

**Ministerio de Tecnologías de la Información  
y las Comunicaciones de Colombia  
Cimetrica Tecnalia**

**Plataforma de tutorías en línea Tutor Shop**

**Manual del Sistema**

**Por**

**Bryan Elian Peña Jaimes  
Camilo Lombana Cruz  
Edgar Alexander Díaz Murillo  
Elkin Saúl Torres Díaz**

**Facilitador  
Edgar Morillo**

**Bogotá, 16 de julio de 2024**

## Contenido

Introducción .....	5
1. Requisitos del Sistema. ....	6
1.1. Requisitos Mínimos. ....	6
2. Instrucciones de Instalación. ....	6
3. Descripción general de la Interfaz de usuario.....	10
4. Diseño técnico del sistema de información.....	16
4.1. Esquema o modelo de requerimientos.....	16
4.1.1. Registro de usuarios.....	16
4.1.2. Búsqueda de tutores. ....	16
4.1.3. Reserva de Sesiones de Tutoría.....	16
4.1.4. Gestión de disponibilidad. ....	16
4.1.5. Autenticación de usuarios. ....	17
4.1.6. Desplegar la interfaz. ....	17
4.2. Componentes y estándares de la aplicación. ....	17
4.2.1. Node(V18.16). ....	17
4.2.2. Express (V4.19.2). ....	18
4.2.3. ReactJS. ....	18
4.2.4. Sequelize (V6.37.3). ....	19
4.2.5. Postgresql(V15). ....	19
4.2.6. Npm(9.6.6). ....	19
4.2.7. Postman.....	19
4.2.8. GitHub. ....	19
4.3. Modelo de datos.....	20
4.3.1. Modelo entidad-relación. ....	20
4.3.2. Diccionario de datos. ....	21
4.3.2.1. Entidad Users.....	21
4.3.2.2. Entidad People. ....	21
4.3.2.3. Entidad Students.....	22

4.3.2.4.	Entidad Tutors. ....	23
4.3.2.5.	Entidad TutorSubjects.....	23
4.3.2.6.	Entidad Subjects. ....	23
4.3.2.7.	Entidad Reservations. ....	24
4.3.2.8.	ReservationTypes .....	25
4.4.	Creación de servicios en el back-end.....	26
4.4.1.	citiesController.js.....	26
4.4.2.	genericController.js.....	26
4.4.3.	reservationController.js.....	27
4.4.4.	studentController.js. ....	27
4.4.5.	subjectController.js.....	28
4.4.6.	tutorController.js. ....	28
4.4.7.	tutorSubjectController.js. ....	29
4.4.8.	userController.js. ....	29
4.5.	Creación de las rutas HTTP en el back-end.....	30
4.5.1.	Users. ....	30
4.5.2.	Students. ....	30
4.5.3.	Tutors.....	32
4.5.4.	Reservations.....	33
4.5.5.	TutorSubjects. ....	35
4.5.6.	Subjects. ....	37
4.5.7.	Generics. ....	38
4.5.8.	Cities. ....	39
4.6.	Componentes del front-end. ....	39
4.6.1.	studentFormSlice. ....	39
4.6.2.	userSlice. ....	40
4.6.3.	Login.....	40
4.6.4.	Logout.....	40
4.6.5.	Schemas.....	41

4.6.6.	Services. ....	41
4.6.7.	Store. ....	41
4.6.8.	Student. ....	41
4.6.9.	Tutor.....	41
4.6.10.	App. ....	42
4.6.11.	Index.....	42
5.	Despliegue de la aplicación en Render.....	42
6.	Diagrama de casos de uso. ....	46
7.	Organización de componentes.....	47
8.	Términos y definiciones. ....	48
9.	Contacto de soporte.....	51

## Introducción

Dentro de los programas de capacitación en habilidades digitales liderados por el Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia y Cimetria Tecnalía se encuentra *Talento Tech*, un programa de formación que busca mediante un entrenamiento intensivo bajo la metodología de bootcamps preparar a los nuevos talentos del país en habilidades digitales avanzadas y desarrollo de proyectos de tecnología como es el caso de nuestra línea de *Desarrollo Web Full Stack*, capacitándolos para ingresar al mercado laboral tecnológico.

Cumpliendo con el Plan de Estudio del nivel intermedio de desarrollo Web Full Stack hemos diseñado la Plataforma de Tutorías en Línea ‘Tutor Shop’ con el fin de conectar a estudiantes con tutores calificados en diferentes materias y niveles educativos, donde los estudiantes pueden buscar y reservar sesiones de tutorías en las modalidades presenciales y virtuales, entre tanto los tutores pueden gestionar su disponibilidad y brindar las tutorías de acuerdo con su materia de elección.

Este manual técnico de la solución de software de la Plataforma de Tutorías en Línea ‘Tutor Shop’ tiene como propósito ilustrar sobre la definición, diseño, organización y estructura del sistema o solución al personal encargado de mantener la prestación del servicio o servicios ofrecidos por el sistema o solución, estos lectores incluyen desarrolladores, arquitectos, ingenieros de pruebas etc.

# 1. Requisitos del Sistema.

## 1.1. Requisitos Mínimos.

A continuación, se detalla la información de los requisitos mínimos utilizados por el cliente para ejecutar la aplicación web:

- Sistema Operativo: La aplicación web puede ser utilizada en cualquier cliente mediante el navegador de su preferencia indiferente del sistema operativo local que se esté usando para su operación.
- Procesador: Intel Core Celeron.
- Memoria RAM: 1GB.
- Disco Duro: 1GB.
- Resolución de pantalla: 1280 x 720 pixeles.
- Periféricos: Teclado, ratón.

## 2. Instrucciones de Instalación.

Con el fin de ejecutar la aplicación desde la computadora local, como pre-requisitos debemos tener instalado el siguiente software para su funcionamiento:

- Node.js: <https://nodejs.org/es/>
- Postgres: <https://www.postgresql.org/>

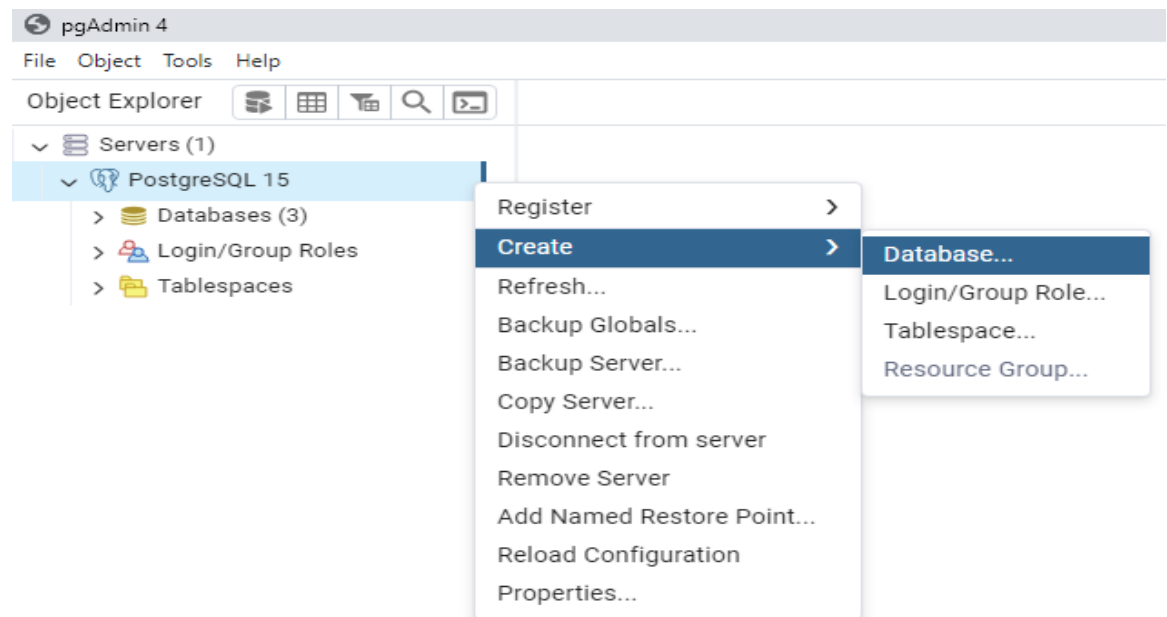
## 2.1. Configurar la base de datos de postgresQL.

2.1.1. En Windows en la opción 'Buscar' digitar 'pgAdmin4' para abrir el gestor de base de datos de postgresQL.

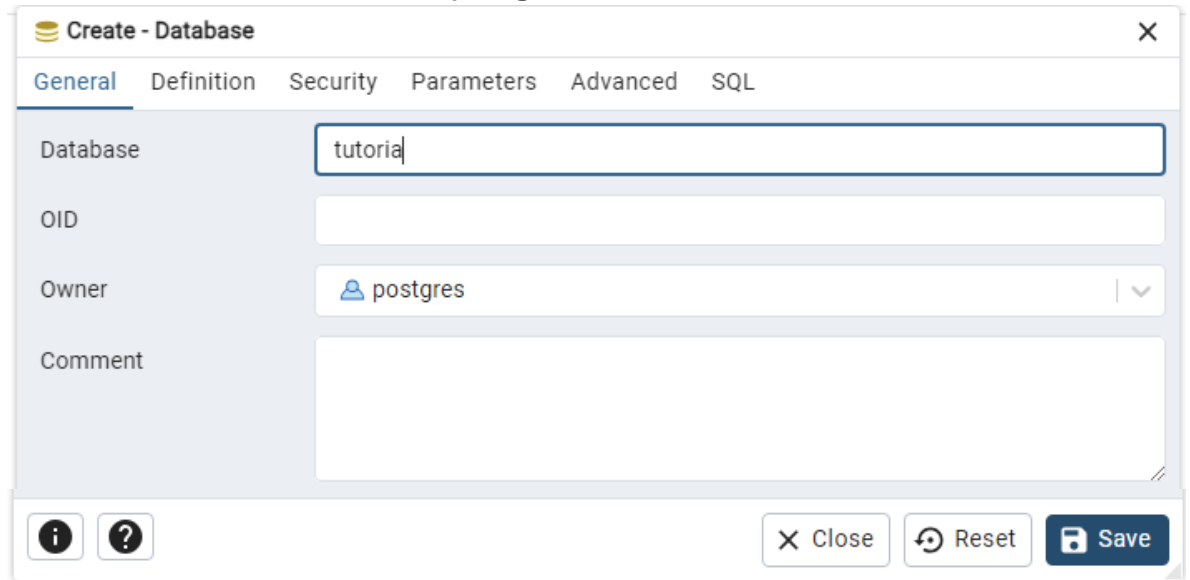
2.1.2. Click en 'Servers' y luego digitar el password que digitó en el momento de la instalación y luego click en Ok.



2.1.3. Click derecho sobre posgreSQL15, click en 'Create' y luego click en Database...

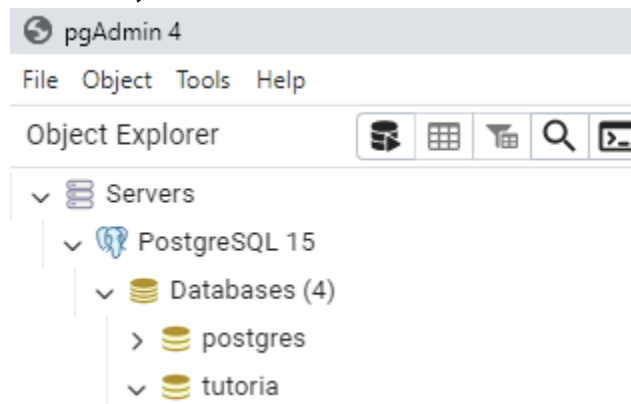


- 2.1.4. En el campo 'Database' digite el nombre de la base de datos que desea crear y luego click en el botón 'Save'.  
Hecho los pasos anteriores se crea la base de datos 'tutoria' en el motor posgreSQL.

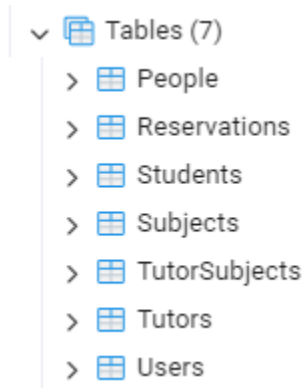


The screenshot shows the 'Create - Database' window in pgAdmin 4. The 'General' tab is selected. The 'Database' field is filled with 'tutoria'. The 'Owner' field shows 'postgres'. The 'Comment' field is empty. At the bottom, there are buttons for 'Close', 'Reset', and 'Save'.

- 2.1.5. Nota: Las tablas de la base de datos 'tutoria' son creadas en forma automática luego de ser ejecutada la aplicación desde el back, mediante el comando: `node app.js`.







2.2. El código resultado del desarrollo de la aplicación web reside en el repositorio de Github:

<https://github.com/darkelian/Talento-Tech.git>

2.3. Si se desea instalar la aplicación en su computadora local sistema Windows es preciso realizar los siguientes pasos:

2.4. En Windows en la opción 'Buscar' digitar la palabra cmd y click en la opción 'Símbolo del sistema', luego crear una carpeta en el directorio deseado con el comando mkdir:

```
mkdir nombre_carpeta
```

2.5. Ubicado en la carpeta local creada, clonar el repositorio remoto en el local por medio de la instrucción:

```
git clone https://github.com/darkelian/Talento-Tech.git
```

2.6. Haciendo uso del editor de código Visual Estudio Code abrir la carpeta donde clonó la aplicación.

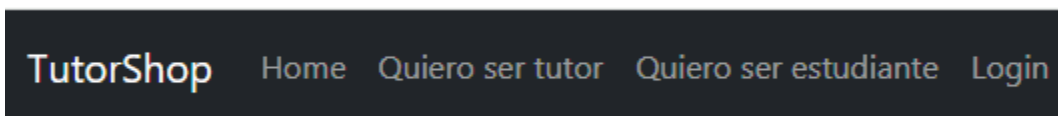
2.7. En la opción 'View' click en 'Terminal' y luego desde la línea de comando digitar: npm install, para la instalación de nuestras dependencias y la carpeta node\_modules.

- 2.8. Luego cambiarse a la carpeta `app\src` donde reside el archivo `app.js` punto de partida de ejecución de la aplicación en el back. Para iniciar la aplicación en el back, digitar comando:  
`node app.js`
- 2.9. A continuación, cambiarse a la carpeta front-end y digitar el comando `npm start` para iniciar la ejecución de la aplicación en el front y visualizar en el navegador la interfaz de usuario.

### 3. Descripción general de la Interfaz de usuario.

La aplicación web contiene la siguiente interfaz de usuario y hace uso del recurso SPA (Single Page Application) o aplicación de una sola página, la cual carga una única página inicial desde el servidor y luego actualiza dinámicamente su contenido mediante el uso de componentes de React y mediante las interacciones del usuario, sin necesidad de cargar páginas adicionales del servidor.

#### 3.1. Menú principal.



El menú principal contiene las siguientes opciones:

**3.1.1. Home:** Contiene el Dashboard de la aplicación.

**3.1.2. Quiero ser tutor:** Contiene las opciones de:

### 3.1.2.1.Registro: Permite al profesor registrarse en el sistema mediante el diligenciamiento del formulario:

**TutorShop** [Home](#) [Tutores ▾](#) [Estudiante ▾](#) [Login](#)

## Regístrate como profesor

Completa el formulario para unirte como tutor a nuestra plataforma educativa

Nombres

Lina María

Apellidos

Escobar Tamayo

Fecha de nacimiento

28/05/1998

Género

Femenino

Tipo de documento

Cédula de ciudadanía

No. de documento

51678098

Departamento

Antioquia

Ciudad

Medellín

No. de celular

3112909876

Correo electrónico

lina.escobar@gmail.com

Profesión

Mercadotecnia

Contraseña

\*\*\*\*\*

Confirmar contraseña

\*\*\*\*\*

Enviar

### 3.1.2.2.Dashboard: Permite al profesor registrar su disponibilidad y aceptar o rechazar solicitudes de tutoría.

## Dashboard del tutor

### Perfil del tutor



#### Sobre mí

Docente en el área de las Ciencias Naturales y Sociales

Pedro Enrique  
Salamanca Rodríguez

Tutor desde 1995-09-23

### Tutorías aceptadas

### Solicitudes de tutorías

### Materias dictadas

[Agregar](#)

Geografía

Eliminar

Historia

Eliminar

### 3.1.3. Quiero ser estudiante: Contiene las opciones de:

#### 3.1.3.1.Registro: Permite al estudiante registrarse en el sistema mediante el diligenciamiento del formulario:

## Regístrate como estudiante

Completa el formulario para unirte como estudiante a nuestra plataforma educativa

Nombres

Gustavo

Apellidos

Valderrama Sabogal

Tipo de documento

Cédula de ciudadanía

Número de documento

1025687543

Número de documento

1025687543

Correo electrónico

gustavo.sabogal@hotmail.com

Género

Masculino

[Siguiente](#)

## Continúa con el registro

Completa el formulario para unirte como estudiante a nuestra plataforma educativa

Edad

14

Departamento de residencia

Bogotá D.C

Número de celular

310798658765

Contraseña

\*\*\*\*\*

Confirme la contraseña

\*\*\*\*\*

☒ Acepta el tratamiento de datos

Siguiente

### 3.1.3.2. Dashboard: Permite a los estudiantes buscar y reservar sesiones de tutorías en las modalidades presenciales y virtuales en la materia de su interés.

TutorShop Home Tutores ▾ Estudiante ▾ Login

## Bienvenido a Tutorshop

Ofrecemos servicios de tutoría en una amplia variedad de materias para ayudarte a alcanzar tus metas académicas. Reserva una cita con nuestros expertos profesores y mejora tus habilidades.

Reservar cita

### Card title

#### Tutoría Individual

Recibe atención personalizada de nuestros expertos profesores para mejorar tu rendimiento académico.

[Más información](#)

### Card title

#### Tutoría en Grupo

Trabaja en equipo con tus compañeros y recibe apoyo de nuestros profesores expertos.

[Más información](#)

## Materias disponibles

### Card title

#### Matemáticas

Desde álgebra hasta cálculo, nuestros expertos profesores te ayudarán a dominar las matemáticas.

[Más información](#)

### Card title

#### Informática

Mejora tus habilidades en programación, bases de datos y más con nuestros tutores especializados.

[Más información](#)

### Card title

#### Lengua y Literatura

Mejora tus habilidades de escritura, comprensión lectora y más con nuestros expertos en el área.

[Más información](#)

## Contáctanos

Nombre

Gustavo Valderrama Sabogal

Correo electrónico

gustavo.sabogal@gmail.com

Mensaje

Estoy interesado en participar en la clase de Matemáticas grado noveno con el tutor Francisco Rodríguez con el fin de afianzar mis conocimientos y práctica en el área de la aritmética.

Enviar

### 3.1.4. Login: Permite al estudiante y profesor autenticarse en la aplicación mediante la opción de Login.

TutorShop Home Tutores ▾ Estudiante ▾ Login

## Ingresa a nuestra plataforma

Correo electrónico

gustavo.sabogalhotmail.com

Contraseña

.....

Ingresar

### 3.1.5. Reserva de tutoría.

Permite al estudiante reservar una tutoría con el profesor de su elección, materia, horario disponible y tipo.

## Reserva tu tutoría

Materia

Geografía

Profesores de la materia

Pedro Enrique

Tipo de tutoría

Virtual-Individual

Correo electrónico

juliana.castro@homail.com

Fecha de inicio

15/07/2024



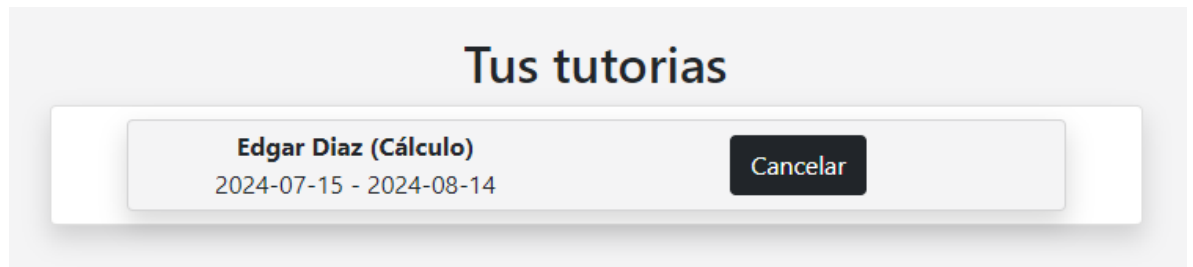
Fecha de finalización

14/08/2024



Reservar tutoría

Una vez realizada la agenda de la tutoría la aplicación le indica al estudiante la tutoría reservada:



## 4. Diseño técnico del sistema de información.

### 4.1. Esquema o modelo de requerimientos.

Las siguientes son las reglas de negocio implementadas dentro del sistema de información:

#### 4.1.1. Registro de usuarios.

Crear una interfaz para que el usuario se registre e inicie sesión (estudiante y tutor).

#### 4.1.2. Búsqueda de tutores.

Los estudiantes podrán buscar tutores por:

- Materia,
- Nivel educativo,
- Área geográfica (para los casos presenciales) y
- Disponibilidad.

#### 4.1.3. Reserva de Sesiones de Tutoría.

Los estudiantes podrán reservar sesiones de tutoría con los tutores disponibles según su horario.

#### 4.1.4. Gestión de disponibilidad.

Los tutores podrán:

- > Establecer su disponibilidad y
- > Aceptar o rechazar solicitudes de tutoría.



#### 4.1.5. Autenticación de usuarios.

El sistema está protegido por login. Permite al usuario registrarse en la aplicación mediante un usuario y password.

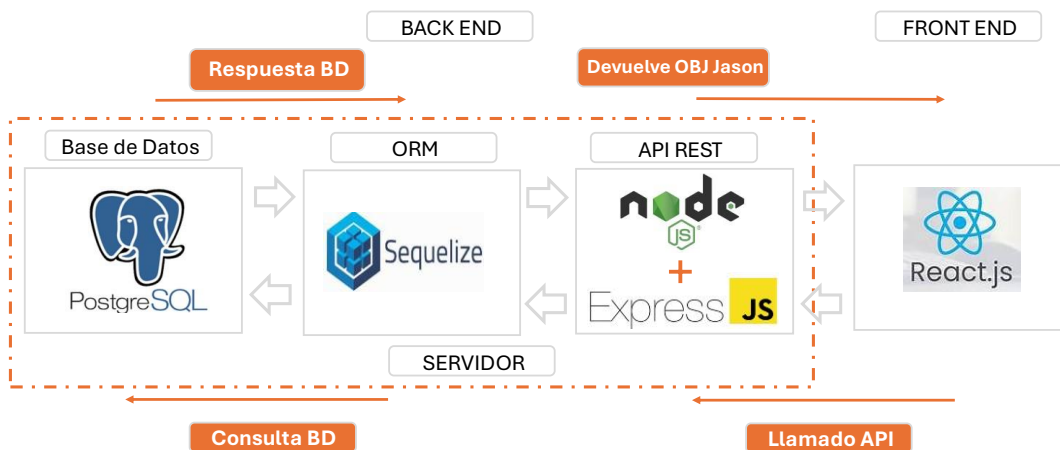
#### 4.1.6. Desplegar la interfaz.

Es posible acceder a la interfaz a través de internet.

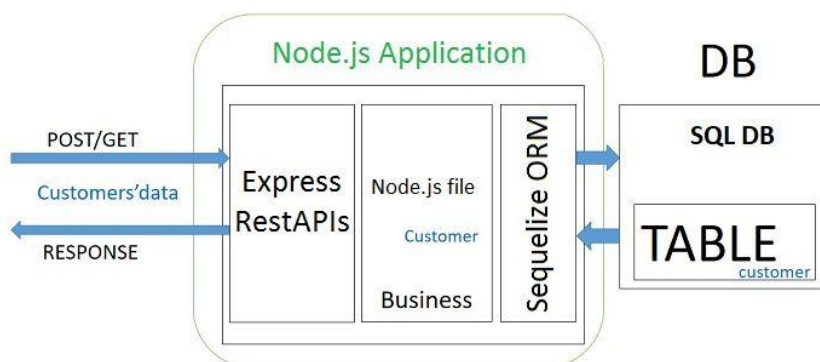
### 4.2. Componentes y estándares de la aplicación.

Los siguientes son los frameworks, librerías y herramientas usadas en la implementación de la aplicación:

#### Arquitectura de la Plataforma Tutorías en Línea



#### 4.2.1. Node(V18.16).

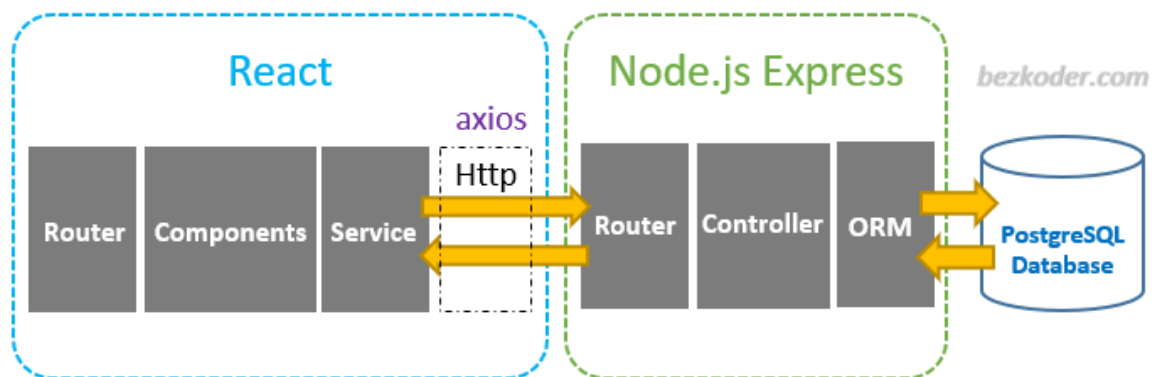


Node.js es un entorno de tiempo de ejecución de JavaScript basado en el motor V8 de Google Chrome que permite a los desarrolladores usar en sus aplicaciones JavaScript tanto en el front-end como en el back-end, proporcionando un entorno eficiente y escalable para construir aplicaciones modernas y de alto rendimiento.

#### 4.2.2. Express (V4.19.2).

Express es un framework web minimalista y flexible para Node.js que facilita la creación de aplicaciones web y APIs de manera rápida y sencilla. Es una capa sobre Node.js que simplifica tareas comunes como el enrutamiento, el manejo de solicitudes y respuestas HTTP, la definición de middleware, entre otras.

#### 4.2.3. ReactJS.



Es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes.

#### 4.2.4. Sequelize (V6.37.3).

Es una herramienta ORM (Object-Relational Mapping) basada en promesas para Node.js, compatible con Postgres, MySQL, MariaDB, SQLite, Microsoft SQL Server, Oracle Database, Amazon Redshift y Snowflake's Data Cloud.

Ofrece un sólido soporte de transacciones, relaciones, carga ansiosa y carga perezosa, replicación de lectura y más.

#### 4.2.5. Postgresql(V15).

PostgreSQL es un potente sistema de base de datos relacional orientado a objetos de código abierto, con más de 35 años de desarrollo activo que le ha valido una sólida reputación en cuanto a confiabilidad, robustez de funciones y rendimiento.

#### 4.2.6. Npm(9.6.6).

Es el registro de software más grande del mundo. Desarrolladores de código abierto de todos los continentes utilizan npm para compartir y utilizar paquetes, y muchas organizaciones también lo utilizan para gestionar el desarrollo privado.

#### 4.2.7. Postman.

Cliente HTTP con el que se realizarán las pruebas al funcionamiento de la API. Postman es una plataforma de API para construir y utilizar APIs. Postman simplifica cada etapa del ciclo de vida de las APIs y optimiza la colaboración, lo que te permite crear APIs de mejor calidad y de manera más rápida.

#### 4.2.8. GitHub.

Es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. GitHub aloja tu repositorio de código y te brinda

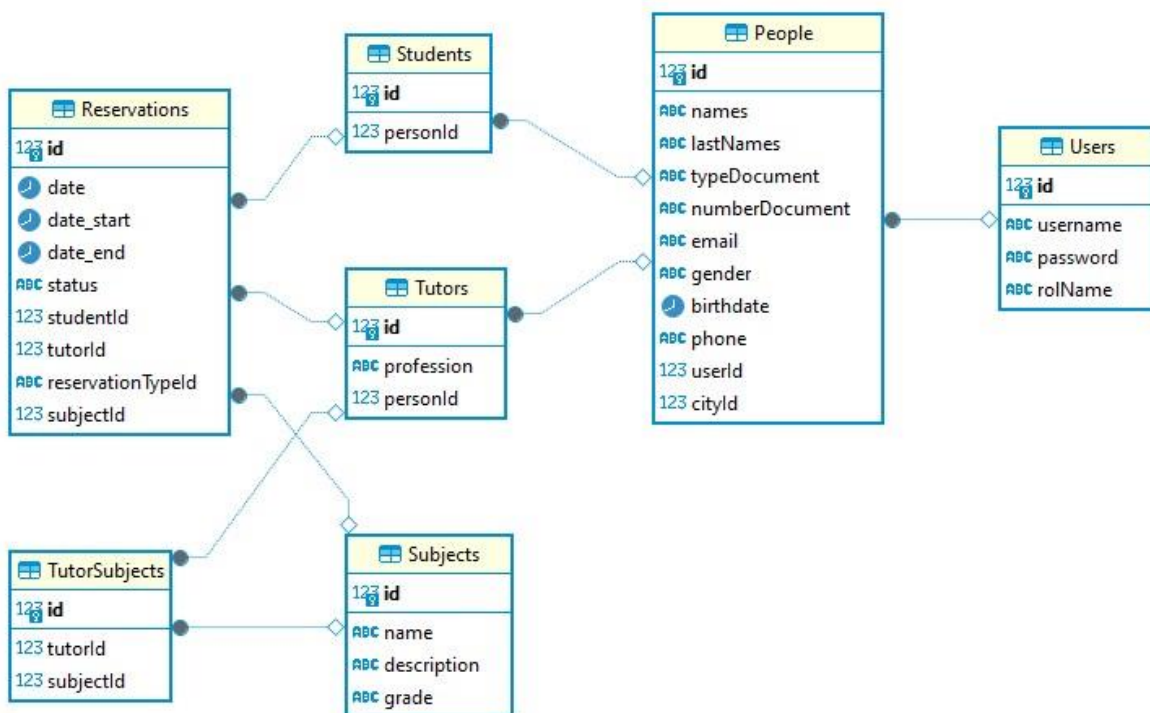
herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

### 4.3. Modelo de datos.

#### 4.3.1. Modelo entidad-relación.

El modelo es una representación de una tabla en una base de datos relacional. Es una clase que define la estructura y las relaciones de la tabla, y proporciona una interfaz para interactuar con los datos de esa tabla utilizando objetos y métodos. También define los campos que deben estar presentes en cada tabla y especifica los tipos de datos, las validaciones y las opciones adicionales para cada campo.

El siguiente es el modelo entidad relación de la base de datos de la aplicación con motor PostgreSQL:



#### 4.3.2. Diccionario de datos.

A continuación, describimos el diccionario de datos de la base de datos de la aplicación con motor PostgreSQL:

##### 4.3.2.1. Entidad Users.

<b>Users:</b> Entidad que contiene los atributos del usuario.				
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>
id	integer	4	Identificador único de la entidad.	Pk
username	string	255	Nombre del usuario.	
password	string	255	Contraseña del usuario.	
rolName	string	255	Rol del usuario. 'user' o 'admin'.	

##### 4.3.2.2. Entidad People.

<b>People:</b> Entidad que contiene los atributos del usuario estudiante y tutor.					
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>	<b>Validación</b>
id	integer	4	Identificador único de la entidad.	Pk	
names	string	255	Nombres del usuario.		
lastNames	string	255	Apellidos del usuario.		
typeDocument	integer	2	Tipo de documento del usuario.		<b>CC:</b> 'Cédula de ciudadanía', <b>CE:</b> 'Cédula de extranjería', <b>TI:</b> 'Tarjeta de identidad', <b>OT:</b> 'Otro',
numberDocument	string	255	Número de documento.		

**People:** Entidad que contiene los atributos del usuario estudiante y tutor.

<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>	<b>Validación</b>
email	string	255	Email del usuario.		
gender	string	2	Género del usuario.		<b>M:</b> 'Masculino', <b>F:</b> 'Femenino', <b>O:</b> 'Otro', <b>U:</b> 'No especificado'
birhdate	date	10	Fecha de nacimiento del usuario.		aaaa-mm-dd
phone	string	255	Teléfono del usuario.		
userId	integer	4	Llave foránea, relación con la entidad Users.	Fk	

#### 4.3.2.3. Entidad Students.

**Students:** Entidad que contiene los atributos del estudiante.

<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>
id	integer	4	Identificador único de la entidad.	Pk
personId	integer	4	Llave foránea, relación con la entidad People.	Fk

#### 4.3.2.4. Entidad Tutors.

<b>Tutors:</b> Entidad que contiene los atributos del tutor.				
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>
id	integer	4	Identificador único de la entidad.	Pk
profession	string	255	Profesión del tutor.	
personId	integer	4	Llave foránea, relación con la entidad People.	Fk

#### 4.3.2.5. Entidad TutorSubjects.

<b>TutorSubjects:</b> Entidad que contiene la relación del tutor con sus asignaturas.				
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>
id	integer	4	Identificador único de la entidad.	Pk
tutorId	integer	4	Llave foránea, relación con la entidad Tutors.	Fk
subjectId	integer	4	Llave foránea, relación con la entidad Subjects.	Fk

#### 4.3.2.6. Entidad Subjects.

<b>Subjects:</b> Entidad que contiene los atributos de las asignaturas.				
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>
id	integer	4	Identificador único de la entidad.	Pk
name	string	255	Nombre de la asignatura.	
description	string	255	Descripción de la asignatura.	
grade	string	255	Grado de la asignatura.	

#### 4.3.2.7. Entidad Reservations.

<b>Reservations:</b> Entidad que contiene los atributos de las reservaciones de tutorías realizadas por los estudiantes.					
<b>Campo</b>	<b>Tipo</b>	<b>Long</b>	<b>Descripción</b>	<b>Llave</b>	<b>Validación</b>
id	integer	4	Identificador único de la entidad.	Pk	
date	date	10	Fecha de creación de la reservación.		aaaa-mm-dd
date_start	date	10	Fecha inicial de la reservación.		aaaa-mm-dd
date_end	date	10	Fecha final de la reservación.		aaaa-mm-dd
status	text	2	Estado de la reservación.		C: 'Creada', A: 'Aceptada', I: 'Iniciada', T: 'Terminada', X: 'Cancelada'
studentId	integer	4	Llave foránea, relación con la entidad Students.	Fk	
tutorId	integer	4	Llave foránea, relación con la entidad Tutors.	Fk	
reservationType	integer	4	Llave foránea, relación con la entidad ReservationTypes.	Fk	



**Reservations:** Entidad que contiene los atributos de las reservaciones de tutorías realizadas por los estudiantes.

Campo	Tipo	Long	Descripción	Llave	Validación
subjectId	integer	4	Llave foránea, relación con la entidad Subjects	Fk	

#### 4.3.2.8. ReservationTypes

**ReservationTypes:** Entidad que contiene los atributos de las descripciones de los tipos de reservaciones.

Campo	Tipo	Long	Descripción	Llave	Validación
id	integer	4	Identificador único de la entidad.	Pk	
description	string	255	Descripción del tipo de reservación.		PI: 'Presencial-Individual', PG: 'Presencial-Grupal', VI: 'Virtual-Individual', VG: 'Virtual-Grupal', GR: 'Grupal', MI: 'Mixta-Individual', MG: 'Mixta-Grupal',

#### 4.4. Creación de servicios en el back-end.

La capa de servicios es la encargada de la lógica de negocio, aquí se realiza la conexión de Sequelize (ORM: (Object-Relational Mapping) con la base de datos y creamos los métodos que realizan las operaciones CRUD (Create, Read, Update y Delete) sobre las entidades de la base de datos.

##### 4.4.1. citiesController.js.

Permite consultar en la API Colombia el nombre del departamento y el nombre de la ciudad asociada al apartamento.

##### 4.4.2. genericController.js.

Permite consultar los tipos de documentos definidos en la entidad documentTypeEnum.js de la base de datos:

- CC: 'Cédula de ciudadanía',
- CE: 'Cédula de extranjería',
- TI: 'Tarjeta de identidad',
- OT: 'Otro'.

Los tipos de género definidos en la entidad genderEnum.js de la base de datos:

- M: 'Masculino',
- F: 'Femenino',
- O: 'Otro',
- U: 'No especificado'

Los tipos de reservación definidos en la entidad `reservationTypeEnum.js` de la base de datos:

- PI: 'Presencial-Individual',
- PG: 'Presencial-Grupal',
- VI: 'Virtual-Individual',
- VG: 'Virtual-Grupal',
- GR: 'Grupal',
- MI: 'Mixta-Individual',
- MG: 'Mixta-Grupal'.

#### 4.4.3. `reservationController.js`.

En este servicio se definen las operaciones CRUD a realizar en la entidad `Reservations` de la base de datos, como son:

- Crear una nueva reservación por el estudiante.
- Consultar las reservaciones registradas en la base de datos.
- Consultar una reservación por su 'id'.
- Consultar las reservaciones realizadas por un estudiante.
- Consultar las reservaciones realizadas por un tutor.
- Actualizar una reservación por su 'id'.
- Eliminar una reservación por su 'id'.

#### 4.4.4. `studentController.js`.

En este servicio se definen las operaciones CRUD a realizar en las entidades `People` y `Students` de la base de datos, como son:

- Crear un nuevo estudiante.

- Crear un estudiante como tutor por el id del estudiante.
- Consultar los estudiantes registrados en la base de datos.
- Consultar un estudiante por su 'id'.
- Actualizar un estudiante por su 'id'.
- Eliminar un estudiante por su 'id'.

#### 4.4.5. subjectController.js.

En este servicio se definen las operaciones CRUD a realizar en la entidad Subjects de la base de datos, como son:

- Crear una nueva materia.
- Consultar las materias registradas en la base de datos.
- Consultar una materia por su 'id'.
- Consultar las materias que enseña un tutor por el 'id' de la materia.
- Actualizar una materia por su 'id'.
- Eliminar una materia por su 'id'.

#### 4.4.6. tutorController.js.

En este servicio se definen las operaciones CRUD a realizar en las entidades People y Tutors de la base de datos, como son:

- Crear un nuevo tutor.
- Crear un tutor como estudiante por el id del tutor.
- Consultar los tutores registrados en la base de datos.
- Consultar un tutor por su 'id'.
- Actualizar un tutor por su 'id'.
- Eliminar un tutor por su 'id'.

#### 4.4.7. tutorSubjectController.js.

En este servicio se definen las operaciones CRUD a realizar en la entidad TutorSubjects de la base de datos, como son:

- Crear un nuevo tutor asociado a una asignatura.
- Consultar la asignatura por el 'id' del tutor.
- Eliminar la asignatura asociada a un tutor por su 'id'.

#### 4.4.8. userController.js.

En este servicio se definen las operaciones CRUD a realizar en la entidad Users de la base de datos, como son:

- Crear un nuevo usuario mediante login.
- Consultar los usuarios registrados en la base de datos.
- Eliminar un usuario por su 'id'.

#### 4.5. Creación de las rutas HTTP en el back-end.

La base de una API RESTFul es el uso correcto del protocolo HTTP y sus tipos de peticiones, por ello en nuestra aplicación utilizamos las siguientes rutas:

Para la aplicación desplegada en Render, se cambia el prefijo <http://localhost/api> por la url asignada por render: <https://tutor-shop.onrender.com/>

Los siguientes endpoints son usados por la aplicación cuando reside en un computador de forma local.

##### 4.5.1. Users.

Endpoints que se encargan del registro y validación de los usuarios registrados en la aplicación mediante login.

##### **GET** all-users

<http://localhost/api/students>

Esta ruta sirve para obtener todos los registros almacenados en la entidad Users.

##### **POST** Registrar-user

<http://localhost/api/user/register/user>

Mediante esta ruta registramos un usuario en la entidad Users de la base de datos.

##### 4.5.2. Students.

Endpoints que se encargan del registro, consulta, actualización y eliminación de los estudiantes registrados en la aplicación.

### **GET** All-students

<http://localhost:4000/api/student/students>

Esta ruta sirve para obtener todos los registros almacenados en la entidad People y su relación con la entidad Students.

### **POST** Registrar-student

<http://localhost/api/student/register>

Mediante esta ruta registramos un estudiante en la entidad People y su relación con la entidad Students de la base de datos.

### **GET** StudentById

<http://localhost:4000/api/student/student/1>

Esta ruta sirve para consultar el estudiante por su llave primaria almacenado en la entidad Students incluida la entidad People donde se encuentra la información básica del estudiante.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Users.

### **PUT** Student-update

<http://localhost:4000/api/student/student/1>

Esta ruta sirve para actualizar el estudiante por su llave primaria almacenado en la entidad Students incluida la entidad People donde se encuentra la información básica del estudiante.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Students y mediante la petición **PUT** realizamos la actualización de la información del estudiante en la entidad People.

### **DELETE** Student-delete

<http://localhost:4000/api/student/student/1>

Esta ruta sirve para eliminar el estudiante por su llave primaria almacenado en la entidad Students incluida la entidad People donde se encuentra la información básica del estudiante.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Students y mediante la petición **DELETE** realizamos la eliminación física de la información del estudiante en la entidad People y en la entidad Students.

### 4.5.3. Tutors.

Endpoints que se encargan del registro, consulta, actualización y eliminación de los tutores registrados en la aplicación.

### **GET** All-tutors

<http://localhost:4000/api/tutor/tutors>

Esta ruta sirve para obtener todos los registros almacenados en la entidad People y su relación con la entidad Tutors.

### **POST** Registrar-tutor

<http://localhost/api/tutor/tutor/new>

Mediante esta ruta registramos un estudiante en la entidad People y su relación con la entidad Users de la base de datos.

### **GET** TutorById

<http://localhost:4000/api/tutor/tutor/1>



Esta ruta sirve para consultar el tutor por su llave primaria almacenado en la entidad Tutors incluida la entidad People donde se encuentra la información básica del tutor.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Tutors.

#### **PUT** Tutor-update

<http://localhost:4000/api/tutor/tutor/1>

Esta ruta sirve para actualizar el tutor por su llave primaria almacenado en la entidad Tutors incluida la entidad People donde se encuentra la información básica del tutor.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Tutors y mediante la petición **PUT** realizamos la actualización de la información del tutor en la entidad People.

#### **DELETE** Tutor-delete

<http://localhost:4000/api/tutor/tutor/1>

Esta ruta sirve para eliminar el tutor por su llave primaria almacenado en la entidad Tutors incluida la entidad People donde se encuentra la información básica del tutor.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Tutors y mediante la petición **DELETE** realizamos la eliminación física de la información del tutor en la entidad People y en la entidad Tutors.

#### 4.5.4. Reservations.

Endpoints que se encargan del registro, consulta, actualización y eliminación de las reservaciones de tutorías realizadas por los estudiantes y los tutores pueden aceptar o rechazar la tutoría de acuerdo con su disponibilidad.

### **GET** All-reservations

<http://localhost:4000/api/reservation/reservations>

Esta ruta sirve para obtener todos los registros almacenados en la entidad Reservations y su relación con la entidad Students.

### **POST** Registrar-reservation

<http://localhost/api/reservation/reservation/new>

Mediante esta ruta registramos una reservación realizada por un estudiante y su relación con la entidad Students de la base de datos.

### **GET** ReservationById

<http://localhost:4000/api/reservation/reservation/1>

Esta ruta sirve para consultar la reservation por la llave primaria de la reservación almacenada en la entidad Reservations.

En el ejemplo '1' indica el id de la reservación por el cual realizamos la consulta a la entidad Reservations.

### **GET** ReservationTutorIdAndStatus

<http://localhost:4000/api/reservation/reservations/tutor/1/Creada>

Esta ruta sirve para consultar la reservation por la llave foránea del tutor y su status almacenado en la entidad Reservations.

En el ejemplo '1' indica el id del tutor por el cual realizamos la consulta a la entidad Reservations y status='Creada' retorna sólo los registros con este valor.

### **GET** ReservationStudentById

<http://localhost:4000/api/reservation/reservations/student/1>

Esta ruta sirve para consultar la reservation por la llave foránea del estudiante almacenado en la entidad Reservations.

En el ejemplo '1' indica el id del estudiante por el cual realizamos la consulta a la entidad Reservations.

### **PUT** Reservation-update

<http://localhost:4000/api/reservation/reservation/1>

Esta ruta sirve para actualizar la reservation por su llave primaria almacenada en la entidad Reservations.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Reservations y mediante la petición **PUT** realizamos la actualización de la información de la reservación en la entidad Reservations de la base de datos.

### **DELETE** Reservation-delete

<http://localhost:4000/api/reservation/reservation/1>

Esta ruta sirve para eliminar la reservation por su llave primaria almacenada en la entidad Reservations.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Reservations y mediante la petición **DELETE** realizamos la eliminación física de la información de la reservación en la entidad Reservations.

#### 4.5.5. TutorSubjects.

Endpoints que se encargan del registro, consulta, actualización y eliminación de las asignaturas asociadas a un tutor.

### **POST** Registrar-tutorSubject

<http://localhost/api/tutorSubject/tutorSubject/new>

Mediante esta ruta registramos una asignatura asignada a un tutor, por lo tanto, previamente validamos que exista el tutor y la asignatura en la base de datos.

### **GET** TutorSubjectByTutorId

<http://localhost:4000/api/tutorSubject/tutorSubject/1>

Esta ruta sirve para consultar la asignatura por la llave foránea del tutor almacenado en la entidad TutorSubjects. En el ejemplo '1' indica el id del tutor por el cual realizamos la consulta a la entidad TutorSubjects.

### **DELETE** TutorSubject-delete

<http://localhost:4000/api/tutorSubject/tutorSubject/1>

Esta ruta sirve para eliminar la asignatura por su llave primaria almacenada en la entidad TutorSubjects. En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad TutorSubjects y mediante la petición **DELETE** realizamos la eliminación física de la información de la asignatura en la entidad TutorSubjects.

### **GET** TutorSubject-UnselectedBySubjectId

<http://localhost:4000/api/tutorSubject/tutorUnselectedSubject/1>

Esta ruta sirve para consultar las asignaturas que no han sido seleccionadas a un tutor por la llave foránea tutorId almacenada en la entidad TutorSubjects. En el ejemplo '1' indica el id del tutor por el cual realizamos la consulta a la entidad TutorSubjects.

#### 4.5.6. Subjects.

Endpoints que se encargan del registro, consulta, actualización y eliminación de las asignaturas.

##### **GET** All-subjects

<http://localhost:4000/api/subject/subjects>

Esta ruta sirve para consultar las asignaturas almacenadas en la entidad Subjects mediante la función findAll() de sequelize.

##### **POST** Registrar-subject

<http://localhost/api/subject/subject/new>

Mediante esta ruta registramos una asignatura en la entidad asignatura en la base de datos.

##### **GET** SubjectById

<http://localhost:4000/api/subject/subject/1>

Esta ruta sirve para consultar la asignatura por la llave primaria de Subjects almacenada en la entidad Subjects. En el ejemplo '1' indica el id de la asignatura por la cual realizamos la consulta a la entidad Subjects.

##### **GET** TutorSubjectBySubjectId

<http://localhost:4000/api/subject/tutorsSubject/1>

Esta ruta sirve para consultar la entidad Subjects con la llave primaria de la asignatura, si existe, busca todos los tutores asignados a la asignatura con la llave foránea tutorId en la entidad TutorSubjects.

##### **PUT** Subject-update

<http://localhost:4000/api/subject/subject/1>

Esta ruta sirve para actualizar la asignatura por su llave primaria almacenada en la entidad Subjects.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Subjects y mediante la petición **PUT** realizamos la actualización de la información de la asignatura en la entidad Subjects de la base de datos.

#### **DELETE** Subject-delete

<http://localhost:4000/api/subject/subject/1>

Esta ruta sirve para eliminar la asignatura por su llave primaria almacenada en la entidad Subjects.

En el ejemplo '1' indica el id por el cual realizamos la consulta a la entidad Subjects y mediante la petición **DELETE** realizamos la eliminación física de la información de la asignatura en la entidad Subjects.

#### 4.5.7. Generics.

Endpoints que se encargan de la consulta de los tipos de género, documento y reservaciones.

#### **GET** Consultar-gender

<http://localhost:4000/api/generic/gender>

Retorna los tipos de género almacenados en la entidad GenderEnum.

#### **GET** Consultar-documentType

<http://localhost:4000/api/generic/reservationType>

Retorna los tipos de reservaciones almacenados en la entidad DocumentTypeEnum.

#### **GET** Consultar-reservationType

<http://localhost:4000/api/generic/documentType>

Retorna los tipos de reservaciones almacenados en la entidad ReservationTypeEnum.

#### 4.5.8. Cities.

Endpoints que se encargan de la consulta de los nombres del departamento y ciudad escogida por el estudiante y tutor cuando se registra en la aplicación.

##### **GET** Consultar-departamento

<http://localhost:4000/api/departments/cities?name=Cundinamarca>

Esta ruta permite consultar la API Colombia para extraer el nombre del departamento escogido por el estudiante y tutor en el momento de registrarse en la aplicación.

##### **GET** Consultar-ciudades-departamento

<http://localhost:4000/api/departments/cities?name=nariño>

Esta ruta permite consultar la API Colombia para extraer el nombre del departamento escogido por el estudiante y tutor en el momento de registrarse en la aplicación y a partir del departamento extraemos el nombre de la ciudad asociada.

#### 4.6. Componentes del front-end.

A continuación, se describen los componentes relevantes de la aplicación en el front-end:

##### 4.6.1. studentFormSlice.

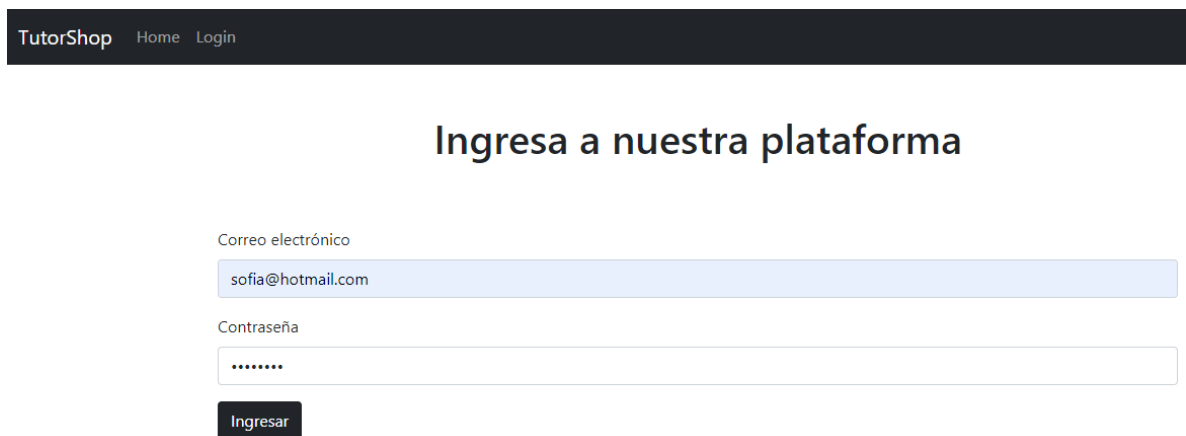
Estados iniciales del formulario de registro del estudiante, datos de contacto y reserva de tutorías.

#### 4.6.2. userSlice.

Se define un slice de Redux llamado userSlice que gestiona el estado del usuario (user). Proporciona acciones (login, logout, setUser) para actualizar y manipular ese estado, y se integra con localStorage para persistir el estado del usuario de manera local en el navegador.

#### 4.6.3. Login.

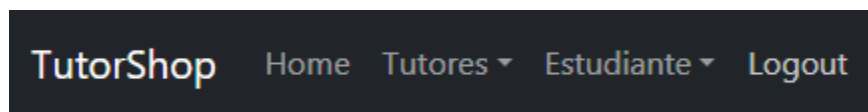
Vista que permite el ingreso del usuario a la aplicación mediante el login.



The screenshot shows a login form within a dark header bar. The header contains the text 'TutorShop' followed by links 'Home' and 'Login'. The main heading of the form is 'Ingresa a nuestra plataforma'. Below this, there are two input fields: the first is labeled 'Correo electrónico' and contains the text 'sofia@hotmail.com'; the second is labeled 'Contraseña' and contains seven dots. At the bottom of the form is a dark button with the text 'Ingresar'.

#### 4.6.4. Logout.

Vista que permite salir al usuario de la aplicación mediante el logout.



The screenshot shows a dark header bar with the text 'TutorShop' followed by links 'Home', 'Tutores' with a dropdown arrow, 'Estudiante' with a dropdown arrow, and 'Logout'.



#### 4.6.5. Schemas.

Esquemas de validación de los campos del formulario del estudiante, contacto y reserva de tutorías mediante el uso de Yup para la validación de formularios en JavaScript.

#### 4.6.6. Services.

Este servicio se utiliza para realizar las peticiones HTTP asíncronas al back mediante la función integrada fetch de JavaScript para la solicitud de la siguiente información relevante:

- Nombre de los departamentos.
- Nombres de las ciudades asociadas al departamento escogido por el usuario en el momento de su registro a la aplicación.
- Género del estudiante y tutor.
- Tipo de documento del estudiante y tutor.
- Información del estudiante y tutor.

#### 4.6.7. Store.

Árbol de objetos que almacena los estados de la aplicación.

#### 4.6.8. Student.

Contiene las vistas de la información de registro, contacto, servicios y reservas de tutorías del estudiante.

#### 4.6.9. Tutor.

Contiene las vistas de la información de registro, selección de asignaturas, aceptar o rechazar tutorías asignadas del tutor.

#### 4.6.10. App.

Contiene las rutas de navegación de la aplicación del front-end.

#### 4.6.11. Index.

Es comúnmente utilizado como un archivo principal dentro de un módulo o directorio en la aplicación, sirve como punto de entrada, organizador de exportaciones y una convención estándar para la estructura del código utilizado en javascript.

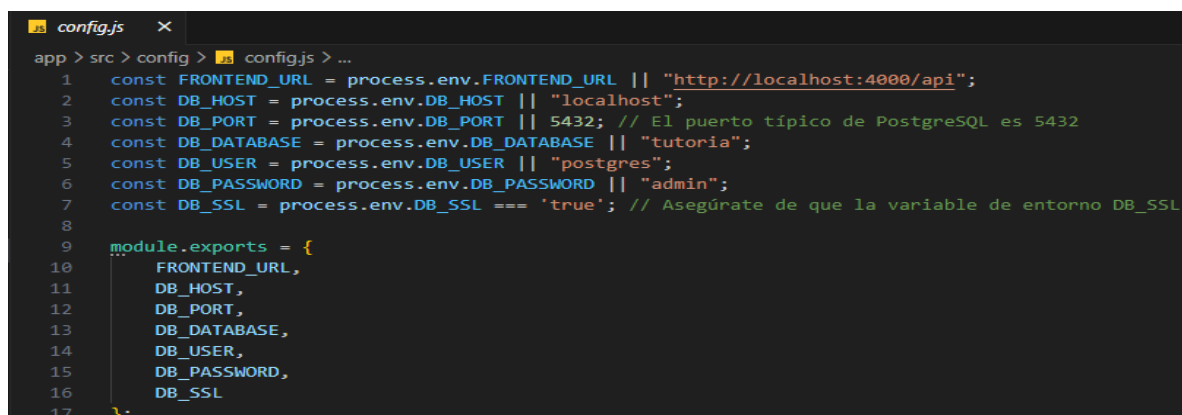
## 5. Despliegue de la aplicación en Render.

### Render.

Es un proveedor de servicios de alojamiento en la nube. Es recomendable hacer el despliegue por separado de los servicios para el back-end, front-end y la base de datos.

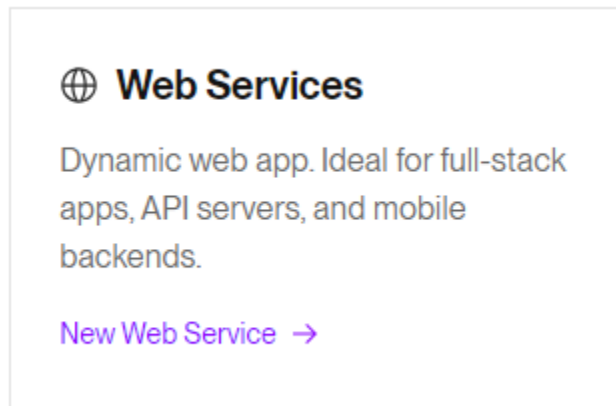
Pasos para realizar el despliegue de la aplicación:

1. Configurar las variables de entorno en la aplicación local en el **back**:
2. Bajo la carpeta src, crear el archivo config.js que va a contener las variables de entorno de la aplicación:

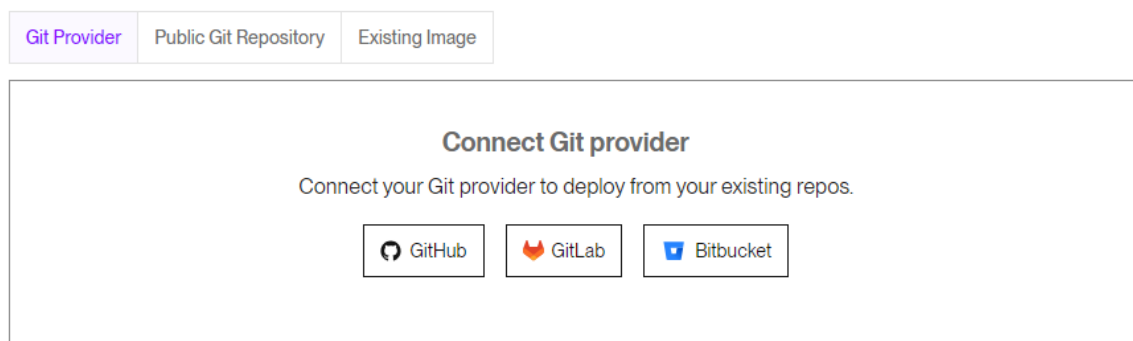


```
config.js
app > src > config > config.js > ...
1  const FRONTEND_URL = process.env.FRONTEND_URL || "http://localhost:4000/api";
2  const DB_HOST = process.env.DB_HOST || "localhost";
3  const DB_PORT = process.env.DB_PORT || 5432; // El puerto típico de PostgreSQL es 5432
4  const DB_DATABASE = process.env.DB_DATABASE || "tutoria";
5  const DB_USER = process.env.DB_USER || "postgres";
6  const DB_PASSWORD = process.env.DB_PASSWORD || "admin";
7  const DB_SSL = process.env.DB_SSL === 'true'; // Asegúrate de que la variable de entorno DB_SSL
8
9  module.exports = {
10     FRONTEND_URL,
11     DB_HOST,
12     DB_PORT,
13     DB_DATABASE,
14     DB_USER,
15     DB_PASSWORD,
16     DB_SSL
17  };
```

3. Ingrese a la página de render en el enlace: <https://render.com/>
4. Cree una cuenta o ingrese mediante autenticación con el servicio de loguearse con Github.
5. En la pestaña Dashboard elige el servicio que necesitas, en nuestro caso requerimos desplegar la aplicación web mediante la opción 'Web Services'.

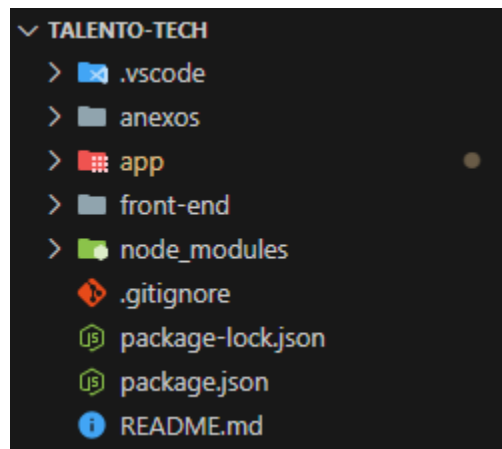


6. Configurar y desplegar su nuevo servicio Web.
7. Vamos a desplegar el servicio web a partir del repositorio de GitHub: <https://github.com/darkelian/Talento-Tech>



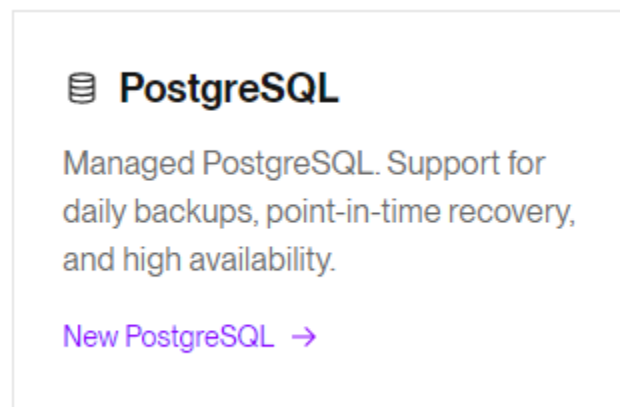
8. En Render pestaña 'settings' configurar el servicio de la aplicación Web:

- **Name:** El nombre de la aplicación.
- **Region:** Oregon (us).
- **Branch:** La rama donde se va a desplegar es master.
- **Root Directory:** Raíz donde se especifica la carpeta 'app' o index.js de la aplicación:



- **Start command:** npm install && npm start

9. Configurar el repositorio de la base de datos motor PostgreSQL mediante la opción 'PostgreSQL'.



- **Name:** Talento-tech.
- **Database:** tutoria.
- **Host:** 5432.
- **User:** posgres

- **Region:** Oregon(us)
- **PosgreSQL Version:** 15
- **Crear:** Click en el botón para crear la base de datos.
- **Advanced:** Luego regresamos a la sección de Web Services y creamos las variables de entorno de la de la base de datos con los valores asignados por render:
  - ✓ DB\_HOST.
  - ✓ DB\_PORT.
  - ✓ DB\_DATABASE.
  - ✓ DB\_USER.
  - ✓ DB\_PASSWORD.
  - ✓ FRONTEND\_URL. (variable del front).
  - ✓ **Crear Web service.** Click en el botón. Esto genera la creación del despliegue del back en render.

#### 10. Crear el despliegue del front.

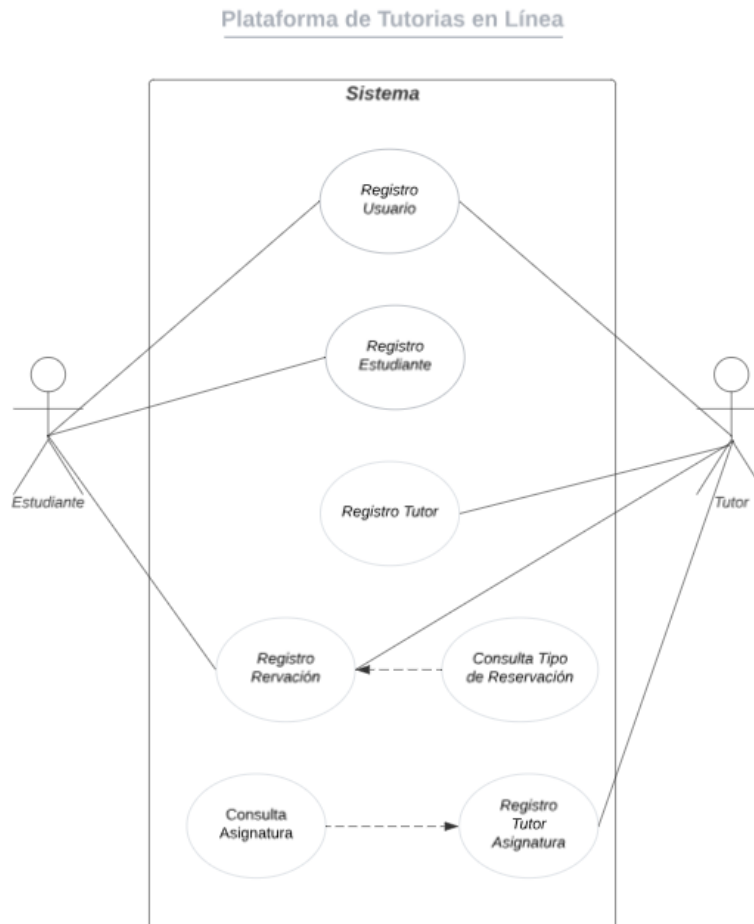
Pestaña DASHBOARD click en botón 'New', click en 'Static Site' y luego click en 'connect'.

Configurar los siguientes valores del front:

- **Name:** front-end.
- **Branch:** main.
- **Root Directory:** front.
- **Build command:** npm run build.
- **Publish directory:** dist.
- **FRONTEND\_URL:** Como valor se coloca el dominio dado por render.
- **Advanced:** En este creamos las variables de entorno.
- **Create static site:** Click en el botón para que render genere el despliegue del **front** de la aplicación.

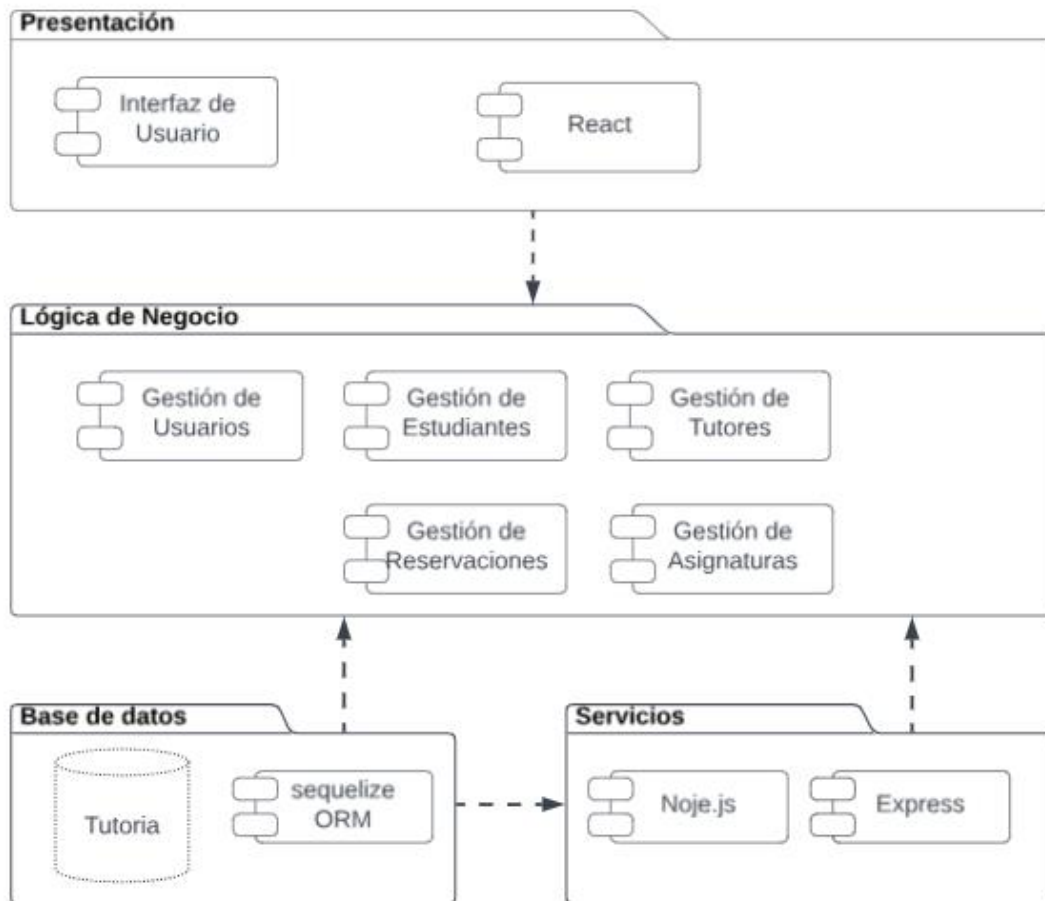
## 6. Diagrama de casos de uso.

A continuación, se muestra el diagrama de casos de uso de la aplicación Plataforma de Tutorías en Línea.



## 7. Organización de componentes.

A continuación, se muestra el diagrama de componentes de la aplicación Plataforma de Tutorías en Línea.



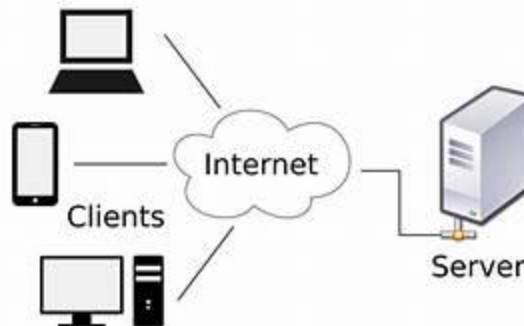
## 8. Términos y definiciones.

A continuación, relacionamos los términos más relevantes para brindar un acercamiento a los conceptos usados en la aplicación web:

**Navegador** (o browser, en inglés) se designa, en informática, la aplicación o programa que permite acceder a páginas web y navegar por una red informática, principalmente Internet, ya sea desde computadoras personales o dispositivos móviles.

Entre los más populares tenemos Google Chrome, Apple Safari, Microsoft Edge, Mozilla Firefox, Opera.

### Modelo cliente servidor



El modelo cliente-servidor, también conocido como “principio cliente-servidor”, es un modelo de comunicación que permite la distribución de tareas dentro de una red de ordenadores.

Un **servidor** es un hardware que proporciona los recursos necesarios para otros ordenadores o programas, pero un servidor también puede ser un programa informático que se comunica con los clientes. Un servidor acepta las peticiones del cliente, las procesa y proporciona la respuesta solicitada.

También existen diferentes tipos de clientes. Un ordenador o un programa informático se comunica con el servidor, envía



solicitudes y recibe respuestas del servidor. En cuanto al modelo cliente-servidor, representa la interacción entre el servidor y el cliente.

### **API** (Interfaz de Programación de aplicaciones).

En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones.

### **Endpoint.**

Un endpoint es una dirección de una API, o bien un backend que se encarga de dar respuesta a una petición, mientras que una REST, en una API, es una interfaz que sirve para la conexión de varios sistemas. Se basa en HTTP y sirve para obtener y enviar datos e información.

### **API RESTFul.**

La transferencia de estado representacional (REST) es una arquitectura de software que impone condiciones sobre cómo debe funcionar una API. En un principio, REST se creó como una guía para administrar la comunicación en una red compleja como Internet. Es posible utilizar una arquitectura basada en REST para admitir comunicaciones confiables y de alto rendimiento a escala. Puede implementarla y modificarla fácilmente, lo que brinda visibilidad y portabilidad entre plataformas a cualquier sistema de API.

### **HTTP.**

Es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque se puede utilizar para otros propósitos también.

## **fetch.**

Es una función proporcionada por los navegadores web modernos que permite realizar peticiones HTTP desde JavaScript hacia recursos en la red, como APIs o servicios web.

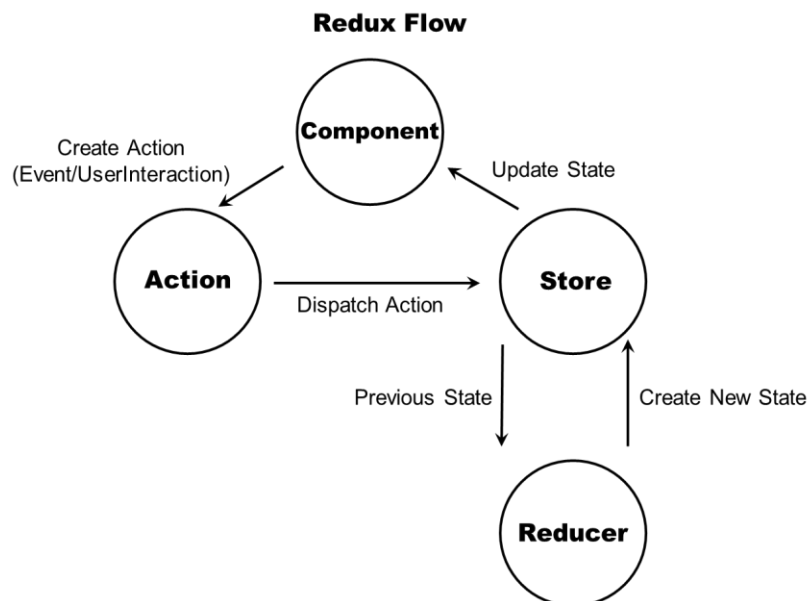
fetch utiliza promesas, lo que facilita el manejo de respuestas asíncronas.

## **Formik.**

Es una biblioteca de gestión de formularios para React, diseñada para hacer el proceso de creación y validación de formularios en aplicaciones web más sencillo y eficiente. Proporciona una manera fácil de manejar el estado de los formularios, la validación de datos, el envío de datos y la interacción con los campos del formulario en aplicaciones React.

## **React-Redux.**

Proporciona una forma eficiente y fácil de conectar componentes React con el store de Redux, permitiendo que los componentes React accedan al estado global y despachen acciones para actualizar ese estado.



**React Router Dom.**

Es una biblioteca de enrutamiento diseñada para React, específicamente para la construcción de aplicaciones de una sola página (SPA). Permite manejar la navegación y la visualización de componentes React según la URL actual del navegador, proporcionando una experiencia de usuario fluida y sin recargar la página completa.

**Cors.**

Cross-Origin Resource Sharing (Compartición de recursos entre diferentes orígenes).

Es un paquete de middleware que se utiliza para habilitar CORS en una aplicación web basada en Express.js.

Esto te permite configurar tus rutas o endpoints para que respondan adecuadamente a las solicitudes CORS desde otros orígenes.

## 9. Contacto de soporte.

El equipo de soporte de Talento Tech está disponible para atender cualquier inquietud y brindar una solución oportuna a su requerimiento.