**DOCUMENTAÇÃO**

OpFlix

Alameda Barão de Limeira,

539 – São Paulo SP sp.senai.br

Sumário

1. [Resumo 2](#_Toc533767843)

[Objetivos](#_Toc533767844)

1. [Descrição do projeto 2](#_Toc533767845)

[Resumo do projeto](#_Toc533767846)

1. [Modelagem de Software 3](#_Toc533767847)

[Modelo Lógico](#_Toc533767848)

[Modelo Físico](#_Toc533767849)

[Modelo Conceitual](#_Toc533767850)

1. [BackEnd 4](#_Toc533767847)

Deploy

Swagger

Nuget

Projeto

Postman

1. [Referências](#_Toc533767847) 5

Links

1. Resumo

**Objetivos**

Este documento tem como objetivo demonstrar todas as funcionalidades da plataforma OpFlix, que visa demonstrar todos os lançamentos de Filmes/Séries perante seus devidos conhecimentos na área.

1. Descrição do Projeto

O OpFlix tem como intuito oferecer a divulgação de novos lançamentos.

**Resumo do Projeto**

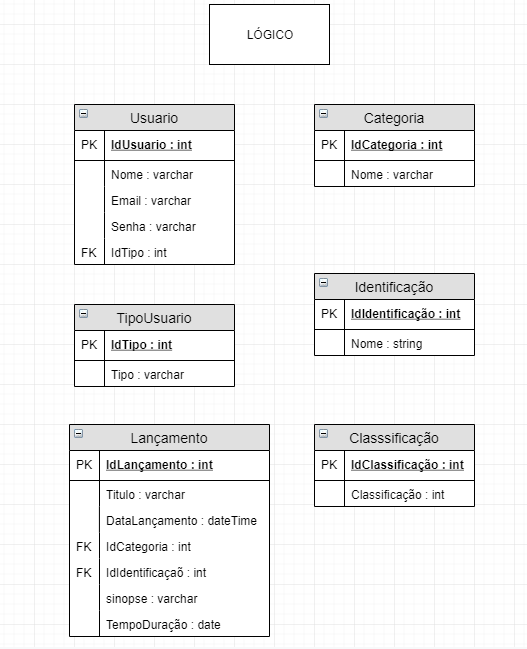
O OpFlix tem a disponibilização de novidades de inúmeros e inúmeras Filmes/Séries que são conhecidas por uma quantidade incrível de pessoas, onde é conhecido como um passatempo de jovens e adultos.

1. Modelagem de Software

Esta sessão tende a demonstrar o que será desenvolvido no banco de dados e quais são as suas representações.

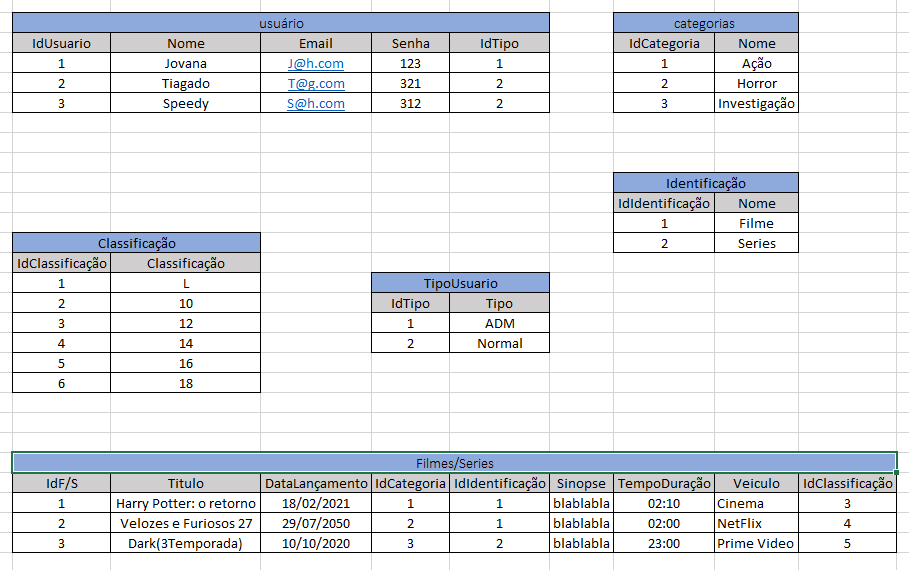
**Modelo Lógico**

A modelagem conceitual baseia-se no mais alto nível e deve ser usada para envolver o cliente, pois o foco aqui é discutir os aspectos do negócio do cliente e não da tecnologia.

****

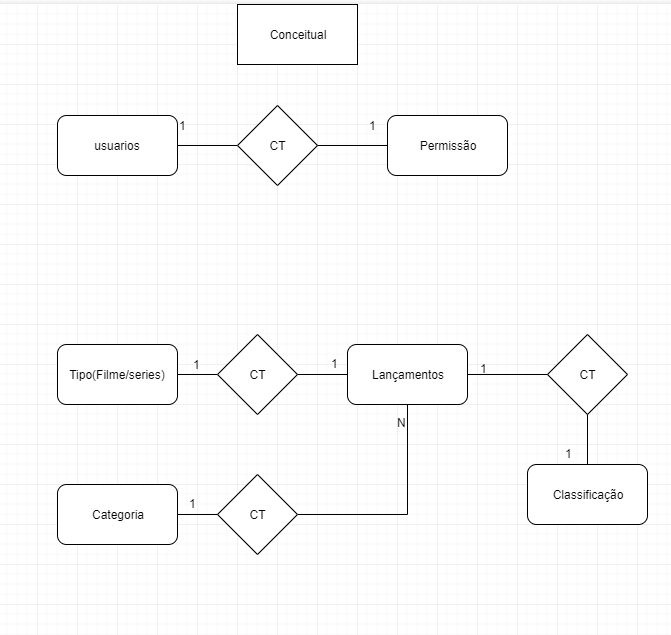
**Modelo Físico**

O modelo lógico já leva em conta algumas limitações e implementa recursos como adequação de padrão e nomenclatura, define as [**chaves primárias** e **estrangeiras**](https://www.luis.blog.br/chave%20primaria-chave-estrangeira-e-candidata.aspx), normalização, integridade referencial, entre outras.

****

**Modelo Conceitual**

No modelo físico fazemos a modelagem física do modelo de banco de dados.

****

1. BackEnd

**Deploy**

Após a elaboração do projeto, criação, avaliação e testes terem sido realizados, temos a questão da publicação e liberação dos projetos para o cliente, onde precisamos ter maneiras simples de implantação do lado do usuário para que o mesmo possa fazer uso da aplicação, seja ela web, mobile ou desktop.

Pensando neste ponto, é necessário termos formas de deploy para que estas aplicações sejam disponibilizadas de forma mais simples e intuitiva.

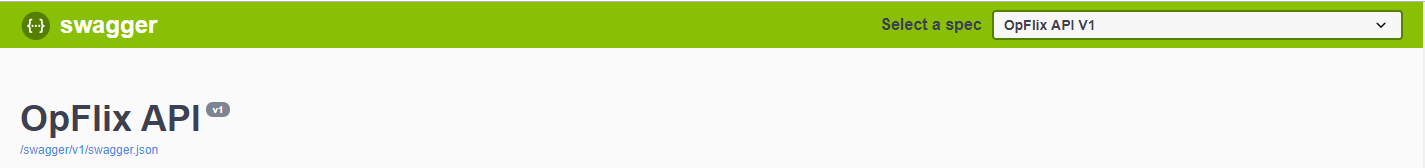


**Swagger**

O Swagger é uma forma de trazer uma melhor organização para que o Cliente fique ciente do que o programador está fazendo e os passos que são tomados, ou seja é uma documentação mais fácil de acessar.

Ao colocar o projeto do Visual Studio 2017 em execução, e esperar com que ele abra a aba do Google ou Edge, vá até a URL e coloca o link abaixo.

[**http://localhost:5000/swagger/index.html**](http://localhost:5000/swagger/index.html)

****

**Pacotes NuGet**

Links baixado pelos pacotes da NuGet.

**- - - - - - - - - - - - - - - - - - - - - -- - - - - - - - - -**  Swagge**r - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

Swashbuckle.AspNetCore 4.0.1

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** SqlClient**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

System.Data.SqlClient 4.6.1

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** EFCore **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

Microsoft.EntityFrameworkCore.SqlServer 2.1.11

Microsoft.EntityFrameworkCore.SqlServer.Design 1.1.6

Microsoft.EntityFrameworkCore.Tools 2.1.11

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** JWT **- - - - - - - - - -- - - - - - - - - - - - - - - - - - - - - - - - - -**

Microsoft.AspNetCore.Authentication.JwtBearer 2.1.1

System.IdentityModel.Tokens.Jwt 5.5.0

**Banco de Dados**

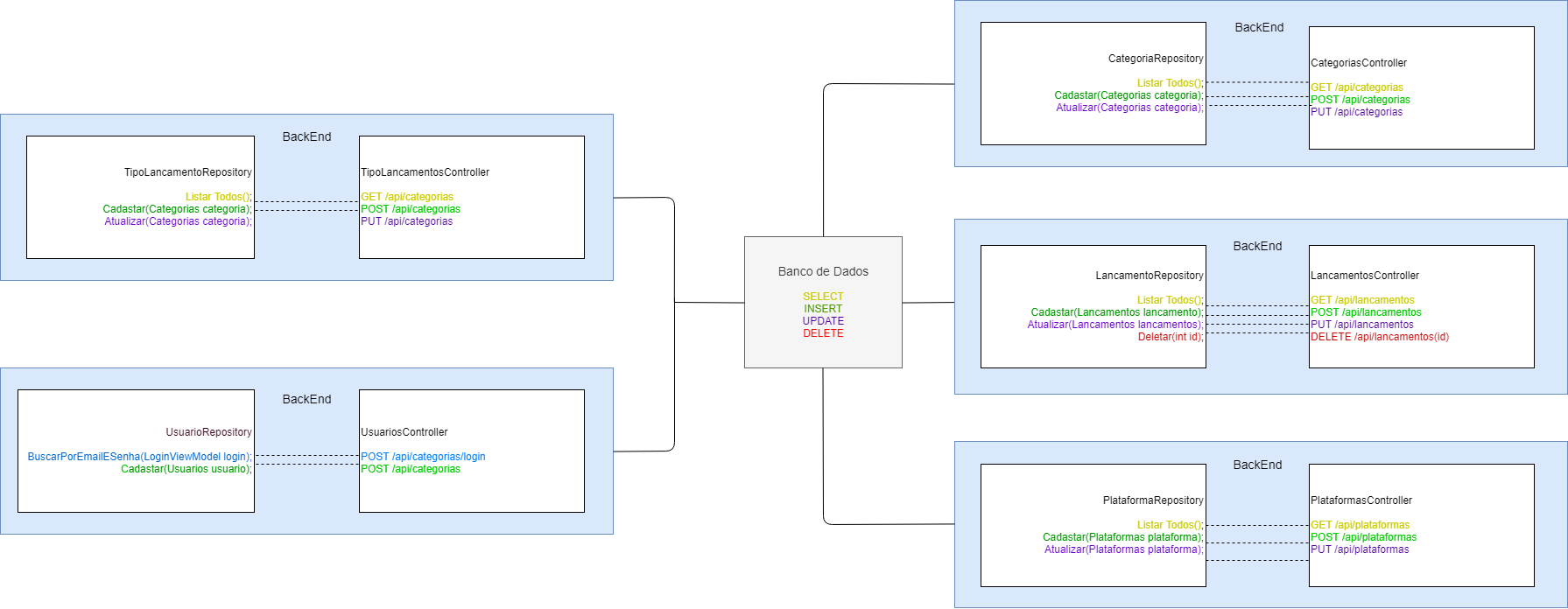
Todas as aplicações foram feitas no Microsoft SQL Server:

* Acesse o Primeiro link do SQL Server para acessar ao Banco de Dados (T\_01\_GiovannaRocha\_DDL.sql) e execute os comandos.
* Acesse o Segundo link do SQL Server para acessar ao Banco de Dados (T\_02\_GiovannaRocha\_DML.sql) e execute os comandos.
* Acesso o Terceiro link do SQL Server para acessar ao Banco de Dados (T\_03\_GiovannaRocha\_DQL.sql) e execute os comandos.

**Depois execute os Desafios Extras (T\_04\_GiovannaRocha\_DesafioExtra.sql)**

**Projeto**

Arquitetura do projeto:

****

Ferramentas Utilizadas

1. Visual Studio 2017
2. Sql Server
3. Swagger
4. Postman
5. Trello
6. GitHub

**Postman**

O Postman é uma ferramenta que tem como objetivo testar serviços RESTful (Web APIs) por meio do envio de requisições HTTP e da análise do seu retorno.



1. Referências

|  |
| --- |
|  |

**LINKS:**

<https://github.com/Giorocha/2s2019-sprint-1-bd/tree/master/OpFlix>