**Especificações Iniciais**

**Sobre a versão do SDK .NET Core.**

Para realizar este projeto foi utilizado o SDK 31.405 do .NET Core, optei por utilizar uma versão LTS.

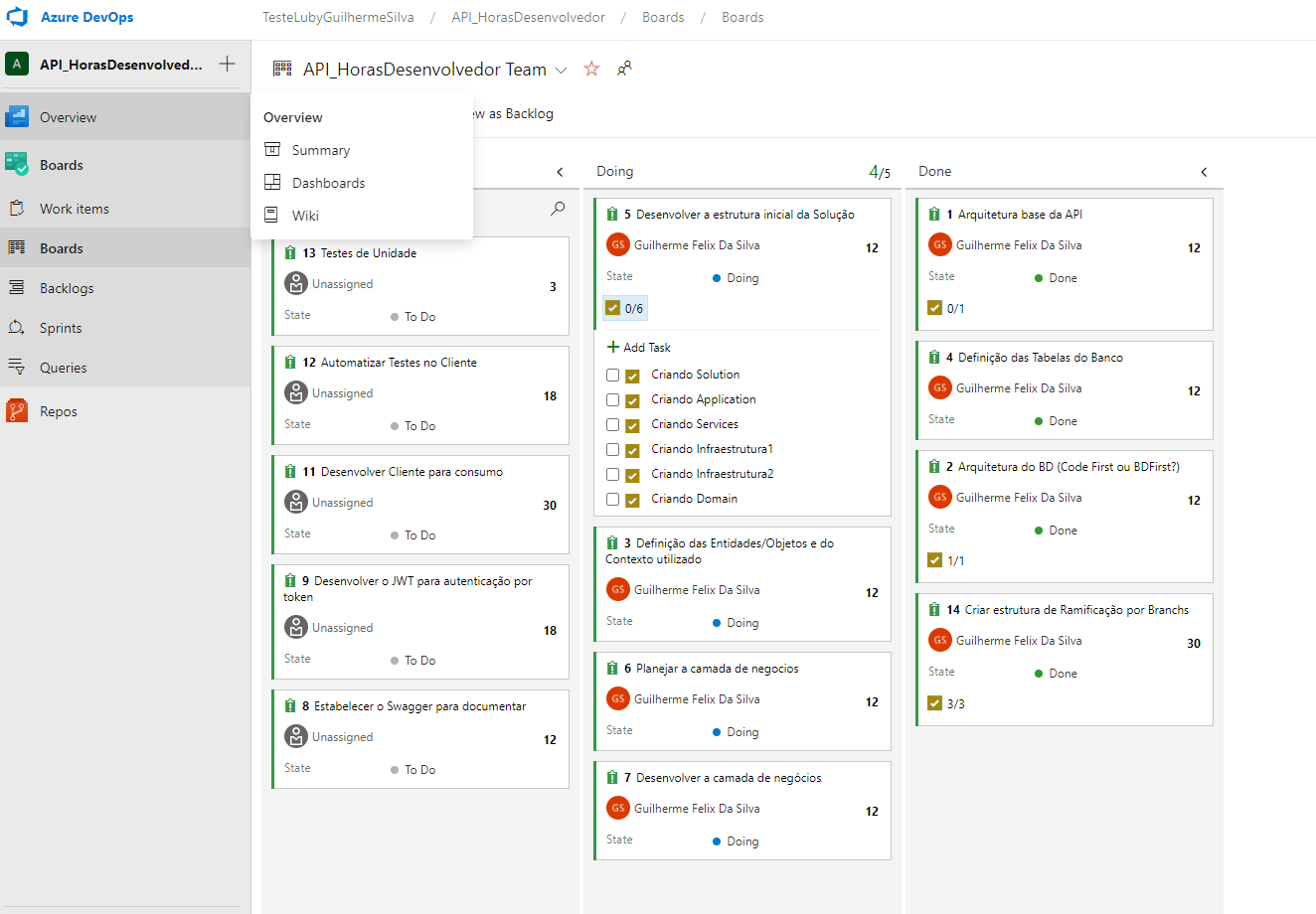
Download efetuado em <https://dotnet.microsoft.com/download/dotnet-core/thank-you/sdk-3.1.405-windows-x64-installer>

**Sobre a IDE de desenvolvimento.**

Foi utilizado o Visual Code, Download efetuado em <https://code.visualstudio.com/download>

**Sobre o controle de versão, desenvolvimento ágil e ramificações.**

Foi utilizado o Azure DevOps para o trabalho e ao término do projeto foi realizada a migração do repositório git para o GitHub.



Quadro Kanban em: <https://dev.azure.com/TesteLubyGuilhermeSilva/API_HorasDesenvolvedor/_boards/board/t/API_HorasDesenvolvedor%20Team/Issues>

Branches do Azure DevOps em:

<https://dev.azure.com/TesteLubyGuilhermeSilva/_git/API_HorasDesenvolvedor/branches>

Branchs do GitHub em:

<https://github.com/GuilhermeFelix/TestApi_HorasDevs>

**Sobre o Banco de Dados**

Foi utilizado o Banco de Dados MySql 5.6 com hospedagem em Nuvem.

Host de Acesso em: testeluby.mysql.uhserver.com

Nome do Banco: testeluby

Usuário: testeluby

Senha: Mudar@12

Optei por utilizar um banco de dados em nuvem para simular o uso de um servidor externo pois é uma situação muito frequente em aplicações no dia a dia, para isto criei uma database e configurei o usuário de acesso em um host privado.

Optei por utilizar Code First + Migrations para adequar a base em qualquer implantação, isto permite que o código seja portável para outros ambientes e que as especificações da base serão mantidas.

Habilitar Migrações do Code First. O recurso de migrações permite que você altere o modelo de dados e implante suas alterações na produção atualizando o esquema de banco sem precisar remover e recriar o banco de dados. (Fonte: Microsoft)

Em ambiente de desenvolvimento, principalmente com um software sendo desenvolvido de forma incremental, a base de dados muda muitas vezes antes de chegar a seu estado final. Nesses momentos a utilização de Migrations é fundamental. **O Entity Framework Migrations fornece uma forma prática para lidar com essas mudanças.** (Fonte Devmedia)

**Sobre as tabelas do Banco:**

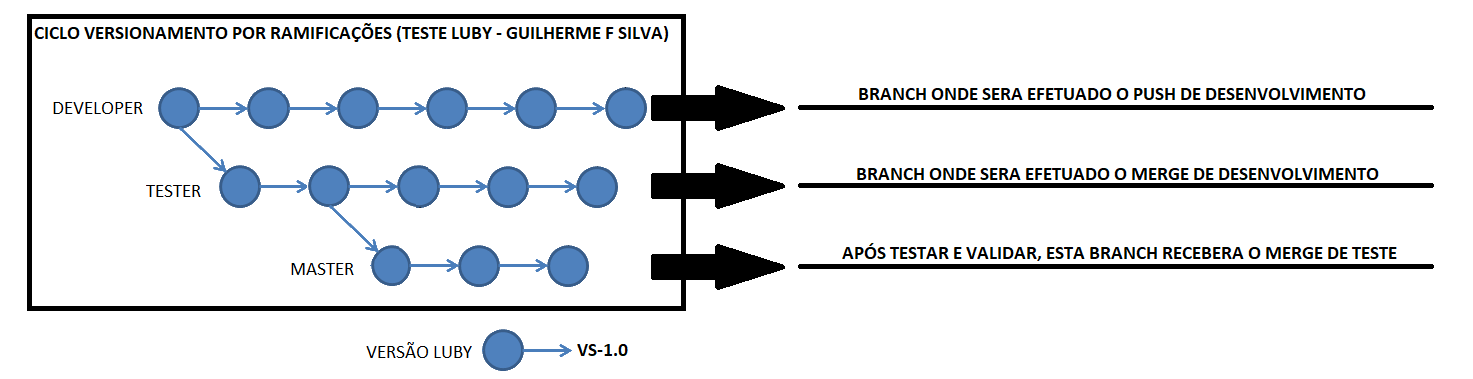
|  |  |  |  |
| --- | --- | --- | --- |
| TblDesenvolvedor | | | |
| Campo | Descrição | Tipo | Observação |
| Id | Numerador do registro da tabela | int | automatico |
| Nome | Nome do Desenvolvedor | varchar(40) | Not Null |
| Sobrenome | Sobrenome do Desenvolvedor | varchar(80) | Not Null |
| CPF | CPF do Desenvolvedor | varchar(11) | Not Null |
| Email | Email do Desenvolvedor | varchar(100) | Not Null |

|  |  |  |  |
| --- | --- | --- | --- |
| TblProjeto | | | |
| Campo | Descrição | Tipo | Observação |
| Id | Numerador do registro da tabela | int | automatico |
| Nome | Nome do Projeto | varchar(30) | Not Null |

|  |  |  |  |
| --- | --- | --- | --- |
| TblDesenvolvedorXProjeto | | | |
| Campo | Descrição | Tipo | Observação |
| Id | Numerador do registro da tabela | int | automatico |
| FK\_Cpf | CPF do Desenvolvedor | varchar(11) | Not Null |
| FK\_Projeto | Id do Projeto | int | Not Null |

|  |  |  |  |
| --- | --- | --- | --- |
| TblDesenvolvedorXLancamentoHoras | | | |
| Campo | Descrição | Tipo | Observação |
| Id | Numerador do registro da tabela | int | automatico |
| FK\_Cpf | CPF do Desenvolvedor | varchar(11) | Not Null |
| FK\_Projeto | Id do Projeto | int | Not Null |
| DataInicio | Data/Hora Inicio do Trabalho | Datetime(2) | Not Null |
| DataFim | Data/Hora Fim do Trabalho | Datetime(2) | Not Null |
| DataLancamento | Data/Hora Lancamento da Requisição | Datetime(2) | Not Null |

**Sobre a estrutura de Branchs**



Estrutura de Ramificação Simples, considerações para o sucesso desta avaliação:

1 - Branch Master( Que sera a branch responsavel pelo ramo mais confiavel antes da homologação para o Github).

2 - Branch Developer( Que sera a branch utilizada para desenvolver)

3 - Branch Tester(Que sera a branch utilizada para Testar).

4 - Cada Task deverá ter sua branch local criada no momento que inicia o trabalho, porem após finalizar, deverá ser efetuado um Merge na branch de Developer.

5 – Deverei atualizar sempre a branch Developer com o conteúdo da Master para isso efetuarei um Merge da Master.

6 – Após realizar a entrega da task, o ideal seria comitar as Features de desenvolvimento e realizar merge na branch de Tester.

7 – Após testar as Features de desenvolvimento, deve ser realizado merge na Master mediante PullRequest.

8 – Ao término da sprint deve ser gerada a versão como não tenho que gerar versão para este caso deverá ser transferido o Git do Azure para o GitHub.

9 – O projeto deverá ter Commits claros e específicos para a task atual no Scrum.

10 – Como estamos falando de um projeto de curto prazo, sera necessário conseguir tempo em uma agenda apertada por conta do atual emprego.

**Sobre a Arquitetura Base da API**

Foi utilizado o padrão arquitetural Domain-Driven Design (DDD) onde o design é direcionado ao domínio de negócios. Optei por utilizar esta arquitetura pois necessito modelar o problema entendendo claramente suas especificações.

Partindo do pressuposto que o DDD é focado no domínio de negócios subentende-se que o foco será resolver o problema proposto para a criação da API que neste caso é lançar horas trabalhadas em um projeto.

**Camadas deste projeto:**

**Domínio**

**Aplicação**

**Serviços**

**Infraestrutura 1** – Migrations e Configurações

**Infraestrutura 2** – Data Access.

Aplicação Ativa em: