,,



Documentação

Opflix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Alameda Barão de Limeira, 539 – São Paulo/SP |  |  |  | sp.senai.br |

Sumário

1. [Resumo 3](#_Toc22908503)

[Objetivos](#_Toc22908504)

1. [Descrição do projeto 3](#_Toc22908505)

[Resumo do projeto](#_Toc22908506)

1. [Ferramentas utilizadas 3](#_Toc22908507)
2. [Modelagem de Software 3](#_Toc22908508)

[Modelo Lógico](#_Toc22908509)

[Modelo Físico](#_Toc22908510)

[Modelo Conceitual](#_Toc22908511)

1. [Criando Scripts do Banco de Dados 5](#_Toc22908512)

[Notas](#_Toc22908513)

[Criando o Script](#_Toc22908514)

[Rodando o Script](#_Toc22908515)

1. [Deploy / Rodar Projeto 6](#_Toc22908516)
2. [Swagger 7](#_Toc22908517)

[Definição](#_Toc22908518)

[Acessando](#_Toc22908519)

1. [Pacotes Nuget 9](#_Toc22908520)

[Pacotes Necessários](#_Toc22908521)

[Como Instalar](#_Toc22908522)

1. [Postman 11](#_Toc22908523)

[Exportando](#_Toc22908524)

[Importando](#_Toc22908525)

1. [Funcionalidades 14](#_Toc22908526)

[Web](#_Toc22908527)

[Mobile](#_Toc22908528)

1. [Protótipos 14](#_Toc22908529)

[Web](#_Toc22908530)

[Mobile](#_Toc22908531)

1. [Front-End 18](#_Toc22908532)

[Web](#_Toc22908533)

[Mobile](#_Toc22908534)

1. [Mobile 19](#_Toc22908535)
2. [Arquitetura do Projeto 19](#_Toc22908536)

[Banco de Dados](#_Toc22908537)

[Back-End](#_Toc22908538)

1. [Referências 20](#_Toc22908539)

[Links](#_Toc22908540)

Resumo

Objetivos

Este documento tem como objetivo planejar, organizar e formalizar o projeto da empresa/cliente Opflix, para que este seja entregue para o cliente.

Descrição do projeto

O produto final desse projeto é a possuir uma aplicação Web (Html, Css, Javascript) e Mobile (React e React Native) com integração de bancos relacionais (SQL Server) e há de ser plataformas adaptáveis a quaisquer resoluções e sistemas Android/IOS. As aplicações devem ser bem performáticas, leves, e bonitas de modo que seus usuários saiam satisfeitos ao navegar pelas interfaces. A necessidade da empresa é anunciar lançamentos de filmes, e os usuários poderão interagir com os lançamentos de modos como favorizando, assistindo ou anunciando outros lançamentos. Os usuários da plataforma serão separados por permissões “comum” ou “admin”, onde cada perfil possui um leque de interação com o app. Os lançamentos possuirão categorias, plataformas onde serão também publicados e o seu tipo de conteúdo, sendo eles filmem ou séries.

Resumo do projeto

O projeto é a construção de aplicações Web/Mobile para lançamento de filmes e séries.

Ferramentas utilizadas

**Nota:** Todas as ferramentas utilizadas são disponibilizadas gratuitamente na internet.

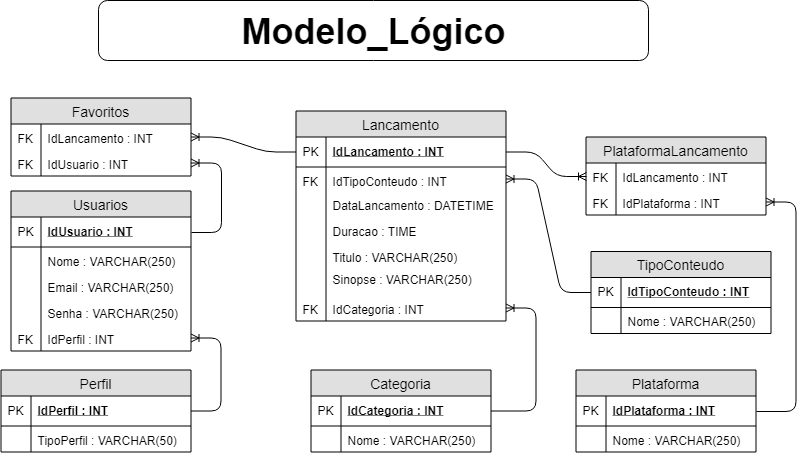
* BackEnd:
  + Microsoft Visual Studio
* Swagger:
  + Postman
* Banco de dados:
  + Microsoft SQL Server Management Studio

Modelagem de Software

As modelagens de software é uma etapa para o planejamento e minimização do tempo de projeto, assim as modelagens ajudam na construção do banco de dados e facilitam o entendimento do armazenamento de dados, assim como traz uma visão geral das entidades e os relacionamentos entre elas. As modelagens seguem representadas abaixo, sendo elas Lógica, Física e Conceitual.

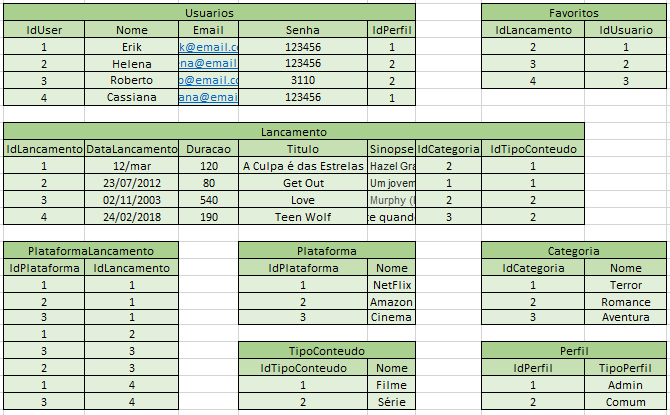
Modelo Lógico

A modelagem Lógica é aquela que traz não só as entidades, mas também seus dados e tipos de dados. Assim ela se assemelha ao resultado final do código do Banco de Dados, e mostra também as relações entre as tabelas e quais seus tipos.



Modelo Físico

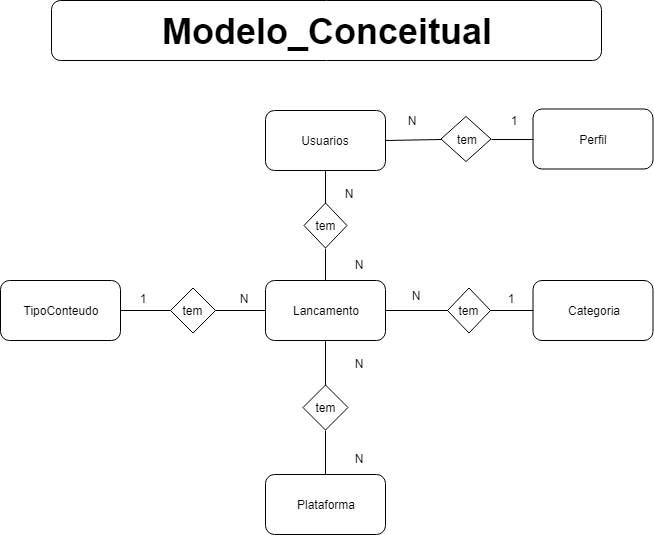
O modelo físico modelado na plataforma Excel observamos em si os dados que possuem na tabela, nele podemos administrar também as relações observando od Id´s de cada tabela. Semelhante ao Lógico porem nesta representação o foco são nos dados em si, e não em relações de entidades.



Modelo Conceitual

A conceitual é aquela em que focamos nas entidades presentes, e vemos um resumo de como se relacionam através dos caracteres 1 e N. Ela serve como uma fonte fundamental de consulta, pois resume para o código qual entidade que carrega uma FK (Chave de relacionamento entre tabelas) referencial a uma outra tabela

.



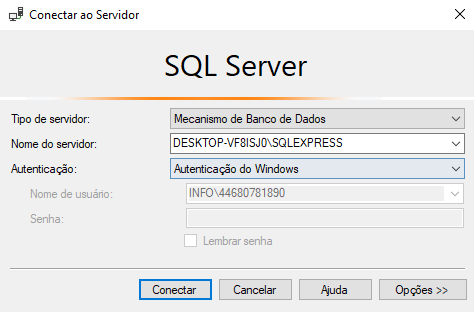
Criando Scripts do Banco de Dados

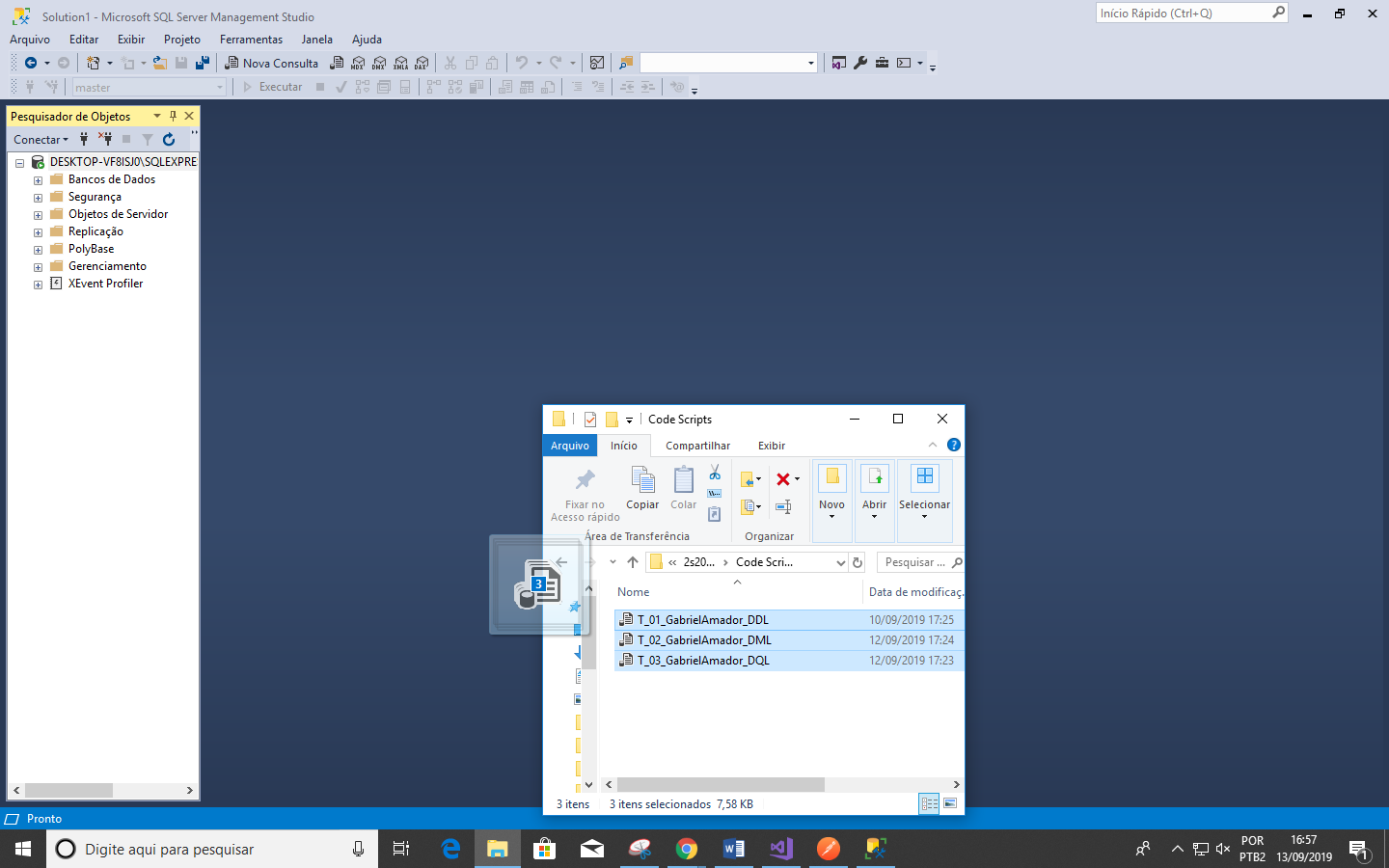
Notas

* Para organizar o banco de dados, foi desenvolvido três scripts separados DDL, DML e DQL.
* Antes de rodar a aplicação backend é necessário ter o banco de dados criado, e sem problemas.
* É necessário ter instalado a aplicação Microsoft SQL Server Studio.

Criando o Script

**1º -** Abra a aplicação, atente-se no tipo de servidor e autenticação, o nome do servidor será diferente em cada máquina. Aperte o botão conectar;



**2º -** Ao se conectar, abra o diretório de pastas e encontre o caminho “2s2019-sprint-1-bd-opflix\Code Scripts”, antes lembre-se de clonar o repositório GitHub disponibilizado no final do documento em **Referencias**. Nesse diretório haverá três documentos, arraste e solte cada um deles no SQL Server, se preferir selecione todos de uma só vez.

Rodando o Script

**1º** - Com a estrutura já criada, é preciso dar o “run” no projeto. No projeto “T\_01\_GabrielAmador\_DDL” e “T\_02\_GabrielAmador\_DML” selecione todo o código e clique em F5, então o projeto estará criado com dados salvos.

Deploy / Rodar Projeto

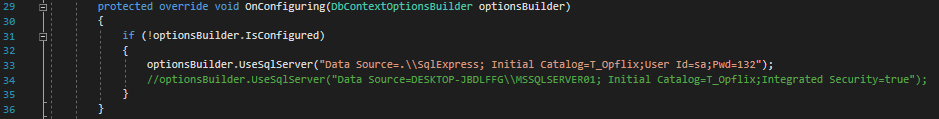
1º - Abra o executável “Visual Studio” já instalado na máquina;

2º - No diretório de pastas encontre o projeto “2s2019-sprint-1-bd-opflix” (repositório clonado do GitHub) e acesse: “2s2019-sprint-1-bd-opflix\BackEnd - API\Senai.OpFlix.WebApi”;

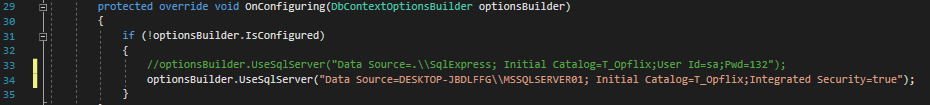
3º - Com um duplo clique no mouse abra o projeto “Senai.OpFlix.WebApi.sln” na plataforma mencionada;

4º - Com o projeto aberto, em **Gerenciador de soluções** acesse a pasta “**Contexts”** e abra o arquivo “**OpflixContext”,** você deve editar as linhas 33 e 34 de modo que as barras sejam editadas**:**

**Antes:**



**Depois:**



**5º Agora na pasta Repositories, acesse o arquivo “LancamentoRepository” e altere as linhas 15 e 16, de modo que:**

**Antes:**



**Depois:**



**6**º - Enfim, pressione a tecla F5 ou F11 e o projeto é executado.

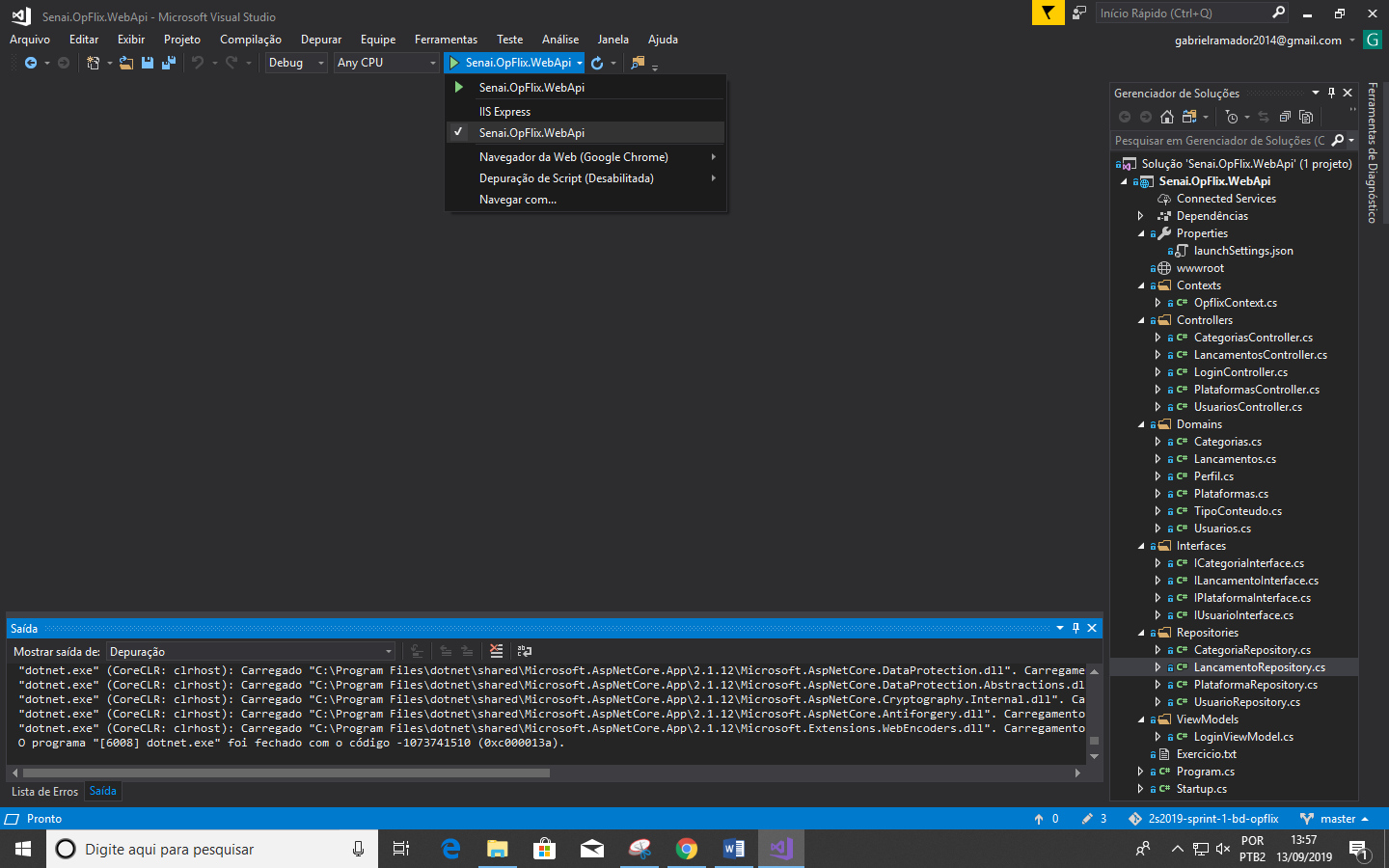
Swagger

Definição

Através do Swagger podemos entender as funcionalidades que o projeto possui. Na documentação são mostrados os EndPoints em que, cada um deles representam um recurso.

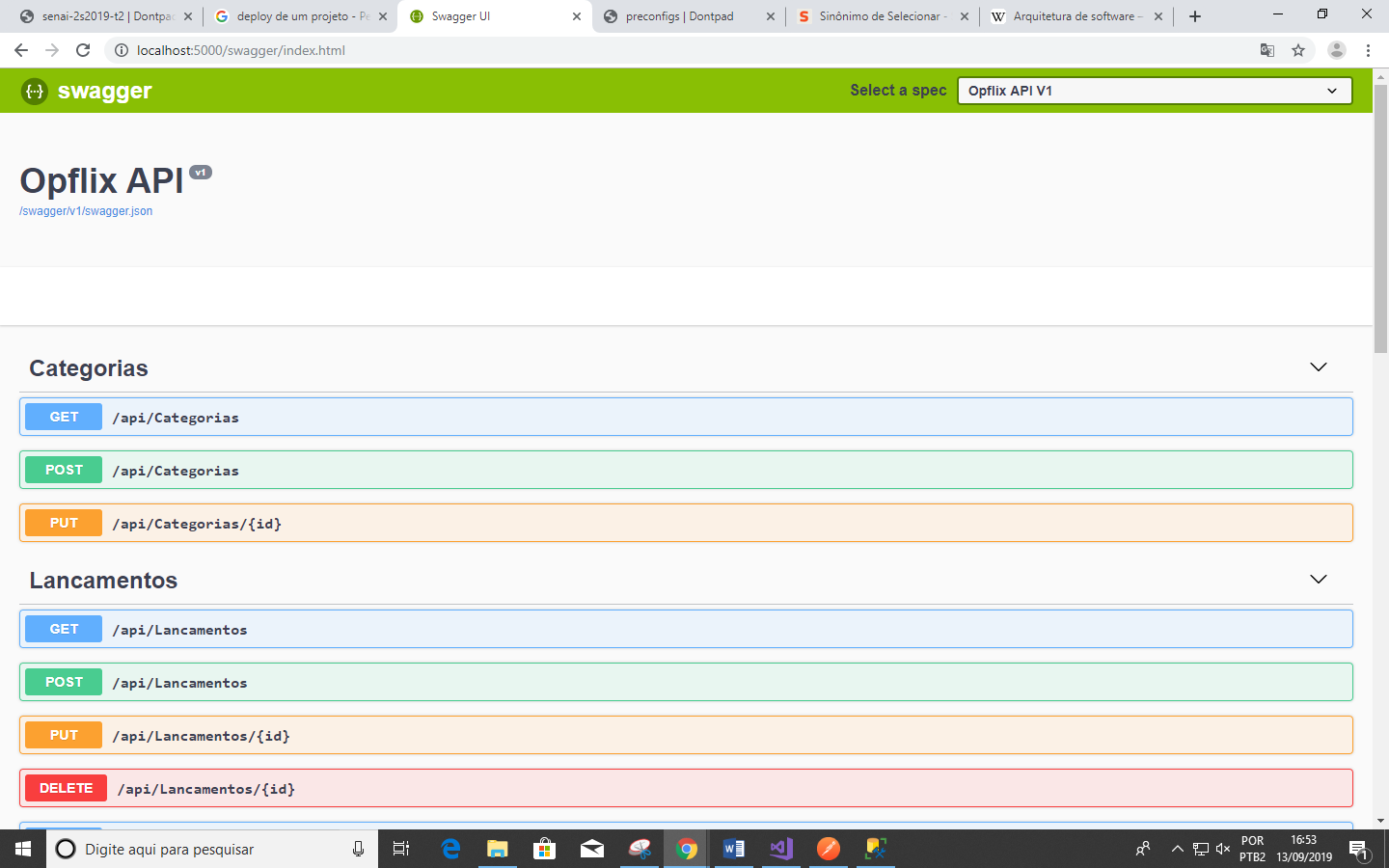
Acessando

**1º -** Antes é necessário “rodar o projeto”, sendo assim, na plataforma Visual Studio (Microsoft) pressione a tecla F11 ou então com o mouse selecione a opção de execução na parte superior do ambiente de desenvolvimento, lembre-se de selecionar a opção com o nome “Senai.OpFlix.WebApi”;



**2º -** Após a execução, abra um navegador e acesse a URL : “<http://localhost:5000/swagger/index.html>”;

**3º -** Com o Swagger aberto, as funcionalidades estarão disponíveis para a visualização e teste.



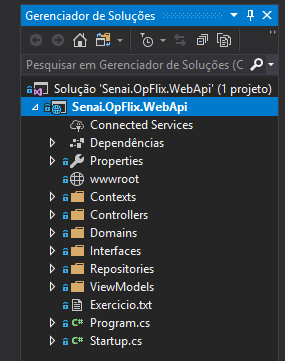
Pacotes Nuget

Pacotes Necessários

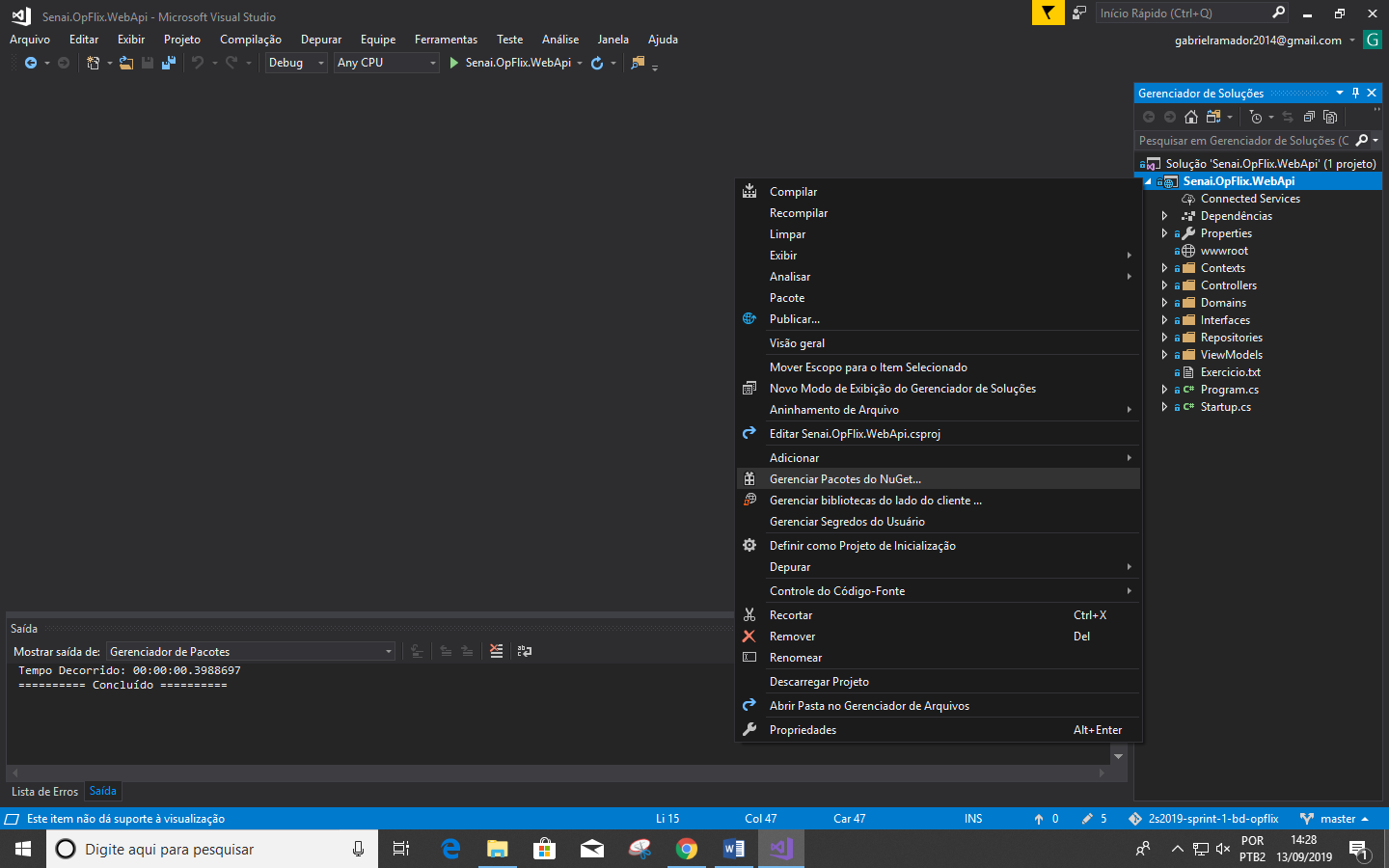
* **Uso Swagger:**
  + Swashbuckle.AspNetCore 4.0.1
* **Uso SqlClient:**
  + System.Data.SqlClient 4.6.1
* **Uso Entity Framework Core:**
  + Microsoft.EntityFrameworkCore.SqlServer 2.1.11
  + Microsoft.EntityFrameworkCore.SqlServer.Design 1.1.6
  + Microsoft.EntityFrameworkCore.Tools 2.1.11
* **Uso JSON Web Token:**
  + Microsoft.AspNetCore.Authentication.JwtBearer 2.1.1
  + System.IdentityModel.Tokens.Jwt 5.5.0

Como Instalar

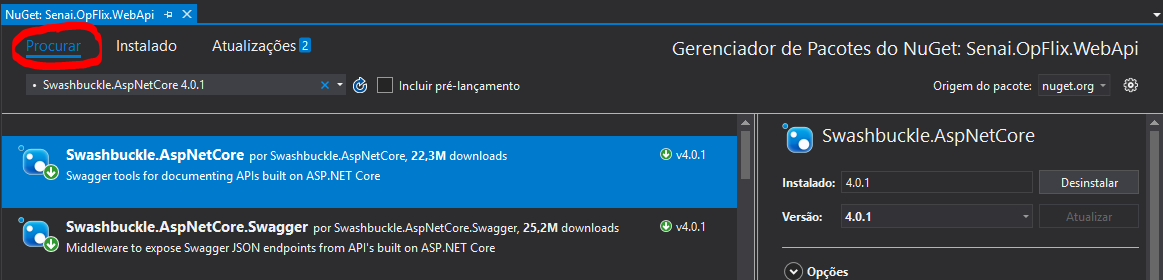
**1º -** Com o projeto aberto, ao lado direito da tela, na janela “**Gerenciador de Soluções**”, com o botão direito do mouse selecione a opção com o nome do projeto;



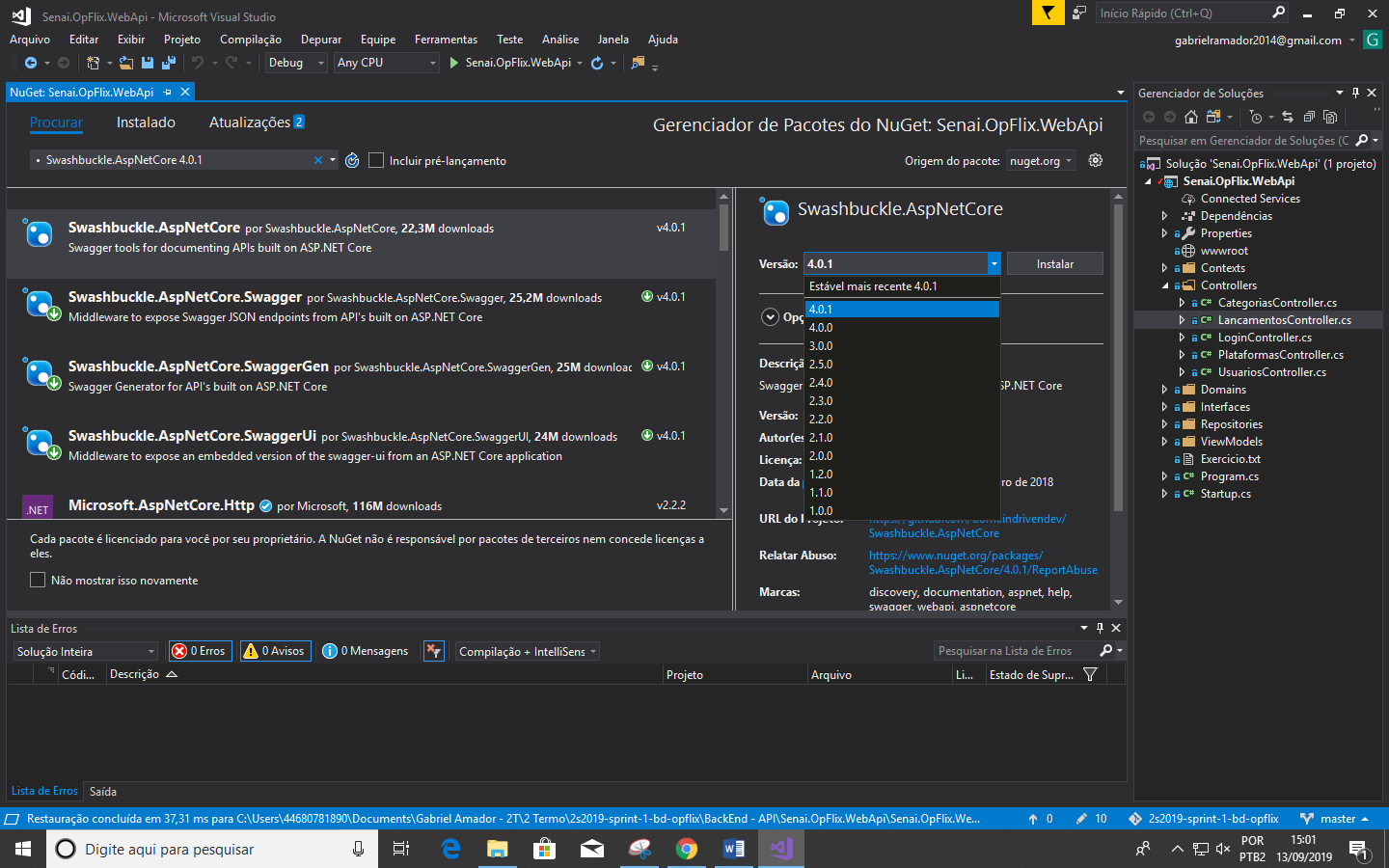
2º - Dentre as opções mostradas, escolha “**Gerenciar Pacotes do Nuget”**



**3º -** Com a nova janela aberta, acesse a sub página “Procurar”, e na barra de pesquisa digite por vez, os pacotes indicados na sessão acima.;



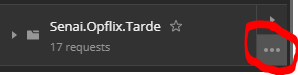
**4º -** Selecionado o pacote, escolha a versão desejada e instale-o.



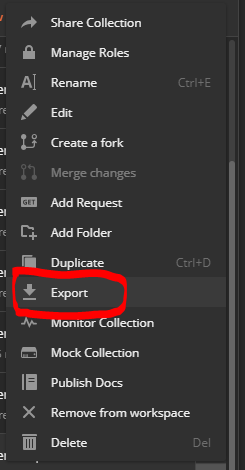
Postman

Exportando

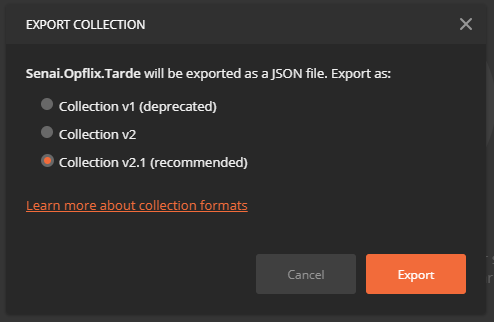
**1º -** Primeiramente, com o aplicativo instalado na máquina, ao lado esquerdo da tela selecione a pasta de EndPoints a ser exportada e clique no ícone circulado em vermelho;



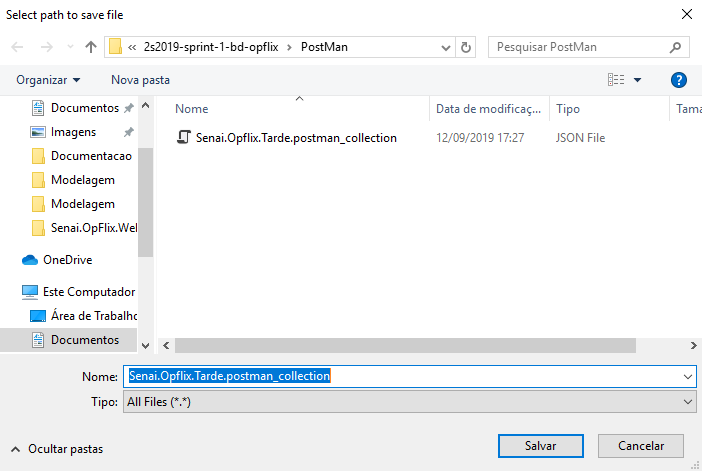
**2º -** No menu aberto, selecione a opção “Export”;



**3º - Na aba de exportação, haverá três opções, ative a opção “Collection v2.1 (recommended)” e exporte o projeto;**

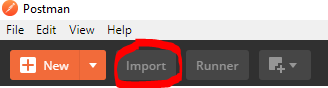


**4º -** Nesse ultimo passo, defina o nome do arquivo e o diretório onde será armazenado o projeto;

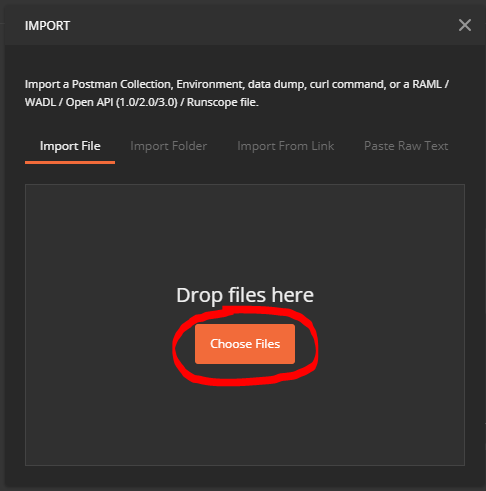


Importando

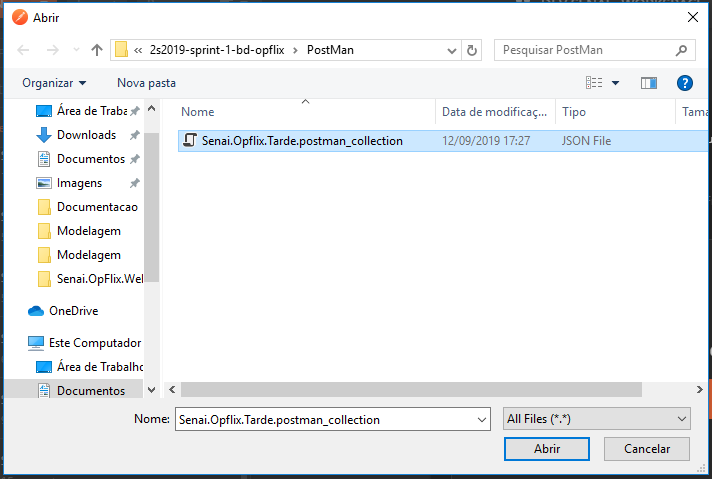
**1º -** Aberto o aplicativo Postman, selecione o botão “Import” no canto superior esquerdo da tela;



**2º -** Na nova aba, há dois caminhos ou arrastar e soltar o arquivo (arquivo de um export anterior) no local solicitado, ou selecionar o caminho do diretório. Respectivamente, pelo segundo caminho clique no botão “Choose Files”



**3º -** Selecione o projeto e clique em abrir.



Funcionalidades

Web

* **Resumo:** As funcionalidades do projeto se especializam pelas páginas, e elas se baseiam nas funcionalidades já criadas na API e mostradas no Post-Man, ou seja um cadastros, listagens, atualizações e deleções. Algumas funcionalidades são restritas a perfis de usuário (comum e admin), como exemplo, os cadastros são efetivados pelo admin.
* **Cadastro:** Usuários (comum e admin) , Lançamentos, Categorias, Lançamentos, Plataformas e Favoritos
* **Atualização**: Lançamento, Categorias e Plataformas;
* **Listagem**: Lançamentos, Categorias e Plataformas;
* **Filtragem** de Lançamentos: Por categorias, Data de lançamento e Favoritos;
* **Login** **e Logout.**

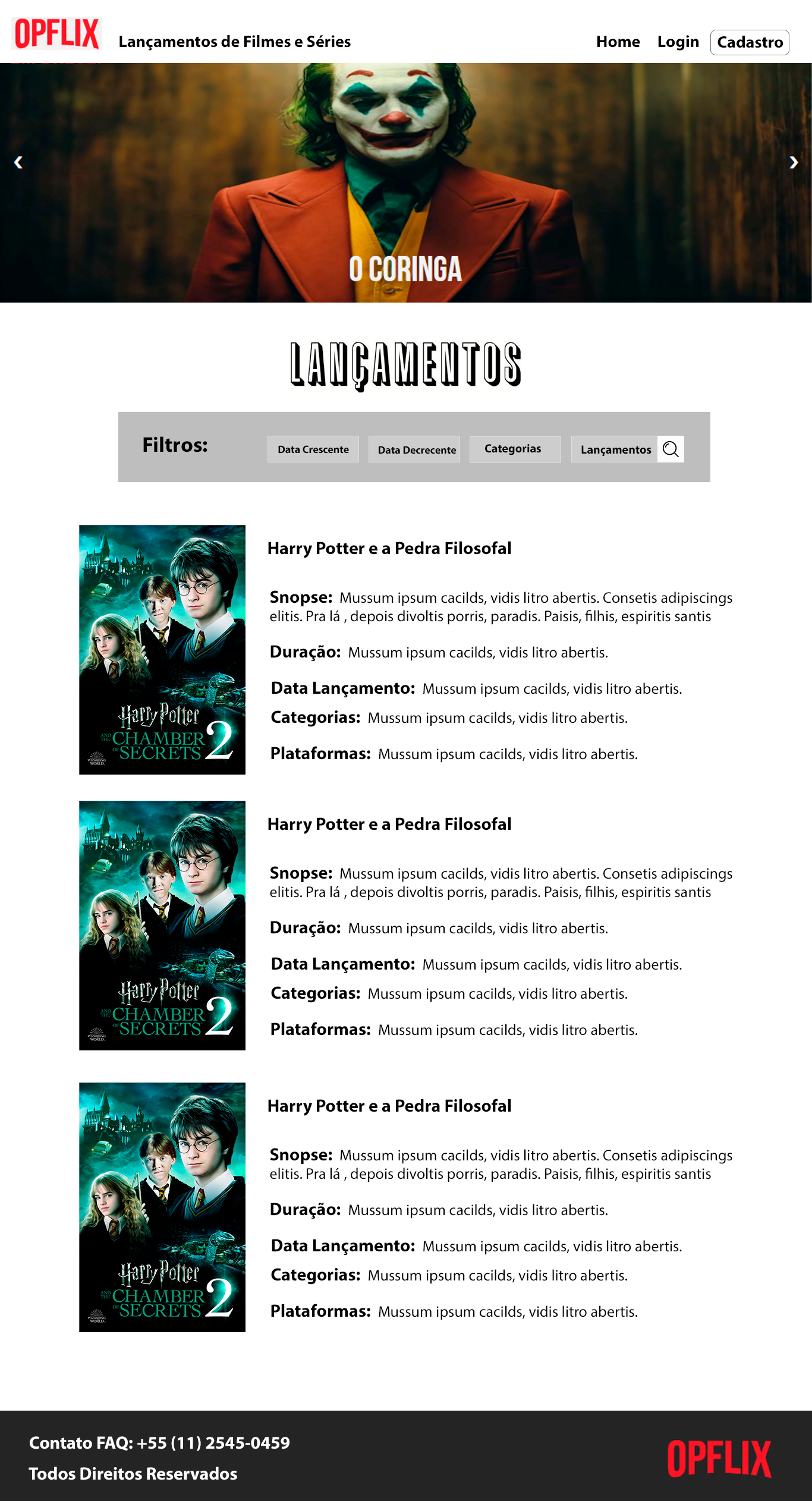
Mobile

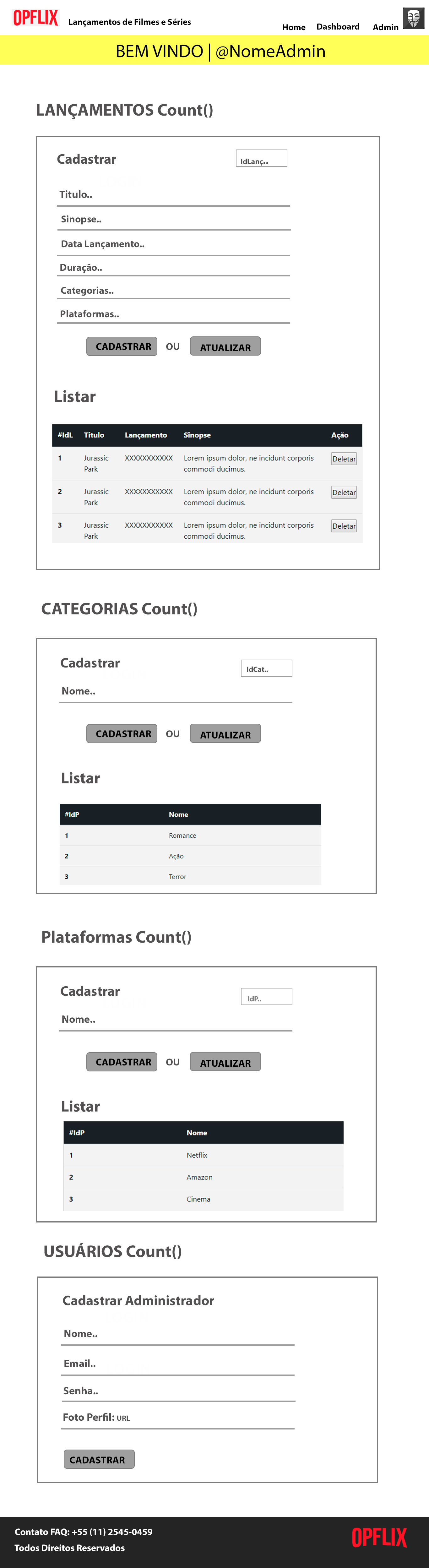
Protótipos

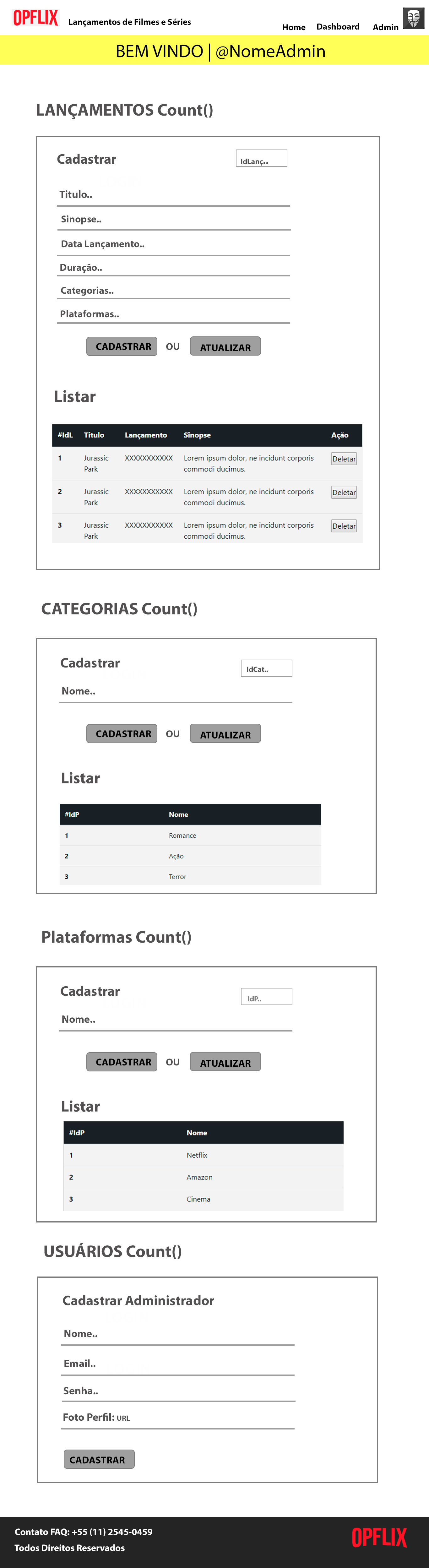
Web

* **Resumo:** A prototipação de alta e baixa fidelidade mostra ao cliente uma prévia de como será o resultado final. Abaixo pela ferramenta “Adobe Photoshop” foi realizado o Wire-frame(esqueleto) de alta fidelidade do site. O produto final conta com três principais páginas, Home, Login/Cadastro e Dashboard (Admin) com exceção da página “Not Found. Os componentes globais (Barra de navegação e rodapé) a todas as páginas foram criados com ‘jogo de contraste’, logo a página se mostra mais simples e fácil de entender.
* **Login e Cadastro:**
  + **Resumo:** Ambas funcionalidades foram inclusas juntas pensando na experiê­ncia do usuário facilitando o encontro pela navegação. O desingn foi realizado de forma limpa sem poluição visual e com cores agradáveis a vistas, elas também criam a identidade da empresa Opflix.



* **Home:**
  + **Resumo:** A home ou página principal do site traz a experiência efetiva do cliente. Nesta página apenas o usuário logado pode acessa-la**.** Nela o usuário pode verificar a lista de lançamentos, filtrá-la, favorizar prediletos e realizar logout. A tipografia é uma característica realçada nesta página e cada lançamento possui informações próprias.
* **Dashboard:**
  + **Resumo:** Esta página é exclusiva do perfil administrador, ela é a mais extensa e com mais funcionalidades. Se baseia em cadastros, listagens e atualizações de dados.

****

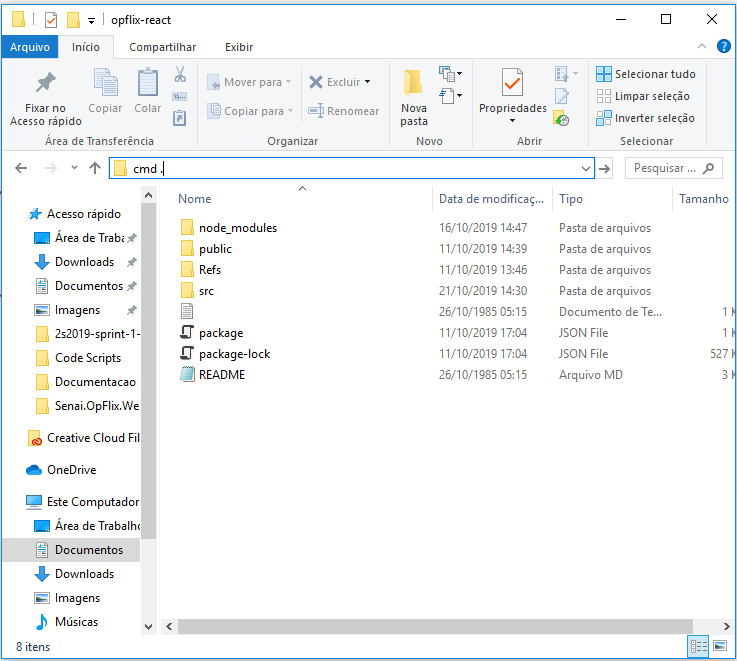


Mobile

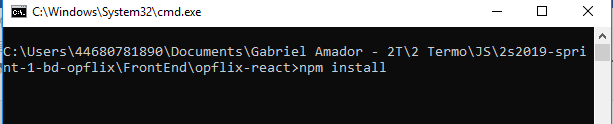
Front-End

Web

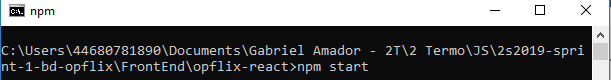
**1º -** Como primeiro passo acesse o link “<https://nodejs.org/en/>” e baixe a versão do node.js mais adequada com seu computador;

**2º -** Agora o repositório já clonado na pasta “2s2019-sprint-1-bd-opflix/FrontEnd/opflix-react/” digite como na imagem “cmd .” na barra de navegação e aperte a tecla ENTER;

**3º -** Com o prompt de comando aberto, na aba inferior “TERMINAL” execute o comando “npm install” e espere a conclusão da instalação, que pode demorar alguns minutos.



**4º** - Por fim, digite “npm start” e espere, a partir daí o sistema estará rodando.



Mobile

­

Mobile

Arquitetura do Projeto

Banco de Dados

**Linguagem**: A linguagem relacional utilizada no banco de dados foi o Microsoft SqlServer

**Organização:** Os scripts foram separados em três partes, são elas a DDL (Data Definition Language) que possui os comandos de criação, deleção e alteração das tabelas e do banco de dados. DML (Data Management Language) que possui códigos para inserir, atualizar e deletar dados das tabelas. Por fim há a DQL (Data Query Language) que seleciona os dados da tabela.

**Entidades:** No projeto foram criadas as entidades Usuários, Lançamentos, Categorias e Plataformas, já como tabelas intermediárias há os Favoritos e as PlataformasLançamentos, por final as tabelas de Perfil(permissão) e TipoConteúdo foram separadas para a performance do projeto;

Back-End

**Linguagem:** A linguagem utilizada foi o C-Sharp ou C#, linguagem em alta no mercado, performática, e, entre as 4 linguagens mais utilizadas atualmente.

**Organização:** O projeto foi organizado pela estrutura **MVC** (Model View Controller), onde na aplicação quando o usuário fizer uma requisição o Controller irá recebê-la e enviar ao Repositório por intermédio da Interface que fará a ponte com o banco de dados receberá o dado da requisição e retornará uma resposta ao Controller que por sua vez irá passá-la para a View (tela do usuário). Esse método organiza o código do desenvolvedor e facilita o desenvolvimento do software. Na programação orientada a objetos, criamos os objetos, chamados como modelos (models), entidades ou domains. Visando a organização do código também foi adotado a metodologia REST.

O Controller tem em si diversas funcionalidades, elas estão amostras no Swagger da aplicação. As Interfaces servem como uma barreira de segurança para a aplicação, onde nelas há disponíveis apenas as funcionalidades do repositório, mas não é possível ver o processo como é feito cada requisição. O Repositório como já mencionado, faz a ligação com o banco de dados, nesta aplicação esta necessidade foi sanada pelo uso do SqlClient em conjunto do Entity Framework Core.

O projeto utiliza mais barreiras de seguranças por via das autorizações dos usuários, onde há a validação e restrição do uso de funcionalidades do aplicativo pelas informações dos usuários.

Verbos: Nas requisições Web, os verbos utilizados pela aplicação se restringiram ao GET, PUT, POST e DELETE

Referências

Links

GitHub: <https://github.com/amadorsenai/2s2019-sprint-1-bd-opflix>