

João Caeiro nº 18825

Patricia Costa nº 18829



Instituto Politécnico de Beja

Escola Superior de Tecnologia e Gestão

<Fevereiro 2020>

João Caeiro nº 18825

Patricia Costa nº 18829

Relatório submetido com requisito parcial

para concluir a cadeira de

Programação de Aplicação para Servidor

Orientado por <Carlos Sancho> <Luís Rosário>





Instituto Politécnico de Beja

Escola Superior de Tecnologia e Gestão

<Fevereiro 2020>

Sumário

Este relatório é um relatório final de um projeto no âmbito da cadeira Programação de Aplicação para Servidor, como objetivo de dar a conhecer o trabalho realizado e as diferentes fazes no seu desenvolvimento.

O projeto é intitulado de "TvChecker", é a criação de uma interface gráfica

com o utilizador, para ajudar a organizar as Séries.

O Relatório é constituído por X partes nos quais se apresentam o trabalho realizado detalhadamente.

Na parte 1 ….

Índice

[Sumário 2](#_Toc31838845)

[Glossário 5](#_Toc31838846)

[Introdução 6](#_Toc31838847)

[8](#_Toc31838848)

[Capítulo 2 9](#_Toc31838849)

[**1. Composer** 10](#_Toc31838850)

[**2. Docs** 10](#_Toc31838851)

[Ecrãs Principais 12](#_Toc31838852)

[Login 12](#_Toc31838853)

[Sign Up 13](#_Toc31838854)

[Home Page 13](#_Toc31838855)

[Favoritos 14](#_Toc31838856)

[Episodios Vistos 14](#_Toc31838857)

[Defenições 15](#_Toc31838858)

[Serie Details 15](#_Toc31838859)

[Model View Model (MVM) 16](#_Toc31838860)

[Implementar um ViewModel 16](#_Toc31838861)

[Ciclo de vida de um ViewModel 16](#_Toc31838862)

[Compartilhar dados entre fragments 17](#_Toc31838863)

[Layout da Base de dados 18](#_Toc31838864)

[Laravel 18](#_Toc31838865)

[Criar controllers 21](#_Toc31838866)

[21](#_Toc31838867)

[Route: 22](#_Toc31838868)

[22](#_Toc31838869)

[Webgrafia 23](#_Toc31838870)

[Conclusão 24](#_Toc31838871)

Índice

[Figura 1-GitHub 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838946)

[Figura 2-Discord 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838947)

[Figura 3-Xampp 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838948)

[Figura 4-Android Studio 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838949)

[Figura 5-php storm 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838950)

[Figura 6-laravel 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838951)

[Figura 7-composer 8](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838952)

[Figura 8-estrutura mvc 9](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838953)

[Figura 9-login 12](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838954)

[Figura 10-sign up 13](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838955)

[Figura 11-home page 13](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838956)

[Figura 12-favorios 14](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838957)

[Figura 13-episodios vistos 14](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838958)

[Figura 14-details 15](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838959)

[Figura 15-Series details 15](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838960)

[Figura 16-lifecycle de um fragment 16](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838961)

[Figura 17-esquema da base de dados 18](#_Toc31838962)

[Figura 18-exemplo de um model 19](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838963)

[Figura 19-exemplo de migration 19](#_Toc31838964)

[Figura 20-exemplo de seeder 20](#_Toc31838965)

[Figura 21-exemplo de tabela 20](#_Toc31838966)

[Figura 22-exemplo de controler 21](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838967)

[Figura 23-route 22](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838968)

[Figura 24-route 22](file:///C:\Users\nagat\Downloads\relatorioaj.asd.docx#_Toc31838969)

Glossário:

**GitHub -** É um sistema de controle de versões, que permite que os usuários acompanhem mudanças no código-fonte durante o desenvolvimento de um ***software***. É possível fazer ***Commits***, ***Branches*, *Pull* *Request*** e ***Resume***;

**Discord-**aplicação de chamadas de vídeo e partilha de ecrã;

**Xampp-**  Programa com os principais servidores de código aberto que incluem o FTP, base de dados MySQL e Apache com suporta nas linguagens PHP e Perl.

**Android Studio-** É um ambiente de desenvolvimento integrado, para o desenvolvimento de apps para android.

**Jetbrain PhpStorm-** editor de código PHP, que mantém-se com as tendências mais recentes tanto do formato PHP, quanto das demais linguagens web, integrando uma variedade de ferramentas modernas, e traz ainda mais extensibilidade com suporte aos principais frameworks PHP.

**Framework-** é um template com diversas funções que podem ser usadas pelo desenvolvedor.

**Php-U**ma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML.

**Web-**significa um sistema de informações ligadas através de hipermédia (hiperligações em forma de texto, vídeo, som e outras animações digitais) que permitem ao utilizador aceder uma infinidade de conteúdos através da internet.

**HTML-** A linguagem HTML é usada para criar documentos eletrônicos, chamados de páginas web, exibidos na World Wide Web.

**JSON-**JSON é um formado de representação de dados baseado na linguagem de programação [Javascript](https://dicasdejavascript.com.br/), daí o nome JavaScript Object Notation. Ou seja, Notação de Objeto em Javascript.

**XML-**é a sigla para **Extensible Markup Language**, que significa em português Linguagem Extensível de Marcação Genérica. É uma recomendação para gerar linguagens de marcação para necessidades especiais. XML é capaz de descrever diversos tipos de dados, e seu objetivo principal é a facilidade de compartilhamento de informações através da Internet.

Introdução

Este trabalho surge no âmbito da unidade curricular – Programação de Aplicação para Servidor, para a conclusão da cadeira de época de finalistas.

O interesse por este tema de trabalho deveu se, principalmente, ao á vontade que temos com as Séries, por estas fazerem parte do nosso dia-a-dia. Como alguma parte do nosso tempo é gasto a ver séries é de imaginar a quantidade de séries existentes e a imensidão de episódios e temporadas vistas.

Na primeira parte do relatório surge a explicação da ideia da **app** e qual a utilidade desta, como foram organizados os grupos e as ferramentas utilizadas.

Na segunda parte será explicado como esta vai ser desenvolvida, qual a linguagem de programação e os métodos utilizados nesta e como funcionam.

Também será explicado como funciona o armazenamento de dados utilizado e como este funciona

Na terceira parte será demonstrado o código utilizado para o funcionamento da app e do armazenamento de dados desta.

Para finalizar será apresentada uma reflexão final sobre o percurso realizado ao longo desta cadeira, como as aprendizagens realizadas ao longo de todo este processo.

A ideia da **app** surgiu ao perceber há uma falta de organização por nossa parte ao assistir as séries. Por vezes deixamos os episódios a meio e passados dias, semanas, meses voltamos a ver a serie, mas não é possível recordar em que minuto ou episódio se ficou. A nossa **app** tem um design simples e limpo, onde o utilizador consegue ter uma fácil utilização da app e atingir os objetivos pretendidos.

**Na** **app** TvChecker é possível marcar os episódios e as temporadas vistas, ou até marcar os minutos em que se ficou no meio do episódio, é possível marcar a serie como favorita e ter um mais fácil acesso a esta. É possível marcar as séries como vistas e aparecer as séries que já viu.

Capitulo 1

Parte I - Ideia

A ideia da **app** surgiu ao perceber há uma falta de organização por nossa parte ao assistir as séries. Por vezes deixamos os episódios a meio e passados dias, semanas, meses voltamos a ver a serie, mas não é possível recordar em que minuto ou episódio se ficou. A nossa **app** tem um design simples e limpo, onde o utilizador consegue ter uma fácil utilização da app e atingir os objetivos pretendidos.

Na **app** TvChecker é possível marcar os episódios e as temporadas vistas, ou até marcar os minutos em que se ficou no meio do episódio, é possível marcar a serie como favorita e ter um mais fácil acesso a esta. É possível marcar as séries como vistas e aparecer as séries que já viu.

Parte II – Organização do Trabalho

Inicialmente foi debatido entre a turma a organização dos grupos, chegando a um consenso de dois grupos de quatro elementos.

Seguidamente foi escolhido o tema do trabalho, obtendo os dois grupos por continuar os trabalhos já antes criados na cadeira do ano anterior.

Uma vez escolhida a ideia, foi utilizado o programa **GitHub,** onde foi criado um repositório com todos os membros do grupo, para que todos tivessem acesso a qualquer alteração ou implementação feita. Nesta aplicação também foi possível determinarmos as tarefas e deveres de cada membro.

Para trabalhar fora da Universidade utilizamos o programa **Discord** sendo assim possível discutir sobre o trabalho e tirar algumas dúvidas os entre membros.

A meio da criação da nossa **app** um dos membros desistiu do curso acabando por ficar no total o grupo com três elementos. Isto dificultou o nosso trabalho, mas conseguimos dividir as tarefas por os três membros não afetando o nosso trabalho.

Já perto do final outro membro desistiu do nosso grupo ficando no total dois membros.

Parte III – Ferramentas Utilizadas

* GitHub;
* Discord;
* Xampp(My SQL);
* Android studio;
* Jetbrains phpstorm;
* Composer;
* Laravel.

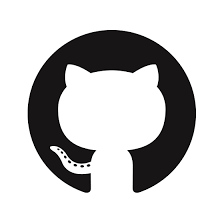


Figura 1-GitHub



Figura 2-Discord



Figura 3-Xampp



Figura 4-Android Studio

Figura 4-php storm

Figura 5-php storm

Figura 5-php storm





Figura 6-laravel

Figura 7-composer

Capítulo 2

Laravel

**Laravel** é uma **framework** de desenvolvimento rápido para **PHP,** livre e de código aberto. O principal objetivo desta **framework** é permitir ao utilizador trabalhar de uma forma rápida e estorturada.

O Laravel foi criado por Taylor Otwell em 2011, este tem continuado a evoluir a framework de forma a faze-la compatível com as recentes tecnologias Web.

Laravel utiliza o modelo de estrutura MVC (Model View Controller) que funciona da seguinte forma:

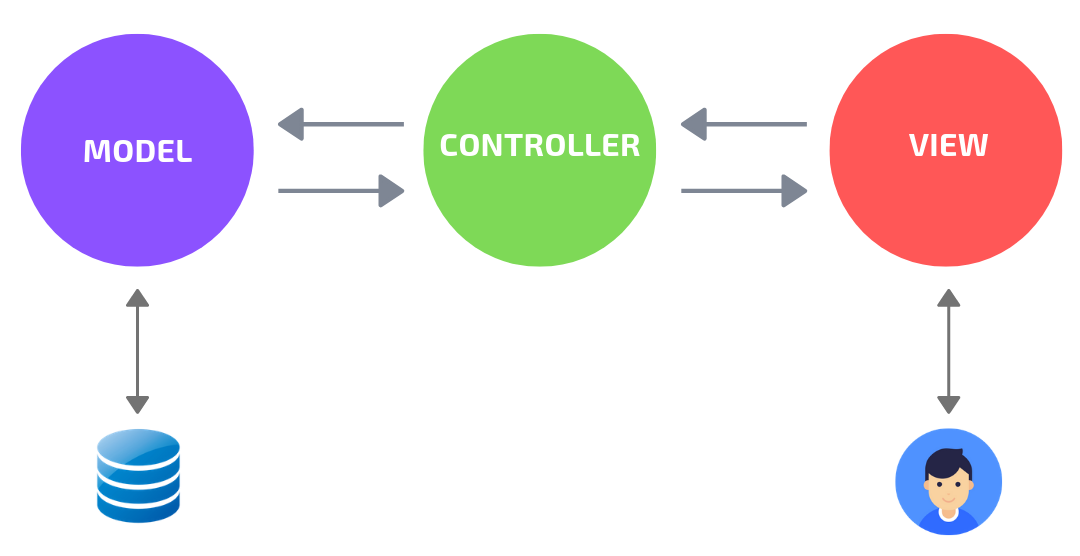
* O Model é a camada responsável pela parte lógica da aplicação, ou seja, todos os recursos da aplicação (consultas á Base de Dados, validações, notificações, etc.).
* A View é a camada responsável por mostrar os dados ao utilizador, em páginas HTML, JSON, XML, ect.
* O Controller é a camada que sabe o que chamar e quando chamar para executar determinada ação.

Figura 8-estrutura mvc

Resumindo ao receber uma requisição, o Controller solicita ao Model as informações necessárias (vindas da Base de dados), obtém e retorna ao Controller. Com estas informações o Controller envia as para a View que irá mostrá-las.

**Principais recursos do framework Laravel**

A estrutura do Laravel possui uma grande variedade de recursos como Composer, Docs, Routes, Blade, Elequent e QueryBuilder, Artisan Console.

# **1. Composer**

O Laravel utiliza o Composer para gerir as suas dependências. O composer é uma ferramenta que gerir facilmente, os pacotes de terceiros na aplicação.

Para fazer a instalação, basta executar o comando na linha de comandos:

composer create-project laravel/laravel

# **2. Docs**

No Docs é possível qualquer pessoa aceder e aprofundar os seus conhecimentos sobre Laravel e ate mesmo tirar duvidas.

<http://laravel.com/docs>

**3. Routes**

As routes são responsáveis pelo mapeamento da URL digitada no navegador e fazer alguma ação nele.

Route::get('hello', function () {  
return 'Hello World';  
});

**4. Blade-Templates**

O Blade é o compilador de templates do Laravel (template engine), o Blade consegue reduzir a quantidade de código PHP inserido no meio do HTML. O blade disponibiliza diretivas que são inseridas junto ao código HTML de acordo com a necessidade da página.

**5. Elequent**

O Eloquent é o ORM padrão do Laravel, este aplica o Design Patter ActiveRecord onde cada tabela da base de dados é mencionada no código, através de uma classe "Model" que é utlizada para interagir com essa tabela. Os Models permitem a consulta de dados nas tabelas bem como operações de insert, update e delete.

**6. QueryBuilder**

O QueryBuilder é um construtor de querys, e pode ser utilizado na maioria das operações da base de dados

**7. Artisan Console**

O ArtizanConsole é uma interface de linha de comandos que fornece vários comandos com o objetivo de facilitar o desenvolvimento da aplicação.

Capitulo 3

Ecrãs Principais

## Login

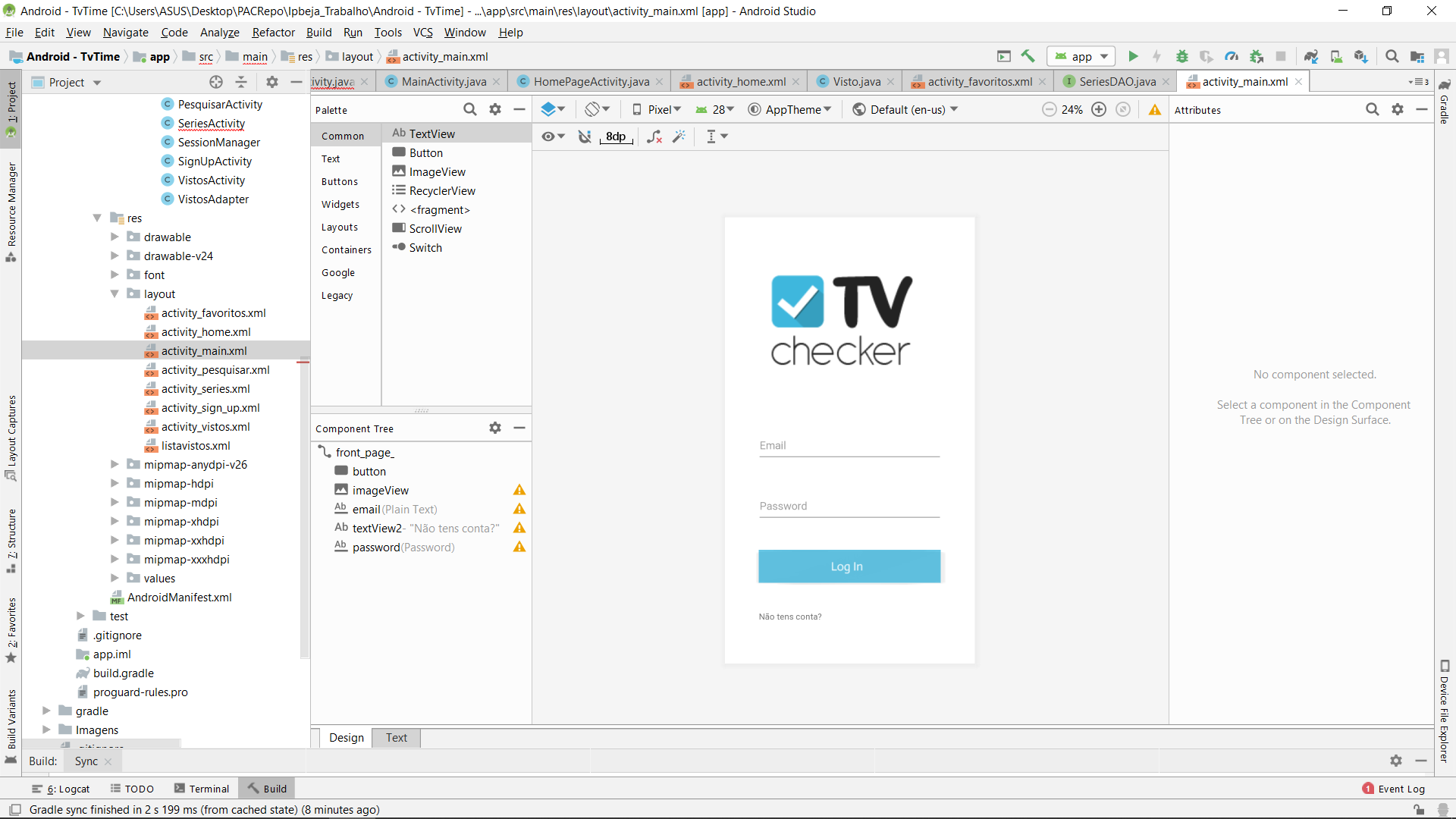


Figura 9-login

* No ecrã da log in o utilizador escreve o email e a password com que criou a conta e ao clicar no botão de log in vai fazer o log in.
* Se os campos do email e da password estiverem vazios o programa faz um aviso para preencher os campos todos.
* se o utilizador se esquecer de preencher tanto o campo do email ou da password, a aplicação manda uma aviso para preencher o respetivo campo.

## Sign Up

Figura 10-sign up

* No ecrã "Sign Up" o utilizador para criar uma conta escreve nos campos designados e carrega no botão "Sign Up".
* Caso qualquer um dos campos não esteja preenchido, o utilizador receberá uma mensagem de aviso com o respetivo campo não preenchido.

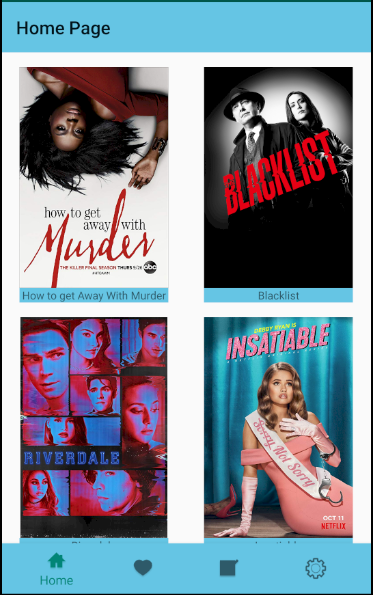
Home Page

Figura 11-home page

* Mostra todas as series inseridas na base de dados com os respetivos nomes.
* Temos a Barra do menu para podermos navegar pelos diferentes Fragments.

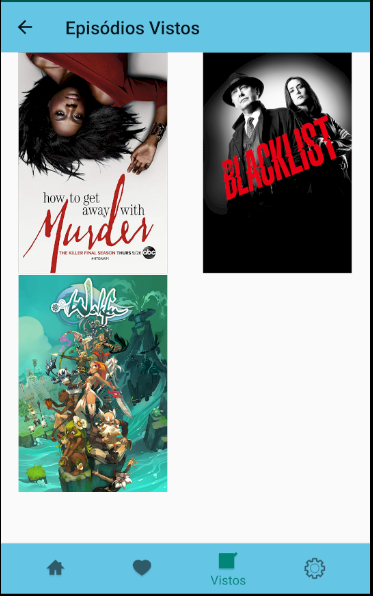
## Favoritos



* Mostra todas as series que o user adicionou aos favoritos que estão guardadas na base de dados.

Figura 12-favorios

## Episodios Vistos



* Mostra todas as series marcadas como vistas ou que o user esta a ver atualmente.

Figura 13-episodios vistos

## Defenições

* Nas definições da para fazer logout

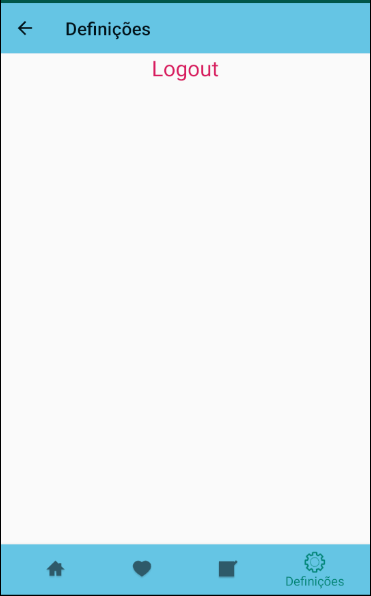


Figura 14-details

Serie Details

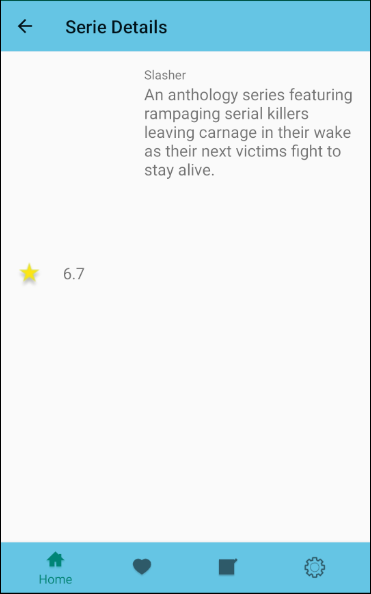


Figura 15-Series details

* Mostra os detalhes sobre as series como a pontuação, o nome, a sinopse, imagem, e os episodios e temporadas.

Model View Model (MVM)

A classe ViewModel no Android Studio foi feita para armazenar e ferir dados relacionado à UI considerando o ciclo de vida, permite tambem que os dados sobrevivam a mudanças de configurção, como a rotação de ecrã.

## Implementar um ViewModel

Durante uma mudança de configuração os objetivos do ViewModel são guardados automaticamente para serem disponibilizados para a proxima activity ou instancia do fragment.

Se uma activity for recriada a mesma ira receber uma Instace MyViewModel que foi criada pela primeria activity, quando a activity estiver concluída o framework usara o onCleared() do ViewModel para limpar os recursos.

## Ciclo de vida de um ViewModel

Objetos ViewModel têm o Lifecycle transferido para o ViewModelProvider ao receber o ViewModel, este permanece na memoria ate o Lifecycle desaparecer premanentemente, no caso de uma activity ate ela desaparecer premanentementee de um fragmente quando ele e desanexado.

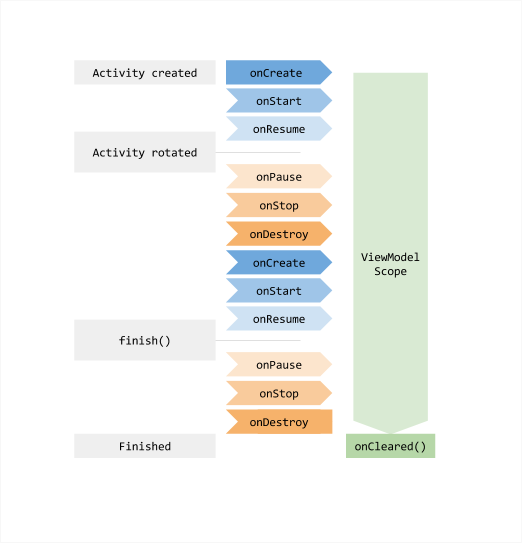
Esta imagem mostra os estado de uma activity que se aplicam a ao Lifecycle de um fragment.

Figura 16-lifecycle de um fragment

Um ViewModel é chamado desde que o sistema chama o método pode ser chamado várias vezes durante a vida, mas o ViewModel existe a partir do momento que é solicitado pela primeira vez ate a activity seja finalizada ou destruída.

## Compartilhar dados entre fragments

Quando dois fragments numa activity precisam partilhar dados entre si, como por exemplo uma list e detalhes, em que um fragment mostra uma lista e um e o outro mostra os detalhes de um item slecionado da lista. Neste tipo de casos a activity precisa unir os dois fragments e os dois fragments precisam lidar com o cenario em que o outro fragment ainda não foi criado ou esta invisivel, isto pode ser resolvido usando um ViewModel partilhado entre os fragments.

Os dois fragments recebem a activity que os contém, assim quando os fragments recebem o ViewModelProvider, vão receber a mesma Instance SharedViewModel que esta na activity.

Assim conseguuimos que a activity não precise fazer nada nem saber essa comunicação, os fragments não precisam saber uns dos outros so do SharedViewModel e se um dos fragments desaparecer o outro continua a funcionar normalmente, assim cada fragment tem o seu proprio Lifecycle e não é afectado pelos Lifecycles dos outros e se um fragment substituir outro a UI continua a funcionar normalmente.

Layout da Base de dados

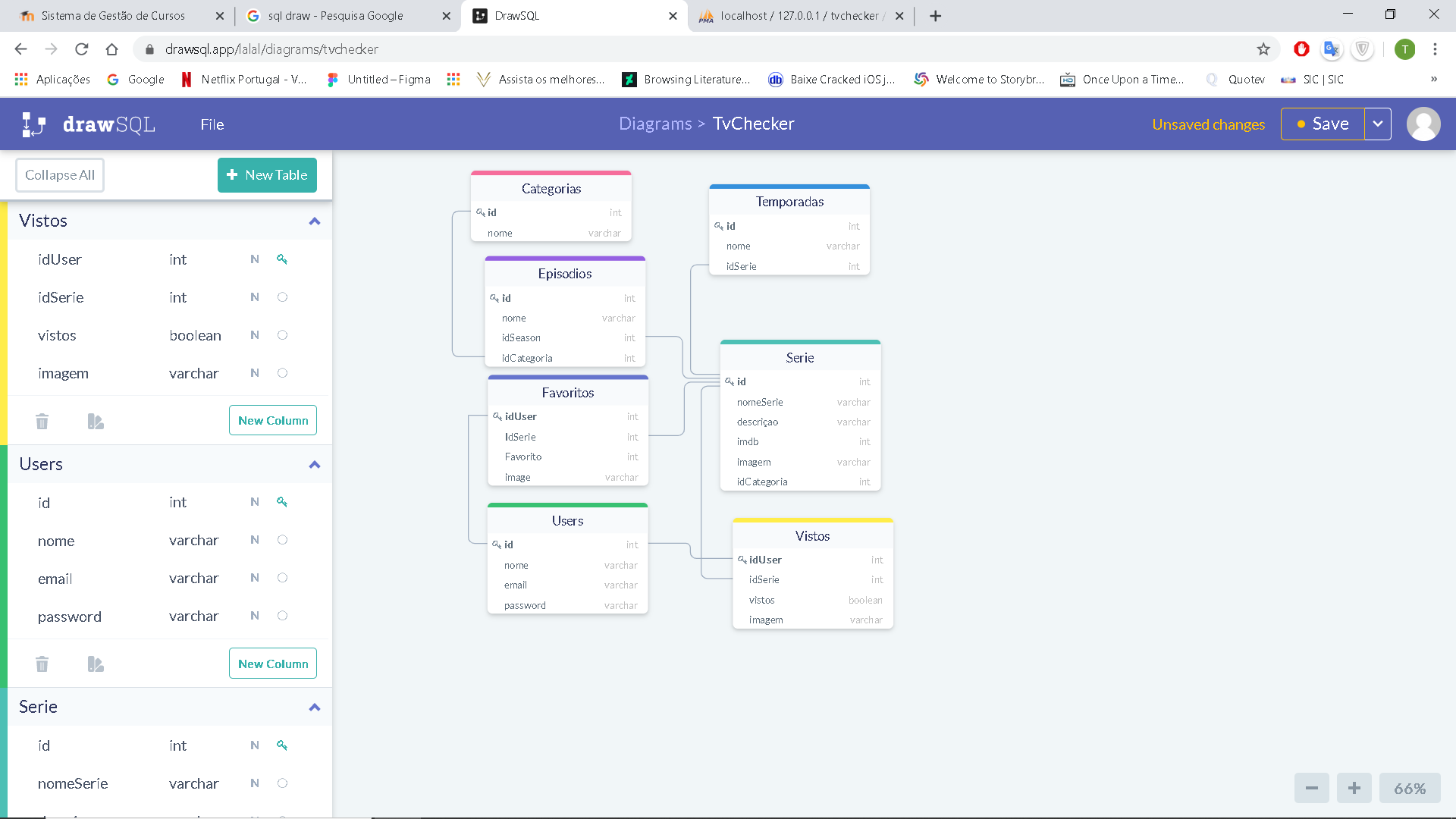


Figura 17-esquema da base de dados

A base de dados é constituída por 7 tabelas(Categorias, Episódios, Favoritos, Users, Temporadas, Serie e Vistos ).

## Laravel

Para criar uma API em Laravel foi necessário utilizar a linha de comandos.

**Criar site laravel:**

-Laravel new Tvchecker

**Criar autenticação:**

-composer require laravel/ui

**Usando bootsrap:**

-php artisan ui --auth bootstrap

**Criar models e Migrations:**

-Php artisan make:model (nome tabela) -m

Os models criados foram das tabelas acima referidas.

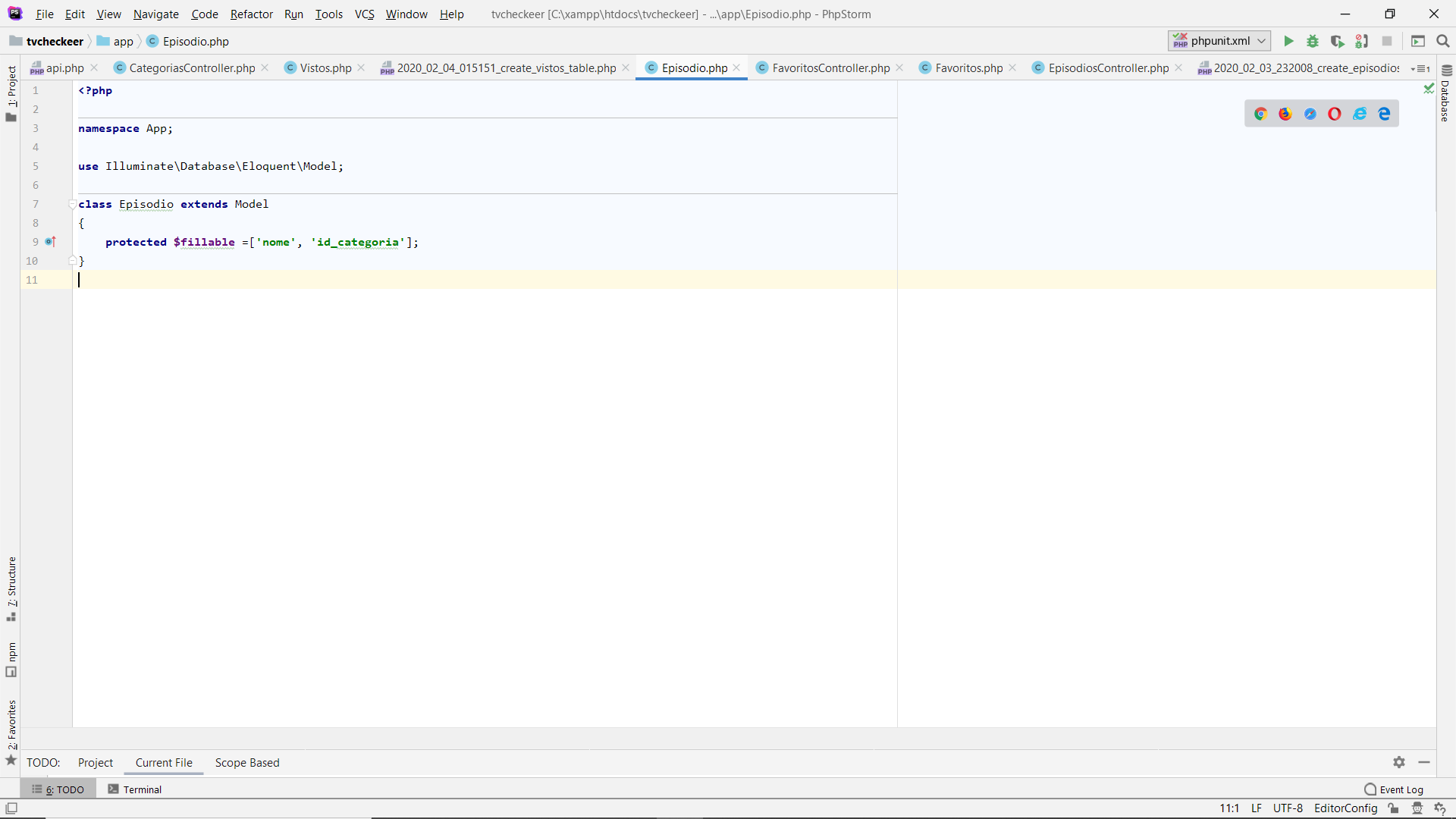


Figura 18-exemplo de um model

Fazer as migrations das tabelas criadas utiliza se na linha de comandos a linha de código:

-php artisan migration

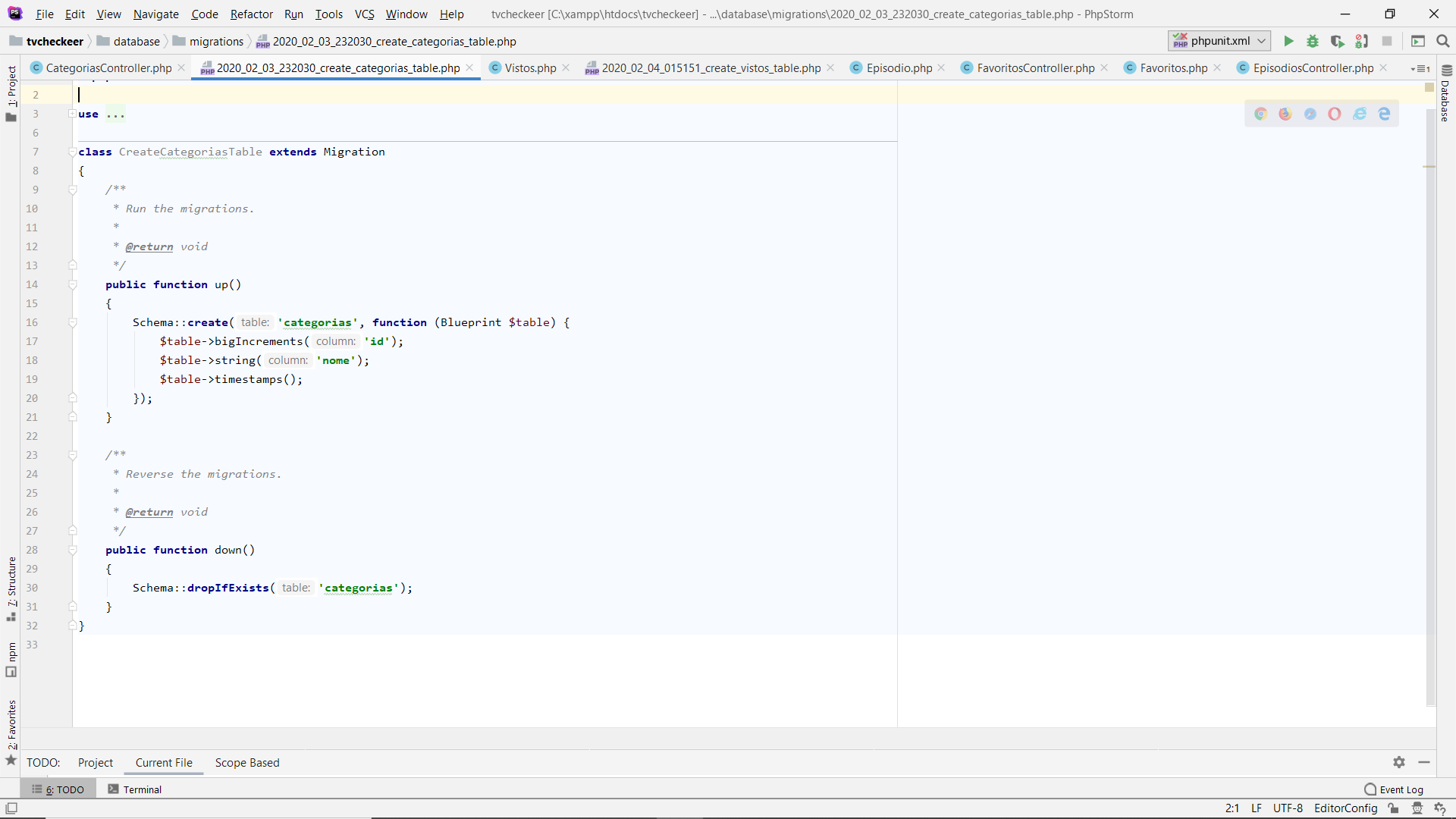


Figura 19-exemplo de migration

Mais uma vez as migrations criadas são de todas as tabelas acima.

Para não ter que se criar os User "á mão" utilizamos o Faker uma biblioteca que gera data para a nossa base de dados, utilizando o processo de seeding.

Na linha de comandos:

-php artisan make:seeder UsersTableSeeder

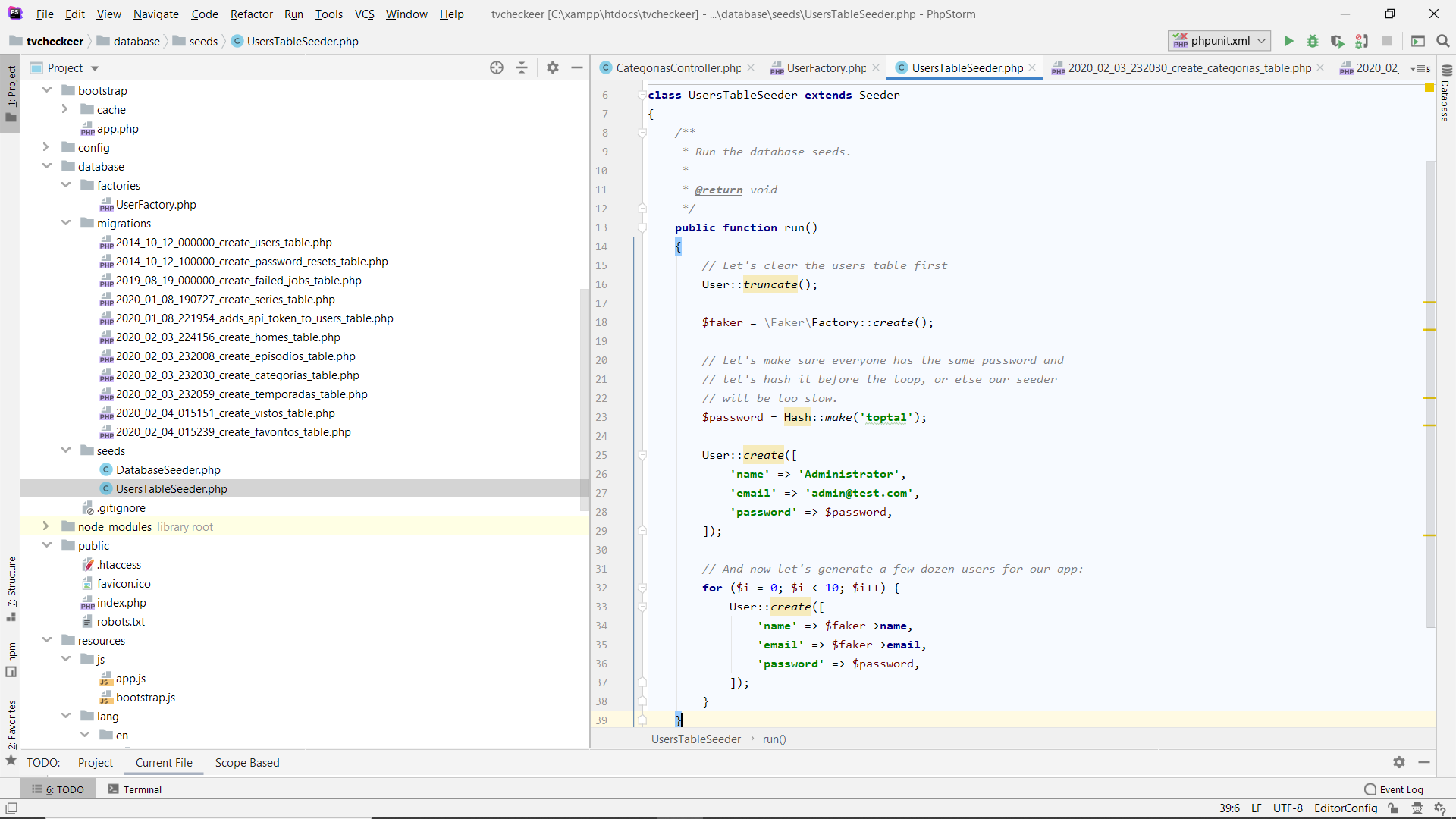


Figura 20-exemplo de seeder

Depois disso utilizamos o comando para fazer a seed da tabela Users:

-php artisan db:seed –class=UserTableSeeder

E assim ficou a nossa tabela de Users:

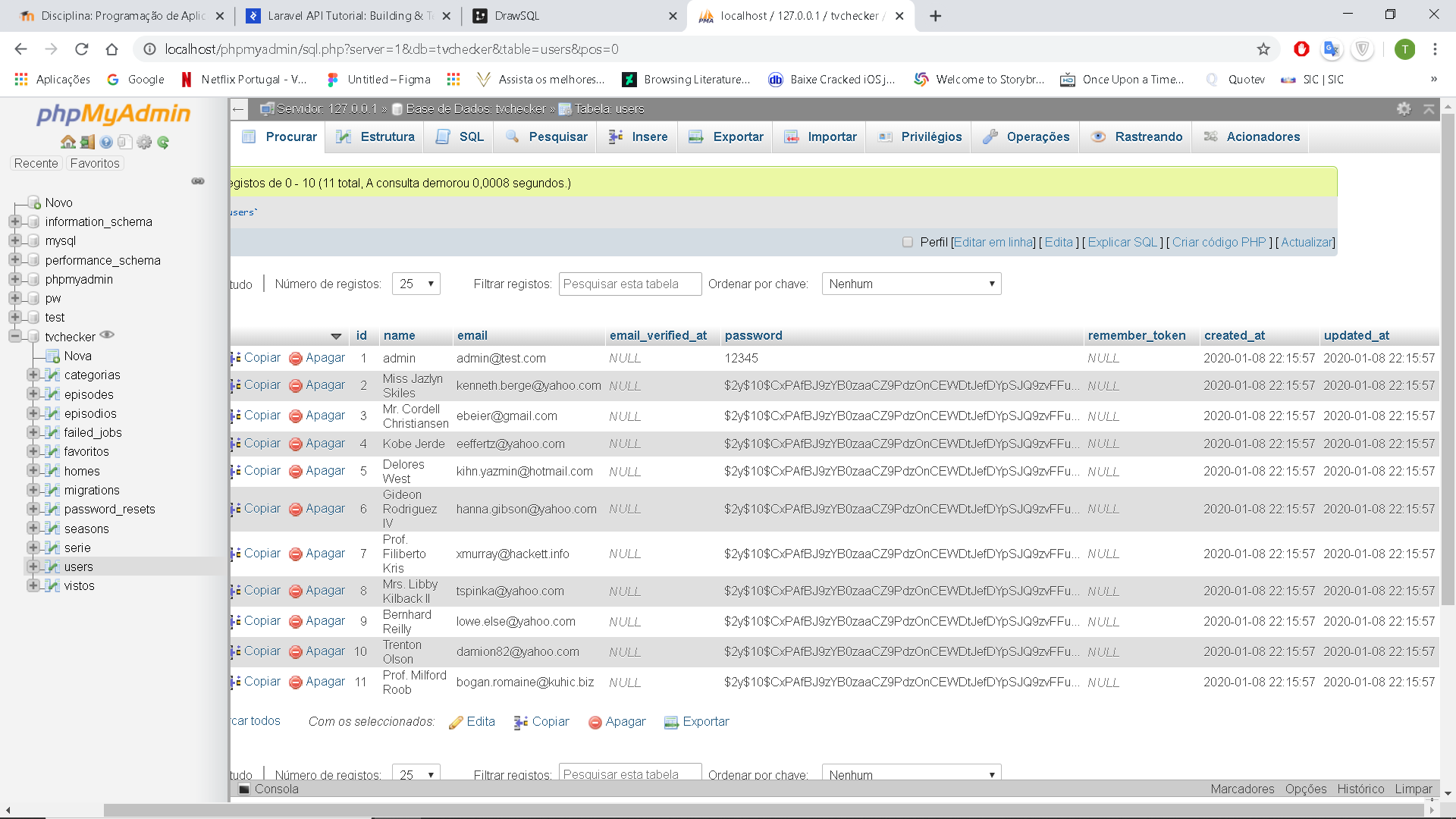


Figura 21-exemplo de tabela

## Criar controllers

Para criar os Controllers utilizou se o seguinte código na linha de comandos:

php artisan make:controller (nome tabela)Controller

## 

Figura 22-exemplo de controler

## Route:

## 

Figura 23-route

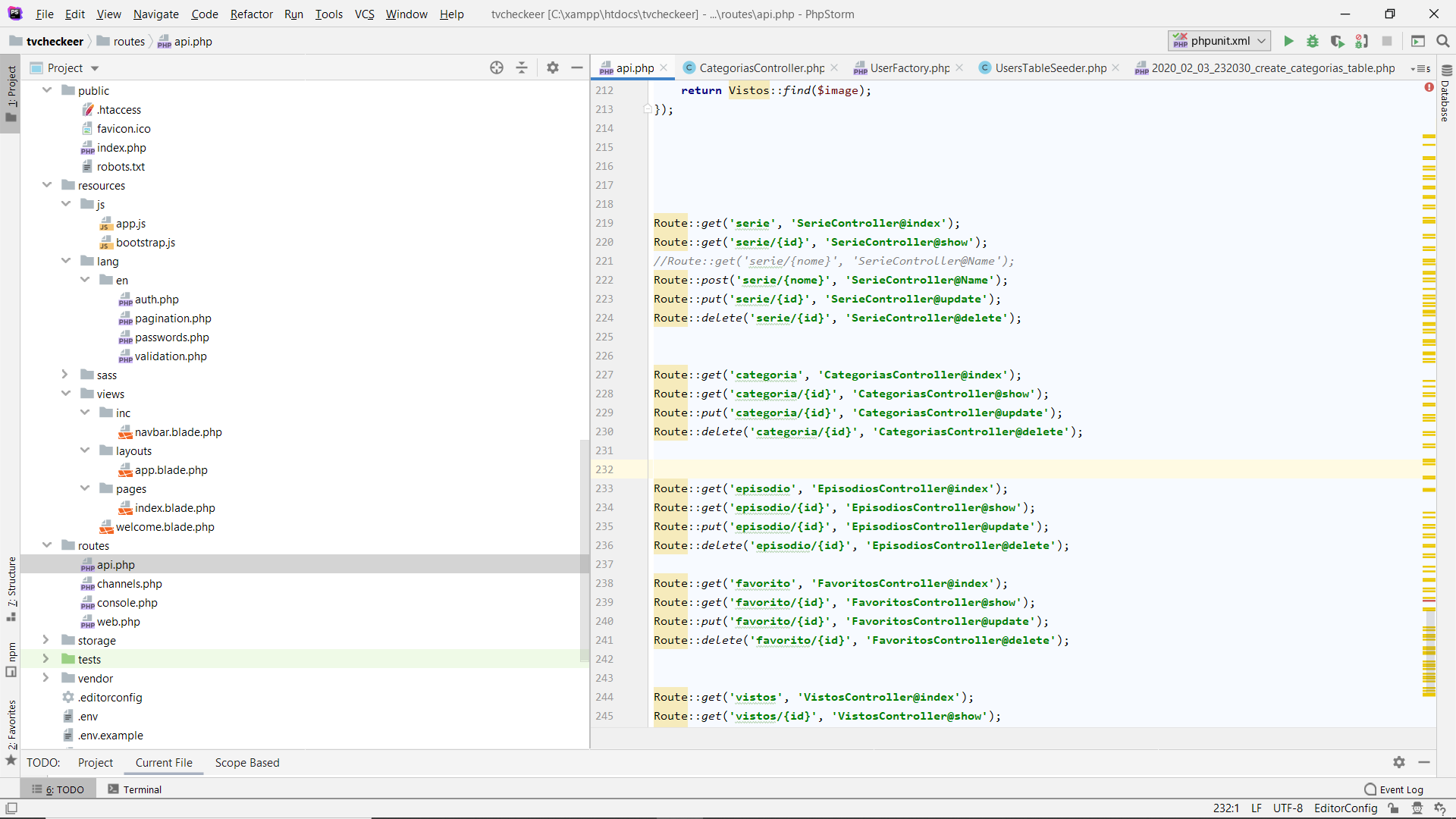


Figura 24-route

Conclusão

Este trabalho foi um trabalho demorado, e foi necessária muita dedicação.

Para este trabalho foram utilizadas as melhores imagens, e ícones para se colocar na aplicação do TvChecker.

Esta reflexão permitiu concluir que é muito simples fazer uma Aplicação android, apenas é necessária alguma motivação e apoio.

Uma das principais vantagens da elaboração deste trabalho, reside no facto de nos proporcionar um melhor conhecimento sobre as linguagens de programação e dos seus conteúdos.

No Geral, foi um trabalho interessante e definitivamente instrutivo.

Webgrafia

Android Studio Developers “Visão geral do ViewModel”

Disponivel:

<https://developer.android.com/topic/libraries/architecture/viewmodel?hl=pt_br>

Visitado em: 5 de Fevereiro de 2020.

Medium “O que é Laravel? Porque usá-lo?” Feito por: João Roberto.

Disponivel:

<https://medium.com/joaorobertopb/o-que-%C3%A9-laravel-porque-us%C3%A1-lo-955c95d2453d>

Visitado em: 4 de Fevereiro de 2020

Treinaweb “O que é Laravel?” Feito por: Ana Paula de Andrade.

Disponivel:

<https://www.treinaweb.com.br/blog/o-que-e-laravel/>

Visitado em: 5 de Fevereiro de 2020

Larashout “What is Laravel and Why You Should Learn it?”

Disponivel:

<https://www.larashout.com/what-is-laravel-and-why-you-should-learn-it>

Visitado em: 5 de Fevereiro de 2020