BootCamp: Arquiteto de Software

DESF5 - Atividade do Desafio Final

Aluno: Guilherme de Souza Reis

Desenvolvimento da solução do Desafio Final

1. **Definição e escolha da Stack e contexto.**
   De acordo com o prazo de entrega do projeto, optei por permanecer em minha stack, dessa forma decidi realizar a criação da API restFUL MVC utilizando C# .NET por ter familiaridade com a linguagem. O domínio (entidade) escolhida foi de acordo com a ideia de que o projeto do desafio poderia vir a se tornar algo utilizável e com aplicação real, por este motivo escolhi o contexto de "Eventos" pois poderia ser utilizada juntamente com uma de minhas aplicações em ambiente de produção, o DattaNew.

2. **Criação da documentação e elaboração das estratégias de desenvolvimento.**
   Realizei a diagramação e estruturação dos componentes que seriam utilizados no projeto para ter maior clareza das necessidades e relações do código, utilizando, como solicitado a diagramação em C4 Model e também o padrão service repository juntamente com a estruturação das pastas.

3. **Adequação do ambiente de desenvolvimento para WebAPI.**

Utilizei o VS Code para realizar a implementação do código, juntamente com as bibliotecas pertinentes para possibilitar uma melhor eficiência e deploy do código. São elas: C# Dev Kit, .NET Install Toll, .NET Extension Pack, EntityFramework [Core, CoreDesign, SQLServer, Tools], Swagger e Prettier. Integrando ao ambiente .NET 9.0.0.

4. Let's Code

Iniciei a implementação criando validando a execução e adequação das bibliotecas, frameworks e dependências rodando a api para certificar de que estava tudo ok. O primeiro passo foi realizar a criação da Model e o dbContext, juntamente com as configurações do EntityFramework para conexão com o banco de dados.

5. Banco de dados

Utilizie um container Docker para emular o SQLServer e possibilitar a conexão e persistência dos dados, desta forma não precisando instalar em minha máquina uma instancia ~~pesada~~ do SQL Server, utilizando apenas o SMSS como SGBD para consultas e conferências.

6. Controller

O próximo passo foi a criação da controller para recepção dos métodos HTTP. Com isto também surgiu a necessidade da criação da classe DTO (Data Transfer Object) para transportar dados entre processos e camadas da aplicação, especialmente entre cliente e servidor.

Realizei a criação dos métodos GET, POST, UPDATE e DELETE de forma assíncrona.

7. Service Repository

Após a criação dos métodos criei a interface para possibilitar o a Injeção de dependência e os relacionamentos entre Service/Repository e a controller, realizando novas validações para constatar o funcionamento do padrão.

8. Métodos do desafio

Utilizei a classe Service para aplicar o máximo possível de utilização dos métodos existentes para implementar as funcionalidades extras solicitadas no desafio como regras de negócio, por exemplo: Contagem, se eu já tenho um método que retorna todos os elementos, preciso apenas contar a quantidade de resultados... Criei os endpoints também na controller e realizei as ultimas validações e testes básicos para constatar o funcionamento.

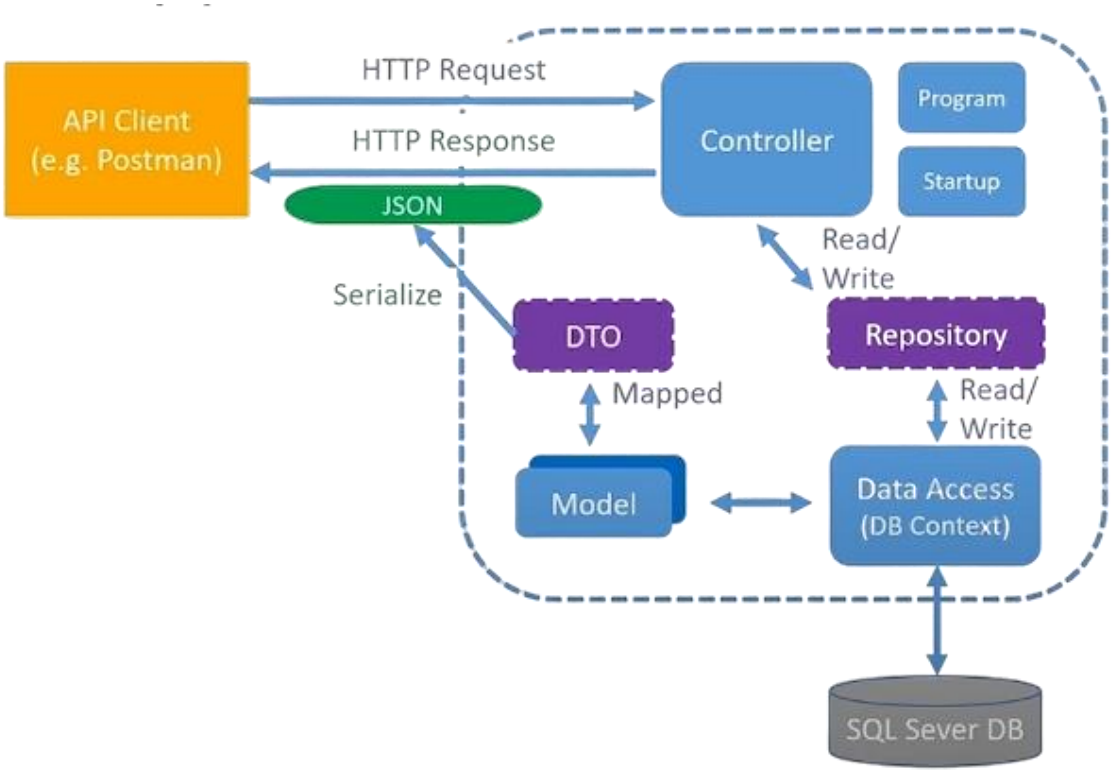
9. Repositório no GitHub

Criei um repositório em minha conta no GitHub e enviei o projeto para possibilidade de clonagem e realização dos testes. Também realizei a criação de um readme com os detalhamentos e orientações para execução.

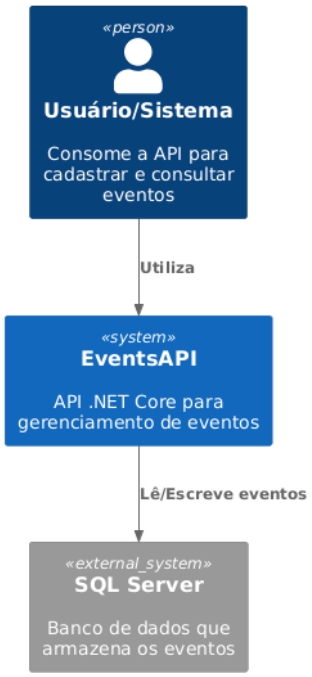
Desenho arquitetural do software

e

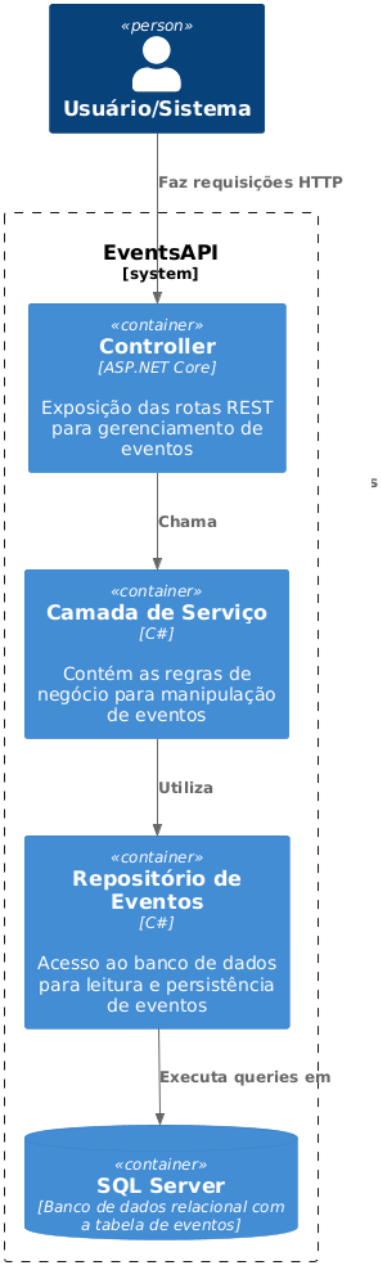
Diagrama C4 Model



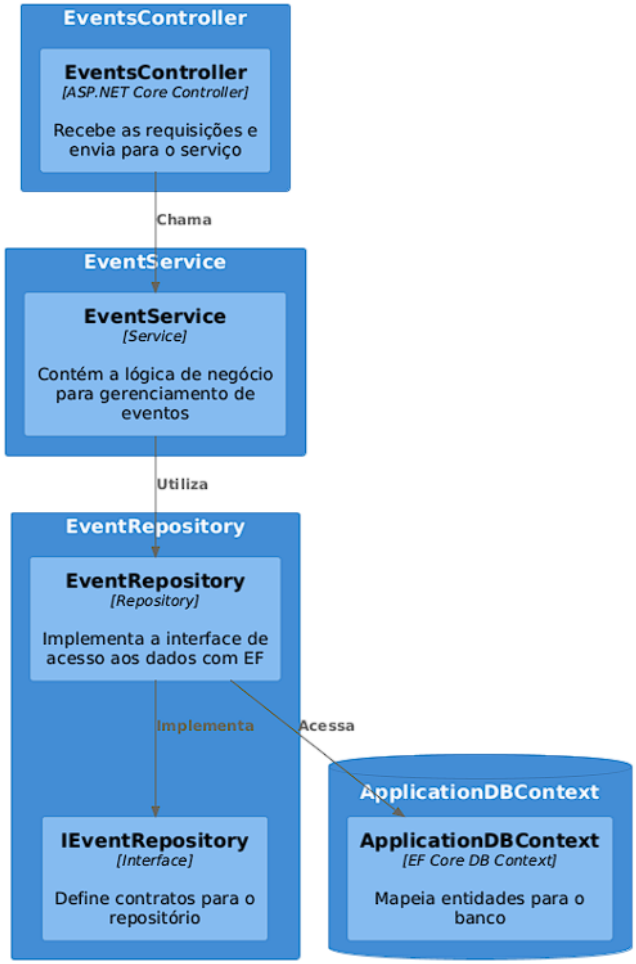
C1 - Diagrama de Contexto



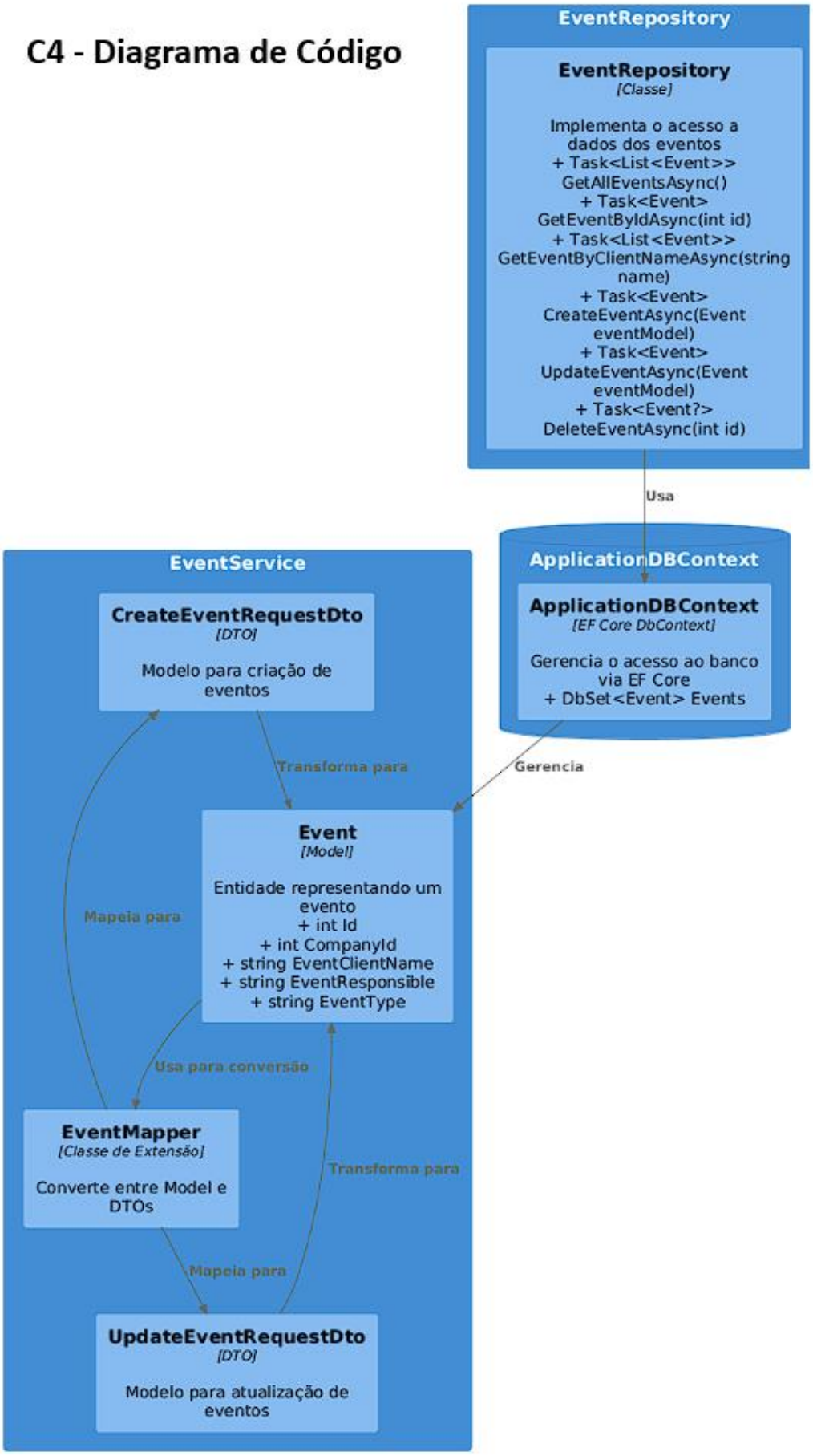
C2- Diagrama de Container



C3 - Diagrama de Componente



C4 - Diagrama de Código



Estrutura das pastas

EventsAPI/

├— Controllers/ - Contém os controladores que manipulam as requisições HTTP e definem os endpoints da API. Ex. Requisições GET/POST ...

├— Models/ - Define a estrutura das entidades utilizadas na aplicação e também mapeia para as tabelas do banco de dados utilizando o EntityFramework. Ex: “Eventos” e seus atributos: ID da empresa, responsável pelo evento, cliente...

├— Repository/ - Contém classes que manipulam as operações no banco de dados para, por exemplo, executar as operações CRUD.

├— Service/ - Contém classes de serviços que implementam a lógica de negócios, atuando como um intermediador entre controller e repositor para aplicar regras específicas.

├— Interface/ - Contém as definições de interface para Service e Repository, definindo “contratos” para injeção de dependência e baixo acoplamento entre componentes.

├— Migrations/ - Contém arquivos gerados pelo EntityFramework para realizar e rastrear as alterações e atualizações no esquema do banco de dados.

├— Properties/ - Armazena configurações específicas do projeto para depuração e execução da aplicação.

├— wwwroot/ - Contém arquivos estáticos como JavaScript, CSS e imagens

├— appsettings.json - Armazenar strings de conexão do banco de dados, configurações de log e outras configurações da aplicação.

├— Program.cs - Contém arquivos e configurações para inicialização e execução da aplicação.

Link do repositório:

https://github.com/guilhermesouzar/EventsAPI-DesafioFinal-BootCamp_Arquiteto_de_software

Conclusão

O desenvolvimento do projeto foi de grande valia pois ainda não havia tido experiencias diretas com a criação de API utilizando o padrão estruturam MVC. Eu gostei muito e tive certa facilidade pois o MVC é minha stack de trabalho, mas foi desafiador pois tive que lidar com versões mais novas do .NET.

Os conhecimentos do bootcamp foram enriquecedores para entender e fixar conceitos de desenvolvimento e boas práticas de código, como design patterns e padrões arquiteturais.

As principais dificuldades foram fazer com que o projeto seja totalmente replicável em outras máquinas para teste, pois quando estamos fazendo projetos pessoais acabamos não dando tanto importância para este fator, mas neste caso houveram algumas complicações em tentar "resetar" tudo e simular a inicialização do banco de dados com Docker e da aplicação em si. Espero ter conseguido deixar o mais claro possível no arquivo README do github sobre os passos para execução do projeto.

Os resultados obtidos foram novos conhecimentos e contato com novas tecnologias e novos conceitos que são muito cobrados e utilizados no mundo profissional, e isso com certeza é muito valioso.

Melhorias futuras consigo enxergar este projeto sendo integrado com minha outra aplicação, possibilitando integrações entre outros sistemas e ainda a possibilidade de rentabilizar com isto. E ainda é possível utilizar os conhecimentos obtidos como base para novos e maiores projetos que irão surgir.