## Proyecto Calculadora CPS

El proyecto consiste en una **calculadora de precios de paquetes** desarrollada en **Angular**, conectada a una **API** que gestiona la lógica de cálculo y la persistencia en **SQL Server**. El sistema requiere un **archivo de configuración de tarifas** y una base de datos para almacenar clientes, cotizaciones y reglas de cálculo. La aplicación debe contemplar requisitos de **tarifa base por zona y servicio**, cálculo de **peso real vs. peso volumétrico**, aplicación de **recargos** (ruralidad, combustible, manejo especial), **seguros opcionales** y **IVA**, siguiendo una fórmula que combina estos elementos para obtener el costo final del envío.

Requisitos clave:

* Frontend en Angular con formulario de cotización y visualización de resultados.
* API para cálculo y almacenamiento de cotizaciones.
* Base de datos en SQL Server con tablas para clientes, tarifas y cotizaciones.
* Fórmula de cálculo:
  + (peso\*tarifa) + 1.66 \* Alto \* largo \* ancho – descuento cliente \* 0.5 \* peso

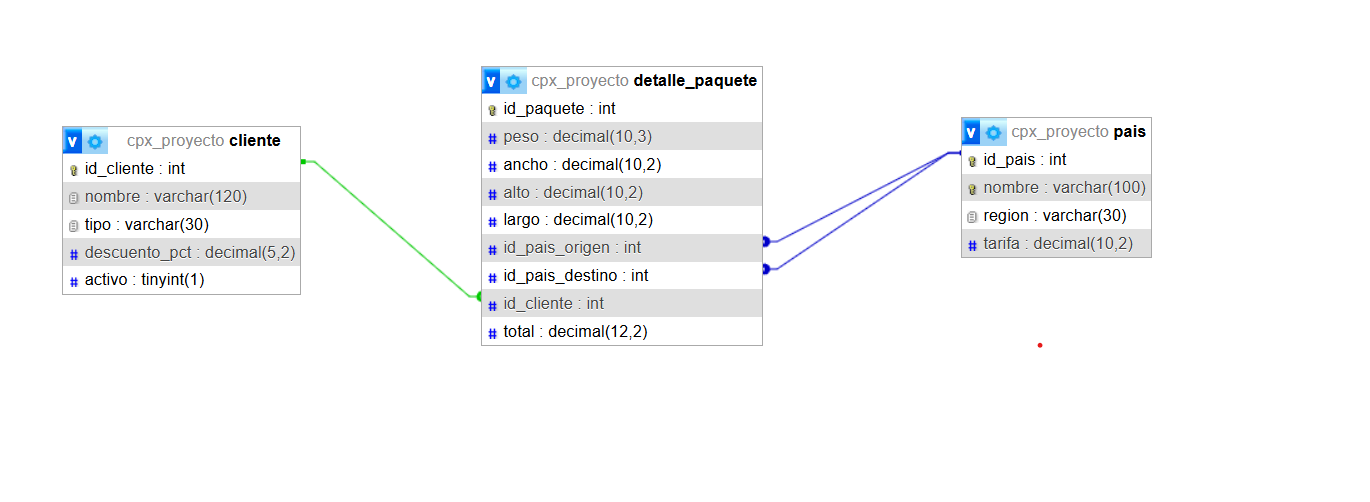
## Base de datos:

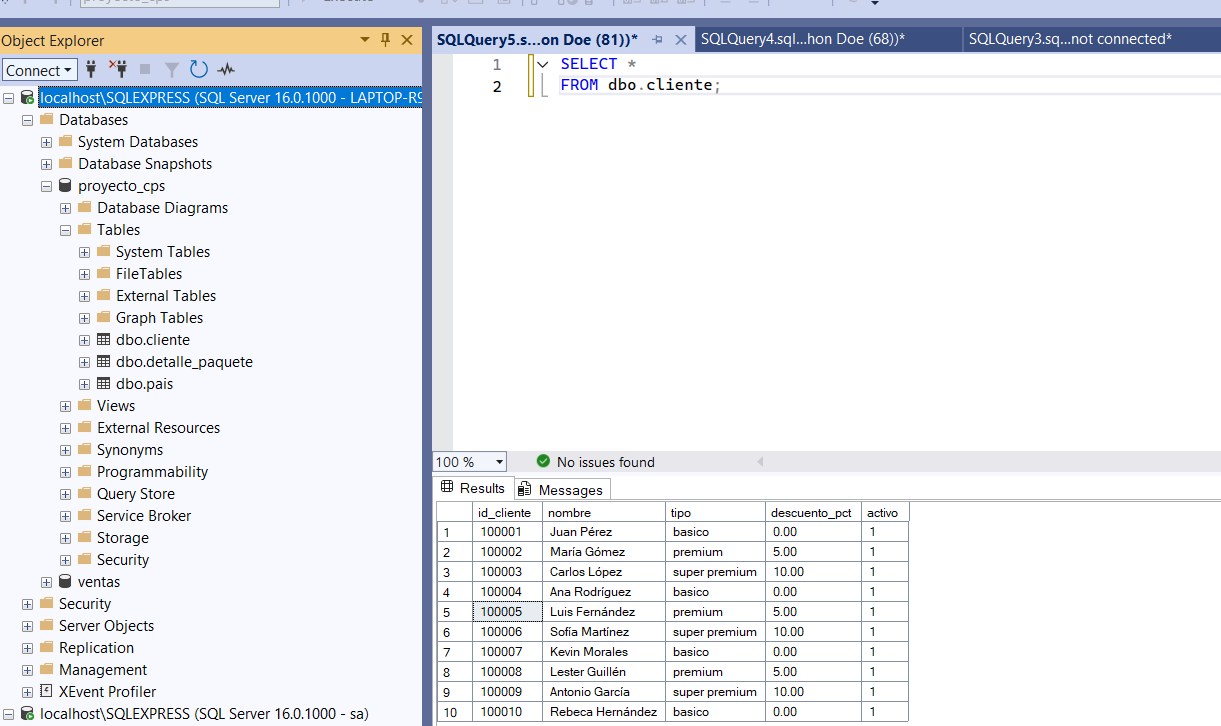
En base a los requerimientos se empezó a maquetar la base de datos, definiendo los campos necesarios según las funciones de la aplicación, con la limitante de utilizar únicamente tres tablas. Primero se realizó un diseño en Excel y luego la implementación en SQL Server.

Se decidió almacenar el resultado del cálculo en la tabla **DetallePaquete**, mantener una lista de **Clientes** con su tipo (para aplicar en el futuro la fórmula de descuentos), y una tabla de **Países** con sus tarifas (para relacionarlas en el cálculo final).

El **IdCliente** y el **IdPais** se declararon como llaves foráneas en la tabla DetallePaquete. Además, se definió el **IdCliente** como número de afiliado asignado a cada cliente, de modo que este mismo identificador sirva para aplicar los descuentos correspondientes.





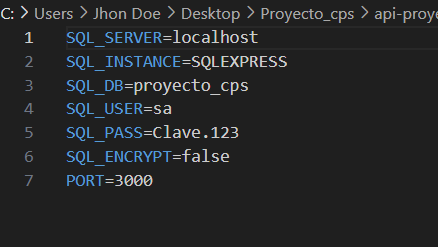


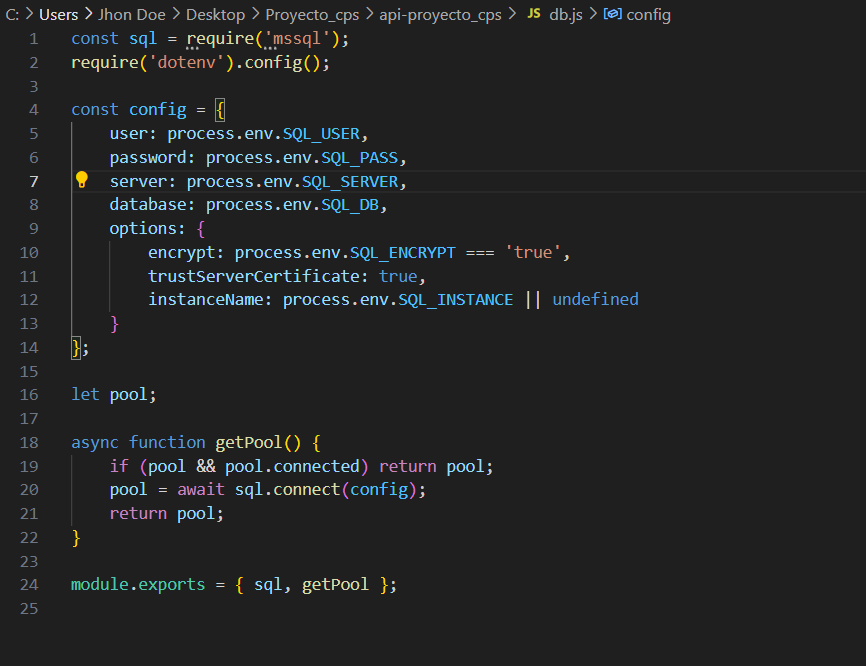
## Desarrollo Api:

Una vez que la base de datos estuvo lista en **SQL Server**, desarrollamos la **API** para conectarnos y trabajar con la información desde Angular. Para ello instalamos tres dependencias principales:

* **express**: para crear la API y manejar las rutas.
* **mssql**: para conectarnos a SQL Server y ejecutar consultas.
* **dotenv**: para guardar las credenciales de la base de datos en variables de entorno.

### Pasos para la conexión

1. **Crear archivo de configuración de la conexión**  
   
2. Establecer la conexión en la API – en el archivo db.js creamos el archivo que abre la conexión



1. Levantamos la api – importamos las rutas e iniciamos el serivodor



En el archivo server.js declaramos los métodos GET y POST.

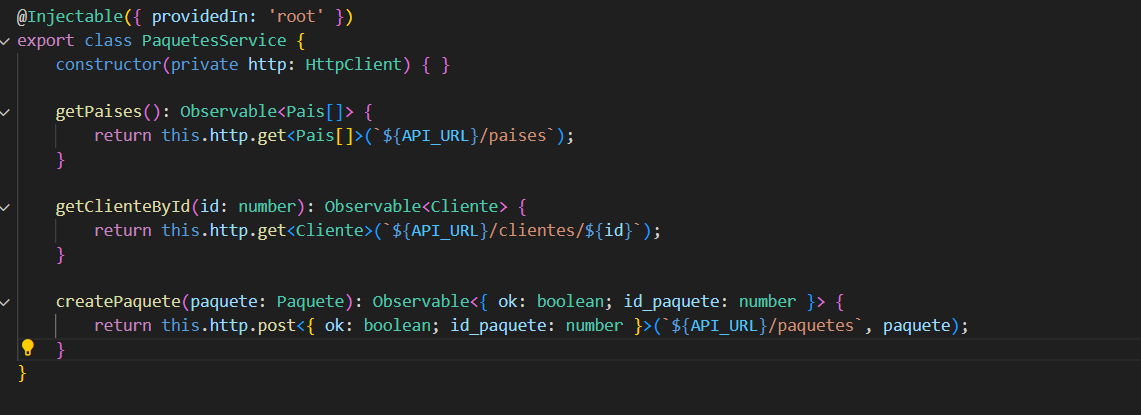
Con GET obtenemos la información de la tabla Paises (tarifas) y de la tabla Clientes (datos y descuentos).

Con POST insertamos en la base de datos el registro en DetallePaquete, guardando la información del envío y el total calculado.

## Angular – API

Luego, para llamar a la API desde Angular, declaramos dentro del archivo **paquetes.service.ts** los siguientes métodos HTTP hacia los endpoints:

* **getPaises()**: se conecta al endpoint /paises y devuelve la lista de países junto con sus tarifas, lo que utilizamos para llenar el combo en el formulario.
* **getClienteById(id)**: se conecta al endpoint /clientes/:id y nos devuelve los datos del cliente, incluyendo el porcentaje de descuento asignado.
* **createPaquete(body)**: envía al endpoint /paquetes toda la información del envío usando POST y guarda el detalle en la base de datos, devolviendo el id\_paquete generado.

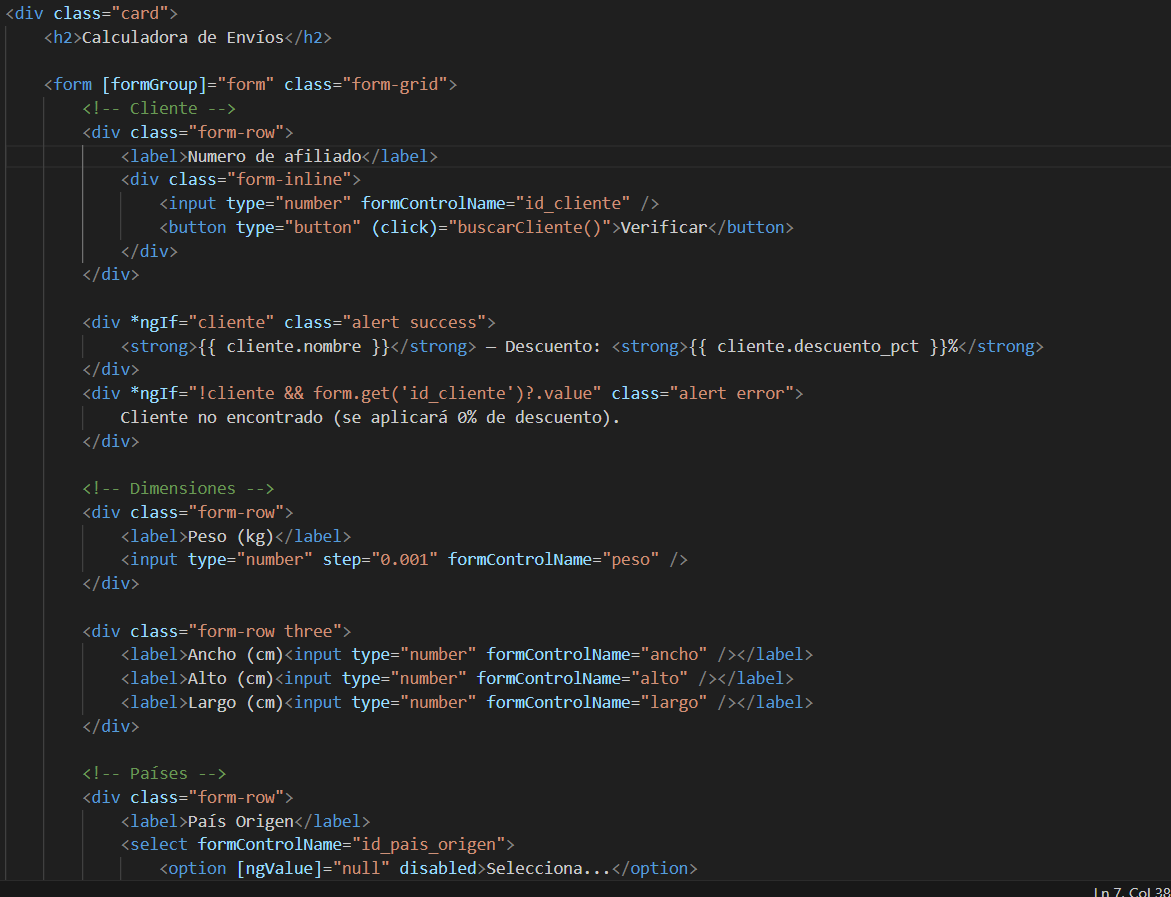


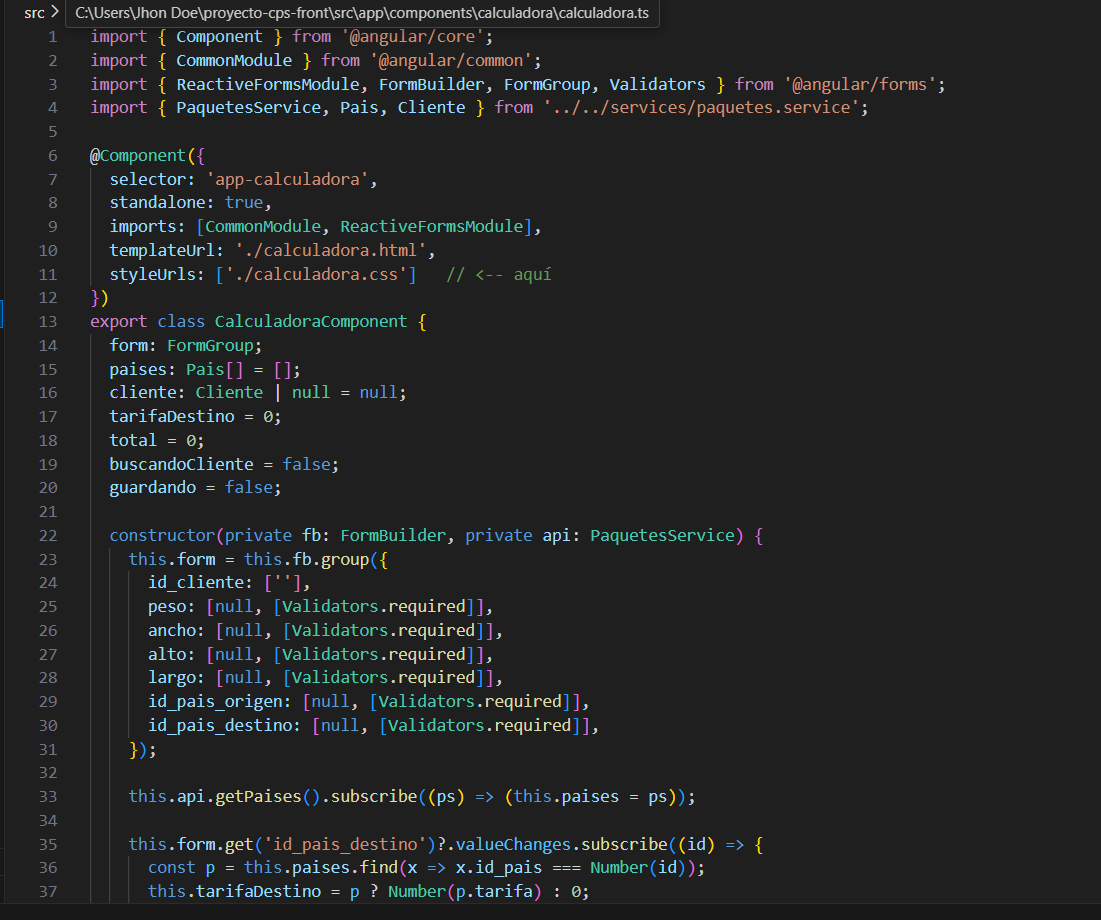
## Formulario Angular:

Para el formulario creamos el componente Calculadora, donde están los archivos calculadora.component.ts y calculadora.component.html. En este componente configuramos un formulario reactivo con los siguientes controles:

* id\_cliente: número de afiliado (opcional).
* peso: peso del paquete en kilogramos.
* ancho, alto, largo: dimensiones del paquete en centímetros.
* id\_pais\_origen y id\_pais\_destino: para seleccionar desde qué país y hacia qué país se enviará el paquete.

En el archivo calculadora.component.html vinculamos cada <input> y <select> utilizando la propiedad formControlName, lo que permite almacenar automáticamente los datos en this.form dentro de calculadora.component.ts. De esta manera, ya tenemos disponibles todos los valores del formulario para usarlos en las validaciones y finalmente enviarlos a la API.





# Descuento Cliente

El cálculo del descuento del cliente lo trabajamos en el archivo calculadora.component.ts, dentro del método buscarCliente().

Obtenemos el id\_cliente directamente del formulario que hicimos en calculadora.component.ts y enlazado con los inputs del archivo calculadora.component.html.

Con ese valor, llamamos al servicio que está en paquetes.service.ts, específicamente al método getClienteById(id), que al mimso timeo hace la petición a la API que declaramos en el archivo server.js.

Los datos que devuelve la API contiene los datos del cliente y el campo descuento\_pct. En el componente calculadora.component.ts guardamos esos datos en la variable this.cliente para poder utilizarlos.

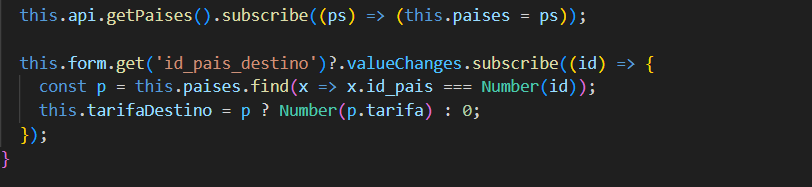
Cuando realizamos el cálculo final del envío tomamos ese descuento\_pct desde this.cliente y lo aplicamos sobre el subtotal o el total que corresponda, de manera que el descuento queda integrado en la fórmula final del costo del paquete.

Si el cliente no existe devuelve un valor nulo y por lo tanto el calculo será en base a 0.

# Tarifa Destino

Para obtener la tarifa del destino trabajamos en el archivo calculadora.component.ts.  
Al iniciar el componente llamamos al servicio con this.api.getPaises() y guardamos la lista completa de países en la variable this.paises.

Luego, cuando el usuario selecciona un id\_pais\_destino desde el formulario, buscamos ese país dentro de la lista this.paises y extraemos su tarifa. Esa tarifa se guarda en la variable this.tarifaDestino, que después se utiliza en la fórmula del cálculo final del paquete.



# Calculo Total

El cálculo del total se hace en el archivo calculadora.component.ts, dentro del método calcular().  
Primero tomamos del formulario los valores de peso, ancho, alto y largo. Con esos datos calculamos el peso volumétrico usando la fórmula (ancho × alto × largo) / 5000.

Luego determinamos el peso facturable, que siempre será el valor mayor entre el peso real y el peso volumétrico. A partir de ahí calculamos la base, que es igual a tarifaDestino × peso facturable.

Después aplicamos el descuento del cliente tomando el campo descuento\_pct. Si el cliente no existe, el descuento es 0. El total final se obtiene con la fórmula base × (1 - descuento/100), redondeado a dos decimales.

Este resultado se muestra en pantalla y es el mismo que se enviará a la base de datos cuando guardemos el paquete. En caso de que en el futuro se quiera hacer el cálculo directamente en el servidor, la misma fórmula se podría copiar dentro del endpoint POST /paquetes en el archivo server.js.