# MANUAL TÉCNICO

## DESARROLLO WEB


Ciudad de Guatemala, Guatemala

**202006681 KEVIN HAROLDO**

ALBIZURES SIRÍN

**202106208 DWIGHT FERNANDO GABRIEL**

CHINCHILLA HERNÁNDEZ

## Datos de desarrollo

**Interprete utilizado**: Javascript

**Herramienta para entorno de ejecución:** Node.js

**Editor de código fuente utilizado:** Visual Studio Code v1.76.2

**Sistema de gestión de datos:** MySQL

**Herramientas de apoyo:** MySQL Workbench

**Framework usado para el desarrollo web:** Angular

## Objetivos

**Objetivo general**: Proporcionar la instrumentalización necesaria de los servicios, métodos y tablas de datos para la implementación de un sitio web académico para llevar una gestión de recomendaciones de cursos.

**Objetivos específicos**:

1. Hacer uso de un sistema de gestión de datos para simplificar los procesos de gestión de información de los cursos, de las publicaciones y de los perfiles de usuario.
2. Utilizar un entorno de ejecución de tiempo real, para poder mantener una conexión constante entre el servidor y el frontend el cuál se encargará de la gestión visual de la aplicación.

## Descripción General:

El programa consiste en el desarrollo de una página web la cual le proporcionara a los estudiantes una herramienta para consultas académicas sobre los cursos y los catedráticos que las imparten, con las posibilidades de poder interactuar entre perfiles y poder establecer cada perfil junto a sus cursos aprobados y acumulación de créditos, también la creación de publicaciones con la posibilidad de almacenar comentarios de manera individual.

## Paradigma utilizado

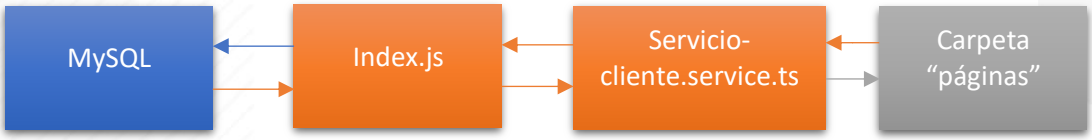
**Modelo Relacional:** este paradigma nos proporciona la posibilidad de almacenar la información en SQL de manera que se tiene una llave primaria la cual nos dará acceso a una fila correspondida donde se tiene la información de dicha llave en forma de un conjunto de elementos facilitando así la administración de los datos.

## Lógica del programa:

- **Carpetas base:** el programa está organizado de forma que se tiene la parte del backend y frontend en carpetas individuales.


  La carpeta backend contiene todas las librerías para que nuestro entorno de ejecución node.js pueda funcionar. Mientras que nuestra carpeta frontend contendrá todos los archivos relacionados con nuestro framework Angular para el desarrollo de la aplicación web.

- **Esquema:**




- **Base de datos:** la base de datos funciona a través de tres tablas las cuales fueron creadas por medio de la herramienta SQL workbench, cada tabla se compone de los siguientes campos:

*Tabla 1. Usuarios*

| | |
|---|---|
| *Registro_A* | Llave primaria, es un int |
| *Nombre* | Cadena |
| *Apellidos* | Cadena |
| *Pass* | Cadena |
| *Correo* | Cadena |

*Tabla 2. Comentarios*

| | |
|---|---|
| *Creador* | Llave primaria, es un int |
| *Id_Creador* | Cadena |
| *Apellidos* | Cadena |
| *Pass* | Cadena |
| *Correo* | Cadena |

*Tabla 3. Curso*

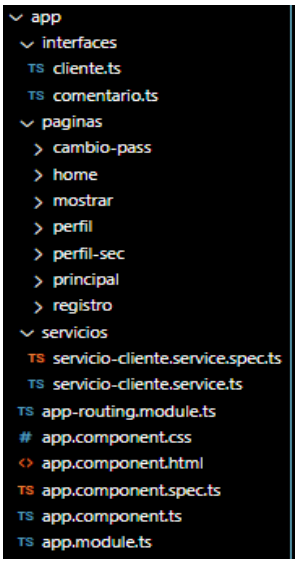
| | |
|---|---|
| *Id_Curso* | Llave primaria, es un int |
| *Curso* | Cadena |
| *Creditos* | Cadena |
| *Id_Creador* | Cadena |

- **Index.js**: se encuentra en la carpeta backend y se centra en mantener las conexiones entre nuestro programa y nuestra base de datos que en este caso es SQL.

- **Querys:** La manera en la que nuestro backend se comunica con nuestra base de datos es a través de las llamadas query's, cada query posee su propia funcionalidad, ahora se listarán los métodos utilizados y cuál fue el query usado en cada uno así como en que tabla fue aplicado su función:

    - **Iniciar sesión:** SELECT en la tabla usuarios.
    - **Crear Usuario:** INSERT en la tabla de usuarios.
    - **Modificar Pass:** UPDATE en usuarios.
    - **Mostrar usuarios:** SELECT en usuarios.
    - **Crear comentario:** INSERT en la tabla comentarios.
    - **Comentarios:** SELECT en la tabla comentarios.
    - **Crear Curso:** INSERT en la tabla cursos.
    - **Curso:** SELECT en la tabla cursos.
    - **Modificar usuario:** SELECT en usuarios.
    - **Buscar Usuario:** SELECT en usuarios.


- **Frontend/src/app/:** En esta ruta se encuentran todos los archivos creados para desarrollar la parte web de la aplicación, pero para poder usarla primero se necesitó instalar el framework de angular en la carpeta de frontend, los archivos creados son los siguientes:




- **Interfaces:** en está carpeta se almacenaron algunas clases que funcionarían como objetos, está funcionalidad solo fue utilizada en dos ocasiones pasan casi desapercibidas.

```
export interface Cliente {

    Registro_A: number;
    Nombre: string;
    Apellidos: string;
    Pass: string;
    Correo: string;

}
```

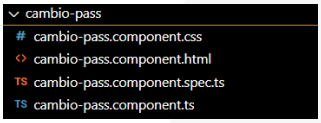
```
export interface Comentario {
    Creador: string;
    Id_Creador: number;
    Curso: string;
    Catedratico: string;
    Comentario: string;
    Subcomentarios: any;
}
```

- **Páginas:** se encontrarán en carpetas individuales, cada página separado en archivos ts, html y css. Para poder crear una carpeta de componentes para una página html nueva se necesita del siguiente comando:

  ng generate component nombre-del-componente

  Con está información se crearon las siguientes carpetas:

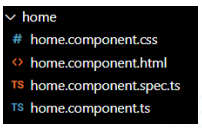
  - Cambio-Pass: en esta carpeta se maneja el html para poder cambiar la contraseña por si se llegara a olvidar.

    Método implementado en archivo ts:

    - ModificarPass(): a través del registro y del correo, luego de verificarlos en la base de datos, se procede a aceptar el cambio de contraseña.

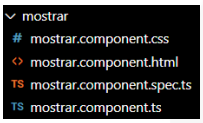
- Home: la carpeta se centra en almacenar el html del inicio de sesión.




  Método implementado en archivo ts:

  - infoLogin(): a través del registro y de la contraseña, luego de ser verificada en la base de datos, otorgará permiso para poder ingresar al portal en la página “principal”.

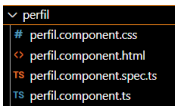
- Mostrar: su función es la de mostrarle al usuario cuales son las publicaciones creadas dentro del portal.




  Método implementado en archivo ts:

  - buscar(dato:number): a través del nombre del catedrático o del curso, se realizará una búsqueda de las publicaciones con alguno de esos parámetros y los mostrará en la página.

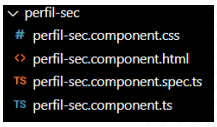
- Perfil: página que muestra la información del usuario que acaba de ingresar sesión.




  Método implementado en archivo ts:

  - modificarCliente(): al habilitar el editar los campos del perfil de usuario y luego al darle al botón de actualizar, el método comenzará a modificar la base de datos.

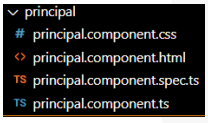
- Perfil-sec: página que muestra la información de perfil del usuario al cuál se buscó o se encontró en las publicaciones del portal.




  Método implementado en archivo ts:

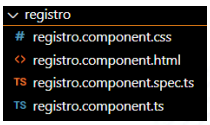
  - Devolver2(): retorna la información del usuario al cuál se ingreso a su perfil.

- Principal: en está carpeta se cumple con la función de poder crear publicaciones como también el crear cursos.




  Método implementado en archivo ts:

  - crearComentario(): en este método se manda a llamar otro método get al cuál se le enviaran los datos de los campos requeridos para crear un comentario.
  - crearCurso(): se reunirá los datos requeridos en los campos del apartado crear curso para poder ingresarlos a la base de datos.




- Registro: su función es la de solicitarle al usuario sus datos para poder crear un perfil nuevo dentro del portal.

  Método implementado en archivo ts:

  - crearCliente(): en este método se manda a llamar otro método get al cuál se le enviaran los datos de los campos requeridos para crear un usuario nuevo.

- **Servicios:** se almacenaron los archivos que contenían los métodos para poder realizar las llamadas post y get.


```
ConsultarClientes(): Observable <any>{···
}

CrearClientes(datos:Cliente): Observable <any>{···
}

Login(datos:any): Observable <any>{···
}

CrearComentarios(datos:Comentario): Observable <any>{···
}

ConsultarComentarios(): Observable <any>{···
}

CrearCursos(datos:any): Observable <any>{···
}

ConsultarCursos(): Observable <any>{···
}
```

```
ModificarClientes(datos:any): Observable <any>{···
}

ModificarPass(datos:any): Observable <any>{···
}

BuscarUsuario(datos:any): Observable <any>{···
}

public ingresarAplicativo(obj:any):boolean{···
}

encapsular(usuario:any){···
}

devolver(){···
}

encapsular2(usuario2:any){···
}

devolver2(){···
}
```