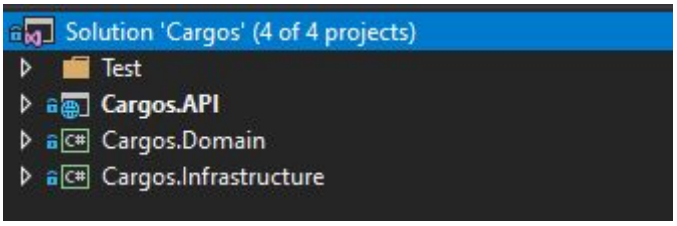
Documentación

Solución

En la raíz del repositorio se encuentran 2 proyectos que son las APIs Cargos y Pagos.

Cada una de las api tiene la siguiente estructura:

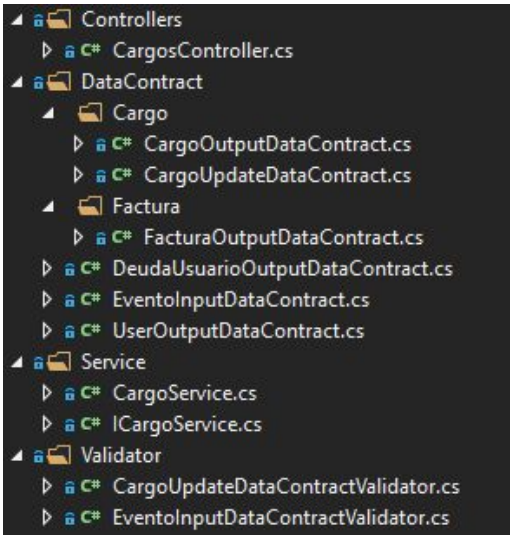


[Test]
Contiene los test de los repositorios.

[Cargos/Pagos].Domain
Contiene las entidades que se modelan.

[Cargos/Pagos].Infrastructure
Contiene la configuración de la base de datos.
Se encarga de persistir y recuperar las entidades en la base.

[Cargos/Pagos].API
Contiene la interfaz de la api, los datacontracts para la comunicación y realiza las validaciones de los request.



Tecnología

- El proyecto está desarrollado sobre el framework Net Core 2.1
- C # como lenguaje
- EntityFrameworkCore como ORM
- Como IoC usa el contenedor por defecto de Net Core
- Los test utilizan MSTest
- El motor de base de datos es SQL Server

Ejecutar localmente

Requisitos:

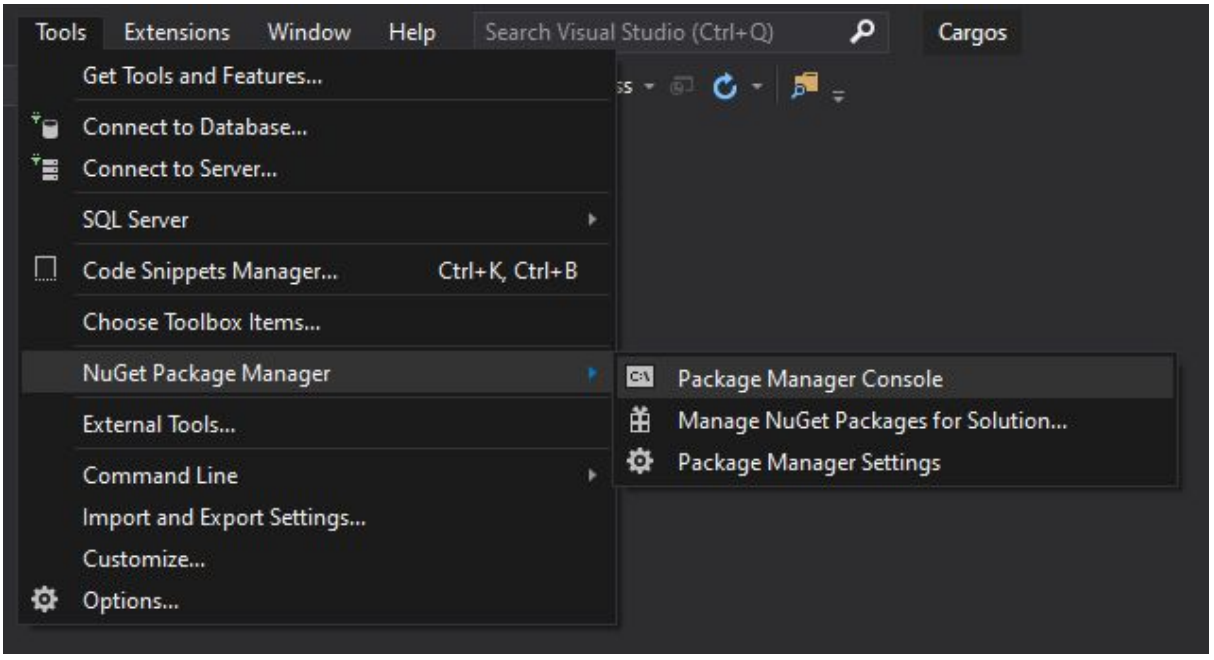
- Framework Net Core 2.1
- Visual Studio Community 2019
- SQL Server Management Studio (versión con localdb)
- Postman

Instrucciones

1) Entrar a la carpeta Cargos, abrir el archivo Cargos.sln con visual studio, esto abrirá la solución.

2) Abrir consola para ejecutar migraciones en la base.

Tools -> NuGet Package Manager -> Package Manager Console



3) Ejecutar el comando:

Update-Database

4) Presionar F5, con esto ya tenemos ejecutando la api cargos.

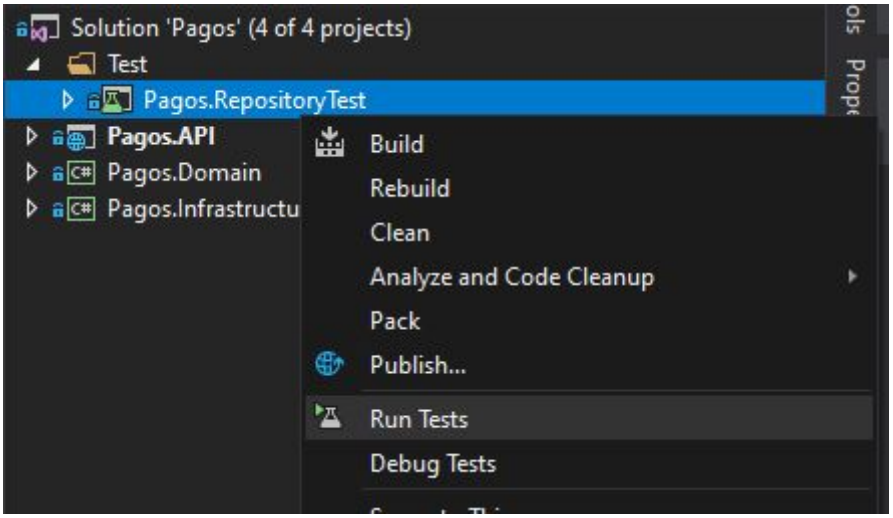
5) Entrar a la carpeta Pagos, abrir el archivo Pagos.sln con visual studio, esto abrirá la solución.

6) Repetir los pasos 2, 3 y 4 en la solución actual (Pagos).

Con esto ya tenemos ejecutando localmente las APIs.

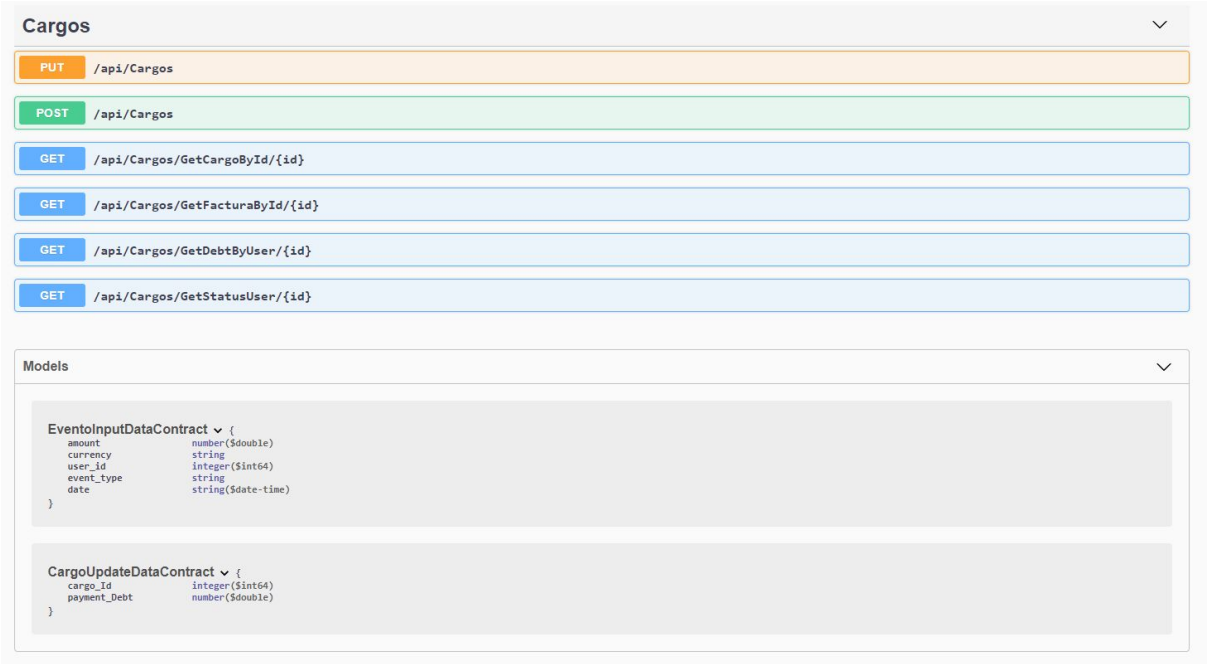
Ejecutar test

Hacer click derecho sobre el proyecto de test y Run Tests



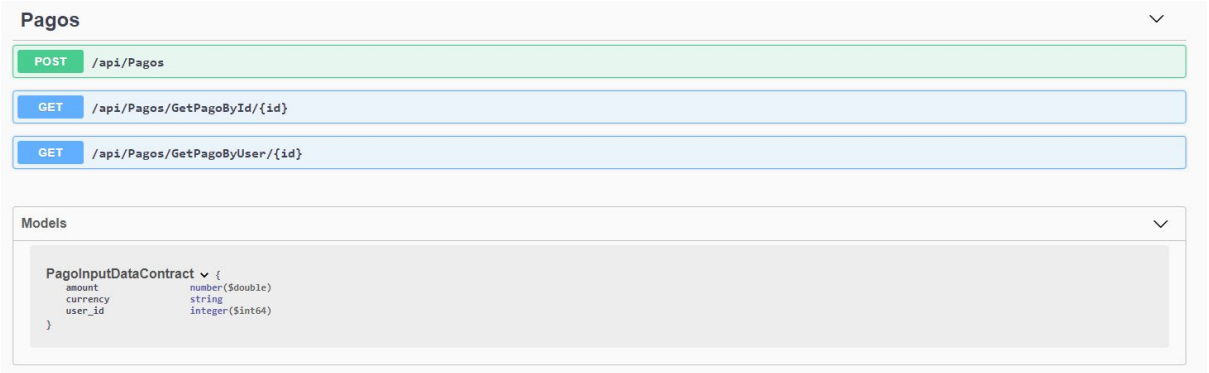
Modo de uso

Interfaz API Cargos:



Se puede acceder a la interfaz web para probar la api desde https://localhost:44311/swagger/index.html

Interfaz API Pagos:

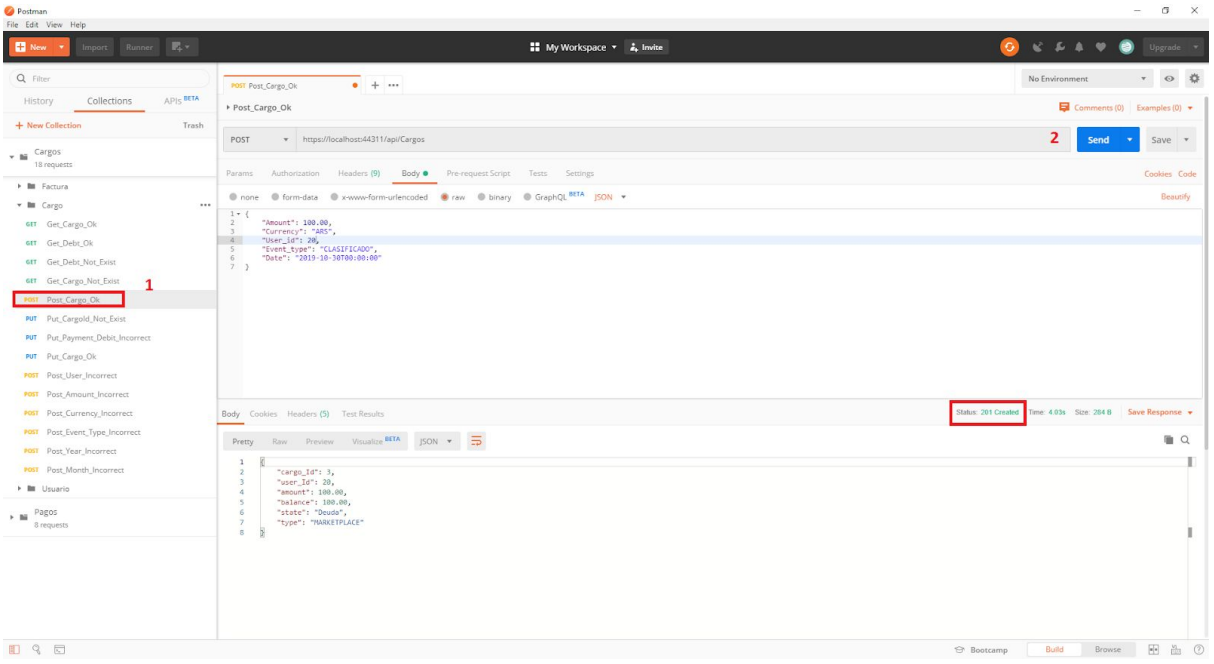


Se puede acceder a la interfaz web para probar la api desde https://localhost:44342/swagger/index.html

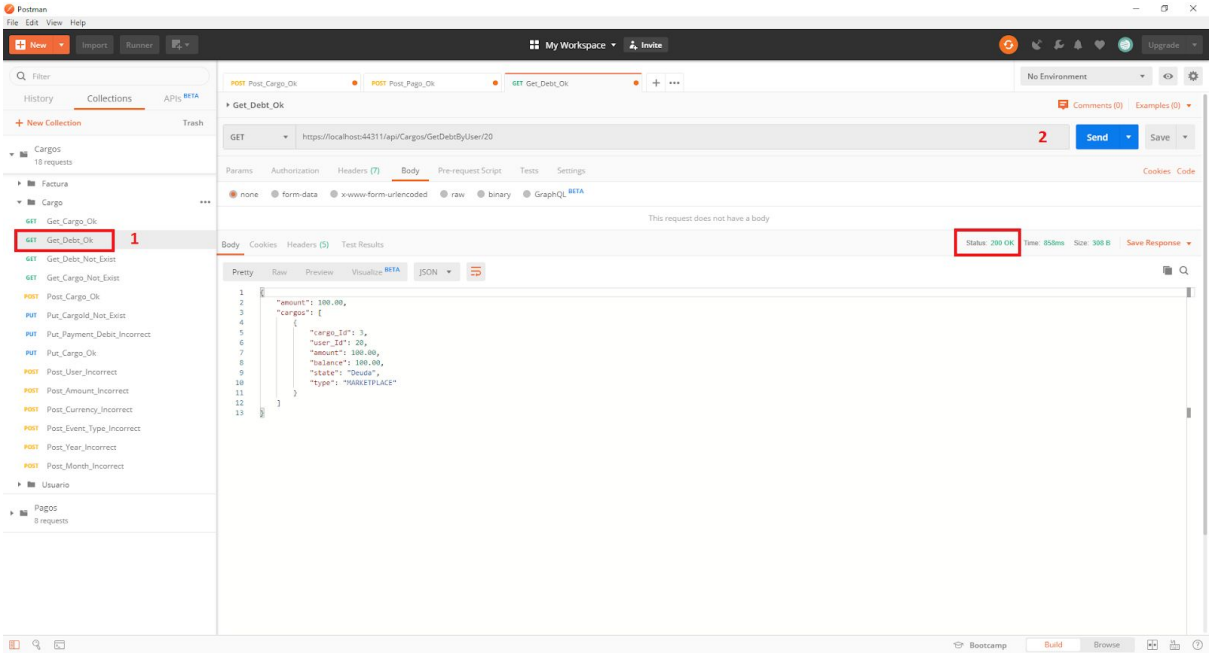
Dentro de la carpeta Postman en el repositorio hay otras consultas en los archivos: Cargos.postman_collection y Pagos.postman_collection se deben abrir usando postman.

Ejemplo: Ingreso un evento, consulto la deuda del usuario y hago el pago

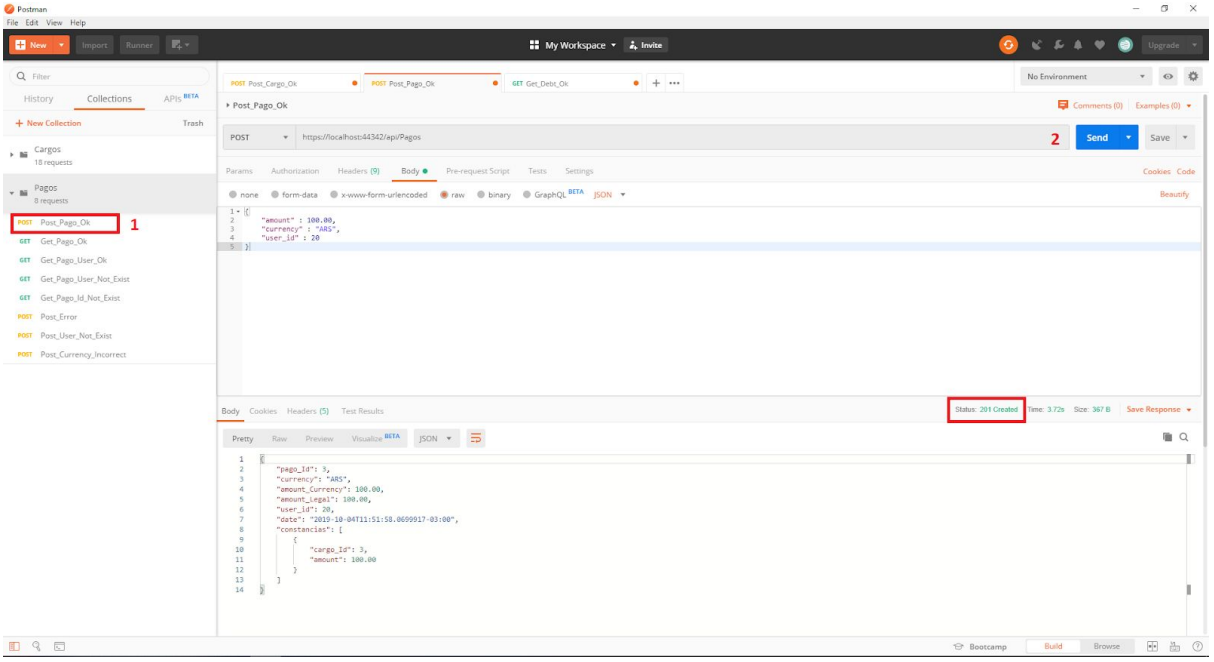
1) Ingresar evento para generar cargo
Seleccionamos la query Post_Cargo_Ok en la colección de Cargos (1), ingresamos los datos y presionamos send(2), esto nos devolverá el cargo que se creó



Podemos consultar la deuda del usuario, usando la query GetDebtByUser de la colección Cargos, ingresamos el id del usuario y presionamos send (3),esto nos devuelve la deuda del usuario con sus respectivos cargos.



2) Ingresar pago
Seleccionamos la query Post_Pago_Ok en la colección de Pagos (1), ingresamos los datos y presionamos send (2), esto nos devuelve el pago que se creó, podemos observar que en la sección constancias, nos indica el id del cargo con el que está asociado el pago.



Si volvemos a consultar la deuda del usuario, vemos que se canceló

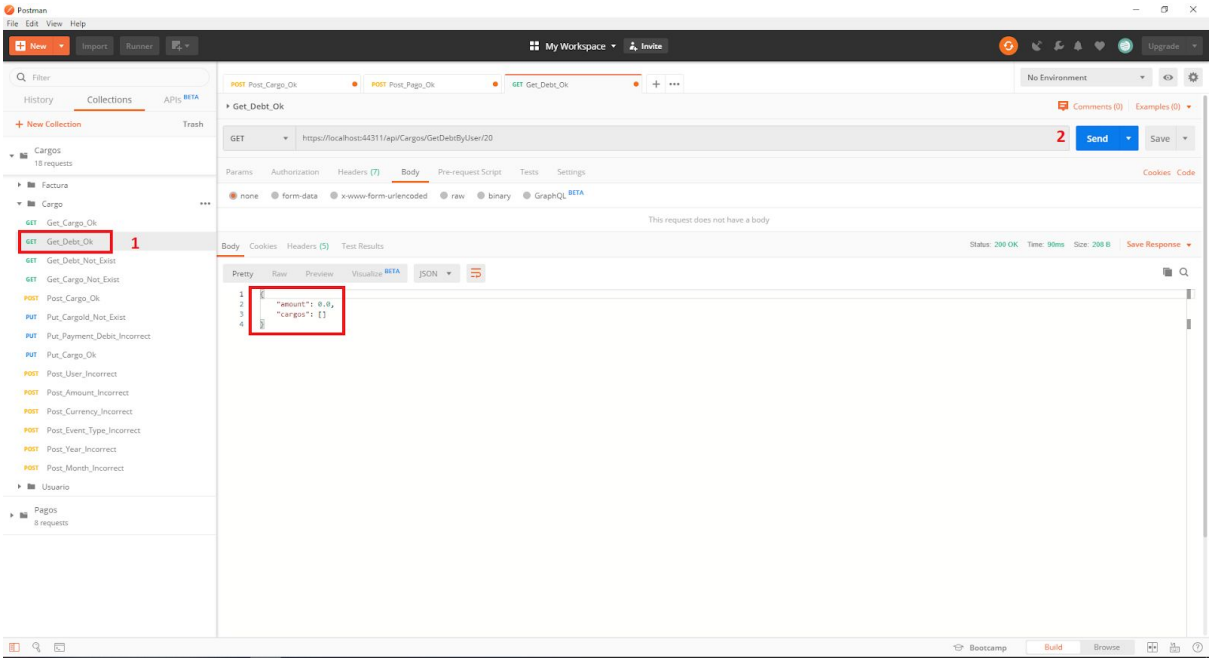
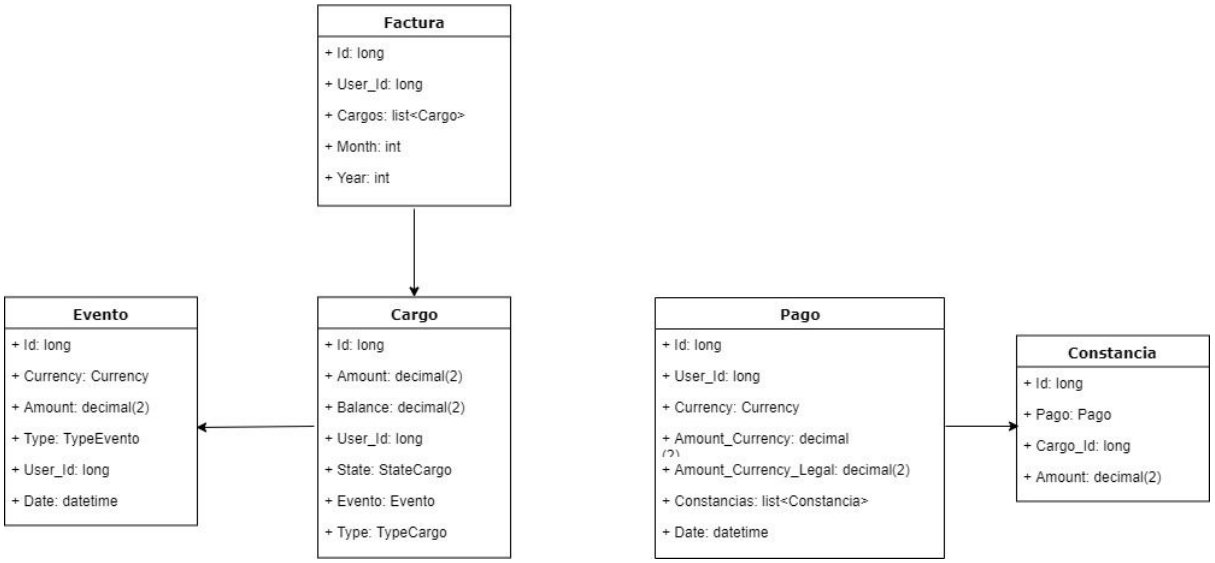


Diagrama de clases



Reglas de negocio

- Evento

amount debe ser mayor a 0 y tener 2 decimales.
Los valores válidos para Currency son: USD y ARS
Los valores válidos para Event_Type son: CLASIFICADO, VENTA, PUBLICIDAD, ENVIO, CREDITO, MERCADOPAGO, MERCADOSHOP o FIDELIDAD
La fecha debe corresponder al periodo actual (mes / año)

- Cargo

El monto se expresa siempre en ARS
Se crea siempre que ocurre un evento
Están asociados a un usuario
Están incluidos en facturas para el periodo actual (mes / año)
Cuando el saldo es mayor a 0 es considerado deuda

- Pago

Está asociado a un cargo producto de un evento
El monto debe ser igual o inferior al monto de la deuda.
Está asociado a un usuario

Tanto los eventos como los pagos puede llegar en dólares pero dentro del sistema solo se opera en pesos.

Consideraciones

En un entorno productivo se contará con un servicio externo para obtener la cotización del dólar al momento de realizar la conversión a ARS.
Para el ejercicio, lo simplifique con una constante.

```
// (1) Consideracion: se contara con un servicio que se encargara de devolver el factor de conversion a ARS.
private const decimal CONVERSION_FACTOR = 60;
```

```
private decimal GetLegalAmount(string currency, decimal amount_currency)
{
    if ((Currency)Enum.Parse(typeof(Currency), currency) != Currency.ARS)
        return amount_currency * CONVERSION_FACTOR;

    return amount_currency;
}
```

Mejoras

Usar assembler para conversiones entre entidades y datacontracts.
Test en validaciones de dominio.
Análisis de la performance la API.