



Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica - Campus Patos de Minas
Engenharia Eletrônica e de Telecomunicações

FÁBIO CAMPOS FERREIRA
TIAGO MOREIRA CARNEIRO
JOSÉ DAVID VARGAS DE MELO NETO

SERVIDOR E CLIENTE IPTV
APLICAÇÃO WEB DE DIVULGAÇÃO DE VÍDEOS EDUCACIONAIS

Patos de Minas
2020

**FÁBIO CAMPOS FERREIRA
TIAGO MOREIRA CARNEIRO
JOSÉ DAVID VARGAS DE MELO NETO**

SERVIDOR E CLIENTE IPTV

APLICAÇÃO WEB DE DIVULGAÇÃO DE VÍDEOS EDUCACIONAIS

Relatório Parcial apresentado como um dos requisitos de avaliação na disciplina Projeto Interdisciplinar do Curso de Engenharia de Eletrônica e Telecomunicações da Universidade Federal de Uberlândia.

Orientador: Júlio César Coelho

Assinatura do Orientador

Patos de Minas
2020

SUMÁRIO

1 RESUMO	4
2 INTRODUÇÃO	4
2.1 ACESSO E QUALIDADE DO APRENDIZADO ONLINE.....	4
2.2 EXPERIÊNCIA	7
2.3 PLATAFORMAS PRESENTES NO MERCADO.....	7
3 CRONOGRAMA.....	8
4 MÉTODO	11
4.1 FLUXOGRAMA	12
4.2 FERRAMENTAS.....	14
4.3 PROTOCOLOS	15
4.4 DIAGRAMA DE CLASSES.....	19
4.5 INTERFACE	24
4 APLICATIVO WEB PROGRESSIVO	35
6 REACT.....	37
7 FIREBASE.....	40
8 ESTRUTURA BÁSICA DOS ARQUIVOS REFERENTES AO PROJETO.....	43
9 TELA DE LOGIN.....	48
10 TELA DE PESQUISA	49
11 TELA DE UPLOAD.....	53
12 RESULTADOS ESPERADOS.....	55
13 CONCLUSÃO	55
REFERÊNCIAS.....	57

1 RESUMO

O objetivo geral é criar um Servidor e Cliente IPTV, mas com algumas funcionalidades, por exemplo, o usuário pode escolher o vídeo que deseja assistir, ou seja, que o programa tenha um filtro de buscas por título, assunto, atores, diretores, etc. No servidor há necessidade de que o usuário consiga inserir ou retirar vídeos do próprio canal e também consiga visualizar e pesquisar. Outro objetivo do trabalho é criar um aplicativo para assistir no celular, e no computador assistir via navegador. E por último criar uma interface para que o administrador geral consiga excluir usuários, caso necessário.

Para alcançar os objetivos propostos nesse trabalho, foi pesquisado algumas ferramentas que podem atender as necessidades do projeto, para criar a página na web e o aplicativo, depois de ser feito pesquisas sobre várias ferramentas, a que escolhemos foi o React JS, que trabalha com a linguagem de programação Java Script, e foi desenvolvida pelo Facebook em 2013 para facilitar a criação de aplicativos. O banco de dados que provavelmente será utilizado é o Firebase.

2 INTRODUÇÃO

2.1 Acesso e Qualidade do Aprendizado Online

por Fábio Campos Ferreira

A evolução dos dispositivos de comunicação tornou o compartilhamento de informações entre pessoa um procedimento comum e de fácil execução. Em uma sociedade onde um grande número de pessoas estão conectadas através das redes de comunicação via internet disponibiliza um grande número de usuários com necessidade de consumo de conteúdo digital. Esta demanda é alimentada por um mercado de produtos online plataformas de vídeos ou de conteúdos em forma de texto, onde grandes corporações um único usuário da rede pode criar um produto ou conteúdo que pode ser consumido por um grande número de pessoas.

No que diz respeito a população brasileira, 79% já são participantes da internet, número que só vem a cada ano, como o de 2019, que apresentou um aumento de

28%. O celular é o principal responsável por estes números elevados, sendo que 99% dos usuários da internet se conectam através deste dispositivo, sendo um aparelho mais prático e com maior custo benefício que um computador (BRIGATTO).

Assim, este grande número de pessoas amplia as possibilidades do público alvo de projetos disponibilizados de forma online. Como o aqui retratado, como sendo uma plataforma de compartilhamento de vídeos, podendo alcançar qualquer pessoa dentro dos 79% de brasileiros, levando em conta, principalmente, a sua otimização para trabalhar como um aplicativo instalado em dispositivos móveis.

Contudo, o conteúdo dos vídeos é o parâmetro que especifica o público alvo. O caso mais interessante e produtivo é a disponibilização e compartilhamento de vídeos de caráter educacional como tutoriais, demonstração de projetos e procedimentos, bem como a elucidação de temas pertencentes a alguma área do conhecimento. Neste caso, a plataforma de visualização de vídeos proporciona aos usuários uma ferramenta de busca e visualização de conteúdos educacionais de forma fácil, rápida e confortável ao interessado em determinado assunto. Estas características devem se estender para os usuários que tem como objetivo utilizar a plataforma para disponibilizar vídeos por eles produzidos, que possam vir a ser úteis para outras pessoas.

Este processo se caracteriza como uma forma de educação a distância ou de forma livre por parte estudante. O consumo e compartilhamento de vídeos online em sites ou aplicativos em uma busca por entretenimento é uma atividade comum a muitos usuários, porém executar esta mesma ação com o objetivo voltado ao aprendizado pessoal é mais desafiador e requer um auto controle e planejamento de estudo. De modo igual, para o criador do vídeo, o professor, criar um produto atrativo e que facilite a vida dos que iram buscar um determinado conhecimento deve ser o principal objetivo destes, tentando superar as dificuldades que este método de aprendizagem por vídeo com a falta de interação com o aluno que pode estar ou não absorvendo corretamente os ensinamentos passados. E ao mesmo tempo, buscando utilizar novos recursos possíveis por este meio de aprendizagem, como acesso ao conteúdo a qualquer momento, por qualquer dispositivo (celular, computador, notebook ou tablet) quantas vezes forem necessárias, ou ainda, localizando uma fração do vídeo onde se encontra uma informação específica (BURNS).

Para exemplificar e comprovar o estudo por meio de vídeo aulas, a Universidade de Auckland, na Nova Zelândia, apresentou um estudo realizado com seus alunos de medicina. Neste, os alunos de vários períodos foram separados em dois grupos, para um destes foram disponibilizados vídeos explicativos de procedimentos para o treinamento dos alunos para realizarem cirurgias. As repostas dos alunos a questionários sobre o seu preparo para realizarem os procedimentos da cirurgia mostrou que os alunos que tiveram acesso ao vídeo se disseram mais autoconfiantes e com uma maior competência na realização dos procedimentos. Em (Arruda), os professores que se dispuseram a implementar uma estudo extra ao alunos por meio de videoaulas enfatizaram dificuldades na gravação e edição destas, como falta de tempo, de um computador e edição para cada professor e a disponibilidade de outros recursos como salas e objetos de estudo específicos. Porém, segundo eles, foi possível a disponibilização de um vídeo final com um elogiável nível de qualidade, mesmo com as referidas dificuldades (ARRUDA).

Um fato inusitado que alavancou a procura de meios de educação a distância e obrigou a professores e alunos, a maioria totalmente despreparada para esta mudança, a vivenciarem citações parecidas com as relatadas em (ARRUDA) foi o distanciamento social implantando nas maiorias dos países do globo, em 2020, com o objetivo de conter o avanço do vírus COVID-19.

Portanto, o projeto da criação de uma plataforma de compartilhamento de vídeos se faz importante ao definir seus objetivos para com os usuários, de realizar a entrega, localização rápida e segura de conteúdos audiovisuais específicos entre um grande conjunto. Reduzindo as preocupações, por parte dos usuários, suas preocupações no seu processo de aprendizagem.

Atualmente os serviços de Streaming oferecidos na internet como exemplo Netflix, YouTube, Spotify, dentre outros, cresceram muito, com um ritmo ainda mais acelerado devido a pandemia provocada pelo novo corona vírus. As pessoas estão utilizando esses meios de comunicação para poder ter renda, através da criação de conteúdo informativo ou de entretenimento ou simplesmente para quem está procurando entretenimento ou informação online.

Pensando nisto foi proposto como trabalho da disciplina de projeto interdisciplinar a criação de um serviço IPTV, onde no qual será oferecido um serviço

de streaming. Os usuários poderão criar conteúdo ou simplesmente buscar por algum assunto desejado.

2.2 Experiência

Por Tiago Moreira Carneiro

Além do benefício para a sociedade como um todo, para nós alunos do curso também ganhamos com este projeto, pois vamos aprofundar nossos conhecimentos em algumas áreas que não são abordadas no curso de Engenharia Eletrônica e Telecomunicações. Um dos assuntos que necessitamos aprofundar é a linguagem de programação JavaScript, que é um tipo de programação *front-end* (interpretada diretamente no navegador).

Para criar a interface de usuários optamos pela ferramenta React, que é uma biblioteca do JavaScript, estas ferramentas foram escolhidas por serem de certa forma mais fácil. O react usa o JSX que é uma extensão opcional para o JavaScript e permite que combine o HTML com a linguagem com o JavaScript.

Outro assunto que será necessário aprofundar para fazer este projeto é trabalhar com banco de dados, e dentre tantas opções escolhemos o Firebase do Google, que é uma plataforma que oferece vários serviços, inclusive banco de dados. O Firebase oferece dois tipos de banco de dados o Realtime e o Firestore, ambos são muito similares, há uma pequena diferença é no quesito de multi-regiões e enhanced querying, que o Firebase Realtime não possui essas opções.

Outro assunto de muita importância para este projeto é o conhecido streaming, como objetivo do aplicativo é streaming de vídeos educacionais este assunto também necessita ser aprofundado. Streaming é a transmissão de dados via internet, geralmente o que é transmitido são vídeos e músicas. Mas além destes que serão de maior importância no trabalho, haverá outros assuntos que serão aprofundados, mas de forma geral estes são os mais importantes.

2.3 Plataformas Presentes no Mercado

Por José David Vargas de Melo Neto

As plataformas de ensino a distância revolucionaram as formas de ensinar e aprender no mundo inteiro. No Brasil esse novo modelo de educação já vem se

consolidando e tende a crescer cada vez mais. Por isso é necessário a utilização de boas plataformas onde professores possam disponibilizar suas aulas.

O YouTube é uma das plataformas de entretenimento mais conhecida do mundo, e a mesma tem sido muito utilizada para a transmissão de conteúdos educacionais através de vídeos que podem ser organizados em playlists de acordo com o assunto abordado. O Youtube EDU, criado em 2013, uma parceria entre o Google do Brasil e a Fundação Lemann foi fundamental para o crescimento de conteúdos educacionais na plataforma, que visa democratizar ainda mais o ensino no país.

Outra plataforma bastante conhecida no meio educacional é a Udemy que se tornou uma das plataformas digitais mais populares no mundo inteiro. O conteúdo dos cursos é bem variado com cursos de autoajuda, teoria musical e cuidar de animais domésticos à programação (da básica à avançada), análise de dados, design, vendas e outros mais com intuito profissionalizante, alguns gratuitos e outros pagos.

As plataformas citadas acima são alguns exemplos de meios utilizados para a transmissão de conteúdo educativos. Seguindo esta mesma tendência o nosso projeto será uma plataforma educativa, simples e de baixo custo. Todo o conteúdo será disponibilizado em forma de vídeo aula e o usuário poderá buscar o assunto que desejar.

3 CRONOGRAMA

Por Fábio Campos Ferreira

A data de início deste projeto se deu no dia 22 de outubro de 2020 e foi a partir desta em que foi montado o cronograma para a sua elaboração, com o término para a finalização e entrega do produto final no dia 23 de dezembro de 2020. Este período de construção do projeto tem ao todo nove semanas, as quais foram usadas como referência para estabelecer a finalização das etapas que juntas resultam na conclusão do projeto. A sua construção foi dividida em nove etapas, que serão descritas a seguir.

A primeira etapa, definida como E1, é referente aos primeiros passos do projeto. Sendo eles o planejamento do cronograma, a construção das interfaces ou páginas que irão constituir o site do servidor IPTV, as ferramentas que irão ser usadas no

decorrer deste projeto, a construção do fluxograma de processos e a organização de tarefas a ser distribuídas aos integrantes da equipe responsável pelo projeto.

A segunda etapa, definida como E2, tem como foco o início da construção do site. Onde, primeiramente, faz-se adição da opção de ao usuário de identificar-se quando dentro do site, para isso, deve ser construído um banco de dados para salvar as contas criadas e buscá-las novamente quando o usuário quiser entrar novamente no site. Além do banco de dados deve ser criada a página onde o usuário conseguirá criar sua conta ou fazer o login, caso já tenha criado.

A terceira etapa, definida como E3, irá se focar no princípio da manipulação dos vídeos contidos no servidor. Através da criação do banco de dados dos vídeos disponíveis, nesta etapa também será criada a página principal, a qual o usuário será redirecionado logo após o login, e nesta o usuário irá poder procurar vídeos contidos no banco de dados por seu título.

A quarta etapa, definida como E4, responsabilizará pela criação da página de pesquisa que aparecerá para o usuário assim que ele fizer uma pesquisa pelo título do vídeo. Consequentemente, também deverá ser implementado nesta etapa a função de filtragem, pertencente a página de pesquisa, com o objetivo de facilitar a busca do vídeo que o usuário espera encontrar no servidor.

A quinta etapa, definida como E5, será utilizada para a criação da página de reprodução e upload de vídeos. Referente a primeira, é exibida ao usuário após, o mesmo, clicar sobre um vídeo, podendo poder assistir, nesta página que deverá ser criada, o vídeo selecionado. Com relação a segunda página, a página de upload dos vídeos, acessada pelo usuário pela página principal, possuindo a opção concedida ao utilizador desta de inserir um novo vídeo no servidor.

A sexta etapa, definida como E6, é vista como a última etapa necessária para se obter um protótipo com as principais funcionalidades estabelecidas anteriormente no planejamento do projeto. Nesta etapa são criadas as páginas de verificação de vídeos e gerenciamento dos usuários. Nesta primeira, deve-se permitir que o administrador de um canal adicione ou exclua um vídeo que foi inserido através da página de upload dos vídeos. Na outra, referente ao gerenciamento de usuários obrigará o administrador a fazer uma pesquisa por contas que deverá ser feita pelo administrador, o qual terá opções necessárias para o gerenciamento das contas cadastradas.

A sétima etapa, definida como E7, foi reservada para os testes de funcionamento das funções adicionadas nas etapas anteriores e correção de possíveis falhas. Esta etapa, também será usada para realizar uma análise e descrição de novas opções que poderiam ser adicionadas ou retiradas com o objetivo de otimizar o projeto e aperfeiçoar a utilização do site que irá ser feita pelo usuário, para finalmente concluir o primeiro protótipo do site de gerenciamento e visualização dos vídeos de um servidor usando IPTV.

A oitava etapa, definida como E8 e subsequente a conclusão do primeiro protótipo, apresenta uma nova fase dentro do projeto, onde a sua estrutura básica está completa e os passos seguintes irão somente alterar o que foi feito até este momento e a inclusão de novas funções serão feitas de maneira pontual. Assim nesta etapa do projeto, serão definidos os pontos fracos do protótipo, tendo, na sequência desta etapa, a sua remoção dos mesmos.

A nona etapa, definida como E9, é o começo da mudança de foco do desenvolvimento, onde o processo e inclusão de novas funções é colocado em segundo plano, deixando como prioridade a criação e aperfeiçoamento da aparência do projeto como um todo, retirando o aspecto de protótipo e fazendo-o se tornar um produto pronto para a utilização. Tendo em mãos o produto finalizado, restará somente a finalização do relatório final, a gravação e postagem do vídeo sobre o desenvolvimento da construção de um site capaz de gerenciar um servidor de vídeos IPTV.

A Tabela 1 apresenta o cronograma de desenvolvimento do projeto, com as datas de conclusão de cada etapa.

Tabela 1 – Cronograma.

Etapa	Data para término
E1	29/10/2020
E2	25/11/2020
E3	12/11/2020
E4	19/11/2020
E5	26/11/2020

E6	03/12/2020
E7	10/12/2020
E8	17/12/2020
E9	24/12/2020

Fonte: Os autores.

4 MÉTODO

Por Fábio Campos Ferreira

A aplicação web com a finalidade de ser um serviço de servidor IPTV, apresentado todas as principais características deste protocolo, que é descrita como objetivo de construção neste relatório apresenta certas complexidades e utiliza-se de cartas tecnologias as quais é necessário um certo entendimento, com o objetivo na melhora da tomada de decisões do planejamento do projeto e no decorrer da construção do mesmo.

Para uma melhor compreensão deste projeto, foram empreendidos alguns passos de planejamento de projeto, para que o mesmo fosse feito seguindo todos estes passos, tendo uma construção de projeto mais clara e efetiva na etapa final de teste das funcionalidades do produto.

Desta forma foi projeto um fluxograma para exemplificar a sequência de ações que a aplicação poderá e deverá seguir. Quanto às ferramentas que deverão ser utilizadas, pode-se dizer que será necessário utiliza-las para a construção da aplicação web, para o serviço de hospedagem do site online, outra para o gerenciamento de banco de dados que serão utilizado para registrar os usuários e os vídeos, também é necessário um sistema de hospedagem para arquivar os vídeos que irão ser reproduzidos no site.

Pensando na escrita do código do projeto, foi desenvolvido um diagrama de classes, onde cada classe é uma tela do site, apresentando certas funções de cada página deve poder exercer. Em sua complementação, juntamente com esse diagrama, foi projeto a aparência que cada uma dessas telas terá quando este projeto for finalizado.

4.1 Fluxograma

Por Fábio Campos Ferreira

O planejamento referente ao fluxograma do projeto foi feito baseando-se em sistemas de sites populares como o YouTube e Netflix, que apresentam semelhanças com o objetivo principal deste projeto. Desta forma, foi construída uma base a qual foi pensada para ocasionar no caminho mais curto para o usuário do site possa realizar a tarefa desejada e que o site tem como objetivo disponibilizar.

Assim sendo, o fluxograma arquitetado, que pode ser visto na Figura 1, mostra todos os caminhos possíveis que a arranjo sistema irá propiciar ao usuário. Sua estrutura consegue cumprir todos os objetivos estabelecidos neste relatório, onde, primeiramente inicia a sua ação como abrindo a tela de login para que o usuário possa criar sua conta ou fazer o login, caso a mesma já tivesse sido criada, posteriormente é identificado, utilizando o banco de dados dos usuários, o tipo de usuário, podendo ser da categoria comum, administrador de um canal ou o administrador principal.

Se a pessoa for um usuário comum, logo é mostrada, para ela, a tela principal onde aparecem vídeos pré-selecionados, onde, tendo clicando em algum deles, ocorre um redirecionamento para a tela de reprodução. A partir da tela principal, também é possível realizar uma pesquisa, utilizando o banco de dados dos vídeos, que encaminhará o usuário para a página de pesquisa contendo os seus resultados e a opção de filtragem, facilitando a escolha de um vídeo. Ao clicar sobre no vídeo, o usuário é redirecionado para a tela de reprodução, onde será possível assistir o tal vídeo, bem como optar para adicionar algum outro vídeo, proveniente da máquina do usuário, o qual acha que o mesmo complementa a explicação do vídeo carregado na página de reprodução.

Se, no entanto, o usuário for administrador de um canal, é adicionado a ele na tela principal, além das opções que um usuário comum teria, a escolha de entrar em uma nova tela, a qual ele teria acesso aos vídeos que os usuários comuns recomendaram adicionar em seu canal. Nesta tela responsável pelo gerenciamento de vídeos, o administrador de um canal poderá rejeitar o vídeo e excluí-lo do banco de dados, ou poderá aprová-lo, adicionando-o na lista de reprodução do seu canal.

A última classe de usuário são os administradores principais, não sendo responsáveis por um canal, mas sim, pelas outras contas do website. A este é adicionado uma opção na tela principal que os possibilita a entrar na página de

gerenciamento dos usuários. Esta colocará à disposição do administrador principal uma ferramenta de pesquisa para encontrar usuários, e quando estes forem encontrados, poderá excluí-los do banco dos dados. De forma antagônica, o administrador poderá inserir novas contas manualmente nesta página.

Como pode ser observado na Figura 1, o usuário poderá seguir um caminho direto para sair do site estando em qualquer tela. Caso esteja logado, também poderá seguir uma via imediata para se direcionar para a página principal, ou a página de gerenciamento dos vídeos, caso seja um administrador principal, ou para a página de gerenciamento dos usuários, no caso de um administrador principal. Esta mesma característica é vista no processo de logout.

Figura 7. Diagrama de fluxo

```

graph TD
    Inicio([Início]) --> Entrar[Entrar no site]
    Entrar --> L((L))
    L --> TelaLogin([Mostrar tela de login])
    TelaLogin --> InfoUser[Informações do usuário]
    InfoUser --> Dec1{Usuário é registrado?}
    Dec1 -- Não --> Criar[Criar conta]
    Criar --> U((U))
    U --> L
    Dec1 -- Sim --> Identificar[Identificar Usuário]
    Identificar --> Dec2{É administrador de um canal?}
    Dec2 -- Não --> Carregar[Carregar vídeos recomendados]
    Carregar --> M((M))
    M --> TelaPrincipal([Mostrar tela principal])
    M --> Pesquisar[Realizar uma pesquisa de vídeos]
    Dec2 -- Sim --> BotaoGer[Mostrar botão de gerenciamento de vídeos em todas telas]
    BotaoGer --> P((P))
    P --> ClicarVideo[Clicar sobre um vídeo]
    P --> Filtragem[Realizar uma filtragem]
    Filtragem --> TelaPesquisa([Mostrar tela de pesquisa])
    ClicarVideo --> TelaReprod([Mostrar tela de reprodução])
    TelaReprod --> Reproduzir[Reproduzir vídeo]
    Reproduzir --> V1((V))
    V1 --> Enviar[Enviar um vídeo]
    Enviar --> TelaUpload([Mostrar tela de upload de vídeos])
    TelaUpload --> UV((UV))
    UV --> TelaReprod
    UV --> TelaGer[Mostrar tela de gerenciamento de usuários]
    TelaGer --> GU((GU))
    GU --> Incluir[Incluir usuário]
    GU --> ExcluirUser[Excluir usuário]
    GU --> PesquisarUser[Pesquisar usuário]
    Incluir --> L
    ExcluirUser --> L
    PesquisarUser --> L
    TelaGer --> A((A))
    A --> SS((SS))
    A --> M
    SS --> CarregarUser[Carrega vídeos enviados por usuários]
    CarregarUser --> GV((GV))
    GV --> Aprova[Aprova vídeo]
    GV --> ExcluirVideo[Excluir vídeo]
    ExcluirVideo --> TelaGer
    Aprova --> V2((V))
    ExcluirVideo --> V2
    V2 --> SairConta[Clicar no botão de sair da conta]
    SairConta --> L
    L --> SairSite[Sair do site]
    SairSite --> Fim([Fim])
  
```

4.2 Ferramentas

Para poder executar o projeto, primeiramente necessitamos de um computador, seja de mesa ou notebook, tendo um sistema operacional capaz de receber o editor de códigos Visual Studio Code, que foi desenvolvido pela Microsoft e lançado em 2015, além de editor de códigos, também é considerado um IDE ou depurador. Este editor suporta vários tipos de linguagens, tais como Python, C, C++,

C#, F#, TypeScript e JavaScript, e além de ter realce de sintaxe, que auxilia no processo de programação.

Falando novamente das linguagens, como mencionado anteriormente nesse documento, para implementar o projeto escolhemos a linguagem de programação JavaScript (JS), esta linguagem é interpretada e estruturada de script em alto nível com tipagem dinâmica fraca e multiparadigma. É uma linguagem que permite que as páginas da web fiquem mais interativas. O JS foi desenvolvido como parte dos navegadores para que scripts fossem executados do lado do cliente e não necessitasse passar pelo servidor. Então se tornou uma linguagem importante para aplicação da web.

Trabalhando em conjunto com o JS o Node.js é um software que executa o código JS no nível backend e frontend. A vantagem do Node.js é que na execução ele necessita apenas de um thread, e isso economiza recursos computacionais.

O banco de dados utilizado para armazenar tanto os dados quanto os vídeos enviados pelos usuários será o Firebase. O Firebase é uma plataforma de desenvolvimento web e mobile adquirida pela Google em 2014. Com foco em ser um back-end completo e de fácil usabilidade, essa ferramenta disponibiliza diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicativos. Os serviços oferecidos pela ferramenta que vamos utilizar para a implementação do projeto são: o authentication, que possibilita a autenticação através de contas do Google, Facebook, Twitter, Github ou um sistema de conats próprio; o Storage, que é responsável por armazenar dados do usuário através de uma conexão segura e permite o compartilhamento dos mesmos; e por último o Realtime Database, que é o banco de dados que sincroniza os dados com os dispositivos em tempo real. Regras de segurança podem ser configuradas para definir quem tem acesso a quais dados.

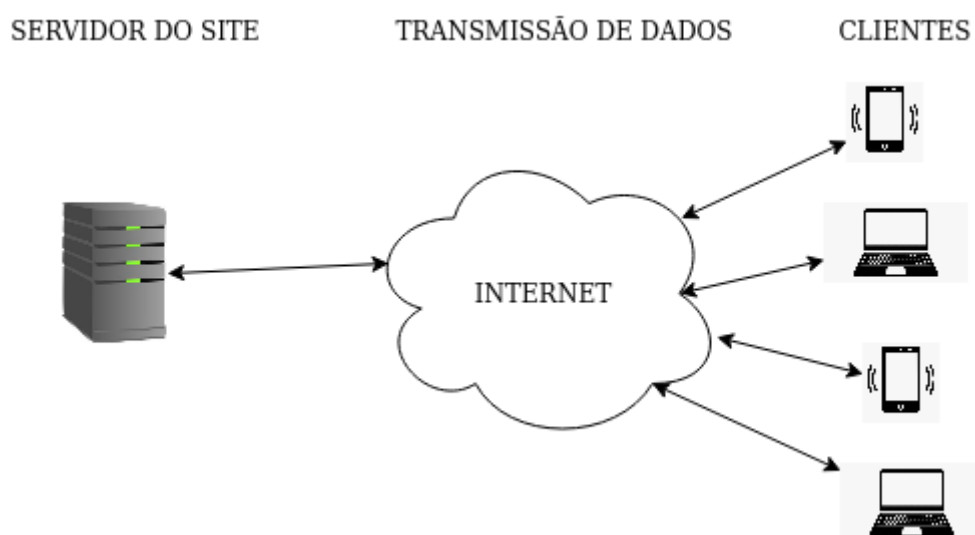
4.3 Protocolos

Por Fábio Campos Ferreira

O objetivo principal deste projeto é criar uma plataforma para o compartilhamento de vídeos educacionais, onde os usuários poderão acessar o site do projeto através de um navegador do celular ou de um computador. Dentro do sistema, eles podem escolher alguns vídeos para assistir, ou ainda, existe a possibilidades de contribuição

para aumentar o acervo do conteúdo da plataforma, ao enviar para servidor novos vídeos. A Figura 2 apresenta o vínculo da transmissão de dados em duplo sentido entre o cliente e servidor.

Figura 2 – Comunicação servidor-cliente.



Fonte: Os autores

A internet será o meio de transporte, o qual será utilizado para realizar a comunicação entre o dispositivo, pertencente a um cliente, e recursos da plataforma, aqui planejada, presente no servidor. Neste tipo de comunicação existem protocolos, com a funcionalidade de implementar essa transferência de arquivos da forma mais eficiente possível, e minimizando as perdas de tempo, devido aos processamentos interno da rede. No caso do projeto descrito, é necessário dar um grau de prioridade maior aos protocolos referentes ao envio de arquivos do fluxo da mídia a ser reproduzida pelo usuário, sendo os mais problemáticos arquivos no processo de transmissão, devido ao seu grande tamanho e necessidade do menor tempo possível de carregamento com a maior qualidade, sendo o objetivo do site armazenar vídeos e enviá-los a vários usuários para serem exibidos.

Os principais protocolos e tecnologias utilizados em um serviço de IPTV e transmissão de vídeos são o MPLS, IP, IGMP, TCP, UDP, RTP e o RTSP. O formato de compactação de vídeo MPEG4, também é essencial para uma melhor qualidade de transmissão, sendo o principal utilizado, juntamente com esses protocolos.

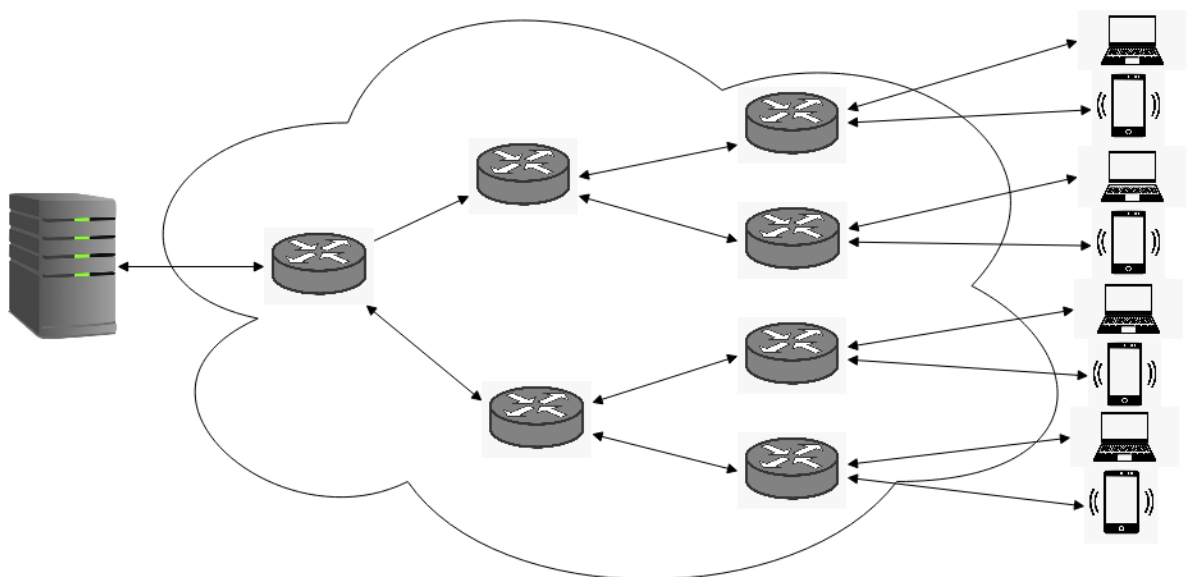
O primeiro protocolo que deve ser citado é o MPLS (Multi Protocol Label Switching). Arquivos enviados pela internet são descritos como pacotes, a função do MPLS é adicionar um rótulo a este pacote com as informações sobre o caminho mais curto a ser percorrido pelo pacote com o destino final, um endereço IP, representando uma máquina. Este protocolo é usado para cumprir os pré requisitos da qualidade de serviço (QoS), esta, se refere à garantia da largura de banda utilizada numa transmissão ou na probabilidade de êxito que um pacote de informações possui ao completar a sua circulação entre dois pontos de uma rede (IPTV, 2020) (XIAO; DU; ZHANG; HU; GUIZANI, 2007) (WEINBERG; JOHNSON, 2018) (FATI; AZAD; PATHAN, 2018).

O cabeçalho do protocolo MPLS também tem a informação do último roteador o qual esse pacote passará para chegar ao seu destino e quantos saltos ele pode realizar antes que este processo resulte no seu descarte. Este rótulo tem o tamanho de 20 bits e substitui os endereços IPs mais longos. Além disso, ele propicia trocas de rótulos dos pacotes mais rapidamente comparado ao método que utiliza uma busca nas tabelas IP routing, com o objetivo de definir seu próximo roteador de destino. A escolha do melhor caminho é feita observando, além da menor distância, a largura de banda que cada rota dispõe. Este protocolo é considerado uma forma conexão segura, embora existam outras mais eficientes neste parâmetro, já que os dados não são criptografados. O protocolo apresenta uma boa resposta em aplicações de transmissão dos dados em tempo real. Os rótulos dos pacotes MPLS não indicam o tipo do formato do arquivo contido no pacote. As características das redes MPLS contribuem para serviços de IPTV como também a transmissão de vídeo, ajudando a atender os requisitos do QoS (IPTV, 2020) (XIAO; DU; ZHANG; HU; GUIZANI, 2007) (WEINBERG; JOHNSON, 2018) (FATI; AZAD; PATHAN, 2018).

O IP (Internet Protocol) é o responsável por assegurar que o pacote chegue a máquina correta, que possui o endereço especificado anteriormente na requisição do pacote. O IGMP (Internet Group Management Protocol) é utilizado quando muitos clientes estão requisitando o recebimento do mesmo pacote. Se não utilizado, é criado o envio simultâneo do mesmo pacote repetidas vezes para o mesmo roteador, gerando tráfego de dados redundantes e falta de otimização da rede. Para evitar este evento, o pacote é enviado para o próximo roteador uma única vez, este repassa o pacote para os próximos roteadores até chegar no qual existe uma bifurcação, onde

é necessário duplicar o pacote para seguir a mais de um roteador ao mesmo tempo, este processo é feito até o pacote chegar a um cliente, atendendo a vários ao mesmo tempo, criando uma árvore de transmissão, este tipo de conexão é chamado de Multicast. Quando um novo usuário deseja entrar nesta rede, ele envia um pacote IGMP de solicitação, que se aceita, é localizado o roteador mais próximo que participa da árvore de transmissão, este será responsável por adicionar mais um ramo de transmissão da árvore que atenderá este novo cliente. A Figura 3 exemplifica esta ideia de árvore usada pelo protocolo IGMP (IPTV, 2020) (XIAO; DU; ZHANG; HU; GUIZANI, 2007) (FATI; AZAD; PATHAN, 2018).

Figura 3 – Árvore de transmissão de um grupo multicast IP.



Fonte: Os autores.

O TCP (Transmission Control Protocol) é pertencente a camada de transporte. Quando um pacote é perdido ou corrompido, este protocolo o reenvia evitando que qualquer informação requisitada seja perdida. O TCP também realiza um gerenciamento de tráfego, evitando que o canal entre em congestionamento, desta forma, provocando uma redução na velocidade de transmissão. Porém, ele não é indicado ao envio de um fluxo dos dados referentes a um vídeo, já que a transmissão do mesmo tem que ser constante e rápida, o que é prejudicado pela retransmissão de dados, que também provoca um aumento na probabilidade de perda da transmissão

do vídeo ou gerar uma perda de qualidade da imagem. Desta forma, o TCP é mais indicado na transmissão de informações referentes a outras funcionalidades de um sistema IPTV (IPTV, 2020) (XIAO; DU; ZHANG; HU; GUIZANI, 2007).

O UDP (User Datagram Protocol) é o mais indicado para a transmissão relacionado a arquivos de vídeo, já que não emite um aviso quando um pacote é perdido, proporcionando um cabeçalho é menor, reduzindo o tempo perdido no processamento e evitando falhas causados pelo processo de retransmissão dos dados. Apesar não re-enviar pacotes perdidos, esta característica do protocolo pode ser tolerada em função do ganho de velocidade. Portanto o UDP é indicado ficar com a transmissão de vídeo e o TCP transmite as informações restantes da aplicação, podendo atuar, os dois protocolos, de forma simultânea (IPTV, 2020) (XIAO; DU; ZHANG; HU; GUIZANI, 2007).

O RTP (Real-time Transport Protocol) baseia-se no UDP para realizar o transporte de dados na rede, sendo mais utilizado para o transporte de dados no formato MPEG4, estes são dados de vídeo e áudio, este protocolo permite a transmissão deste em tempo real para redes Unicast ou Multicast (NFON, 2020). O RTSP (Real Time Streaming Protocol) é a união e utilização dos protocolos UDP, TCP e RTP, com a função de transmissão via streaming, ou seja, dados que são gerados e logo transmitidos como um fluxo contínuo. O tipo de conexão que utiliza o RTSP não suporta todos os tipos de arquivo, também não é recomendado para transmissões de vídeos que não podem sofrer perdas de qualidade, pois este protocolo é vulnerável ao extravio de pacotes, bem como atrasos na transmissão e congestionamentos da linha. Porém, oferece uma boa resposta a respeito do envio a um grupo de pessoas grande, independente da distância entre as mesmas (AVSILLC, 2020).

O MPEG4 (Moving Picture Experts Group) é um formato de compactação do vídeo, que consegue reduzir tamanho da mídia, encurtando a largura de banda utilizada, e otimizando o envio de vídeos por meio da criação da estrutura GoP, ou grupos de imagens, representando um fluxo do vídeo de alguns segundo e que são estes os enviando pela rede até a máquina de destino (XIAO; DU; ZHANG; HU; GUIZANI, 2007) (FATI; AZAD; PATHAN, 2018).

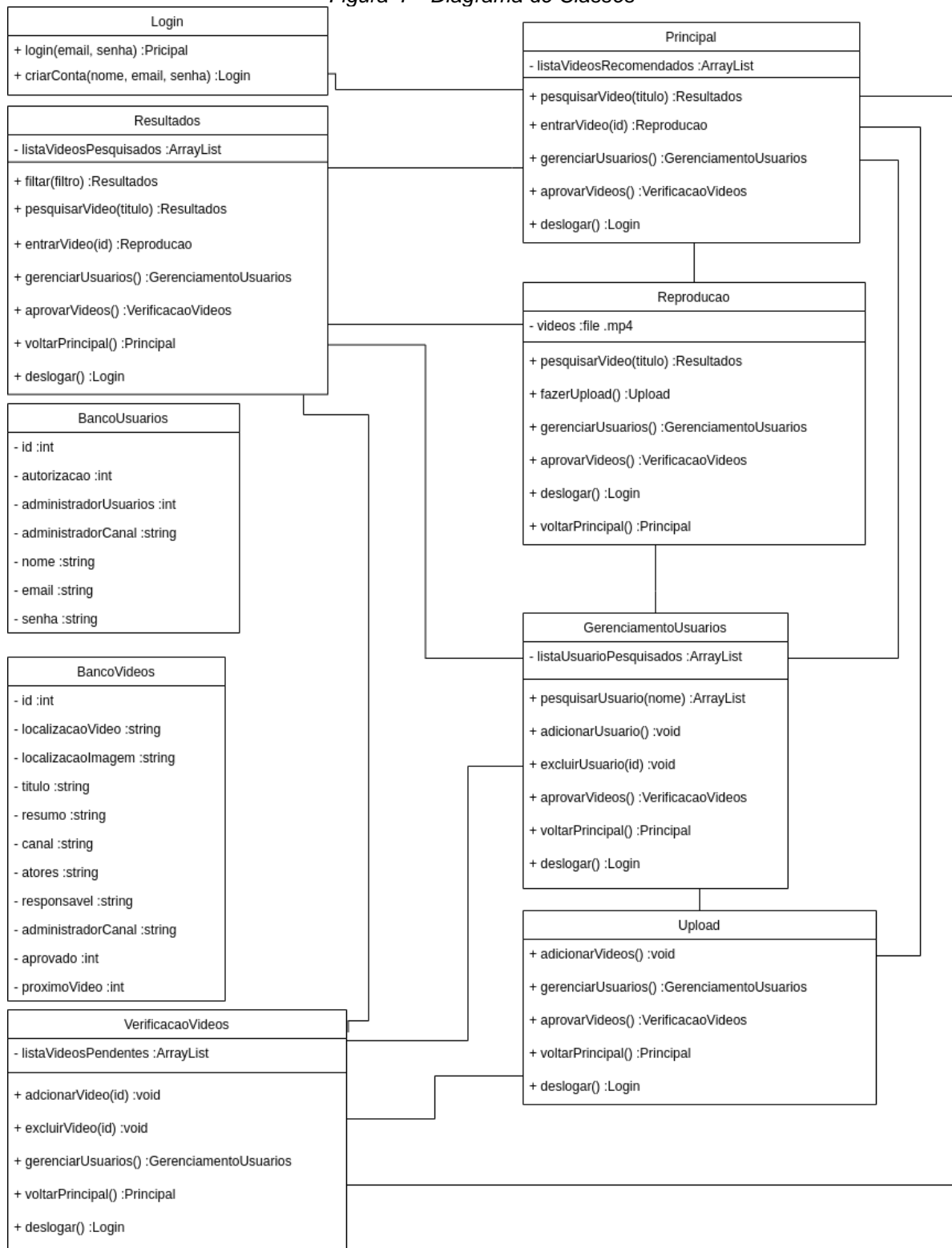
4.4 Diagrama de Classes

Por Fábio Campos Ferreira

O diagrama de classes da aplicação web, presente na Figura 4, apresenta todas as telas que o site deverá possuir. No diagrama, cada tela possui um grupo único de funções, que poderiam ser chamadas quando o usuário estiver navegando na referida tela. Também é possível existir uma variável, podendo ser uma lista dos vídeos relevantes ao usuário ou de usuários, os quais serão usados para preencher o conteúdo da tela.

No diagrama de classes, também estão presentes os bancos de dados dos usuários e de vídeos, com seus respectivos itens que irão caracterizar cada elemento do banco. As telas dos sites são sete ao todo, sendo elas: “Login”, “Principal”, “Resultados”, “Reproducao”, “GerenciamentoUsuario”, “Upload” e “VerificacaoVideos”.

Figura 4 – Diagrama de Classes



Fonte: Autores.

A tela “Login” apresenta duas funções. A função “login()” recebe o e-mail do usuário e sua senha para buscá-los no banco de dados, identificando-o durante a

utilização do restante da aplicação. A função “criarConta()” recebe o nome de usuário, e-mail e senha para cadastrar um novo cliente no banco de dados.

A tela “Principal” apresenta uma lista e cinco funções. A lista “listaVideosRecomendados” é do tipo “ArrayList”, apresentando uma lista dos vídeos recomendados para serem exibido nesta página. A função “entrarVideo()” é chamada quando o usuário clica sobre um vídeo, então ela recebe o “id” do vídeo clicado e chama a tela “Reprodução”, na qual é possível assistir o tal vídeo. A função “pesquisarVideo()” recebe a string digitada na barra de busca e pesquisa no banco de dados, vídeos tendo títulos parecidos com a palavra digitada. A função “gerenciarUsuarios()” é chamada quando o usuário clica sobre o botão correspondente, ela chama a tela “GerenciamentoUsuarios”. A função “aprovarVideos()” é chamada quando o usuário clica sobre o botão correspondente, ela chama a tela “VerificacaoVideos”. A função “deslogar()” realiza a perda de informações do usuário na seção atual, consequentemente ele é redirecionado para a tela “Login”.

A tela “Resultados” apresenta uma lista e sete funções. A lista “listaVideosPesquisados” é uma ArrayList e representa todos os vídeos que corresponderam a pesquisa realizada, estão estes são mostrados na tela. A função “filtrar()” recebe os tipos de filtros selecionados, os quais serão utilizados para buscar os vídeos que satisfazer os critérios dos filtros, esses pertencentes a lista “listaVideosPesquisados”. As funções “entrarVideo()”, “pesquisarVideo()”, “gerenciarUsuario()”, “aprovarVideos()” e “deslogar()” são as mesmas descritas anteriormente. A função “voltarPrincipal()” é chamada quando o usuário clica sobre o botão correspondente, ela chama a tela “Principal”.

A tela “Reproducao” apresenta o arquivo vídeo carregado, pronto para a sua reprodução e mais sete funções. A função “fazerUpload()” é chamada quando o usuário clica em cima de um respectivo botão presente na tela, ela irá redirecionar para a tela “Upload”. As funções “voltarPrincipal()”, “pesquisarVideo()”, “gerenciarUsuario()”, “aprovarVideos()” e “deslogar()” são as mesmas descritas anteriormente.

A tela “GerenciamentoUsuarios” apresenta uma lista e seis funções. A lista de nome “listaUsuario” é uma ArrayList, contendo os usuários que correspondem a pesquisa, eles iram ser mostrados na tela. A função “pesquisarUsuario()” pesquisa no

banco de dados os usuários com nomes correspondentes a palavra digitada na barra de pesquisa, retornado a “listaUsuario” com os “id” dos usuários encontrados. A função “adicionarUsuario()” adiciona um novo usuário manualmente, com todos os parâmetros do banco de dados preenchidos e sem a necessidade de realizar o processo de criação da conta na tela de login. A função “excluirUsuario()” recebe um “id”, procura-o no banco de dados e o exclui dos registros. As funções “voltarPrincipal()”, “aprovarVideos()” e “deslogar()” são as mesmas descritas anteriormente.

A tela “Upload” apresenta cinco funções. A função “adicionarVideo()” recebe todos os parâmetros que constituem um item do banco de dados dos vídeos de forma a estarem todos preenchidos, além de receber o arquivo do vídeo e salvá-lo no servidor. As funções “voltarPrincipal()”, “gerenciarUsuario()”, “aprovarVideos()” e “deslogar()” são as mesmas descritas anteriormente.

A tela “VerificacaoVideos” possui uma lista e cinco funções. A lista “listaVideosPendentes” é uma ArrayList, contendo todos os vídeos com o item “aprovado” igual a 0 e com o nome do item “canal” igual ao elemento “administradorCanal” do usuário. A função “adicionarVideo()” recebe o “id” do vídeo a ser aprovado e muda o valor “aprovado”, deste vídeo, para 1, também muda o valor “proximoVideo” do vídeo que está listado na lista de reprodução do canal logo anterior a este para o mesmo valor “id” do vídeo recém adicionado, com o objetivo de atualizar esta lista. A função “excluirVideo()” recebe o “id” do vídeo que não está aprovado, o exclui-o do banco e apaga o arquivo do servidor. As funções “voltarPrincipal()”, “gerenciarUsuario()” e “deslogar()” são as mesmas descritas anteriormente.

O banco de dados referente aos usuários, “BancoUsuarios”, será usado para gerenciar os usuários, realizando seu cadastro, exclusão ou login. Para isso, tem-se o elemento “id”, sendo do tipo inteiro, é usado para localizar um elemento no banco, sendo um número diferente para cada usuário. A propriedade “autorizacao” recebe valor 0 ou 1, sendo responsável por autorizar um usuário a navegar pelo site, neste caso o valor é 1. Na propriedade “autorizacao”, na criação de uma nova conta, o valor predefinido é 0, devendo ser mudado para 1 pelos administradores de usuários na tela “GerenciamentoUsuarios”. O elemento “administradorUsuario”, sendo do tipo inteiro, irá receber somente os valores 0 ou 1, sendo responsável por identificar se a conta que está logada é um administrador principal, se sim seu valor é 1. O elemento

“administradorCanal”, é uma string, apresenta o nome do canal que o usuário é responsável, se não tiver direito de administração para nenhum canal, a string é vazia. O elemento “nome”, sendo uma string, representa o nome de usuário da pessoa cadastrada. O elemento “e-mail”, sendo uma string, representa o e-mail do usuário e é usado para realizar o login e para suceder futuros contatos. O elemento “senha”, sendo uma string, é a senha utilizada no login pelo usuário.

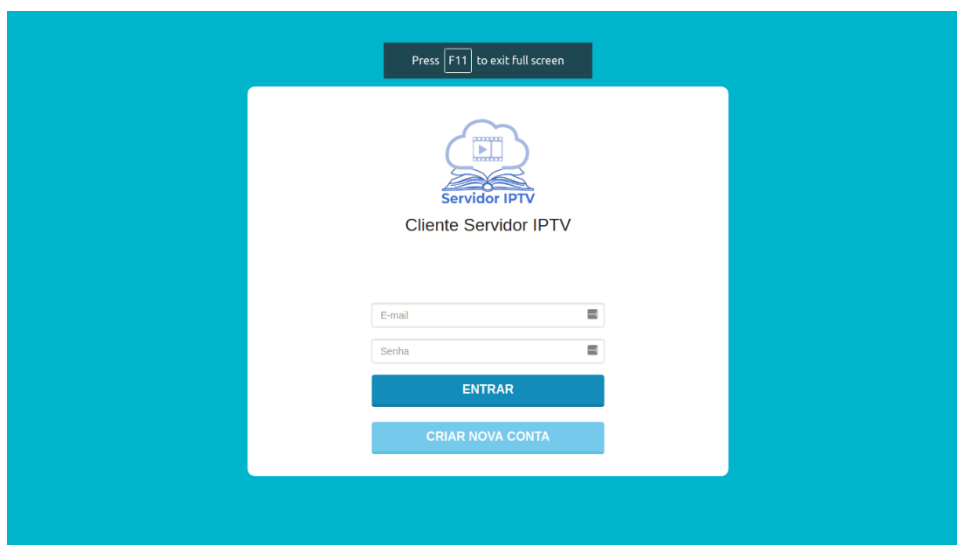
O banco de dados utilizado para o gerenciamento de vídeos é identificado pelo nome “BancoVideos”. Cada vídeo registrado no canal representa um elemento deste banco. Para localizar os vídeos e gerenciá-los durante a execução da aplicação, cada vídeo terá elementos para a sua descrição. O primeiro é a variável “id”, é um inteiro usado para identificar o vídeo, ele nunca será repetido dentro do banco de dados. O elemento “localizacaoVideo” é uma string, sendo o caminho do diretório onde está arquivado o vídeo. O elemento “localizacaoImagem” é uma string, sendo o caminho do diretório onde está arquivado a imagem de apresentação do vídeo, a thumbnail. O elemento “titulo” é uma string, sendo o título do vídeo. O elemento “resumo”, sendo uma string, é o resumo do seu conteúdo. O elemento “canal”, sendo uma string, apresenta o nome do canal a qual aquele vídeo pertence. O elemento “atores” é uma string e apresenta os nomes das pessoas que aparecem no vídeo. O elemento “responsável”, sendo uma string, identifica o nome de usuário da pessoa que enviou esse vídeo para o servidor. A variável “administradorCanal” é uma string representando o nome do usuário que é responsável pelo canal e que aprovou a inclusão do referido vídeo. O elemento “aprovado” é um inteiro de valor 0 ou 1, é 0 quando o administrador do canal ainda não aprovou este vídeo para ser incluído na playlist do canal. O elemento “proximoVideo”, sendo um inteiro, contém o item “id” do próximo vídeo a ser reproduzido após o atual.

4.5 Interface

Por Fábio Campos Ferreira

O site terá sete interfaces, a “Login”, presente na Figura 5, a “Principal”, presente na Figura 6, a “Pesquisa”, presente na Figura 7, a “Reprodução”, presente na Figura 8, a “Upload”, presente na Figura 9, a “GerenciamentoUsuarios”, presente na Figura 10 e finalmente a “VerificacaoVideos”, presente na Figura 11.

Figura 5 – Interface “Login”



Fonte: Os autores.

Por Tiago Carneiro

A montagem da página principal foi feita durante esse período de uma semana, todo o layout e caminhos das páginas foram feitos. Para a próxima etapa do trabalho será implementado as funções pertinentes desta página, tais como buscas, e gerenciamento de forma geral.

Continuando com o projeto, esta semana foi feito algumas funções da tela principal, com o grau de dificuldade um pouco elevado, foi implementado uma busca de vídeos já interligada com o banco, mas ainda não está mostrando o resultado na tela. Para implementar estas funções foi modificado boa parte do código, isso foi feito para que o Java Script aceitasse a função. Para a próxima semana é esperado que seja feito o retorno da busca na própria página e algumas outras melhoras no código, tipo trazer a tela de recomendações personalizadas, mas isto depende do andamento dos objetivos principais.

Dando sequência ao projeto, nesta semana implementamos mais algumas funções no aplicativo, em específico na página principal implementou as funções para organizar os vídeos de forma automática, ou seja, quando o usuário abre a página principal, já aparece vídeos diferentes, todos as informações do vídeo vem do banco de dados, incluindo a imagem de exibição, para o trabalho a ideia principal é de trazer os vídeos de acordo com a data que o vídeo foi inserido no banco de dados, mas futuramente é possível pensar em trazer os vídeos recomendados, isso aproveitando

o histórico do usuário. Lembrando que todo esse processo está ligado diretamente ao banco de dados do aplicativo. Outra função que foi implementada, foi a função de interligar a página principal com a página de pesquisa, assim o usuário digita o que deseja buscar e quando clica no ícone de busca, ele é direcionado para a página de pesquisa com o resultado da busca, a figura mostra a parte do código que faz esta conexão.

Figura 6 - Conexão entre pagina principal e página de pesquisa.

```
buscaVideo = () => {  
  localStorage.textoVindoTelaPrincipal = document.getElementById("tituloInput").value;  
}  
  
<div className={classes.search}>  
  <input type="text" className={classes.searchTerm} id='tituloInput' placeholder="O que você esta procurando?" />  
  <Link type="submit" to="/pesquisa" className={classes.searchButton} onClick={this.buscaVideo}>  
    <i className="fa fa-search"></i>  
  </Link>  
</div>
```

Fonte: Autores

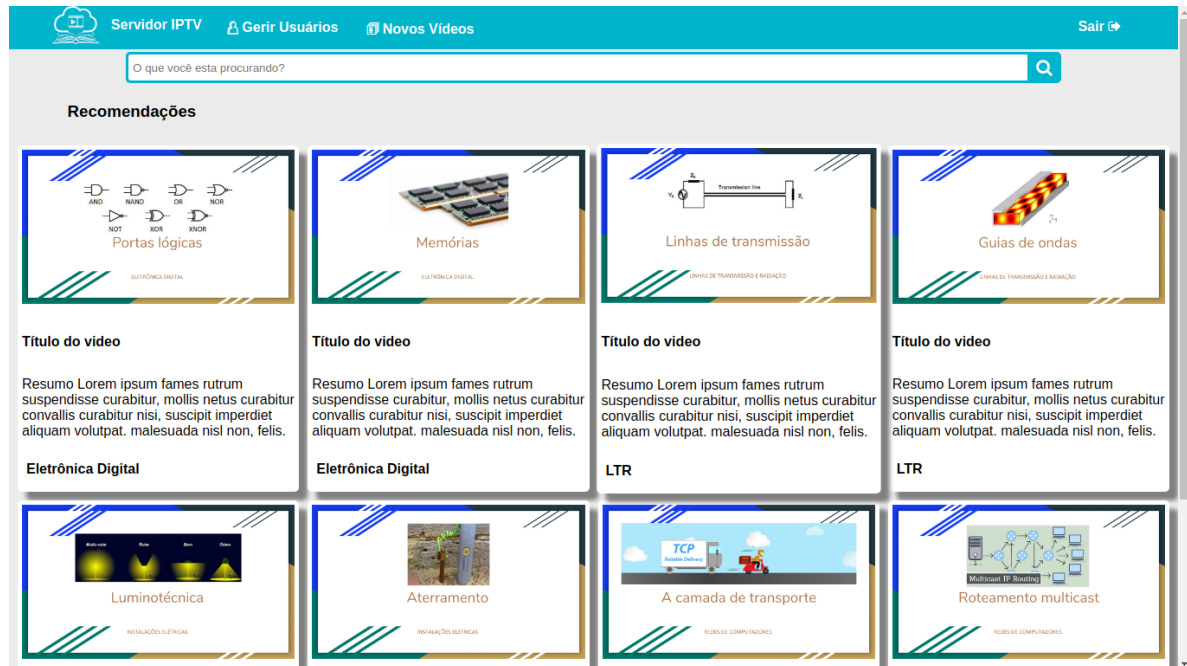
Outro detalhe importante adicionado na página principal, foi que, quando o usuário escolhe um vídeo que está na seção de recomendados, ele é direcionado para a tela de reprodução para poder assistir ao conteúdo. Além destes detalhes foi melhorado alguns outros detalhes, para o melhor funcionamento do aplicativo. A figura a seguir mostra o código utilizado para fazer a ligação com a tela de reprodução.

Figura 7 - Passando qual vídeo foi selecionado na tela principal para a tela de reprodução.

```
assistirVideo = (chaveVideo) => {  
  localStorage.setItem('telaReproducaoVideo', JSON.stringify(chaveVideo));  
}  
  
<Link onClick={() => this.assistirVideo(k)} to="/reproducao" className={classes.espacoVideo}>
```

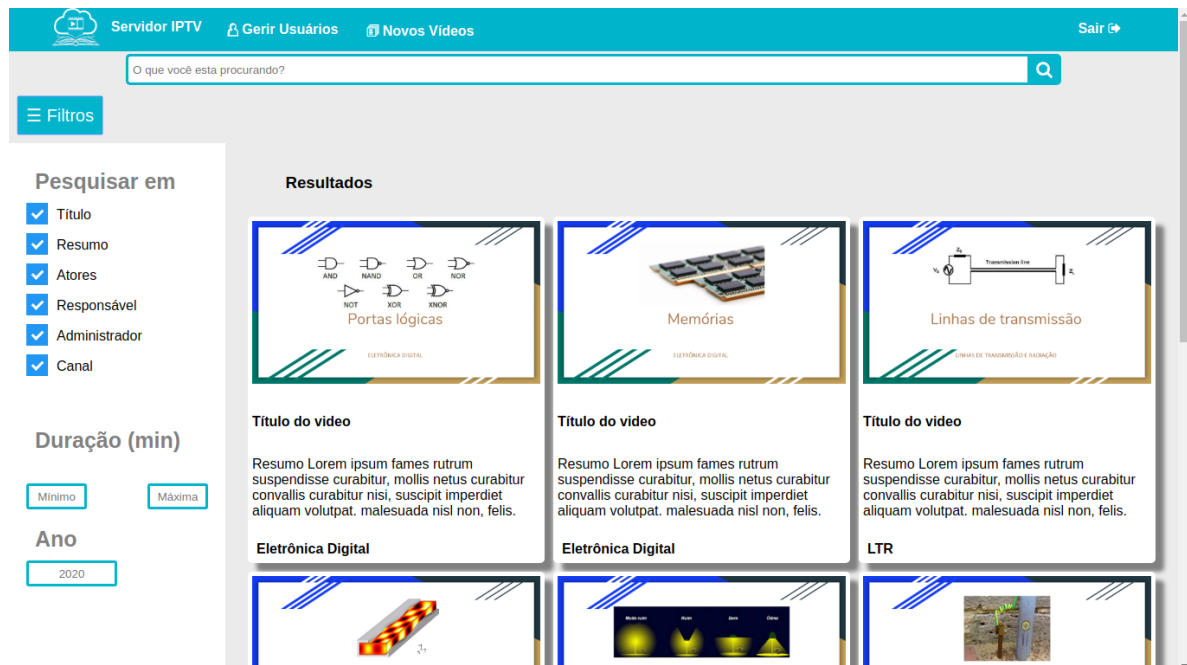
Fonte: Autores.

Figura 8 – Interface “Principal”.



Fonte: Autores.

Figura 9 – Interface “Pesquisa”.



Fonte: Autores.

Por José David

A página de reprodução também foi montada no período de 1 semana, e seu layout pode ser visto na figura 10. Todas as suas funcionalidades já estão pré-programadas, sendo necessário agora realizar sua conexão com o banco de dados.

Dando continuidade ao projeto da interface de reprodução, nesta semana final foram implementadas todas as funcionalidades. O usuário ao clicar em um vídeo na tela principal ou na tela de pesquisa é redirecionado para a tela de reprodução onde o vídeo será reproduzido. Esta interface também tem a funcionalidade de sugerir um próximo vídeo para o usuário, com base no assunto do vídeo que está sendo exibido. Para implementar o algoritmo responsável por indicar o próximo vídeo tivemos uma certa dificuldade por estarmos fazendo de uma forma mais complexa. O próximo vídeo é buscado com base no título dos vídeos que está sendo reproduzido, ou seja, é feita uma busca no banco de dados com base em palavras chaves semelhantes presentes nos títulos dos vídeos. Os administradores terão acesso a todas as funcionalidades desta página, enquanto usuários comuns serão restringidos a alguns em específico.

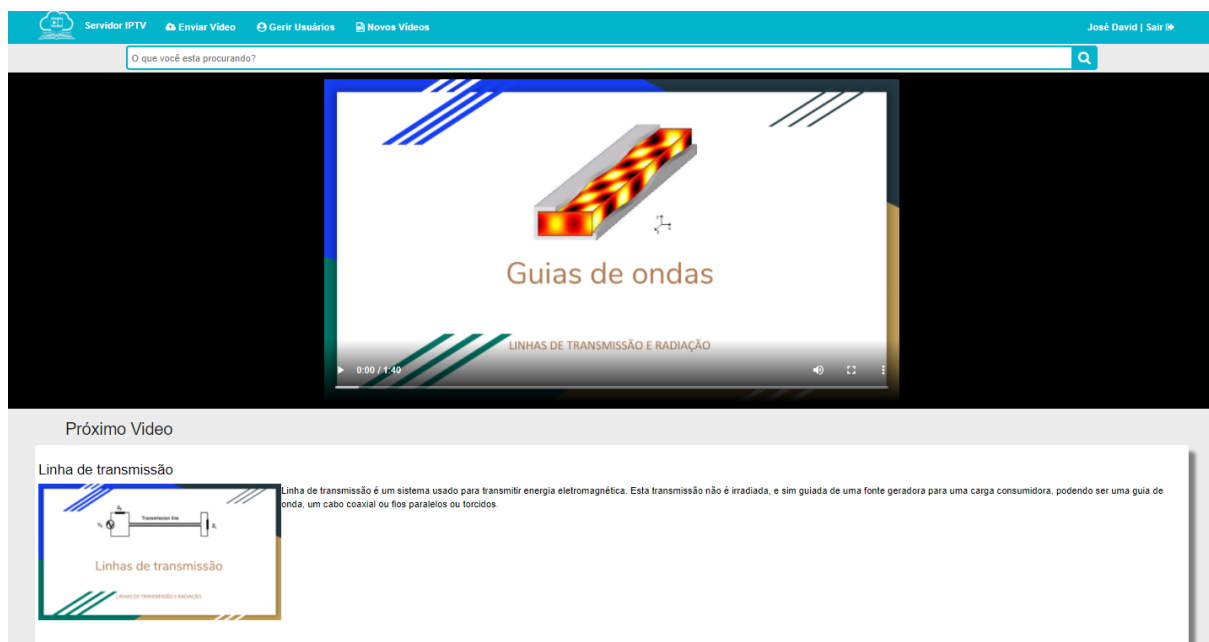
A função comentada acima responsável por buscar um próximo vídeo é exibida na figura 10 logo abaixo. Ainda serão feitas algumas modificações para aprimorar a interface de reprodução.

Figura 10 - Função responsável por chamar o próximo vídeo.

```
adicionarProximoVideo = ([video, chave]) => {
  console.log(video);
  const proximoVideo = (
    <div className={classes.espacoVideo} onClick={this.redirectToNextVideo(chave)} style={{ width: "95%", display: "block", marginLeft: "auto", marginRight: "auto" }}>
      <h4 style={{ padding: 20 }}>{video.titulo}</h4>
      <p>
        <img style={{ float: "left" }} src={video.localizacaoImagem} height="190" width="360px" alt="" />
        {video.resumo}
      </p>
    </div>
  );
  this.setState({ nextVideo: proximoVideo });
}
```

Fonte: Autores.

Figura 11 – Interface “Reprodução”.



Fonte: Os autores.

Figura 12 - Interface “Upload”.

Fonte: Os autores

Por José David

Foi implementada também a página gerenciamento de usuários onde a mesma é ilustrada na figura 10. Aqui o administrado poderá adicionar novos usuários, pesquisar qualquer usuário cadastrado, editar dados e remover qualquer usuário. Na próxima semana será iniciada a implementação de cada função desta página e também sua conexão com o banco de dados.

Dando continuidade no projeto da página, esta semana foi dedicada para o estudo do Firebase, que é o banco de dados que será utilizado para este projeto. A maior dificuldade está sendo em criar funções para interligar as páginas ao banco de dados. O intuito é conseguir implementar estas funções até semana que vem de modo que as páginas funcionem corretamente.

Prosseguindo com projeto da interface gerenciamento de usuário, nesta semana foram feitas algumas mudanças na estrutura da página. Foi retirado o botão adicionar usuário, pois a função desta tela apenas editar os dados dos usuários ou excluir os mesmos. Os dados que antes eram exibidos apenas no console, agora já estão sendo exibidos na tela a partir do nome pesquisado. Isto foi possível através de uma função criada que é chamada dentro da função pesquisa. O próximo passo é implementar a função editar que será responsável por atualizar quando necessário os dados de cada usuário cadastrado no banco de dados.

Partindo para reta final do projeto, nesta semana foram feitos os últimos ajustes na tela gerenciamento usuário. Foram feitas algumas modificações com o intuito de deixar a interface bem simplificada e intuitiva de modo que o usuário não tenha dificuldades na hora que estiver utilizando a mesma como mostra a figura 15.

Todos os usuários pendentes de autorização serão exibidos na interface, de modo que o administrador possa autorizar ou excluir o usuário que está sendo analisado. Para editar certas informações do usuário como definir se ele é administrador, definir um canal ou alterar o status de autorizado foi criada uma nova interface como mostra a figura 16. Para acessar a interface responsável por editar os dados do usuário o administrador fará uma busca pelo nome do usuário, esta busca irá retornar o usuário desejado e automaticamente irá aparecer o botão editar usuário.

As principais funções da interface gerenciamento de usuários responsáveis por exibir os usuários ainda não autorizados, deletar usuários, autorizar usuários e para setar os dados e redirecionar para a tela editar usuários são exibidas nas figuras 12, 13 e 14.

Figura 13 - Função responsável por exibir os usuários ainda não autorizados.

```
//Função responsável por procurar os usuário não autorizados (autorização 0)
searchUnauthorizedUsers = () => {
  //Query no firebase que retorn todos os usuários com autorização 0
  firebase.database().ref('usuarios').orderByChild('autorizacao').equalTo(0).on('value',
    (snapshot)=> {
      //Estrutura de repetição que renderiza cada usuário não autorizado na tela
      snapshot.forEach(snapshotChild => {
        const usuario = snapshotChild.val();
        const userKey = snapshotChild.key;

        const novoUsuario = (
          <div key={userKey} style={{ display: "grid", margin: "30px" }}>
            <div style={{ padding: "10px", backgroundColor: "white", maxHeight: "270px", maxWidth: "100%", display: "grid", gridTemplateColumns: "200px auto" }}>
              <div></div>
              <div className="userStatus" >
                <font size="1"></font> Usuário pendente de autorização
              </div>
              <img className="userPhoto" src={require("./img/people.jpg")} height="200px" />
              <p style={{ padding: "5px", lineHeight: 1.5 }}>
                <b>Nome: </b><b>{usuario.nome}</b><br/>
                <b>Administrador de Usuários: </b><b>{usuario.administradorUsuarios == 1 ? 'Sim' : 'Não'}</b><br />
                <b>Canal: </b><b>{usuario.canal ? usuario.canal : 'Canal não registrado'}</b><br />
                <b>E-mail: </b><b>{usuario.email}</b><br />
                <b>Autorização de Acesso: </b><b>{usuario.autorizacao == 1 ? 'Sim' : 'Não'}</b><br/>
              </p>
            </div>
            <button onClick={() => this.deleteUser(userKey)} className="deleteBtn" type="button">Excluir usuário</button>
            <button onClick={() => this.authorizeUser(userKey)} className="authorizeBtn" type="button">Autorizar usuário</button>
          </div>);
        this.usuario.push(novoUsuario);
        this.setState((listaUsuarios: React.createElement('div', {}, this.usuario)))
      });
    }
  );
}
```

Fonte: Autores.

Figura 14 – Funções responsáveis por deletar e autorizar usuários.

```
//Função responsável por deletar usuário no banco de dados.
deleteUser = (userKey) => {
  //Limpa a lista de usuários mostrados
  this.usuario = [];
  //Query que deleta o usuário baseadondo-se na key do mesmo
  firebase.database().ref(`usuarios/${userKey}`).remove();
  window.location.reload()
}

//Função responsável por atualizar a autorização do usuário no banco de dados.
authorizeUser = (userKey) => {
  //Limpa a lista de usuários mostrados
  this.usuario = [];
  //Query que atualiza o usuário baseadondo-se na key do mesmo
  firebase.database().ref(`usuarios/${userKey}`).update({
    autorizacao: 1
  });
  window.location.reload()
}
```

Fonte: Autores.

Figura 15 - Função responsável por setar os dados do usuário a ser editado e redirecionar para a página de edição do usuário.

```
//Função responsável por setar os dados do usuário a ser editado no localStorage e redirecionar para a pagina de edição do usuário.
handleNavigateToEditProfile = (user, userKey)=>{
  //Criação do objeto com os dados do usuário para armazenar no localStorage
  const userFormatted = {
    nome: user.nome,
    email: user.email,
    autorizacao: user.autorizacao,
    administradorUsuarios: user.administradorUsuarios,
    canal: user.canal ? user.canal : 'Canal não registrado',
    userKey,
  }
  //Armazenando os dados do usuário convertidos em string no localStorage
  localStorage.setItem('@IPTV:UserToBeAltered', JSON.stringify(userFormatted));
  //Redireciona para a pagina de edição do usuário
  this.props.history.push('/edit-user')
}
```

Fonte: Autores.

Figura 16 - Interface “Gerenciamento de Usuários”.

The interface displays a list of users under the heading "Procurar Usuários". The search bar contains the text "Quem você está procurando?". Below the search bar, the section "Usuários Encontrados" lists two users:

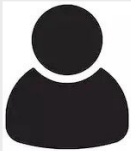
- Usuário 1:**
 - Nome: Ana
 - Administrador de Usuários: Não
 - Canal: Canal não registrado
 - E-mail: ana@gmail.com
 - Autorização de Acesso: Não
 - Status: Usuário pendente de autorização
 - Buttons: Autorizar usuário, Excluir usuário
- Usuário 2:**
 - Nome: José
 - Administrador de Usuários: Não
 - Canal: Canal não registrado
 - E-mail: jose@hotmail.com
 - Autorização de Acesso: Não
 - Status: Usuário pendente de autorização
 - Buttons: Autorizar usuário, Excluir usuário

The footer indicates "© 2020 Servidor IPTV".

Fonte: Autores

Figura 17 - Interface " Editar Usuário".

Servidor IPTV Gerir Usuários Novos Vídeos José David | Sair



José David

jose@gmail.com

Canal não registrado

Administrador de Usuários
Sim

Autorização de Acesso
Sim

Salvar Alterações

Fonte: Autores

Por Tiago Carneiro

A página de verificação de vídeos foi desenvolvida durante a semana, e todo o funcionamento e layout foram trabalhados nesse período, para a próxima etapa do projeto é desejável que implemente as funções pertinentes a esta página, tais como, adicionar vídeo, excluir vídeo, gerenciar usuários, sair, entre outros.

Como na página principal, está semana foi feito a implementação de algumas funções, do tipo busca de vídeos diretamente no banco de dados, mas esta função ainda não está retornando o resultado na tela, como o esperado. O trabalho previsto para a semana seguinte é implementar estas funções com retorno na própria tela do site ou aplicativo. Uma dúvida apareceu nesta parte do projeto, sobre a ideia de usuários incluir vídeos ou não, mas pretendemos discutir o assunto na próxima aula.

A interface de verificação de vídeo foi concluída, os aspectos mais importantes serão comentados. Para a montagem da página de forma automática, foi feito uma função que filtra os vídeos do banco de dados com o status de *aprovado* igual a 0, após esta função estar funcionando, foi feita outra função que pega estes dados e trazem eles para a tela no formato esperado. As imagens a seguir mostram as duas funções que possibilitaram esta tela de funcionar.

Figura 18 - Buscando os vídeos para aprovar.

```
var adicionarVideo = this.adicionarVideo;
firebase.database().ref('videos').orderByChild('aprovado').equalTo(0).on('value', function (snapshot) {
  var adicionarVideos = snapshot.val();
  if (!adicionarVideos) {
    return
  }
  var keys = Object.keys(adicionarVideos);
  for (var i = 0; i < keys.length; i++) {
    var k = keys[i];
    adicionarVideo(adicionarVideos, k);
  }
})
```

Fonte: Autores.

Figura 19 - Funções de aprovar, deletar vídeo e a de montar as imagens na tela.

```
deletarVideo = (k) => {
  firebase.database().ref(`videos/${k}`).remove();
  window.location.reload()
}

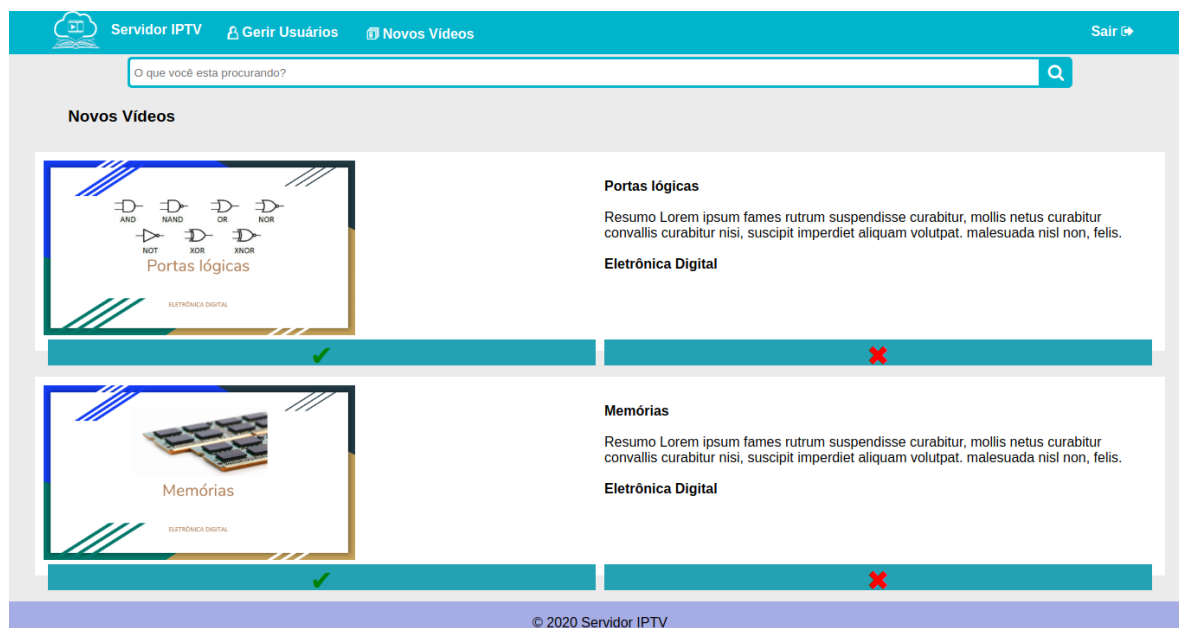
aprovarVideos = (k) => {
  firebase.database().ref(`videos/${k}`).update({
    aprovado: 1
  });
  window.location.reload()
}

adicionarVideo = (videosaprovar, k) => {
  const novosVideos = (
    <div className={classes.style}>
      <div className={classes.style1}>
        <img src={videosaprovar[k].localizacaoImagem} height="100%" width="360px" alt="" /><p style={{ padding: "5px" }} >
          <b>{videosaprovar[k].titulo}</b><br />
          <br />{videosaprovar[k].resumo}<br />
          <br /><b>{videosaprovar[k].canal}</b></p>
        <div className={classes.style3}>
          <i onClick={() => this.aprovarVideos(k)} style={{ cursor: 'pointer' }} className="fa fa-check" aria-hidden="true"></i>
        </div>
        <div className={classes.style4}>
          <i onClick={() => this.deletarVideo(k)} style={{ cursor: 'pointer' }} className="fa fa-times" aria-hidden="true"></i>
        </div>
      </div>
    </div>
  );
  this.videos.push(novosVideos);
  this.setState({ gridVideos: React.createElement('div', { className: classes.gridContainer }, this.videos) });
}
```

Fonte: Autores.

O intuito da página é de que o administrador seja capaz de autorizar os vídeos adicionados pelos donos de canais ou vídeos sugeridos por algum outro usuário da plataforma, então para a tela foi adicionado funções em cada um dos botões de confirmar e recusar. Para o botão de confirmar, adicionou a ele uma função que altera diretamente no banco de dados o campo de *aprovado*, que é onde definimos se o vídeo vai para a página principal ou não. No botão de recusar a função implementada foi a de apagar o vídeo diretamente no banco de dados, ambas as funções estão na figura 12.

Figura 20 - Interface “VerificacaoVideos”.



Fonte: Os autores

4 APLICATIVO WEB PROGRESSIVO

Por Fábio Campos Ferreira

Aplicativos nativos são softwares criados de forma específica para cada dispositivo onde são instalados, são estes os mais tradicionais utilizados em computadores e celulares. Estes aplicativos apresentam uma estrutura complexa e estável, com a possibilidade de utilizar recursos do hardware como USB, Bluetooth, placa de rede ou Wi-Fi e realizar operações de escrita e leitura no armazenamento do dispositivo. Outra característica, volta para o design da interação com o usuário é a presença de ícones nas telas iniciais dos dispositivos, facilitando a inicialização do aplicativo instalado na máquina e a maioria não necessita de uma conexão à internet para realizar a maioria de suas funções (PETE LEPAGE, 2020).

Aplicativos web são páginas web rodando em servidores que podem ser acessadas nas máquinas de clientes com acesso à internet através de navegadores como o Chrome, Firefox ou Safari. Utilizando um computador, notebook ou smartfone, o acesso à internet é um critério acessível a maioria da população mundial, tornando o alcance dos aplicativos web maior que os nativos (PETE LEPAGE, 2020).

Um aplicativo web progressivo (PWA) é construído com o objetivo de unir as capacidades presentes em um aplicativo nativo com o alcance específico das

aplicações web. Esta união é possível através da união de três pilares: a capacidade, a confiabilidade e a instabilidade. Estas condições devem ser verdadeiras durante todo o processo de alcance de qualquer pessoa, em qualquer lugar, usando qualquer dispositivo (PETE LEPAGE, 2020).

A capacidade condensa a possibilidade de executar funções como geolocalização, notificações, etc (PETE LEPAGE, 2020).

A confiabilidade representa a capacidade de resposta rápida do PWA de forma independente da velocidade de conexão internet ou a total falta desta. A velocidade de resposta de aplicativos está entre os principais motivos de um usuário continuar usando ou abandonar a interação com o aplicativo, a resposta gráfica a interações do usuário também ajuda o aumento da qualidade da experiência (PETE LEPAGE, 2020).

A instabilidade permite que o PWA execute em uma janela exclusiva, fora do ambiente normal do navegador e pode ser inicializado da tela inicial, barra de ferramentas ou outro local que seja possível posicionar o ícone de atalho do aplicativo instalado. Estas características provocam no usuário uma nova visão mais positiva sobre o aplicativo, mudando a forma como este o utiliza e se relaciona (PETE LEPAGE, 2020).

Porém, PWAs ainda podem ser classificados como somente aplicativos web, contudo, as ferramentas disponibilizadas pelos navegadores modernos permitem a construção de uma aplicação que proporciona uma experiência totalmente nova ao usuário (PETE LEPAGE, 2020).

Para as empresas que optaram por atualizar suas páginas na web para PWA, como o Twitter e Nikkei, relataram um aumento de acessos, de interações, das inscrições e um tráfego mais orgânico (PETE LEPAGE, 2020).

Os principais navegadores utilizados, Chrome, Safari, Edge, Firefox e Opera apresentam suporte para PWAs. Porém alguns destes não apresentam suporte para certos recursos, por exemplo, o Firefox não possibilita a captura de vídeos em dispositivos IOS, ao contrário do Chrome (SANTONI, 2018).

A criação ou adaptação de aplicativos web em PWAs pode ser feita adicionando alguns arquivos e criando certas funções em JavaScript. Porém está não é uma tarefa simples, principalmente para os programadores não tem conhecimento completo sobre as condições e características necessárias para o correto

funcionamento do PWA. Felizmente existem ferramentas gratuitas e de grande utilização que realizam toda a estruturação do projeto, restando ao programador a realização de pequenas adaptações e personalizações do projeto.

As principais ferramentas utilizadas para a confecção de aplicativos web progressivos são o Reactjs, Ionic, AngularJS e Polymer (DAVID, 2020).

6 REACT

Por Fábio Campos Ferreira

O React é uma biblioteca pertencente a linguagem JavaScript, tendo como principal objetivo ser uma ferramenta para a construção de interfaces de usuário. Esta biblioteca permite seu uso parcial na estrutura do site ou ser a base de sua construção (INTRODUÇÃO, 2020).

O React ganhou popularidade entre os desenvolvedores devido a sua possibilidade de criar aplicações de forma rápida pelo uso de paradigmas de programação intuitivos, realizando a tarefa de unir a estrutura de um código JavaScript com uma sintaxe semelhante à linguagem de marcação HTML, chamada de JSX (MORGAN, 2020).

A prova de sua popularidade é a sua utilização por empresas, além do Facebook, como Uber, Airbnb Facebook Pinterest, Netflix, Instagram, Amazon, Twitter, etc. Ainda deve-se citar que esta ferramenta de código aberto está em 159,5 mil projetos no GitHub (STACKSHARE, 2020).

Esta biblioteca vai em direção contrária a estrutura tradicional de uma página da web, sendo formadas por três arquivos, o arquivo HTML apresentando a organização dos elementos que constituem as páginas, o arquivo CSS que define as propriedades de estilos referentes a estes objetos e o arquivo JS, sendo o responsável por adicionar as funcionalidades de tratamento de informação, tornam a página dinâmica. Então, o React dá a possibilidade de armazenar todas as informações contidas nestes arquivos em um único arquivo JavaScript.

Embora o React possa ser categorizado como um código do lado do cliente, com a possibilidade de inclui-lo diretamente no arquivo HTML como um script. É mais prático e recomendado instalar o React através do npm. O npm é o gerenciador de pacotes padrão do Node.js, que por sua vez é um JavaScript fora do navegador. O

Node.js é baseado no motor JavaScript V8 do Chrome e tem a função de juntar múltiplos arquivos JavaScript (Subramanian, 2017).

O Facebook disponibiliza uma função chamada create-react-app para os programadores, este é comando descarta as várias etapas manuais da configuração do sistema de construção que converte a sintaxe moderna do React deposta em uma estrutura de diretórios base para o código legível a todos os navegadores (MORGAN, 2020) (CREATE... 2020).

Neste comando são incluídos pacotes JavaScript necessários para executar o projeto, funcionalidade de escuta de mudanças, teste e sistemas de construção dos arquivos do projeto. Este ainda cria um projeto com estrutura genérica com a estrutura apresentada na Figura 12. Assim os programadores não precisam se preocupar com a parte inicial mais técnica da criação do projeto (MORGAN, 2020).

Figura 21 - Estrutura pastas geradas pela create-react-app.

```
my-app
├── README.md
├── node_modules
├── package.json
├── gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── serviceWorker.js
    └── setupTests.js
```

Fonte: (CREATE... 2020).

Para a criação da estrutura de arquivos apresentada na Figura 12 é necessário executar, primeiramente, instalar as versões mais recentes do Node.js e do npm. Após esta instalação, é executado o comando *npm install -g create-react-app* para instalar comando de forma global na máquina de programação. Então é executado o comando *create-react-app my-app* para criar um aplicativo React genérico. Executando o comando *npm start* na pasta *my-app* é possível criar um servidor local que irá abrir no navegador o aplicativo contido nesta pasta (ISLAM, 2019).

A pasta *node_modules* contém todas as bibliotecas JavaScript que podem ser usadas na aplicação, inclusive o React (MORGAN, 2020).

A pasta *public* contém arquivos base como HTML, JSON e arquivos de imagens. Esta pasta é a raiz do projeto. Quando o comando *npm start* é executado ele busca o arquivo *public/index.html* para iniciar a construção do site. Este arquivo HTML pega os dados do arquivo *public/manifest.json* que definem informações referentes a características próprias de aplicativos como nome, ícones a serem usados, cor do tema e do fundo (MORGAN, 2020).

O diretório *src* contém os arquivos com o código React JavaScript, sendo estes os responsáveis pela definição da estrutura do site. Por padrão, o arquivo *public/index.html* executa o código contido no *src/index.js* (MORGAN, 2020).

Contudo, a estrutura genérica gerada necessita de algumas adaptações para ganhar as características pertencentes a um aplicativo web progressivo e ser possível a sua instalação tanto em dispositivos móveis com computadores de mesa (ISLAM, 2019).

Para verificar quais critérios que o aplicativo está ou não cumprindo é possível utilizar o Lighthouse, uma ferramenta disponível tanto em forma de extensão para os navegadores Chrome e Mozilla, ou como uma função presente nas ferramentas de desenvolvimento (DevTools) dos mesmos (ISLAM, 2019).

Para satisfazer requisitos de um PWA, no arquivo *src/index.js* é preciso mudar o comando *serviceWorker.unregister()* para *serviceWorker.register()*, assim, o arquivo *src/serviceWorker.js*, responsável por gerenciar solicitações dos arquivos fonte do site ao servidor. Esta mudança permite que o aplicativo carregue mais rápido em visitas subsequentes, apresentando recursos quando offline, por meio do armazenamento em cache de arquivos fonte (ISLAM, 2019).

Mudanças reativas ao design do aplicativo são feitas através do arquivo *public/manifest.json* (ISLAM, 2019).

Após estas etapas, resta somente a compilação do código para o formato legível ao navegador, que é feita pela execução do comando *npm run build*, que cria uma pasta build no diretório my-app com os arquivos que devem ser alocados no servir na rede. Desta forma, executando na pasta contendo estes arquivos um servidor local ou na rede é possível observar o PWA instalável construído com React (ISLAM, 2019).

7 FIREBASE

Por Tiago Carneiro

O Firebase de Google é uma plataforma digital utilizada para facilitar o desenvolvimento de aplicativos web ou móveis, de uma forma efetiva, rápida e simples. Graças às suas diversas funções, é utilizado como uma técnica de Marketing Digital, com a finalidade de aumentar a base de usuários e gerar maiores benefícios econômicos (ROCKCONTENT, 2020).

Seu principal objetivo é melhorar o rendimento dos apps mediante a implementação de diversas funcionalidades que farão do aplicativo um instrumento muito mais maleável, seguro e de fácil acesso para os usuários (ROCKCONTENT, 2020).

O Firebase possui a grande vantagem de que pode ser utilizado nas maiores plataformas do mercado atual, por exemplo, Android, IOS e web. Outro ponto importante é que é possível gerar lucro com o ele, na forma de publicidade, é possível habilitar anúncios e arrecadar dinheiro com isso. Outro aspecto é que é possível fazer seu app sem custo algum, isso para níveis baixos, quando o nível está mais avançado a plataforma oferece planos que podem atender melhor o cliente.

Os serviços oferecidos pelo Firebase são os seguintes:

- Real time data base: que trabalha com a base de dados em tempo real;
- Autenticação: executa para identificar usuários por meio de e-mail ou redes sociais;
- Nuvem de armazenamento: armazena e envia arquivos à escala de Google;
- Hosting: é onde hospedamos o site;

- Remote config: permite alterar alguns aspectos do app sem necessidade de atualizar;
- Test lab: permite testar o aplicativo antes de publicar;
- Crash reporting: reporta erros do aplicativo.

Para este trabalho, o banco de dados escolhido foi o Firebase Realtime database. O Firebase Realtime Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps em plataformas cruzadas com nossos SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes (FIREBASE GOOGLE, 2020).

Os recursos que se destacam são:

- Em vez de solicitações HTTP típicas, o Firebase Realtime Database usa a sincronização de dados. Sempre que os dados são alterados, todos os dispositivos conectados recebem essa atualização em milissegundos. Crie experiências colaborativas e imersivas sem se preocupar com códigos de rede (FIREBASE GOOGLE, 2020);
- Os apps do Firebase permanecem responsivos mesmo off-line, pois o SDK do Firebase Realtime Database mantém seus dados em disco. Quando a conectividade é restabelecida, o dispositivo cliente recebe as alterações perdidas e faz a sincronização com o estado atual do servidor (FIREBASE GOOGLE, 2020);
- O Firebase Realtime Database pode ser acessado diretamente de um dispositivo móvel ou navegador da Web, sem um servidor de aplicativos. A segurança e a validação de dados estão disponíveis por meio de regras de segurança baseadas em expressão do Firebase Realtime Database, executadas quando os dados são lidos ou gravados (FIREBASE GOOGLE, 2020).

O Cloud Storage para Firebase é um serviço de armazenamento de objetos poderoso, simples e econômico criado para a escala do Google. Com os SDKs do Firebase para Cloud Storage, você usa a segurança do Google para fazer o upload e o download de arquivos nos aplicativos Firebase, independentemente da qualidade da rede. Use os nossos SDKs para armazenar imagens, áudio, vídeo ou outros

conteúdos gerados pelo usuário. No servidor, utilize o Google Cloud Storage para acessar esses mesmos arquivos (FIREBASE GOOGLE, 2020).

Os principais recursos:

- Com os SDKs do Firebase para Cloud Storage, os uploads e downloads são feitos independentemente da qualidade da rede. Os uploads e downloads são mais confiáveis, o que significa que eles são retomados no ponto em que foram interrompidos, poupando tempo e largura de banda dos usuários (FIREBASE GOOGLE, 2020).
- Os SDKs do Firebase para Cloud Storage estão integrados ao Firebase Authentication para fornecer uma autenticação simples e intuitiva para os desenvolvedores. Use o nosso modelo de segurança declarativa para que o acesso seja concedido com base no nome, tamanho, tipo de conteúdo e outros metadados do arquivo (FIREBASE GOOGLE, 2020).
- O Cloud Storage para Firebase foi projetado para suportar a escala de exabyte que o seu app terá quando se tornar um sucesso. Desenvolva sem esforço, do protótipo até a produção, com a mesma infraestrutura utilizada pelo Spotify e pelo Google Fotos (FIREBASE GOOGLE, 2020).

No código do aplicativo foi necessário utilizar funções que são ligadas diretamente ao banco de dados, por exemplo as funções de busca são bem parecidas, o que difere é o que o usuário está buscando, para mostrar isso, a figura 13 mostra um trecho do código de busca. Esta função busca dentro do banco de dados a *string* gravada na variável *tituloVideo*, como o firebase trabalha com subclasse, por exemplo ele salva a chave do vídeo e dentro dessa chave do vídeo você encontra título, resumo, autor, etc. então é necessário passar o destino e qual a subclasse que está buscando, e ele varre o banco de dados procurando por uma *string* igual ao que foi passado para ele, e retorna ela em forma de objeto, e precisa ser tratado para ter valor como um *array*.

Figura 22 - Função buscaVideo.

```
buscaVideo = () => {  
  var tituloVideo = document.getElementById("tituloInput").value;  
  firebase.database().ref('videos').orderByChild('titulo').equalTo(tituloVideo).on('value', function (snapshot) {  
    console.log(snapshot.val());  
  });  
}
```

Fonte: Autores.

Outra função que foi muito utilizada, foi a função que gravar dados e ler dados do banco de dados, como é necessário o cadastro de novo usuários e vídeos, logo a função foi utilizada em algumas páginas do aplicativo. A figura 14 mostra o trecho do código onde é feito a gravação dos dados no banco de dados do firebase.

Figura 23 - Gravando dados no banco de dados.

```
if(snapshot.val()!==null){  
  var data ={nome:nome.value,email:email,senha:senha};  
  firebase.database().ref().child('usuarios').push(data);  
}
```

Fonte: autores.

Para gravar os dados, armazena os dados das variáveis nome, e-mail, senha, etc., dentro da variável data, feito isso você passa o caminho de onde deseja gravar os dados no banco de dados, neste caso a intenção era de gravar na classe filha (*child*) então utiliza o *push* para gravar os dados.

8 ESTRUTURA BÁSICA DOS ARQUIVOS REFERENTES AO PROJETO

Por Fábio Campos

O aplicativo aqui descrito foi construído tendo como base na estrutura apresentada na Figura 12. Quando inicializado, o arquivo "public/index.html" é aberto pelo sistema do servidor, este arquivo busca o JavaScript "src/index.js" e executa-o. O código executado apresenta duas chamadas, a primeira é o *ReactDOM.render()*, responsável por construir a estrutura DOM da página web pela codificação da estrutura construída usando o React e o JSX, o segundo comando é *serviceWorker.register()* que inicializa as funções presentes no arquivo "src/serviceWorker.js" para o gerenciamento do cache.

O PWA projetado apresenta sete telas, sendo elas a de login, principal, pesquisa, reprodução, upload, gerenciamento de usuários e verificação vídeos apresentados de forma gráfica nas Figuras (5), (6), (7), (8), (9), (10) e (11), respectivamente. Para uma melhor organização do projeto, foi estabelecido a construção de cada página de forma isolada, uma em relação a outra. Desta forma, os testes do código bem como sua manutenção ou aperfeiçoamento podem ser feitos

de forma individual, sem a necessidade de acionar funções as variáveis presentes as outras páginas.

Então, cada tela, na estrutura da pasta, apresenta dois arquivos ligados diretamente e somente a sua construção, tanto da parte gráfica construída em JSX, como as funções em JavaScript para a execução das principais funcionalidades. O primeiro arquivo é um arquivo de extensão ".js", que levará a nomenclatura1 "tela" seguido pelo nome de sua respectiva tela, apresentando todas as funções e a construção do DOM pelo JSX. O segundo arquivo, que é importado pelo primeiro é o arquivo de extensão ".css", sendo um arquivo de linguagem de estilo que descreve as características gráficas, agrupada em forma de classes que são referenciadas pelo JSX para caracterizar uma tag específica ou de forma abranger todas as tags de mesmo tipo do DOM. O arquivo CSS tem estrutura igual aos utilizados para aplicações webs tradicionais. Cada arquivo CSS referência à tela que é apresentada em sua própria nomenclatura. Outro arquivo CSS é acionado através de um link no arquivo "public/index.html", contendo estilos de alta qualidade, disponibilizados para uso público. Para uma melhor organização estrutural dos arquivos do projeto, todos os de extensão ".css" foram separados e agrupados na pasta 'src/css'.

De forma semelhante, todas as imagens utilizadas na construção básica do site, como as do logo, foram agrupadas na pasta 'src/img'.

Contudo, como cada tela é construída por um arquivo diferente, a função *ReactDOM.render()* do arquivo "src/index.js" deve ser capaz de acionar corretamente a construção DOM de cada arquivo. Para realizar esta função, foi instalada outra biblioteca, também pertencente ao React, chamada React Router Dom pelo comando

```
npm install --save react-router-dom
```

A estrutura do ReactDOM.render é apresentada na Figura 13(*ReactDOMRender*) (SOUTO, 2018).

Figura 24 - Código para o roteamento entre telas.

```
import { BrowserRouter, Switch, Route } from 'react-router-dom'
ReactDOM.render(
  <BrowserRouter>
    <Switch>
      <Route path="/" exact={true} component={TelaLogin}/>
      <Route path="/pesquisa" exact={true} component={TelaPesquisa}/>
      <Route path="/upload" exact={true} component={TelaUpload}/>
      <Route path="/gerenciamentousuario" exact={true} component={TelaGerenciamentoUsuario}/>
```

```

<Route path="/principal" exact={true} component={TelaPrincipal}/>
<Route path="/reproducao" exact={true} component={TelaReproducao}/>
<Route path="/verificacaovideo" component={TelaVerificacaoVideo}/>
<Route path='*' component={Tela404} />
</Switch>
</BrowserRouter>
,
document.getElementById('root')
);

```

Fonte: Os autores.

Na Figura 13 é possível observar que as tags importadas da biblioteca React Router Dom associam um endereço da URL ao arquivo JavaScript referente a uma tela, sendo que este deve ser importado. Portanto, para cada tela, é associado um caminho URL exclusivo. A tela de login é referenciada por "/", sendo o caminho principal do site. A troca entre telas pode ser realizada de duas formas. A primeira é através de outra tag do React Router Dom, chamada Link. A segunda é chamando a método *createBrowserHistory()* oriunda da biblioteca *history*, que muda a URL atual e força uma atualização para alterar a tela atual, um processo mais lento que o primeiro, mas necessário o seu uso em alguns casos (SOUTO, 2018).

Assim, neste arquivo são importadas as bibliotecas React, React DOM e React DOM Router, bem como os arquivos JavaScript referentes a cada página e o "src/serviceWorker.js".

A estrutura básica de um arquivo JavaScript referente a uma tela segue o padrão mais utilizado em aplicações que utilizam a biblioteca React como base, sendo a mais indicada para a máxima utilização de todos os recursos que o React permite executar. Esta estrutura é composta um componente React, ou seja, uma classe que é a extensão da classe *React.Componente*. Esta classe possui os seguintes métodos base *constructor()*, *render()*, *getDerivedStateFromProps()*, *componentDidMount()*.

O método *constructor()* é chamado antes de qualquer outro, ou seja, quando o componente é inicializado. Desta forma, ele é usado para iniciar o *state*, sendo este o objeto que armazena variáveis pertencentes ao componente.

O método *render()* é obrigatório para todo componente, este é responsável por retornar a estrutura gráfica construída em JSX. O método *getDerivedStateFromProps()* é a última chamada antes da construção do DOM, ideal para atualizar as variáveis contidas no *state*. E finalmente, o método

`componentDidMount()` é o método chamado após a renderização da estrutura retornada pelo `render()`, ou seja, quando o DOM já foi construído, sendo usado para executar instruções que necessitem acessar elementos presentes no DOM.

Além da presença da classe componentes nos arquivos JavaScript referentes às telas, há a presença de importações das bibliotecas utilizadas, como também das imagens e do arquivo CSS. Para que o arquivo seja importado pelo "src/index.js", ele deve exportar o componente que presente no mesmo. A Figura 14 apresenta a estrutura básica utilizada para criar os arquivos JavaScript de todas as telas.

Figura 25 - Código JavaScript básico para cada tela.

```
import React from 'react';
import classes from "./css/nomeTela.module.css";
import { Link } from 'react-router-dom';
import firebase from "./autenticacao.js";
import { createBrowserHistory } from 'history';

class TelaNome extends React.Component{
  constructor(props) {
  }

  static getDerivedStateFromProps(props, state){
  }
  componentDidMount() {
  }

  //Outros métodos são adicionados

  render(){
    return(
      //Estrutura JSX
    );
  }
}

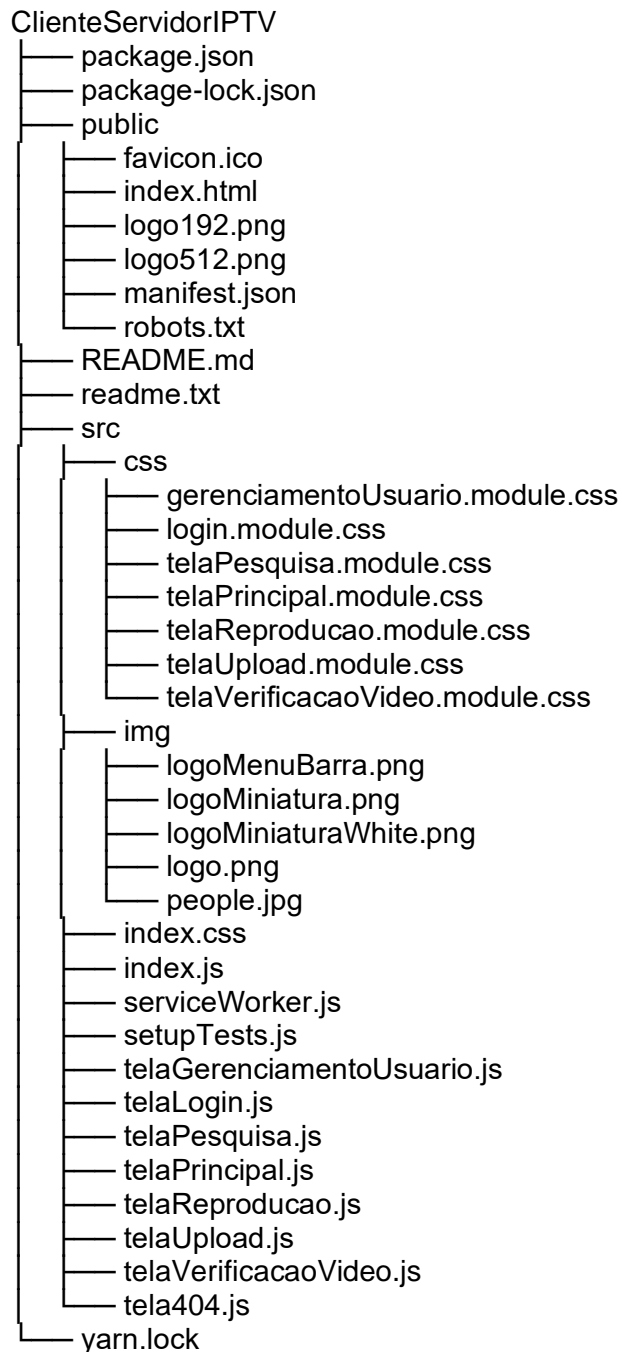
export default TelaNome;
```

Fonte: Os autores.

Finalmente, após a discussão geral sobre cada categoria de arquivo que fazem parte do projeto e permitem a criação de todas as telas dos projetos, com cada uma

contendo seus respectivos métodos, é apresentado a árvore de arquivos contendo todos que fazem parte do projeto na Figura 15.

Figura 26 - Estrutura de arquivos do site.



Fonte: Os autores.

9 Tela de Login

Por Fábio Campos Ferreira

O arquivo "telaLogin.js" contém todo o código utilizado para a construção da tela de login. A estrutura básica deste arquivo é a mesma apresentada na Figura 21. A parte gráfica da tela, apresentada na Figura 5, foi construída através da estrutura JSX, adicionada no método "render()". Além dos métodos comuns como o "constructor()", "getDerivedStateFromProps()" e "render()", foram criadas outras funções na configuração de seta (*arrow function*) para a sua chamada em outras funções da própria classe deste arquivo e ao acionar eventos como o clique em botões criados pelo "render()". Estas funções são "clickLinkLogin()", "entrar()" e "criarConta()".

A função "constructor()" define a variável "state", própria a classe, como sendo um objeto *array* de um elemento de endereçamento "url" que pode assumir valor "/" ou "/principal". O "url" terá valor "/" enquanto o usuário não for autorizado a entrar no site, quando autorizado o valor é mudado para "/pesquisa".

A função "getDerivedStateFromProps()" é chamada antes da construção dos elementos pertencentes ao DOM. Esta realiza a análise o valor contido na variável "localStorage.autenticacao", armazenada localmente, onde é armazenado o identificador do usuário e confirma a sua presença no banco de dados. Se for confirmada, é mudado o caminho URL da própria janela e é forçado uma atualização para redirecionar o usuário já logado para "/principal", sendo a tela principal. Se não for encontrado o usuário, a variável é limpa e é mantido. Este procedimento tem o único objetivo manter o usuário com estado logado, mesmo quando o aplicativo ou janela do navegador é fechada.

A função "entrar()" é chamada quando o usuário clica sobre o botão Entrar. O início desta função é a obtenção dos valores digitados nos campos de e-mail e senha. É verificado se realmente foi digitado algum valor, se sim é buscado no banco de dados que apresenta o mesmo valor de e-mail e senha. Se o usuário for encontrado, o valor "state" de posição "url" é mudado para "/principal" e a função "clickLinkLogin()" é chamada. Caso algo ocorra de forma a não permitir que usuário acesse sua conta, o texto de uma *tag* "p" é mudado para uma mensagem em vermelho que informa qual condição não foi cumprida.

A função `clickLinkLogin()` na função `entrar()`. Esta função localiza a *tag* `Link` própria da biblioteca React, que no DOM é transformada na *tag* `a`, que tem como destino o valor da variável `url` do `state` definido no `constructor()`. Então é simulado um clique nesta *tag* para o redirecionamento para a tela de caminho igual a `url`.

A função `criarConta()` é chamada quando o usuário clica sobre o botão Criar Conta. O início desta função é a obtenção dos valores digitados nos campos de e-mail, senha e nome, este último é escondido inicialmente e se torna visível quando a função `criarConta()` é chamada pela primeira vez. Estes valores passam por verificações que comprovem a sua estrutura conforme as regras do site. Estas regras afirmam que o nome de usuário deve ter tamanho mínimo de três dígitos, o e-mail deve seguir o padrão `'algumanome@algunacorporação.com'`, não podendo já estar cadastrado no banco de dados e a senha deve ter tamanho mínimo de seis dígitos. Se estas condições forem cumpridas, é criado um array com estes valores e valores padrão para as demais propriedades do banco de usuários. Estas propriedades podem ser vistas na Figura 4. Os valores padrão são 0 para `autorizacao` e para `administradorUsuarios` e uma *string* varia para `administradorCanal`. A última etapa é adicionar estas informações no banco de dados. Caso algo ocorra de forma a não permitir que usuário acesse sua conta, o texto de uma *tag* `p` é mudado para uma mensagem em vermelho que informa qual condição não foi cumprida.

As funções discutidas acima constituem todas as funções presentes no arquivo `telaLogin.js` e que se relacionam diretamente com as funções de criação de conta e login na conta já criada que foram planejadas para serem implementadas nesta tela.

10 Tela de Pesquisa

Por Fábio Campos Ferreira

O arquivo `telaPesquisa.js` contém todo o código utilizado para a construção da tela de pesquisa. A estrutura básica deste arquivo é a mesma apresentada na Figura 21. A parte gráfica da tela, apresentada na Figura 7, foi construída através da estrutura JSX, adicionada dentro do método `render()`. Além dos métodos comuns como o `constructor()`, `getDerivedStateFromProps()`, `componentDidMount()` e `render()`, foram criadas outras funções na configuração de seta (arrow function) para a sua chamada dentro de outras funções da própria classe presente neste arquivo e

ao acionar eventos como o clique em botões criados pelo "render()". Estas funções são "sair()", "openNav()", "closeNav()", "filtros()", "pesquisar()", "adicionarATela()" e "assistirVideo()".

A função "constructor()" define a variável "state", própria a classe, como sendo um objeto *array* de elementos com endereçamento "url", "autenticacao" e "gridVideos". O "url" terá valor "/pesquisa" enquanto o usuário estiver autorizado a entrar no site, se não autorizado o valor é mudado para "/", redirecionando para a tela de login sem que a tela de pesquisa apareça totalmente. A "autorizacao" assume valor da "localStorage.autorizacao", sendo o valor de identificação de usuário que realizou login, se este valor não pertencer a um usuário dentro do banco de dados, o site é redirecionado para a tela de login. O "gridVideos" é inicialmente vazio, porém, após a realização de pesquisa e preenchido por elementos contendo estrutura JSX que representaram a estrutura gráfica onde apareceram os vídeos que condizem com a pesquisa.

A função "getDerivedStateFromProps()" é chamada antes da construção dos elementos DOM. Esta realiza a análise do valor contido na variável "localStorage.autenticacao", armazenada localmente, onde é armazenado o identificador do usuário. É analisado a presença deste identificador dentro do banco de dados. Se não for confirmada a sua presença, é mudado o caminho URL da própria janela e é forçada uma atualização para redirecionar o usuário para "/", sendo a tela de login. Se for encontrado o usuário, nada é feito. Este procedimento tem o único objetivo manter o usuário com status logado ou deslogado, mesmo quando o aplicativo ou janela do navegador é fechada. Desta forma, mesmo o usuário não logado digitando a URL "/pesquisa", sua entrada é bloqueada nesta página.

A função "componentDidMount()" é chamada quando todos os componentes são renderizados. Esta realiza a busca do usuário no banco de dados pelo seu identificador e é verificado seus status de gerenciador de usuários e de gerenciador de canal. Se o resultado de gerenciador de usuários for afirmativo, o link para a tela de gerenciamento de usuários é redefinido de escondido para visível. Se o resultado de gerenciador de canal for afirmativo, o link para a tela de verificação de vídeos é redefinido de escondido para visível. Posteriormente, a função "pesquisar()" é chamada, onde está recebe o valor do texto digitado na barra de pesquisa na tela principal, realizando assim sua busca logo quando a página é carregada,

apresentando os resultados ao usuário de forma imediata após transição da tela principal para tela de pesquisa.

A função "sair()" é chamada quando o botão Sair é clicado. Está limpa a variável "localStorage.autenticacao", define o endereçamento "url" de "state" para "/" e simula o clique na *tag* "Link" que tem como direcionamento justamente o valor representado pelo endereçamento "url" de "state". Desta forma, o usuário é redirecionado para a tela de login e os status de logado são definidos para deslogado.

A função "filtros()" é chamada quando o usuário clica sobre o botão Filtros com o objeto de visualizar ou esconder as opções de filtragem. Esta realiza uma comparação da largura atual da aba, se igual a zero a função "openNav()" é chamada para permitir a visualização da aba. Caso contrário a função "closeNav()" é chamada para esconder a aba.

A função "openNav()" é chamada dentro da função "filtros()". Esta função é responsável por aumentar a propriedade largura da aba contendo as opções de filtragem e aumentar a margem esquerda do espaço destinado aos vídeos. Desta forma, a aba de filtros se torna visível e não se sobrepõe a nenhum vídeo.

A função "closeNav()" é chamada dentro da função "filtros()". Esta função é responsável por reduzir a propriedade largura da aba contendo as opções de filtragem e reduzir a margem esquerda do espaço destinado aos vídeos. Desta forma, a aba de filtros se torna invisível e o espaço destinado os vídeos pesquisados pode utilizar toda a largura da página, sem deixar espaços em branco na zona onde a aba voltar a aparecer.

A função "pesquisar()" é chamada quando pela função "componentDidMount()", quando o botão pesquisar é clicado ou quando ocorre o pressionamento de tecla quando a área de texto está selecionada. No entanto, esta função só permitirá continuar sua operação quando a tecla pressionada for o Enter. Quando não for chamada pelo "componentDidMount()", que já envia o texto a ser pesquisado, esta função obtém o valor digitado na área de texto. Se o texto não apresenta caracteres a operação é cancelada e é mudado o valor de uma *tag* "p" para a mensagem que informe o usuário o motivo da falha. Para cada chamada desta função o valor de endereçamento "grid" de "state" e a variável "videos" definida na função "construtor()" são redefinidas para um valor vazio de forma a limpar os vídeos encontrados em pesquisas passadas. Na continuação da operação, obtém-se os

parâmetros de filtragem, podendo ser título, resumo, atores, responsável, administrador e canal, selecionados e o texto a ser pesquisado é transformado em uma lista de palavras chaves. Cada palavra chave é pesquisada para cada um dos parâmetros de filtragem. Esta pesquisa é possível, pois para cada vídeo, além das propriedades apresentadas na estrutura do banco de vídeos vista na Figura 4, é adicionado propriedades de valor booleano com a identificação sendo a junção da opção de filtragem com uma palavra chave presente dentro do texto a qual a opção de filtragem se refere. A Figura ?? apresenta esta ideia de palavras chave com um exemplo de estrutura de um vídeo dentro do banco de dados. Para cada palavra chave é verificado se o seu retorno é *true*, assim é confirmado a sua presença no vídeo e o mesmo é retornado. Quando o vídeo é retornado, a função "adicionarATela()" é chamada e é enviada a ela as informações do vídeo encontrado. Para evitar selecionar o mesmo vídeo mais de uma vez ao pesquisar novamente todos os vídeos para cada parâmetro de filtragem, a variável "listadVideosEncontrados" contendo os identificadores de cada vídeo já adicionado a tela é atualizada e antes da chamada da função "adicionarATela()" é verificado se o vídeo já está presente nesta variável.

A função "adicionarATela()" é chamada pela função "pesquisar". Esta recebe um *array* apresentando todas as informações de um vídeo, através desta, uma variável "novoVideo" contendo uma construção JSX é criada. Esta estrutura JSX apresenta as variáveis do título, resumo, link da *thumbnail* e canal do vídeo recebido que permitem o preenchimento específico da tela de pesquisa com blocos que representam os vídeos encontrados na pesquisa. Como é visto na Figura 7. Este "novoVideo" é armazenado no fim do *array* "videos" definido no construtor. O valor do "state" de endereçamento "gridVideos" é mudado para o retorno da função "React.createElement()" que irá criar uma *tag* "div" com a classe apropriada e tendo como elementos filhos todas as variáveis "novoVideo" contidas em "videos". O valor do "state" de endereçamento "gridVideos" é colocado dentro da função "render()" de forma apropriada, assim, a cada mudança desta variável, a renderização acompanhará o novo valor e mudará os vídeos apresentados na tela de forma condizente.

A função "assistirVideo()" é utilizada pela "adicionarATela()" como uma função de chamada quando ocorrer um clique sobre a *tag* "Link" do respectivo. A função "assistirVideo()" recebe o identificador do vídeo da *tag* e salva na variável

"localStorage.telaReproducaoVideo", que será buscada na tela de reprodução, definindo o video que será apresentado nesta.

11 Tela de Upload

Por Fábio Campos Ferreira

O arquivo "telaUpload.js" contém todo o código utilizado para a construção da tela de upload. A estrutura básica deste arquivo é a mesma apresentada na Figura 21. A parte gráfica da tela, apresentada na Figura 11, foi construída através da estrutura JSX, adicionada dentro do método "render()". Além dos métodos comuns como o "constructor()", "getDerivedStateFromProps()", "componentDidMount()" e "render()", foram criadas outras funções na configuração de seta (arrow function) para a sua chamada dentro de outras funções da própria classe presente neste arquivo e ao acionar eventos como o clique em botões criados pelo "render()". Estas funções são "sair()", "adicionarVideo()", "duracaoVideo()", "atualizarBancoVideos()" e "mudarNomeArquivo()".

A função "constructor()" define a variável "state", própria a classe, como sendo um objeto *array* de elementos com endereçamento "url" e "autenticacao". O "url" terá valor "/upload" enquanto o usuário estiver autorizado a entrar no site, se não autorizado o valor é mudado para "/", redirecionando para a tela de login sem que a tela de pesquisa apareça totalmente. A "autorizacao" assume valor da "localStorage.autenticacao", sendo o valor de identificação de usuário que realizou login, se este valor não pertencer a um usuário dentro do banco de dados, o site é redirecionado para a tela de login.

A função "getDerivedStateFromProps()" é chamada antes da construção dos elementos DOM. Esta realiza a análise do valor contido na variável "localStorage.autenticacao", armazenada localmente, onde é armazenado o identificador do usuário. É analisado a presença deste identificador dentro do banco de dados. Se não for confirmada a sua presença, é mudado o caminho URL da própria janela e é forçado uma atualização para redirecionar o usuário para "/", sendo a tela de login. Se for encontrado o usuário, nada é feito. Este procedimento tem o único

objetivo manter o usuário com status logado ou deslogado, mesmo quando o aplicativo ou janela do navegador é fechada.

A função "componentDidMount()" é chamada quando todos os componentes são renderizados. Esta realiza a busca do usuário no banco de dados pelo seu identificador e é verificado seus status de gerenciador de usuários e de gerenciador de canal. Se o resultado de gerenciador de usuários for afirmativo, o link para a tela de gerenciamento de usuários é redefinido de escondido para visível. Se o resultado de gerenciador de canal for afirmativo, o link para a tela de verificação de vídeos é redefinido de escondido para visível. Posteriormente, é verificado se foram recebidas corretamente as informações do vídeo atual da tela de reprodução. Se estas informações são inválidas, não é possível adicionar um vídeo na sequência do atual da tela de reprodução, então a interface de upload não é apresentada ao usuário e o erro é alertado ao usuário.

A função "sair()" é chamada quando o botão Sair é clicado. Está limpa a variável "localStorage.autenticacao", define o endereçamento "url" de "state" para "/" e simula o clique na *tag* "Link" que tem como direcionamento justamente o valor representado pelo endereçamento "url" de "state". Desta forma, o usuário é redirecionado para a tela de login e os status de logado são definidos para deslogado.

A função "adicionarVideo()" é chamada quando o botão adicionar vídeo é clicado. Esta função, primeiramente define todas as variáveis utilizadas referentes a informações do vídeo que serão armazenadas no banco de dados. Em seguida, é verificado se a imagem de capa e o vídeo foram devidamente selecionados, tendo ambos formatos predefinidos, se caso os arquivos não estarem dentro dos parâmetros estabelecidos é mostrado uma mensagem de aviso. Os campos de título, resumo e atores também são verificados se corretamente preenchidos, se não uma mensagem de aviso é apresentada. Em sequência, os dois arquivos são enviados ao armazenamento do Firebase e um link para este é retornado, sendo armazenado no banco de dados do respectivo vídeo. Após completado o envio dos arquivos é chamada a função "atualizarBancoVideos()", os campos são resetados e o usuário é redirecionado de volta a tela de reprodução.

A função "duracaoVideo()" é chamada pela função "adicionarVideo()" para poder retornar, utilizando a variável "localStorage.duracao" a duração do vídeo

selecionado em minutos. Esta função utiliza a estrutura de uma API sendo "URL.createObjectURL()".

A função "atualizarBancoVideos()" é chamada pela função "adicionarVideo()" e recebe vários parâmetros que serão utilizados para constituir as informações necessárias do vídeo e armazená-las no banco de dados de forma apropriada. Estas informações são adicionadas no banco de dados, e uma rede de parâmetros de palavras chaves contidas no título, resumo, canal e atores também são adicionadas, de forma isolada pela sua origem, ao banco de dados. Esta estrutura de palavras chaves permite a pesquisa e a filtragem de vídeos. O parâmetro "proximoVideo" do vídeo atual na tela de reprodução é mudado para o atualmente adicionado, com o intuito de ser este o próximo vídeo na lista de reprodução do canal, se inserindo depois do vídeo atual e antes do próximo vídeo, originalmente.

A função "mudarNomeArquivo()" é chamada quando ocorre uma mudança dos arquivos selecionados, tanto para a imagem como para o vídeo. Assim, quando um vídeo é selecionado, o nome do arquivo é apresentado na tela.

12 RESULTADOS ESPERADOS

Por José David Vargas de Melo Neto

Os resultados esperados para este sistema é que o usuário consiga escolher um vídeo para assistir utilizando a filtragem por título, resumo, duração, disciplina, tópicos, assuntos, atores, ano, responsável, administrador, canal, etc. O sistema também deverá ser compatível com navegador Web e com aplicativo celular. O administrador do sistema será capaz de incluir ou excluir usuários como também aprovar ou reprovar os vídeos enviados por eles. Outros vídeos com assuntos semelhantes ao pesquisado pelo usuário deverão ser sugeridos na sequência de cada vídeo.

13 CONCLUSÃO

Por Fábio Campos Ferreira

Neste relatório é apresentado todos os paços utilizados na construção de uma aplicação progressiva na web, instalável em qualquer dispositivo com navegador instalado que tenha suporte a aplicativos PWA, ou seja, celulares, desktops, entres outros. O aplicativo desenvolvido consegue realizar o login e cadastro de usuários, bem como, depois de cadastrados, mudar seus privilégios de administradores, podendo ou não gerenciar privilégios de outros usuários, ou permitir a inclusão de novos vídeos no armazenamento do site. Sendo funções acessíveis na barra de menus se os usuários tiverem estes privilégios. Depois de logado, o usuário é direcionado para uma tela onde já é apresentado opções de vídeos disponíveis para a visualização, como também a possibilidade de pesquisa de vídeos utilizando várias opções de filtragem. Na tela onde o vídeo é reproduzido, uma opção é apresentada para o usuário recomendar um vídeo presente em seu dispositivo para ser um complemento ao vídeo atualmente reproduzindo. Desta forma, o vídeo enviado, se aprovado pelo usuário com privilégios de gerenciamento de vídeos, será apresentado como uma sequência do vídeo anteriormente já presente no sistema do site.

O aplicativo foi alocado em um servidor online gratuito e pode ser acessado e testado pelo link <<https://clienteservidoriptv.000webhostapp.com>>. O código fonte do projeto foi armazenado no GitHub pode ser obtido de forma completa pelo link do projeto <<https://github.com/FabioCamposFerreira/pi2020/tree/master>>. No código fonte, o arquivo "readme.txt" apresenta as características do projeto, bem como os passos de instalação dos recursos necessários para executar e compilar este código fonte.

O projeto, como todo com software, sempre recebe atualizações periódicas para alterar algumas características, aprimorar interface, realizar otimizações e adicionar novas funcionalidades. Desta forma, o projeto descrito tem como tarefa futura aprimorar o código existente, eliminando redundâncias e otimizando funções, bom como a mudança do design das páginas para uma versão melhorada e o complemento com novas funções como a atualização automática quando detectada uma nova versão do aplicativo.

REFERÊNCIAS

AVSILLC. Real Time Streaming Protocol. 2020. Disponível em: <https://www.avsillc.com/real-time-streaming-protocol/>. Acesso em: 29 mar. 2020.

CREATE React App. 2020. Disponível em: <https://github.com/facebook/create-react-app>. Acesso em: 25 nov. 2020.

DAVID, Matthew. The 5 best tools for building progressive web apps fast. Disponível em: <https://techbeacon.com/app-dev-testing/5-best-tools-building-progressive-web-apps-fast>. Acesso em: 18 nov. 2020.

FATI, S.m.; AZAD, S.; PATHAN, A.s.k. IPTV Delivery Networks: Next Generation Architectures for Live and Video-On-Demand Services. Spi Global, Chennai, India: John Wiley & Sons, Incorporated, 2018. 372 p. Disponível em: <https://books.google.com.br/books?id=O3uMtgEACAAJ>. Acesso em: 27 mar. 2020.

INTRODUÇÃO. Facebook Inc.. Disponível em: <https://pt-br.reactjs.org/docs/getting-started.html>. Acesso em: 25 nov. 2020.

IPTV: Protocolos de comunicação. 2020. Disponível em: <https://iptv2016site.wordpress.com/protocolos-de-comunicacao/>. Acesso em: 27 mar. 2020.

ISLAM, Taohidul. Developing our first PWA using React. 2019. Disponível em: <https://blog.usejournal.com/developing-our-first-pwa-using-react-f75c9e9c9b43>. Acesso em: 25 nov. 2020.

MORGAN, Joe. How To Set Up a React Project with Create React App. 2020. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app>. Acesso em: 25 nov. 2020.

NFON (org.). RTP. 2020. Disponível em: <https://www.nfon.com/en/service/knowledge-base/knowledge-base-detail/rtp>. Acesso em: 29 mar. 2020.

PETE LEPAGE. Google Developers. What are Progressive Web Apps? 2020. Disponível em: <https://web.dev/what-are-pwas/>. Acesso em: 18 nov. 2020.

SANTONI, Muriel. Progressive Web Apps browser support & compatibility. 2018. Disponível em: <https://www.goodbarber.com/blog/progressive-web-apps-browser-support-compatibility-a883/>. Acesso em: 18 nov. 2020.

SOUTO, Mario. Roteamento no React com os poderes do React Router v4. 2018. Disponível em: <https://medium.com/collabcode/roteamento-no-react-com-os-poderes-do-react-router-v4-fbc191b9937d>. Acesso em: 02 dez. 2020.

STACKSHARE. React. 2020. Disponível em: <https://stackshare.io/react>. Acesso em: 25 nov. 2020.

Subramanian, V. (2017). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. Apress.

Y. Xiao, X. Du, J. Zhang, F. Hu and S. Guizani, "Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet," in *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126-134, November 2007.

WEINBERG, Neal; JOHNSON, Johna Till. What is MPLS: What you need to know about multi-protocol label switchinig: multi-protocol label switching is a way to insure reliable connections for real-time applications, but it's expensive, leading enterprises to consider sd-wan as a way to limit its use. Multi-protocol label switching is a way to insure reliable connections for real-time applications, but it's expensive, leading enterprises to consider SD-WAN as a way to limit its use. 2018. Disponível em: <https://www.networkworld.com/article/2297171/network-security-mpls-explained.html>. Acesso em: 27 mar. 2020.