**Documentação**

**OpFlix**

Sumário

1. Resumo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Objetivos

1. Descrição\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Resumo do projeto

1. Banco de Dados (Sprint 1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1
2. Modelagem de Software\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Modelo Lógico

Modelo Físico

Modelo Conceitual

Cronograma

1. BackEnd – API (Sprint 2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4
2. FrontEnd – React (Sprint 3)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6
3. Mobile – React Native (Sprint 4)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8
4. MongoDB – (Sprint 5)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

Resumo

* Objetivos

Criar um banco de dados com as informações indiadas pelo cliente

Descrição do Projeto

Criar/desenvolver uma companhia chamada OpFlix, onde mostrará o catalogo

geral de lançamentos de filmes e séries com suas respectivas características.

* Resumo do Projeto

No projeto, foram necessários construir as modelagens exigidas. Após isso armazenamos e organizamos no banco de dados as informações da empresa.

Modelagem de Software

Modelagem de software é uma recuperação simplificada e algo real. Por

exemplo, uma planta de uma casa, ela apresenta todos os cômodos que o

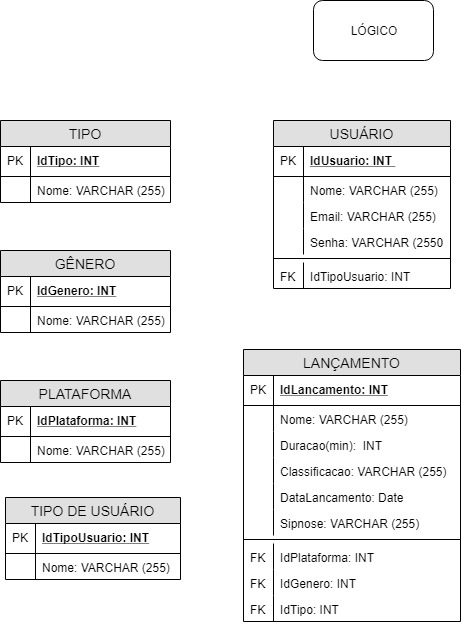
imóvel terá, suas medidas e as características da casa. Quando modelamos um

software, também identificamos o que esse futuro sistema deverá fazer.

Estaremos tratando as questões funcionais e seus fluxos de dados.

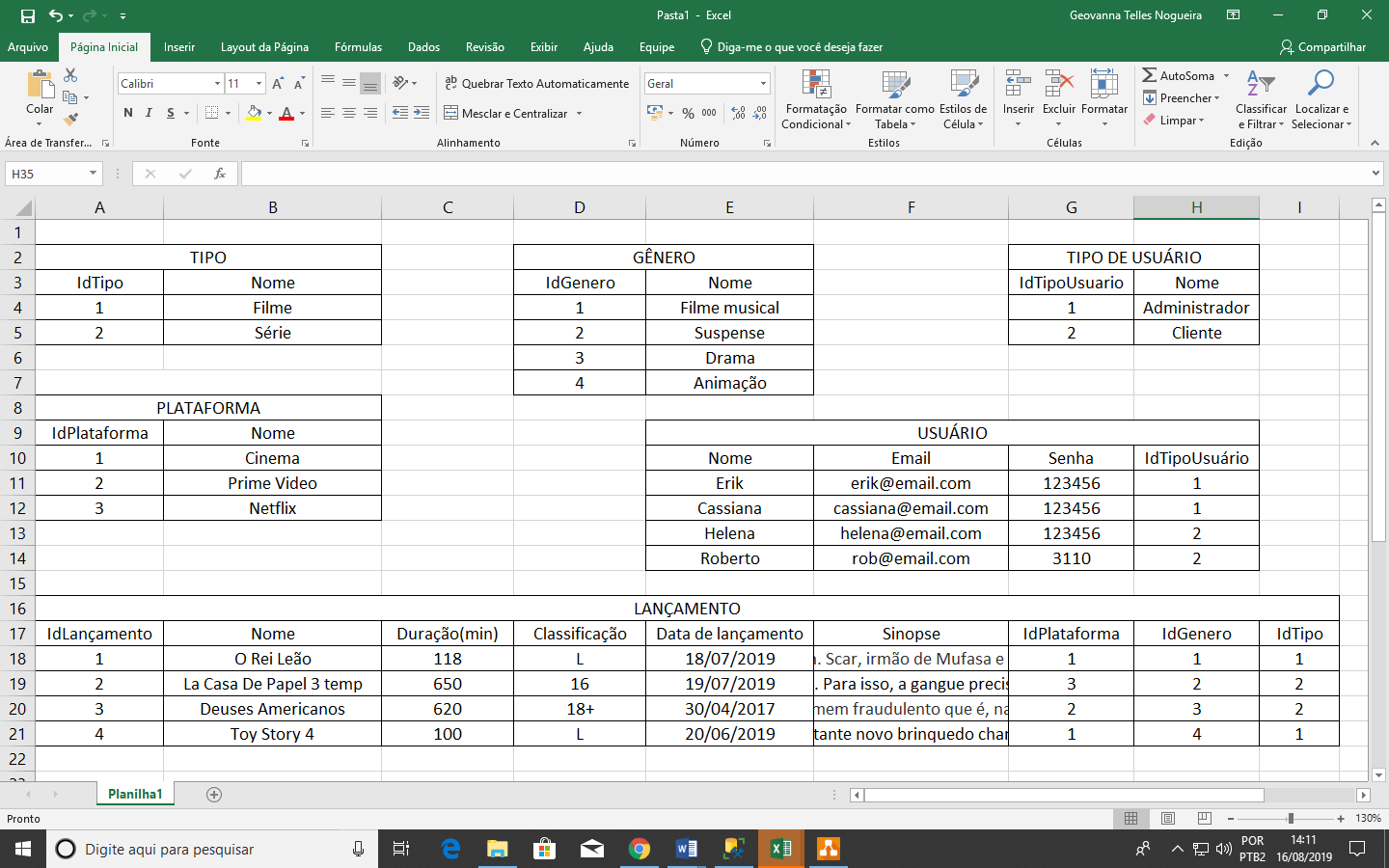
* Modelo Lógico

O modelo lógico já leva em conta algumas limitações e implementa recursos como adequação de padrão e nomenclatura, define as chaves primárias e estrangeiras, normalização, integridade referencial, entre outras. Para o modelo lógico deve ser criado levando em conta os exemplos de modelagem de dados criados no modelo conceitual.



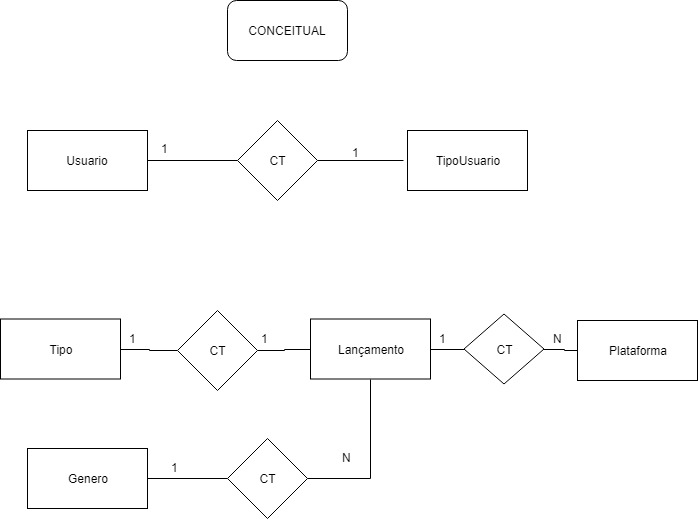
* Modelo Físico

No modelo físico fazemos a modelagem física do modelo de banco de dados. Neste caso leva-se em conta as limitações impostas pelo SGBD escolhido e deve ser criado sempre com base nos exemplos de modelagem de dados produzidos no item anterior, modelo lógico.



* Modelo Conceitual

É o modelo de mais alto nível, ou seja, que está mais próximo da realidade dos usuários. O nível conceitual é desenvolvido com alto nível de abstração, a partir dos requisitos do sistema, extraídos na fase de levantamento de requisitos.



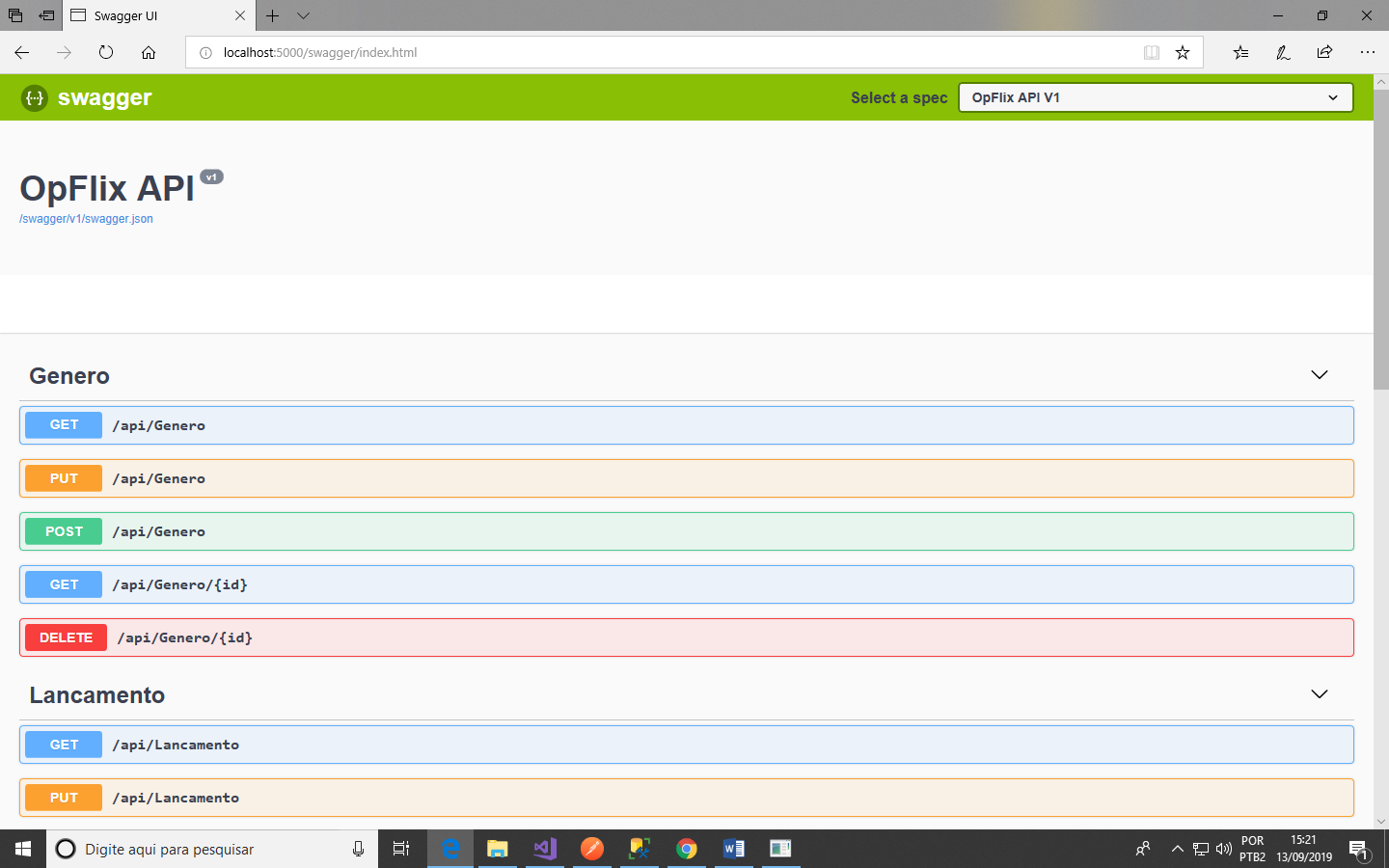
Cronograma

|  | Dia 1 | Dia 2 | Dia 3 | Dia 4 | Dia 5 | Dia 6 | Dia 7 | Dia 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Modelo Lógico** |  | X |  |  |  |  |  |  |
| **Modelo Físico** |  | X |  |  |  |  |  |  |
| Modelo Conceitual | X |  |  |  |  |  |  |  |

BackEnd – API (Sprint 2)

* Como acessar o Swagger

O Swagger é uma ferramenta que possibilita um fácil acesso para você fazer suas simulações, quando seu repositório e o seu controller estiverem terminados. Para acessar o Swagger basta você baixar o pacote dele no backend, colocar em uso, rodar o projeto e digitar na web a sua url: “htto://localhost:5000/Swagger/index.html”.



* Fazer deploy do projeto

Para fazer um deploy é necessário você ter o projeto salvo em alguma pasta do seu computador, usar um simples comando para copiar ou implantá-los com qualquer pacote de instalação que quiser. Isso serve para publicar componentes prontos dentro de um contexto de runtime.



* Quais pacotes do NuGet foram utilizados

Microsoft.EntityFrameworkCore.SqlServer - 2.1.11

Microsoft.EntityFrameworkCore.SqlServer.Design - 1.1.6

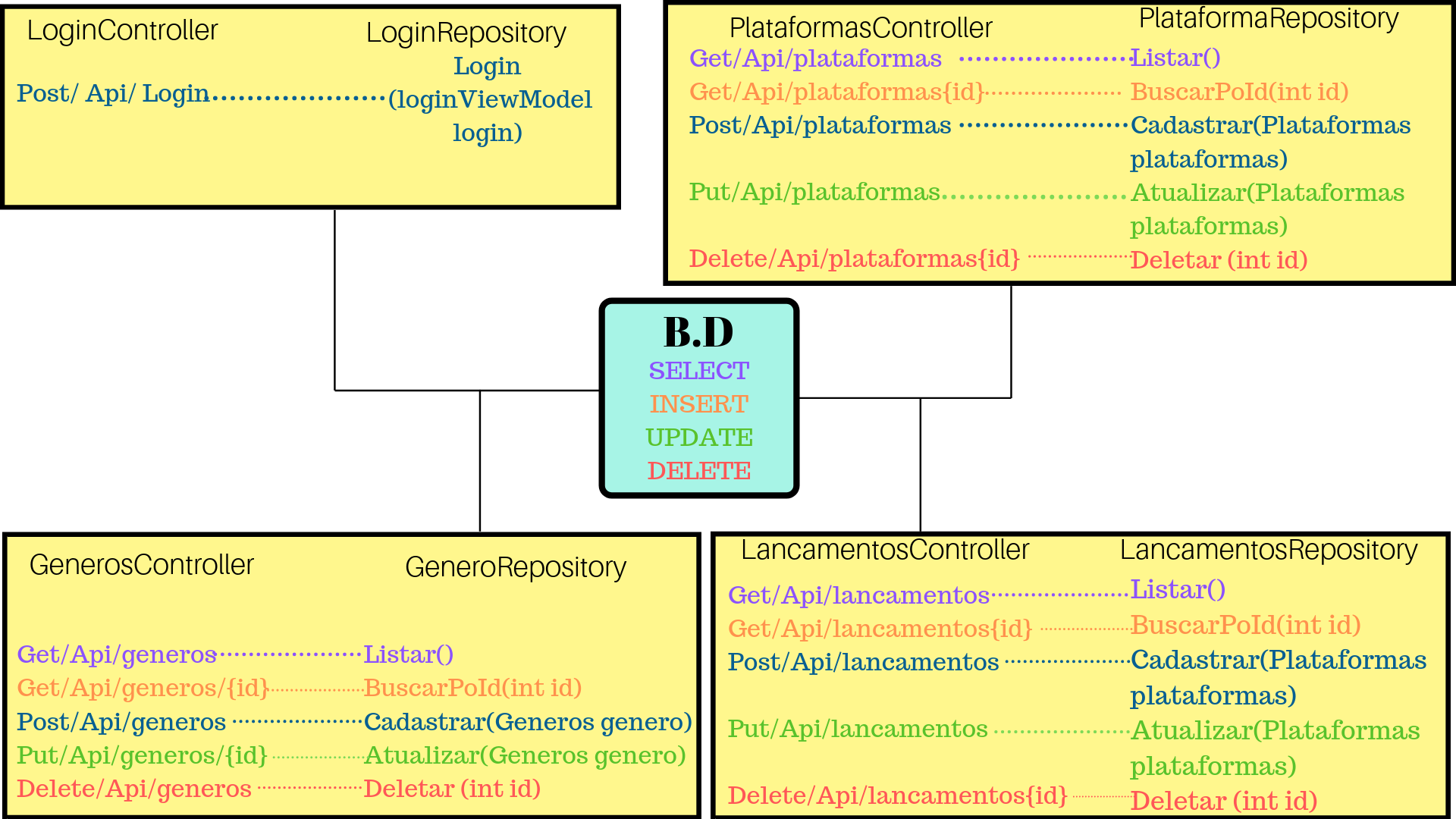
Microsoft.EntityFrameworkCore.Tools - 2.1.11

System.IdentityModel.Tokens.Jwt - 5.5.0

Microsoft.AspNetCore.Authentication.JwtBearer - 2.1.1

Swashbuckle.AspNetCore - 4.0.1

* Como está a arquitetura do projeto



* Como exportar e importar o Postman em outra máquina

Para importar/exportar o projeto é fundamental que tenha o arquivo pronto para importação e a exportação. Basta você clicar no botão “Import”, selecionar o arquivo/caminho que você deseja. Para exportação, é necessário clicar nos 3 pontinhos que ficam ao lado da sua pasta e clicar na opção export. Em seguida, será aberta uma caixa com opções do tipo de arquivo que quer criar, logo será aberto uma caixa para você salvar seu arquivo.

* Como realizar a criação de todo o banco e rodar o projeto no backend

Para isso damos um Scaffold

Exibir => Outras Janelas => Console do Gerenciador de Pacotes

Por exemplo:

“Scaffold-DbContext "Data Source=.\SqlExpress;Initial Catalog=T\_OpFlix;User Id=sa;Pwd=132;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Domains -ContextDir Contexts -Context OpFlixContext”

* Quais ferramentas foram utilizadas

Visual Studio 2017

Microsoft Sql Server Management Studio 18

Postman

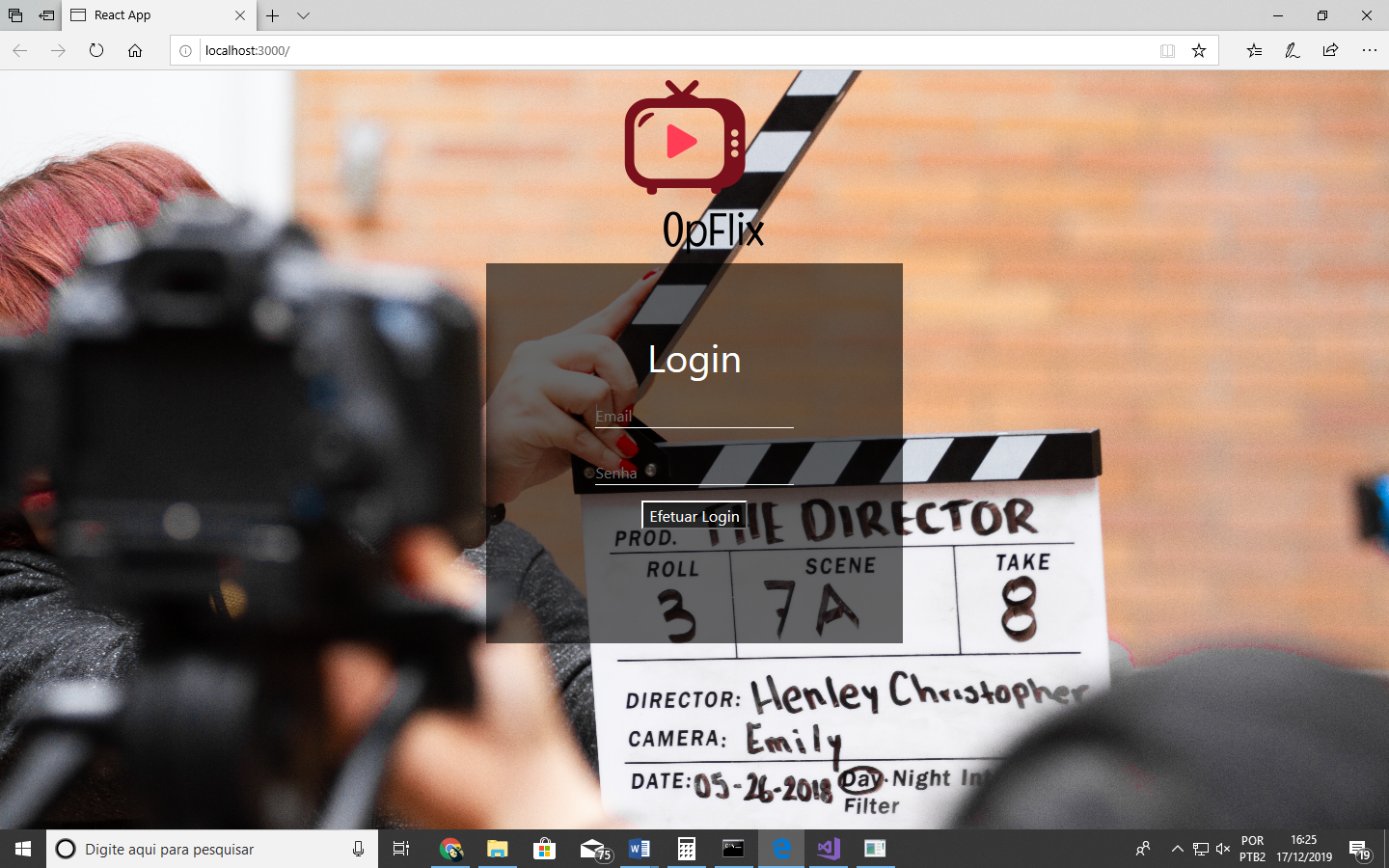
Swagger UI

FrontEnd – React (Sprint 3)

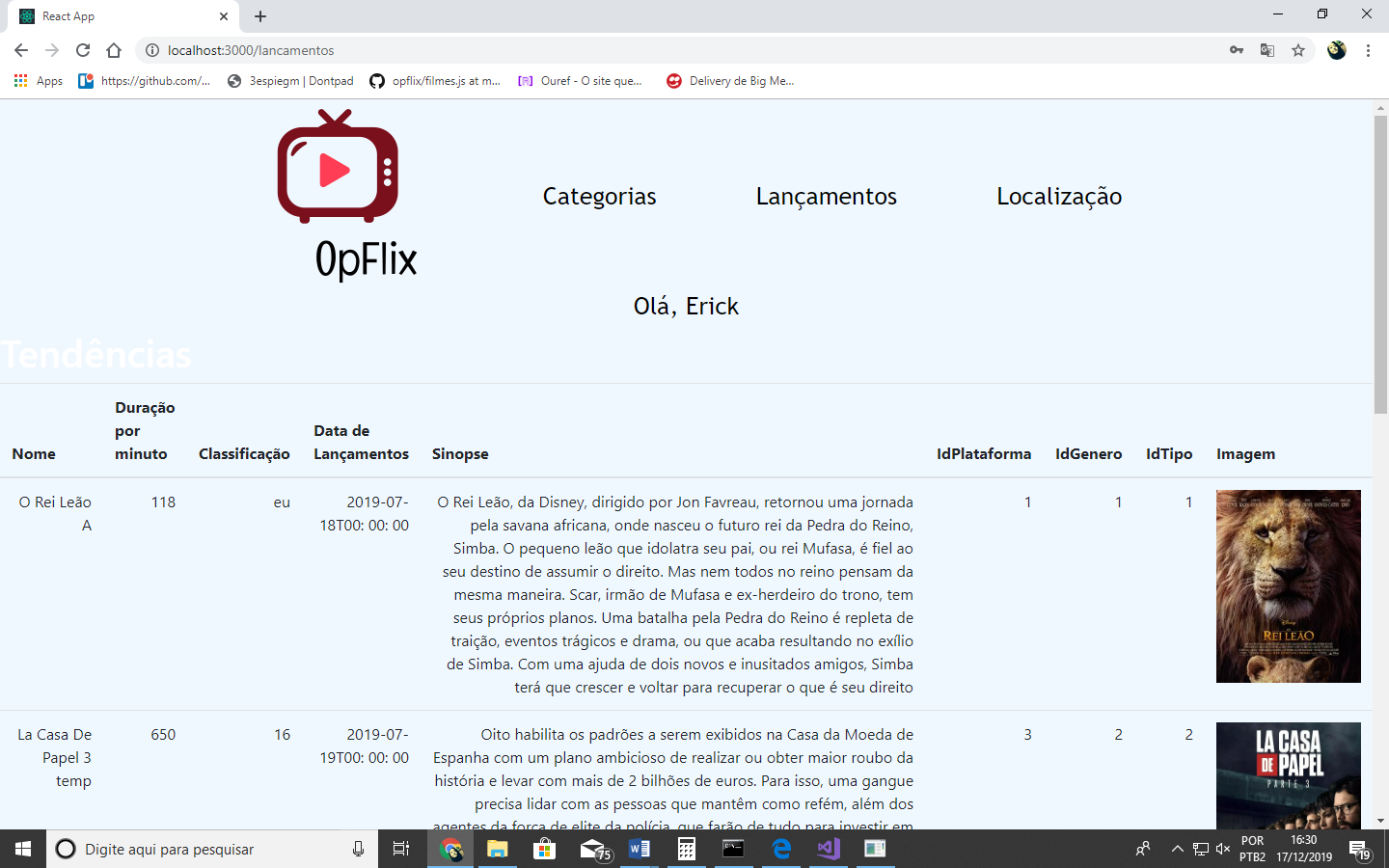
Administrador – Poderá cadastrar um novo lançamento e ter acesso a tudo do site.

Cliente – Poderá ver apenas a listagem de lançamentos e a página de localização.

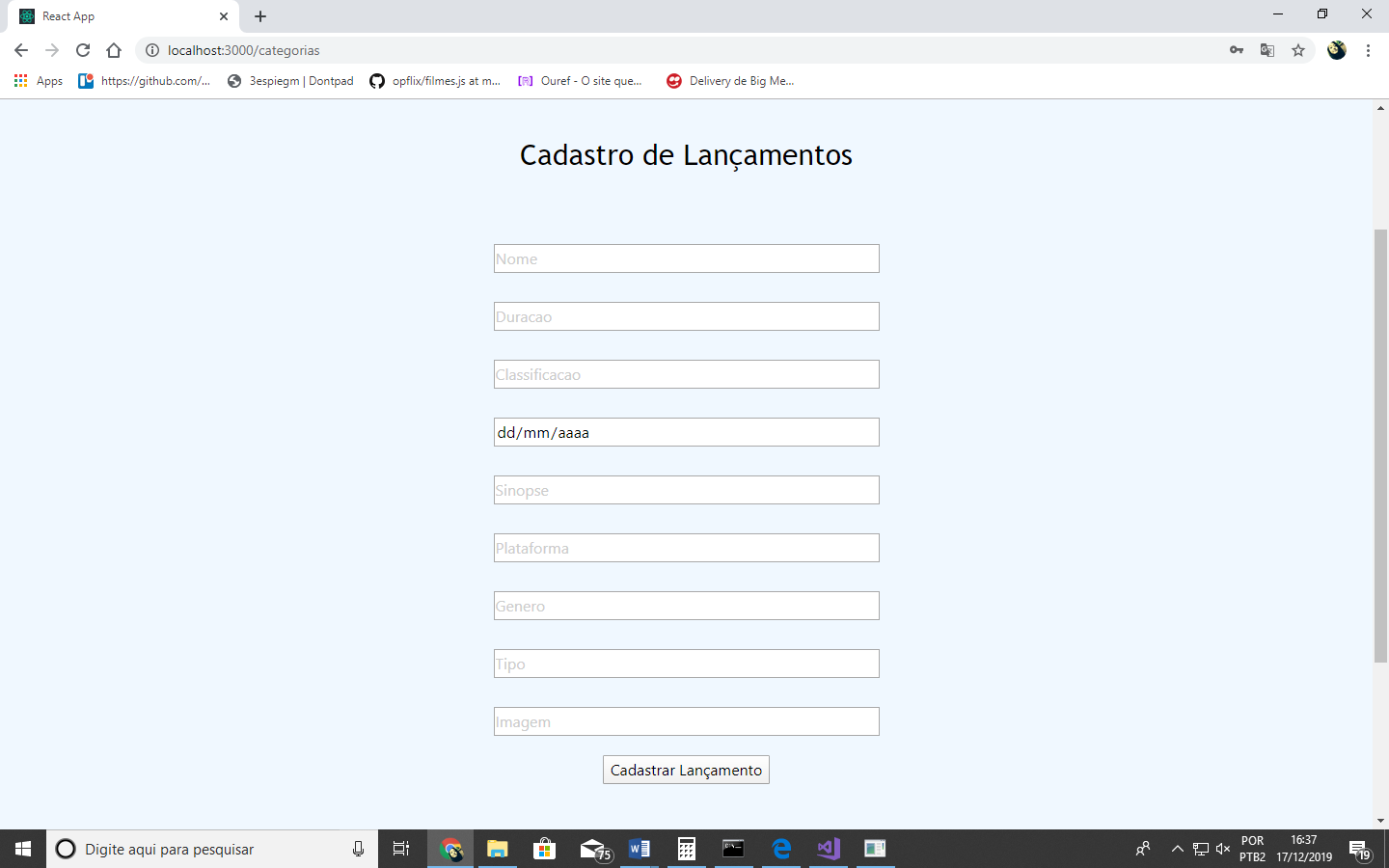
* Página de Login



* Página de Lançamentos



* Página de Cadastro de Lançamentos



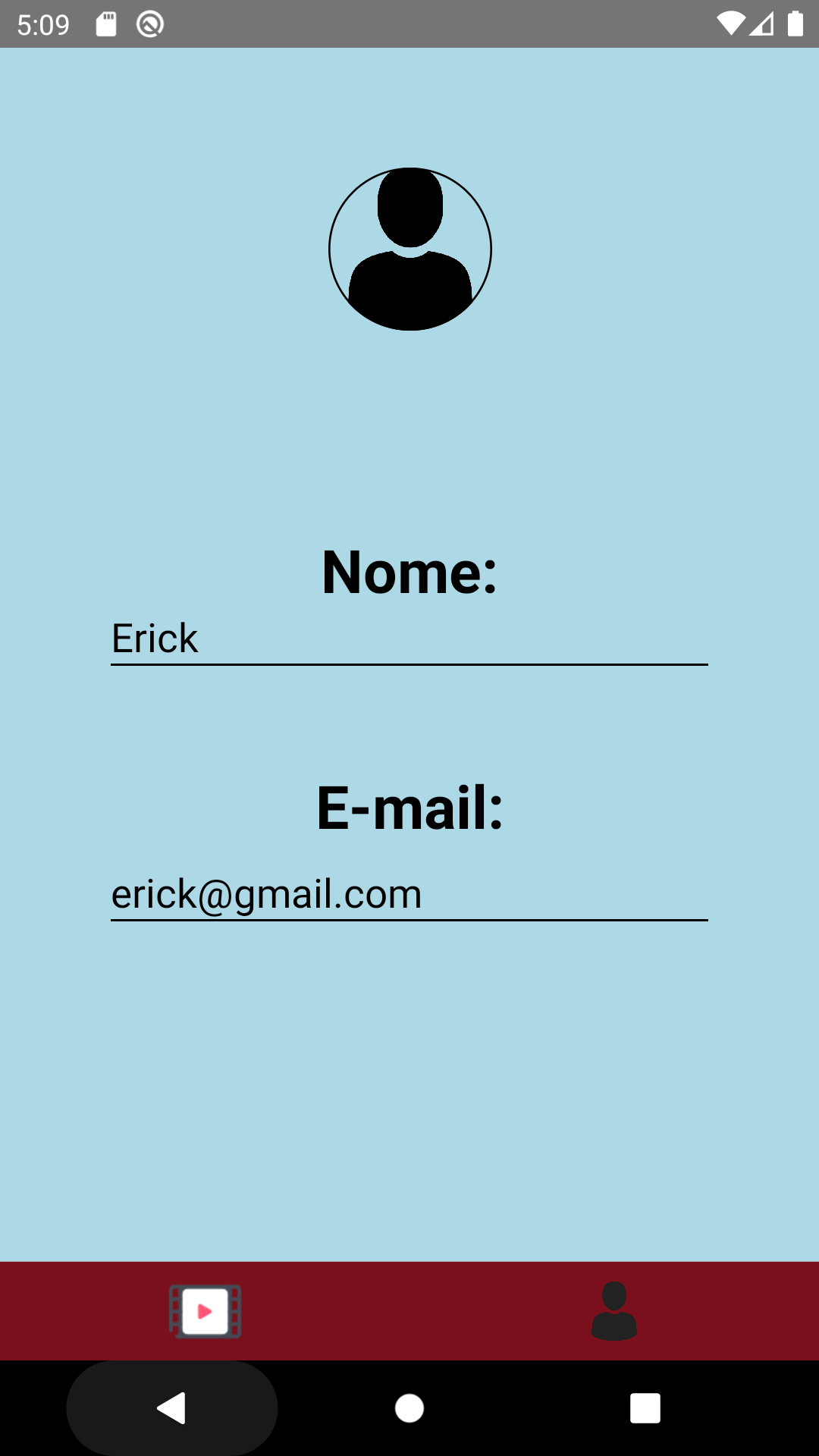
Mobile – React Native (Sprint 4)

1. Página de Login
2. Página de Listagem de Lançamentos





1. Página de Cadastro de Usuário
2. Página de Dados de Usuário (Perfil)

MongoDB – (Sprint 5)

No Mongo tivemos que fazer uma página de localização para os usuários localizarem onde os filmes estão sendo feitos.

