Proyecto Final Ciclo 3

Como se mencionó al inicio del curso, cada grupo de estudiantes recibe un reto en el cual deben trabajar para proponer una solución. Aquí se verán aplicados los conocimientos adquiridos y el producto final debe contar con la documentación asociada y la solución tecnológica implementada, a continuación se describe lo que se debe realizar para el desarrollo del proyecto, la parte inicial describe algunos comportamientos y la segunda parte indica requerimientos técnicos o de configuración que se deben encontrar en el código del proyecto.

Antes de iniciar, por favor lea atentamente la descripción e identifique posibles requerimientos:

El proyecto final del ciclo consta de una parte de investigación en donde los estudiantes deberán suponer que son una organización dedicada a una temática relacionado con el reto e identificar los productos o servicios que solucionan el problema planteado, esta tarea inicial de investigación puede ser consignada en diferentes documentos o anexos que deben estar relacionados con su respectiva tarea en Github. Cada equipo debe encontrar un mínimo de 3 soluciones que se convertirán en los servicios, productos o herramientas que ofrecen a través de su portafolio de servicios.

Portafolio de servicios

Cada grupo deberá generar un sitio web público (landing page) en donde se deben encontrar una sección superior con un menú, el menú debe contar con enlaces a las demás secciones del sitio (o páginas si han elegido crear este recurso) y contar con un botón de login. La sección superior también debe contar con un banner (con al menos 3 imágenes que cambien tipo carrusel) de temática acorde al tipo de soluciones que se ofrezca.

Una sección siguiente donde, de acuerdo a la información de su reto, describan de manera general en qué sector y que tipo de soluciones ofrecen como empresa, un texto corto, infografía o imagen donde expresen el porqué las soluciones que ustedes ofrecen son de alta calidad.

La siguiente sección deberá contar con los servicios que la empresa ofrece, estos servicios deben ser gestionados desde el backend del proyecto y deberán contar con un nombre o título, imagen acorde al tipo de servicio o producto y una descripción del mismo. Se recomienda crear una página independiente donde puedan detallar la investigación que se hizo acerca del producto y una mejor descripción de lo que ofrecerían o de la tecnología que utilizan en el servicio o producto ofertado.

Se deberá implementar una sección inferior donde creen algunos casos de éxito o testimonios de las soluciones que ustedes ofrecen, donde con una foto, un texto y el nombre de la persona y/o empresa, van a referenciar porque su solución fue exitosa.

Recuerden que estos datos son de dummy y deberán ser construidos por los integrantes del equipo.

Finalmente debe contar con un footer donde deberá estar toda la información de contacto, que podrían ser un par de correos electrónicos, las diferentes ciudades donde trabajan, un teléfono y celular (no utilizar números telefónicos reales), también contener un enlace al repositorio del proyecto final de github, el repositorio debe ser público.

Backend o Zona de Gestión

El backend o la zona de gestión, deberá contar con la administración de los servicios que se van a visualizar en el sitio web principal o **Portafolio de Servicios,** Los nombres de las rutas, componentes y demás aspectos técnicos están definidos en la sección **requisitos obligatorios estructura backend.**

El backend implementará el módulo de autenticación realizado en la semana 3 para validar los usuarios y sus roles, la zona de administración debe contar inicialmente con la gestión o CRUD de Artículos y Categorías (como dos elementos de nombre genérico independiente del servicios que se vaya a implementar) y posteriormente se implementará la gestión de usuarios allí mismo.

Para el manejo de rutas, se debe implementar el router-view con el fin de sacar provecho del poder de Vue y su característica de SPA.

El back deberá proveer una API que permitirá realizar las diferentes peticiones a sus componentes desde el landing page.

Requisitos Obligatorios Estructura Backend

Rutas

login Endpoint:

se debe contar un una ruta por medio de método post para el inicio de sesión de la siguiente manera:

```
'/api/usuario/login'
```

cuando esta ruta es consumida desde el frontend la api debe responder en tres casos diferentes :

1. Cuando el usuario se loguea exitosamente ,debe responder con un status 200 y propiedad tokenReturn de la siguiente manera :
   ```
   res.status(200).json({ user, tokenReturn });
   ```

2. El usuario no existe en la bases de dato, debe responder con un status 404 de la siguiente manera:
   ```
   res.status(404);
   ```

   esta respuesta se puede complementar con un mensaje como por ejemplo:
   ```
   res.status(404).send('User Not Found.');
   ```
3. El usuario ingresa una contraseña inválida, debe responder con un status 401 de la siguiente manera:
   ```
   res.status(401);
   ```
   esta respuesta se puede complementar con un mensaje y propiedades dependiendo de lo que espera recibir en el fontend como por ejemplo:
   ```
   res.status(401).send({ auth: false, accessToken: null, reason: "Invalid Password!" });
   ```

Categorías Endpoints:

listas de categoría:

```
'/api/categoria/list'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

agregar un nuevo categoría:

```
'/api/categoria/add'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

update categoría:

```
'/api/categoria/update'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

activar estado del categoría:

```
'/api/categoria/activate'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un status 200:

```
res.status(200)
```

desactivar estado del categoría:

```
/api/categoria/deactivate
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

Artículos Endpoints:

listas de artículos:

```
'/api/articulo/list'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

agregar un nuevo artículo:

```
'/api/articulo/add'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

update artículo:

```
'/api/articulo/update'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

activar estado del artículo:

```
'/api/articulo/activate'
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un status 200:

```
res.status(200)
```

desactivar estado del artículo:

```
/api/articulo/deactivate
```

cuando la solicitud se procesa correctamente el sistema deberá responder con un **status 200:**

```
res.status(200)
```

Nota: es obligatorio crear las rutas con estas características con el fin que el sistema ejecute satisfactoriamente las pruebas unitarias, obtener la calificación de cada grupo sin ningún problema.

bases de datos :

en el directorio config/config.json encontrarán las credenciales para la conexión de las bases de datos de la siguiente manera:

queda de elección de cada grupo utilizar la bases de datos localmente que deseen ya sea la predeterminada en el archivo o utilizar mysql con el sitio web Free MySQL como se explicó en las sesiones anteriores, estas modificacion solo se deben realizar en el objeto “**development**”, las otras por ningún motivo deben ser modificadas esto podría alterar el resultado de la prueba y por ende su calificación.

Modelos:

Modelo Usuario:

npx sequelize-cli model:generate --name Usuario --attributes rol:string,nombre:string,password:string,email:string,estado:integer

como podemos observar tenemos nuevos atributos uno de ellos es rol , lo utilizaremos para manejar restricciones de usuario según su rol, los roles serán:

- Administrador:con acceso total al sistema
- Vendedor:solo para acceso al módulo de ventas.
- Almacenero:solo acceso al módulo de ingresos ,artículos y categorías.

Modelo Categoría:

npx sequelize-cli model:generate --name Categoria --attributes nombre:string,descripcion:string,estado:integer

Modelo Artículos:

npx sequelize-cli sequelize model:generate --name Articulo --attributes codigo:string,nombre:string,descripcion:string,estado:integer,categoriaId:integer

Nota: es obligatorio desarrollar estos modelos con estas características con el fin de que los seeders del sistema se ejecuten satisfactoriamente para realizar las pruebas unitarias y obtener la calificación de cada grupo sin ningún problema,si se requiere añadir más campos asegúrese que sean no obligatorios.

Finalmente, como todos los proyectos, deben ser subidos a un repositorio en Github y gestionar sus tareas a través de la misma herramienta utilizando los boards y los issues.

Para iniciar su proyecto debe descargar el siguiente repositorio:
https://github.com/Tecnalia-Cilco-3/semana-4

Páginas de Ayuda

A continuación se listan algunas páginas web que pueden contener secciones interesantes que servirían como una base para la elaboración del mockup de lo que buscaremos hacer para la landing page:

- MinTic [https://www.mintic.gov.co/portal/inicio/]
- MisionTic [https://www.misiontic2022.gov.co/portal/]
- Datos Abiertos [https://www.datos.gov.co/]
- Globant [https://www.globant.com/]
- Interacpedia [https://interacpedia.com/]
- Fortinet [https://www.fortinet.com/lat]
- Cámara de comercio de Pereira [https://www.camarapereira.org.co/es/]
- Tecnalia Colombia [http://tecnaliacolombia.org/]