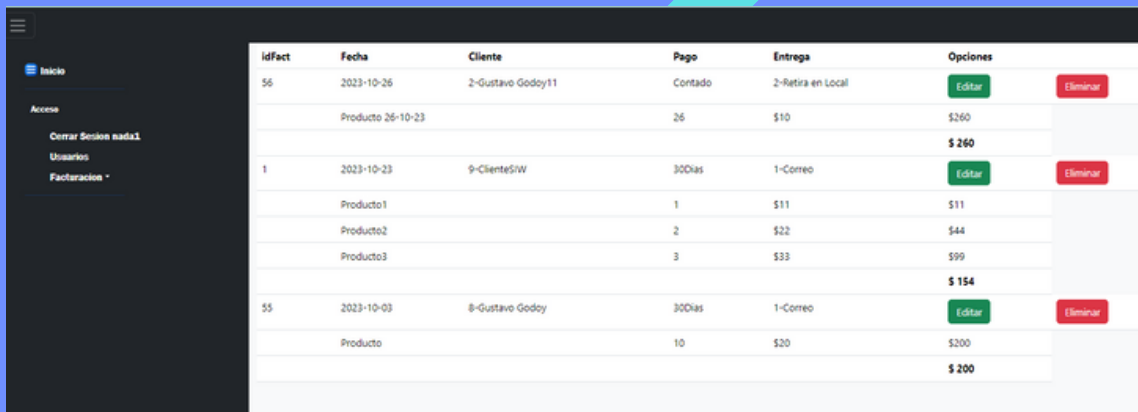




Octubre 2023

PRUEBA-FULLSTACK

EVIDENCIA: Prueba tecnica SIW cargo



The screenshot shows a web application interface. On the left is a dark sidebar with a menu containing 'Inicio', 'Cerrar Sesión nada1', 'Usuarios', and 'Facturacion'. The main area displays a table with columns: 'IdFact', 'Fecha', 'Cliente', 'Pago', 'Entrega', and 'Opciones'. The table lists three orders with their details and sub-items.

IdFact	Fecha	Cliente	Pago	Entrega	Opciones
56	2023-10-26	2-Gustavo Godoy11	Contado	2-Retira en Local	Editar Eliminar
		Producto 26-10-23	26	\$10	\$260
					\$ 260
1	2023-10-23	9-ClienteSIW	30Dias	1-Correo	Editar Eliminar
		Producto1	1	\$11	\$11
		Producto2	2	\$22	\$44
		Producto3	3	\$33	\$99
					\$ 154
55	2023-10-03	8-Gustavo Godoy	30Dias	1-Correo	Editar Eliminar
		Producto	10	\$20	\$200
					\$ 200

Nombre: Gustavo Godoy
Email: gustavog@live.com.ar



El candidato realizara una aplicación donde pueda registrarse como usuario, iniciar sesión y listar la factura luego de iniciar sesión. Desarrollando tanto el backend, como el frontend.

Backend:

La aplicación ni bien inicia, aparece el login. Este se vincular a una base de datos, si el Email o contraseña no coincide, muestra un cartel informado error de usuario:

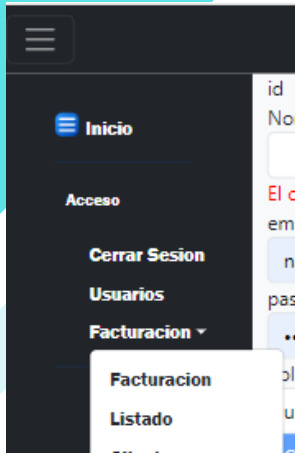
El registro se puede acceder directamente desde el login o desde el menú lateral. Una vez logeado, el menú de registro, desaparecerá.

Cuando ya se esta logeado, la App, nos mostrara un nuevo menú. Este se puede seguir filtrando según roles, para este ejemplo, solo se deja un rol de administrador.

En el menú “Usuario” podemos ver todos los usuario, editar y eliminar. Además, poder cargar un nuevo.

El formulario tiene características de validación. Por ejemplo, el botón quedara invalido, hasta que no se cumpla el control del formulario, como indica la imagen, no dejar el campo vacío, ya que es required.

id	name	email	password	rol	Opciones
3	nada1	nada2@gmail.com	nada	user	Editar Eliminar
29	name	email@nada.com	password	user	Editar Eliminar
52	gustavog	gustavog@live.com.ar	Cata1908	admin	Editar Eliminar
56	Laura	Laura@gmail.com	Laura5	admin	Editar Eliminar
57	Nombre nuevo	NombreNuevo@gmail.com	NombreNuevo	user	Editar Eliminar
58	gustavogodoy2	gustavog@live.com.ar	gustavog	user	Editar Eliminar



En el menú se agregó una lista de facturaciones y sus relaciones, como por ejemplo clientes.

El formulario de facturación, solo para este ejemplo, los ítems de producto se agregan en la misma base de datos. En la realidad, son dos formularios distintos relacionado por un id principal, que sería en este caso la factura.

En el form, se agregó lista desplegable y conectada a una base de datos. Por ejemplo, clientes puede aumentar o salir de la lista, según necesidad del usuario. Los campos deben llenarse y la app, dará aviso cual es el error, y no permitirá avanzar.

La cantidad y precio se multiplica y suma en un total, además de realizar una suma total dinámica. Este si se quisiera se podría hacer en un view de Mysql.

Facturacion

Fecha* 27/10/2023

Cliente*

Forma de pago

Forma de Entrega*

Observacion

El campo es requerido

El campo es requerido

Item de facturacion

#	Producto	Cantidad		Total \$
1	Prueba de producto	10	20	200
2	<input type="text"/>	<input type="text"/>	\$	\$
3	<input type="text"/>	<input type="text"/>	\$	\$
4	<input type="text"/>	<input type="text"/>	\$	\$
				200

Facturar

Clientes como facturación, se pueden listar y hacer el CRUD que se necesite. En el caso de la factura, cuando se presiona el botón editar, aparece el formulario de edición, una vez editado se vuelve a ocultar.

En la app, solo esta ejemplificado la edición del encabezado de la factura, no habría problemas de agregar todos los productos, como como mencione anteriormente, por una cuestión de tiempo, no se llega a realizar dos form que cuando se inserte ambos datos, se relacionen por un numero único, en este caso el número de factura. Para el buscador, tengo ejemplos realizados con filter.

IdFact	Fecha	Cliente	Pago	Entrega	Opciones
56	2023-10-26	2-Gustavo Godoy11	Contado	2-Retira en Local	<button>Editar</button> <button>Eliminar</button>
		Producto 26-10-23	26	\$10	\$260
					\$ 260
1	2023-10-23	9-ClienteSW	30Dias	1-Correo	<button>Editar</button> <button>Eliminar</button>
		Producto1	1	\$11	\$11
		Producto2	2	\$22	\$44
		Producto3	3	\$33	\$99
					\$ 154

ClienteSW	IdCliente	ClienteSW	Domicilio	Localidad	Opciones
Laura Figueroa1	9	ClienteSW	Domicilio	1-Cordoba Capital	<button>Editar</button> <button>Eliminar</button>
Domicilio					
General Paz 747	2	Gustavo Godoy11	Cordoba	1-Cordoba Capital	<button>Editar</button> <button>Eliminar</button>
Localidad					
Monte Buey	11	Cata Godoy	General Paz 747	3-Hernando	<button>Editar</button> <button>Eliminar</button>
	10	Laura Figueroa1	General Paz 747	4-Monte Buey	<button>Editar</button> <button>Eliminar</button>
	8	Gustavo Godoy	Cordoba	4-Monte Buey	<button>Editar</button> <button>Eliminar</button>
	3	ClienteSW2	Domicilio4	4-Monte Buey	<button>Editar</button> <button>Eliminar</button>



La base de datos esta creada y gestionada con MYSQL. En la misma se genero las relaciones y vistas para los listado:

The screenshot shows a MySQL database interface with the following components:

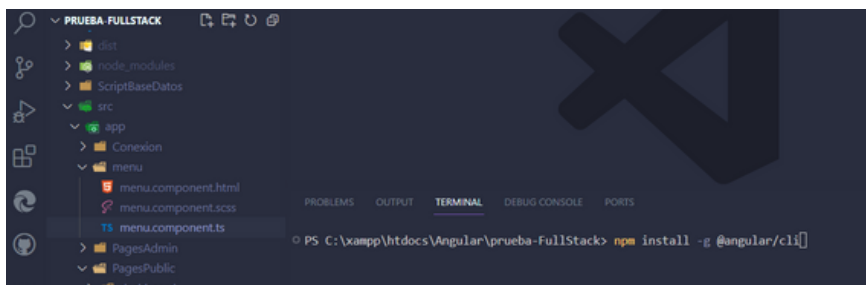
- Database:** c2110488_Prlsps
- Table:** SIWFacturacion
- Columns:** idFact (int(11)), NumFact (int(11)), Fecha (date), Cliente (int(11)), Pago (varchar(20)), Entrega (int(11)), Producto1 (varchar(50)), Producto2 (varchar(50)), Producto3 (varchar(50)), Producto4 (varchar(50)), Cantidad1 (decimal(6,0)), Cantidad2 (decimal(6,0)), Cantidad3 (decimal(6,0)), Cantidad4 (decimal(6,0)), Precio1 (decimal(6,0)), Precio2 (decimal(6,0)), Precio3 (decimal(6,0)), Precio4 (decimal(6,0)), Observacion (varchar(100)).
- Relationships:**
 - idFact (SIWFacturacion) to idCliente (SIWCliente)
 - idFact (SIWFacturacion) to idEntrega (SIWEntrega)
- Table Structure:**
 - SIWFacturacion:** idFact (int(11)), NumFact (int(11)), Fecha (date), Cliente (int(11)), Pago (varchar(20)), Entrega (int(11)), Producto1 (varchar(50)), Producto2 (varchar(50)), Producto3 (varchar(50)), Producto4 (varchar(50)), Cantidad1 (decimal(6,0)), Cantidad2 (decimal(6,0)), Cantidad3 (decimal(6,0)), Cantidad4 (decimal(6,0)), Precio1 (decimal(6,0)), Precio2 (decimal(6,0)), Precio3 (decimal(6,0)), Precio4 (decimal(6,0)), Observacion (varchar(100)).
 - SIWCliente:** idCliente (int(11)), ClienteSIW (varchar(50)), Domicilio (varchar(50)), Localidad (int(11)).
 - SIWEntrega:** idEntrega (int(11)), FormaEntrega (varchar(50)).
- Data Table:**

idFact	idCliente	ClienteSIW	Domicilio	Localidad
2	Gustavo Godoy11	Cc		
9	ClienteSIW	Dc		
11	Cata Godoy	Ge		
3	ClienteSIW2	Dc		
8	Gustavo Godoy	Cc		
10	Laura Figueroa1	Ge		

Link de repositorio: <https://github.com/gustavogidearg1/prueba-FullStack>

IMPORTANTE: Para poder probar angular en el servidor local, debe agregar los siguientes comandos en el powershell:

- **npm install -g @angular/cli**
- Luego instalada la carpeta con toda la biblioteca de node_modules, agregar el siguiente comando: **ng serve -o**



Link app subida a la web, conectada con la misma base de datos y en funcionamiento : <https://planidear.com.ar/prueba-FullStack/>

La App, es response (Adaptada a todas las pantallas), con estilos scss y Bootstrap.

El backEnd se utilizo php y base de datos Mysql. Con Angular a través de las API, se puede usar cualquier backEnd, como por ejemplo C# o Java.

El FondEnd se realizo con Angular, lenguaje TypeScript, de JavaScript. Como hable en la entrevista, este sería la evolución de React.

Se utilizó la importación librerías de Nodejs.

En los servicios, se utiliza los constructores GET, POST y PUT.

Las rutas son iguales a React, además, luego de logearse a través de la funcione, redirección a través de Router.