Ejercitación integral DDS

- Desarrollar el **backend** y **frontend** de una aplicación utilizando 2 instancias de Visual Studio Code 1 para el back en port 4000 y la otra para el front en port 3000. Se deberá generar una funcionalidad nueva a los proyectos existentes:
- El proyecto del backend está en el siguiente repositorio GIT **https://github.com/arcba/DDSBack.git**
- El proyecto del frontend está en el siguiente repositorio GIT **https://github.com/arcba/DDSFront.git**

Clonar ambos repositorios en la carpeta ***tmp*** de la estación de trabajo en linux. Y realizar las modificaciones solicitadas en dichos proyectos.

Importante para instalar las librerías dependientes del back y del front:

1. Eliminar (si existe) el archivo package-lock.json en ambos proyectos
2. ejecutar el comando **npm install** en ambos proyectos (misma ruta donde está el archivo **package.json**).

Tiempo de resolución: 1h 30 minutos.

Desarrollo del back: se deberá agregar una funcionalidad de consulta para un recurso nuevo "clientes**"**

- ✓ Se deberá desarrollar el código para conectar con la BD, crear la tabla si no existe e insertar los valores asignados. En el archivo script.sql (que está en la raiz del proyecto) están los script sql necesarios.
- ✓ Crear el modelo de sequelize que mapee la tabla clientes, la cual tiene la siguiente estructura:

```
CREATE table clientes(
    IdCliente INTEGER PRIMARY KEY AUTOINCREMENT
    , ApellidoYNombre TEXT NOT NULL UNIQUE
```

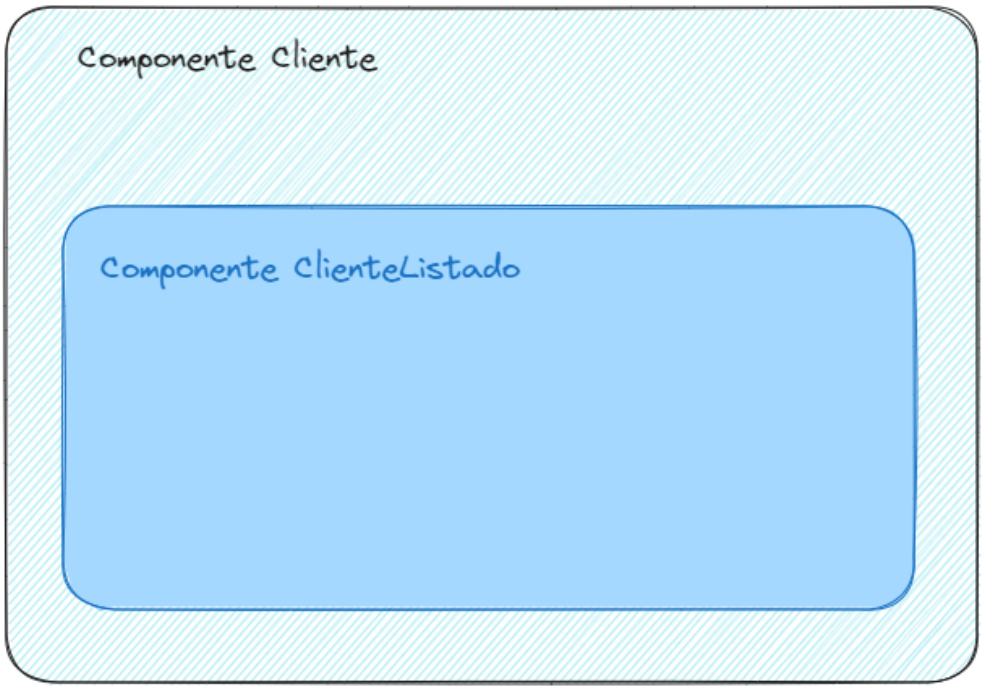
```
, DNI INTEGER NOT NULL
);
```

✓ se debe programar para la ruta clientes lo siguiente:

- Un endpoint Clientes que mediante el verbo/metoto **GET** reciba un parámetro: ApellidoYNombre y utilizando el model ya desarrollado recupere los registros filtrando por el parámetro recibido. Deberá devolver todos los registros en cuyo campo ApellidoYNombre contenga el valor del parámetro recibido. Devolver todos los campos de la tabla
- La aplicación deberá registrar como middleware un **router** con la ruta "/api/clientes".

Desarrollo del frontend: se deberá agregar una interface de consulta para el recurso "clientes" desarrollado en el backend

- ✓ Debe desarrollar un componente Cliente con el siguiente esquema:



- ✓ Agregar el código necesario para poder realizar la consulta de clientes. Se deberá realizar los componentes de react y consumir el backend desarrollado en el punto anterior.
- ✓ En el componente Cliente, incluya la interface de búsqueda con el campo a filtrar y el botón buscar.

  Deberá cumplir los siguientes requisitos:

  - Consumir vía axios el endpoint del back
  - Utilizar para el formulario de búsqueda la librería react-hook-form
- ✓ En el componente ClienteListado incluya una tabla de html para mostrar los resultados de la búsqueda, incluya en la misma todos los registro y sus campos, (no se pide paginación)

- ✓ Agregar al componente menú de la aplicación, el acceso al nuevo componente.

Para la entrega del ejercicio, realizar dos archivos .ZIP, uno para cada proyecto, sin incluir la carpeta **node_modules**: front_legajo.zip y back_legajo.zip (reemplazar legajo por su numero de legajo)

TIPS PARA RESOLUCION:

Observaciones back:

- El back ya está configurado para trabajar en el puerto 4000 (ver index.js)

```
const port = 4000;
app.listen(port, () => {
  console.log(`Servidor iniciado en el puerto ${port}`);
});
```

- recordar que el backend no inicia el browser, si quiero testearlo debo abrir el browser explícitamente o usar postman
- Si la tabla pymes ya existe, debemos borrarla para que se autogenere y se ejecute el nuevo código que permita crear la tabla artículos.
- en Sequelize las validaciones de las propiedades de los modelos solo se usan en altas y modificaciones… para una consulta no hacen falta implementarlas.
- el back podemo ejecutarlo con: npm run dev
- Paso a paso simulacro back:
    1 - modificar sqlite-init.js (para crear y cargar la tabla según script.sql). No olvidar exportar la definición de cliente al finalizar
    2 - modificar sequelize-init.js (para crear el modelo cliente)
    3 - crear el /routes/clientes.js (para crear el método GET que reciba el parámetro y devuelva los registros que coincidan)
    4 - modificar el index.js para que se monte la ruta creada en punto anterior

Observaciones Front:

- equipos debian con problemas al ejecutar npm install
  deberan ejecutar el comando: npm config delete registry (elimina el proxy golum)
- El front arranca por defecto en el puerto 3000
- Asegúrese al probar el front que el código del back está en ejecución
- Por simplicidad todos los componentes están en una única carpeta!
- No se pide implementar servicios, pero no está mal si así lo hiciera.
- ***paso a paso simulacro front***
    - ***crear Clientes.jsx desde articulos.jsx***
    - ***crear ListadoClientes.jsx desde ListadoArticulos.jsx***
    - ***agregar a Menu.jsx el link a clientes***
    - ***en app.js importar clientes.jsx y agregarlo route***
    - `<Route path="/clientes" element={<Clientes />} />`