UNIVERSIDADE DE CABO VERDE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

Yetu Yaliy-Plataforma *OTT* Implementação *Web e Android TV*

Erickson de Carvalho Vaz

Orientador | Eloy Tavares Mendes, M.Sc.

Palmarejo-Praia, Outubro de 2021

UNIVERSIDADE DE CABO VERDE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

Yetu Yaliy-Plataforma *OTT* Implementação *Web e Android TV*

Erickson de Carvalho Vaz

Orientador | Eloy Tavares Mendes, M.Sc.

Palmarejo-Praia, Outubro de 2021

O corpo jurado abaixo assinado, aprova o trabalho final de curso, ***YetuYaliy*-Plataforma *OTT* Implementação *Web e Android TV***, elaborado por **Erickson de Carvalho Vaz**, como requisito parcial à obtenção do título de Licenciatura em Engenharia Informática e de Computadores.

Presidente: _____

Arguente: _____

Orientador: _____

Cidade da Praia, _____de _____2021

Dedico este trabalho aos meus pais que ao longo da minha caminhada acadêmica, acreditaram sempre em mim, apoiaram-me e deram-me a oportunidade de concretizar mais um sonho.

Um obrigado muito especial

RESUMO

Plataformas *OTT* (*Over-The-Top*), são aplicações que disponibilizam conteúdos de vídeo através da *internet*, ao invés da tradicional transmissão a cabo ou por satélite, desta forma é possível transmitir instantaneamente vídeos para dispositivos móveis (*smartphones* e *tablets*), *web* e televisão, hoje o mercado das *OTTs* no mundo é bastante concorrido, porém no nosso país em particular e no continente africano em geral, ainda é algo bastante discreto e em alguns casos inexistente. Como forma de colmatar esta deficiência e disponibilizar ao público um meio de divulgação e acesso aos conteúdos *"made in africa"* foi pensada e desenvolvida a plataforma *YetuYaliy*, composta por dois módulos, *web* e *android tv*. Neste trabalho serão mostradas todas as etapas que levaram a conceção dos mesmos, utilizando tecnologias como *PHP*, *Java*, *JavaScript*, *Android* e *MySQL* por forma a satisfazer as necessidades do público, contudo, devido a dimensão do tema e complexidade do sistema tornou-se impossível disponibilizar algo completo logo no início, principalmente numa área de negócio muito pouco explorada, por isso, o resultado final apresentado aqui será um *MVP* (Mínimo Produto Viável) que tem por objetivo testar a ideia, obter *feedback* rápido e melhora-la em versões futuras.

Palavras-chave: plataforma *OTT*, conteúdos *pay-per-view*, distribuição de conteúdos multimédia, vídeos sob demanda, *streaming,* televisão digital.

ABSTRACT

OTT (Over-The-Top) platforms are applications that provide video content over the internet, instead of the traditional cable or satellite transmission, this way it is possible to instantly transmit videos to mobile devices (smartphones and tablets), web and television, today the OTT market in the world is very competitive, but in our country in particular and in the African continent in general, it is still quite discreet and in some cases non-existent. As a way to address this deficiency and provide the public with a means of dissemination and access to "made in Africa" content, the YetuYaliy platform was designed and developed, consisting of two modules, web and android tv. This work will show all the steps that led to their design, using technologies such as PHP, Java, JavaScript, Android and MySQL in order to meet the needs of the public, however, due to the size of the theme and complexity of the system it became impossible provide something complete right from the start, especially in a very little explored business area, so the final result presented here will be an MVP (Minimum Viable Product) that aims to test the idea, get quick feedback and improve it in future versions .

Keywords: OTT platform, pay-per-view content, multimedia content distribution, on-demand videos, streaming, digital television.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS, ABREVIATURAS E SIGLAS

ALPN	Negociação de Protocolo da Camada de Aplicação
AOTT	*Over The Top* com Publicidades
API	Interface de Programação de Aplicação
APK	*Android Package*
CAPTCH	*Completely Automated Public Turing test to tell Computers and Humans Apart*
CDN	Rede de Distribuição de Conteúdos
CPU	Unidade Central de processamento
CVV	Código de Verificação do Cartão
DSD	*Direct Stream Digital*
DVB	*Digital Video Broadcasting*
DVBC	*Digital Video Broadcasting* – por Cabo
DVBS	*Digital Video Broadcasting* – por Satélite
DVBT	*Digital Video Broadcasting* — Terrestre
ECMA	Associação Europeia de Fabricantes de Computadores
ELEC	Centro de entretenimento integrado do *Linux*
eMMC	Multimédia-*Card* integrado
FPS	*Frames* por Segundos
GPIO	Pinos de Entrada e Saída de Propósito Geral
GPL	Licença Pública Geral
GPU	Unidade de processamento Gráfico
HDMI	Interface Multimédia de Alta Definição
HDR	Alto Alcance Dinâmico
HLG	Registro de Gama Hibrido

HTML	Linguagem de Marcação de Hipertexto
HTTP	Protocolo de transferência de Hipertexto
HTTPS	Protocolo de Transferência de Hipertexto Seguro
IDE	Ambiente de Desenvolvimento Integrado
IEEE	Instituto de Engenheiros Eletricistas e Eletrónicos
IPTV	Transmissão Televisiva por redes *IP*
J2EE	*Java Platform, Enterprise Edition*
J2ME	*Java Platform, Micro Edition*
J2SE	*Java Platform, Standard Edition*
JSON	*JavaScript Object Notation*
MCU	Microcontroladores
MIMO	Múltiplas Entradas, Múltiplas Saídas
MPI	*Merchant Plug-In*
MVC	Modelo, Visão, Controlador
MVP	Modelo, Visão, Apresentador
OTT	*Over The Top*
PHP	*Personal Home Page: Hypertext Preprocessor*
POO	Programação Orientada a Objetos
POS	Ponto de Venda
REST	Estado de Transferência Representativa
RFATV	Requisito Funcional *Android TV*
RFPC	Requisito Funcional Painel de Controle
RFSP	Requisito Funcional Site Principal
RN	Requisitos de Negócio

RNF	Requisitos Não Funcionais
RSDB	*Real Simultaneous Dual Band*
SBC	Computador de Placa Única
SGBD	Sistema de Gerenciamento de Base de dados
SGBDR	Sistema de Gerenciamento de Base de Dados Relacional
SISP	Sociedade Interbancária e Sistemas De Pagamento
SO	Sistema Operativo
SQL	Linguagem de Consulta Estruturada
SVOD	Subscrição em Vídeos sob Demanda
TLS	Segurança de Camada de Transporte
TVOD	Vídeos sob Demanda Transacional
UHD	Ultra Alta Definição
UML	Linguagem de Modelação Unificada
URL	Localizador Uniforme de Recursos
USB-C	Porta Serial Universal do Tipo C
VOD	Vídeos sob Demanda
VPU	Unidade de processamento de Visão
WAV	*Waveform Audio File Format*
WOL	*Wake-on-LAN*

ÍNDICE

1 INTRODUÇÃO

Longe vão-se os dias em que as empresas de comunicação social tradicional tinham total controle sobre o que os espectadores podiam assistir. Antigamente, se quiséssemos ouvir uma música específica ou até mesmo assistir a um filme estávamos 100% dependentes destes meios de comunicação. Contudo, há já vários anos ocorreu uma revolução na forma como consumimos conteúdos audiovisuais, e a sigla *OTT* do inglês *Over-The-Top* está envolvida nesta mudança radical.

De uma forma resumida e assim como descrito na Amazon Advertising em [1], o termo *OTT* diz respeito a plataformas de distribuição de conteúdos pela *internet*, no qual o utilizador assiste sob demanda, ou seja quando bem entender, aquilo que deseja. Essa conexão é feita diretamente entre a plataforma e o utilizador final, sem o intermédio de outras empresas de teletransmissão, alguns exemplos de plataformas *OTT* são a já muito falada *Netflix*, a *HBO Go* ou a *Amazon Prime*.

Atualmente grandes plataformas de distribuição de vídeos, como é o caso do *YouTube*, mudaram a forma de produzir e consumir conteúdos, especialmente em vídeo, agora para além de consumirmos conteúdos, todos podemos ser produtores e distribuidores ativos.

Essa reviravolta no mundo das *OTTs*, deu-se principalmente pela chegada da tecnologia de *streaming*, trazendo a possibilidade de assistir vídeos sem a necessidade de fazer o *download* dos arquivos. Foi assim que a distribuição via *internet* no formato *OTT* começou, com a possibilidade de ver quantos vídeos quisermos, em qualquer lugar do mundo e com qualquer dispositivo.

Enquanto a *internet* e as novas tecnologias digitais estão remodelando o mercado de produção e distribuição de média em todo o mundo, o continente africano em geral e o nosso país em particular, apesar de serem ricos em produções de conteúdos relacionados a cultura, música, natureza, desporto, eventos e notícias, ainda enfrentamos desafios ligados a rentabilização destes tipos de produtos e a facilitação de acesso por quem queira entrar neste mercado como produtor ou distribuidor, mas também de quem procura e deseja consumir, e isto deve-se em grande parte pela inexistência de plataformas dedicadas no país e a sua escassez no continente.

É neste contexto que surgiu a ideia deste projeto, que inicialmente recebeu o nome de *YetuYaliy* (termo que nasceu da junção das palavras em Suaíli, um idioma africano,” *Yetu*” = nosso + “*Yaliy*”=conteúdo). Esta plataforma permitirá que qualquer pessoa ou empresa de produção de conteúdos possa criar distribuidoras virtuais, colocar, promover e vende-los. O

YetuYaliy é especializado em conteúdos africanos, mas também está virado para o mercado global.

Sendo assim, este relatório tem como objetivo detalhar todo o processo de especificação e implementação *web* e *Android TV* desta plataforma, de ressaltar ainda que, o que será mostrado aqui é apenas uma parte inicial da implementação de um projeto muito maior que será aperfeiçoado a nível governamental e *interface* de utilizador.

1.1 Objetivos Gerais

Desenvolver os módulos *web* e *Android TV* da plataforma *OTT YetuYaliy* que permite a gestão e a distribuição de conteúdos audiovisuais.

1.2 Objetivos Específicos

Os objetivos específicos para este trabalho são:

- Estudar e entender os conceitos por detrás das *OTTs*
- Estudar os diferentes formatos de vídeos implementáveis numa *OTT*
- Estudar e entender os diferentes modelos de negócios associados as *OTTs*
- Analisar e estudar as plataformas *OTTs* disponíveis no mercado
- Desenvolver e Implementar o modelo de dados para a plataforma *OTT* a ser desenvolvida
- Estudar e implementar o conceito de *MVC* na programação *web*
- Estudar e aplicar a biblioteca *Google Android TV Leanback* para o desenvolvimento *Android*
- Desenvolver *APIs* para comunicação entre o módulo *web* e *Android TV*
- Estudar e aplicar a biblioteca *Okhttp3* para as requisições *REST API* na aplicação *Android TV*
- Estudar e implementar os sistemas de pagamentos usando *PayPal* e *SISP*
- Fornecer ao público em geral uma nova alternativa para distribuição de conteúdos audiovisuais.

1.3 Justificativa

Depois da análise do problema e de ser constatada a inexistência no mercado cabo-verdiano de plataformas abertas e especializadas na distribuição de conteúdos audiovisuais, foi idealizada a criação de uma plataforma *OTT* denominada *YetuYaliy*, que atende de forma rápida e prática as necessidade identificadas, nomeadamente o de acesso a conteúdos audiovisuais produzido no pais e no continente e a facilidade de distribuição dos mesmos, e para alcançar estes objetivos ficou estabelecido que a plataforma a ser desenvolvida estaria dividida em dois módulos, *web* e *Android TV*, sendo assim , neste relatório será detalhada toda a implementação destes dois módulos.

1.4 Metodologias

Para implementação da plataforma pretendida, primeiramente procedeu-se ao levantamento de requisitos para o desenvolvimento do *software* e como seria a sua utilização no dia-dia, em seguida passou-se para a fase de planeamento onde se fez as estimativas de prazos, identificação dos recursos utilizados e elaboração do cronograma do projeto, com os requisitos levantados e com um plano de trabalho traçado, o próximo passo foi analisar os requisitos e elaborar uma ideia daquilo que seria o projeto de *software* em si, para que então pudesse ser possível o seu desenvolvimento, testes e finalmente sua implantação num ambiente real.

Por tanto, para o desenvolvimento da plataforma *YetuYaliy* foi utilizada em específico a metodologia clássica em cascata, que nos próximos capítulos será melhor explicada a sua utilização no projeto desenvolvido.

Em suma a metodologia aqui utilizada buscou a aplicação prática dos conhecimentos para colmatar as necessidades identificadas no mercado cabo-verdiano no que tange a distribuição e disponibilização de conteúdos audiovisuais.

1.5 Abrangência e Público Alvo

Esta plataforma *OTT* permitirá que qualquer pessoa ou produtor de conteúdos no país e no mundo crie uma distribuidora, coloque, promova ou venda seus conteúdos, sendo assim, ela será especializada em conteúdos africanos, mas também estará voltada ao mercado mundial.

Com a aplicação desta plataforma pretende-se abrir uma janela para divulgação dos conteúdos produzidos no continente, permitindo que todos desfrutam de qualquer lugar e a qualquer momento um show incrível, chamado “Africa” acompanhado da diversidade do mundo.

1.6 Organização do Relatório

Este documento encontra-se dividido em sete capítulos, o primeiro apresenta a introdução onde estão os objetivos, a justificativa e a metodologia para o desenvolvimento do projeto.

No segundo capítulo temos o referencial teórico sobre as tecnologias, ferramentas, linguagens de programação e conceitos utilizados na especificação, análise e desenvolvimento do sistema.

O terceiro capítulo apresenta todo o planeamento feito para levar a cabo o desenvolvimento da solução proposta, como a identificação das atividades realizadas, os recursos utilizados e o cronograma seguido.

O quarto capítulo apresenta os resultados das análises e especificação de requisitos necessários para o projeto e sua implementação.

No quinto capítulo temos os resultados obtidos por meio de alguns *screenshots* e também serão mostrados os resultados obtidos mediante a aplicação de testes de desempenho na plataforma desenvolvida.

No sexto capítulo estão presentes as conclusões, as dificuldades encontradas e as sugestões para trabalhos futuros.

Para finalizar, temos o sétimo capítulo, onde encontram-se todas as referências bibliográficas utilizadas na elaboração deste relatório segundo o estilo de referências *IEEE*.

2 REFERENCIAL TEÓRICO

O principal objetivo deste projeto é o desenvolvimento dos módulos *web* e *Android TV* da plataforma *YetuYaliy*, para tal foram utilizadas linguagens de programação, tecnologias e ferramentas que auxiliassem da melhor forma todo o processo, as linguagens de programação utilizadas foram: *JavaScript*, *PHP* e *Java* sendo que, *JavaScript* e *PHP* utilizadas no desenvolvimento *frontend* e *backend* da aplicação *web* e o *Java* no desenvolvimento *Android TV*.

Neste capítulo serão apresentadas tais linguagens por forma a justificar o porquê da sua utilização, também serão mostradas outras ferramentas e tecnologias utilizadas ao longo do desenvolvimento. Como é o caso do *Yii framework 2.0* para o desenvolvimento *web*, já para o desenvolvimento *android* foi utilizada a biblioteca *Leanback Android TV* que facilita o desenvolvimento para a plataforma da *Google*. Será mostrada e especificada o *board SBC Khadas*, que para além de possibilitar o teste da aplicação *android*, fornece recursos para decodificação de sinais de tv digital, funcionalidade que será implementada também na parte TV. Também será explicada os conceitos relacionados as *APIs REST*, já que esta plataforma integra com outros sistemas, como é o caso do *SISP* e do *Paypal*, utilizada nas questões relacionadas a pagamentos de subscrições.

2.1 *JavaScript*

Conforme descrito em [2], *JavaScript* é uma linguagem de programação dinâmica para computadores. A sua utilização mais comum se dá em desenvolvimento de páginas *webs*, cujas implementações permitem que o script do lado do cliente interaja com o utilizador e crie páginas dinâmicas.

O *JavaScript* nasceu em 1995 como um projeto interno da empresa *Netscape*, com objetivo de resolver as necessidades de criação de páginas mais dinâmicas. Inicialmente, a linguagem era chamada de *LiveScript*, porém, devido a popularidade da linguagem *Java* que também surgiu na mesma época, os responsáveis decidiram que a linguagem iria se chamar *JavaScript*, numa jogada de *marketing* em parceria com a *Sun Microsystem* para alavancar a nova linguagem dos navegadores.

A linguagem *JavaScript* como conhecemos hoje só foi possível graças a padronização da mesma feita em 1997, pela empresa *Netscape* juntamente com a associação europeia de padrões *ECMA* (acrônimo para *European Computer Manufacturers Association*), devido a este

facto, hoje a linguagem é conhecida oficialmente como *ECMAScript* e ela é definida pelas especificações estabelecidas na norma *ECMA-262*.

Uma das grandes vantagens do *JavaScript* e que ajuda a entender o porquê da sua utilização neste projeto, deve-se ao facto de ela ser uma linguagem que não consome tantos recursos no processo de desenvolvimento, já que ela não requer ferramentas de desenvolvimentos tão pesados. Outras vantagens desta linguagem são:

- Menos interação com o servidor - Podemos validar a entrada do utilizador antes de enviar para o servidor. Isso economiza o tráfego, o que significa menos carga no servidor.
- *Feedback* imediato para os visitantes - não é necessário esperar que a página seja recarregada para saber se o utilizador esqueceu de inserir algo.
- Maior interatividade – pode-se criar *interfaces* que reagem quando o utilizador passa o mouse sobre elas ou ative-as através do teclado.
- *Interfaces* mais ricas – Podemos usar *JavaScript* para incluir itens como componentes de arrastar e soltar, e controles deslizantes para fornecer uma interface avançada aos visitantes do site.

2.2 *PHP 7.4*

O *PHP* é uma linguagem de *script* de código aberto e uso geral, especialmente usado em desenvolvimento *web*. O *PHP* pode ser facilmente incorporado ao *HTML*, o que o torna uma linguagem acessível e flexível.

Essa linguagem de *script* foi desenvolvida em 1994 por Rasmus Lerdorf, que o criou inicialmente para o seu uso pessoal, por esse motivo o nome da linguagem ter significado durante muitos anos: "*Personal Home Page*", ou seja, a linguagem na verdade era um conjunto de ferramentas para uso pessoal em páginas *HTML*. Apesar de seu começo humilde, o *PHP* é hoje uma das linguagens de programação mais utilizada no mundo do lado dos servidores, isso ocorre porque o *PHP* possui uma curva de aprendizagem simples, o que permite que mesmo os iniciantes entendam ela e possam utiliza-la de uma forma bem rápida, ao mesmo tempo fornece recursos avançados dos quais desenvolvedores mais experientes podem tirar proveito, conforme explicado em [3].

Ao longo dos anos o *PHP* vem sofrendo alterações que o melhoraram em muitos aspetos, na sua versão 7.4 as mudanças mais relevantes em relação as versões anteriores foram:

- Aumento em velocidade e *performance*: que se traduz num aumento de 31% em relação a versão 7.0 e 10% em relação a versão 7.2 e 7.3.
- Possibilidade de adição de vírgula à direita para argumentos de função e parâmetros de chamada: essas vírgulas são adicionadas à uma lista de elementos, parâmetros ou propriedades. Eles podem ser úteis, pois permitem um código mais limpo, mais simples de entender e editar.

A adoção do *PHP* neste projeto deve-se ao facto de ela ser uma linguagem que se integra muito bem com o *MySQL*, que é o *SGBD* usado na plataforma *YetuYaliy*, e também de ser a linguagem de programação usada no *framework Yii* 2.0 utilizada para agilizar o processo de desenvolvimento *web frontend* e *backend* desta plataforma.

2.3 *Java*

Esta linguagem de programação assim como foi explicada em [4], foi originalmente desenvolvida pela *Sun Microsystem*, iniciada por James Gosling e lançada em 1995 como componente principal da plataforma *Java* da *Sun Microsystems* (*Java* 1.0 [*J2SE*]).

A versão mais recente do *Java Standard Edition* no momento da escrita deste relatório é o *Java SE* 15. Com o avanço do Java e sua popularidade generalizada, várias configurações foram feitas para atender a vários tipos de plataformas. Por exemplo: *J2EE* para aplicativos corporativos, *J2ME* para aplicativos móveis.

Conforme mostrada em [4], uma das características mais popular do Java é o facto de ela ser uma linguagem multiplataforma, ou seja, ela permite que desenvolvedores escrevam um único código e possam executa-la em diversas plataformas, isso é possível graças ao *Java Virtual Machine*, que faz a conversão do código para qualquer sistema operacional, outras características relacionadas a esta linguagem deve-se ao o facto dela ser:

- Orientado a Objeto;
- Simples
- Seguro
- Portável
- *Multithreading*
- Performativo
- Distribuído
- Dinâmico

O Java junto com o *Kotlin* e o *Dart* são hoje as principais linguagens de programação usadas no desenvolvimento de aplicativos *Android*, apesar da forte campanha da Google para fazer do *Kotlin* a sua linguagem principal de desenvolvimento, este processo ainda ira levar alguns anos. Neste projeto foi escolhido a linguagem Java pelo facto de utilizar bibliotecas como o *Android TV Leanback*, que é fortemente influenciado pela sintaxe *Java*, essa biblioteca fornece classes e métodos que permitem agilizar o processo de criação de aplicativos *Android TV*, respeitando sempre as diretrizes do Google.

2.4 *MySQL 5.7*

O *MySQL* é um sistema de gerenciamento de base de dados relacional de código aberto (SGBDR) da Oracle, baseado na linguagem de consulta *SQL* (*Structured Query Language*). O *MySQL* roda em praticamente todas as plataformas, incluindo *Linux*, *UNIX* e *Windows*. Embora possa ser usado em uma ampla gama de aplicativos, o *MySQL* é mais frequentemente associado a aplicativos *web*.

Originalmente concebido pela empresa sueca MySQL AB em 1995, o *MySQL* foi adquirido pela *Sun Microsystems* em 2008 e depois pela Oracle quando comprou a Sun em 2010. Os desenvolvedores podem usar o *MySQL* sob a *GNU* General Public License (*GPL*).

A escolha do *MySQL* como *SGBD* para o desenvolvimento deste projeto se sustenta nos seguintes factos apresentados em [5]:

- *Open Source*, não é preciso pagar nada para usa-lo;
- O *MySQL* é um *SGBD* muito poderoso por si só. Ele lida com grandes subconjuntos de funcionalidades e pacotes de base de dados robustos.
- O *MySQL* funciona em vários sistemas operacionais e em conjunto com diversas linguagens, incluindo *PHP*, *Java* que são as linguagens usadas na plataforma *YetuYaliy*.
- O *MySQL* tem uma alta performance, mesmo gerenciando conjuntos de dados gigantescos.
- O *MySQL* suporta grandes volumes de dados nas tabelas, até 50 milhões de linhas ou mais. O limite de tamanho de arquivo padrão para uma tabela é de 4 GB, que pode ser aumentada se o sistema operacional puder suportar para um limite teórico de 8 milhões de terra bytes (TB).

2.5 *APIs REST*

Para definir "*API REST*", primeiramente é preciso esclarecer o que é "*REST*" e o que é "*API*".

De acordo com L. Richardson em [6], *REST* significa "*Representational State Transfer*". É um padrão de arquitetura para gerenciar e representar informações na *web*. Os conceitos *REST* são referidos como recursos e geralmente são representados no formato *JSON* "*JavaScript Object Notation*".

API significa "Interface de Programação de Aplicativos". Estabelece um conjunto de regras que permitem que um aplicativo possa comunicar com outro. Essas "regras" podem incluir operações de criação, leitura, atualização e exclusão.

As *APIs REST* podem ser usadas em qualquer aplicativo que se conecta à *internet*. Se os dados de um aplicativo puderem ser criados, lidos, atualizados ou excluídos usando outro aplicativo, isso geralmente significa que uma *API REST* está sendo utilizada.

Na plataforma *YetuYaliy* foi aplicada os conceitos das *APIs REST*, já que é necessário utilizar informações provenientes da base de dados alocado no servidor onde a aplicação *web* se encontra hospedado, e com isso, realizar algumas operações relacionadas a validação dos utilizadores no aplicativo *Android TV*, permitindo assim, a utilização de uma única base de dados facilitando a gestão das informações.

2.6 Sistemas de Pagamento

Sendo o *YetuYaliy* uma plataforma de distribuição de conteúdos audiovisuais pode haver situações que impossibilitam a distribuição dos mesmos de forma gratuita, seja pelo modelo de negócio da distribuidora que detém os direitos sobre, ou por outras questões jurídicas e direitos autorais.

Por esses motivos é necessário ter na plataforma meios que possibilitam as distribuidoras fazerem cobranças sobre os conteúdos. E durante essas operações de cobranças e pagamentos a plataforma tem de garantir a máximo de segurança. Hoje quer no mercado nacional e internacional existem soluções prontas e de fácil integração com sistemas *web*.

A nível mundial o mais popular é o *gateway* de pagamento- *PayPal* e a nível nacional temos o *SISP* que integra com a Rede Vinti4 de operações bancarias nacionais.

2.6.1 *Paypal*

A integração do *PayPal* na plataforma foi feita utilizando as *APIs* da ferramenta conhecida oficialmente como "*PayPal Express Checkout*", ela é uma solução concebida para vendedores *on-line* que permite que os clientes comprem produtos ou serviços facilmente, sem a necessidade de inserir os detalhes de remessa e cobrança.

Isso garante que as informações pessoais e de pagamento dos clientes sejam transmitidas com segurança. Também torna as compras *on-line* mais rápidas e fácil.

Ao utilizar a ferramenta *PayPal Express Checkout*, o desenvolvedor tem a opção de decidir onde as transações podem ocorrer, são elas:

- Fazer um *redirect* para a página *web* do *PayPal*, onde o cliente efetua o login na sua conta, escolhe o método de pagamento, efetua o pagamento e é direcionado de volta para a página do comerciante juntamente com as informações do *status* da transação (sucesso ou insucesso),
- Abrir um *iframe* na página do comerciante, realizar todo o passo descrito anteriormente, porém sem a necessidade de sair da página e voltar

Por uma questão de simplicidade de código e de maior segurança, no projeto optou-se pela primeira opção, e para que tudo suceda sem nenhum problema foi preciso configurar no servidor alguns valores de configuração para o acesso a *API* do *PayPal* conforme descrito em [7]:

- *EndPoint* (*URL* de pedido): para ambientes de testes pode-se utilizar o *link sandbox* (https://api-3t.sandbox.paypal.com/nvp), para ambiente *live* o *link* é(https://api-3t.paypal.com/nvp)
- *APIUsername*: informação do nome da conta do comerciante para onde será feita a transação, ela é disponibilizada pelo *PayPal* para contas do tipo negócio (*Business Account*)
- *APIPassword*: é o *password* de acesso a *API* da conta comerciante para se fazer o deposito
- *APISignature*: *Token* de acesso à *API* da conta do comerciante para onde será depositado o valor da transação, necessário para maior segurança, já que a troca de informação é feita *peer-to-peer*
- *ReturnUrl*: define para onde as informações referentes ao *status* da transação serão processadas no site do comerciante, quando o cliente retornar ao mesmo.

- *CancelUrl*: define onde é processado o cancelamento do pedido de compra do cliente no servidor do comerciante, caso este desista no meio da transação

2.6.2 SISP

O *SISP* fornece para soluções *web* ferramentas que possibilitam o pagamento *online* num modo de funcionamento muito parecido com o *PayPal*.

O serviço *web* disponibilizado pela plataforma nacional consiste num conjunto de protocolos que permitem a integração de pagamentos no site do comerciante, por meio da rede vinti4 estabelecido através da troca de mensagens entre a página do comerciante e o *site* da *SISP*, usando o método *post*.

De acordo com as especificações da documentação em [8], cabe ao comerciante no seu *site* garantir a escolha do produto/serviço a ser comprado/consumido pelo cliente e, após o *checkout* deve-se enviar os parâmetros necessários via método *post* à uma página *biz_vbv_clientdata.jsp* do *MPI* (*Merchant Plug-In*, que é a aplicação responsável pelo processamento das transações na *internet*), onde será feita a recolha dos dados do cartão do cliente, permitindo processar a transação e tratamento do resultado da transação que é enviada ao site do comerciante via método *post* no parâmetro *urlMerchantResponse*.

Aqui também é necessário configurar alguns valores no servidor que são:

- *EndPoint* (URL de Pedido): para os ambientes de testes o *link* da *API* é o seguinte (https://mc.vinti4net.cv/Client_VbV_v2/biz_vbv_clientdata.jsp)
- *PosID*: identifica o terminal virtual do comerciante na rede vinti4, é gerado pelo *SISP* e atribuído ao comerciante.
- *PosAuthCode*: código para verificar a autenticidade do terminal virtual do comerciante na rede vinti4, é gerado pelo SISP e atribuído ao comerciante.
- *MerchantID*: código que identifica o comerciante, ou seja, para onde as transações se destinam na rede vinti4, é gerado pelo *SISP* e atribuído ao comerciante.

2.7 *Yii Framework 2.0*

Conforme explicada na documentação oficial em [9], o *Yii* é um *framework PHP* baseada em componentes que permite o desenvolvimento rápido de aplicações *web* modernas. O nome *Yii* (pronuncia-se *Yee* ou *[ji:]*) significa "simples e evolutivo" em chinês. Em inglês a sigla é um acrônimo para *Yes It Is*!

Devido à sua arquitetura baseada em componentes e ao sofisticado suporte de armazenamento em cache, o *Yii* é adequado para o desenvolvimento de aplicativos como: portais, fóruns, sistemas de gerenciamento de conteúdo (*CMS*), projetos de comércio eletrônico, serviços *RESTful web* e assim por diante. As características que o *Yii* possui que o tornam um *framework* fácil de usar e extremamente poderosa são:

- Implementação do padrão arquitetural *MVC* (*Model-View-Controller*) que promove a organização do código.
- Adoção da filosofia de que o código deve ser escrito de uma maneira simples e elegante, porém *Yii* nunca tentará forçar o desenvolvedor a usar algo principalmente, com o objetivo de seguir estritamente algum padrão de *design*.
- Sendo um *framework* de desenvolvimento *fullstack*, o *Yii* fornece muitos recursos testados e prontos para uso tais como, *query builders* e *ActiveRecord*, para base de dados relacionais e *NoSQL*, suporte ao desenvolvimento de *API RESTful*, suporte a armazenamento em *cache* de várias camadas entre outras.
- O *Yii* é extremamente extensível. Dá ao desenvolvedor a liberdade de personalizar ou substituir quase todas as partes do código do núcleo. Com isso é possível tirar proveito da arquitetura de extensão do *Yii* para usar ou criar extensões próprias.
- O *Yii* prioriza o desempenho e a alta performance das aplicações.

Na plataforma *YetuYaliy*, o *Yii* foi utilizado no desenvolvimento da aplicação *web* e no desenvolvimento das *APIs REST*, tudo isso feito de forma rápida e automatizada, utilizando os conceitos de componentes e da arquitetura *MVC*, inerentes ao *core* do *Yii* 2.0 *framework*.

2.8 *Android*

O *Android* é um sistema operacional baseado em *Linux* e de código aberto para dispositivos móveis, como *smartphones*, *tablets* e mais recentemente TV. O *Android* foi desenvolvido pela *Open Handset Alliance*, liderada pelo Google e outras empresas.

O *Android* oferece uma abordagem unificada para o desenvolvimento de aplicativos, o que significa que os desenvolvedores precisam apenas criar suas aplicações, e elas podem ser executados em diferentes dispositivos que rodam o *Android*.

Desde sua primeira aparição em 2008, com a sua versão 1.0, até hoje o *Android* já conta com 11 versões principais e mais de 26 subversões, sendo a mais recente o *Android 11* liberada em 2020, conforme descrito em [10].

O gerenciamento de pacotes para os aplicativos desenvolvidos para o sistema operacional *Android* é feito usando o formato *APK* e habitualmente elas são desenvolvidas usando a linguagem *Java* ou *Kotlin*, neste projeto assim como já foi citado anteriormente, a linguagem escolhida foi *Java*.

2.8.1 *Android TV*

Android TV é um sistema operacional baseado no *kernel* do *Linux* projetado principalmente para televisores, que se complementam com outros dispositivos, como *smartphones*, *tablets*, e relógios inteligentes. O *Android TV* foi apresentado em 25 de junho de 2014, na Conferência de Desenvolvedores *Google I/O*, conforme pode ser conferido em [11].

Atualmente algumas empresas parceiras da Google, já vêm com o sistema operacional *android* nos seus dispositivos tv, são elas:

- Sony
- Sharp
- Panasonic
- LG
- Samsung
- Philips
- Semp/TCL

Porém, neste projeto por uma questão de negócio e de maior mobilidade optou-se por utilizar o sistema operacional *Android TV*, a partir de um *SBC* (*Single Board Computer*) da marca *Khadas*, que em parceria com a *Super Celeron* disponibiliza para seus *boards* a versão 7

,8 e 9 do *Android TV* licenciado pela Google. E por esse motivo ela conta com algumas funcionalidades incorporadas tais como:

- Acesso ao Google *Play*, loja de aplicativos da Google para *smart tv*
- Google *assistant*, um assistente pessoal inteligente desenvolvido pela Google que entre outras tarefas é utilizada nos sistemas de buscas dentro do *SO*
- Google *Cast*, que é uma tecnologia que permite interação do utilizador com o que está passando na tv como, series e filmes, também permite a partilha de ecrãs entre dispositivos *android*

A versão 7 do *Android TV* também conhecida como *nougat* será a utilizada neste projeto.

2.8.2 *Android TV Leanback Library*

O desenvolvimento *Android TV* difere em muitos aspetos do desenvolvimento *Android* padrão (*mobile*), a principal é a navegação que se dá em um televisor e que é feita usando um controle remoto. Devido a esta característica o Google, empresa proprietária da *Android*, elaborou documentações especificando as características que programadores e *designers Android TV* devem ter em conta ao construir suas aplicações. E para auxiliar a implementação de tais especificações a mesma criou uma biblioteca em *java* chamado *Android TV Leanback Library*.

O *Leanback Library* oferece uma ampla variedade de classes para a criação de aplicativos TV, como componentes, *widgets* de *interface* do utilizador e até manipulação de média e componentes.

Para o desenvolvimento *Android TV* da plataforma *YetuYaliy*, a utilização desta biblioteca revelou-se essencial e os resultados serão mostrados no capítulo cinco.

2.9 *OKHTTP3 Library*

Neste relatório já foi explicado o que são as *APIs REST* já que a mesma foi utilizada especialmente na aplicação *Android TV*. Porém, para que tal sucedesse foi preciso configurar clientes *HTTP* para executar as requisições.

Este cliente foi configurado usando a biblioteca java *OKHTTP* que de acordo com a documentação oficial da biblioteca em [12], é um cliente *HTTP* eficiente que suporta a versão 2 do *HTTP* e permite que todas as solicitações ao mesmo *host* compartilhem um soquete com o seu *pool* de conexões, o que reduz a latência na solicitação se o mesmo não estiver disponível,

utilizando o compactador *GZIP* reduz-se o tamanho de *downloads* das requisições e com o seu *cache* evita-se o acesso a rede para solicitações repetidas.

Outra característica importante desta biblioteca é a sua alta recuperação em redes com conexões problemáticas, se o serviço contiver vários endereços *IP*, o *OkHttp* tentará endereços alternativos se a primeira conexão falhar, isso é necessário para *IPv4 + IPv6* e serviços hospedados em *data centers* redundantes. O *OkHttp* suporta recursos *TLS* modernos (*TLS 1.3*, *ALPN*, fixação de certificado). Ele pode ser configurado para retornar à uma ampla conectividade.

Com tantas características e benefícios associados era mais do que evidente a utilização do cliente *OKHTTP* na aplicação *Android TV* da plataforma *YetuYaliy*.

2.10 *Khadas Board*

Neste projeto para testar e avaliar a aplicação *Android TV* num ambiente adequado, foi utilizado o *SBC Khadas*.

O *Khadas* é um *board SBC open source* que possibilita a instalação de sistemas operacionais proporcionando aos utilizadores um ambiente computacional completo, para este caso de estudo, o sistema operacional instalado foi o *Android TV 7(Nougat)*.

2.10.1 Especificações

A versão do *board* utilizado para este projeto foi o *Khadas VIM 2*, que de acordo com o site oficial em [13], este *SBC* é um poderoso computador de placa única do tamanho de um cartão de crédito, com diversas opções de gabinetes, nas últimas versões lançadas ela já vem com o *Android 7.1* pré-instalado.

Esse *SBC* possibilita a instalação de grandes variedades de sistemas operacionais de código aberto, como *Armbian*, *Arch Linux* ou *Ubuntu* 18.04, e até *Android* de inicialização tripla, *Linux* e *LibreELEC* com imagens personalizadas do *eMMC*, tudo isso desenvolvido e mantido pela comunidade e parceiros do *Khadas*. O poderoso processador *Solog Amlogic A53* de 8 núcleos e 1,5 GHz facilita o trabalho de abrir aplicativos e executar tarefas do tipo *desktop*.

Uma característica importante deste box, e que foi utilizada no projeto é que ela pode transformar-se numa poderosa *TV Box* ou *Home Media Center*, instalando um sistema operacional de media como *LibreELEC, CoreELEC ou Android TV*. A *GPU* Mali *T820* fornece

decodificação *UHD H.265 / VP9* de hardware a 60 quadros por segundo e processamento de vídeo *HDR10* e *HLG HDR*, para uma reprodução de vídeo 4K suave.

A nível de áudio o *Khadas VIM 2* possibilita a acoplação da placa tom *Khadas* para saída de áudio *RCA* de alta fidelidade, para além disso, com o *volumio* instalado, é possível reproduzir arquivos de áudio sem perdas, como *WAV* ou *DSD256*, enquanto controla toda a sua experiência auditiva com o aplicativo para *smartphone* do *volumio*.

De uma forma resumida podemos reduzir o *Khadas VIM 2* em seis características principais:

- Alta performance: garantida pelos seguintes atributos *CPU Octa-Core* de 1,5 GHz e 64 bits, *GPU T820MP3* e *DDR4* de até 3 GB e *eMMC* de 64 GB.
- Conectividade ilimitada: garantida pelos seguintes atributos 2x2 *MIMO* 802.11ac *WiFi*, 5.x Bluetooth, *LAN* Gigabit e uma porta *USB-C 2.0*.
- Características avançadas: já que ela possui o *WOL* (*Wake On LAN*) e o *RSDB WiFi* que melhoram a experiência do utilizador.
- Fácil integração com outras placas, que é facilitado graças a existência de 40 pinos *GPIO*, *slot* para ventilador de refrigeração, um *MCU* programável e *Khadas TST*.
- Um poderoso central multimídia: que conta com uma porta *HDMI2.0a* e *VPU* que suporta reprodução de 10 *bits 4K H.265/VP9@60fps*.
- Formato reduzido, o mais interessante de tudo isto é que todas estas incríveis características cabem num *board* do tamanho de um cartão de crédito.

Nas figuras 1 e 2 são mostradas as legendas da vista de cima e de baixo respetivamente do *SBC Khadas Vim 2*:

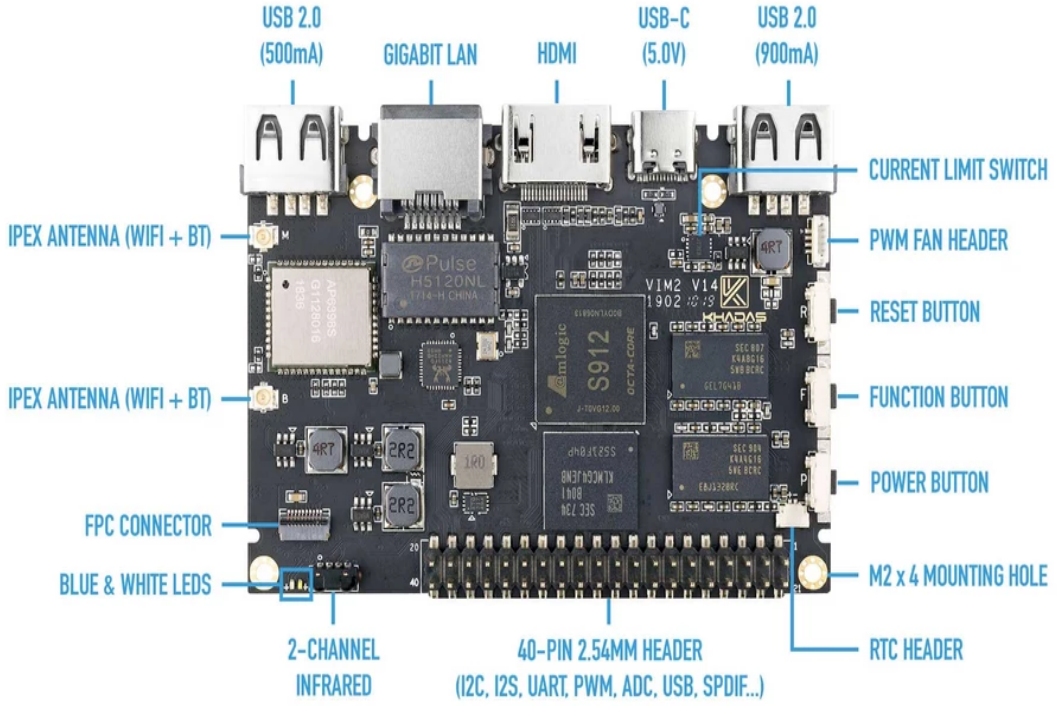

Figura 1: Imagem e Legenda do *Khadas VIM 2*, Vista de Topo [14]

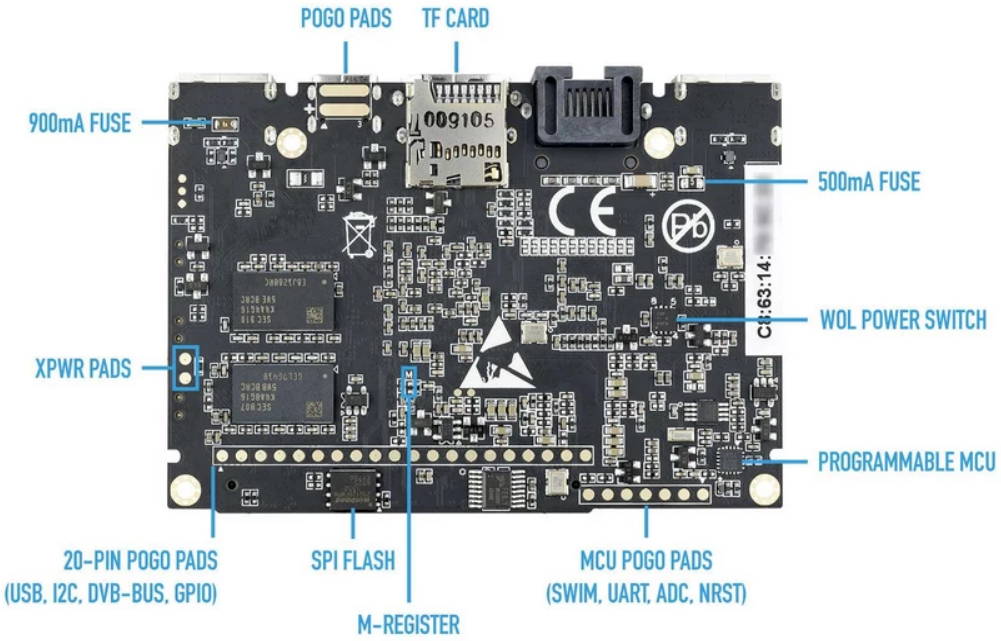

Figura 2: Imagem e Legenda do *Khadas VIM 2*, Vista de Baixo [15]

2.10.2 Descodificador *DVBT2*

Como forma de incrementar mais funcionalidades a aplicação *Android TV* desenvolvida, foi acoplado ao *Khadas board* uma placa descodificadora de sinais digitais de tv, esta placa conta com um descodificador compatível com as normas:

- *DVBS*
- *DVBS2*
- *DVBT*
- *DVBT2*
- *DVBC*

Desta forma a aplicação *android* desenvolvida permitirá que utilizadores possam assistir TV normalmente dentro da aplicação sem a necessidade de ter conexão à internet, nas figuras 3 e 4 encontram-se as legendas da placa descodificadora utilizada.

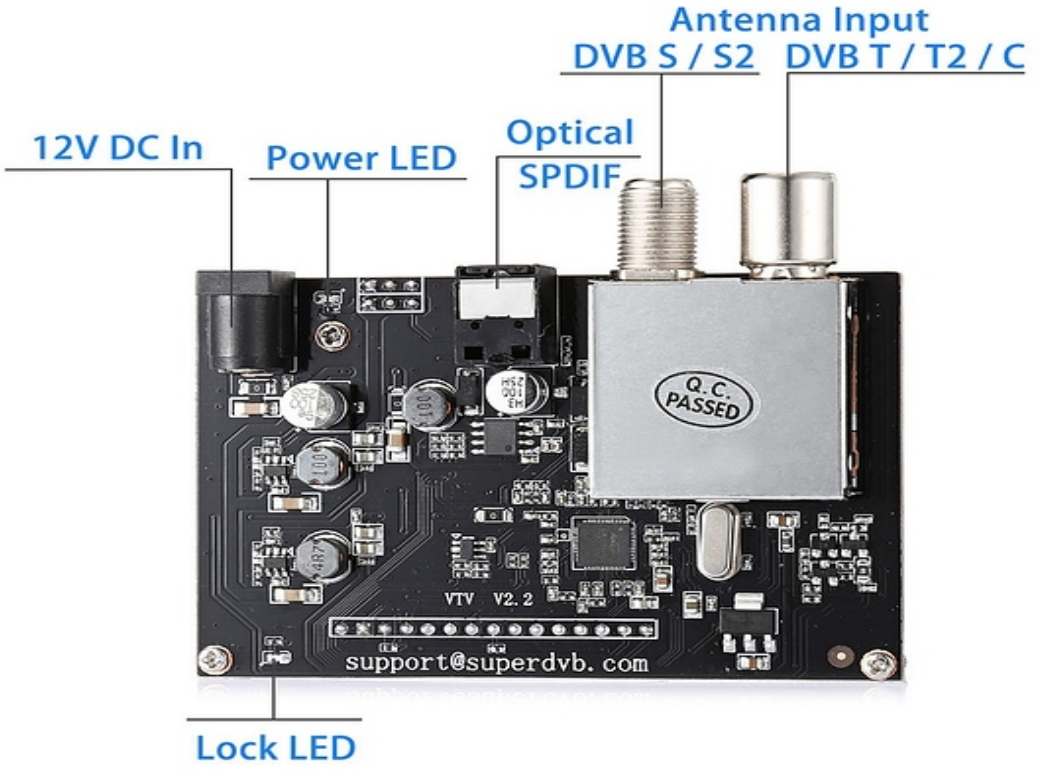

Figura 3: Imagem e Legenda da Placa Descodificadora de Sinais de TV Vista de Topo [16]

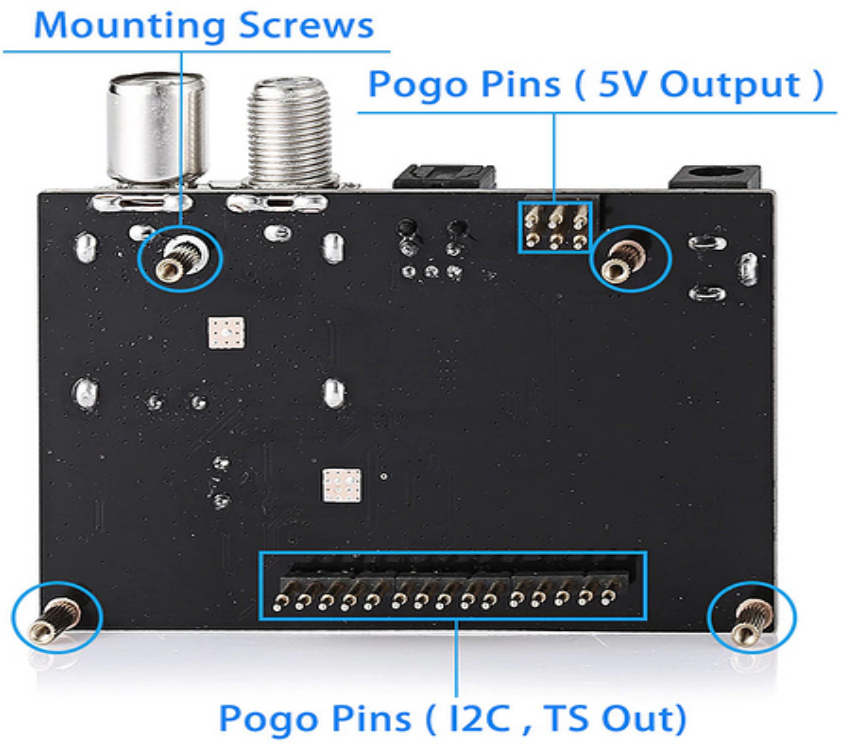

Figura 4: Imagem e Legenda da Placa Descodificadora de Sinais de TV Vista da Base [16]

2.11 Plataformas *OTTs*

A plataforma aqui desenvolvida enquadra num estilo de negócio conhecido atualmente como plataformas *OTTs*, como citado na introdução, este tipo de plataforma dá aos utilizadores liberdade de consumir conteúdos através da *internet*. Na plataforma *YetuYaliy* os conteúdos consumidos são conteúdos em vídeo, através de uma abordagem popular e difundido principalmente pelas redes sociais, que são os formatos *VOD* (sigla para *Vídeos On Demand*).

Mas afinal de contas, que outros formatos de vídeos existem neste tipo de negócio? E mais, quais são as formas de gerar receitas com essas plataformas?

2.11.1 Formatos de Vídeos das *OTTs*

Conforme explicado na Amazon Advertising em [1], um aspeto importante quando se cria uma plataforma *OTT* diz respeito ao tipo de vídeo que se pretende transmitir. E neste âmbito existem 2 tipos principais, que dependendo da estratégia das empresas pode-se optar por um ou por outro ou até mesmo por ambos, são elas:

Live

A *live* é uma transmissão ao vivo pela *internet*, o vídeo é gravado e distribuído em tempo real.

É um formato que as redes sociais estão trabalhando de forma mais ativa há anos, destacando a função para os seus utilizadores. As *lives* já são bem populares em redes sociais como o *Facebook*, *Instagram* e *YouTube*.

Apesar da popularidade entre influenciadores digitais, as *lives* podem ser usadas para diversas outras finalidades, como transmissão de palestras, cursos ou eventos. Além do mais, fazer uma *live* é uma ótima maneira de gerar interação com o público e gerar receitas.

VOD

Sigla para vídeo *on demand*, é o vídeo que é gravado antes de ir ao ar. Sendo assim, neste formato, diferentemente das *lives*, podem haver edições no vídeo e regravações de algumas partes para entregar um conteúdo final com maior qualidade.

Este é o formato de vídeo mais comum, pois estamos mais habituados a consumi-los em filmes, séries e outros géneros dentro da *Netflix* e *YouTube*, por exemplo.

Os vídeos *on demand* recebem este nome justamente porque são os espectadores que decidem o que querem assistir e quando querem assistir, sem depender de uma programação fixa.

Na plataforma aqui apresentada o formato de vídeo utilizado será o *VOD*.

2.11.2 Modelos de Negócios Associados as *OTTs*

Um outro aspeto importante sobre as plataformas *OTTs* mostrada em [1], é como elas podem gerar receitas. No que tange a este aspeto são três os modelos de negócios principais:

- *TVOD* ou Transacional: este é o modelo comumente conhecido como *pay-per-view*, ou seja, o cliente só paga o conteúdo que consumir, sem a necessidade de uma assinatura.
- *SVOD* ou Subscrição: já este modelo funciona por assinatura, desta forma é possível fazer cobranças periódicas, geralmente feitas mensalmente, semestralmente ou anualmente.
- *AOTT* ou Publicidade: a fonte de receita neste modelo de negócio vem da inserção de anúncios dentro dos vídeos ou dentro da plataforma.

2.12 Modelo de Negocio do *YetuYaliy*

YetuYaliy é uma plataforma que permite que pessoas e organizações que lidam com produções de conteúdos, possam criar suas distribuidoras virtuais, submeter seus conteúdos, gerenciar informações sobre elas e disponibilizá-los para o público utilizando tanto a plataforma *web* como também a tv.

O *YetuYaliy* é uma plataforma que inicialmente será especializada em distribuir três tipos de conteúdos distintos:

- Filmes
- Séries
- Canais de TV em sinal fechado no formato *IPTV*

Todos esses conteúdos podem ser disponibilizados gratuitamente ou associá-los a algum tipo de subscrição, dependendo da estratégia de negócio de cada distribuidora.

Resumindo, nesta plataforma serão disponibilizados conteúdos no formato *VOD* e as receitas serão geradas através de subscrições que os utilizadores farão para poder aceder aos mesmos, essas subscrições serão: mensais, semestrais e anuais.

2.13 Engenharia de *Software*

Segundo Pressman em [17], podemos definir a engenharia de *software* como uma abordagem sistemática e disciplinada para o desenvolvimento de *software*.

No desenvolvimento de um projeto de *software* conforme citado por Jalote em [18], quanto mais complexo ele é, maior é o empenho que os desenvolvedores devem empregar, para tal, executa-se um conjunto de atividades que fazem parte daquilo que é denominado modelo de processo de desenvolvimento de *software*, onde destacam-se: a análise de requisitos, o projeto de *software*, a codificação e os testes.

Cada uma das atividades do processo de desenvolvimento do *software* é realizada fazendo-se uso de técnicas segundo uma determinada metodologia, podendo ou não serem utilizadas ferramentas. Nesta seção serão descritas todas as técnicas e metodologias seguidas para o desenvolvimento do *software* e a arquiteturas utilizada para o desenvolvimento dos módulos *web* e *Android TV*.

2.13.1 Metodologia de Desenvolvimento em Cascata

Para desenvolver este projeto a metodologia seguida foi a metodologia sequencial ou em cascata, apesar desta metodologia hoje em dia não ser a mais adequada para projetos de *software* grandes, uma vez que ela não lida muito bem com mudanças de requisitos e resultados imediatos conforme explicado em [19]. Para este projeto ela se enquadra perfeitamente, já que, o *software* aqui desenvolvido é um produto mínimo viável onde os requisitos a serem implementados inicialmente estão bem definidos. A "equipa" de desenvolvimento é constituído apenas por uma pessoa, portanto não há riscos de "membros da equipa" ficarem esperando para poderem executar alguma tarefa. Consequentemente é mais difícil executar atividades em paralelo. Um outro ponto importante é que não existe um "cliente" que encomendou o projeto propriamente dito, desta forma os requisitos são os levantados a partir da observação de *softwares* já lançados e em execução no mercado, como a *Netflix*, *YouTube* e outras plataformas *OTTs*.

Voltando ao modelo cascata, ela sugere uma abordagem sequencial e sistemática para o desenvolvimento de *software*. Dessa forma começamos com o levantamento de requisitos ou necessidades junto ao cliente, depois vem a fase de planeamento onde definimos estimativas e cronogramas, após isso partimos para a modelagem onde fazemos a análise e projeto, seguindo pela implementação onde codificamos e testamos. E por fim, a implantação onde efetuamos a entrega e suporte do *software* desenvolvido.

Basicamente na etapa de levantamentos de requisitos ou necessidades estabelecemos junto do nosso público alvo os requisitos do produto, que consiste nos serviços que devem ser fornecidos, limitações e objetivos do *software*. Esta etapa também consiste da documentação e o estudo de viabilidade do projeto para determinarmos o processo de início de desenvolvimento do sistema. Na etapa de planeamento temos a definição de estimativas, cronograma baseando-se nos requisitos e na determinação das tarefas, que por sua vez é estabelecida pelos requisitos. A etapa de modelagem é uma prévia da próxima etapa de implementação, nesta etapa define-se a estrutura de dados, arquitetura do *software*, *interfaces*. A etapa de implementação abrange o desenvolvimento, onde os programas são efetivamente criados e também os testes onde se verificam as lógicas internas do *software* e as funcionalidades externas. As funcionalidades internas normalmente são realizadas com o uso de testes unitários e as externas podem ser realizadas por testes e pelo próprio cliente. Por fim, a etapa de implantação abrange a disponibilização do *software* ao mercado, que é onde instalamos o *software* no servidor junto com outros utilitários como base de dados ou outros itens dependendo do *software* construído.

O suporte é onde tiramos dúvidas dos clientes e a manutenção consiste na correção de erros que não foram previamente detetados.

Na figura 5 é ilustrado de forma mais clara como foi utilizada a metodologia cascata neste projeto. É de realçar que a versão do modelo cascata seguido aqui prevê retornos entre atividades para esclarecimentos e aprimoramentos.

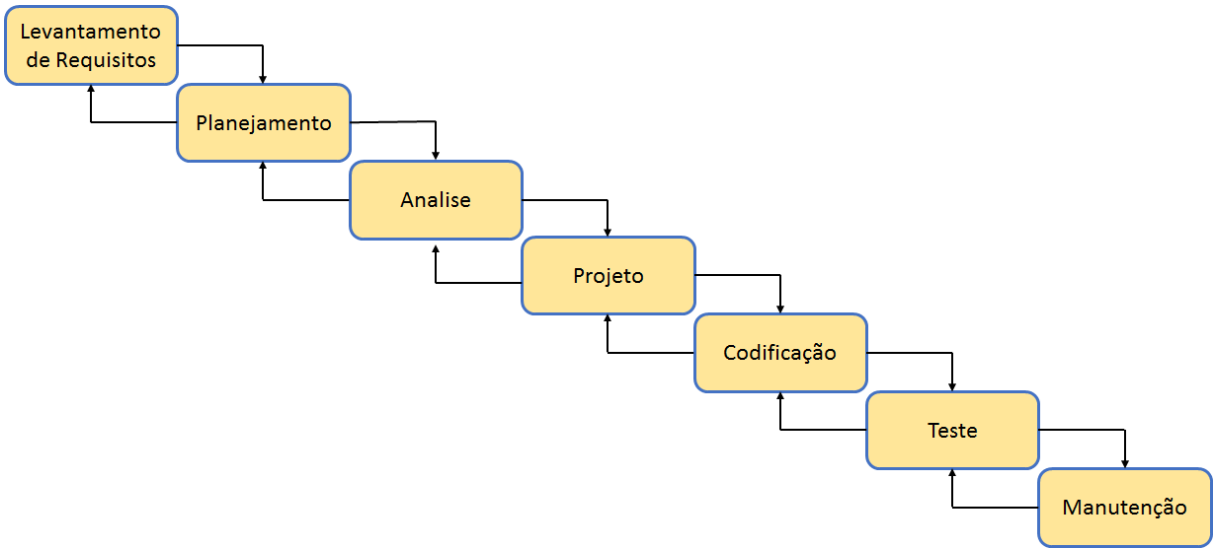

Figura 5: Modelo Cascata Utilizado Neste Projeto [17]

2.13.2 Projeto Arquitetura de *Software*

De acordo com Pressman [17], projeto de arquitetura de um *software* representa a estrutura de dados e os componentes de programas necessários para construir um sistema computacional. Aqui é considerado o estilo de arquitetura que o sistema assumira, a estrutura e as propriedades dos componentes que o constituem, bem como as inter-relações que ocorrem entre todos os componentes da sua arquitetura.

Nesta secção será explicada quais foram os estilos de arquiteturas usadas no desenvolvimento da plataforma *YetuYaliy*, e quais foram os padrões arquiteturais usadas na implementação dos módulos *web* e *Android TV*.

2.13.3 Arquitetura Cliente-Servidor

Na fase do projeto do sistema ficou estabelecido que a arquitetura geral da plataforma *YetuYaliy* seria a arquitetura cliente-servidor, já que a mesma conta com dados armazenados em uma base de dados e servidores *streaming* para disponibilização de conteúdos audiovisuais que precisam ser acedidos por vários clientes, devido a este facto torna-se portanto imperativo

também, a replicação dos serviços em outros servidores evitando a sua sobrecarga, isto na *web* é conhecido como serviços *CDN* (*Content Delivery Network*) que é um serviço que pode ser disponibilizado pelas empresas de hospedagem ou configurado pela equipe de desenvolvimento.

Segundo Sommerville [20], um sistema que segue o padrão cliente-servidor é organizado como um conjunto de serviços e servidores associados e clientes que o acedem e usam os serviços. Os principais componentes desse modelo são:

1. Um conjunto de servidores que oferecem serviços a outros componentes, por exemplo: servidores de arquivos que oferecem serviços de gerenciamento de arquivos.
2. Um conjunto de clientes que podem chamar os serviços oferecidos pelos servidores. Em geral, haverá várias instâncias de um programa cliente executado simultaneamente em computadores diferentes.
3. Uma rede que permite aos clientes aceder esses serviços. A maioria dos sistemas cliente-servidor é implementada como sistemas distribuídos, conectados através de protocolos de *internet*.

As arquiteturas cliente-servidor são normalmente consideradas arquiteturas de sistemas distribuídas, mas o modelo lógico de serviços independentes rodando em servidores separados pode ser implementado em um único computador. Uma vantagem importante deste padrão é a separação e a independência serviços e servidores, onde é possível realizar modificações sem afetar outras partes do sistema.

Em suma numa arquitetura cliente-servidor, a funcionalidade do sistema está organizada em serviços, cada serviço é prestado por um servidor. Os clientes acedem para fazer uso deles.

Mas afinal como que todos estes conceitos são aplicados na plataforma *YetuYaliy*?

Para ajudar a responder a esta pergunta, na figura 6 esta desenhada a arquitetura cliente-servidor da plataforma *YetuYaliy* seguindo o modelo de três camadas onde toda a regra de negocio encontram se no servidor:

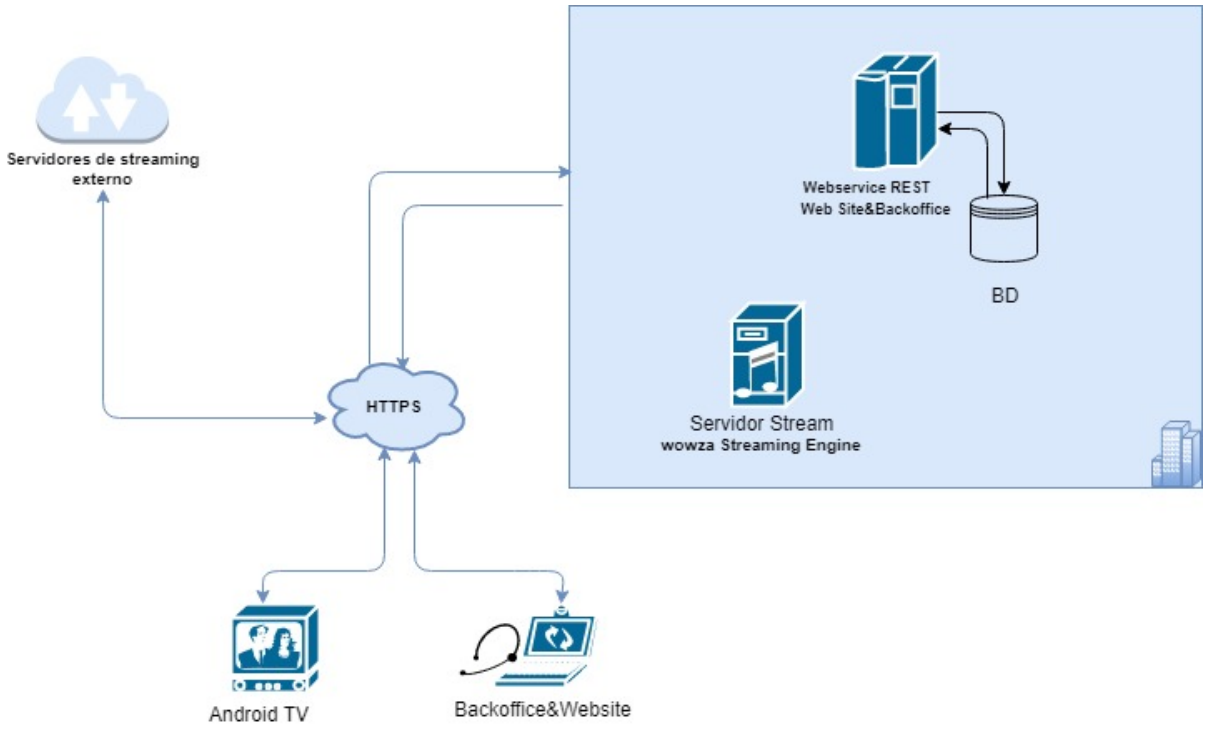

Figura 6: Arquitetura Cliente Servidor Plataforma *YetuYaliy*

Na figura 6 podemos observar que, as aplicações *web* e *Android TV* fazem requisições através da rede para o servidor da plataforma *YetuYaliy* solicitando informações provenientes da base de dados, também as aplicações do *YetuYaliy* podem fazer requisições a servidores de *streaming* externos para aceder conteúdos de vídeos que não estão armazenados no servidor interno.

A adoção da arquitetura cliente-servidor refletiu de forma direta na organização e arquitetura do código fonte das aplicações *web* e *Android TV,* de seguida serão mostras e detalhadas os padrões arquiteturais usadas nos dois módulos desenvolvidos.

2.13.4 *Model View Controler(MVC)*

Segundo Sommerville [20], este padrão é a base do gerenciamento de interações em muitos sistemas baseado na *web*. Ela separa a apresentação e a interação dos dados do sistema.

O Sistema é estruturado em componentes lógicos que interagem entre si, o componente *model* gerência o sistema de dados e as operações associadas a esses dados, o componente *view* define e gerência como os dados são apresentados ao utilizador, o componente *controler* gerência a interação do utilizador, por exemplo: teclas, cliques do mouse e passa essas interações para o *view* e o *model*.

Esse padrão de arquitetura de código é usado quando existem várias maneiras de se visualizar e interagir com os dados. Também quando são desconhecidos os futuros requisitos de interação e apresentação dos mesmos, o que é muito bem-vindo em projetos que utilizam metodologias ágeis.

Já foi referido antes que para o desenvolvimento *web* da plataforma *YetuYaliy* foi utilizada o *framework Yii 2.0* que utiliza o padrão *MVC* junto com outras entidades, e seu funcionamento é descrito no diagrama da figura 7:

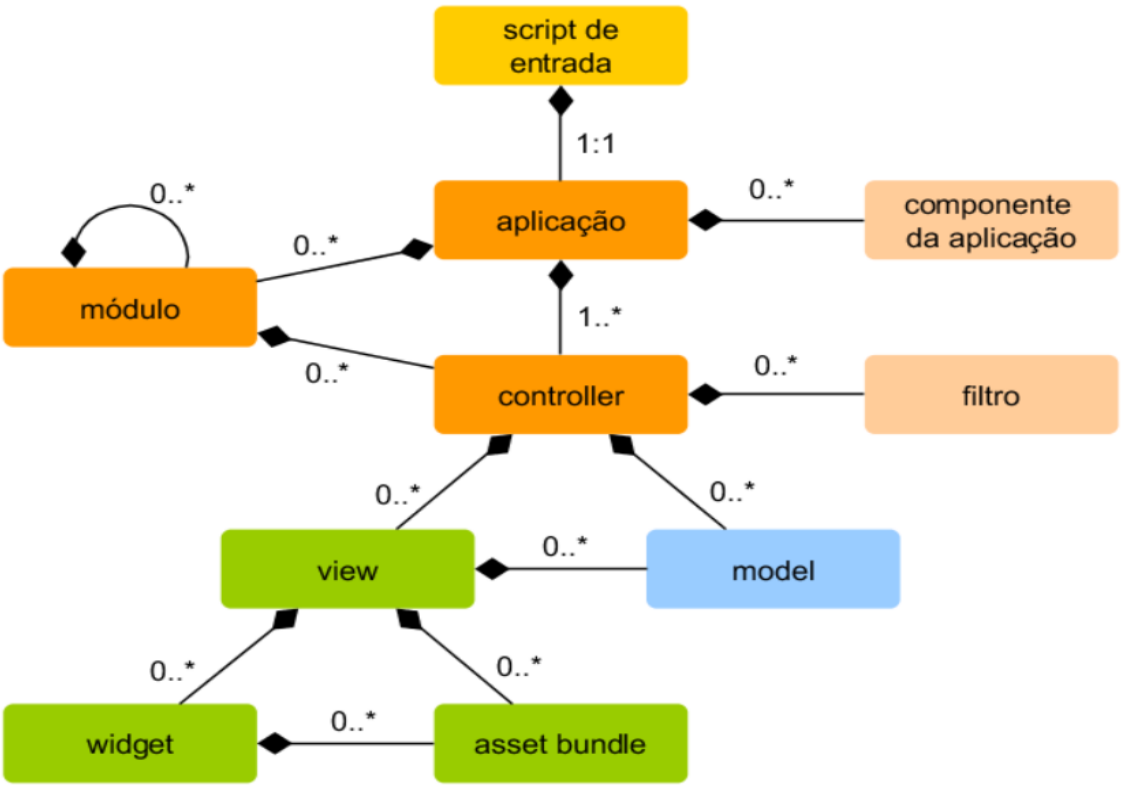

Figura 7: Funcionamento das Aplicações *Yii* Usando *MVC* Junto Com Outras Entidades [21]

Neste diagrama e possível ver a inserção de outras entidades a interatuarem juntos com o padrão *MVC* são elas:

- *scripts* de entrada: são scripts *PHP* acessíveis aos utilizadores finais. São responsáveis por iniciar o ciclo de tratamento de uma requisição.
- aplicações: são objetos globalmente acessíveis que gerenciam os componentes da aplicação e os coordenam para atender as requisições.

- componentes da aplicação: são objetos registrados com as aplicações e fornecem vários serviços para atender as requisições.
- módulos: são pacotes autocontidos que inclui o *MVC* na sua estrutura. Uma aplicação pode ser organizada em termos de múltiplos módulos.
- filtros: representam código que precisa ser chamado pelos *controllers* antes e depois do tratamento propriamente dito de cada requisição.
- *widgets*: são objetos que podem ser embutidos em *views*. Podem conter lógica de *controller* e podem ser reutilizados em diferentes *views*.

2.13.5 *Model View Presenter(MVP)*

O padrão *Model View Presenter* é uma sub-variação do *MVC* e neste projeto foi utilizada na arquitetura do código do módulo *Android TV*, basicamente o seu funcionamento é bastante similar ao *MVC* com a diferença de que aqui o componente *controller* é substituído pelo componente *presenter*, que para além de lidar com as interações do utilizador ela também serve para agrupar e definir como é que os dados serão apresentados ao utilizador, e por isso do nome *presenter* (apresentador)

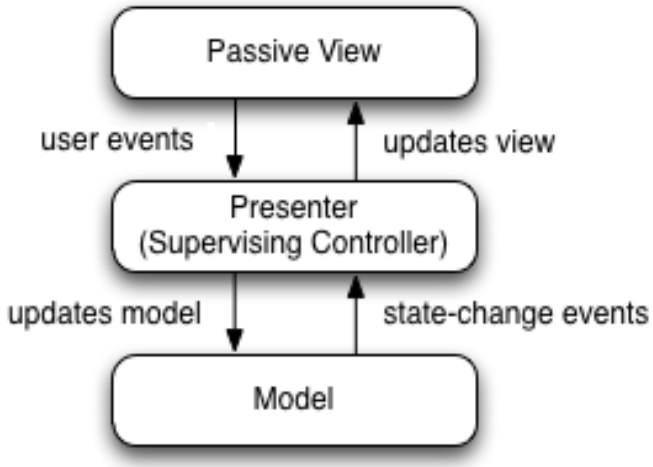

Figura 8: Funcionamento do Padrão de Arquitetura *MVP* [9]

Com base na figura 8, podemos afirmar que o *MVP* é um padrão de arquitetura projetado para organizar melhor o código através da separação da lógica e da apresentação, facilitando os testes da aplicação onde:

- O *model* é onde estão definidos os dados a serem exibidos.
- O *view* é uma *interface* passiva que exibe dados (o modelo) e roteia comandos do utilizador (eventos) para apresenta-los na *interface*.
- O *presenter* age de acordo com o *model* e a *view*. recupera dados de repositórios (o *model*) e os formata para exibição.

3 PLANEAMENTO DO PROJETO

O objetivo do planeamento do projeto de *software* é estabelecer planos razoáveis para a execução das atividades de engenharia de *software* e para a gestão do projeto. Ela compreende a elaboração de estimativas do trabalho a ser realizado, o estabelecimento de compromissos necessários, a definição de planos para a realização do trabalho bem como os recursos necessários para execução do projeto.

Nesta ótica será apresentada neste capítulo o resultado do planeamento do projeto, dando maior ênfase as atividades que foram consideradas como sendo as fundamentais para o desenvolvimento do *software*.

3.1 Atividades do Projeto

Como base na metodologia cascata descrita no capitulo dois, estabeleceu-se que para este projeto as principais atividades seriam:

1. Encontros com o orientador, esta foi uma tarefa realizada quando necessário e possível ao longo do ciclo de vida do projeto, para esclarecimentos e coordenação das atividades do projeto
2. Levantamento, estudo e análise das plataformas similares existentes no mercado, no que diz respeito ao desenvolvimento propriamente dito, ficou determinado que seria uma ótima ideia estudar as plataformas como a *Netflix*, o *Roku* o *YouTube* que têm modelos de negócios muito parecidos com o que se quer implementar, afim de aproveitar as boas ideias de todas elas e construir um sistema que atenda as necessidades destacadas no início do projeto da melhor forma possível.
3. Identificação e estudo das ferramentas a serem usados para a implementação do sistema, estando ciente do ambiente onde as aplicações iriam operar, foram identificadas um conjunto de ferramentas e tecnologias que auxiliam no processo de implementação, e obviamente algumas destas ferramentas e tecnologias eram desconhecidas. Então revelou-se ser imprescindível o estudo das mesmas antes de aplicar no projeto
4. Levantamento e análise de requisitos, na verdade o levantamento de requisitos começou desde o primeiro momento que surgiu a ideia de desenvolver a solução, porém como era numa fase bem inicial, aquelas que foram levantadas eram

requisitos superficiais, por isso a medida que o projeto ia se aproximando da fase de implementação foi necessário fazer-se um estudo mais aprofundado e levantar requisitos técnicos essenciais para a construção de um produto mínimo viável.

5. Modelação do sistema, com os requisitos levantados esta atividade foi mais uma compilação dos resultados obtidos, por forma a ter uma visão de como seria o sistema a ser desenvolvido.
6. Implementação do sistema, nesta atividade foram postas em prática todas as ferramentas e tecnologias estudadas para implementar os requisitos identificados de acordo com os resultados da modelação do sistema.
7. Testes do Sistema, a medida que a fase de implementação inicial ia se aproximando do fim, alguns módulos que já estavam prontos foram testados com o objetivo de encontrar falhas e corrigi-las. Também esta atividade serviu como forma de avaliar se o que estava a ser produzido ia ao encontro daquilo que se propôs inicialmente.
8. Escrita dos documentos do projeto, ao longo de todo o seu ciclo de vida foram produzidos documentos, que tinham como objetivo explicar, descrever e detalhar conceitos importantes do sistema, por fim foi produzido este relatório que no fundo é a compilação de todos os documentos escritos desde o primeiro momento.

3.2 Estimativas de Prazos

O projeto desenvolvido foi realizado num período de 113 dias iniciado à 09 de março de 2020 e terminado à 30 de junho de 2020, e a distribuição das semanas para realização de cada atividade ficou da seguinte forma:

- Encontros com o orientador (16 semanas)
- Levantamento, estudo e análise das plataformas similares existentes no mercado (1 semana)
- Identificação e estudo das ferramentas a serem usados para a implementação do sistema (2 semanas)
- Levantamento e analise de requisitos (3 semanas)
- Modelação do sistema (2 semanas)
- Implementação do sistema (5 semanas)

- Testes do Sistema (5 semanas)
- Escrita dos documentos do projeto (16 semanas)

3.3 Recursos Utilizados

Para o desenvolvimento da plataforma *YetuYaliy*, os principais recursos utilizados foram:

1)Materiais

- Computador: utilizada para programação e teste local da aplicação *web*
- *Khadas board VIM 2*: para teste da aplicação *Android TV*

2)*Softwares*

- *Visual Studio Code*: editor de código fonte da aplicação *web*
- *Android Studio*: *IDE* desenvolvimento *android*
- *Apache Server*: servidor de teste local
- *Bitbucket*: repositório online para o controle de versão dos códigos e documentação das aplicações usando o *git*
- *Microsoft OneDrive*: gerenciamento das documentações do projeto
- *Drawio*: para construir os diagramas *UML* do projeto

3.4 Cronograma do Projeto

O cronograma do projeto ficou da seguinte forma:

Atividade	Data Previsto de Inicio	Data Previsto de termino	Duração
1-Encontros com o orientador	09/03/2020	30/06/2020	113 dias
2-Levantamento, estudo e analise de plataformas similares existente no mercado	09/03/2020	16/03/2020	7 dias
3-Identificação e estudo das ferramentas a serem usadas para a implementação do sistema	30/03/2020	12/04/2020	14 dias
4-Levantamento e análise de requisitos	13/04/2020	04/05/2020	21 dias
5-Modelação do sistema	05/05/2020	19/05/2020	14 dias
6-Implementação do sistema	20/05/2020	28/06/2020	39 dias
7-Testes	20/05/2020	28/06/2020	39 dias
8-Redigir relatório Final	09/03/2020	30/06/2020	113 dias

Tabela 1: Cronograma do Projeto

4 ANÁLISE E PROJECTO DE *SOFTWARE*

No âmbito do desenvolvimento de *software* o termo "análise" está relacionado à definição do problema a ser solucionado no domínio do negócio. Já o termo "projeto" está ligado à solução do problema definido na fase de análise. Para que o projeto seja feito, é necessário, então, uma análise previamente realizada dos problemas. E para solucionar estes problemas é necessário aborda-los utilizando algum paradigma.

Neste projeto, devido as suas especificidades e também das linguagens de programação utilizadas, o paradigma de desenvolvimento utilizado foi o de Programação Orientado a Objetos (*POO*), este paradigma tem como principal objetivo aproximar o mundo real do mundo digital através da aplicação de conceitos que permitem solucionar problemas identificados.

Desde que o paradigma orientado a objetos apareceu e começou a ser difundido no mundo da programação de *softwares*, surgiram inúmeros métodos orientados a objetos que propunham formas de aplica-las em projetos reais, porém, só em 1997 que todos os métodos existentes até então foram unificadas dando origem a um, chamado *UML* (*Unified Modeling Language*) que em suma é a linguagem padrão para especificar, visualizar, documentar e construir artefactos de um sistema, podendo ser utilizada em todos os processos de desenvolvimento orientado a objetos, ao longo de seus ciclos de desenvolvimento, usando diferentes tecnologias de implementação dando a quem desenvolve uma visão mais ampla do sistema que deve ser construído, normalmente estas visões são dadas em formas de diagramas.

Neste capítulo serão mostrados os principais diagramas produzidos fruto do resultado da análise do problema a ser resolvido, tais diagramas serviram como base para o projeto e implementação cujo resultado será mostrado no capítulo 5.

4.1 Requisitos Funcionais(RF)

Antes de apresentar os diagramas produzidos é necessário indicar os principais requisitos identificados que serviram de base para a criação dos mesmos, sendo assim, na fase de análise foram identificados os seguintes requisitos funcionais para a plataforma *YetuYaliy*:

Site principal(RFSP)

RFSP1. A plataforma *YetuYaliy* deve fornecer listas de distribuidoras, filmes, séries e *IP TV* para os utilizadores que navegam nela.

RFSP2. A plataforma deve permitir que utilizadores possam obter informações descritivas acerca dos filmes, séries e *IP TV* disponíveis.

RFSP3. Quanto aos conteúdos relacionados a séries, o site principal deve permitir que utilizadores possam navegar por temporadas e episódios.

RFSP4. A plataforma deve fornecer mecanismos de buscas globais de distribuidoras e conteúdos por nome.

RFSP5. A plataforma deve fornecer mecanismos de filtragem de distribuidores e conteúdos por género e/ou categorias.

RFSP6. O sistema deve possibilitar a interação dos utilizadores com os conteúdos através dos comentários.

RFSP7. O sistema deve permitir que utilizadores possam subscrever nas distribuidoras quando for preciso.

RFSP8. A plataforma deve fornecer aos utilizadores listas de distribuidoras nas quais ele se encontram subscrito.

RFSP9. O sistema deve permitir que utilizadores possam proceder a pagamentos de subscrições usando tanto o SISP quanto o *Paypal*

RFSP10. O sistema deve possibilitar que utilizadores possam assistir aos conteúdos de filmes, séries e *IP TV Lives* diretamente no browser.

RFSP11. A plataforma deve permitir que utilizadores possam criar contas *YetuYaliy*, tanto de utilizadores comum como também de provedor de conteúdos.

RFSP12. A plataforma deve fornecer a possibilidade de utilizadores comuns poderem se tornar parceiros do *YetuYaliy* e assim serem provedores de conteúdos.

RFSP13. Para utilizadores inscritos como provedores de conteúdos, a plataforma deve permitir que os mesmos possam criar novas distribuidoras.

RFSP14. O sistema deve permitir que donos das distribuidoras possam gerenciar listas de utilizadores, que são colaboradores das mesmas: adicionar um novo colaborador, bloquear ou atribuir permissões.

RFSP15. A plataforma deve fornecer uma seção de contacto, onde utilizadores da mesma podem submeter suas dúvidas, opiniões, sugestões e problemas e enviar para os gestores do sistema.

RFSP16.O sistema deve permitir que utilizadores possam navegar do site principal para o painel de controle das distribuidoras onde tem permissão de gestão.

RFSP17.No site principal a plataforma deve permitir que utilizadores registrados possam editar informações de perfil

Painel de Controle gestão das distribuidoras(RFPC)

RFPC1. O sistema deve fornecer aos gestores de conteúdos um *dashboard* com um conjunto de informações numéricas referentes a distribuidora, tais como:

- Números de filmes registrados
- Números de Séries registrados
- Números de *TV Lives*
- Números de Colaboradores
- Números de comentários feitos
- Receitas geradas pela distribuidora

RFPC2. No painel de controle da distribuidora, deve ser possível editar informações de perfil do utilizador *logado*.

RFPC3. No painel de controle deve ser possível para o utilizador navegar entre o site principal e o painel sem a necessidade de se fazer *logout*.

RFPC4.O sistema deve fornecer aos gestores das distribuidoras seções separadas de navegação no painel de controle para gerenciar:

- Filmes
- Séries
- *TV Lives*
- Comentários
- Utilizadores (Gerir colaboradores)
- Subscrições (valores para pagamentos mensais, semestrais e anuais)
- Notificações (quando algum evento ocorre ex: novo comentário adicionado)
- *Logs* (Relatório do que cada colaborador fez dentro do sistema)

RFPC5.Na seção de filmes deve ser possível para o utilizador:

- Adicionar um novo filme
- Editar informações de algum filme já criado
- Procurar algum filme pelo título

RFPC6.Na seção de séries deve ser possível para o utilizador:

- Adicionar uma nova série
- Editar informações de algum já criado
- Adicionar nova temporadas a uma série já criada
- Adicionar novos episódios a uma série
- Navegar para lista de todas as temporadas já criadas, independentemente da série e realizar ações como: editar informações e adicionar novos episódios
- Navegar para uma lista de episódios já criados independentemente da série e editar suas informações
- Procurar por séries, temporadas e episódios a partir dos seus títulos

RFPC7. Na seção de *TV Lives* deve ser possível para o utilizador:

- Adicionar um novo *TV Live*
- Editar informações de algum *TV Live* já criado
- Procurar *TV Live* pelo título

RFPC8. Na secção comentários deve ser possível ao utilizador gerenciar e informar como serão tratados os mesmos, quando feitos a algum conteúdo da distribuidora, sendo assim será possível especificar:

- Se os comentários são ou não permitido
- Se os comentários precisam passar por aprovação antes

RFPC9. Na secção de comentários deve ser possível filtrar por conteúdos e/ou por estado.

RFPC10. Na secção utilizadores o sistema deve permitir que seja possível adicionar novos colaboradores.

RFPC11. Na secção utilizadores o sistema deve permitir que seja possível editar informações dos colaboradores.

RFPC12. Na secção utilizadores o sistema deve permitir que seja possível procurar colaboradores por nome.

RFPC13.Na secção subscrição deve ser possível para os gestores de conteúdos informar se a distribuidora é paga ou grátis.

RFPC14. Na secção subscrição deve ser possível informar os valores de subscrição, mensal, semestral e anual.

RFPC15.Na secção de notificações os gestores de conteúdos devem ser capazes de informar quais eventos disparam notificações no painel de controle, tais como:

- Quando um novo conteúdo é criado
- Quando um comentário precisa de ser validado

RFPC16. Na secção de notificações o sistema deve permitir que utilizadores filtrem as mesmas por tipo de conteúdo.

RFPC17.Na secção *logs* os utilizadores podem especificar se serão ao não registrados os mesmos sempre que os colaboradores realizarem alguma ação.

RFPC18.O sistema deve permitir que seja possível filtrar os *logs* por colaborador e/ou por datas.

***Android TV*(RFATV)**

RFATV1.Na tela inicial do aplicativo *android* deve haver um atalho para mudar de utilizador *logado*.

RFATV2.O aplicativo *Android TV* na tela inicial deve fornecer atalho para aceder ao aplicativo tv digital.

RFATV3.O aplicativo *Android TV* deve fornecer ao utilizador forma de atualizar a versão do aplicativo instalado no dispositivo.

RFATV4.O aplicativo deve fornecer um menu lateral que permite o acesso rápido a outras telas, tais como *home*, lista de filmes, lista de séries, lista de *tv lives* criados, lista de conteúdos que têm permissão de acesso, listas de distribuidoras registrados no *YetuYaliy* e lista de aplicativos instalados no dispositivo.

RFATV5.O aplicativo *Android TV* deve fornecer ao utilizador a possibilidade de filtrar filmes e séries por géneros.

RFATV6.O aplicativo deve permitir que o utilizador filtre canais de tv por categorias.

RFATV7.Caso seja necessário o aplicativo *Android TV* deve permitir, a subscrição dos utilizadores nas distribuidoras.

RFATV8.O aplicativo *Android TV* deve permitir assistir os conteúdos em vídeos por parte dos utilizadores.

4.2 Regras de Negócio (RN)

As restrições e condições de funcionamento da plataforma *YetuYaliy*, também foram identificadas na fase de analise, e são as seguintes:

RN1.A navegação no site principal pelos utilizadores é feita independentemente se o mesmo possui ou não uma conta.

RN2.Para assistir qualquer conteúdo na *web*, o utilizador tem de ter uma conta e estar *logado*.

RN3.Para se subscrever em uma determinada distribuidora o utilizador tem de estar *logado*.

RN4.Para deixar comentários em qualquer conteúdo na plataforma, o utilizador tem de estar *logado*.

RN5.Não é possível um utilizador subscrever numa distribuidora se ele for dono ou colaborador da mesma.

RN6.Se o utilizador for dono ou colaborador de uma distribuidora terá acesso liberado a todos os conteúdos da distribuidora.

RN7.O pagamento de subscrições é feito inteiramente no módulo *web*.

RN8.Para ser parceiro da plataforma *YetuYaliy* e disponibilizar conteúdos, o utilizador tem de possuir uma conta distribuidora na plataforma *YetuYaliy* e concordar com os termos de parceria do sistema.

RN9.Para criar uma distribuidora o utilizador tem de ter uma conta *YetuYaliy*, estar *logado* e ser parceiro.

RN10.Somente os donos dos canais podem adicionar utilizadores colaboradores nas suas distribuidoras.

RN11.Só é permitido adicionar utilizadores como colaboradores em uma determinada distribuidora se este não for dono, possuir uma conta *YetuYaliy* registrada e ser parceiro.

RN12.Para aceder ao painel de controle de uma distribuidora o utilizador tem de ser dono do mesmo ou colaborador.

RN13.Os colaboradores de uma distribuidora só poderão realizar tarefas nas quais possuem permissões.

RN14.A permissão de acesso dos colaboradores ao painel de controle de uma distribuidora é dada pelo dono da mesma.

RN15.Estando no painel de controle os utilizadores podem criar sem nenhuma restrição conteúdos de filmes e *TV Lives*

RN16.Para registar uma série o utilizador tem de seguir uma determinada sequência:

1. Registrar informações gerais da série
2. Adicionar pelo menos uma temporada à série
3. Com as temporadas criadas, deve adicionar episódios

RN17.Para adicionar custos de subscrição numa distribuidora o utilizador tem de seguir uma determinada sequência:

1. Informar se a distribuidora possui ou não subscrições pagas
2. Atualizar o preço base das subscrições
3. Indicar quais serão os descontos para subscrições mensais, semianuais e anuais

RN18.O valor referente a cada subscrição será calculado através do preço base e dos valores de desconto, utilizando as seguintes fórmulas:

1. Subscrição mensal:
    - Preço Base - (Preço Base x (Desconto Mensal / 100))
2. Subscrição semianual:
    - (Preço Base - (Preço Base x (Desconto Semianual / 100))) x 6
3. Subscrição anual:
    - (Preço Base - (Preço Base x (Desconto Anual /100))) x 12

RN19.Para aceder aos conteúdos na aplicação *Android TV*, o utilizador tem de possuir uma conta e realizar o login no aplicativo.

RN20.Sempre que houver uma nova atualização do *apk*, as informações referentes estarão na opção *update* na tela inicial.

RN21.Os utilizadores poderão navegar por toda a aplicação e ver as informações descritivas de cada conteúdo disponibilizado

RN22.Só é possível assistir conteúdos se o utilizador *logado* no aplicativo for o dono da distribuidora que possui os conteúdos, ou um dos colaboradores ou se ele estiver subscrito na mesma.

RN23.Os utilizadores podem realizar subscrições grátis nas distribuidoras a partir do aplicativo *Android TV*.

4.3 Requisitos Não Funcionais(RNF)

Segundo Sommerville [20], os requisitos não funcionais descrevem restrições sobre os serviços ou funções oferecidos pelos sistemas, estes requisitos são tão ou mais importantes quanto os requisitos funcionais ou regras de negócio. Infelizmente é muito difícil encontrar empresas que levam isso a sério, e isto é a causa do fracasso de muitos projetos de *software*.

Os principais motivos que justificam a pouca importância que os RNFs têm na maioria dos projetos são que:

- Utilizadores não sabem o que é um Requisito Não-Funcional
- RNF é algo difícil de estimar

E sabendo destes detalhes importantes, na fase de análise do sistema *YetuYaliy* foram destacados um conjunto de requisitos não funcionais extremamente importantes para o sucesso da aplicação que se pretende desenvolver.

Segurança

RNF1.O sistema deve permitir o controle de acesso de acordo com o nível de cada utilizador.

RNF2.Utilizar a função *crypt* (Encriptação unidirecional de *string*) do *PHP* junto com o algoritmo de encriptação *Unix Standard DES-based* para proteger os *passwords* dos utilizadores na base de dados.

RNF3.No formulário de contatos na página *web* utilizar *CAPTCHA* para validar os utilizadores antes de submeterem as informações.

RNF4.Implementar um sistema de *logs* que registra as ações realizadas pelos colaboradores numa determinada distribuidora.

Interoperabilidade

RNF5.As aplicações *web* e *Android TV* devem comunicar utilizando os conceitos das APIs REST.

RNF6.Para validar os pagamentos efetuados na plataforma, há que fazer integração com as *APIs* fornecidas tanto pelo *SISP* quanto pelo *PayPal*.

Usabilidade

RNF7.A aplicação *web* terá de apresentar uma *interface* simples, compreensível e de fácil manuseamento por parte dos utilizadores.

RNF8.A aplicação *Android TV* terá de apresentar uma interface simples, compreensível e de fácil manuseamento por parte dos utilizadores.

Compatibilidade

RNF9.O servidor onde a aplicação *web* será hospedado terá de ser compatível com a linguagem *PHP* 7 ou superior.

RNF10.A versão do *SGBD MySQL* usado no servidor terá de ser a versão 5 ou superior.

RNF11.O kit de desenvolvimento nativo para *Android TV* terá de ser a versão *API level* 24 ou superior.

RNF12.O descodificador de sinal de TV digital utilizado no *Khadas SBC* terá de ser compatível com as normas *DVBT2* e *DVBS*.

Padrões

RNF13.O padrão arquitetural a ser utilizado para o desenvolvimento *web* será o *MVC*.

RNF14.O padrão arquitetural a ser utilizado para o desenvolvimento *Android TV* terá de ser o padrão *MVP*.

Legais

RNF15.O sistema terá de comunicar com as *APIs* externas respeitando as normas e os protocolos de segurança estabelecidos nas documentações das mesmas.

RNF16.A aplicação *Android TV* desenvolvido terá de respeitar as políticas de segurança e privacidade do Google.

RNF17.Os conteúdos disponibilizados na plataforma terão de respeitar as legislações dos direitos autorais vigentes no país.

4.4 Atores do Sistema

Para esta plataforma, na fase de análise foram identificados e documentados os seguintes atores:

Utilizador Comum: pessoas que criam uma conta no *YetuYaliy* apenas com a intenção de navegar pelas plataformas *web* e *Android TV*, afim de encontrar conteúdos e consumi-los.

Produtor de Conteúdos: pessoas que criam uma conta e posteriormente se tornaram parceiros do *YetuYaliy*, afim de poderem criar suas distribuidoras virtuais e poderem disponibilizar conteúdos, ou também colaborarem em outras distribuidoras.

Sociedade Interbancária e Sistemas de Pagamento (*SISP*): este sistema permite validar e processar todos os pagamentos feitos com cartões nacionais registradas na rede vinti4.

***PayPal*:** este sistema disponibiliza um conjunto de *APIs* que permitem validar e processar pagamento feitos, tanto com cartões registrados no *PayPal* como também os internacionais não registrados como *VISA* e *Mastercard*.

4.5 Diagrama de Casos de Uso

Os casos de uso identificados na fase de analise foram divididos em 4 grupos: Navegação no site principal, Gerenciamento de Distribuidoras, Gerenciamento de Pagamentos e Navegação na aplicação *Android TV*.

- Navegação no site principal
    - Registrar na plataforma
    - Realizar Login
    - Editar informações pessoais
    - Procurar conteúdos
    - Ver informações descritivas dos conteúdos
    - Assistir conteúdos
    - Escrever comentários
    - Procurar distribuidoras
    - Ver informações descritivas das distribuidoras
    - Ver lista de conteúdos disponibilizados pela distribuidora
    - Subscrever na distribuidora
    - Ver Lista de Distribuidoras subscrito
    - Tornar-se parceiro do *YetuYaliy*

    - Criar distribuidora
    - Ver lista de distribuidoras com permissão de acesso
    - Gerenciar lista de colaboradores das distribuidoras

- Navegação no aplicativo *Android TV*
    - Realizar Login
    - Mudar de utilizador *logado*
    - Aceder ao *PlayStore* da *Google*
    - Ver *TV digital*
    - Atualizar versão do *apk*
    - Procurar conteúdos
    - Ver informações descritivas dos conteúdos
    - Assistir conteúdos
    - Procurar distribuidoras
    - Ver informações descritivas das distribuidoras
    - Ver lista de conteúdos disponibilizado pela distribuidora
    - Ver lista de conteúdos com permissão de acesso
    - Ver lista de distribuidoras com permissão de acesso

- Gerenciamento de distribuidoras
    - Criar conteúdos
    - Gerenciar conteúdos
    - Gerenciar comentários
    - Adicionar Colaboradores
    - Gerenciar lista de colaboradores
    - Criar preços de subscrição na distribuidora
    - Gerenciar entradas de notificações
    - Gerenciar finanças da distribuidora
    - Ver *logs* de acesso ao painel de controle

- Gerenciamento de Pagamentos
    - Validar pagamentos feitos com cartões vinti4
    - Validar pagamentos feitos com cartões internacionais

A seguir, serão apresentados os diagramas de casos de uso. Esse diagrama segundo J.Rumbaugh [22] documenta o que o sistema faz do ponto de vista do utilizador. Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os utilizadores do mesmo. Nesse diagrama não nos aprofundamos em detalhes técnicos que dizem como o sistema executa as funcionalidades, nesse projeto para apresentar da melhor forma possível os diagrama optou-se por separa-la em três grandes grupos, o primeiro são os referentes a casos de uso de quando um utilizador navega pela página *web* principal do *YetuYaliy*(incluindo os casos de uso para validação de pagamentos), o segundo são os referentes a navegação no painel de controle de uma determinada distribuidora e o último são os de quando um utilizador navega pela aplicação *Android TV*.

4.5.1 Navegação no Site Principal

Neste diagrama estão envolvidos os seguintes atores:

- Utilizador Comum
- Produtor de Conteúdo
- SISP
- *PayPal*

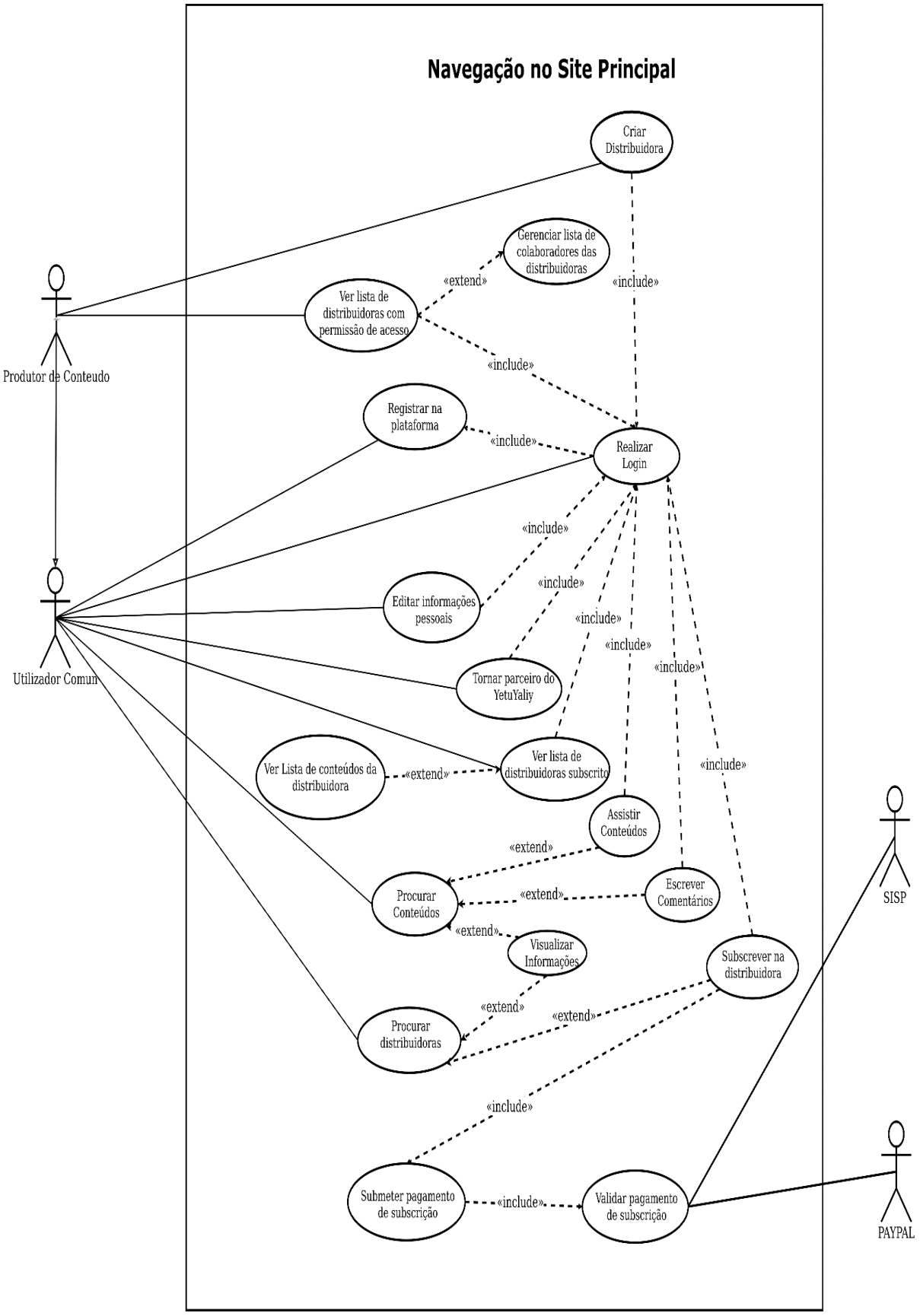

Figura 9: Diagrama de Casos de Uso Navegação no Site Principal

A figura 9 ilustra o diagrama de casos de uso referente a navegação no site principal do *YetuYaliy,* as ações realizadas por cada um dos atores são as mesmas descritas pelos requisitos funcionais de navegação no site principal apresentada na secção 4.1.

As relações entre os casos de uso é normalmente representadas por *extend*(herança) e *include*(inclusão) e elas significam o seguinte:

Include

Quando o caso de uso A "inclui" o caso de uso B, significa que sempre que o caso de uso A for executado o caso de uso B também será executado. A direção do relacionamento é do caso de uso que está incluindo para o caso de uso incluído.

Extend

Quando o caso de uso B estende o caso de uso A, significa que quando o caso de uso A for executado o caso de uso B poderá (poderá – talvez não seja) ser executado também. A direção do relacionamento é do caso de uso extensor (aqui o caso de uso B) para o caso de uso estendido (aqui o caso de uso A).

Os demais diagramas apresentadas nesta secção seguem os mesmos princípios explicados no neste.

4.5.2 Navegação no Painel De Controle

Neste diagrama de casos de uso estão envolvidos apenas os utilizadores que recebem o nome genérico de Produtor de Conteúdo, eles têm como privilégio adicionar conteúdos na plataforma *YetuYaliy* e gerenciar outros aspetos ligados a uma distribuidora.

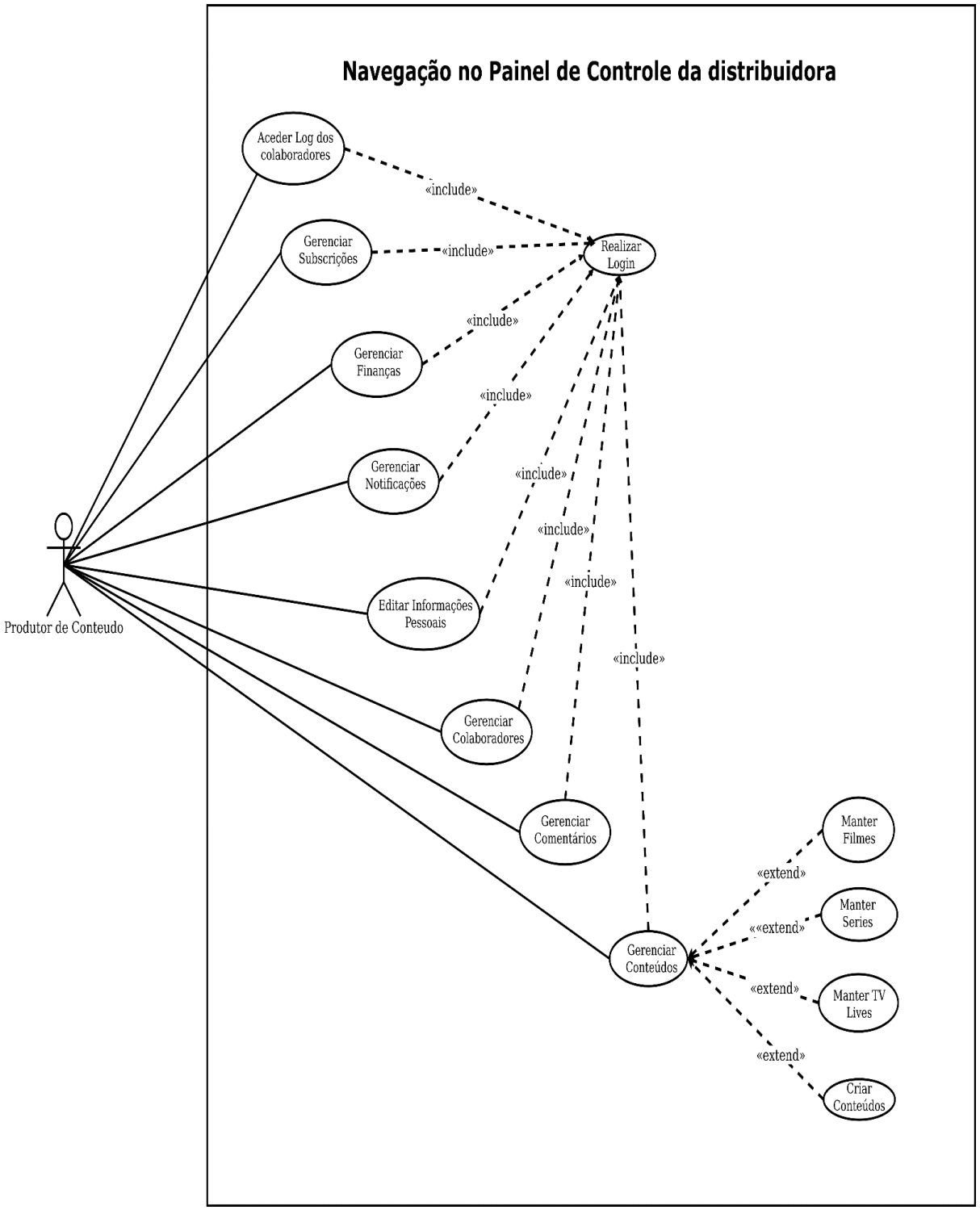

Figura 10: Caso de Uso Navegação no Painel de Controle das Distribuidoras

4.5.3 Navegação no Aplicativo *Android TV*

Este diagrama mostra um conjunto de funcionalidades que a aplicação *Android TV* possui e como ela se relaciona com os atores, que neste diagrama é um ator genérico que recebe simplesmente o nome de Utilizador, já que representa tanto os produtores de conteúdos quanto os utilizadores comuns.

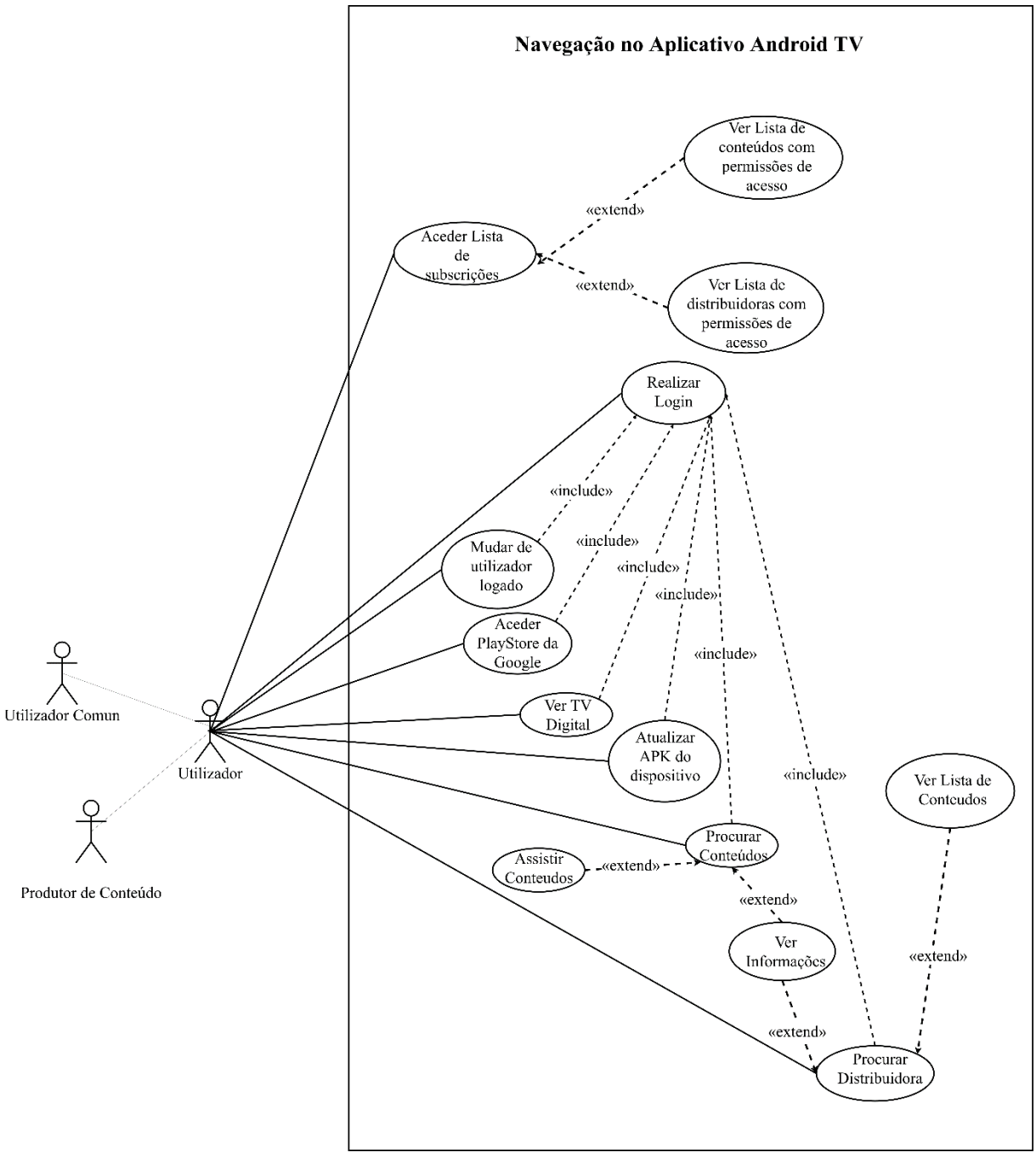

Figura 11: Caso de Uso Navegação no Aplicativo *Android TV*

4.6 Diagrama Entidade Relacionamento

Um diagrama entidade relacionamento (ER) é um tipo de fluxograma que ilustra como "entidades", p. ex., pessoas, objetos ou conceitos se relacionam entre si dentro de um sistema, de acordo com a definição apresentada no manual de referencia UML em [22]. Diagramas ER são mais utilizados para projetar ou fazer depuração em base de dados relacionais nas áreas de engenharia de *software*, sistemas de informações empresariais, educação e pesquisa. Também conhecidos como DERs, ou modelos ER, usam um conjunto definido de símbolos, tais como retângulos, diamantes, ovais e linhas de conexão para representar a conectividade de entidades, relacionamentos e seus atributos. Eles espelham estruturas gramaticais, onde entidades são substantivos e relacionamentos são verbos.

Diagramas ER estão relacionados com diagramas de estrutura de dados (DEDs), que incidem sobre as relações de elementos dentro de entidades em vez de relações entre as próprias entidades. Diagramas ER são também muitas vezes utilizados junto com diagramas de fluxo de dados (DFDs), que mapeiam o fluxo de informações para processos ou sistemas.

Na fase de análise do *software* foi criada o DER do sistema a ser desenvolvido usando o estilo de pé de galinha com o intuito de resolver possíveis problemas de lógica ou de implementação, como resultado foi esboçado o diagrama ER que se encontra na figura 12, os atributos das tabelas podem ser vistos na secção 4.8 onde se encontra o dicionário de dados de cada uma das tabelas.

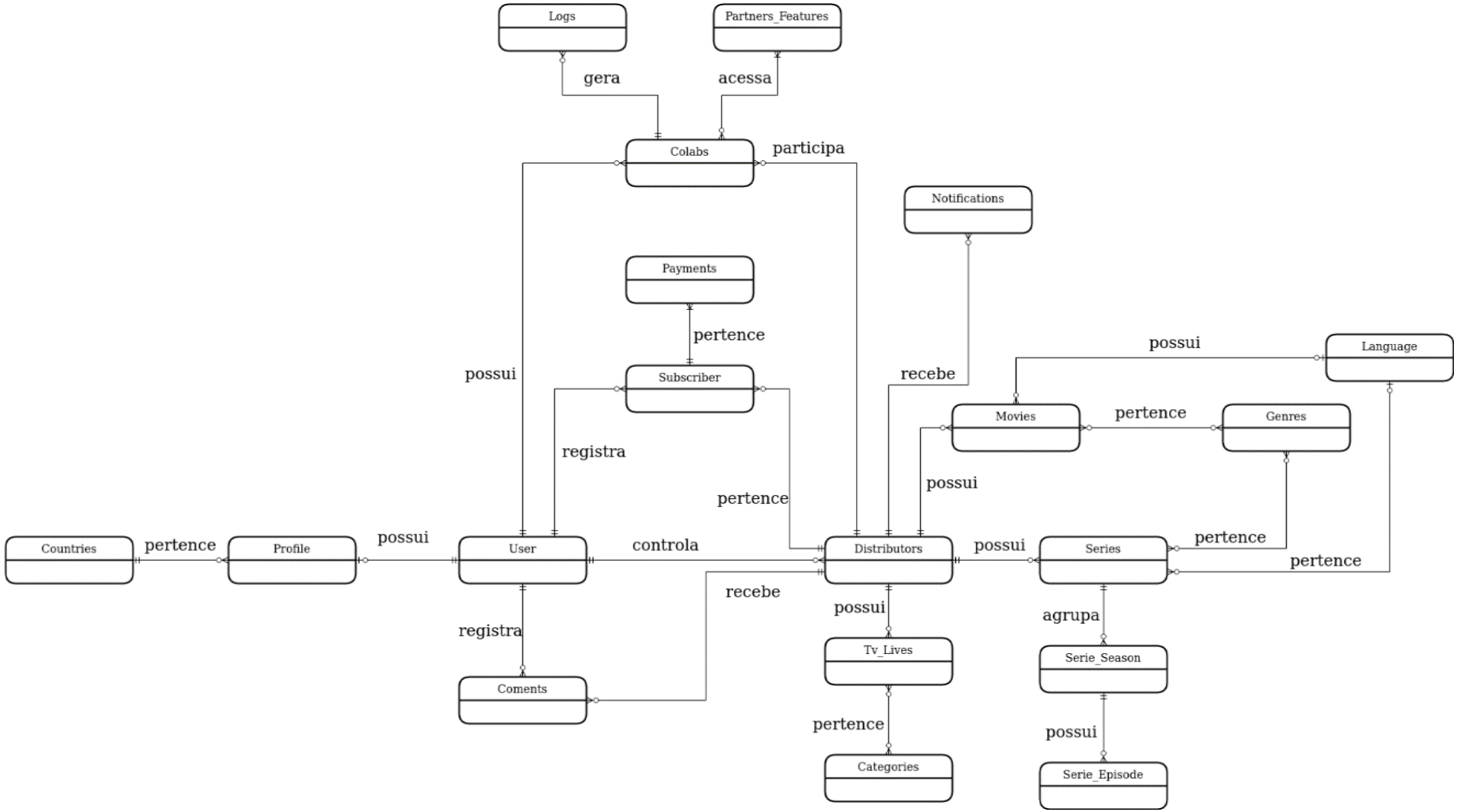

Figura 12: Diagrama ER do Sistema Usando o Estilo Pé de Galinha

Neste diagrama o destaque vai para as entidades:

- *User*
- *Distributor*
- *Tv Lives*
- *Movies*
- *Series*
- *Subscriber*
- *Payments*
- *Colabs*(Colaboradores)

Onde um utilizador representado pela entidade *User* pode ser responsável pela gestão de uma ou mais distribuidoras, controlando assim todos os aspetos relacionados a ela. Por outro lado, quando um utilizador não é o responsável pela distribuidora ele pode ser colaborador em várias, e esta relação é representada pela entidade *Colabs*.

Quando um utilizador não for responsável pela distribuidora ou colaborador, ele tem a opção de subscrever numa ou em várias distribuidoras, esta relação é representada pela entidade *Subscriber*, sendo que ao longo do tempo um utilizador pode fazer pagamentos frequentes de subscrições e a entidade responsável para isso é o *Payments*.

Já a distribuidora representada pela entidade *Distributor* pode possuir um ou vários conteúdos tais como: filmes(*Movies*), Séries(*Series*) e *Tv Lives*.

4.7 Diagramas de Sequência

Um diagrama de sequência é um tipo de diagrama de interação, pois descreve como e em qual ordem um grupo de objetos trabalham em conjunto. Estes diagramas são usados por desenvolvedores de *software* e profissionais de negócios para entender as necessidades de um novo sistema ou para documentar um processo existente. Diagramas de sequência são conhecidos como diagramas de eventos ou cenários de eventos, definição esta que pode ser encontrada no manual de referencia UML em [22].

Os benefícios da criação de um diagrama de sequência dentro de um projeto de *software* é que ela permite:

- Representar os detalhes de um caso de uso *UML*.
- Modelar a lógica de um processo, função ou operação sofisticada.

- Ver como objetos e componentes interagem uns com os outros para concluir um processo.
- Planejar e compreender a funcionalidade detalhada de um cenário existente ou futuro.

Para este projeto, durante a análise foram desenhados três diagramas de sequência que especificam cenários de usos mais importantes para o sistema, tais como:

- Logar no sistema
- Criar conteúdos (uma visão geral independente do tipo de conteúdo)
- Realizar subscrição paga numa distribuidora

4.7.1 Diagrama de Sequencia *Logar* na Plataforma *YetuYaliy*

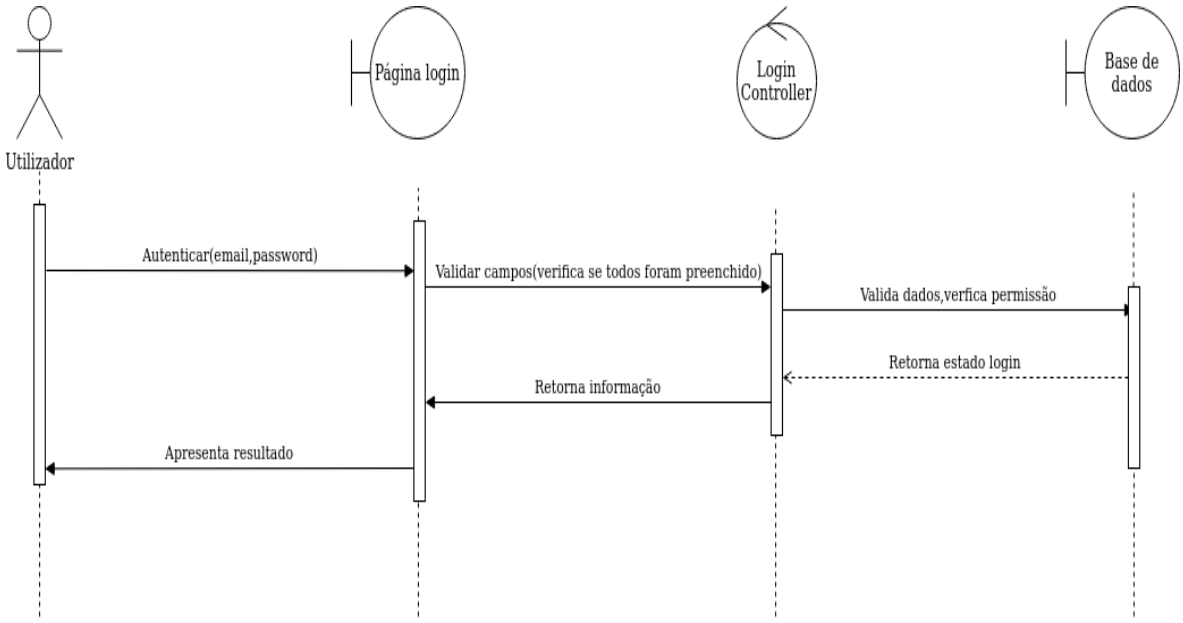

Figura 13: Diagrama de Sequencia *Logar* no Sistema

Neste diagrama primeiramente o utilizador digita o email e o password da sua conta, de seguida o sistema valida os campos para ver se todos foram preenchidos corretamente, em caso negativo a aplicação retorna de volta para a tela de *login* informações de que os campos email ou password são obrigatórios, em caso afirmativo a aplicação passa os dados para o *controller* que faz uma verificação na base de dados buscando uma tupla que satisfaça a condição email e password igual ao digitado pelo utilizador, caso seja encontrado alguma correspondência a sessão do utilizador é iniciada/criada, caso não seja encontrado nenhuma correspondência a

aplicação retorna para a pagina de *login* com a seguinte informação “email ou password incorretos”.

Os restantes diagramas de sequencia seguem os mesmos princípios de verificação e fluxo explicados neste.

4.7.2 Diagrama de Sequencia Criar Conteúdos

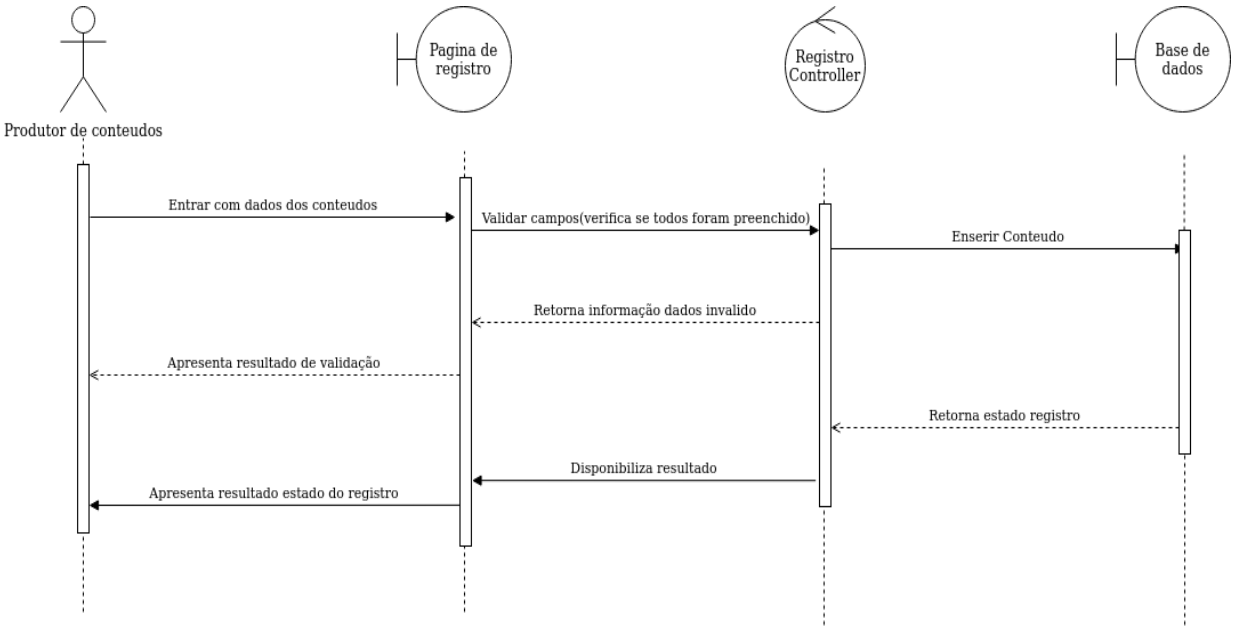

Figura 14: Diagrama de Sequência Criar Conteúdos

4.7.3 Diagrama de Sequência Subscrever numa Distribuidora

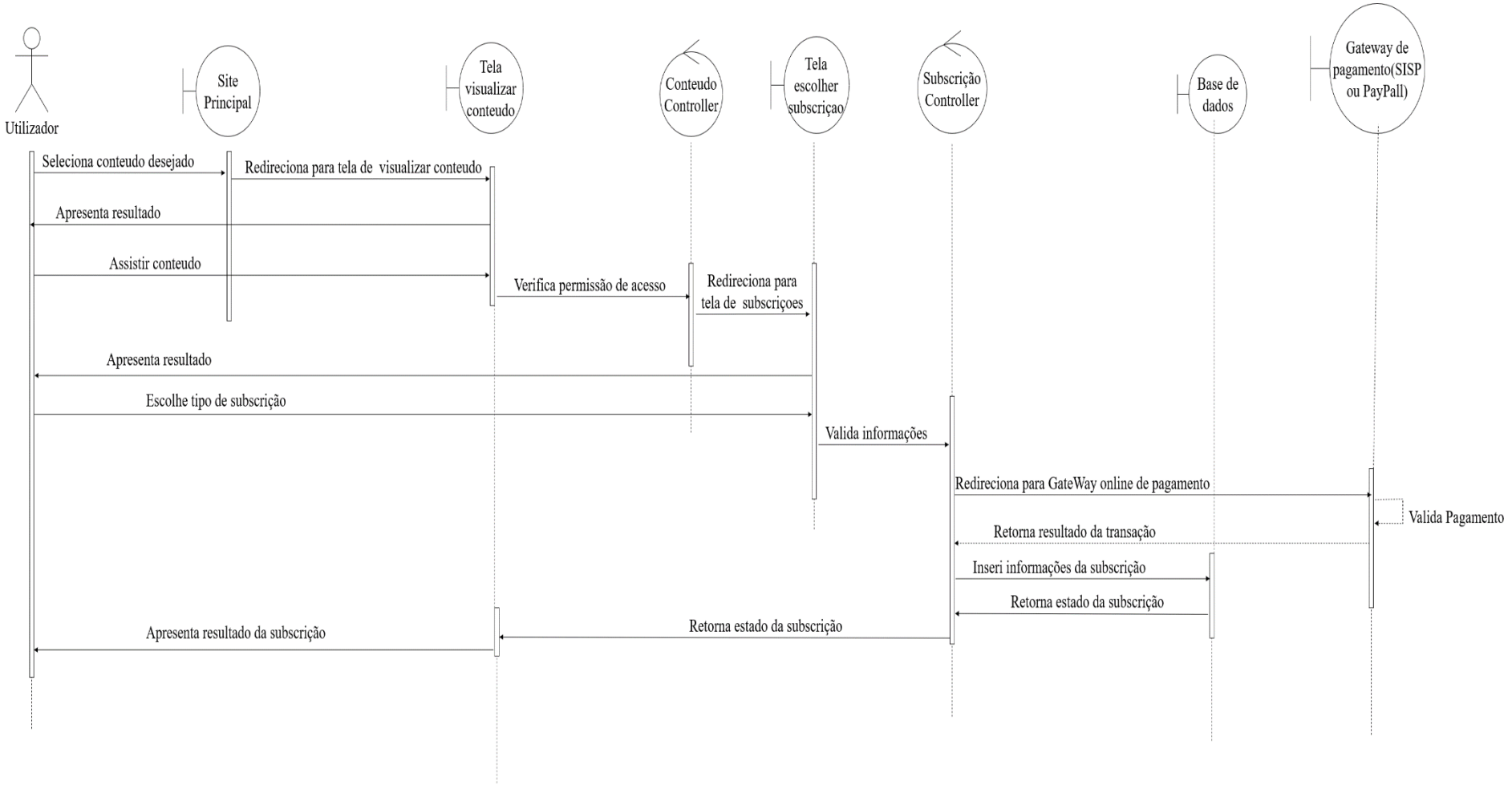

Figura 15: Diagrama de Sequência, Subscrever numa Distribuidora Paga

4.8 Dicionário de Dados do Modelo Físico da Base de Dados

O modelo físico da base de dados descreve como o sistema será implementado usando um sistema *SGBD* específico. Esse modelo geralmente é criado pelo *DBA* e pelos desenvolvedores. O objetivo é a implementação real da base de dados, ele oferece abstrações e ajuda a gerar esquemas, isso ocorre devido à riqueza de metadados oferecidos pelo modelo, com ela é possível visualizar a estrutura dos dados e modelar colunas especificando restrições, índices, *triggers* e outros recursos inerentes a um *SGBD* relacional.

No projeto *YetuYali* como já foi citado anteriormente, o *SGBD* escolhido foi o *MySQL*, sendo assim, de seguida serão apresentados cada tabela que constitui o modelo físico da base de dados juntamente com suas colunas, tipos de dados e tamanho de cada campo num formato conhecido com dicionário de dados, esse dicionário foi gerado de acordo com as especificações do *SGBD* utilizado.

4.8.1 Tabela Categorias

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
Name	varchar	50
Image	varchar	500
app_small_icon	varchar	300
app_large_icon	varchar	300
web_icon	varchar	300
Status	tinyint	1

Tabela 2: Categorias

4.8.2 Tabela Categoria *TV Lives*

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
id_category (FK)	inteiro	11
Id_tv_live (FK)	inteiro	11

Tabela 3: Categorias *TV Lives*

4.8.3 Tabela *Colabs*

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
id_user (FK)	inteiro	11
id_dist (FK)	inteiro	11
Role	varchar	200
created_at	Datetime	
updated_at	Datetime	
Status	Tinyint	1

Tabela 4: *Colabs*(Colaboradores)

4.8.4 Tabela *Colabs* Acesso

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
id_features (FK)	inteiro	11
id_colabs (FK)	inteiro	11
last_update	datetime	
Status	tinyint	1

Tabela 5: *Colabs* Acesso(Acesso aos Colaboradores)

4.8.5 Tabela Comentarios

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
content_comment	varchar	30
id_contens	inteiro	11
Texto	varchar	500
Priority	tinyint	1
Type	char	1

mod_status	tinyint	1
id_com_before (FK)	inteiro	11
id_dist (FK)	inteiro	11
id_user (FK)	inteiro	11
created_at	datetime	
Status	tinyint	1

Tabela 6: Comentários

4.8.6 Tabela Países

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
country_code	varchar	2
country_name	varchar	100

Tabela 7: Países

4.8.7 Tabela Distribuidoras

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Name	varchar	100
company_name	varchar	300
Logo	varchar	200
Website	varchar	300
created_at	datetime	
update_at	datetime	
id_user (FK)	inteiro	11
Status	tinyint	1
allow_comments	tinyint	1
validate_comments	tinyint	1
Type	varchar	1

base_price	decimal	8
monthly_disc	inteiro	3
semi_disc	inteiro	3
annual_disc	inteiro	3
n_comments	tinyint	1
n_comments_pending	tinyint	1
n_movies	tinyint	1
n_series	tinyint	1
n_tv_lives	tinyint	1
reg_log	tinyint	1
reg_only_logs_colabs	tinyint	1

Tabela 8: Distribuidoras

4.8.8 Tabela Géneros

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Name	varchar	10
Image	varchar	300
icon_app_small	varchar	300
icon_app_large	varchar	300
web_icon	varchar	30
Status	tinyint	1

Tabela 9: Géneros

4.8.9 Tabela Géneros Filmes

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
id_movie (FK)	inteiro	11
id_genre (FK)	inteiro	11

Tabela 10: Género dos Filmes

4.8.10 Tabela Géneros Serie

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
id_serie (FK)	inteiro	11
id_genre (Fk)	inteiro	11

Tabela 11: Géneros Serie

4.8.11 Tabela Linguagem

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Name	varchar	20
Status	tinyint	1

Tabela 12: Linguagem

4.8.12 Tabela *Logs*

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Action	varchar	500
created_at	datetime	
id_dist (Fk)	inteiro	11
id_user (FK)	inteiro	11

Tabela 13: *Logs* do Sistema

4.8.13 Tabela Filmes

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
Title	varchar	100
Description	varchar	500
Tags	varchar	500
Cover	varchar	300
Poster	varchar	300
src_teaser	varchar	10
teaser_url	varchar	300
upload_type	varchar	5
video_url	varchar	300
video_original_src	varchar	50
video_id	varchar	30
total_view	inteiro	11
total_rate	Float	11
rate_avg	Float	11
created_at	datetime	
updated_at	datetime	
id_dist (FK)	inteiro	11
id_lang (FK)	inteiro	11
Status	tinyint	1

Tabela 14: Filmes

4.8.14 Tabela Notificações

Nome do campo	Tipo de dados	Tamanho
Id (PK)	inteiro	11
Contente	varchar	50
id_content	varchar	30

Link	varchar	100
created_at	datetime	
id_dist (FK)	inteiro	11
Status	tinyint	1

Tabela 15: Notificações

4.8.15 Tabela *Partner Features*

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Features	varchar	200
Active	varchar	100
Icon	varchar	50
Link	varchar	100
Description	varchar	200
created_at	datetime	
updated_at	datetime	
Status	tinyint	1

Tabela 16: *Partner Features*

4.8.16 Tabela Pagamentos

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
id_subs (FK)	inteiro	11
Value	decimal	10
created_at	datetime	
finish_date	date	
Status	tinyint	1

Tabela 17: Pagamentos

4.8.17 Tabela Método de Pagamentos

Nome do campo	Tipo de dados	Tamanho
id (PK)	Inteiro	10
card_type	Varchar	40
number_card	Inteiro	16
card_month	Varchar	20
card_year	Inteiro	4
Cvv	Inteiro	3
check_out_code	Varchar	10
Region	Varchar	100
City	Varchar	100
address_line	Varchar	100
Zip	Varchar	20
Telephone	Varchar	20
created_at	datetime	
updated_at	datetime	
id_user (FK)	Inteiro	11
Status	Tinyint	1

Tabela 18: Método de Pagamentos

4.8.18 Tabela Perfil

Nome do campo	Tipo de dados	Tamanho
id (PK)	Inteiro	11
first_name	Varchar	100
last_name	Varchar	100
Photo	Varchar	300
Gender	Char	1
birth_date	Date	
country_birth	varchar	100

country	varchar	100
school_level	varchar	100
last_update	datetime	
id_user (FK)	inteiro	11
Status	tinyint	3

Tabela 19: Perfil

4.8.19 Tabela Series

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Name	varchar	30
Poster	varchar	300
Cover	varchar	300
Description	varchar	300
Tags	varchar	500
language (FK)	inteiro	11
id_dist (FK)	inteiro	11
created_at	datetime	
updated_ad	datetime	
Status	tinyint	1

Tabela 20: Series

4.8.20 Tabela Episódios

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Title	varchar	20
Number	inteiro	11
Sinopse	varchar	300
type_upload	varchar	5

video_url	varchar	300
video_src	varchar	2
video_id	varchar	30
avg_time	inteiro	11
created_at	datetime	
update_at	datetime	
id_season (FK)	inteiro	11
Status	tinyint	1

Tabela 21: Episódio

4.8.21 Tabela Temporada

Nome do campo	Tipo de dados	Tamanho
id (PK)	inteiro	11
Name	varchar	20
Number	inteiro	11
description	varchar	300
teaser_src	varchar	10
teaser_id	varchar	20
id_serie(FK)	inteiro	11
created_at	datetime	
updated_at	datetime	
Status	tinyint	1

Tabela 22: Temporada

4.8.22 Tabela Subscritor

Nome do campo	Tipo de dados	Tamanho
id(PK)	inteiro	11
id_usuario(FK)	inteiro	11
id_dist(FK)	inteiro	11

Type	tinyint	1
new_deal_at		
token_payment	varchar	500
Amount	decimal	12
Status	tinyint	1

Tabela 23: Subscritor

4.8.23 Tabela *TV Live*

Nome do campo	Tipo de dados	Tamanho
id(PK)	inteiro	11
Name	varchar	200
description	varchar	500
Poster	varchar	500
Brand	varchar	500
type__upload	varchar	6
video_src	varchar	2
video_id	varchar	500
video_url	varchar	500
stream_link	varchar	1000
Tags	varchar	300
Language	inteiro	11
limite_user	inteiro	11
created_at	datetime	
update_at	datetime	
Status	tinyint	1
Posted	tinyint	1
id_dist(FK)	inteiro	11

Tabela 24: *TV Live*

4.8.24 Tabela Utilizadores

Nome do campo	Tipo de dados	Tamanho
id(PK)	inteiro	11
username	username	255
auth_key	varchar	32
Password_hash	varchar	255
Password_reset_token	varchar	255
Email	varchar	255
Status	Smallint	6
created_at	Inteiro	11
updated_at	Inteiro	11
verification_token	Varchar	255
Level	Tinyint	1

Tabela 25: Utilizadores

4.8.25 Tabela Visualizações

Nome do campo	Tipo de dados	Tamanho
id(PK)	inteiro	11
Contente	varchar	30
id_content	inteiro	11
start_view	datetime	
end_view	datetime	
Id_user(FK)	inteiro	11

Tabela 26: Visualizações

4.8.26 Tabela Ver Mais Tarde

Nome do campo	Tipo de dados	Tamanho
id(PK)	inteiro	11
id_user(FK)	inteiro	11
Contente	varchar	30
id_content	inteiro	11
add_at	datetime	
Status	tinyint	1

Tabela 27: Ver Mais Tarde

5 APRESENTAÇÃO DAS APLICAÇÕES DESENVOLVIDAS E TESTES

Nesta seção serão apresentadas algumas páginas das aplicações *web* e *Android TV* acompanhados de uma breve descrição para uma melhor compreensão dos mesmos, no final serão apresentados os resultados obtidos mediante a aplicação de testes de desempenho das aplicações.

5.1 Aplicação *Web*

5.1.1 Página Principal

Quando o utilizador acede o site do *YetuYaliy* é direcionado para a página principal da aplicação, que nada mais é do que um resumo do que pode ser visto e feito no mesmo, conforme ilustrados nas figuras 16 e 17.

No menu principal é possível aceder as páginas *TV Lives*, *Movies*, *Series*, *Distributors*. Ao clicar nas "reticências" é mostrado mais opções tais como: *Sign Up Free*, *About*, *Contacts*, conforme ilustrado na figura 18.

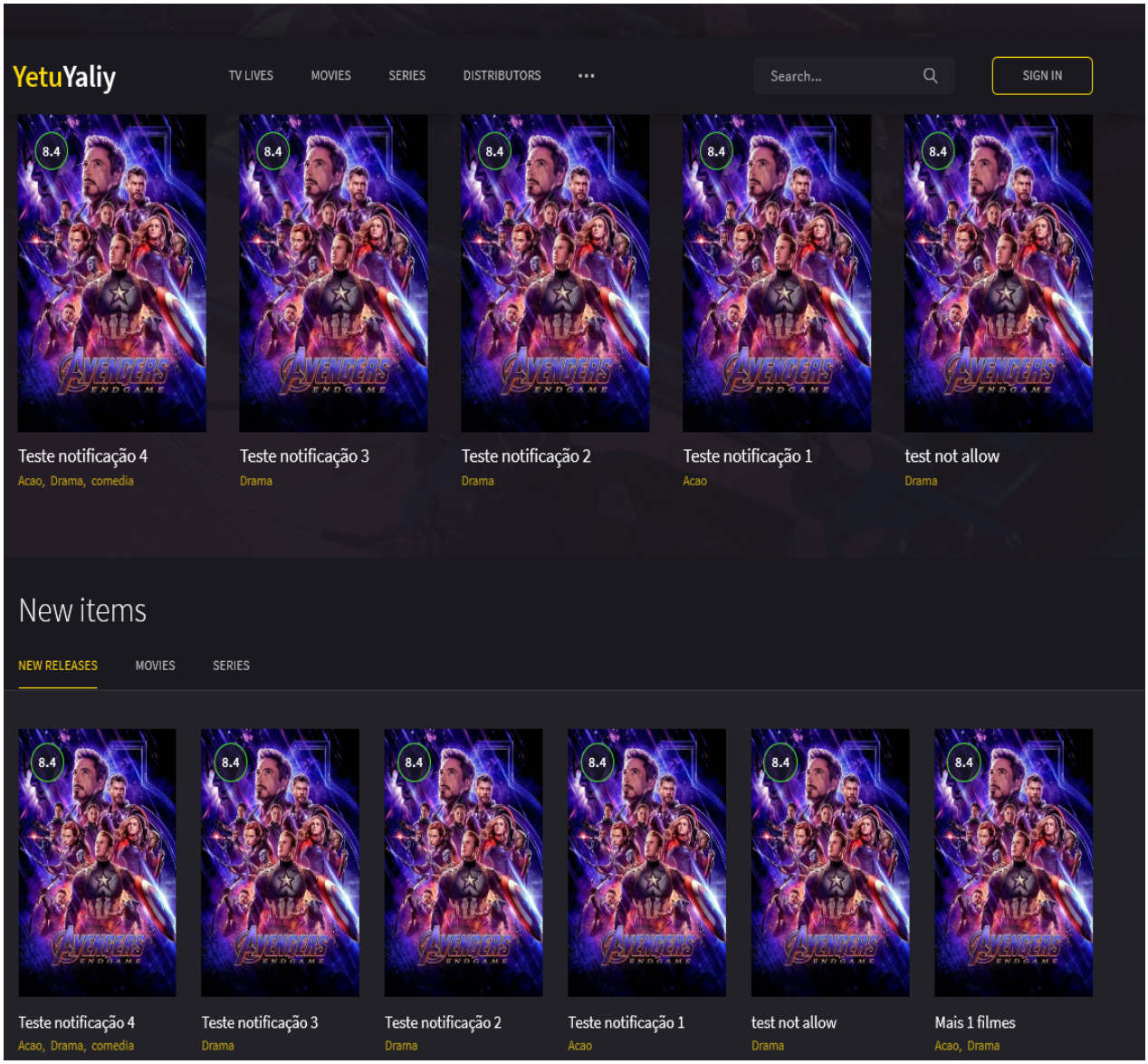

Figura 16: Página Principal I

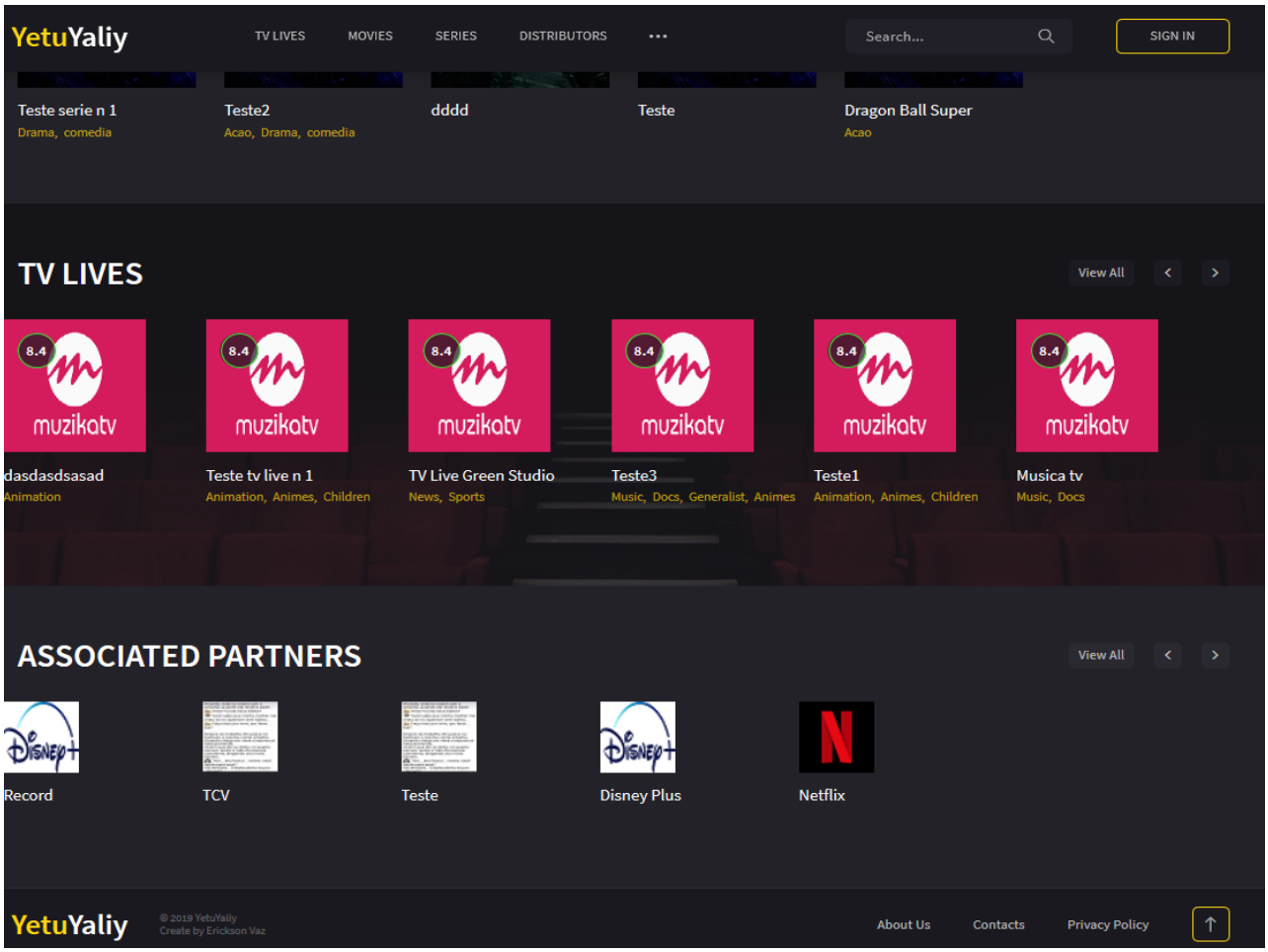

Figura 17: Página Principal II

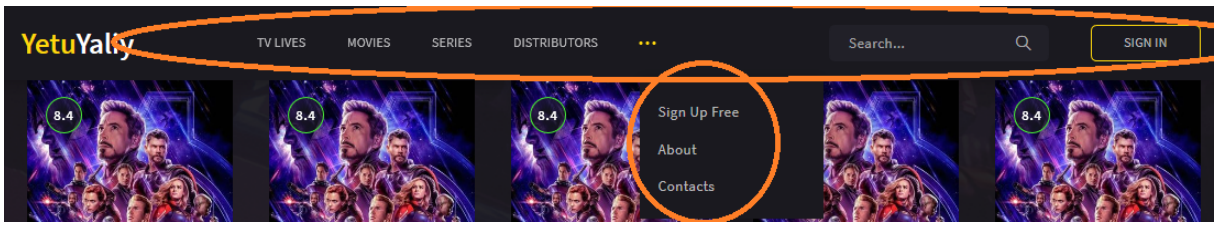

Figura 18: Menu Principal da Página *Web*

5.1.2 Página *TV Lives*

Nesta pagina é possível listar todos os canais de TV disponíveis para serem assistidos online quer na pagina *web*, quer no aplicativo *Android TV* desde de que o utilizador *logado* tenha permissão para tal. Também nesta página é possível fazer pesquisas por nome e filtragem dos canais por categoria, conforme ilustrado na figura 19.

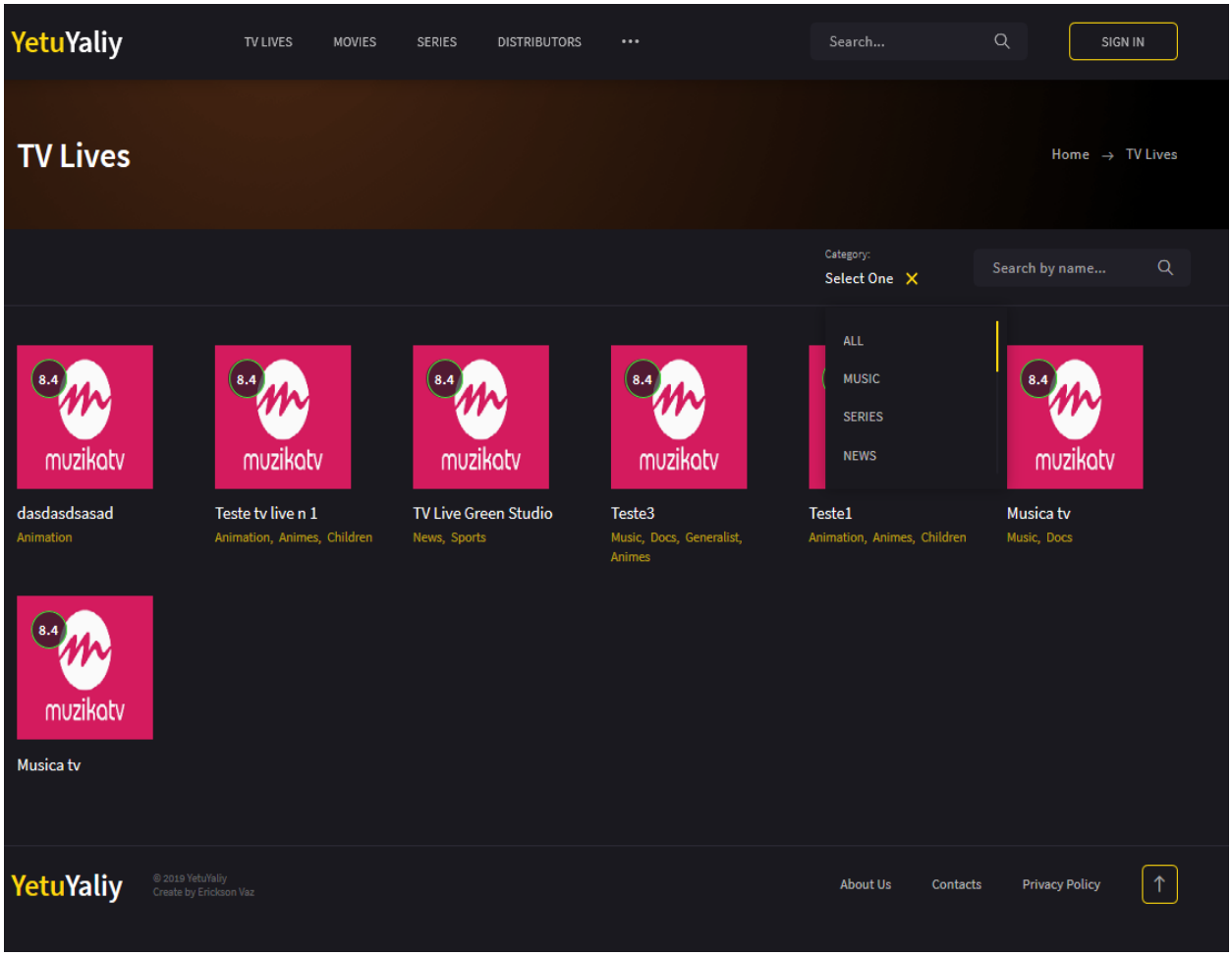

Figura 19: Página *TV Lives*

5.1.3 Página *Movies*

Nesta página encontram-se listados todos os filmes disponíveis para serem assistidos nas aplicações *web* e *android* desde que, o utilizador tenha permissão para tal. Também aqui pode-se fazer pesquisas dos conteúdos por nome e efetuar a filtragem dos mesmos por género, na figura 20 temos uma visualização desta página.

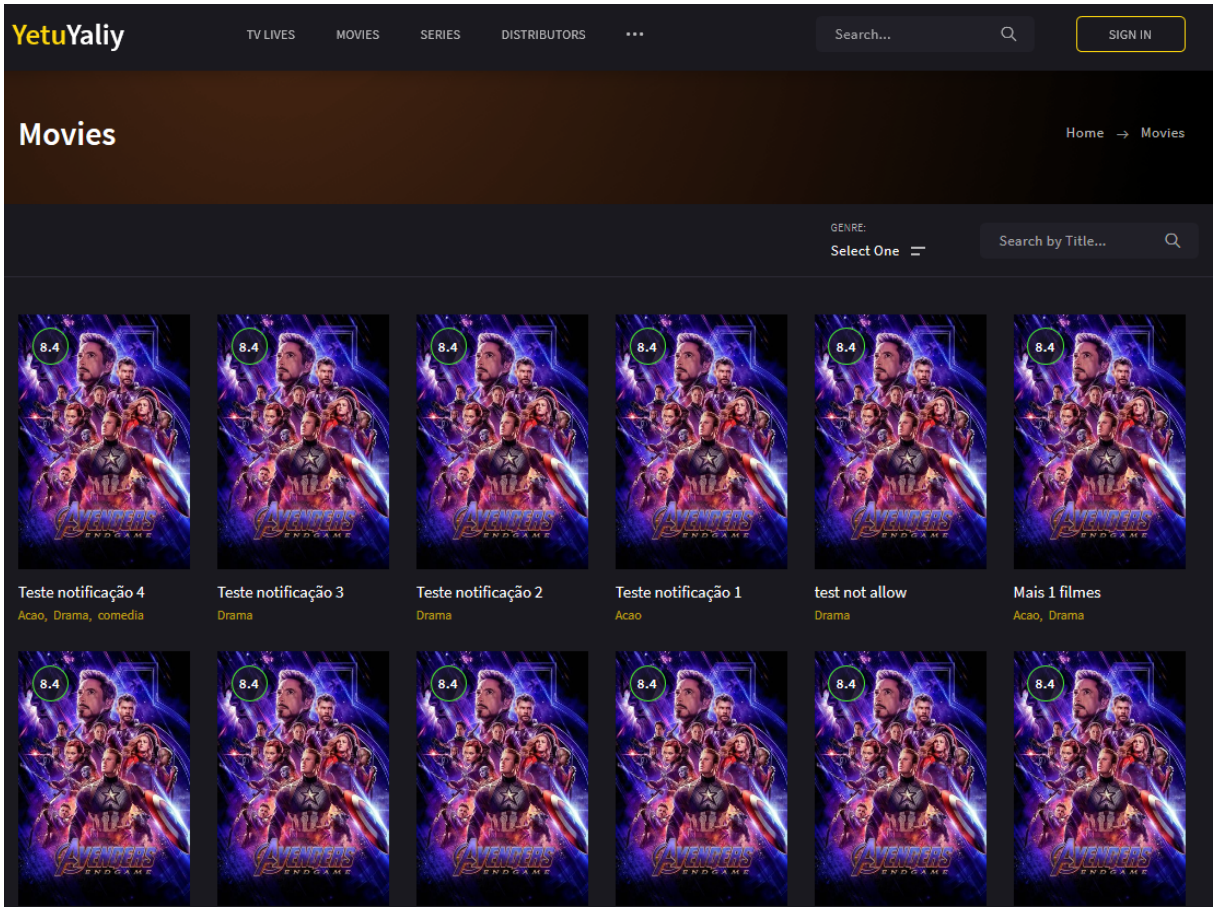

Figura 20: Página *Movies*

5.1.4 Página *Series*

Nesta página encontram-se listados todos as séries disponíveis para serem assistidas nas aplicações *web* e *android* desde que, o utilizador tenha permissão para tal. Aqui também pode-se fazer pesquisas das series por nome e efetuar a filtragem das mesmas por género. Na figura 21 temos uma visualização desta página.

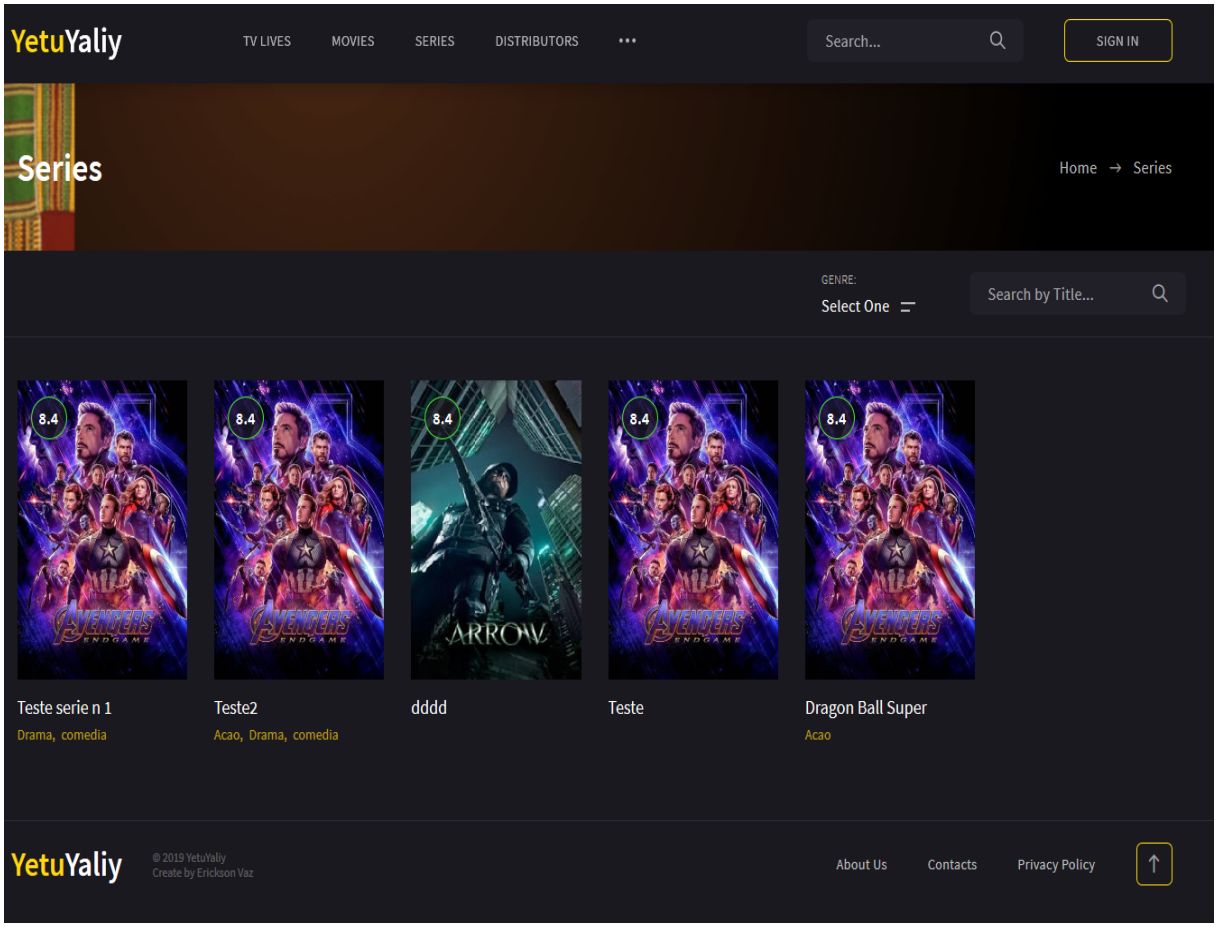

Figura 21: Página *Series*

5.1.5 Página *Distributors*

Nesta página encontram-se listadas todas as distribuidoras registradas no sistema, que no final das contas são elas quem disponibilizam filmes, séries e *tv lives*. Podem ser assistidos pelos utilizadores, aqui também é possível realizar pesquisas das distribuidoras por nomes, a figura 22 ajuda a entender melhor esta página.

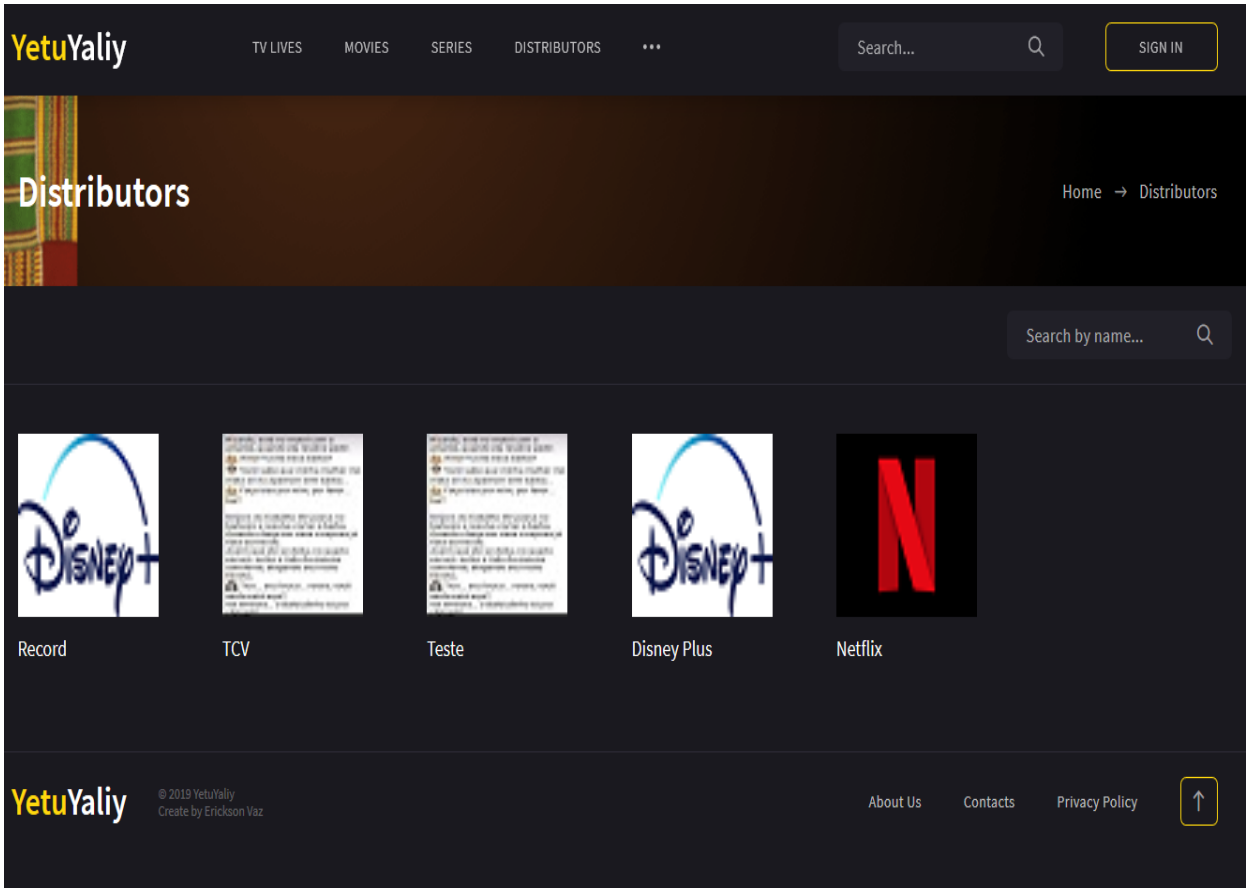

Figura 22: Página *Distributors*

5.1.6 Página *Signup free*

Quando o utilizador não estiver *logado* ele tem a possibilidade de inscrever-se na plataforma registrando uma conta *YetuYali*. Caso não tenha, este registro é feito na página *Signup Free*, informando o nome, *email* e *password* desejado, conforme ilustrado na figura 23.

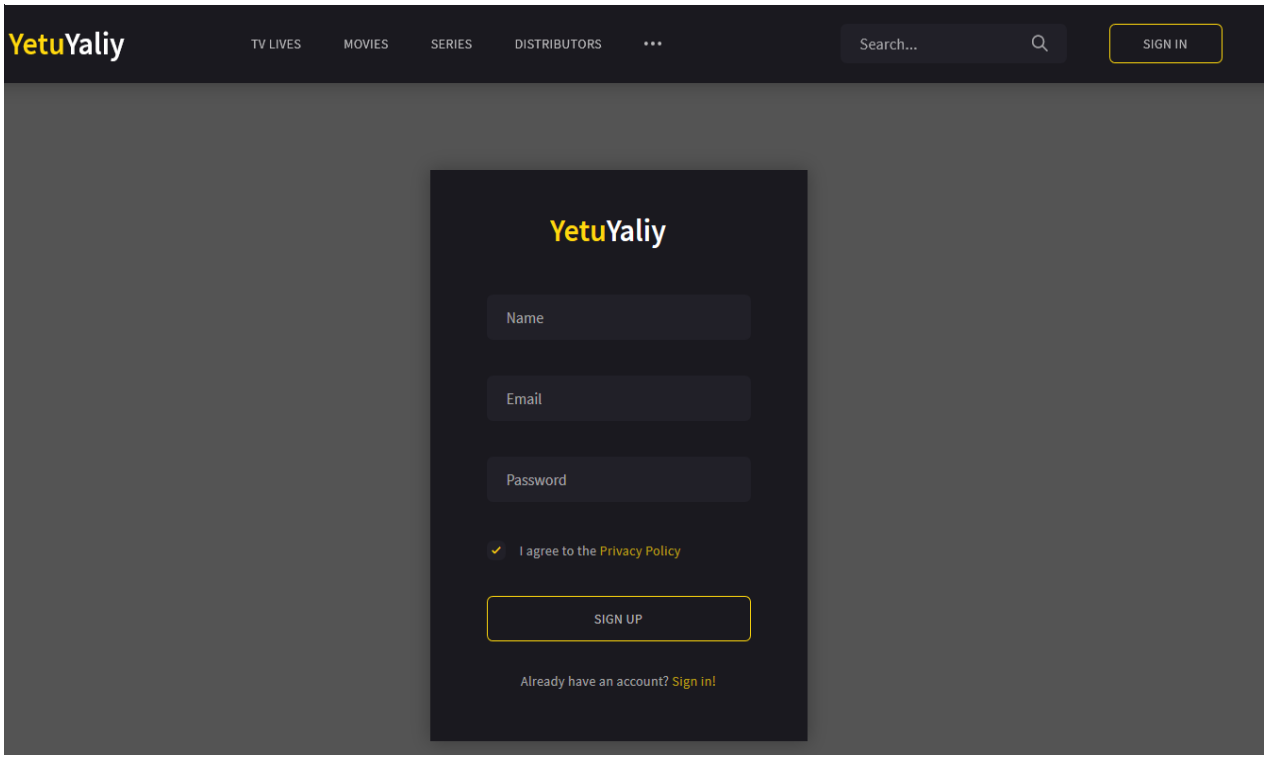

Figura 23: Página *Signup Free*

5.1.7 Página *Login*

Caso o utilizador já tenha uma conta *YetuYali* criada ele pode fazer o *login* através da página *login* informando o *e-mail* e o *password*, conforme ilustrado na figura 24.

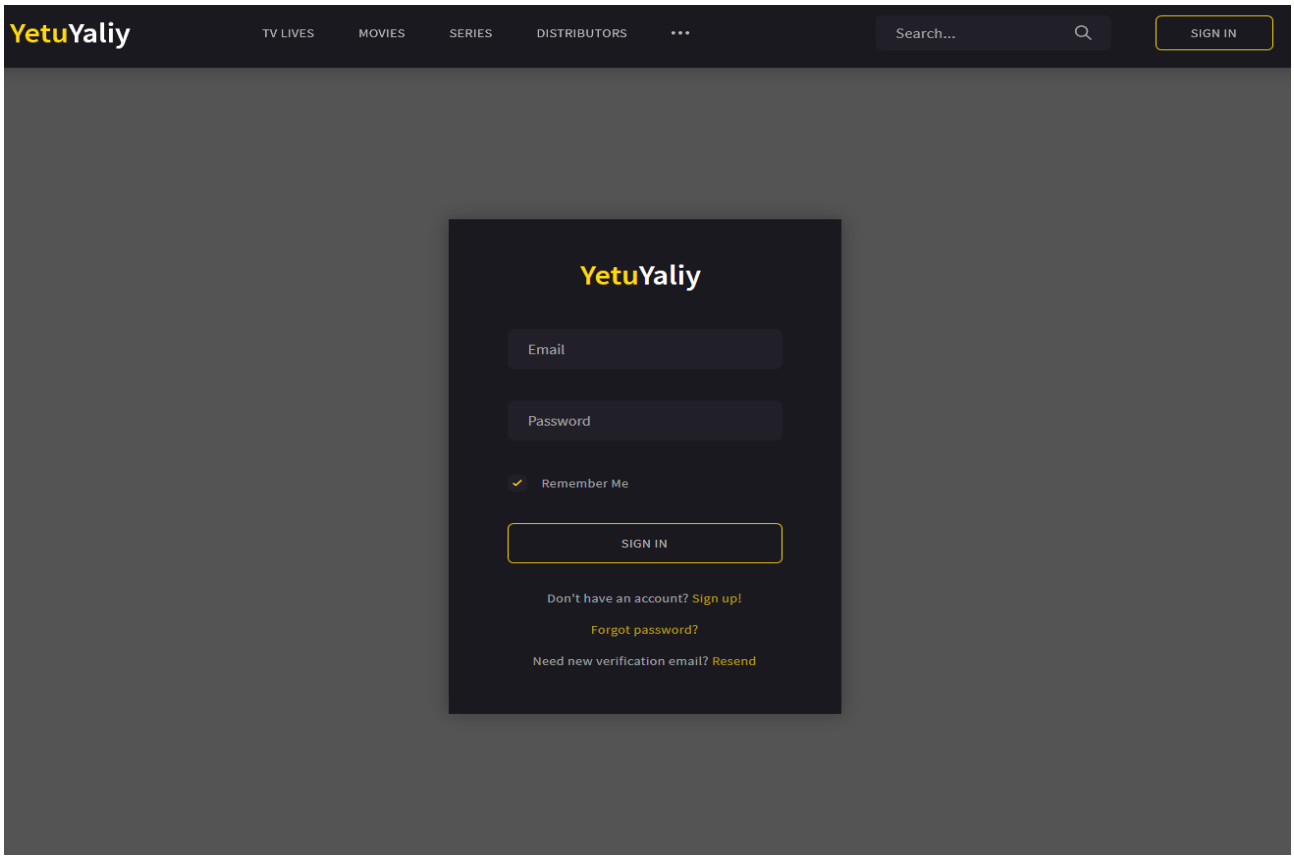

Figura 24: Página *Login*

5.1.8 Página *User Profile*

Quando um utilizador estiver *logado* e quiser atualizar informações do seu perfil poderá fazer através da opção *User Profile* que aparece no canto superior direito como mostrado na figura 25, na pagina de atualização o utilizador poderá alterar informações como foto de perfil, primeiro nome, ultimo nome, género, data de nascimento e país de residência conforme a figura 26 ilustra.

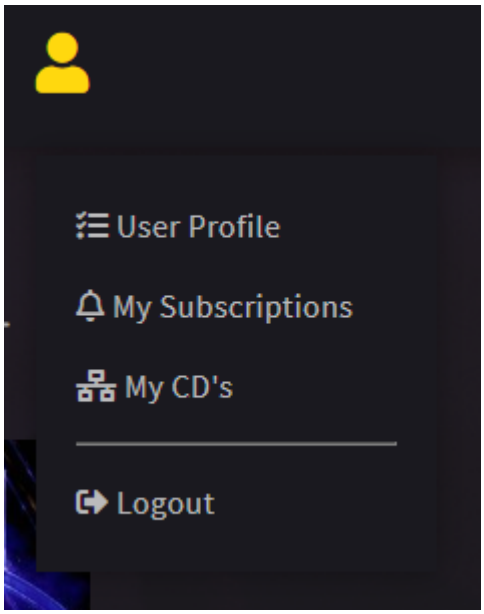

Figura 25: Menus de Acesso Opções do Utilizador

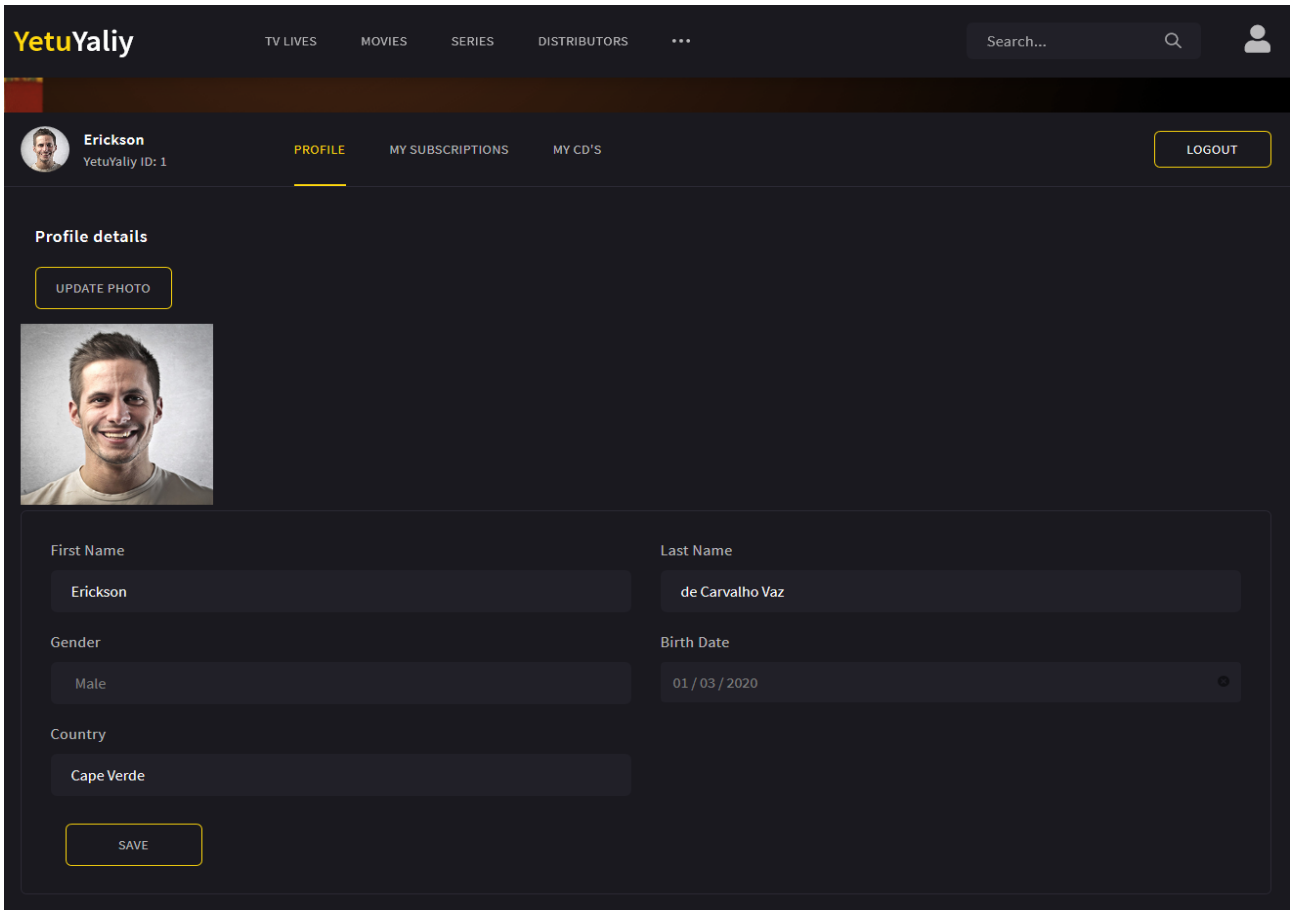

Figura 26: Página *User Profile*

5.1.9 Página *My Subscription*

Uma outra opção que o utilizador *logado* tem acesso é a página *My Subscription*, esta página permite que o utilizador veja a lista completa de distribuidoras na qual ele esta subscrito, exatamente como é mostrado na figura 27.

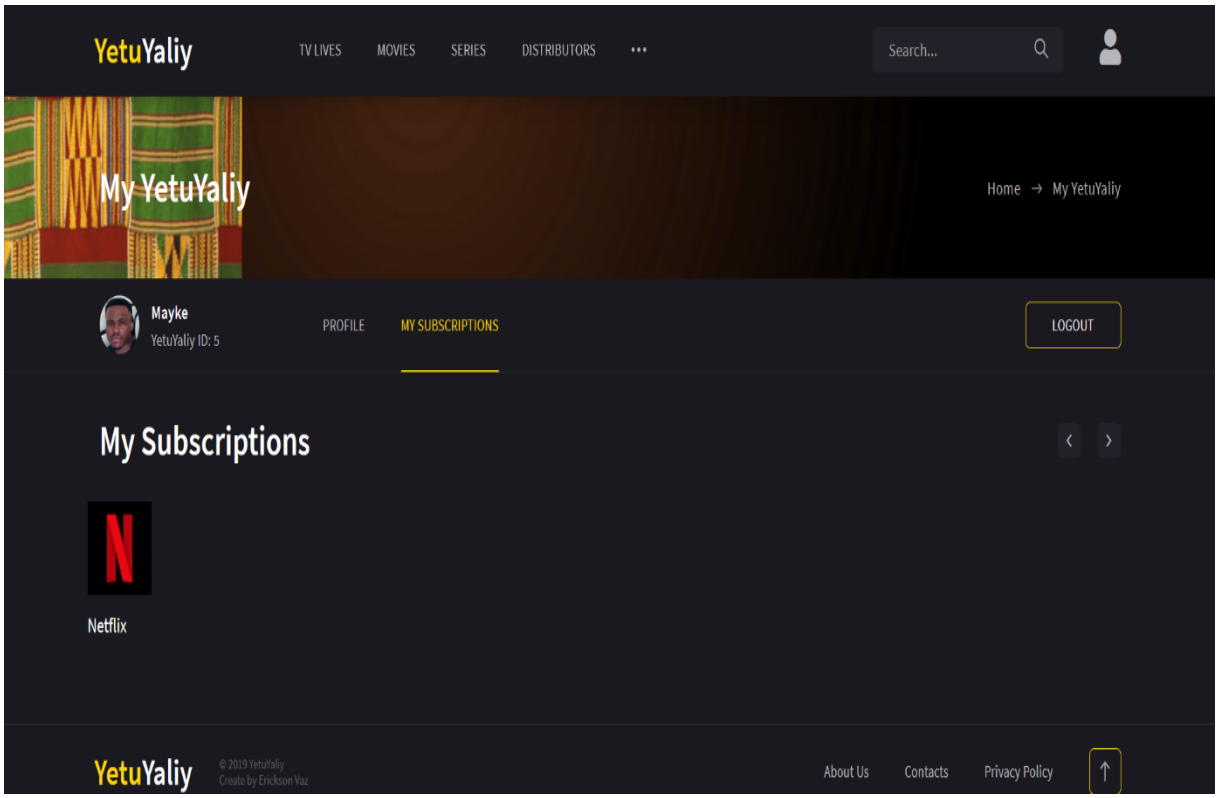

Figura 27: Página *My Subscriptions*

5.1.10 Página *My CD's(My Contents Distributors)*

Nesta página o utilizador *logado* pode criar e visualizar listas de distribuidoras na qual ele poderá aceder o painel de controle afim de adicionar novos filmes, séries, *tv lives* e executar outras tarefas de gerenciamento, também é possível fazer a gestão dos colaboradores adicionando, removendo ou bloqueando estes, a figura 28 ilustra a página.

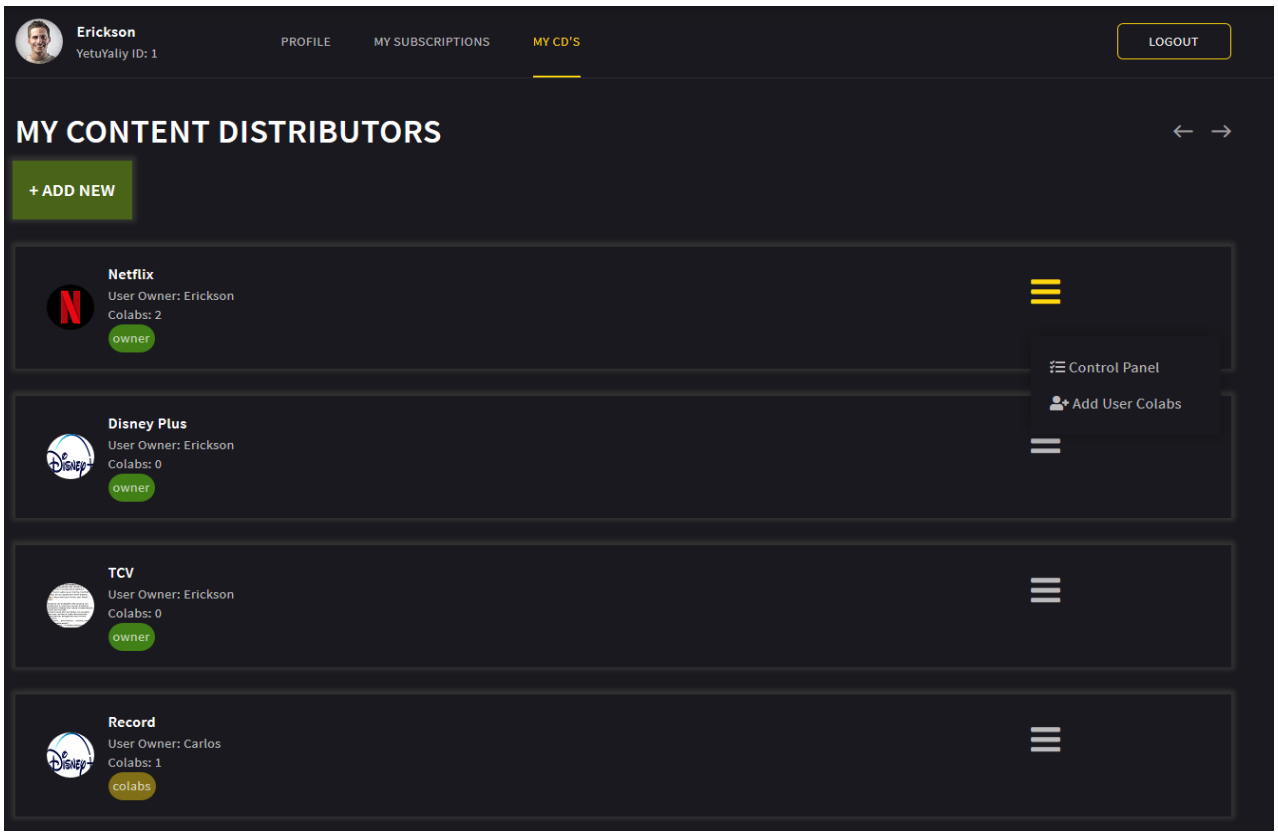

Figura 28: Página *My CD's*

5.1.11 Painel de Controle Distribuidora

Quanto um utilizador e parceiro da plataforma e possui pelo menos uma distribuidora criada ele tem acesso ao painel de controle ao clicar na opção *control panel*, ao entrar no painel de controle é direcionado primeiramente para o *dashboard* onde poderá ver um resumo numérico das atividades referentes a distribuidora conforme ilustrado na figura 29.

No menu lateral é apresentado uma lista de opções que o utilizador, neste caso o dono da distribuidora, poderá realizar nomeadamente registro de Filmes, Séries *TV Lives*, controle de comentários, controle de utilizadores (Colaboradores da distribuidora), controlar os custos de subscrições na distribuidora, controlar as finanças da distribuidora, controlar o recebimento de notificações referentes a distribuidora e os *logs*, conforme pode ser visto na figura 30.

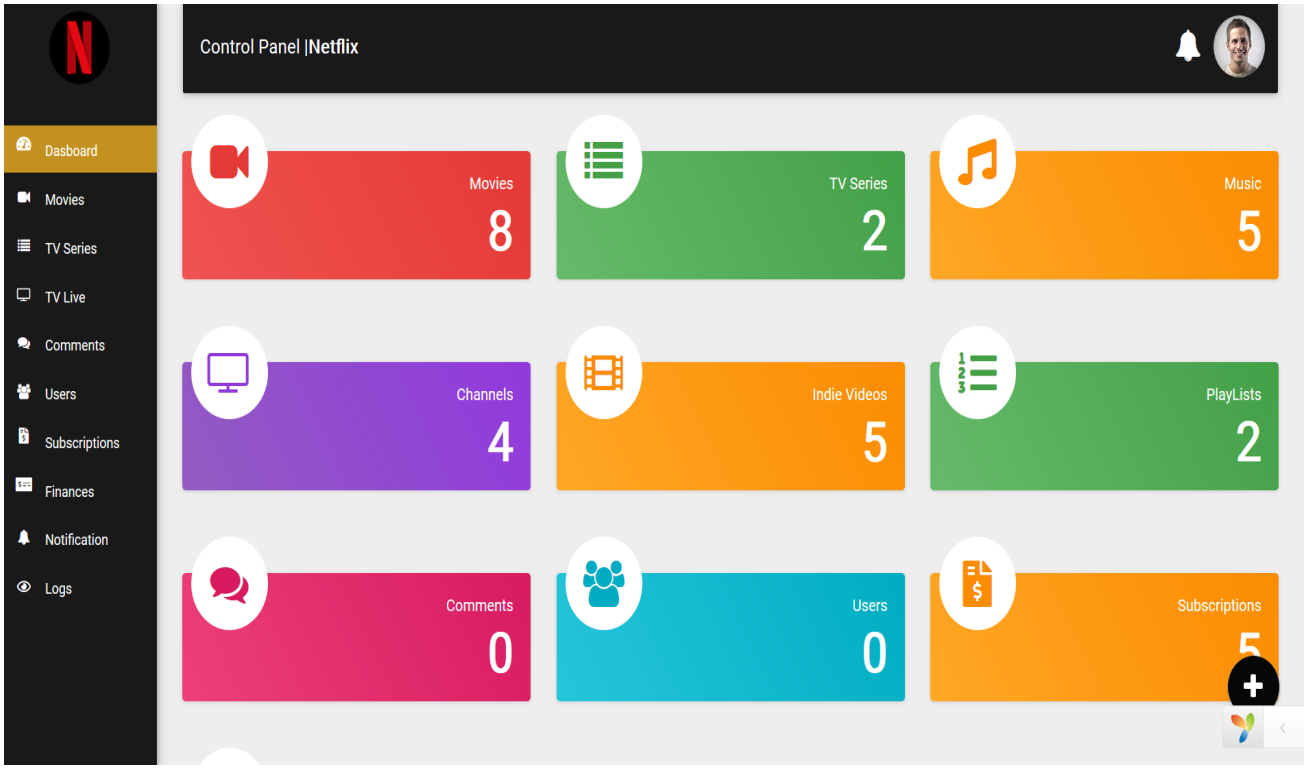

Figura 29: Painel de Controle

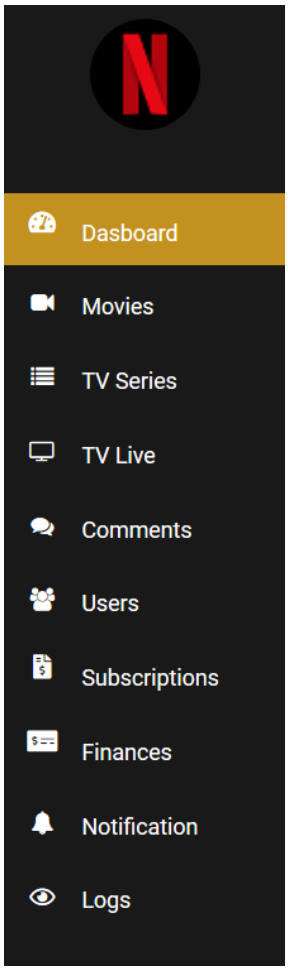

Figura 30: Menu Lateral Painel de Controle

5.1.12 Página *Manage Movies*

Nesta página o utilizador poderá ver uma lista completa dos filmes já registrados na distribuidora, além também, de poder adicionar novos fazendo com que ela fique pública e acessível a todos que queiram assistir. Nesta página é possível proceder a edição das informações de cada filme já registrado, e também mudar o status de visibilidade de cada um, a figura 31 ajuda a ilustrar esta página.

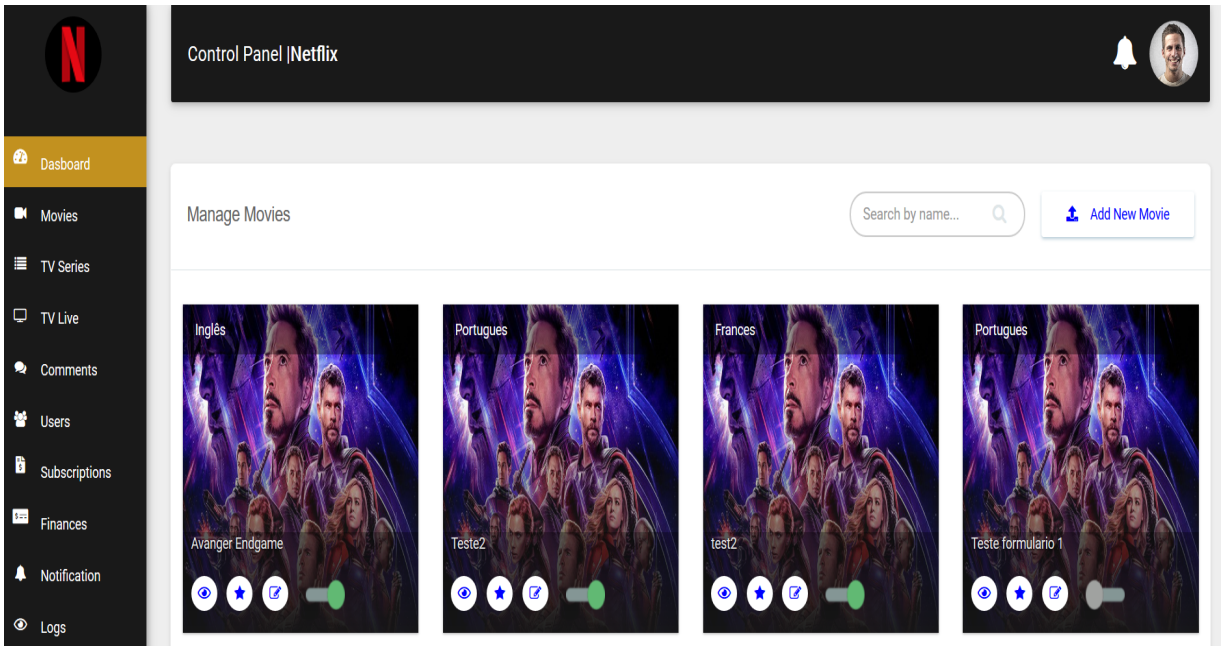

Figura 31: Página *Manage Movies*

5.1.13 Página *Manage Series*

Está pagina é semelhante a página anterior, possui as mesmas funcionalidades só que voltada para a gestão de séries, temporadas e episódios das distribuidoras conforme mostrada na figura 32.

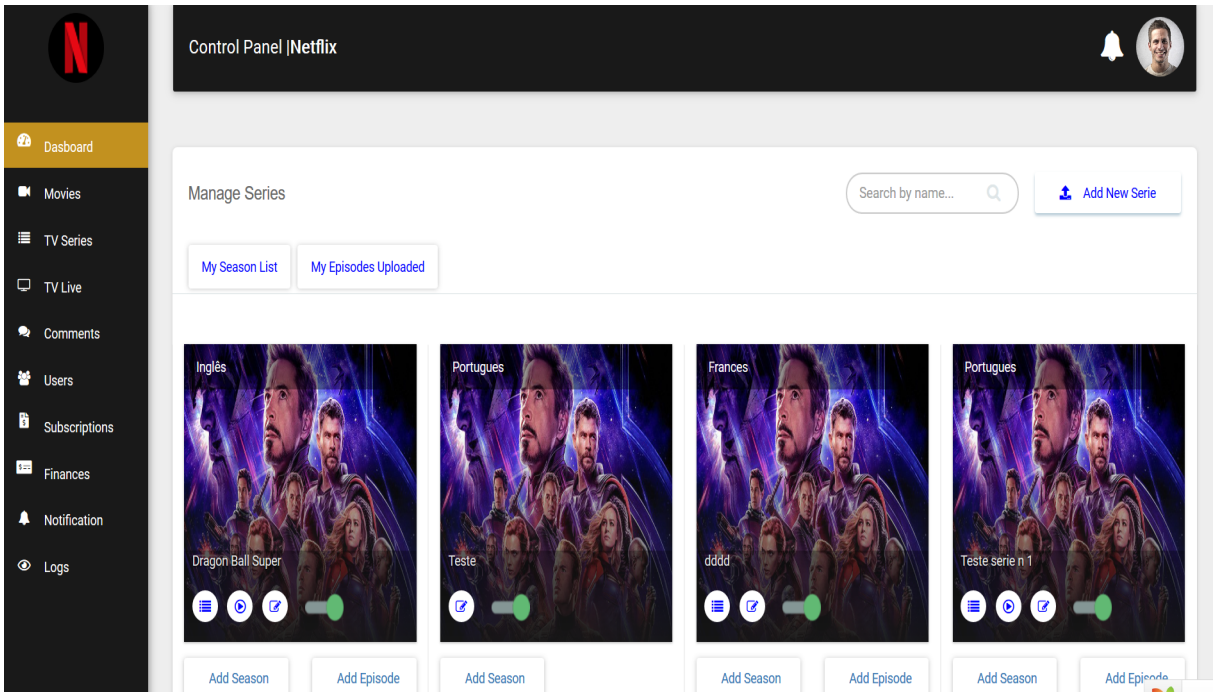

Figura 32: Página *Manage Series*

5.1.14 Página *Manage TV Lives*

Está página é semelhante a página anterior, possui as mesmas funcionalidades só que aqui é voltada para a gestão de *TV Lives* das distribuidoras conforme mostrada na figura 33.

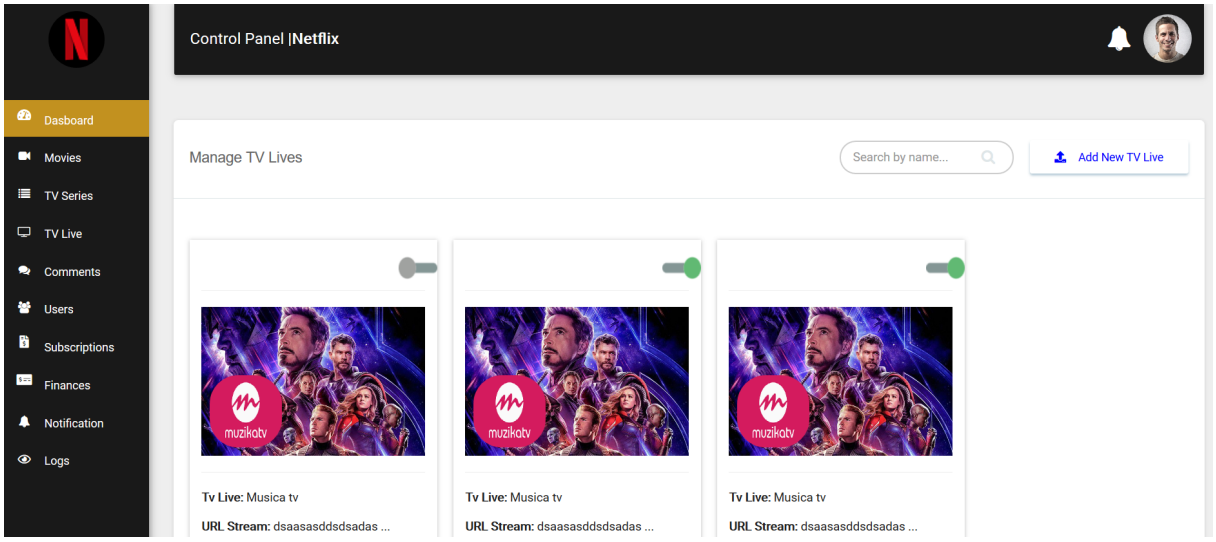

Figura 33: Página *Manage TV Lives*

5.1.15 Página *Manage Comments*

Nesta pagina é possível gerenciar todos os comentários feitos nos diferentes tipos de conteúdos pertencentes a uma distribuidora conforme ilustrado na figura 34. Neste painel o utilizador poderá configurar explicitamente se a distribuidora irá aceitar comentários, se o responsável pela distribuidora terá que aprovar quais quer comentários feitos antes de se tornarem públicos.

Caso houver vários comentários listados é possível inclusive filtrar os mesmos e até eliminar algum caso houver necessidade.

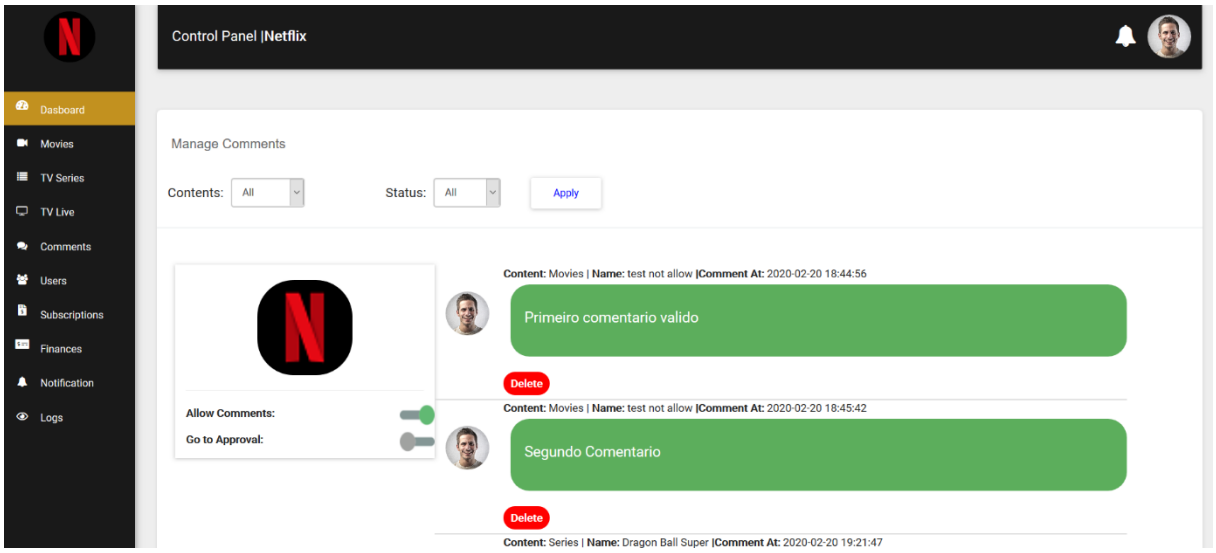

Figura 34: Página *Manage Comments*

5.1.16 Página *Manage Users*

Nesta página será onde o responsável pela distribuidora poderá ver a lista de todos os outros utilizadores que são colaboradores na distribuidora. Pode-se inclusive adicionar um novo, bloquear acesso, editar privilégios de acesso e ver mais informações, esta página pode ser vista na figura 35.

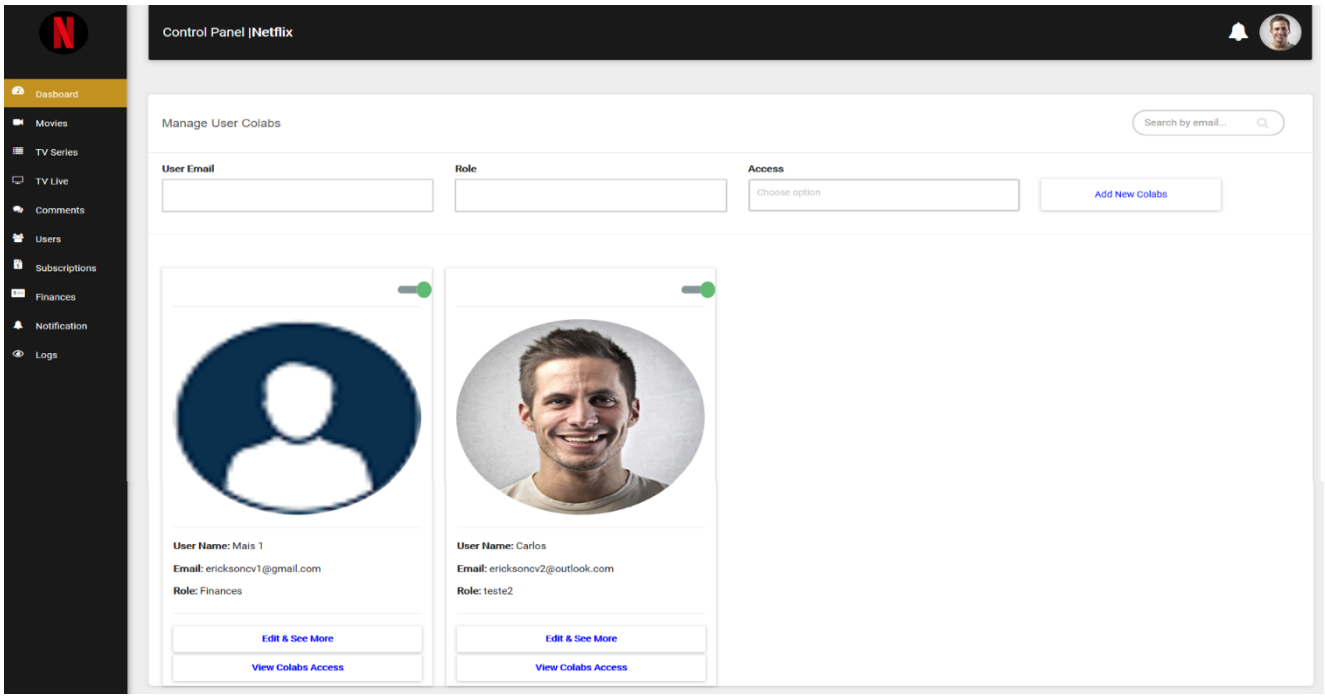

Figura 35: Página *Manage Users*

5.1.17 Página *Manage Subscriptions*

Esta página é onde os responsáveis pela distribuidora poderão gerenciar e determinar os custos relacionados a subscrição da mesma, aqui o responsável indicará primeiramente se a distribuidora é paga ou gratuita e qual é o preço base de subscrição, ou seja, o preço mensal. Os restantes valores serão calculados mediante uma fórmula, para determinar os custos semestral e anual, a figura 36 ajuda a entender melhor está página.

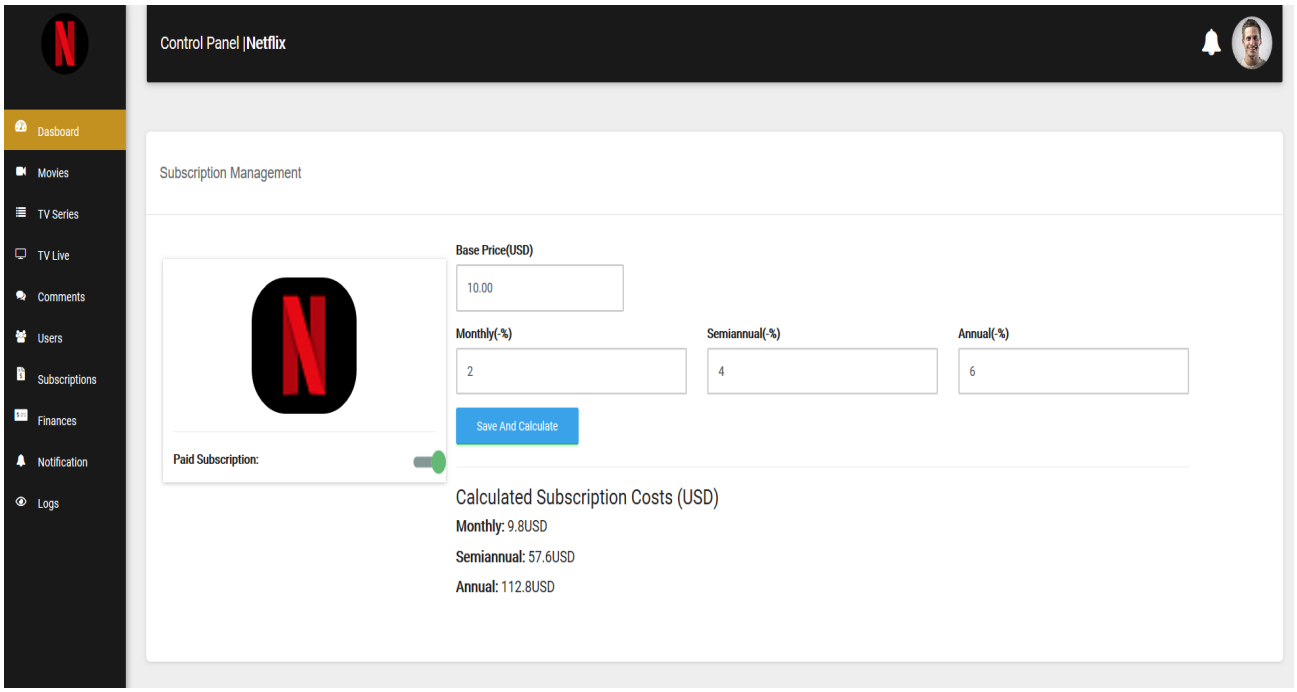

Figura 36: Página *Manage Subscription*

5.1.18 Página *Manage Notification*

Nesta página o dono da distribuidora ou um colaborador com permissão para tal, poderá configurar exatamente quais notificações pretende receber quando alguma alteração for feita em cima da distribuidora em questão ou em cima de um dos conteúdos da mesma, as notificações que o utilizador escolhe receber são:

- Notificar quando novos filmes forem criados por um colaborador.
- Notificar quando uma nova série for criada por um colaborador.
- Notificar quando novos canais de *TV Live* forem criados por um colaborador.
- Notificar quando um novo comentário for adicionado a um conteúdo da distribuidora.

Também esta página serve para listar todas as notificações recebidas, pode-se filtrar estas notificação para facilitar a gestão das mesmas, a figura 37 ajuda a ilustrar melhor esta página.

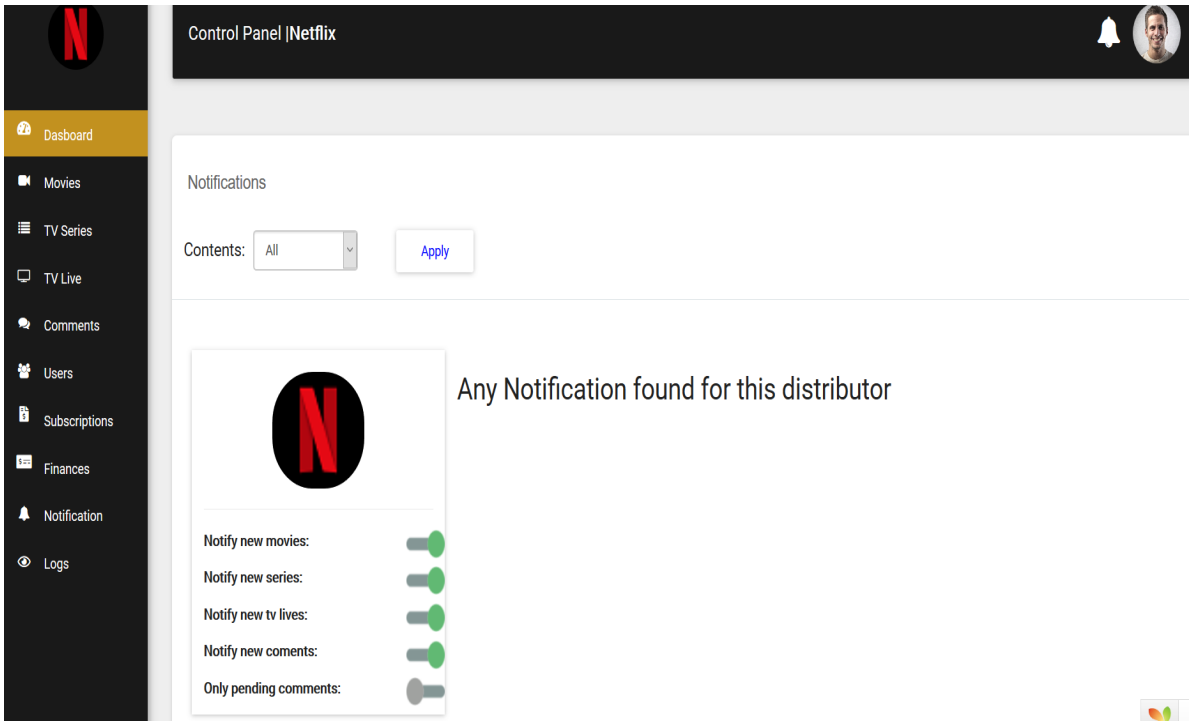

Figura 37: Página *Manage Notification*

5.1.19 Página Manage *Logs*

Nesta página é apresentada principalmente, para o dono da distribuidora um conjunto de informações relacionadas ao acesso dos colaboradores e ações realizadas pelos mesmos dentro da distribuidora, tornando assim possível um maior controle de acesso por parte dos responsáveis pela distribuidora. Assim, como acontece na página notificação, aqui também é possível determinar se deseja ou não receber tais informações, neste caso registrar o *log*, a figura 38 apresenta esta página.

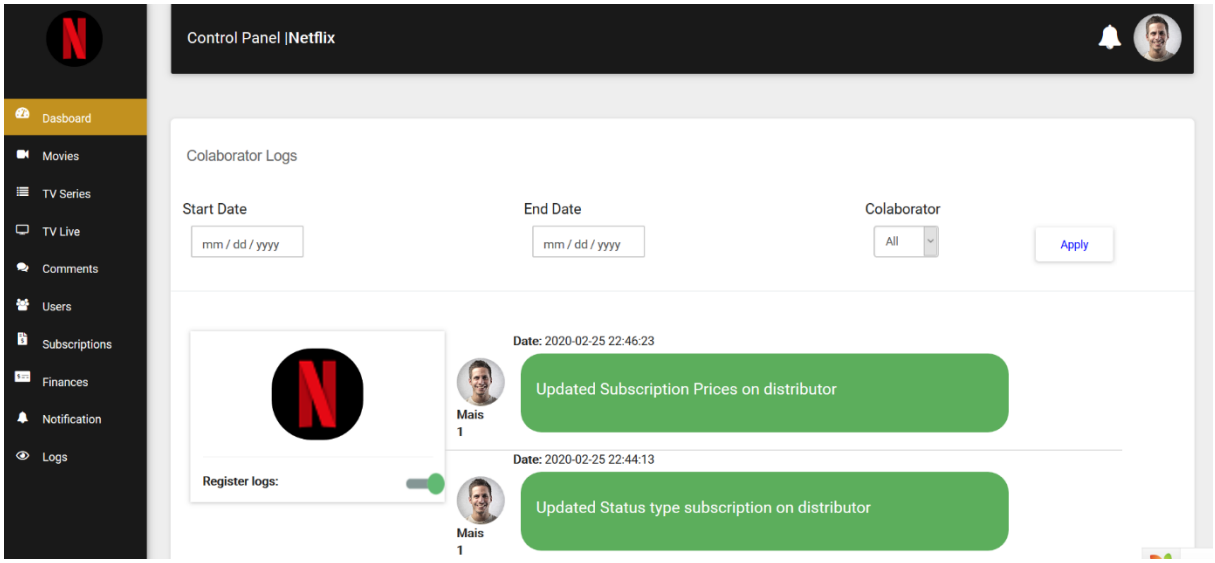

Figura 38:Página *Manage Logs*

5.2 Aplicação *Android TV*

Pensando numa maior acessibilidade e mobilidade por parte dos utilizadores da plataforma *YetuYaliy*, foi concebido uma aplicação para *Android TV* que nada mais é do que uma extensão da aplicação *web*. Nela o utilizador *logado* poderá aceder a todos os conteúdos disponíveis inclusive visualizar filmes e séries, e também a possibilidade de se subscrever nas distribuidoras.

Uma outra característica importante desta aplicação é que ela pode funcionar como um *launcher* (aplicação inicial) no dispositivo onde for instalado, devido a isso foi levado em conta no seu desenvolvimento todos os aspetos relacionados ao acesso dos recurso do sistema operativo *android*, como acesso a *playstore*, acesso a lista de *app* instalados, acesso a definições do sistema, porém a proposta inicial é de que ela irá rodar no *SBC Khadas VIM2*, que entre outras várias funções, ela permite a incorporação de um descodificador de sinal digital,

permitindo que a aplicação *android* do *YetuYaliy* tenha acesso a sinal de TV digital em uma determinada região geográfica.

De seguida serão apresentadas as principais telas da aplicação *Android TV* da plataforma *YetuYaliy*.

5.2.1 Tela de *Login*

Quando o utilizador aceder a aplicação pela primeira vez e tiver acesso a *internet* é lhe apresentado a tela de login, onde o mesmo deverá informar as suas credenciais de acesso (*e-mail* e *password*), essas credenciais deverão ser as mesmas utilizadas para aceder a plataforma *web* do *YetuYaliy*, a figura 39 ajuda a ilustrar melhor está tela.

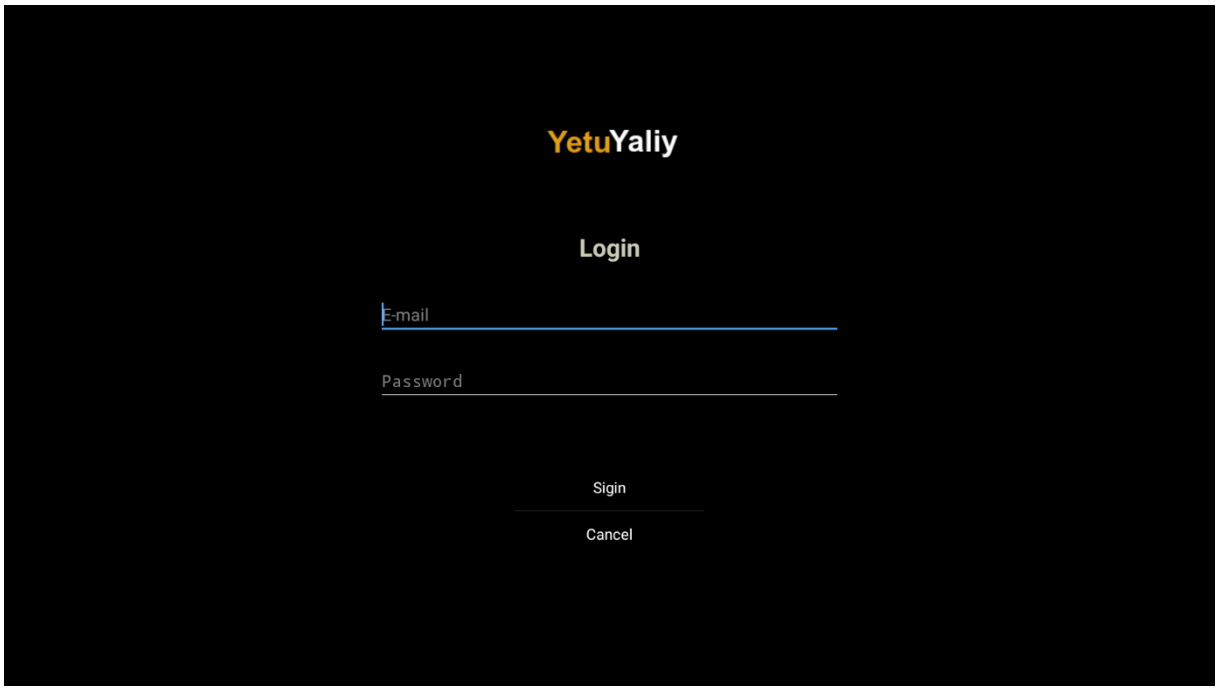

Figura 39: Tela *Login* Aplicação *Android*

5.2.2 Menu Lateral Aplicação *YetuYaliy*

Quando o utilizador efetuar com sucesso o *login* na aplicação, é direcionado para a tela inicial da aplicação, no lado esquerdo o utilizador tem o menu lateral retráctil onde poderá aceder as principais páginas da aplicação tais como:

- *Home*
- *Movies*
- *Series*
- *TV Lives*
- *Distributors*
- *My Contents*
- *Apps*
- *Settings*

A figura 40 ajuda a visualizar melhor este menu.

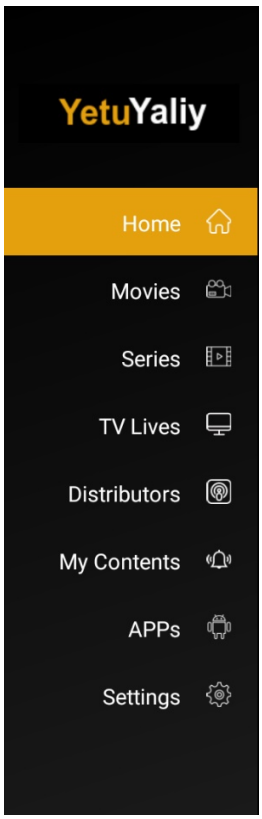

Figura 40: Menu Lateral Aplicação *Android TV*

5.2.3 Página *Home*

Esta página é um resumo de tudo o que é possível fazer-se dentro da aplicação *Android TV*, nela temos no topo um conjunto de opções de acesso rápido a recursos do sistema *android* e algumas funcionalidades da aplicação *YetuYaliy* como:

- Acesso a aplicação de tv digital
- Acesso ao *player* de media do sistema *android*
- Acesso à *playstore*
- Acesso a opções de conta do utilizador *logado*
- Acesso a opções de *updates* da aplicação *YetuYaliy*

Nela também é possível ver os conteúdos mais recentes disponibilizados na plataforma como os filmes, séries, *tv lives* e distribuidoras adicionadas. A figura 41 ilustra está tela.

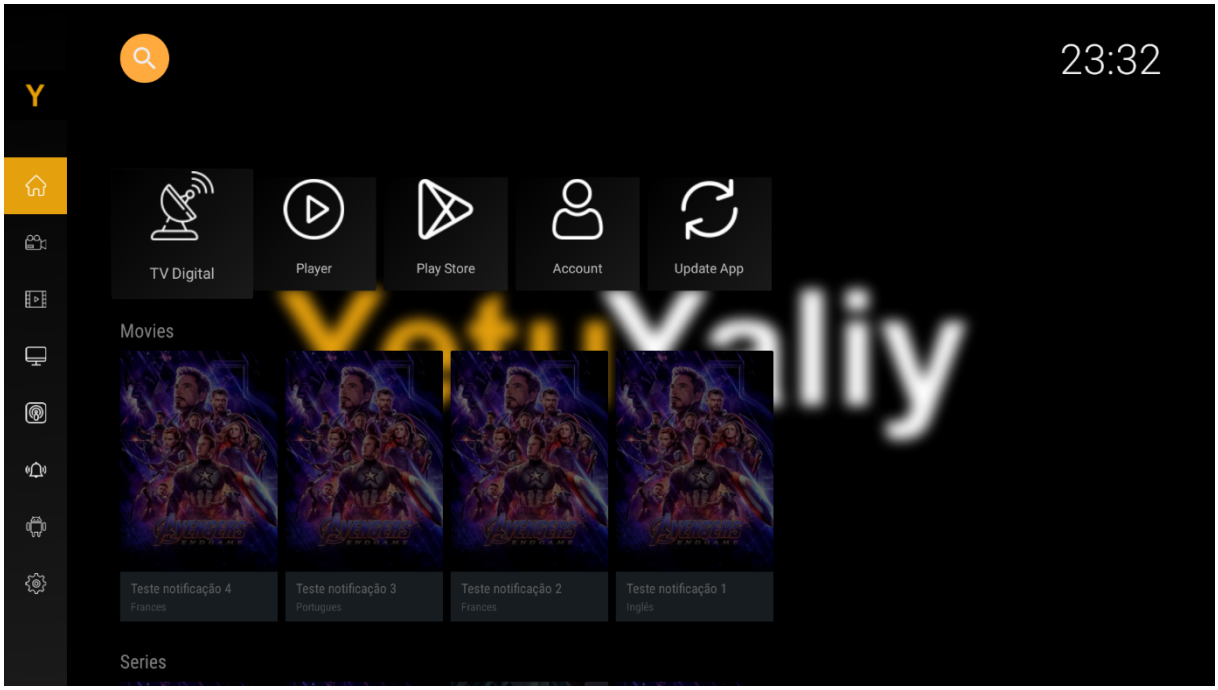

Figura 41: Tela *Home* (Opções de acesso rápido)

5.2.4 Tela *Movies*

Nesta tela o utilizador poderá encontrar todos os filmes disponibilizados na plataforma, logo que entrar nesta página é apresentado uma lista dos nove últimos filmes adicionados na plataforma, porém, se o utilizador preferir uma busca mais precisa poderá aceder as opções de filtro que aparecem no topo da página, estas opções permitem filtrar de forma alfabética os filmes por género. A figura 42 ajuda a visualizar melhor uma dessas telas.

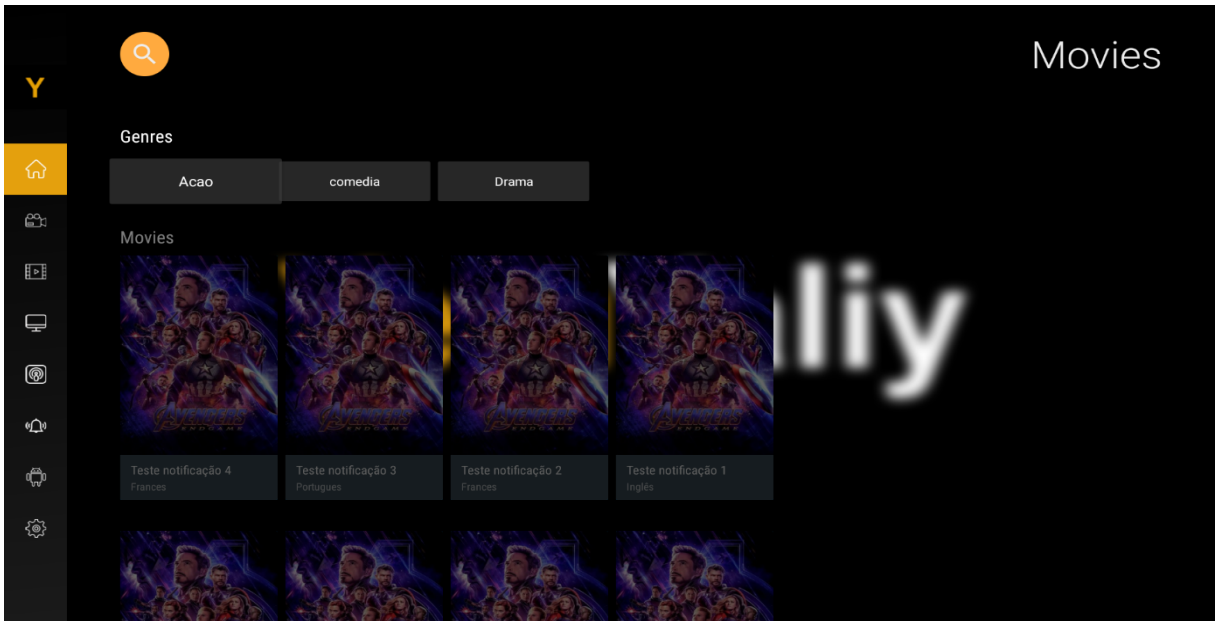

Figura 42: Tela Listar Filmes

5.2.5 Tela Series

Nesta tela é listada para o utilizador um conjunto das últimas nove séries registradas na plataforma, aqui também se o utilizador preferir poderá aceder ao filtro no top da página, podendo buscar as séries por género e ordenado alfabeticamente conforme mostrado na figura 43.

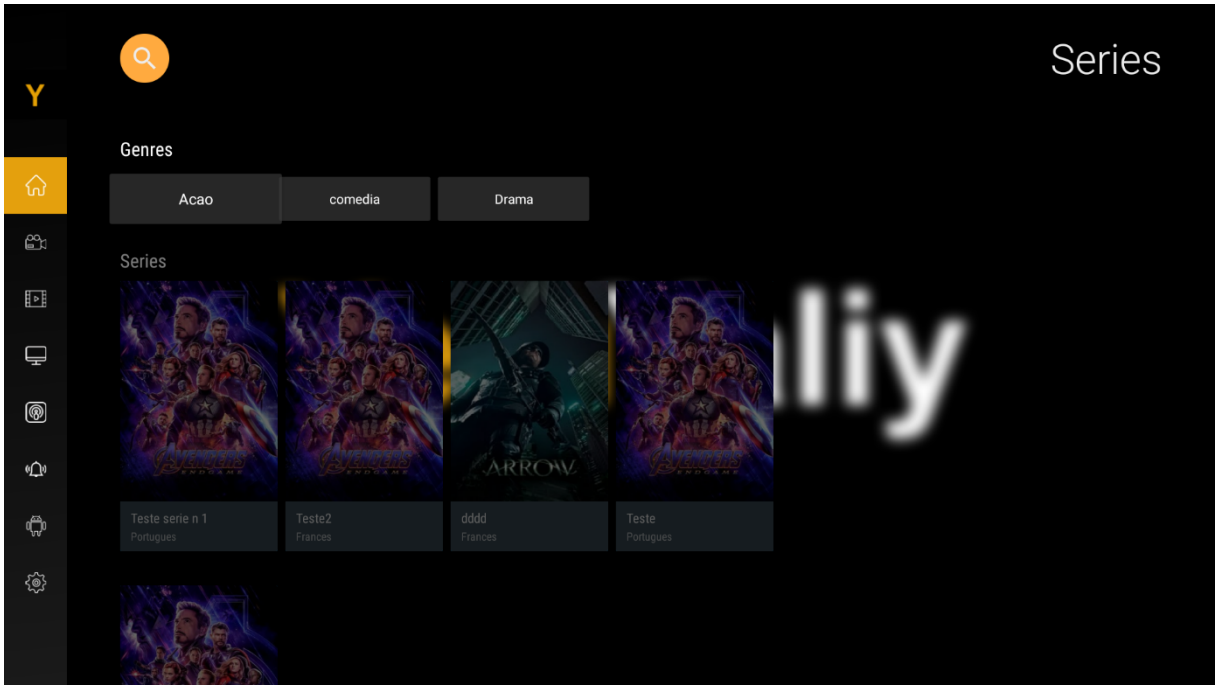

Figura 43: Lista de *Series*

5.2.6 Tela *TV Lives*

Nesta tela é possível ver uma lista dos 9 últimos canais de *tv lives* adicionados na plataforma, porém se o utilizador preferir uma busca mais detalhada tem no topo um conjunto de categorias que permite filtrar os canais de *live* e mostrar por ordem alfabética conforme pode ser visto na figura 44.

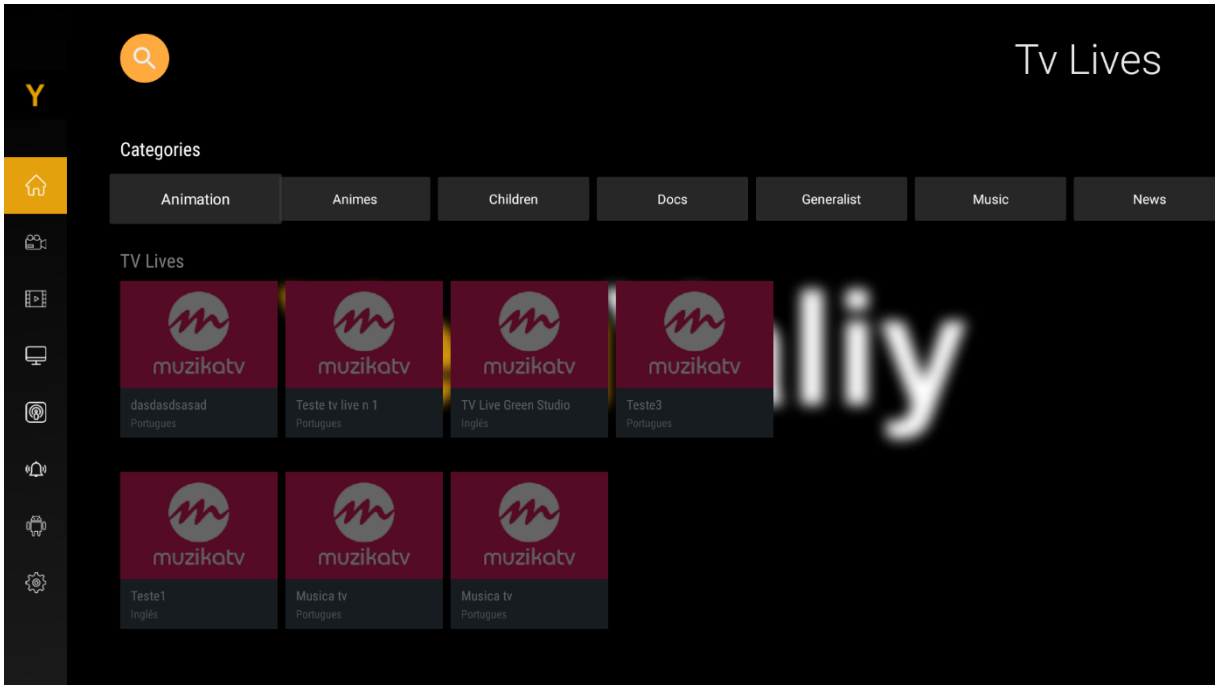

Figura 44: Tela *TV Lives*

5.2.7 Tela *Distributors*

Nesta tela é possível listar as principais distribuidoras que se encontram registradas na plataforma, as distribuidoras dentro do *YetuYaliy* desempenham um papel importante já que todos os conteúdos filmes, series, *tv lives* estão ligados a uma, já que são elas as fontes de origem de todas, a figura 45 mostra esta pagina.

Figura 45: Tela *Distributors*

5.2.8 Tela *My Contents*

Esta tela dá ao utilizador a opção de visualizar de forma rápida e completa todos os conteúdos disponíveis na plataforma *YetuYaliy* que tem total acesso, quer por ser dono ou um dos colaboradores da distribuidora que contem o conteúdo, ou quer seja pelo facto de ter comprado alguma subscrição na distribuidora que disponibiliza o conteúdo, no topo da pagina o utilizador tem a opção de filtrar os conteúdos por tipo, a figura 46 ilustra está tela.

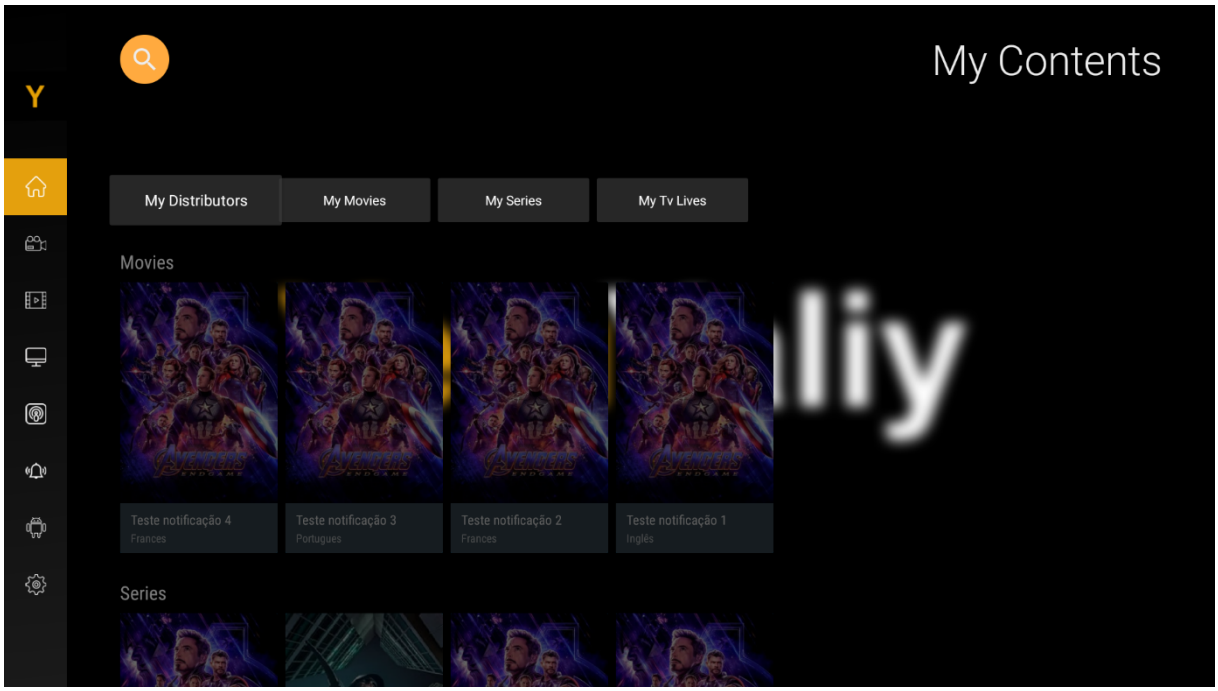

Figura 46: Tela *My Contents*

5.2.9 Tela *Apps*

Uma das características desta aplicação é o facto dela também poder funcionar como um *launcher*, isto é , uma aplicação de tela inicial do *android*, sendo assim é necessário prover algumas funcionalidades de acesso a recursos do sistema operacional, um destes recursos é a lista de aplicativos instalados, sendo assim foi incluído nesta aplicação uma pagina que permite que utilizadores possam aceder e executar seus *apps* favoritos, a figura 47 ilustra ela.

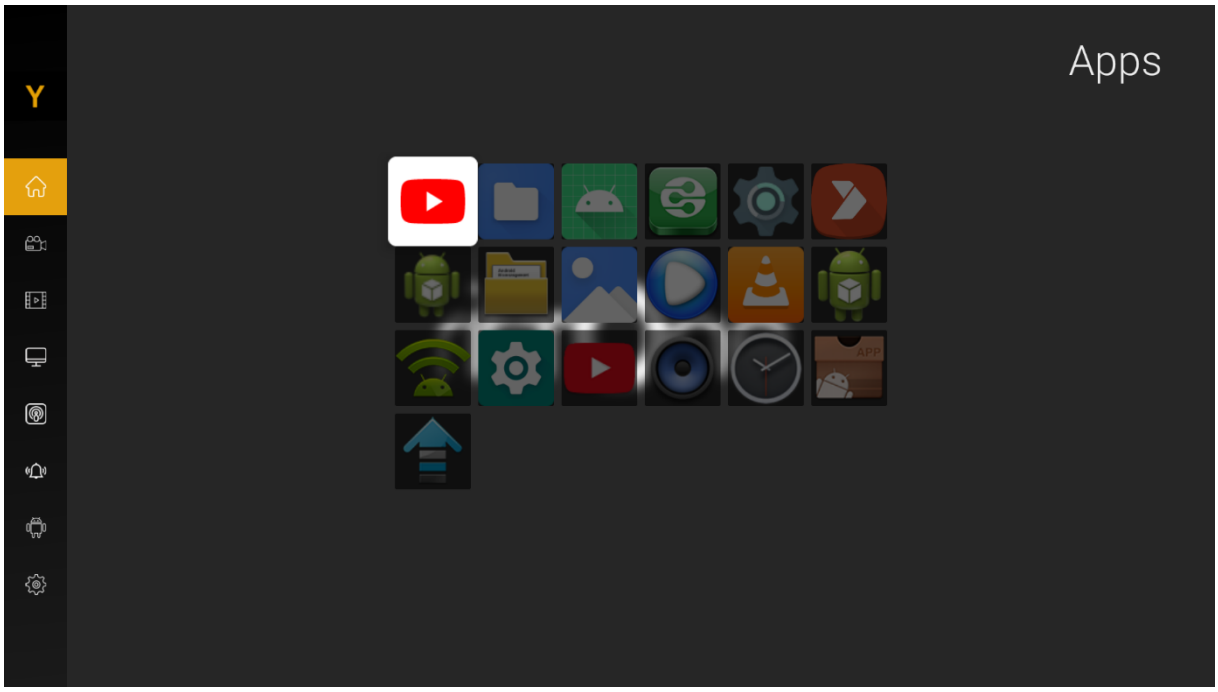

Figura 47: Tela Lista de *Apps* do Sistema

5.3 Testes de Software

Teste de *software* é uma das atividades do processo de desenvolvimento de sistema de *software* que visa executar um programa de modo sistemático com o objetivo de encontrar falhas, em um nível elevado, o teste de sistema tem o objetivo de provar que:

- A funcionalidade, entregue pela equipe de desenvolvimento, esta de acordo com as especificações do negócio no documento de especificação de requisitos.
- O *software* é de alta qualidade e dará suporte as funções de negocio desejadas e alcançar os padrões requisitados.

Na figura 48 encontra-se o modelo "V" que segundo Sommerville [20] mostra o processo de teste ideal, onde a preparação de teste começa assim que a definição de requisitos é produzido. O planeamento de teste de sistema começa em um estagio inicial, e por esta razão

o teste de sistema vai se beneficiar de iniciativas de qualidade ao longo do ciclo de vida do projeto.

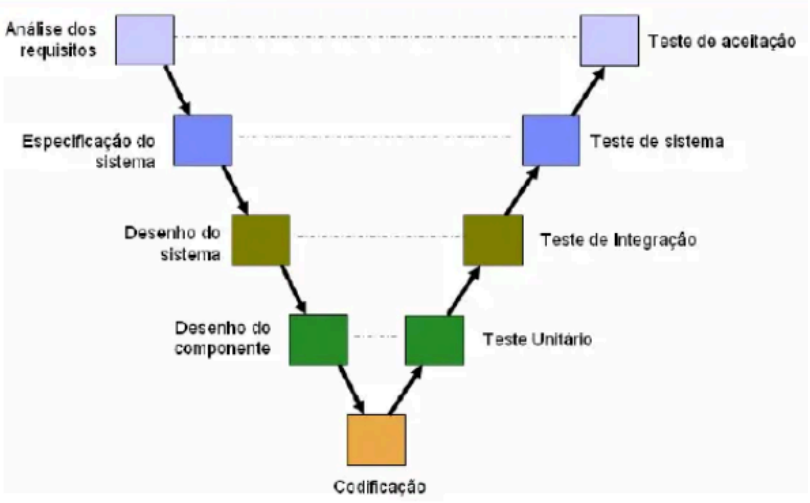

Figura 48: Envolvimento da Garantia de Qualidade de *Software*(modelo V). [20]

Existem diferentes testes que podem ser realizados ao longo do ciclo de vida dos projetos, sendo que os mais comuns são:

Testes de Integração

Estes testes provam que todas as áreas do sistema fazem interfaces entre si corretamente e que não há problemas no fluxo de dados. O teste final de integração prova que o sistema funciona como uma unidade integrada quando todas as partes estiverem completadas.

Teste Funcional

O objetivo deste teste é garantir que cada elemento do aplicativo atende os requisitos funcionais e de negócio.

Testes de Aceitação

Estes testes asseguram que o sistema opera da maneira esperada, e que o material de suporte está correto e serve ao seu propósito.

Testes de desempenho

Estes testes asseguram que o sistema fornece tempos aceitáveis de resposta.

Nesta secção será mostrada o resultado dos testes de desempenho da plataforma *YetuYaliy* com o objetivo de verificar o tempo de resposta da aplicação *web* e *Android TV*, já que a mesma lida com grandes quantidades de informações multimédias (imagens e vídeos).

5.3.1 Teste de Desempenho da Plataforma

Na realização dos testes de desempenho da plataforma foram verificados os seguintes pontos:

- Verificar o tempo de resposta de conexão ao servidor *web* em relação aos terminais (aplicação *web* e *Android TV*).
- Verificar o tempo de consulta/atualização das informações
- Verificar se o tempo de resposta para operações que envolvam dados multimédias (imagens, vídeos) não ultrapassam 15 segundos.

5.3.2 Resultados Obtidos

5.3.2.1 Tempo Médio de Conexão ao Servidor *Web*

- Aplicação *web*: 2.6 segundos;
- Aplicação *Android TV*: 2.6 segundos;

5.3.2.2 Tempo Médio de Consulta/Atualização dos Dados

- Aplicação *web*: 402 milissegundos;
- Aplicação *Android TV*: 402 milissegundos;

5.3.2.3 Tempo Médio de Acesso aos Recursos Multimédias

- Aplicação *web*: 4.49 segundos;
- Aplicação *Android TV*: 4.49 segundos;

Tais resultados podem ser conferidos na secção 5.3.3.

5.3.3 *Logs* com os Resultados Obtidos

5.3.3.1 Tempo Médio de Conexão ao Servidor *Web*

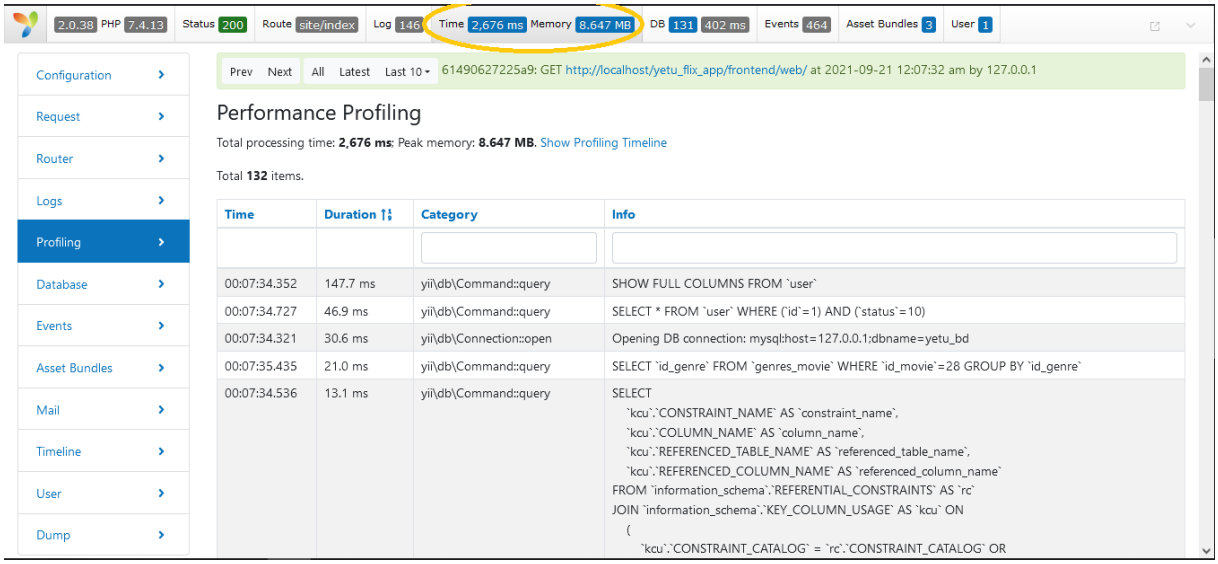

Figura 49: Tempo Médio de Conexão ao Servidor Web

5.3.3.2 Tempo Médio de Consulta/Atualização dos Dados

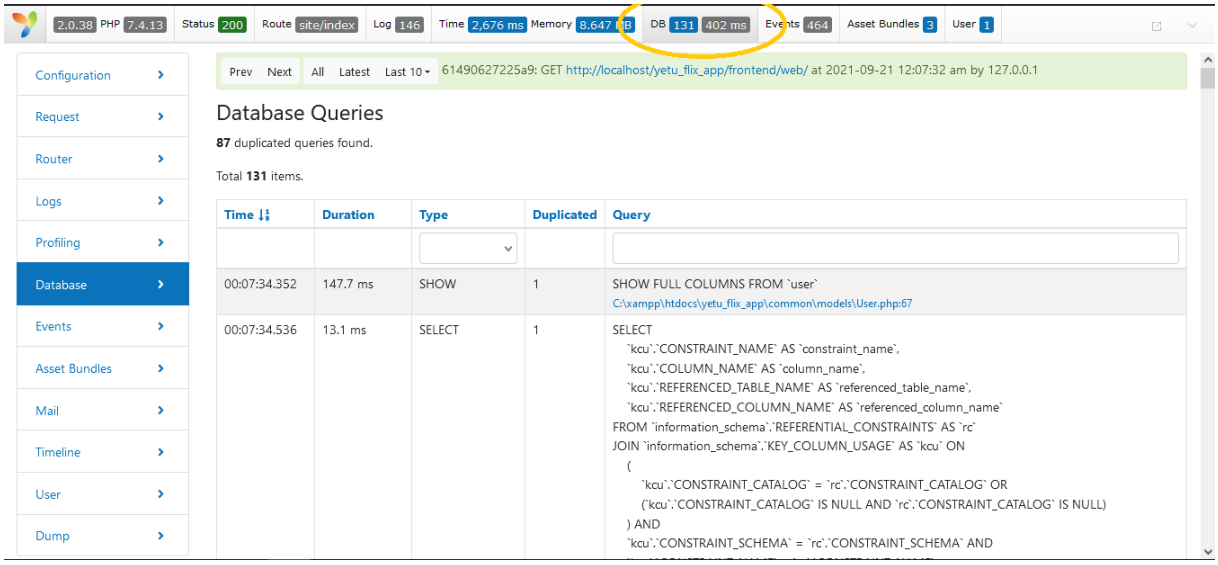

Figura 50: Tempo Médio de Consulta/Atualização dos Dados

5.3.3.3 Tempo Médio de Acesso aos Recursos Multimédias

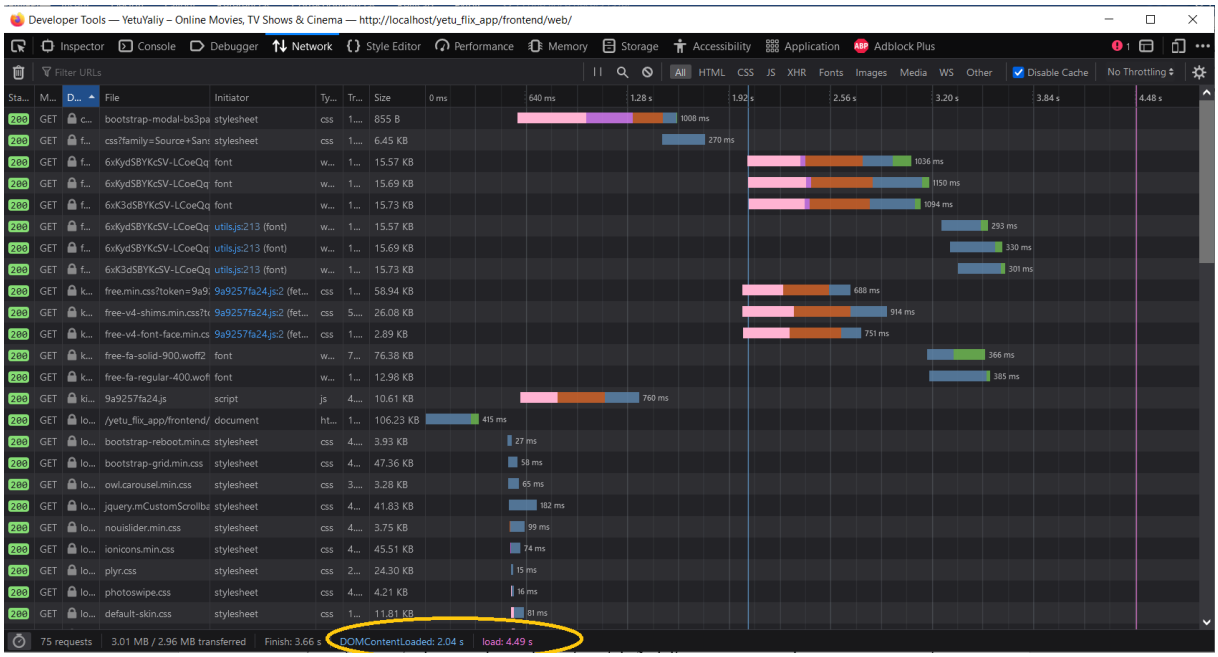

Figura 51: Tempo Médio de Acesso aos Recursos Multimédias

6 CONCLUSÃO

No contexto do continente africano em geral e do país em particular foi constatada a escassez de plataformas especializadas na promoção e divulgação de conteúdos áudio visuais, já que tanto a nível nacional quando continental não faltam pessoas que desejam consumir estes tipos de conteúdos.

Devido a estes factos foi desenvolvido a plataforma *YetuYaliy*, cujo todo o processo foi explicado e detalhado aqui neste relatório final, no cômputo geral os resultados finais obtidos foram bem conseguidos e o objetivo geral proposto no início, que era a de desenvolver uma plataforma *OTT* dividida em dois módulos, *web* e *Android TV* foram alcançados.

Para que tal fosse possível foi necessário aplicar um conjunto de técnicas de programação e engenharia juntamente com ferramentas que atendessem aos requisitos levantados na fase de análise. Tudo isto para que o sistema *web* e a aplicação *Android TV* estivessem à altura de satisfazer as necessidades identificadas do público geral.

Dentre as ferramentas utilizadas é de se destacar a biblioteca *Android TV Leanback* da Google, escrita em java e que permitiu de forma rápida e fácil construir a interface de utilizador da aplicação de acordo com as normas e especificações da Google.

A nível de *hardware* o destaque vai sem dúvida ao *SBC* da *Khadas* que aqui neste projeto teve um papel sumamente importante, já que sem ela seria mais difícil desenvolver uma aplicação para *Android TV* que se adequasse a um dispositivo real.

Já no que tange ao modelo de negócio da plataforma, que desde o início ficou determinado que seria com base na subscrição dos utilizadores tal não seria possível se não fossem as plataformas online de pagamentos utilizadas neste projeto que foram o *PayPal* e o *SISP*.

Porém, não é só de resultados positivos que este trabalho se sustentou, ao longo do desenvolvimento vários desafios e problemas foram apresentados e o maior deles tem haver com a dimensão deste tema e consequentemente desta plataforma, num mundo onde a tecnologia evolui a cada dia e as plataformas de *streaming* e *OTT* acompanham esta evolução e constantemente surgem novas soluções no mercado, torna-se impossível para uma pessoa só desenvolver uma plataforma completa que chegue aos pés das existentes, devido a isto o sistema aqui desenvolvido enquadra naquilo que o mercado denomina de *MVP*(Produto Mínimo Viável) e como tal futuramente funcionalidades serão incrementadas.

6.1 Trabalhos Futuros

Como trabalhos futuros pode-se apontar:

- Desenvolvimento de um painel para controle e gestão de finanças gerais das distribuidoras;
- Desenvolvimento de um módulo *web* para controle total da plataforma *YetuYaliy* para gestão de utilizadores, distribuidoras e conteúdos;
- Adequação e aperfeiçoamento das *API's REST* para listar canais e conteúdos de forma categorizada e paginada permitindo menos sobrecarga do servidor;
- Implementação de pagamentos online direto na aplicação *Android TV*;
- Desenvolvimento de aplicativos nativos para *mobile*;
- Implantação e hospedagem do site para o uso num ambiente real;
- Tradução dos apps em diferentes idiomas(Internacionalização)

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Amazon Advertising, “What is OTT (over-the-top)? A complete guide,” Amazon, 08 Fevereiro 2019. [Online]. Available: https://advertising.amazon.com/library/guides/what-is-ott. [Acedido em 13 Novembro 2020].

[2] D. Flanagan, JavaScript: The Definitive Guide, 7th Edition, Sebastopol: O'Reilly, 2020.

[3] The PHP Group, “PHP: O que é o PHP? - Manual,” The PHP Group, 12 Outubro 2015. [Online]. Available: https://www.php.net/manual/pt_BR/intro-whatis.php. [Acedido em 15 Abril 2019].

[4] D. L. Patrick Niemeyer, Learning Java, 4th Edition, Sebastopol: O'Reilly, 2013.

[5] Oracle Corporation, “MySQL 5.7 Reference Manual-Including MySQL NDB Cluster 7.5 and NDB Cluster 7.6,” Oracle Corporation, Redwood City, Califórnia, EUA, 2018.

[6] L. Richardson e S. Ruby, RESTful Web APIs, Newton (Massachusetts): O'Reilly, 2007.

[7] PayPal, “NVP API,” 22 Setembro 2020. [Online]. Available: https://developer.paypal.com/docs/nvp-soap-api/gs-PayPalAPIs/. [Acedido em 22 Setembro 2020].

[8] Sociedade Interbancaria e Sistemas De Pagamento, “Especificação de Serviço V4.1,” Sociedade Interbancaria e Sistemas De Pagamento, Praia, 2020.

[9] Yii, “Introdução: Sobre o Yii | Guia Definitivo para Yii 2.0 | Yii PHP Framework,” 12 Março 2008. [Online]. Available: https://www.yiiframework.com/doc/guide/2.0/pt-br/intro-yii. [Acedido em 21 Novembro 2019].

[10] Android Authority, “The history of Android: The evolution of the biggest mobile OS in the world,” Android Authority, 12 Novembro 2020. [Online]. Available: https://www.androidauthority.com/history-android-os-name-789433/. [Acedido em 10 Dezembro 2020].

[11] M. Hachman, “Google launches Android TV -- and here's what it looks like | PCWorld,” PC World From IDG, 25 Junho 2014. [Online]. Available: https://www.pcworld.com/article/2367696/google-launches-android-tv-and-heres-what-it-looks-like.html. [Acedido em 15 Novembro 2020].

[12] Square, Inc., “OkHttp,” 22 Novembro 2019. [Online]. Available: https://square.github.io/okhttp/. [Acedido em 22 Novembro 2019].

[13] Khadas, “VIM2 | Khadas,” Khadas, 20 Junho 2020. [Online]. Available: https://www.khadas.com/vim2. [Acedido em 20 Setembro 2020].

[14] Khadas, “VIM2 Top | VIM 2,” 19 12 2020. [Online]. Available: https://www.khadas.com/vim2?pgid=k7frxwrp-4944cba7-3863-4afe-992e-2665f3d8001a .

[15] Khadas, “VIM2 Bottom | VIM2,” Khadas, 19 12 2020. [Online]. Available: https://www.khadas.com/vim2?pgid=k7frxwrp-494f0ccd-6bb6-4933-b1d0-878d7bc52475 . [Acedido em 19 12 2020].

[16] Khadas, “DVB/DTV Extension Board for VIM2 (VTV),” Khadas, 17 Julho 2017. [Online]. Available: https://forum.khadas.com/t/dvb-dtv-extension-board-for-vim2-vtv/880. [Acedido em 19 12 2020].

[17] R. S. Pressman, Software Engineering, Apractitioner’s Approach, America,New York: Fifth Edition, 2001.

[18] P. Jalote, An Integrated Approach to Software Engineering. 3. ed, New York: Springer, 2005.

[19] International Journal of Engineering Sciences & Research Technology, “Comparative Study: Waterfall VS Agile Model,” *Comparative Study: Waterfall VS Agile Model,* p. 7, 04 Março 2015.

[20] I. Sommerville, Engenharia De Software, Brasil,São Paulo: Pearson Education do Brasil, 2011.

[21] Yii Framework, “Application Structure,” 12 Março 2015. [Online]. Available: https://www.yiiframework.com/doc/guide/2.0/en/images/application-structure.png. [Acedido em 15 Dezembro 2020].

[22] J. Rumbaugh, I. Jacobson e G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition), Boston: Addison-Wesley, 2004.