Se le denomina estilos en cascada debido a que se aplican de arriba hacia abajo y en caso de existir ambigüedad, se siguen una serie de normas para resolver estos conflictos.

4.4 Angular

Angular es uno de los framework's más potentes en la actualidad que se usa TypeScript el cual es de código abierto para la programación, este framework está diseñado para desarrollar en aplicaciones dinámicas de una sola página.

Angular facilita el desarrollo de aplicaciones SPA por sus siglas en ingles “single page aplication”, es decir todas las vistas, se muestran en la misma página y también se considera un framework MVC (Modelo Vista Controlador), este es desarrollado por Google para el desarrollo Web y FrontEnd

4.5 TypeScript

TypeScript es un lenguaje de programación orientado a objetos de código abierto y distribución libre a la vez un superconjunto de JavaScript que agrega tipados estáticos y objetos basados en clases, es decir a como se declaran las variables.

4.6 Node.js

Node.js es un entorno de tiempo de ejecución, es JavaScript en el lado del servidor, este utiliza un modelo de entrada (peticiones) y salida (respuestas) para una comunicación eficaz con el FrontEnd.

4.7 Express

Express es un marco de desarrollo para node, este permite estructurar un programa de una manera más ágil, en si es un framework que ayuda al desarrollo de la aplicación eficazmente.

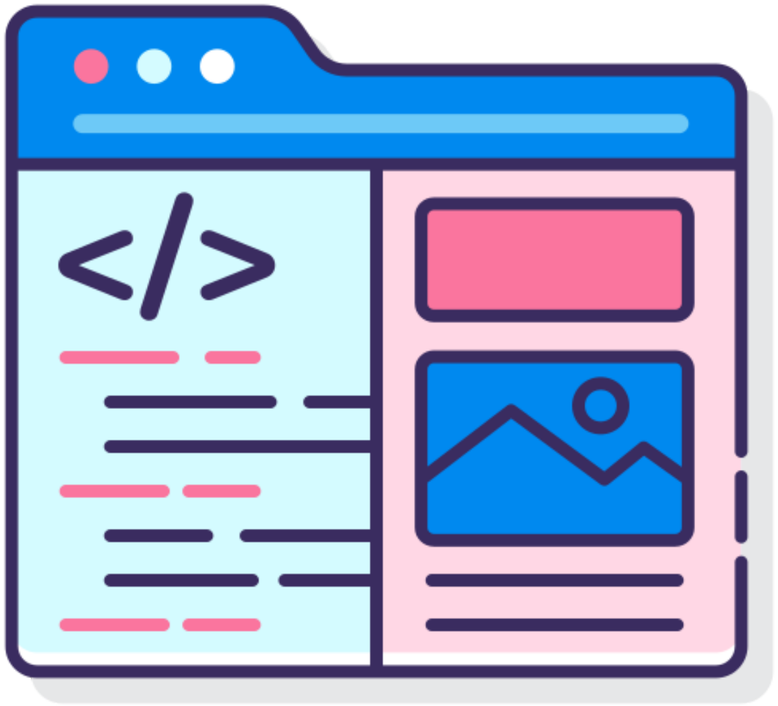
4.8 MySQL

MySQL es un sistema de gestión de base de datos relacional, el cual utiliza tablas que se conectan entre si para almacenar la información y organizarla correctamente.

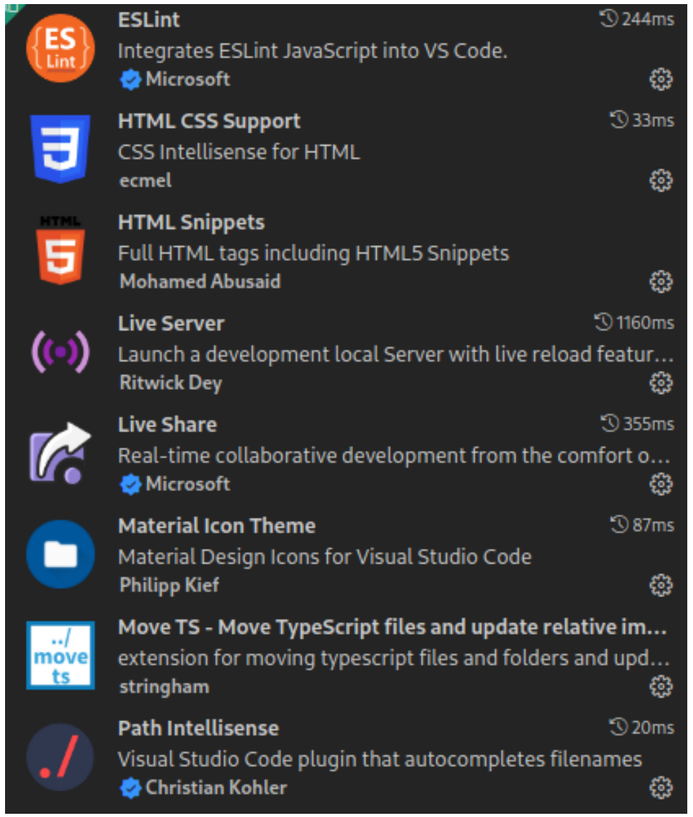
5. Configuración: FrontEnd

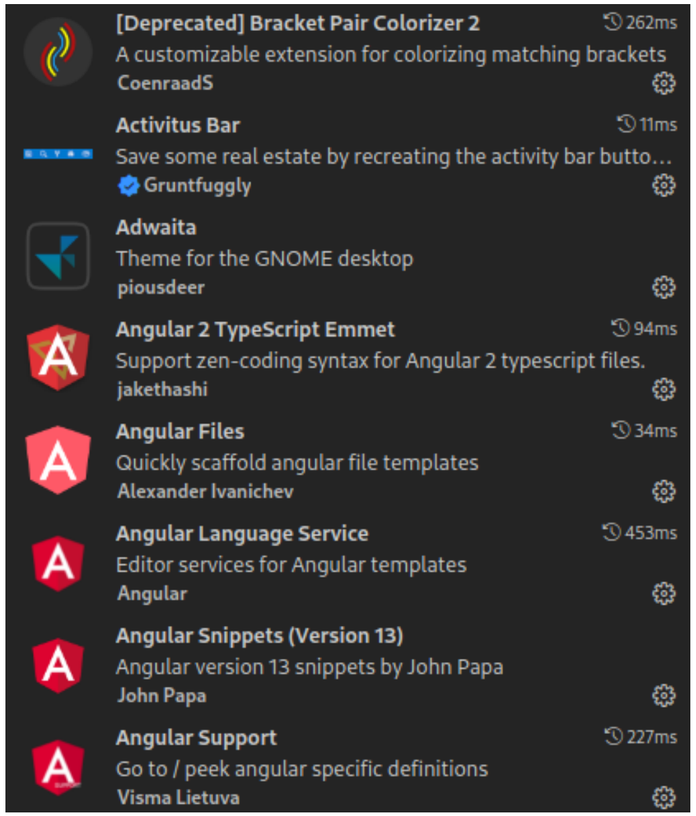
Configurar entorno de desarrollo

Para configurar el entorno debes tener ya instalado las dependencias mencionadas en los requisitos de software.



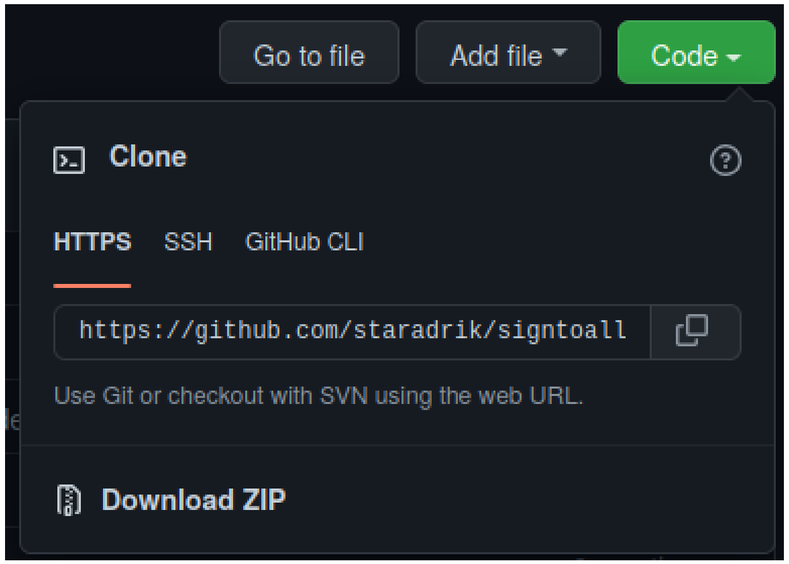
5.1 Extensiones recomendadas para visual studio code





Estas son las extensiones usadas por el equipo original de Sign to all, recomendamos investigar sobre EsLint o ng lint, ya que el proyecto usa los lineamientos y buenas practicas de Angular y esto puede causar errores al no seguir estas.

5.2 Descargar repositorio del proyecto



El repositorio del proyecto se encuentra en github y en el repositorio de la universidad

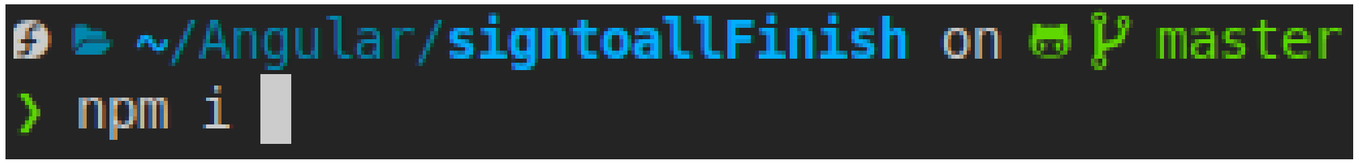
Descargar desde github

Link: https://github.com/staradrik/signtoallFinish.git

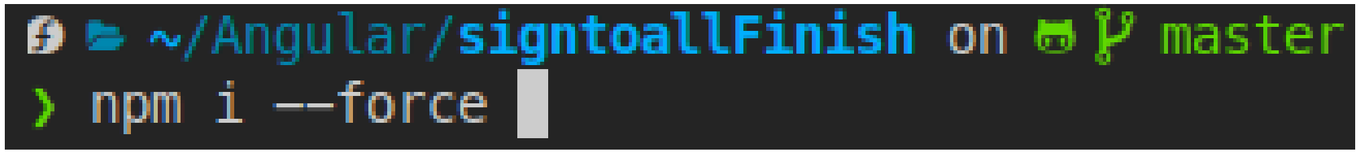
Descargar desde repo de la universidad

5.3 Descargar dependencias del proyecto

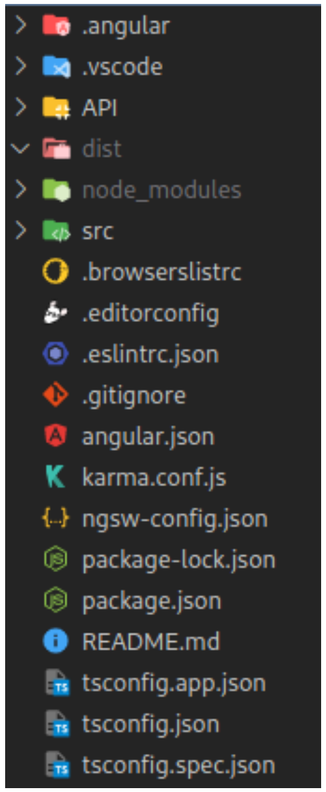
Para descargar las dependencias debes abrir la terminal en la carpeta del proyecto y ejecutar el siguiente comando.



En caso de error ejecutar el siguiente

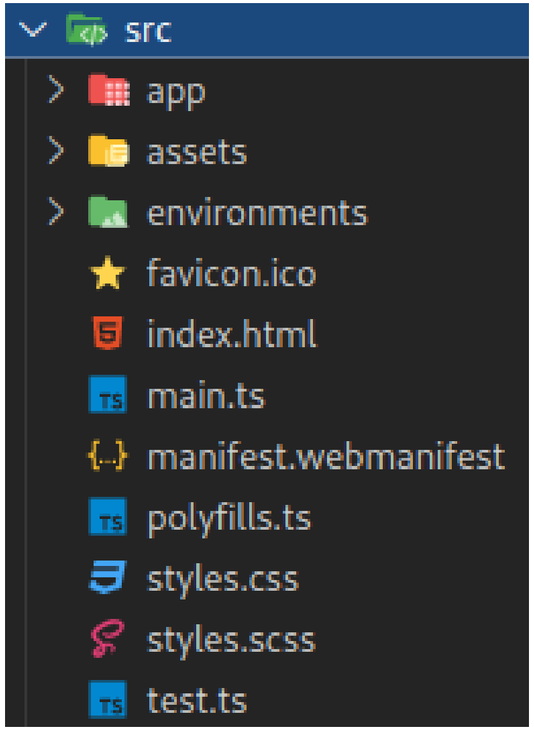


5.4 Estructura del proyecto. "./"



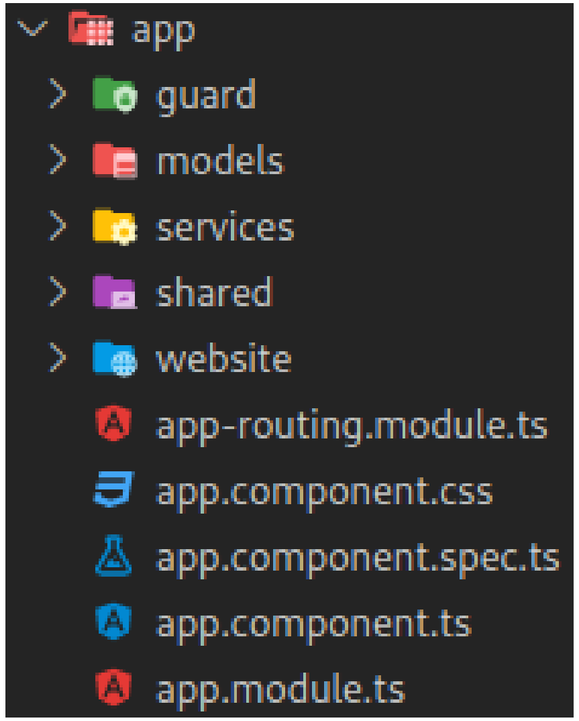
- API: La api para el BackEnd.
- dist: El proyecto listo para producción.
- node_modules: Se encuentran las dependencias del proyecto.
- src: El codigo del proyecto.
- angular.json: Las configuraciones del proyecto.
- package.json: Declaradas las dependencias del proyecto pero se recomienda npm para administrarlas.

5.4.1 Estructura del proyecto. "./src"



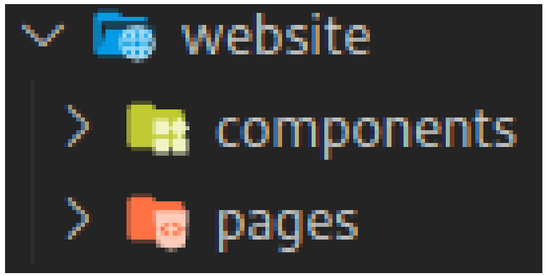
- app: El codigo principal
- assets: Archivos como los .json, imagenes de las actividades.
- environments: Los archivos de los entornos

5.4.2 Estructura del proyecto. "./src/app"



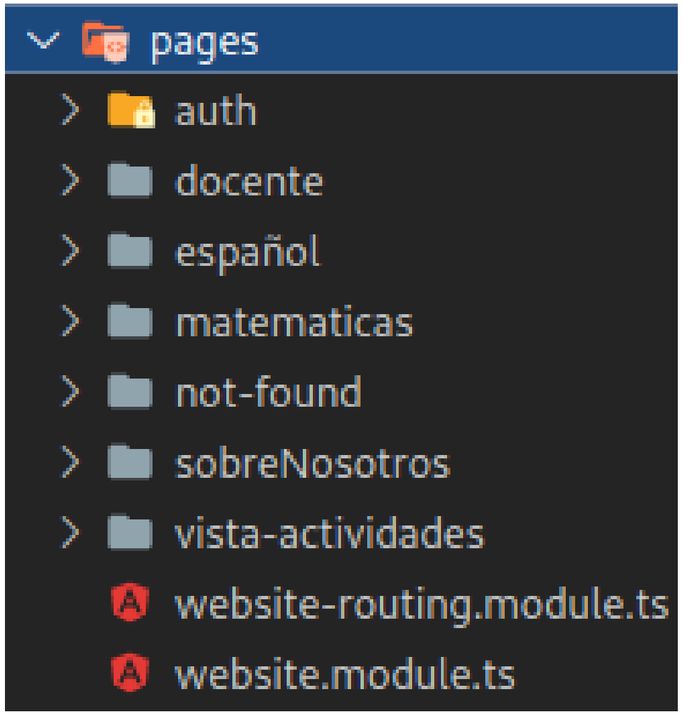
- guard: Los guards.
- models: Modelos del proyecto.
- services: Servicios del proyecto.
- shared: componentes compartidos entre modulos.
- website: las carpetas component y pages.

5.4.3 Estructura del proyecto. "./src/app/website"



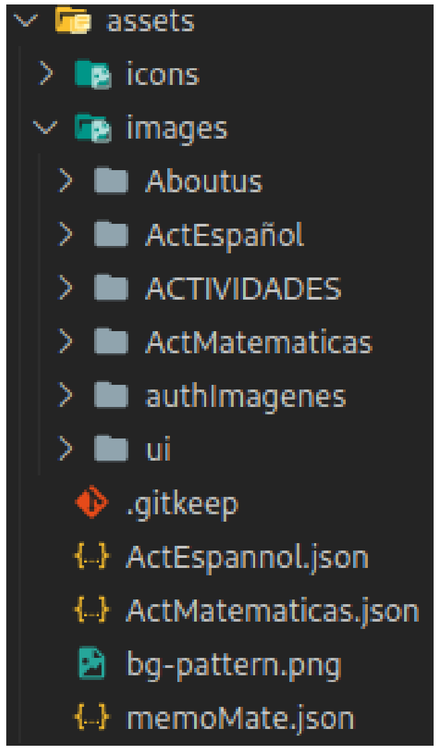
- Components: Componentes como layout, logout, breadcrumb y panel-menu.
- pages: Los modulos de auth, docente, español y matematicas, ademas tiene los componentes not-found, sobreNosotros y vista actividades.

5.4.4 Estructura del proyecto. "./src/app/website/pages"



- auth: Los componentes de inicio de sesión y registro.
- docentes: El componente de crud-docente y el panel menu modificado.
- español: capeta donde contiene todas las actividades del area de español.
- matematicas: Capeta donde contiene todas las actividades del area de matematicas.

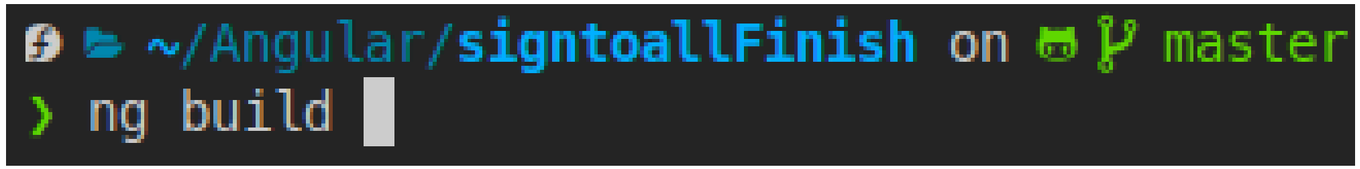
5.4.5 Estructura del proyecto. "./src/assets"



Images: Contiene las carpetas principales de las imagenes, aboutus, ActEspañol con las imagenes usadas en la imagenes de las actividades de español, Actividades imagenes de la vista-actividades, ActMatematicas con las imagenes usadas en la imagenes de las actividades de matematicas, authImagenes con las imagenes usadas en la autentificacion.

5.5 Construir el proyecto.

Para construir el proyecto debemos asegurarnos de tener la URL del enviroment.prod.ts correcta del servidor del BackEnd en la variable API, ademas de que en caso de tener problemas al construir el proyecto haber modificado correctamente los tamaños de la aplicación en budget en el archivo angular.json, el comando necesario para construir el proyecto para producción es el siguiente



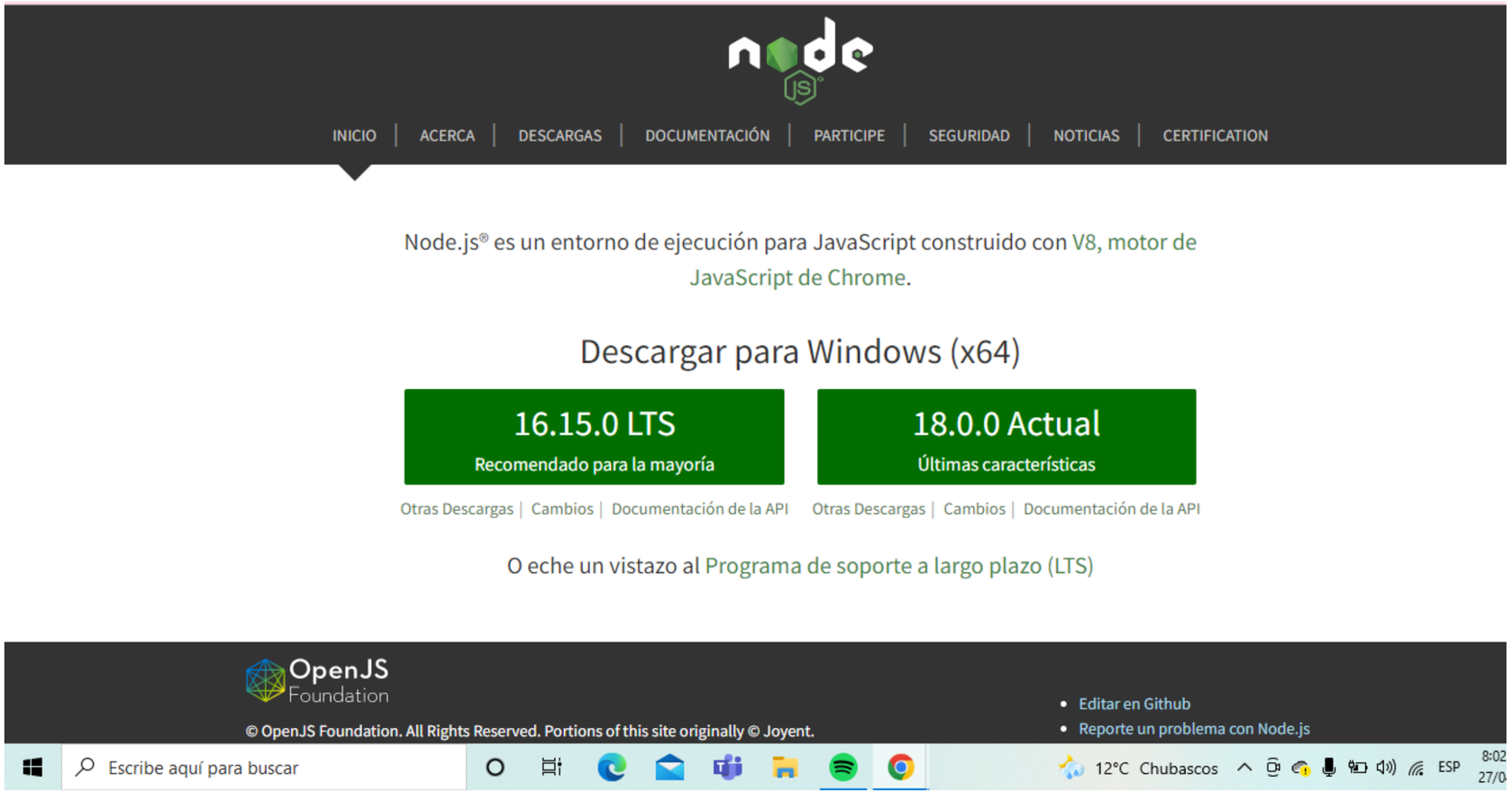
Nota: se recomienda firebase para subir el proyecto.

6. Configuración: BackEnd

Configurar entorno para BackEnd

Para configurar el entorno debes tener ya instalado las dependencias mencionadas en los requisitos de software.

6.1 Instalar node.js (desarrollador)



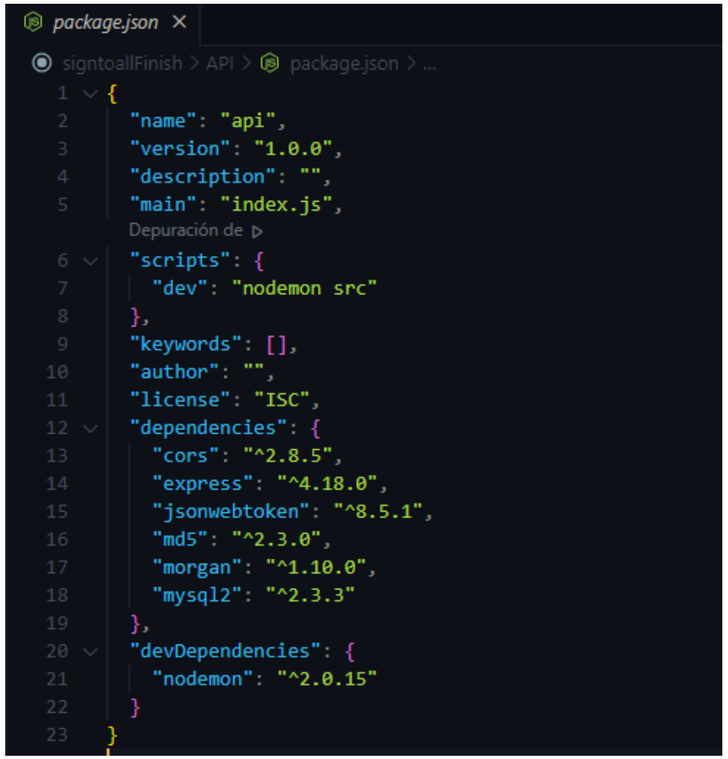
6.2 Descargar dependencias de la API

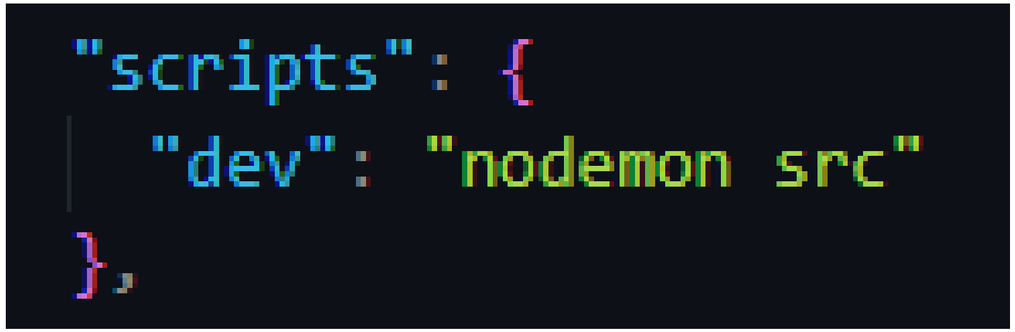
De igual manera que para el FrontEnd del proyecto, la API necesita las dependencias necesarias para funcionar, para esto ve a la consola posicionado en la carpeta del servidor y ejecuta el siguiente comando.

```
PS C:\Users\ACER\Desktop\Software\signtoallFinish\api> npm i
```

6.3 Configurar package.json

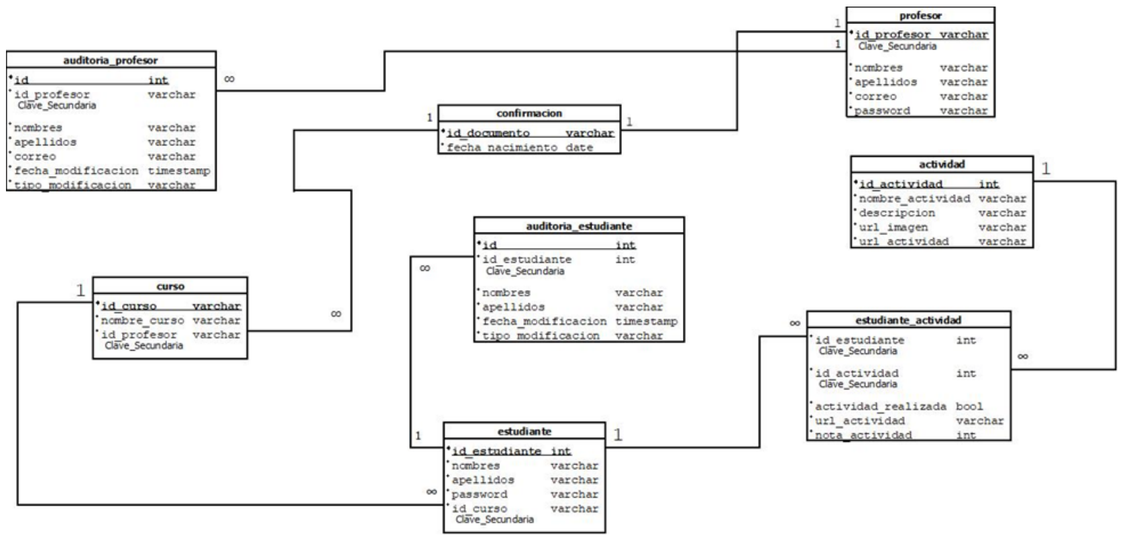
Ya con las dependencias descargadas se procede a configurar el archivo con los scripts necesarios para el desarrollo del proyecto, según sea el caso.

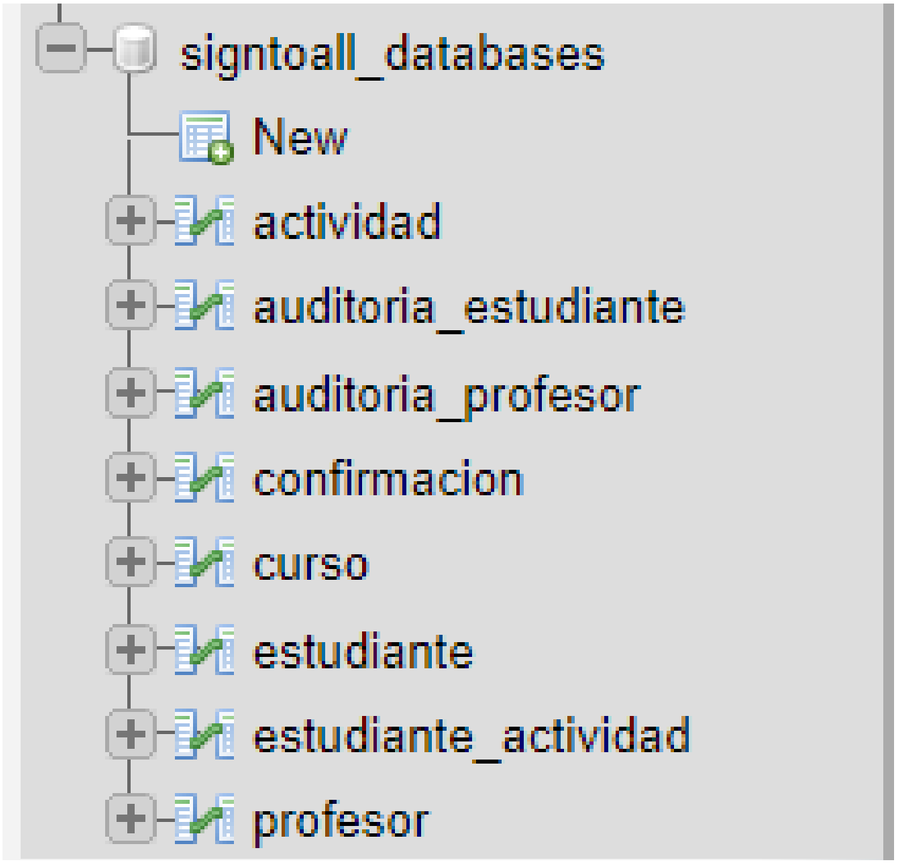




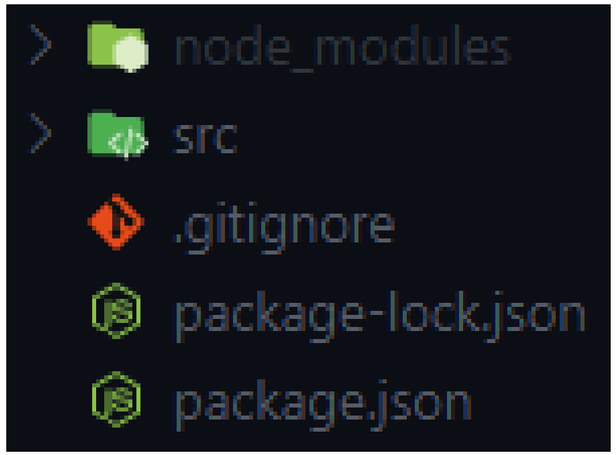
6.4 Bases de Datos

La base de datos se construyo con el gestor mysql, teniendo en cuenta el diseño estructural de la misma, dando en total 8 tablas que se relacionan entre si.



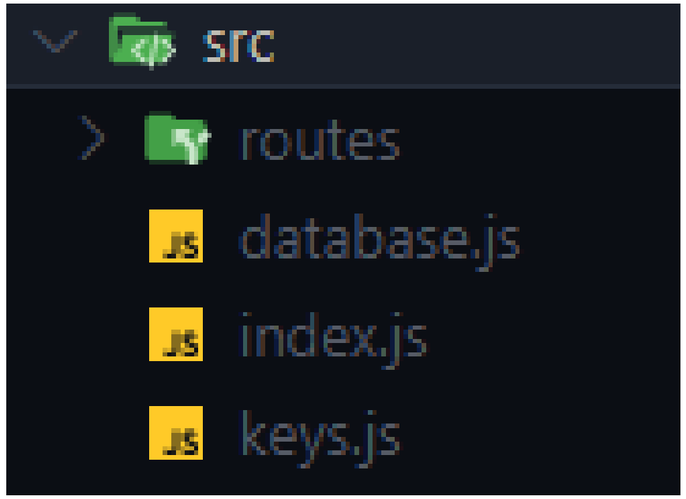


6.5 Estructura del proyecto. "./"



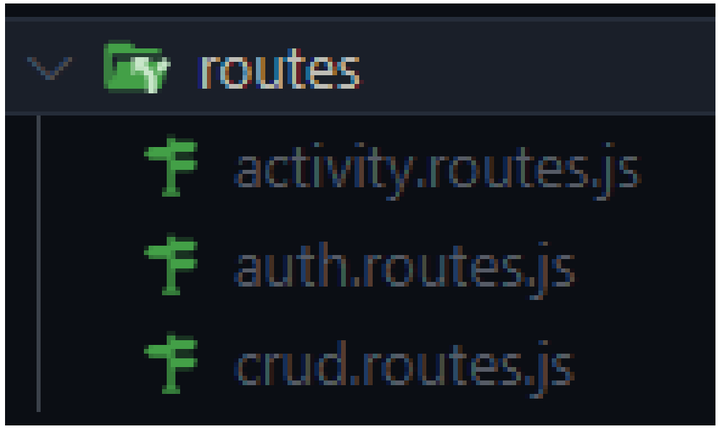
- node_modules: Se encuentran las dependencias de la API.
- src: El codigo de la API.
- package.json: Declaradas las dependencias de la API pero se recomienda npm para administrarlas.

6.5.1 Estructura del proyecto. "./src"



- routes: Rutas usadas para el desarrollo de la API.
- Archivos JS para la configuración del servidor y de la base de datos.

6.5.2 Estructura del proyecto. "./src/routes"



- activity: Las rutas especificas para las actividades
- auth: Las rutas para el registro e inicio de sesión tanto del profesor como del estudiante.
- crud: Las rutas para las funciones básicas que tiene el profesor con los datos del estudiante

6.6 API. "index.js"

En este archivo se configura el servidor utilizado para la ejecución de la API

- Se importan las dependencias y las rutas utilizadas en la API.
- Se configura el puerto.
- Se llaman los middlewares para el desarrollo del proyecto.
- Se configuran las rutas importadas.
- Se configura la ejecución del servidor.

API. "index.js"

```
const express = require("express");
const morgan = require("morgan");
const cors = require("cors");

const authRoutes = require("./routes/auth.routes");
const crudRoutes = require("./routes/crud.routes");
const activityRoutes = require("./routes/activity.routes");

const app = express();
```

```
//Configuración
app.set("port", process.env.PORT || 3000);

//   middlewares
app.use(morgan("dev"));
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

//Routes
app.use(authRoutes);
app.use(crudRoutes);
app.use(activityRoutes);

// Starting
app.listen(app.get("port"), () => {
  console.log("Server is in port", app.get("port"));
});
```

6.6.1 API. "keys.js"

En el archivo keys se configura el modulo de base de datos con el host, usuario, contraseña y nombre de esta.

Se define la llave de seguridad utilizada para el token de los usuarios.

```
module.exports = {
  database: {
    host: process.env.HOST || "localhost",
    user: process.env.USER || "user",
    password: process.env.PASSWORD || "",
    database: "signtoall",
  },
  jwtSecret: process.env.JWT_SECRET || "llave",
};
```

6.6.2 API. "database.js"

Se hace la configuración para conectar la base de datos importando lo que se definió en el archivo keys.

```
const mysql2 = require("mysql2/promise");

const { database } = require("./keys");

const pool = mysql2.createPool(database);
```

API. "database.js"

```
pool.getConnection((err, connection) => {
  if (err) {
    if (err.code === "PROTOCOL_CONNECTION_LOST") {
      console.error("Database connection was closed.");
    }
    if (err.code === "ER_CON_COUNT_ERROR") {
      console.error("Database has to many connections");
    }
    if (err.code === "ECONNREFUSED") {
      console.error("Database connection was refused");
    }
  }

  if (connection) connection.release();
  console.log("DB is Connected");

  return;
});

module.exports = pool;
```

6.6.3 API. "auth.routes.js"

- Se importan las dependencias necesarias para el desarrollo.
- Se definen el nombre de las rutas a utilizar.
- Se crean las funciones para el token de los usuarios.
- Se hace la lógica para cada una de las rutas.

API. "auth.routes.js"

```
const express = require("express");
const router = express.Router();
const md5 = require("md5");
const jwt = require("jsonwebtoken");

const pool = require("../database");
const { jwtSecret } = require("../keys");

function tokenProfesor(id_profesor, correo) {···
}

function tokenEstudiante(id_estudiante, nombre) {···
}
```

```
router.post("/registrarP", async (req, res) => {···
});

router.post("/iniciarSesionP", async (req, res) => {···
});

router.post("/registrarE", async (req, res) => {···
});

router.post("/iniciarSesionE", async (req, res) => {···
});
```

6.6.4 API. "crud.routes.js"

- Se importan las dependencias necesarias para el desarrollo.
- Se definen el nombre de las rutas a utilizar para el CRUD de la API.
- Se hace la lógica para cada una de las rutas dependiendo de la acción que se quiera realizar.

```
const express = require("express");
const router = express.Router();
const md5 = require("md5");

const pool = require("../database");

router.get("/lista", async (req, res) => {···
});

router.get("/listaEstudiante/:id", async (req, res) => {···
});

router.post("/crear", async (req, res) => {···
});

router.put("/actualizar/:id", async (req, res) => {···
});

router.delete("/eliminar/:id", async (req, res) => {···
});

module.exports = router;
```

6.6.5 API. "activity.routes.js"

- Se importan las dependencias necesarias para el desarrollo.
- Se definen el nombre de las rutas a utilizar para las actividades.
- Se hace la lógica para cada una de las rutas, dependiendo del id pedido en cada una.

API. "activity.routes.js"

```
const express = require("express");
const router = express.Router();
const pool = require("../database");

router.put("/actualizarActividad/:id_estudiante/:id_actividad", async (req, res) => {···
);

router.get("/verActividad/:id_estudiante/:id_actividad", async (req, res) => {···
});

router.get("/Actividades/:id_estudiante", async (req, res) => {···
});

module.exports = router;
```

6.7 API y BD

La base de datos se subió a alwaysdata.

La API se subió a Heroku.