BRUNO QUEIROZ SANTOS

DESENVOLVIMENTO DE SOFTWARE PARA UNIFICAR PODCASTS EM UM ÚNICO SISTEMA WEB

LAVRAS – MG

2020

BRUNO QUEIROZ SANTOS

DESENVOLVIMENTO DE SOFTWARE PARA UNIFICAR PODCASTS EM UM ÚNICO SISTEMA WEB

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Rafael Serapilha Durelli
Orientador

LAVRAS – MG

2020

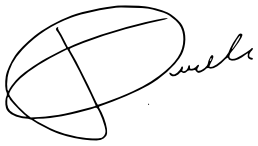
Bruno Queiroz Santos

DESENVOLVIMENTO DE SOFTWARE PARA UNIFICAR PODCASTS EM UM ÚNICO SISTEMA WEB

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADO em 01 de setembro de 2020.

Dr. Rafael Serapilha Durelli	UFLA
Dr. Mauricio Ronny de Almeida Souza	UFLA
Dr. Paulo Afonso Parreira Junior	UFLA



Prof. Dr. Rafael Serapilha Durelli

Orientador

LAVRAS – MG
2020

Dedico aos meus pais, Alexsandro Eustáquio dos Santos e Raquel Queiroz Pucineli e meus avós José Eustáquio dos Santos, Marta de Fátima Santos e Heloiza Machado

AGRADECIMENTOS

Agradeço a toda minha família que sempre me apoiou a fazer um curso superior, principalmente a minha avó que sem ela não estaria escrevendo está monografia. Agradeço a UFLA, por me dar esta oportunidade. Agradeço ao meu orientador Dr. Rafael Serapilha Durelli, por me ajudar no processo de execução deste trabalho.

RESUMO

Este projeto acadêmico consiste em descrever todas as atividades feitas na empresa Grow Further LLC. Está empresa busca acelerar o desenvolvimento de vários projetos de StartUps, tentando buscar um produto de mercado sólido de forma rápida e eficiente para finalmente buscar investidores, sendo assim este relatório irá demonstrar técnicas e tecnologias utilizadas que ajudaram a desenvolver o sistema até conseguir visibilidade pelos usuários. O software que está sendo desenvolvido se chama Jifcast, o qual é constituído por um sistema web, mobile e admin que irá servir de forma de gratuita Podcasts, o sistema admin serve para administrar as atividades dos usuário, determinar a retenção, fazer envio de notificações e gerir o software em geral. A principal tarefa desse projeto é conseguir grandes números de usuários em um curto espaço de tempo, para conseguir medir isso, utilizamos algumas ferramentas que geram métricas e estatísticas de como o usuário interage com cada página da aplicação.

Palavras-chave: Web. Desenvolvimento. Startup.

ABSTRACT

This academic project consists of describing all the activities done at Grow Further LLC. This company seeks to accelerate the development of several StartUps projects, trying to search for a solid market product quickly and efficiently to finally seek for investors, so this report will show techniques and technologies used that helped to develop the system until achieving visibility by users. The software being developed is called Jifcast, which consists of a web, mobile and admin system that will serve free Podcasts, the admin system is to manage user activities, determine retention, send notifications and manage the software in general. The main task of this project is to get large numbers of users in a short time, to be able to measure this, we use some tools that generate metrics and statistics of how the user interacts with each page of the application.

Keywords: Web. Development. Startup

LISTA DE FIGURAS

Figura 1.1 – Horas trabalhadas em todo estágio 9
Figura 3.1 – Google Analitycs do dia 04/06/2020 14
Figura 3.2 – Mapa de calor gerado pelo Software Yandex Metrica 14
Figura 4.1 – Arquitetura geral do sistema 15
Figura 5.1 – Página de recomendações 17

SUMÁRIO

1 INTRODUÇÃO 7
1.1 Contextualização 7
1.2 Justificativa 7
1.3 Objetivos do estágio 7
1.4 Descrição do ambiente de trabalho 8
1.5 Estrutura do documento 9
2 Referencial Teórico 10
2.1 Manifesto Ágil 10
2.2 Back-end 10
2.2.1 MongoDB 10
2.2.2 NodeJs 11
2.2.3 Python 11
2.3 Front-end 11
2.3.1 ReactJs 11
2.3.2 NextJs 12
3 Metodologia de Desenvolvimento 13
4 Arquitetura do Sistema 15
4.1 Servidores 15
4.1.1 Servidor da API 15
4.1.2 Servidor Front end do cliente 16
4.1.3 Servidor Front end do administrador 16
4.1.4 Servidor para coleta de RSS Feeds 16
4.1.5 Servidor para inserção/atualização de novos podcasts 16
5 Jifcast 17
6 CONCLUSÃO 19
6.1 Disciplinas que auxiliaram na condução do projeto 19
6.2 Considerações finais 19
REFERÊNCIAS 21

1 INTRODUÇÃO

O objetivo principal desse documento é apresentar todas as atividades feitas pelo estagiário na empresa Grow Further LLC.

1.1 Contextualização

O software desenvolvido tem como função principal reunir podcasts disponíveis na internet. Os podcasts são arquivos de áudio ou de vídeo disponibilizado através da internet que apresentam temas variados como cinema, literatura, programação, política entre outros. Eles podem ser comparados com programas de rádio, mas com uma grande vantagem: podem ser acessados a qualquer momento.

Qualquer usuário pode criar podcasts e disponibilizar na internet, utilizando um padrão feed RSS. Esse padrão que permite usuários subscreverem a um determinado podcast e acompanhar as próximas postagens do autor.

Segundo a especificação 2.0 do RSS, o (RSS...,) é um formato de organização de conteúdo da WEB que significa ***R**eally **S**imple **S**yndication*

1.2 Justificativa

Como existem pessoas publicando podcasts em diversas plataformas, a Grow Further LLC criou um meio de unificar todos os podcasts em único sistema web. No inicio vamos buscar apenas os podcasts da plataforma Itunes, para efetuar alguns testes e ver o funcionamento do software, logo ápos o foco da empresa será expandir e buscar conteúdo de outras plataformas, como Spotify, Deezer. Assim será muito mais fácil encontrar diversos tipos de conteúdos em um único lugar. E este lugar é a plataforma Jifcast, que foi desenvolvida como objetivo deste estágio.

1.3 Objetivos do estágio

O objetivo desse estágio é finalizar o software Jifcast em uma versão beta, onde o usuário conseguirá buscar e ouvir podcasts. Como existem vários desenvolvedores o estagiário deve cuidar e revisar o código de todos antes de aprovar, para isso é utilizado as ferramentas de

gerenciamento de versão o Git[1]. e o BitBucket[2]. Após a aprovação do código é necessário também subir o código pronto para a produção, onde os usuários poderão começar a utilizar. O estagiário também deve cuidar de todo o desenvolvimento de novas funcionalidades para a plataforma.

1.4 Descrição do ambiente de trabalho

O ambiente de trabalho é em casa, toda a empresa funciona *Home Office*, sendo assim existem várias normas para se cumprir. O meio de comunicação é feito pelo o Skype utilizando apenas texto, sem video chamadas, a linguagem utilizada é o inglês.

Existem outros desenvolvedores na empresa, porém não existe o contato direto entre eles, apenas o chefe é quem gerencia tudo e atribui as tarefas para cada um. Os outros desenvolvedores não são fixos, o chefe contrata algumas pessoas para executar apenas uma tarefa, sendo assim é preciso fazer um controle de qualidade de código, está tarefa é atribuída para o estagiário.

A atribuição de tarefas é feita no Trello utilizando o método Kanban juntamente com o Scrum, porém este Scrum é mais resumido, todo dia é atribuída uma quantidade de tarefas que devem ser entregues no dia, as chamadas Daily Sprint. O quadro do Trello é separado nas seguintes secções: To Do (tarefas a serem feitas), Today (tarefas para terminar no dia), Still not done (tarefas que nao foram terminadas no dia), Review (tarefas enviadas para revisão) e Done (tarefas concluídas).

Para o controle de horas de cada desenvolvedor, é necessário utilizar um sistema que conta as horas trabalhadas e verifica se o funcionário está realmente trabalhando. Esse software se chama TimeDoctor (TIMEDOCTOR,), para utilizá-lo o desenvolvedor precisa acioná-lo antes de começar o trabalho, e pausá-lo para almoço ou outras atividades. Ele também pausa sozinho após alguns minutos de ociosidade na máquina.

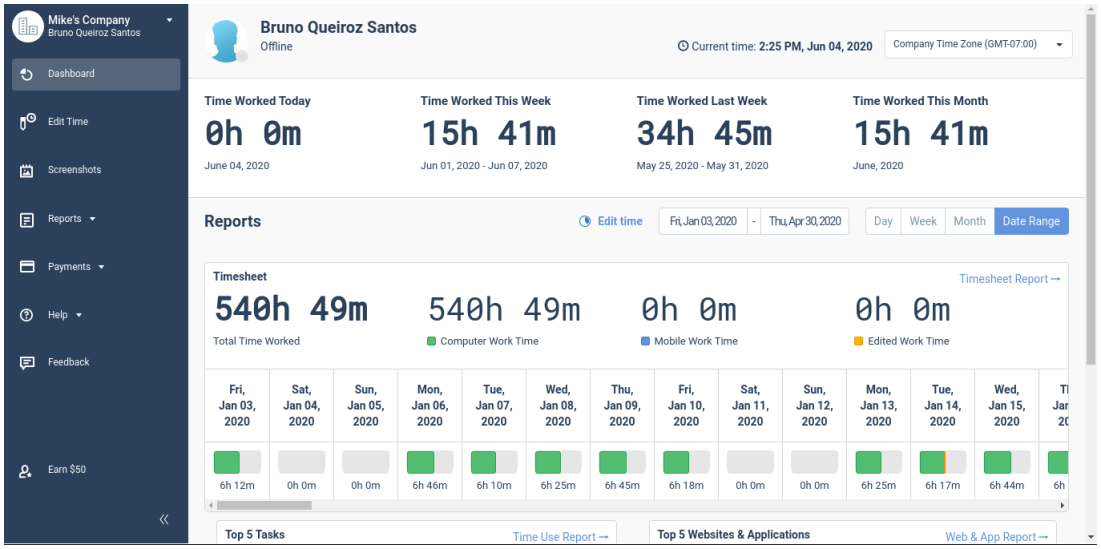
Além de cronometrar as horas trabalhadas e monitorar tempo de ociosidade, o software também tira fotos da tela do computador e verifica quais conteúdos o funcionário da empresa acessa mais, gerando um relatório mostrando qual software, site o empregado está utilizando para trabalhar.

[1] Git <https://git-scm.com/about>
[2] Bitbucket <https://bitbucket.org/product/features/>

Também é necessário pausar o TimeDoctor ao finalizar uma determinada tarefa e iniciá-lo novamente descrevendo a tarefa seguinte, assim o software pode fazer estatísticas de quais tarefas o desenvolver gastou mais tempo para realizar.

Abaixo na Figura 1.1 está ilustrado a interface que mostra o total de horas trabalhadas dentro do intervalo de datas que foi realizado o estágio.

Figura 1.1 – Horas trabalhadas em todo estágio



Fonte: do autor (2020)

1.5 Estrutura do documento

Nos próximos capítulos, o documento explica no capítulo 2 o referencial teórico que descreve as tecnologias utilizadas no Jifcast e alguns termos utilizados no documento. O capítulo 3 fundamenta algumas técnicas de desenvolvimento ágil utilizadas pela Grow Further LLC, no capítulo 4, toda a arquitetura do sistema é descrita, no capítulo 5, todas as principais funcionalidades do sistema são explicadas. Por fim, no capítulo 6 o trabalho é sumarizado mostrando os benefícios e dificuldades que o estágio proporcionou.

2 REFERENCIAL TEÓRICO

Nos tópicos seguintes o documento explica cada uma das tecnologias utilizadas e alguns termos utilizados no documento.

2.1 Manifesto Ágil

Citando (MANIFESTO. . . ,), é uma declaração de princípios que fundamentam o desenvolvimento ágil de software. Ele se divide em 4 premissas:

1. Indivíduos e interações mais que processos e ferramentas
2. Software em funcionamento mais que documentação abrangente
3. Colaboração com o cliente mais que negociação de contratos
4. Responder a mudanças mais que seguir um plano

2.2 Back-end

Back-end são os programas que rodam no lado do servidor, onde o usuário não tem contato direto. Abaixo segue as tecnologias *back-end* utilizadas.

2.2.1 MongoDB

O banco de dados escolhido foi o MongoDB[1]., um banco de dados NoSQL não relacional. Este tipo de banco é bom para trabalhar com Big Data (área do conhecimento que estuda como tratar, analisar e obter informações a partir de conjuntos de dados grandes demais), ele fornece uma consulta aos dados mais rápida, independente da volume de dados.

Como o sistema possui um script que busca por podcasts a todo momento, milhões de dados serão adicionados ao MongoDB. Existe aproximadamente 40 milhões de podcasts salvos no Jifcast, e a partir de um mês esse número pode dobrar facilmente. Como o Jifcast não precisa gerar relatórios bem elaborados para os clientes e para o admin, não foi necessário utilizar um banco relacional como MySql[2], Postgres[3].

[1] MongoDB <https://docs.mongodb.com/manual/>
[2] MySql <https://dev.mysql.com/doc/refman/8.0/en/>
[3] Postgres <https://www.postgresql.org/about/>

2.2.2 NodeJs

O NodeJs[4] foi o framework utilizado para servir a API (*Application Programming Interface*) da aplicação, ele é um framework que utiliza Javascript e está bastante difundido no mercado. Ele foi escolhido para diminuir os custos com infraestrutura e por ser bastante rápido. Como o NodeJs não abre uma thread (tarefa que um determinado programa realiza) para cada conexão o servidor fica menos carregado, e apenas um simples servidor consegue lidar com um certo número de usuários. Para fazer as requisições HTTP *HyperText Transfer Protocol*) foi utilizado o Express, que é um microframework para contruir APIs REST (*Representational State Transfer*) de forma simples e ágil.

2.2.3 Python

A linguagem de programação Python também é utilizada em algumas tarefas que o NodeJs não resolveria bem, como por exemplo a adição constante de podcasts no sistema.

Para adicionarmos muitos podcasts de forma rápida foi necessário o uso de processos e threads, por isso escolhemos Python. Assim todos os scripts externos a aplicação são feitos em Python, visto que trabalhar com threads não é muito viável em Nodejs.

2.3 Front-end

Front-end são os programas que rodam ao lado do cliente, ou seja onde o usuário tem contato direto, abaixo segue as tecnologias front-end utilizadas.

2.3.1 ReactJs

O ReactJs[5] é uma biblioteca Javascript utilizada para criar interfaces de usuário em páginas web. Ela também é bastante difundida no mercado, é umas das melhores tecnologias para consumir dados de APIs REST e organizá-los na interface para o usuário utilizar.

Está biblioteca foi criada em 2013 para substituir o modo antigo de desenvolvimento front-end, ela possui uma arquitetura de componentes o que facilita bastante a organização de grandes projetos.

Ao invés de renderizar a página inteira cada interação do usuário, esta biblioteca já traz toda a aplicação em uma única página HTML (*Hypertext Markup Language*), assim o usuário

[4] Nodejs <https://nodejs.org/en/about/>
[5] Reactjs <https://reactjs.org/>

consegue ir navegando e montando novas paginas sob demanda, sem ter que renderizar uma por uma. Mas isso as vezes é um problema para a plataforma do Google encontrar o website com os motores de busca, por isso utilizamos está tecnologia apenas para o sistema admin, visto que não precisamos divulgar a plataforma admin no Google.

2.3.2 NextJs

O NextJs[6] é um framework que trabalha acima do ReactJs, ele veio para ajudar na otimização dos motores de busca do Google. Isso só é possível devido ao SSR (*Server Side Renderer*) que é a principal funcionalidade do NextJs.

A ideia principal é renderizar a página no servidor apenas na primeira chamada do usuário, assim quando o Google for buscar pelas páginas do website, todos os dados já estarão presentes do arquivo HTML, assim o Google consegue classificar o sistema em melhores posições no resultado da busca. Quando a página utiliza apenas o ReactJs os motores de busca do Google perdem muita informação pois os dados da página será populado apenas quando toda pagina chegar no browser do cliente.

Em termos mais técnicos o NextJs tem o trabalho de renderizar a página no servidor na primeira requisição que o cliente faz ao servidor, ou seja quando o usuário entra em alguma rota do site, no primeiro carregamento o HTML virá preenchido com todos os dados, conforme o usuário vai mexendo e alterando a página as próximas alterações na página são feitas de forma dinâmica utilizando o poder do ReactJs.

[6] Nextjs <https://nextjs.org/>

3 METODOLOGIA DE DESENVOLVIMENTO

A empresa utiliza a metodologia LEAN para dar início aos projetos de Startup. Esta metodologia tem foco em evitar desperdícios de dinheiro e tempo, buscando assim apenas o necessário para efetuar um tarefa. O termo LEAN significa enxuto, reforçando a ideia de use apenas o que for necessário.

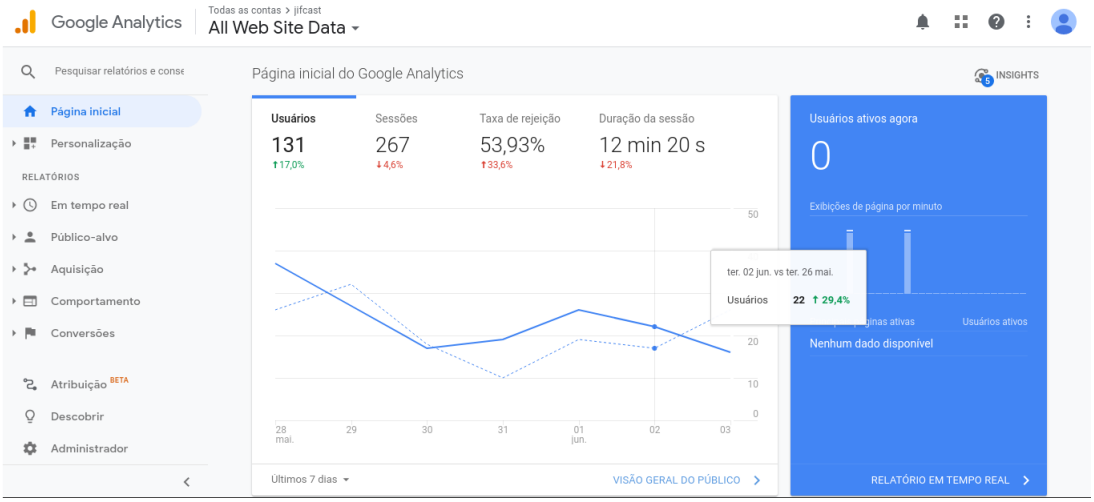
Isso significa que não é necessário reunir todos os requisitos antes de começar a desenvolver, é possível começar a desenvolver sob demanda de acordo com o que cliente pede. Está metodologia está associada ao Manifesto Ágil, que também prevê uma série de práticas para enxugar e dar mais leveza aos processos de desenvolvimento.

A partir dessa metodologia foi criado pelo empreendedor americano Eric Ries o termo Lean startup, onde ele apresentou o conceito em seu livro A Startup Enxuta (RIES, 2019). O livro trás ideias que podem ser aplicadas a qualquer tipo de empresa seja de grande ou pequeno porte que queira iniciar de forma rápida e gerar resultados sem grandes complicações.

Assim a Grow Futher LLC busca otimizar seus produtos e serviços de forma contínua e revisar seus processos internos, sempre levando o feedback do cliente em consideração. O usuário que utiliza o Jifcast que nos informa o que deve ser implementado até descobrirmos o produto ideal que trará retenção de usuários.

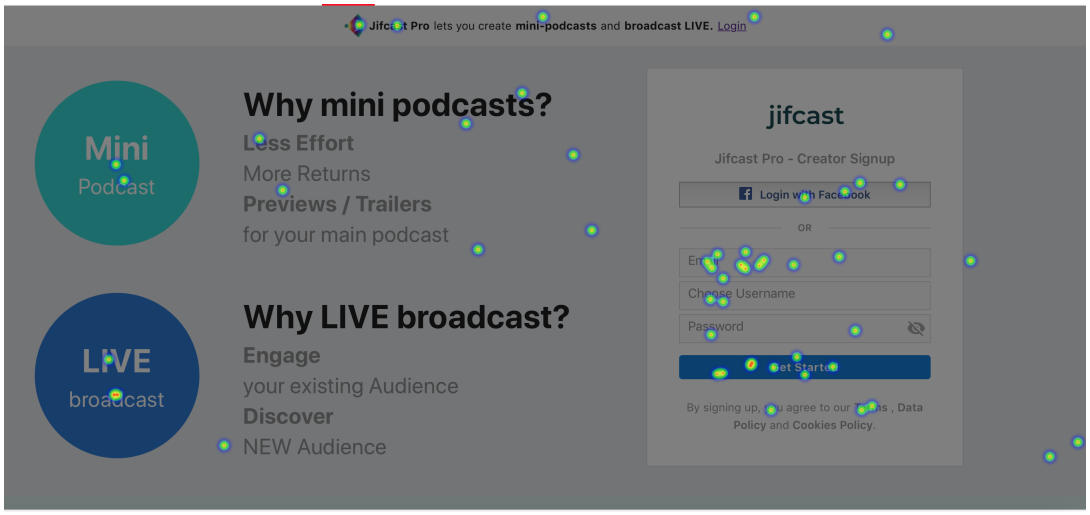
Para descobrir o que o cliente está procurando ou como o cliente utiliza o sistema, utilizamos duas ferramentas: o Google Analitycs (GOOGLE...,) e o Yandex Metrica (YANDEX...,), as duas ferramentas trazem métricas bastante interessantes. O Google Analitycs mostra a quantidade de usuários que está acessando o sistema, a taxa rejeição, localidade de usuários e outras métricas importantes, ilustrados na Figura 3.1. Já o Yandex Metrica mostra um mapa de calor, ilustrado na Figura 3.2, indicando a onde o usuário esta clicando ou procurando por funcionalidades no sistema. Está métrica é a mais importante, pois com ela podemos moldar um layout de acordo com a usabilidade da maioria dos usuários. Ele também gera um vídeo mostrando toda a sessão do usuário, assim é possível ver tudo que o usuário esta fazendo, onde está clicando, onde deseja novas funcionalidades, e o principal podemos saber se o cliente está conseguindo usar o sistema sem muitas complicações. Para utilizar estas duas ferramentas é simples, e necessário injetar um código javascript na página principal do sistema, assim ele consegue capturar todas essas informações, tudo isso de forma segura, não é possível ver dados digitados pelo o usuário nas fotos de mapa de calor ou nos vídeos das sessões.

Figura 3.1 – Google Analitycs do dia 04/06/2020



Fonte: do autor (2020)

Figura 3.2 – Mapa de calor gerado pelo Software Yandex Metrica



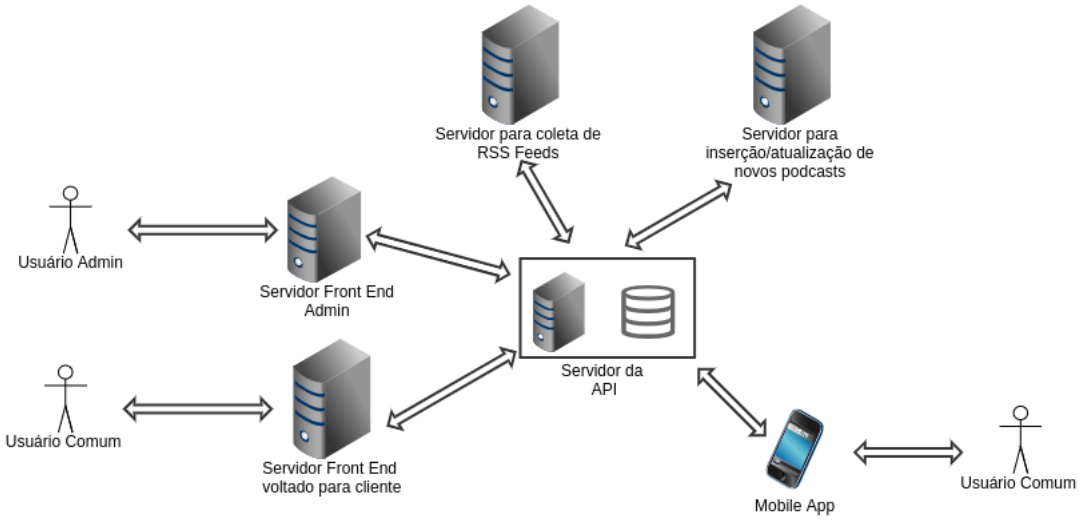
Fonte: do autor (2020)

4 ARQUITETURA DO SISTEMA

O sistema é divido em 4 partes principais que são o sistema web e mobile voltado para o cliente utilizar, um sistema admin voltado para reunir estatísticas e fazer gerenciamento do sistema web voltado para clientes, e por fim temos a API rest que interage com todos os dados. Existem também mais 2 servidores separados responsáveis por buscar e popular o banco de dados do sistema com novos podcasts que estão disponíveis na internet.

Logo abaixo está ilustrado na Figura 4.1 todo o esquema da arquitetura.

Figura 4.1 – Arquitetura geral do sistema



Fonte: do autor (2020)

4.1 Servidores

Os servidores utilizados são máquinas separadas independentes uma das outras que rodam o sistema operacional GNU/Linux [1].

4.1.1 Servidor da API

Este servidor é onde está localizada toda a parte *back end* do sistema, juntamente com o banco de dados. Aqui é reunida toda a lógica de programação que o sistema do cliente e admin necessita para funcionar. Aqui também fica localizado o banco de dados, para deixar a aplicação mais rápida para buscar os dados. Para deixar este servidor em produção utilizamos um

[1] GNU/Linuxhttps://www.gnu.org/gnu/linux-and-gnu.pt-br.html

proxy reverso do Nginx[2] para o aplicação. Um proxy reverso é um servidor de rede geralmente instalado para ficar na frente de um servidor Web.

4.1.2 Servidor Front end do cliente

Este servidor é onde fica toda a aplicação web *front end* para o usuário comum acessar. Aqui é onde o usuário irá fazer ações e estas ações farão acesso direto ao servidor da API rest para enviar e/ou receber dados.

4.1.3 Servidor Front end do administrador

Este servidor é onde fica toda a aplicação web *front end* para o administrador do sistema acessar. Aqui é onde o administrador do sistema irá fazer ações e estas ações farão acesso direto ao servidor da API rest para enviar e/ou receber dados.

4.1.4 Servidor para coleta de RSS Feeds

Este servidor busca todos os feed RSS do Itunes utilizando um script que armazena o link do feed e mais informações no banco de dados. O feed RSS é um arquivo XML (*Extensible Markup Language*) disponibilizado de forma online que contém todas as informações dos episódios de um podcast, sempre que alguém publica um novo podcast este feed é atualizado informando que um novo episódio foi adicionado.

4.1.5 Servidor para inserção/atualização de novos podcasts

Após a coleta dos feed RSS pelo o servidor descrito acima, esse servidor roda um script para ler todos os feed RSS armazenados buscando por novos episódios. Este script é praticamente o coração do sistema, pois ele é responsável por trazer todos os podcasts para o usuário final acessar. Este script sempre fica rodando, após a primeira execução, sempre executamos novamente, assim ele faz um parse constante de todos os feed RSS buscando por novos episódios.

[2] Nginx <https://www.nginx.com/company/>

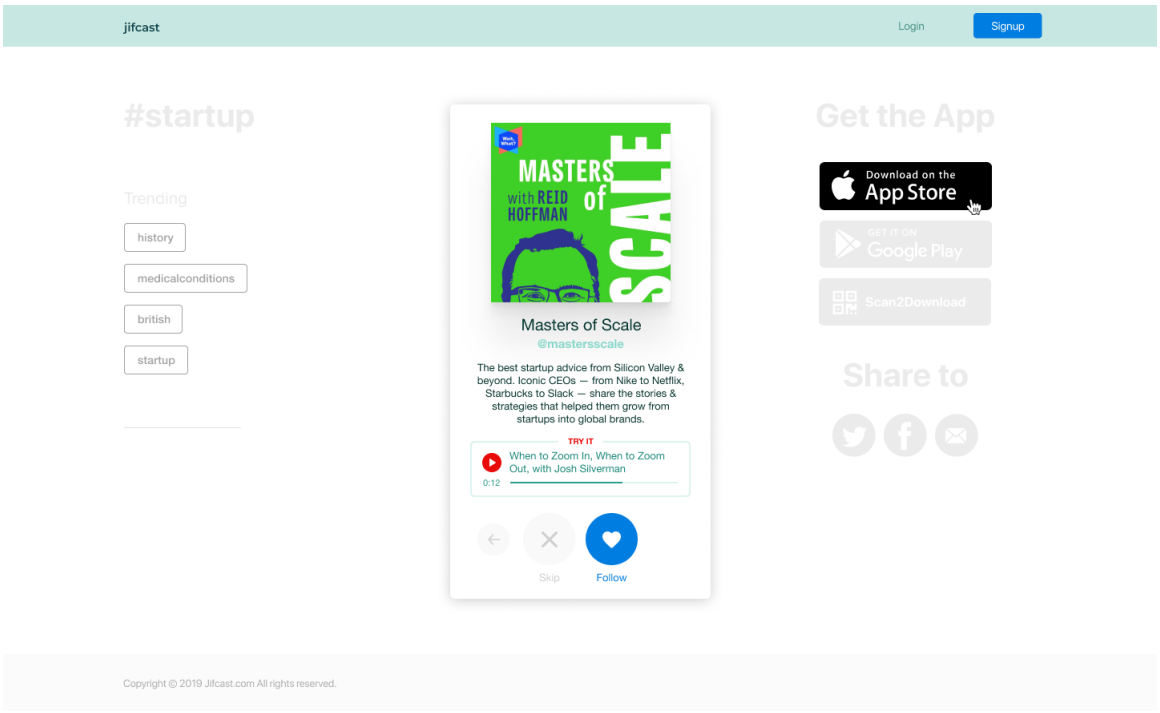
5 JIFCAST

O Jifcast irá disponibilizar podcasts gratuitos para usuários ouvirem tanto no celular quanto no computador. É um software que se parece com o Instagram, mas ao invés de mostrar fotos ele exibe apenas podcasts.

Para disponibilizar os podcasts é utilizado uma estrutura de canais e episódios, onde os episódios são os podcasts. O usuário pode seguir canais de podcasts, assim ao logar no sistema, é mostrado um feed com todos os episódios recentes dos canais que o usuário está seguindo.

Existe também uma inteligência artificial de recomendações, que de acordo com os tipos de canais que o usuário segue, é gerado para cada usuário uma serie de canais que possuem o mesmo conteúdo que o usuário está acostumado a ouvir, a tela desta funcionalidade está ilustrada na Figura 5.1. Assim quando o usuário for buscar por novos podcasts ele encontrará uma lista completa de recomendações.

Figura 5.1 – Página de recomendações



Fonte: do autor (2020)

Como os podcasts são buscados na internet, os podcasters podem reivindicar suas contas que foram adicionadas no sistema. E também podem atualizar algumas informações diretamente na plataforma. Isto só é possível pois o feed RSS contém as informações do criador do podcast como email e nome de usuário.

Para manter todos os usuário informados, a cada novo episódio adicionado no sistema, se o usuário seguir o canal que possui esse novo episódio, uma notificação será gerada e enviada para o aplicativo mobile e para a caixa de email. Isto ocorre também a cada novo canal recomendado gerado.

6 CONCLUSÃO

O estágio realizado na Grow Futher LLC, proporcionou um grande aprendizado, por ser uma empresa internacional, foi bastante interessante aprender sobre como funcionam o ambiente de uma StartUp americana, e discutir sobre assuntos técnicos em inglês. Este foi o primeiro estágio internacional realizado de forma remota sendo aproveitado como trabalho de conclusão de curso. Visto que quase todos os cursos da UFLA precisam de um ambiente presencial para executar o trabalho, o curso de Ciência da Computação tem essa nova possibilidade de trabalhar remotamente para empresas internacionais. Este trabalho serviu como auxilio para aplicar toda a teoria aprendida na faculdade, as principais experiências adquiridas foram:

1. Desenvolvimento ágil voltando o foco para o usuário
2. Rastrear o modo que as pessoas usam o sistema através das ferramentas Google Analytics e Yandex Metrica.
3. Otimização das ferramentas de buscas utilizando práticas de renderização de páginas no servidor com NextJs.
4. Utilizar MongoDB para aplicações que possuem grande volume de dados

6.1 Disciplinas que auxiliaram na condução do projeto

Todas as disciplinas auxiliaram para a conclusão deste estágio, mas as que mais foram utilizadas em todo o processo de desenvolvimento do software Jifcast foram: todas as disciplinas que envolvem programação como algoritmos em grafos, estrutura de dados, essa base auxiliou para trabalhar com os fluxos de dados do aplicativo. A matéria de banco de dados foi crucial para entender a necessidade de utilizar o MongoDB como banco do sistema. A matéria de sistemas operacionais juntamente com a matéria de redes de computadores que ajudou para configurar os servidores GNU/Linux que rodam todas as aplicações de forma segura.

6.2 Considerações finais

O software está com as principais funcionalidades prontas, o script que busca novos podcasts na internet precisa ser otimizado, pois alguns podcasts são publicados hoje, mas o Jifcast consegue adicionar ele depois de 2 dias aproximadamente. A retenção de usuários ate o

momento final do estágio ainda estava ruim, vários usuários entram na plataforma se cadastram e não entram mais, a empresa conclui que esse problema vem do layout das páginas, algo que deve ser melhorado no futuro. As notificações sobre novos podcasts adicionados estão com alguns bugs que a empresa acredita que seja problemas no script de busca de podcasts, como este script que alimenta todo o sistema, ele precisa ser refatorado buscando novas soluções para melhorar a experiência do usuário com os novos podcasts.

REFERÊNCIAS

GOOGLE Analytics. Disponível em: <https://analytics.google.com/analytics/web/#/>. Acesso em: 2 jun. 2020.

MANIFESTO para Desenvolvimento Ágil de Software. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 23 mai. 2020.

RIES, E. **A startup enxuta**. [S.l.]: Editora Sextante, 2019.

RSS 2.0. Disponível em: <https://www.rssboard.org/rss-specification>. Acesso em: 23 mai. 2020.

TIMEDOCTOR. Disponível em: <https://www.timedoctor.com/about-us.html>. Acesso em: 23 mai. 2020.

YANDEX App Metrica. Disponível em: <https://appmetrica.yandex.com/about/>. Acesso em: 2 junho 2020.