

Descrição do projeto

Projeto com a proposta de apresentar uma abordagem mais ampla de conhecimentos para montagem de uma solução completa em aspNet core.

A solução desenvolvida segue o conceito conhecido como "Clean Architecture", utilizado em sistemas com arquiteturas modernas, permitindo maior facilidade para manutenção e expansão, e melhor entendimento do código, com divisão clara de responsabilidades para cada uma das camadas que compõem a solução.

A solução está dividida nas seguintes camadas/projetos:

- **Projeto Domain e Projeto Application** - Contendo as camadas com as regras de negócio. Na atual configuração da solução, não houve necessidade a priori, de implementar a camada Application.
- **Projeto Infrastructure** - Contem a camada de persistência
- **Projeto CrossCutting** - Contém a camada com classes referenciadas pelos demais projetos
- **Projeto User Interface** - Contem a camada de apresentação/Api
- **Projeto Consumer** – Contem a camada que vai abrigar a classe consumer da mensageria

O projeto implementa os seguintes patterns:

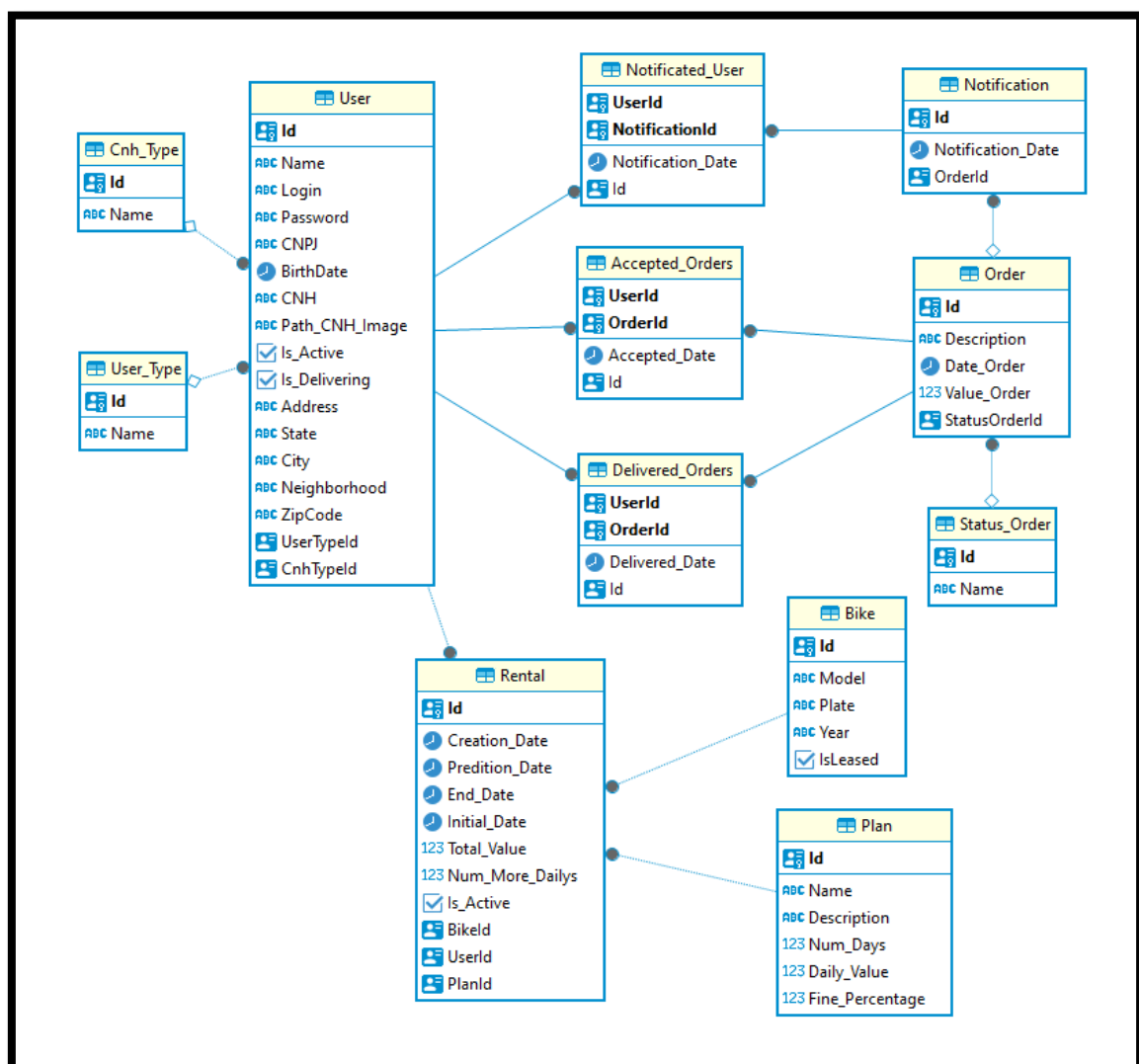
- Clean Architecture
- n-Tier
- Unit Of Work
- Repository
- Class Mapping
- ORM

O projeto utiliza mensageria por meio do **RabbitMQ** em conjunto com a biblioteca de abstração **MassTransit**.

Para o banco de dados, foi utilizado o **Postgres versão 16.2**, implementando a tecnologia ORM por meio do **Entity Framework Core**, para abstração do banco de dados.

Além disso, por meio do EF Core, foi adotado a **abordagem Code First** para modelagem de banco de dados.

A seguir, a modelagem proposta do banco de dados, de acordo com possíveis funcionalidades imaginadas para esse sistema fictício de gestão de locações de moto.



Para dar suporte a execução da solução, é preciso ter instalado o **Docker** localmente. Pelo Docker, serão criados **2 containers** que darão suporte ao sistema. **Um container do banco de dados e outro do Broker de mensageria RabbitMQ.**

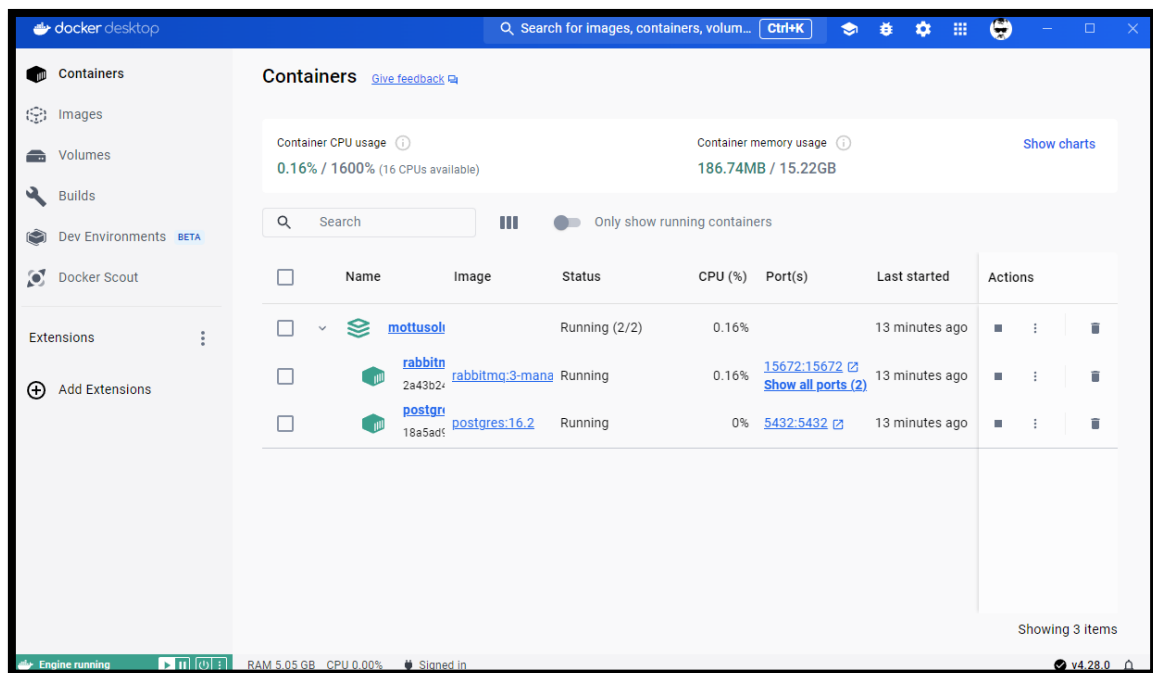
Com a presente solução, se propõe mostrar uma meríade de tecnologias sendo utilizadas num projeto aspnet core, de forma a contemplar boas práticas e a adoção de convenções e padrões de mercado, aplicados ao desenvolvimento de sistemas.

Imagens e containers do Docker

Na raiz do sistema, no nível da solution, existe um arquivo docker-compose.yml. Abrir um powershell nessa pasta e executar o comando abaixo:

Docker-compose up -d

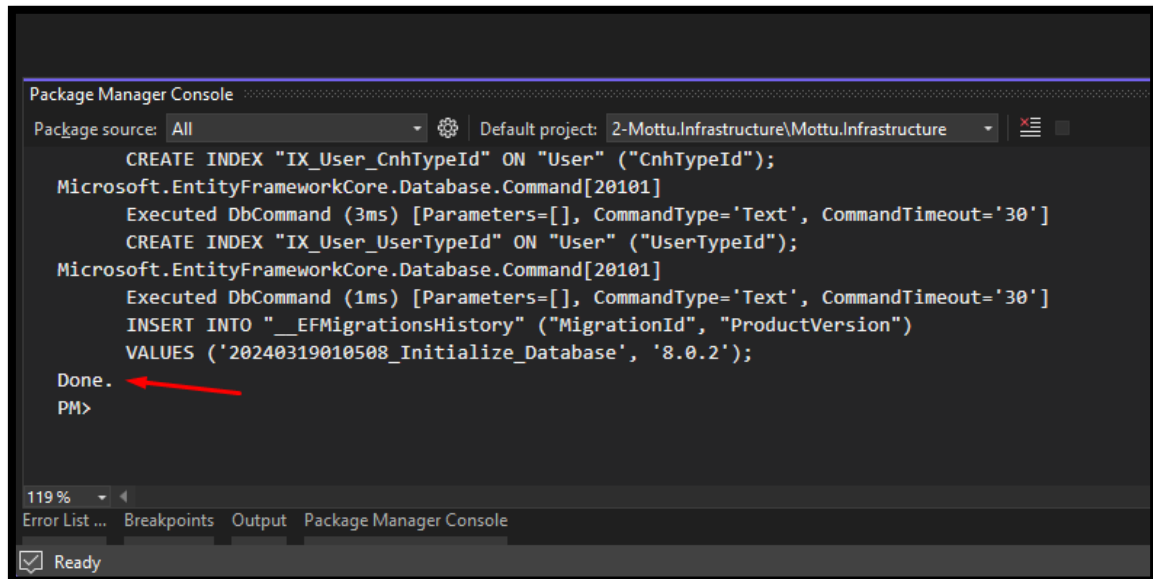
Uma vez executado com sucesso, os containers a seguir serão apresentados, já em execução.



A seguir, deve-se abrir a solution do projeto, abrir uma **Package Manager Console**, selecionar o projeto **Mottu.Infrastructure** na dropdown Default Project, e rodar o comando abaixo que a migration e inicialização do banco seja executada:

Update-database

A mensagem **Done** indicará que a criação do banco de dados ocorreu com sucesso.



```
Package Manager Console
Package source: All Default project: 2-Mottu.Infrastructure\Mottu.Infrastructure

CREATE INDEX "IX_User_CnhTypeId" ON "User" ("CnhTypeId");
Microsoft.EntityFrameworkCore.Database.Command[20101]
Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
CREATE INDEX "IX_User_UserId" ON "User" ("UserId");
Microsoft.EntityFrameworkCore.Database.Command[20101]
Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
INSERT INTO "__EFMigrationsHistory" ("MigrationId", "ProductVersion")
VALUES ('20240319010508_Initialize_Database', '8.0.2');

Done.
PM>
```

Utilize um gerenciador de banco de dados para abrir o modelo e visualizar o banco criado. A sugestão é usar o **Dbeaver**.



Para conectar o banco, informe os dados conforme apresentados no print abaixo. O usuário é **postgres**, a senha **12345** e o nome do banco **MottuDB**.

Conectar a um banco de dados

Configurações de conexão

Configuração de conexão para PostgreSQL

Principal PostgreSQL Driver properties SSH SSL + Network configurations...

Servidor

Conecte usando: ☒ Host ☐ URL

URL:

Host: Porta:

Banco de dados: ☐ Exibir todos os bancos de dados

Autenticação

Autenticação:

Nome de usuário:

Senha: ☒ Salvar senha

Advanced

Role da sessão: Cliente local:

[Você pode usar variáveis nos parâmetros de conexão.](#) [Detalhes da conexão \(nome, tipo, ...\)](#)

Nome do driver: PostgreSQL [Configurações de driver](#) [Driver license](#)

[Testar conexão ...](#) [< Voltar](#) [Avançar >](#) [Concluir](#) [Cancelar](#)

MottuDB - localhost:5432

- Bancos de dados
 - MottuDB
 - Schemas
 - public
 - Tabelas

> Accepted_Orders	16K
> Bike	24K
> Cnh_Type	24K
> Delivered_Orders	16K
> Notificated_User	16K
> Notification	16K
> Order	24K
> Plan	32K
> Rental	32K
> Status_Order	24K
> User	32K
> User_Type	24K
> __EFMigrationsHistory	24K
 - Foreign Tables

Descrição das Entidades do Banco

- Accepted_Orders (Pedidos Aceitos)
- Bike (Moto)
- Cnh_Type (Tipo de Cnh)
- Delivered_Orders (Pedidos Entregues)
- Notificated_User (Usuários Notificados)
- Notification (Notificações)
- Order (Pedidos)
- Plan (Planos)
- Rental (Locações)
- Status_Order (Status dos Pedidos)
- User (Usuários)
- User_Type (Tipo de Usuário)

Como testar

Para testar a aplicação, deve por dois projetos pra iniciarem simultaneamente. Um é o projeto **Mottu.Api** e o outro é o projeto **OrderConsumer**, de forma a permitir ver o processo de mensageria sendo executado.

Diferenciais

O projeto contempla como diferenciais:

- Uso de ORM por meio do EF Core
- Uso de containers Docker e Docker Compose
- Adoção de design patterns
- Documentação (esta que está sendo lida) além do swagger e de comentários ao longo do código
- Tratamentos de erros em diversos pontos do sistema, buscando a maior estabilidade ao projeto
- Solução dividida em layers adotando arquitetura limpa, ou Clean Architecture, como é conhecida
- Modelagem de dados por meio de classes e geração da estrutura de banco pela abordagem Code-First com uso de migrations

- Logs em alguns pontos na classe Consumer que atende ao Broker de mensageria RabbitMQ
- Adoção de convenções de codificação, adotadas pelo mercado de desenvolvimento
- Todo o código desenvolvido em inglês, mas comentários e mensagens do sistema ainda permanecendo em português

Download da solução

O projeto se encontra versionado no endereço [olivertech/MottuSolution \(github.com\)](https://github.com/olivertech/MottuSolution)