Logotipo, nombre de la empresa

Descripción generada automáticamente

Manual del programador para el sistema de información

Apsi Col

UNIREMINGTON

**Edwin Rivera Ballesteros**

**Isabella Gómez Jurado**

Corporación Universitaria Remington

Programa de Ingeniería de Sistemas

Manizales, Colombia

2023

Manual del programador para el sistema de información

**Apsi Col**

**Versión 1.0**

**Edwin Rivera Ballesteros**

**Isabella Gómez Jurado**

Trabajo de investigación presentado como requisito parcial para optar al título de:

**Ingeniero de Sistemas**

Asesor:

M.Sc. Mauricio Mejía Lobo

Corporación Universitaria Remington

Programa de Ingeniería de Sistemas

Manizales, Colombia

2023

INDICE

[INTRODUCCION 4](#_Toc156571532)

[ARQUITECTURA DEL SISTEMA 6](#_Toc156571533)

[Diagrama de arquitectura 6](#_Toc156571534)

[Tecnologías utilizadas 7](#_Toc156571535)

[REQUISITOS DEL SISTEMA 8](#_Toc156571536)

[INSTALACION 8](#_Toc156571537)

[Requisitos previos 8](#_Toc156571538)

[Configuración Base de datos 8](#_Toc156571539)

[Configuración del backend – Express. 9](#_Toc156571540)

[Configuración del frontEnd – Angular. 10](#_Toc156571541)

[ESTRUCTURA DEL CODIGO 11](#_Toc156571542)

[Organización del proyecto 11](#_Toc156571543)

[Estructura backEnd 11](#_Toc156571544)

[Estructura frontEnd 13](#_Toc156571545)

[Convenciones de codificación FrontEnd 15](#_Toc156571546)

[Nombramiento de Archivos y Carpetas 15](#_Toc156571547)

[Nombramiento de Clases y Componentes 15](#_Toc156571548)

[Selector de componente 15](#_Toc156571549)

[Nombramiento de variables 15](#_Toc156571550)

[Indentación y espacios 16](#_Toc156571551)

[Gestión de rutas 16](#_Toc156571552)

[Convenciones de codificación FrontEnd 16](#_Toc156571553)

[CamelCase para Variables y Funciones 16](#_Toc156571554)

[Nombres Descriptivos 16](#_Toc156571555)

[Modularización 16](#_Toc156571556)

[Manejo de Promesas y Async/Await 16](#_Toc156571557)

[Middleware con Descriptores de Funciones 16](#_Toc156571558)

[Base de Datos 17](#_Toc156571559)

[Modelo de Datos 17](#_Toc156571560)

[Script de Creación de la Base de Datos 17](#_Toc156571561)

# INTRODUCCION

**El uso de tecnología en la agricultura no es un fenómeno reciente, ya que ha sido útil desde hace mucho tiempo como un instrumento mediador entre el hombre y su entorno. Según Saiz-Rubio et al. (2020), la función básica de la tecnología en la agricultura es contribuir sustancialmente a transformar la naturaleza para beneficio de la gente que vive del campo. Su uso se ha destinado en esencia para transformar lo tradicional a lo moderno.**

**La automatización y el uso de robots en la agricultura son ejemplos claros de cómo la tecnología ha sido utilizada en el campo para lograr cambios en la fuerza de trabajo y así sacar mayor provecho de esta. De acuerdo con la FAO (2018), los robots pueden ayudar a recolectar cultivos, clasificar productos agrícolas y realizar otras tareas en la granja, reduciendo la necesidad de trabajo manual y mejorando la eficiencia.**

**En la actualidad, el uso de tecnología en la agricultura se enfoca en suplir en mayor medida el uso de tierra y la mano de obra, implementando plataformas digitales que mejoran la conectividad entre agricultores, compradores y proveedores de servicios. Según informa la FAO (2020), estas plataformas pueden ayudar a los agricultores a encontrar mejores precios para sus productos y acceder a servicios como financiamiento y seguros agrícolas.**

**Hoy día se puede evidenciar la importancia que tienen las tecnologías de la información, y la manera en que herramientas como los sistemas puedan brindar a la sociedad y en especial a las empresas mayores facilidades para el manejo ordenado de los procesos que surgen en la producción diaria (García-Peñalvo, 2019). Con dichos sistemas de información se hace posible efectuar análisis crítico de los datos y en general de la información proporcionada por las transacciones que se den a través de este, para finalmente obtener resultados que servirán para tomar decisiones con base en información histórica y predictiva (Chen et al., 2019).**

**En general, las tecnologías están ayudando a los agricultores a ser más eficientes y a producir alimentos de manera más sostenible. Además, estas tecnologías pueden ayudar a mejorar la seguridad alimentaria al aumentar la cantidad y la calidad de los cultivos (Hajkowicz & Kandulu, 2015). Por lo tanto, con el presente documento, se desea formular, evaluar la viabilidad e implementar un proyecto de nivel tecnológico para el sistema de información APSI COL, cuyo objetivo es optimizar varios procesos, esto con el fin de lograr que el sistema de información facilite el registro de los sectores que requieran el servicio.**

# **ARQUITECTURA DEL SISTEMA**

## Diagrama de arquitectura

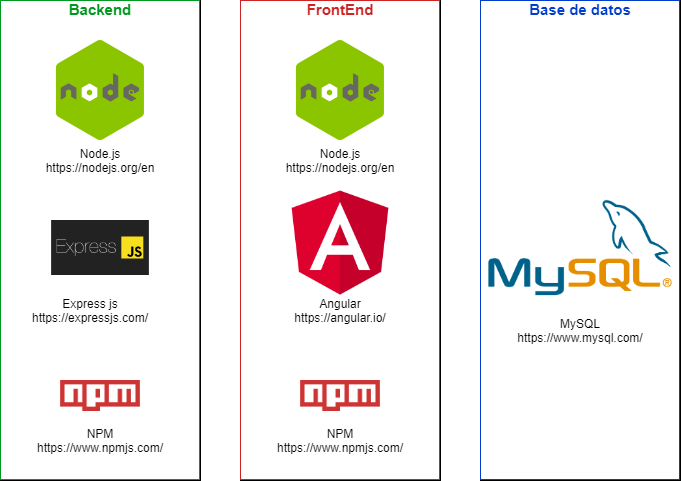
El siguiente diagrama ha sido creado para respaldar la arquitectura de componentes de la solución, diseñada para facilitar el uso del producto por parte de los usuarios.

El sistema ofrece acceso a tres tipos de usuarios: Administrador, Consultor y Granjero. Cada uno de ellos debe autenticarse en el sistema mediante datos de acceso, utilizando un sistema de autenticación basado en JWT a través del protocolo HTTP REST. Una vez autenticado, el usuario puede hacer uso del sistema y acceder a las pantallas correspondientes a su rol autorizado.



## Tecnologías utilizadas

Para el proyecto se utilizó Node.js el cual hace uso de las siguientes dependencias.



Para mayor información de las librerías y dependencias usadas en los proyectos se puede consultar el archivo **package.json** ubicada en la raíz del proyecto tanto del frontend como el backend.

# REQUISITOS DEL SISTEMA

* La página web podrá ser visualizada en los siguientes navegadores “Google Chrome [Versión. 120], Mozilla Firefox [Versión. 121], Opera [Versión. 120], safari [Versión. 90], Microsoft Edge [Versión.120].
* La aplicación contara con un servicio las 24 horas al día los 7 días a la semana.
* La aplicación contara con un requerimiento de datos mínimo (mientras no haga uso de transferencia multimedia.)

# INSTALACION

Esta guía proporciona instrucciones detalladas sobre cómo instalar y configurar un entorno de desarrollo para un proyecto web con Node.js en el frontend utilizando Angular y en el backend con Express.

## Requisitos previos

Antes de comenzar, asegúrate de tener instalados los siguientes componentes en tu sistema:

1. Node JS: [Versión V18.xx.x](https://nodejs.org/download/release/v18.19.0/)
2. Angular CLI: [Versión 15.x.x](https://v15.angular.io/docs)
3. Express: [Versión 4.18.2](https://www.npmjs.com/package/express/v/4.18.2)
4. NPM: [Versión 8.xx.x](https://docs.npmjs.com/cli/v8/commands/npm)
5. MySQL: [Versión 8.1.12](https://www.mysql.com/downloads/)

## Configuración Base de datos

1. Visite la Página oficial de mysql/downloads o descargue el [XAMPP](https://www.apachefriends.org/es/download.html) en su versión 8 o superior.
2. Luego de tener instalado la base de datos, se procede con la creación de la base de datos del proyecto y las tablas.
3. Ubíquese en la raíz del proyecto de backend, donde encontrara el script para crear la base de datos: “Apsicol-back-end\BBDD\apsicol.sql”.
4. Puede ejecutar el Script de la base de datos en cualquier gestor de base de datos como [DBeaver](https://dbeaver.io/download/) o el de su preferencia.
5. Ejecute el script para restaurar la estructura y los datos base de la base de datos de ser necesario.
6. Verifique que la estructura de la base de datos coincida con la estructura esperada.

## Configuración del backend – Express.

1. Cree el directorio en la ubicación donde requiera que este el proyecto instalado.
   1. Ejemplo: C:\Users\Public\Documents\apsicol
   2. mkdir Apsicol-back-end
   3. cd Apsicol-back-end
2. Descargue el proyecto o clone el proyecto a la carpeta seleccionada desde el GitHub.
3. Abra el proyecto con su editor de código preferido, se recomienda usar [VSCode.](https://code.visualstudio.com/download)
4. Abra una terminal dentro del editor e instale todos los paquetes necesarios para que el proyecto se pueda ejecutar correctamente.
   1. npm install
5. Luego de terminar la instalación de los paquetes proceda con la ejecución del servidor.
   1. npm run dev – ambiente de desarrollo.
   2. npm run start – ambiente de producción.
   3. npm run build – Construir archivos de instalación del proyecto.

## Configuración del frontEnd – Angular.

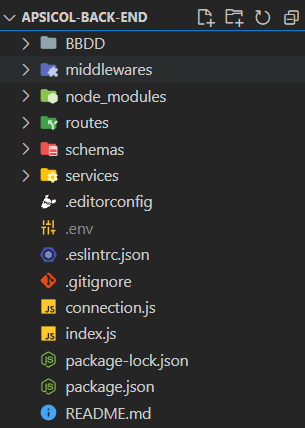
1. Cree el directorio en la ubicación donde requiera que este el proyecto instalado.
   1. Ejemplo: C:\Users\Public\Documents\apsicol
   2. mkdir Apsicol-front-end
   3. cd Apsicol- front-end
2. Descargue el proyecto o clone el proyecto a la carpeta seleccionada desde el GitHub.
3. Abra el proyecto con su editor de código preferido, se recomienda usar [VSCode.](https://code.visualstudio.com/download)
4. Abra una terminal dentro del editor e instale todos los paquetes necesarios para que el proyecto se pueda ejecutar correctamente.
   1. npm install
5. Luego de terminar la instalación de los paquetes proceda con la ejecución del servidor.
   1. ng serve – ambiente de desarrollo.
   2. ng build – Construir archivos de instalación del proyecto.

**Nota:** Para realizar el despliegue de la aplicación se debe revisar la documentación pertinente del sitio como por ejemplo [vercel](https://vercel.com/new) el cual permite realizar el despliegue realizando algunos ajustes al proyecto y se despliega desde GitHub, Bitbucket, entre otros.

# ESTRUCTURA DEL CODIGO

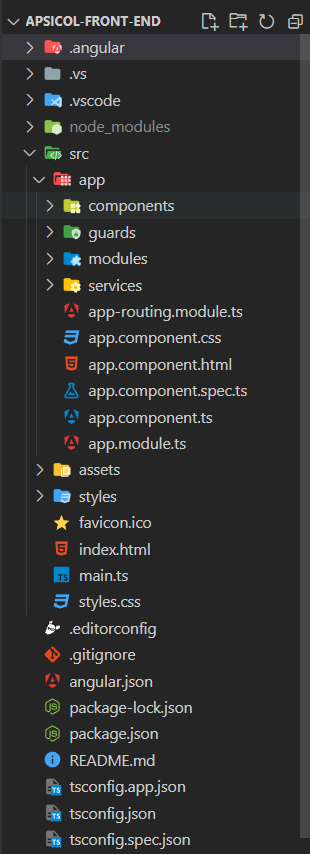
## Organización del proyecto

### Estructura backEnd



* **/BBDD :** Este directorio contiene el script de la base de datos.
* **/middlewaers:** Aquí se almacenan los objetos encargados de acceder a la request – response y a un siguiente middleware de ser necesario “Proceso de solicitud HTTP”.
* **/node\_modules:** Contiene todos los módulos o librerías que se están usando en la aplicación.
* **/routes:** Este directorio contiene todas las rutas de acceso a los endpoint.
* **/schemas:** Este directorio contiene todas las validaciones de las request entrantes a los endpoint.
* **/services:** Este directorio contiene todos los servicios de la aplicación.
* **.editorconfig:** Contiene las contiguraciones del IDE.
* **.env:** Contiene todas las variables de entorno.
* **.eslintrc:** Herramienta para revisión de código.
* **.gitignore:** Directorio en el cual se indica los archivos que se ignorar al subir los cambios al gestor de versiones GitHub.
* **Connection.js:** Archivo que contiene la conexión a la base de datos.
* **Index.js:** Archivo de configuración que contiene todos los parámetros y configuraciones del servidor.
* **Package.\* :** Archivos que contienen las configuraciones y las dependencias de un proyecto de nodeJS.
* **READMI.md :** Archivo donde se almacena información pertinente al proyecto.

### Estructura frontEnd



* **.angular - .vs - .vscode:** Contienen configuraciones de los IDEs.
* **node\_modules:** Este directorio contiene todos los componentes y/o dependencias del proyecto de angular.
* **src/app:** Directorio donde se almacena todos los directorios de la implementación del proyecto.
* **Components:** Directorio que contiene todos los componentes del proyecto.
  + **Component/.css:** Contiene los estilos del componente.
  + **Component/.html:** Contiene las etiquetas html del componente o la vista del componente.
  + **Component/.spec.ts:** Archivo usado para implementar test o pruebas a los componentes.
  + **Component/.ts:** Archivo de TypeScript que contiene toda la lógica o funcionalidades del componente.
* **Guards:** Directorio que contiene todos los archivos que se encargan de aplicar la capa de seguridad a el proyecto.
* **Modules:** Directorio que contiene módulos o implementaciones que se pueden usar en el proyecto.
* **Services:** Contiene todos los servicios del proyecto los cuales se comunican mediante protocolo HTTP con la API.
* **app-routing.module.ts:** Archivo que contiene las rutas del proyecto.
* **App-component.ts:** Archivo base donde se renderizan los componentes que se crean.
* **App-component.spec.ts:** Archivo que realiza pruebas al app-component.ts.
* **App-component.css:** Archivo que aplica estilos al componente base.
* **App-module.ts:** Archivo que contiene la configuración o preparación del modulo principal o base del proyecto.
* **Assets:** directorio que contiene contenido estático que se requiera incluir en el proyecto como: imágenes, Json, videos, etc.
* **Styles:** Contiene archivos de css, scss u otro archivo encargado de aplicar estilos al proyecto.
* **Favicon.ico:** Icono del proyecto que se visualiza en la pestaña del navegador.
* **Index.html:** archivo principal del proyecto, donde se renderiza el app-component.ts y todos los subcomponentes.
* **Main.ts:** Archivo que contiene las configuraciones globales del proyecto.
* **Style.css:** Archivo que contiene los estilos globales del proyecto.
* **.editorconfig:** Contiene las contiguraciones del IDE.
* **.gitignore:** Directorio en el cual se indica los archivos que se ignorar al subir los cambios al gestor de versiones GitHub.
* **Angular.json:** Contiene las configuraciones del angular.
* **Package.\* :** Archivos que contienen las configuraciones y las dependencias de un proyecto de nodeJS.
* **READMI.md :** Archivo donde se almacena información pertinente al proyecto.
* **Tsconfig.\*:** Contiene las configuraciones necesarias para el uso de TypeScript en angular.

## Convenciones de codificación FrontEnd

El proyecto está realizado siguiendo el estándar ES6 o ECMAScript 6 de acuerdo con las indicaciones contenidas en el documento ECMA-262 proporcionado por Ecma International (European Computer Manufacturers Association).

### Nombramiento de Archivos y Carpetas

* Los nombres de archivos y carpetas son en minúsculas.
* Se utiliza la notación de kebab-case para nombrar archivos y carpetas. Por ejemplo, mi-componente.component.ts o mi-servicio.service.ts.

### Nombramiento de Clases y Componentes

* Los nombres de clases y componentes usan la notación de camelCase. Por ejemplo, MiComponente.
* Los nombres de archivos y carpetas para componentes coinciden con el nombre de la clase.

### Selector de componente

* El selector de un componente es en minúsculas y en notación de kebab-case. Por ejemplo, <app-mi-componente></app-mi-componente>.

### Nombramiento de variables

* Se usa la notación de camelCase para nombrar variables y métodos. Por ejemplo, miVariable o miMetodo().

### Indentación y espacios

* Se aplica una sangría de 2 espacios.
* Se utiliza un espacio después de los dos puntos en las propiedades de los objetos.

### Gestión de rutas

Las rutas se definen utilizando el archivo app-routing-module y se configuran en un array de objetos.

Se utiliza la notación de PascalCase para nombrar los componentes de las rutas.

## Convenciones de codificación FrontEnd

El proyecto está realizado siguiendo el estándar ES6 o ECMAScript 6 de acuerdo con las indicaciones contenidas en el documento ECMA-262 proporcionado por Ecma International (European Computer Manufacturers Association).

### CamelCase para Variables y Funciones

* Se aplica el uso de camelCase para nombres de variables y funciones. Por ejemplo: miVariable, miFuncion().

### Nombres Descriptivos

* Se utiliza nombres descriptivos para las variables y funciones. Esto mejora la legibilidad del código.

### Modularización

* Divide tu código en módulos y utiliza la importación/exportación de módulos de ES6 para mantener un código más organizado.

### Manejo de Promesas y Async/Await

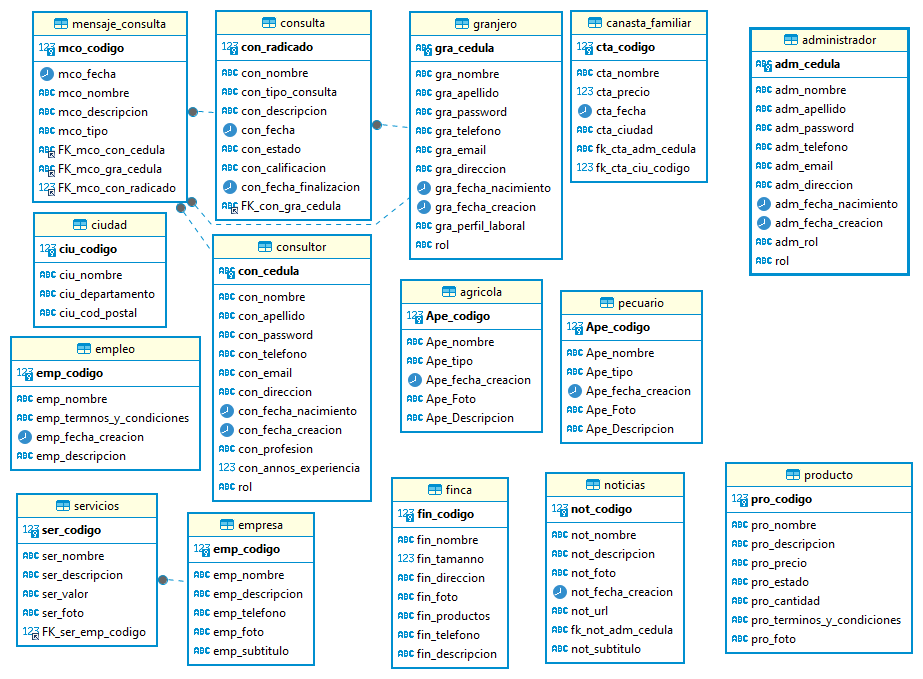
* Se utiliza Promesas y Async/Await para gestionar el flujo asíncrono de manera más legible.

### Middleware con Descriptores de Funciones

* Se utilizan middleware al nombrar las funciones de manera descriptiva.

# Base de Datos

## Modelo de Datos



## Script de Creación de la Base de Datos

Los Scripts de creación de base de datos están ubicados en el backend: \Apsicol-back-end\BBDD\apsicol.sql.