Especificación de Requerimientos
Proyecto Tenjin

Natalia Pérez
Juan Juarez

26 de mayo de 2023

1. Enunciado del Problema

Crear un producto de software cuyos objetivos son los siguientes: Desarrollar una aplicación web que sirva como plataforma para que como alumno puede ofrecer asesorías sobre asignaturas a otros alumnos que la requieran. Debe permitir la evaluaciones y comentarios de los asesores para que los alumnos puedan elegir. El diseño de ser atractivo pero no abrumador, debe ser intuitivo y debe estar ordenado.

2. Glosario de términos

Termino	Significado
Usuario	Cualquier persona que ha realizado el registro tanto como alumno o asesor
Alumno	El usuario que contrata al asesor
Asesor	El usuario que ofrece la asesoría

3. Casos de uso

- Registro de usuario. El sistema permite a los usuarios registrarse el sistema, dando su: nombre, correo, contraseña, carrera y una descripción.
- Modificación de usuario. El sistema debe permitir a los usuarios modificar su nombre, su carrera, las asesorías que ofrece, su foto de perfil y su descripción.
- Eliminación de usuarios. El sistema debe eliminar usuarios mediante su identificador.
- Manejo de sesión del usuario. El sistema debe iniciar y cerrar sesión de los usuarios.
- Usuario hace comentario. El sistema permite a un usuario dejar comentarios y hacer una evaluación en base de 5 estrellas a otro usuario.
- Borrar comentarios. El sistema permite borrar comentarios usando su identificador.

4. Prototipos

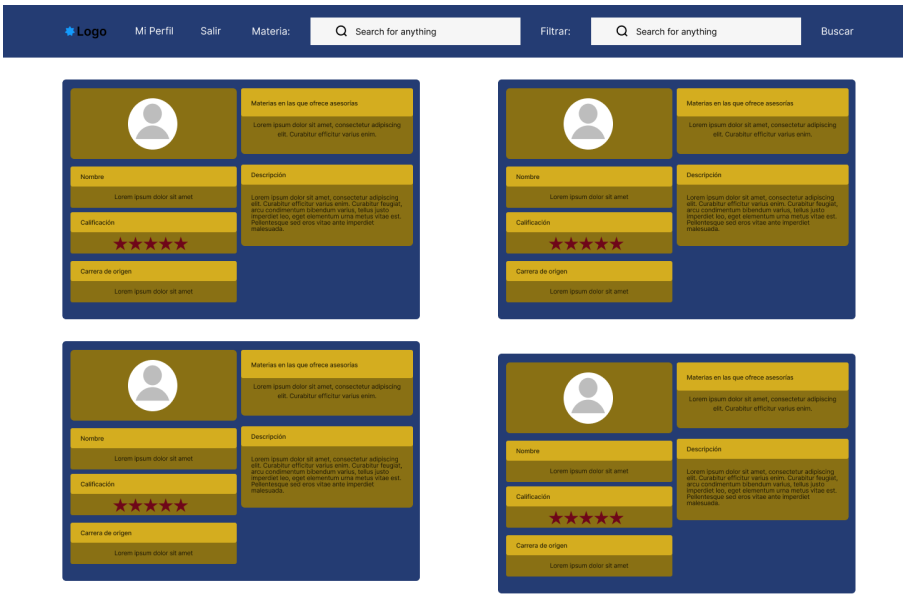

Figura 1: Vista de tarjeta del asesor

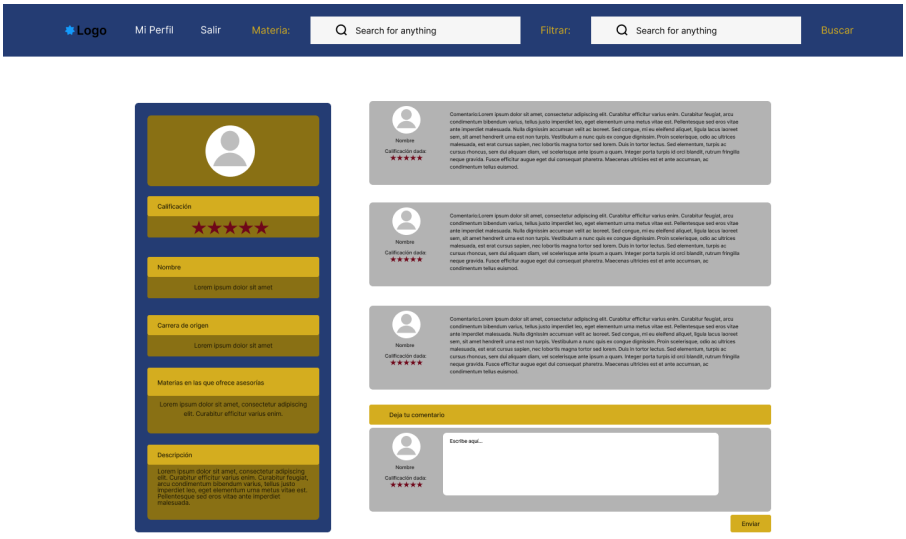

Figura 2: Vista de perfil del usuario

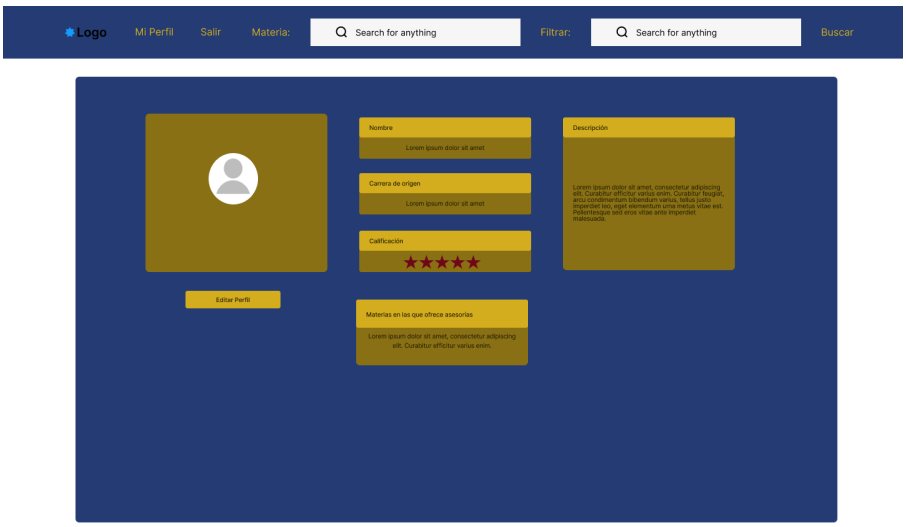

Figura 3: Vista de comentarios del asesor

5. Peticiones

5.1. Auth

Para las peticiones de autenticación, la ruta comienza con `/auth`, por ejemplo `localhost:8000/auth`, y tenemos 2 posibles peticiones.

5.1.1. `/login`

Esta ruta es para validar las credenciales de los usuarios.

No	Recurso	Método HTTP	Parámetro
1	`localhost:8000/auth/login`	POST	-

Endpoint

Datos de Entrada Header

No	Key	Value	Obligatorio	Comentario
1	Content-type	application/json	si	el tipo de contenido de retorno será JSON

Body
Ejemplo de body

```
{
  "correo": "juan@ciencias.unam.mx",
  "contrasena": "juanContrasena",
}
```

No	Nombre	Tipo	Obligatorio	Comentario
1	Correo	String	Si	El correo del usuario
2	Contrasena	String	Si	La contraseña del usuario

Datos de salida En caso de que la validación sea exitosa, devuelve un status 200 y una cookie para identificar al usuario; en caso de que la validación sea incorrecta devuelva un status 403.

5.1.2. `/registro`

No	Recurso	Método HTTP	Parámetro
1	`localhost:8000/auth/registro`	POST	-

Endpoint

Datos de Entrada Header

No	Key	Value	Obligatorio	Comentario
1	Content-type	application/json	si	el tipo de contenido de retorno será JSON

Body
Ejemplo de body

```
{
  "nombre" : "juan juarez",
  "correo": "juan@ciencias.unam.mx",
  "contrasena": "juanContrasena",
  "carrera" : "Ciencias de la computaci n"
  "info" : "Mi contacto es 5555555555"
}
```

Datos de salida En caso de que la validación sea exitosa, devuelve un status 201, en caso de fallar, un status 400 y un JSON con información del estado.

El JSON del error se ve de la siguiente manera:

```
{
  "success": false,
  "msg": "Un mensaje con informacion del error",
}
```

El JSON en caso de crearlo exitosamente tiene un atributo con success: true

No	Nombre	Tipo	Obligatorio	Comentario
1	Nombre	String	Si	El nombre del usuario
2	Correo	String	Si	El correo del usuario
4	Contrasena	String	Si	La contraseña del usuario
5	Carrera	String	Si	Es la carrera de origen del usuario
6	Asesorias	String	No	Es un arreglo con las materias en las que podria dar asesorias
1	Info	String	No	Información de contacto

5.2. Comentarios

Con estas peticiones podemos agregar o quitar comentarios en el perfil de algún asesor. La ruta comienza con `/api/comentarios`. Cuando la ruta es algo como `/:correo`, los 2 puntos indican que ahí va un correo, ejemplo `/api/comentarios/juan@ciencias.unam.mx`

5.2.1. `/:correo`

Tipo : GET, esto nos regresa todos los comentarios asociados a un correo, y regresa un arreglo de objetos, los objetos son los comentarios y se compone de 4 atributos: "to" a quien le dejaron el comentario, "comentario", el contenido del comentario, çalif', la calificación dada por el usuario y "id" que es el id que mongo le da al comentario y lo podemos usar para borrar el comentario.

Ejemplo peticion : GET localhost:4000/api/comentarios/juan@ciencias.unam.mx

Ejemplo respuesta:

```
[
{
"_id" : 42340923,
"to" : "juan@ciencias.unam.mx",
"comentario" : "No es claro"
"calif" : 1
},
{
"_id" : i0dn4irc,
"to" : "juan@ciencias.unam.mx",
"comentario" : "Excelente asesor a"
"calif" : 5
}
]
```

5.2.2. `/`

Tipo: POST Para hacer un comentario a un usuario, necesita "to" a quien va el comentario, "comentario" el contenido del comentario y çalif"la calificación del comentario.

Ejemplo peticion : POST localhost:4000/api/comentarios/ Ejemplo body:

```
[
{
"to" : "juan@ciencias.unam.mx",
"comentario" : "Esto es un comentario"
"calif" : 3
}
]
```

5.2.3. `/:id`

Tipo : DELETE, usando el id de un comentario, podemos borrarlo.

Ejemplo peticion : localhost:4000/api/comentarios/34384294

Devuelve status 200.

5.3. Usuarios

La ruta comienza con "/api/usuarios", aqui podemos buscar, modificar y borrar usuarios. Nota que crear usuarios se hace en Auth.

5.3.1. /MyProfile

Tipo : GET, esta ruta se usa para obtener la informacion del perfil del usuario con sesion activa, y uso la cookie que da el login para obtenerla, si no hay cookie entonces la peticion no es exitosa.

Ejemplo peticion : GET localhost:4000/api/usuarios/MyProfile

Ejemplo respuesta:

```
[
{
"to" : "juan@ciencias.unam.mx",
"comentario" : "Esto es un comentario"
"calif" : 3
}
]
```

5.3.2. /:correo

Tipo : GET Podemos obtener la información de un usuario con su correo, esta la uso para cuando le das click a la tarjeta de un asesor, poder mostrar su información.

Ejemplo petición : GET localhost:4000/api/usuarios/juan@ciencias.unam.mx

Ejemplo respuesta:

```
{
    "fotoPerfilId": "noProfile_jaqk9o",
    "_id": "62ba331706558a155a745f91",
    "nombre": "Juan J",
    "correo": "juan@ciencias.unam.mx",
    "carrera": "Ciencias de la Computacion",
    "asesorias": [
    "Mate 2",
    "Intro a CC",
    "Biologia"
    ],
    "info": "Me llamo juan prueba",
    "__v": 0
}
```

5.3.3. /asesores/:materia

con esta petición, obtenemos un arreglo de objetos que representan la información de todos los asesores que dan la materia buscada.

Ejemplo petición : GET localhost:4000/api/usuarios/asesores/Mate 2

Ejemplo respuesta:

```
   [
    {
    "fotoPerfilId": "noProfile_jaqk9o",
    "_id": "62ba331706558a155a745f91",
    "nombre": "Juan J",
    "correo": "juan@ciencias.unam.mx",
    "carrera": "Ciencias de la Computacion",
    "asesorias": [
    "Mate 2",
    "Intro a CC",
    "Biologia"
    ],
    "info": "Me llamo juan prueba",
```

```
    "__v": 0
    },
    {
    "fotoPerfilId": "noProfile_jaqk9o",
    "_id": "62ba332e06558a155a745f95",
    "nombre": "Pedro",
    "correo": "pedro@ciencias.unam.mx",
    "carrera": "Ciencias de la Computacion",
    "asesorias": [
    "Mate 2"
    ],
    "info": "Me llamo pedro",
    "__v": 0
    }
]
```


5.3.4. /:id

Tipo : PUT con esta petición, modificamos la información de perfil del usuario usando su id, es la que se usa en el botón actualizar de mi perfil. En el body, solo necesita como parámetros los elementos que se van a cambiar, no es necesario poner todos los atributos de un usuario.

Ejemplo peticion : PUT localhost:4000/api/usuarios/3493098

Ejemplo body:


```
{
"info" : "Estoy actualizando mi info",
"Asesorias" : ["Fisica"]
}
```


5.3.5. /:id

Tipo : DELETE con esto podemos borrar un usuario usando su id. Ejemplo peticion : DELETE localhost:4000/api/usuarios/94980432 Regresa un status 200.

6. Persistencia de Datos. Esquemas

6.1. Comentarios

```
const comentarioSchema = new Schema({
    comentario: {type: String, required: true},
    to: {type: String, required: true},
    calif: {type: Number, required: true}
});
```

Figura 4: Esquema del usuario

6.2. Usuarios

```
const usuarioSchema = new Schema(
    {
        nombre: {type: String, required: true},
        correo: {type: String, required: true, validate: custom},
        contrasena: {type: String, required: true},
        carrera: {type: String, required: true},
        // Esta es la foto por default
        fotoPerfilURL : {
            type: String,
            default: "https://res.cloudinary.com/dzya3fvwj/image/upload/v1674083608/noProfile_jaqk9o.png"
        },
        fotoPerfilId : {type: String, required: false},
        asesorias: [String],
        calif: Number,
        info: String
    }
);
```

Figura 5: Esquema del usuario