JOY**LET - The Biggest OutLET E-commerce.  
  
 API Documentation**

**Summary Pages:**

1. **Part 1 - Web API – Platform, Version and Frameworks**
2. **API Documentation – Swagger**
3. **Bullet Points: Patterns, Design and BP**
4. **Part 2 – Unit Tests**
5. **Run and Test**

This documentation is separated in two parts, API and Tests as follow:  
  
**Part 1 – Web API (Server-Side)**

1. Platform version and Frameworks:  
     
   **Platform:**  
    - This project was writing in .NET 5.0 with C# Version 9.0
2. **Frameworks:** - Swagger for Documentation  
    **-** EF Core with Code First Approach  
    - MySql database in cloud based (BRAZIL)  
    - LINQ  
    - Automapper

* **API Methods - Swagger Documentation**:

Swagger API for Test and Open API support Documentation for describing REST API Methods.

To avoid DRY coding, I’ve created one method for calculate ONE or ALL carts where ID is optional, so this is the only point for pay attention. (Method: CartsController > **GetCartsResultAsync**).

This method **GetCartsResultAsync** has been updated for each required LEVEL i.e. the final results considered for LEVEL is: **LEVEL 1 – No Rules  
LEVEL 2 – Delivery Fee  
LEVEL 3 – Discount applied by Article**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Bullet Points**

* **Code first ‘approach’:**

You will see class and methods following each step of the test. Example: LEVEL 1 you gonna see only 3 classes (Articles, Carts and CartsItems). LEVEL 2 you will see one more class called DeliveryFee. This process is repeated in LEVEL 3.  
I’ve used this pattern once that I built it using MySql in cloud instead static Json file. I chose this option for produce more reality for the application. By the way there are some additional attributes and some entities.

For history data you can check the uploads done at: DbContext Class (Joilet.API > Data > AppDbContext) and in the Migration Folder too.

**Important:** You are able to create more **Carts** and calculate them as well.

* **Unit of Work**:

Unit of Work pattern to avoid data lost in data entries. In this case EF Core handler this process doing the rollback automatically.

pattern to avoid data lost in data entity entries

* **DTOs**:

DTO pattern to transfer data between layers and avoid expose unnecessary data.  
For that I used Automapper framework.

* **DDD and Clean Code**:

The project was writing following the best practices and patterns as you can see like Repositories, Services, DI, Clean Code etc.

**Part 2 – Unit Test – XUnit Framework with Moq**

**Important:** Almost test using Seed Data method to load In Memory Data, so is not possible to RUN ALL tests in View **i.e. One by One**.

All tests were passed out.

1. **Platform:**  
    - This project was writing in .NET 5.0
2. **Frameworks:** - XUnit  
    **-** Moq

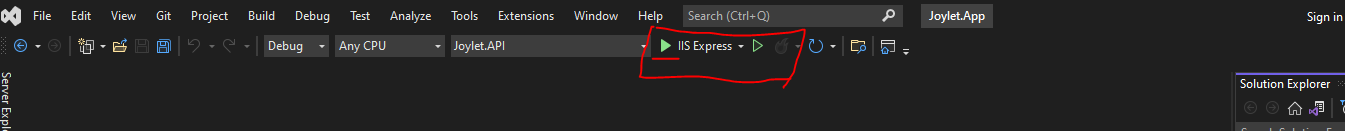
Test Project follow the same required steps for LEVEL 1, LEVEL 2 – Delivery Fee Applied and LEVEL 3 – Discounted by Article Applied.

**Run and Testing Project**

For Run and Testing check and follow the steps bellow:

1 – Open a project in Visual Studio 2022 (preference)

After opened click on ‘IIS Express’



2 – After APIs has been loaded select one method for check result:

Here I’ll explain about the main methods.

Here an example for the method “**getcartsresult”** which return final balance for each cart or for a specific cart if needed (By Id).

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

3 – It’s possible send 1 or more Carts. But is important let ID field as **0** for each Cart send. I could be removed this field in DTO CreateCart, however I decided to let it most approximated from the schema provided.

In this example bellow you see a POST where is being sent 2 carts. The first one has only 1 product while the second one has 2 products in the Cart.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente com confiança média

4 – Evidence for API results for each LEVEL

1. LEVEL 1 – Project Name: “Joylet Level 1”  
   Remember that I’m using a single database for this test for this reason even in level one you will see MORE Carts concerning from LEVEL 3. Once that in Level 1 and 2 there was NO data for Carts ID 3+.

Padrão do plano de fundo

Descrição gerada automaticamente

1. LEVEL 2 - Project Name: “Joylet Level 2”   
   With Delivery Fees, but no discount applied.  
     
   Interface gráfica do usuário, Texto

   Descrição gerada automaticamente
2. LEVEL 3 - Project Name: “Joylet Level 3”

Final version with Delivery Fees and Discount applied.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média