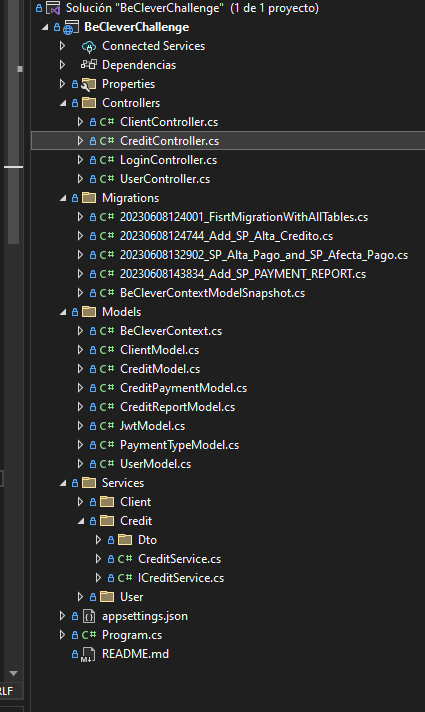
**Estructura del codigo:**

Controladores: Utilizamos los verbos get y post para declarar las rutas de nuestras Api, pero la logica de las consultas la delegamos a los servicios.

Servicios: Contienen la logica de negocio y la relacion con el contexto de la base de datos.

Models: Modelamos la base.

Migrations: Establecemos migraciones para modelar nuestra base en modo Code First.



**Endpoints – Utilizar postman para mejores resultados**

Todos los endpoints comienzan con el prefijo del puerto asignado por el localhost seguido de “api” por ejemplo en el caso de correr localmente es “**https://localhost:7119/api/Login**”

Login: “/api/login” (Post)

Body:(Json)

{

"userName": "Admin",

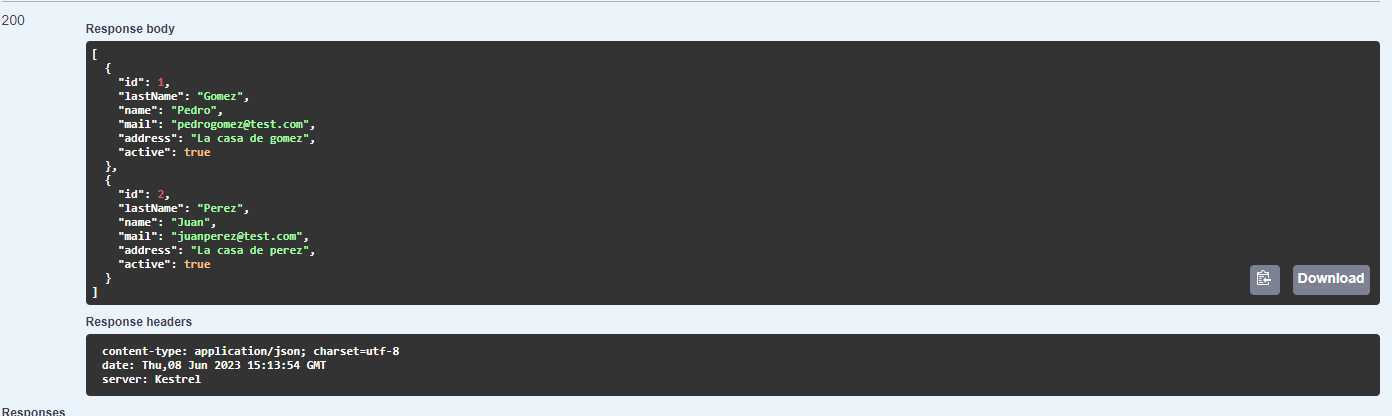
"password": "admin123"

}

Al ejecutar las migraciones de la base de datos se crea un usuario de prueba con las credenciales descriptas. El logeo retornara un token que luego debe ser copiado y colocado como autorizacion “Baerer-Token” y caducidad de 1hr.

Consulta Clientes: “/api/Client/GetAllClient” (Get)

Al ejecutar las migraciones de la base de datos se crea dos clientes de prueba. Dicha consulta sirve para obtener el id del cliente que usaremos para crear un credito.



Consulta Tipo de pagos: “/api/Credit/GetAllPayments” (Get)

Al ejecutar las migraciones de la base de datos se crea tres tipos de pagos (Efectivo, Debito y Credito). La consulta sirve para obtener el Id de tipo de pago para el alta del pago.

A picture containing software, multimedia software, screenshot, text

Description automatically generated

Consulta Tipo de Credito por Cliente: “/api/Credit/GetCreditByClient” (Get)

Conociendo el id del cliente podemos saber si el mismo tiene creditos.

Crear Credito: “/api/Credit/CreateCredit” (Post)

Body:(Json)

**{**

**"amount": 0, (Monto del credito sin intereses en enteros con dos decimales)**

**"interestPercent": 0, (Porcentaje de interes respect al monto en enteros)**

**"quantityQuote": 0, (Cantidad de cuotas en enteros)**

**"iva": 0, (porcentaje de iva en enteros)**

**"expirationDay": 0, (dia del mes que expira la cuota y comienzan los intereses del 1 al 31)**

**"delayInterestPercent": 0, (porcentaje del monto que se incrementa el valor de la cuota por interes diario ej:0.01)**

**"userId": 0, (Usuario que da de alta el credito en este caso siempre 1)**

**"clientId": 0 (cliente id, en este caso 1 o 2)**

**}**

Al insertar el Stored Procedure se encarga de calcular:

AmountWithInterest: Es elm onto total sumado a los intereses y el iva.

PureQuoteValue: Valor de la cuota pura respecto al monto.

Quote: Valor de la cuota respecto al monto con intereses.

Obtener Valor de Cuota Credito: “/api/Credit/GetCreditQuote” (Get)

Body:(Json)

{

“Id”: int

}

Retorna el valor de la cuota que debe pagar seguin el id del credito. Si la consulta es en una fecha mayor al dia que expira la cuota se le aplicara el porcentaje de interes antes mencionado.

Agregar un pago de cuota: “/api/Credit/AddPayment” (Post)

Body:(Json)

**{**

**"creditId": 0, Habiendo consultado el credito por cliente tiene conocimiento del id del credito)**

**"amount": 0, (Habiendo consultado la cuota puede colocar el monto)**

**"payType": 0, (Habiendo consultado los tipos de pago tiene conocimiento del id del mismo)**

**"operationNumber": "string" (Referencia del pago que suele ser codigo de pago)**

**}**

Al insertar el Stored Procedure se encarga de calcular:

Quote: El numero de cuota que registra. Y hace la actualizacion en la tabla de creditos respecto a que cuota va pagando.

Tambien si el pago llega a la cuota estipulada en las cantidades de cuotas del credito, el proximo pago ya no se imputara y se dara por finalizado el credito.

Obtener Valor de Cuota Credito: “/api/Credit/GetCreditReport” (Get)

Body:(Json)

{

“Desde”: datetime,

“Hasta”: datetime

}

Arroja los creditos comprendidos entre esas fechas con el campo extra de la suma de sus pagos correspondientes.