JOÃO PEDRO FACHINI ALVARENGA

APRIMORAMENTO DE UM SERVIÇO DE STREAMING DE VÍDEO AO VIVO ADAPTATIVO

LAVRAS – MG

2021

JOÃO PEDRO FACHINI ALVARENGA

APRIMORAMENTO DE UM SERVIÇO DE STREAMING DE VÍDEO AO VIVO ADAPTATIVO

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

Prof. Dr. André Vital Saúde
Orientador

LAVRAS – MG

2021

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Alvarenga, João Pedro Fachini

Aprimoramento de um serviço de streaming de vídeo ao vivo adaptativo / João Pedro Fachini Alvarenga. – Lavras : UFLA, 2021.

51 p. : il.

Relatório de estágio supervisionado(gradução)–Universidade Federal de Lavras, 2021.

Orientador: Prof. Dr. André Vital Saúde .

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

JOÃO PEDRO FACHINI ALVARENGA

APRIMORAMENTO DE UM SERVIÇO DE STREAMING DE VÍDEO AO VIVO ADAPTATIVO
THE IMPROVEMENT OF AN ADAPTIVE LIVE VIDEO STREAMING SERVICE

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

APROVADA em 16 de novembro de 2021.

Prof. Dr. Eric Fernandes de Mello Araújo
Prof. Dr. José Monserrat Neto

Prof. Dr. André Vital Saúde
Orientador

LAVRAS – MG
2021

À minha mãe, Anice Maria Fachini Alvarenga, que dedicou sua vida em nome dos filhos, sempre me dando forças e sabedorias por todos os momentos de minha vida. Ao meu pai, Geraldo Alvarenga, que sempre se esforçou e se sacrificou pelos filhos e, sustentou meus estudos, permitindo minha evolução constante. A todos os meus amigos e colegas que conheci nessa caminhada rumo à formatura, que permitiram meu amadurecimento como pessoa, com suas relações de amizade. Por fim, mas não menos importante, à minha querida companheira, Mylene Léia Guedes de Lima, que, com seu companheirismo e amor, permitiu minha superação nas situações mais frustrantes e difíceis durante essa caminhada.

AGRADECIMENTOS

A toda a equipe da Zeester, pelo ótimo companheirismo e trabalho em equipe. Em especial, ao Thiago Alves Brum, pela oportunidade, ajuda e compreensão.

À Universidade Federal de Lavras e ao Departamento de Ciência da Computação (DCC), pelas oportunidades e exigências, que nos fazem melhorar a cada dia.

RESUMO

Este trabalho consiste em um relatório de estágio realizado em uma empresa que trabalha no desenvolvimento e manutenção de uma plataforma de conteúdo cristão baseada principalmente em streaming de vídeo. O objetivo do estágio foi o desenvolvimento e aperfeiçoamento do serviço de streaming de vídeo ao vivo adaptativo da plataforma. Para atingir esse objetivo, metodologias ágeis foram utilizadas, assim como conceitos teóricos e técnicos tais como sistemas distribuídos, transcodificação de vídeo, protocolo de streaming HLS e protocolo RTMP. Foram utilizadas tecnologias como NGINX como servidor web e FFmpeg para transcodificação de vídeo. Como resultado, o sistema obteve pertinentes melhorias ao seu serviço de streaming de vídeo ao vivo. A experiência trouxe um grande aprendizado sobre o comportamento de um software de streaming de vídeo, além de entender conceitos importantes sobre o desenvolvimento profissional de software.

Palavras-chave: Back-end, live video streaming, HLS, NGINX, FFmpeg.

ABSTRACT

This work consists of an internship report carried out in a company that works in the development and maintenance of a Christian content platform based mainly on video streaming. The objective of the internship was the development and improvement of the platform's adaptive live video streaming service. To achieve the objective, agile methodologies were used, as well as theoretical and technical concepts such as distributed systems, video transcoding, HLS streaming and RTMP protocols. Technologies such as NGINX, for web server, and FFmpeg, for video transcoding, were also used. As a result, the system has gained pertinent improvements to its live video streaming service. The experience brought a great deal of learning about the behavior of a video streaming software, in addition to understanding important concepts about professional software development.

Keywords: Back-end. Streaming. HLS. NGINX. FFmpeg.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem por objetivo abordar sobre os desenvolvimentos e aprendizados ocorridos durante um período de estágio na empresa Zeester. O processo de estágio ocorreu num tempo total de 7 meses e meio, enquanto o período a ser descrito neste relatório foi desenvolvido num intervalo de 3 meses e meio desse tempo. A função desenvolvida durante este período é a de desenvolvedor Back-end para a plataforma Blesss, com a finalidade de exercer atividades relacionadas ao seu serviço de streaming de vídeo ao vivo.

O grande desafio e objetivo do período de estágio foi o aprimoramento do serviço de streaming de vídeo ao vivo adaptativo à taxa de transmissão, do sistema Blesss. Esse serviço tem por objetivo receber uma transmissão de vídeo do cliente e, em tempo real, convertê-la em múltiplos arquivos em diferentes resoluções no disco do servidor e servi-los para os usuários do sistema.

Devido à demanda do cliente da plataforma, foi necessário deixar o serviço mais robusto, para atender com maior qualidade os usuários do sistema. Assim, a liderança da empresa obteve essas demandas e com base na necessidade do cliente, juntamente à liderança da equipe técnica formalizou pontos para melhoria do serviço. Esses pontos definiram os objetivos do trabalho, trata-se de adicionar novas funcionalidades, como mais opções de resoluções para as transmissões, além de melhorias na segurança e comunicação com outro serviço interno do sistema da plataforma.

A abordagem utilizada para atingir tal resultado foi incrementar tais funcionalidades no já existente serviço, estudando e compreendendo ele adequadamente, para assim, implementar os requisitos necessários e adequar seus processos à visão da liderança. Dessa forma, foram trabalhadas com inúmeras tecnologias como NGINX, HLS, FFmpeg, tudo isso em um ambiente de desenvolvimento pautado em metodologias ágeis.

Como resultado, obteve-se um serviço com todas as melhorias desejadas e integrado às aplicações da plataforma Blesss, sendo utilizado por vários usuários, além de ser adquirido novos conhecimentos aprofundados sobre as tecnologias e conceitos trabalhados.

Neste texto serão primeiramente abordados detalhes sobre a organização e o ambiente de trabalho do período de estágio (Seção 2). Em seguida, conceitos importantes para o entendimento do trabalho (Seção 3). Após isso, será detalhada a metodologia e ferramentas trabalhadas (Seção 4) e os resultados do período de desenvolvimento (Seção 5). Por fim, será apresentada uma conclusão, contendo reflexões sobre aprendizados obtidos durante o período em que o trabalho foi realizado (Seção 6).

2 ORGANIZAÇÃO

O estágio foi realizado na empresa Zeester, localizada na cidade de Lavras, Minas Gerais. A empresa desenvolve soluções tecnológicas para projetos de terceiros. Este capítulo apresenta o produto objeto deste trabalho (Seção 2.1), principal produto da empresa e a constituição das equipes (Seção 2.2).

2.1 Blesss

A Blesss é uma plataforma de conteúdo cristão focada em conteúdo de vídeo, sob demanda ou ao vivo. Além de vídeos, a plataforma possui conteúdos diversificados como artigos e livros. Possui mais de 1000 horas de conteúdo de vídeo sob demanda. Para o gerenciamento de seu conteúdo, dispõe de uma ferramenta, a Blesss Studio, que oferece a possibilidade de gerenciamento de conteúdos e usuários da plataforma.

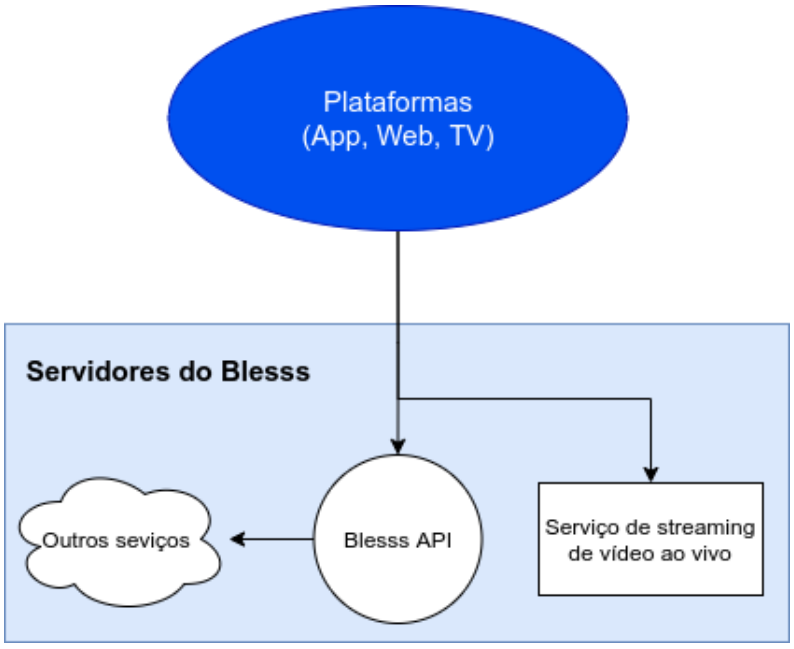
A Figura 2.1 mostra como funciona a estrutura e comunicação entre módulos do sistema do Blesss. A comunicação entre a plataforma e serviços é feita através de requisições HTTP. As setas indicam a ação de uma requisição. A plataforma possui diversos serviços presentes em seus servidores, porém, entre eles há uma API (Application Programming Interface) principal, denominada Blesss API, que realiza o gerenciamento do conteúdo da plataforma. Além disso, realiza comunicações e controle de alguns outros serviços secundários, como, por exemplo, os serviços de gerenciamento de usuário e armazenamento de arquivos estáticos (como imagens).

2.2 Estrutura das equipes

A empresa possui ao total cinco grandes equipes. São elas as de liderança, design, desenvolvedores, marketing e comunicação. Suas funções são:

- **Equipe de liderança**: É responsável por administrar e decidir o caminho da empresa, tomando as grandes decisões sobre os seus produtos.
- **Equipe de design**: Trabalha no projeto e protótipo visual das plataformas.
- **Equipe de desenvolvimento**: Trabalha no desenvolvimento e manutenção da plataforma e seu sistema, atendendo aos requisitos e visão da liderança e design.
- **Equipe de marketing**: Trabalha no alcance de público da plataforma, visando deixá-la mais conhecida e divulgando conteúdos.
- **Equipe de comunicação**: É responsável pelo contato direto com os usuários da plataforma, seja em casos de dúvidas ou eventuais problemas.

Figura 2.1 – Esboço da estrutura e comunicação principal do sistema Blesss



Fonte: do autor

A equipe de desenvolvimento é composta tanto por desenvolvedores Back-end, que se encarregam do desenvolvimento dos serviços e APIs, quanto Front-

end, que se encarregam de desenvolver as interfaces e programas que interagem diretamente com o usuário. Por fim, na equipe existem também desenvolvedores com a responsabilidade sobre a infraestrutura dos serviços e do servidor.

3 REFERENCIAL

Este capítulo tem o propósito de discorrer sobre as tecnologias, ferramentas, conceitos e técnicas utilizadas durante o desenvolvimento e aperfeiçoamento dos módulos trabalhados durante o estágio. Serão primeiramente abordados conhecimentos relacionados à metodologia e processo de desenvolvimento. Em seguida, serão apresentadas noções sobre streaming e transcodificação de vídeo. Ao final, será abordado o servidor NGINX e seu módulo Real Time Messaging Protocol (RTMP), utilizado neste trabalho.

3.1 Metodologias ágeis e Scrum

O termo *metodologias ágeis* surgiu em 2001, com o estabelecimento de princípios comuns entre diversos métodos que seriam futuramente conhecidos como métodos ágeis (como Scrum, XP, DSDM, entre outros) (SOARES, 2004). O amadurecimento das metodologias ágeis resultou no *manifesto ágil*, que se apresentou como um movimento de resposta aos métodos de desenvolvimento tradicionais que se caracterizavam como inflexíveis e lentos (STOPA; RACHID, 2019).

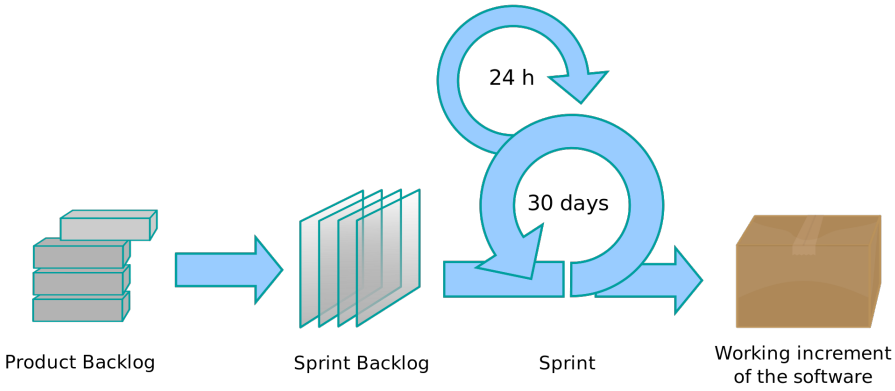
Os valores do movimento ágil foram estabelecidos como:

> "Indivíduos e interações ao invés de processos e ferramentas.
> Software executável ao invés de documentação.
> Colaboração do cliente ao invés de negociação de contratos.
> Respostas rápidas a mudanças ao invés de seguir planos."
> (BECK et al., 2001)

Um dos maiores exemplos de metodologia ágil é o Scrum, um framework utilizado desde a década de 1990 para gerenciar o trabalho em produtos complexos (STOPA; RACHID, 2019).

A Figura 3.1 detalha os processos do Scrum resumidamente, onde temos os seguintes conceitos (PAULA, 2016):

Figura 3.1 – Processos do Scrum



Fonte: (PAULA, 2016)

- Sprint: É o nome dado a cada ciclo de entrega do projeto.
- Product Backlog: É o conjunto de objetivos de um projeto.
- Sprint Planning Meeting: São reuniões que acontecem no início de cada Sprint, com a finalidade de definir os itens do Product Backlog que serão desenvolvidos durante a Sprint.
- Sprint Backlog: São as tarefas do Product Backlog que serão realizadas na Sprint. É produto da Sprint Planning.
- Daily Scrum: Reunião diária para acompanhamento das tarefas da Sprint.
- Sprint Review Meeting: Reunião realizada ao final da Sprint para refletir sobre o que foi realizado, os resultados e valor do que foi produzido e ao fim, finalizar a Sprint atual e começar uma próxima.

3.2 Git e GitLab

O Git é uma ferramenta de código aberto para controle de versionamento de projetos. Ele possui várias funcionalidades que permitem o gerenciamento de

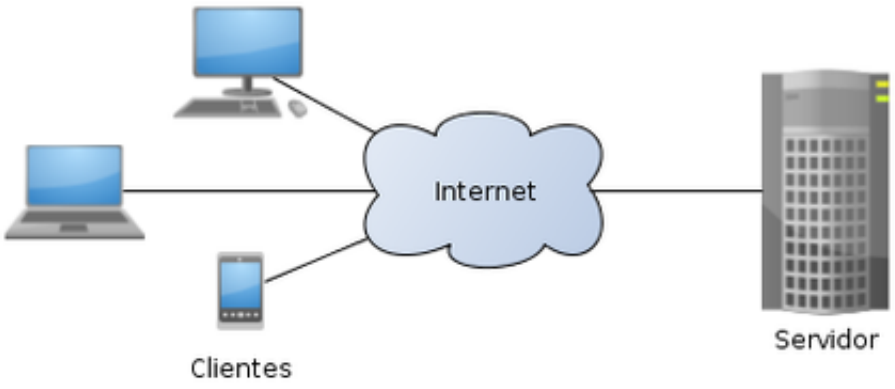
diferentes ramos de um código-fonte, permitindo a colaboração de múltiplos indivíduos em um projeto facilmente (GIT, 2021).

Já o GitLab, é uma plataforma de hospedagem e gerenciamento de repositórios de código-fonte que utiliza o controle de versão Git (RIBEIRO, 2021).

3.3 Arquitetura Cliente-Servidor

A arquitetura cliente-servidor, ilustrada na Figura 3.2, é uma arquitetura de sistemas distribuídos. Nela o sistema possui duas categorias de processos: um responsável por gerir as informações (servidor) e outro pela obtenção e visualização dos dados pelo usuário (cliente) (CANALTI, 2018).

Figura 3.2 – Modelo Cliente Servidor



Fonte: (UFRJ, 2016)

Dentro desse modelo surgem diversas aplicações, dentre elas os servidores web, que servem, via protocolo HTTP, páginas contendo arquivos HTML, CSS ou JavaScript (JS) ao cliente, um navegador web (MOZILLA, 2021). Essas páginas podem se comunicar e consumir outros serviços via rede de Internet, utilizando o mesmo modelo cliente-servidor.

Do conceito cliente-servidor surgem outros dois: Front-end e Back-end. O Front-end é o conjunto de elementos da interface. Fornece toda a experiência do

usuário na página e tem como responsabilidade toda a comunicação com serviços externos, como APIs. O Back-end, por sua vez, fornece os serviços (APIs, por exemplo) e implementa os processos que funcionam nos servidores, assim como as regras de negócio necessárias para o funcionamento da aplicação.

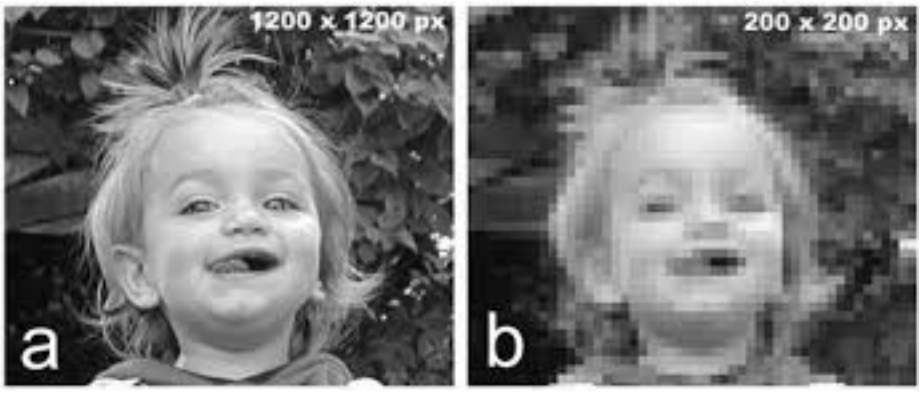
É importante observar que o processo das aplicações Back-end não são visíveis ao usuário. Quem se encarrega de interagir com um usuário numa aplicação web é o Front-end, que utiliza as aplicações Back-end de auxílio para suas operações (HSM, 2020).

3.4 Resolução de vídeo

A resolução de um vídeo é o número de pixels contidos em cada quadro dele (LEONARD; KURNIAWAN, 2021).

Essa medida é obtida pelas quantidades de pixels na horizontal e vertical. Por exemplo, 1280×720 significa que temos 1280 pixels horizontais e 720 verticais. Na Figura 3.3 é possível observar que quanto maior a resolução mais definida a imagem fica. Por consequência, maior é a quantidade de pixels para representá-la.

Figura 3.3 – Diferença de uma resolução de 1200 × 1200 px para 200 × 200 px



Fonte: (LAURENT, 2020)

Como a proporção padrão atual é a 16:9, as resoluções mais comuns encontradas em vídeos são 7680 × 4320 pixels (8K), 3840 × 2160 (4k ou Ultra HD), 2560 × 1440 (2k ou Quad HD), 1920 × 1080 (1080p ou Full HD) e 1280 × 720 (720p ou HD) (LEONARD; KURNIAWAN, 2021). Além dessas, temos resoluções abaixo do HD, como, por exemplo, 640 × 360 (360p) e 320 × 180 (180p).

3.5 Taxa de bits

A taxa de bits ou *bitrate* é um indicador utilizado para determinar a quantidade de dados transmitidas durante um período de tempo (GOGONI, 2020). Normalmente a medida kb/s é utilizada para representá-la.

A taxa de bits por segundo de um vídeo determina o volume de dados máximo que um segundo do vídeo possui (FELIPE, 2020). É importante notar que quanto maior a taxa de bits, mais informações é possível armazenar no vídeo, permitindo assim o armazenamento de vídeos com mais detalhes. Quanto mais detalhes tem um vídeo, maiores são os arquivos que o armazenam. No contexto de streaming de vídeo, quanto mais detalhado o vídeo (maior resolução, por exemplo), maior a quantidade de dados a ser transmitida ao cliente, portanto, maior é a taxa de bits exigida, o que significa um maior consumo de rede para transmissão dos conteúdos.

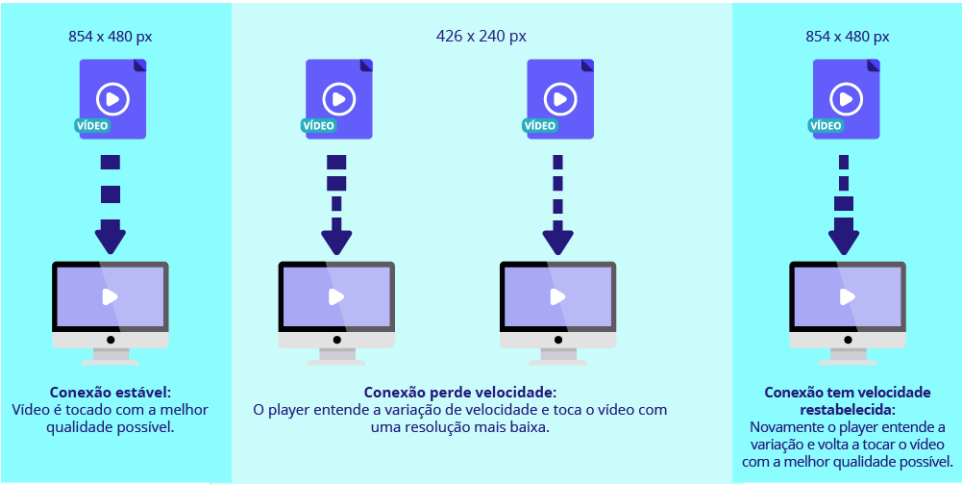
3.6 Streaming de vídeo adaptativo e HLS

Streaming adaptativo, *adaptative streaming* ou até mesmo *adaptative bitrate streaming* é uma categoria de streaming de vídeo que visa solucionar o problema de acessibilidade de streaming de vídeo via Internet em diferentes dispositivos e conexões. É uma técnica utilizada com intuito de fazer a adaptação do volume de dados enviados ao usuário ao longo da reprodução de um vídeo conforme a conexão do usuário (GOMES, 2020).

Esse controle tem por base a taxa de bits ligada a diferentes resoluções do vídeo. Dessa forma, cada resolução possui uma taxa de bits adequada. Resoluções menores requerem menor taxa de bits e vice-versa. O streaming adaptativo consiste em controlar a transmissão do vídeo, para ser entregue ao usuário dependendo da largura de banda disponível. A técnica permite que o usuário veja a melhor resolução que sua banda de conexão permite. (JONES, 2018).

Na Figura 3.4 podemos observar como a técnica pode ser utilizada para controle de qualidade conforme a banda do usuário. É importante notar na figura que a qualidade se adapta à conexão em tempo real. Para fluir a transmissão sem possíveis travamentos para o usuário, o vídeo pode ser regulado adequadamente conforme a conexão do usuário.

Figura 3.4 – Comportamento de um streaming de vídeo adaptativo.

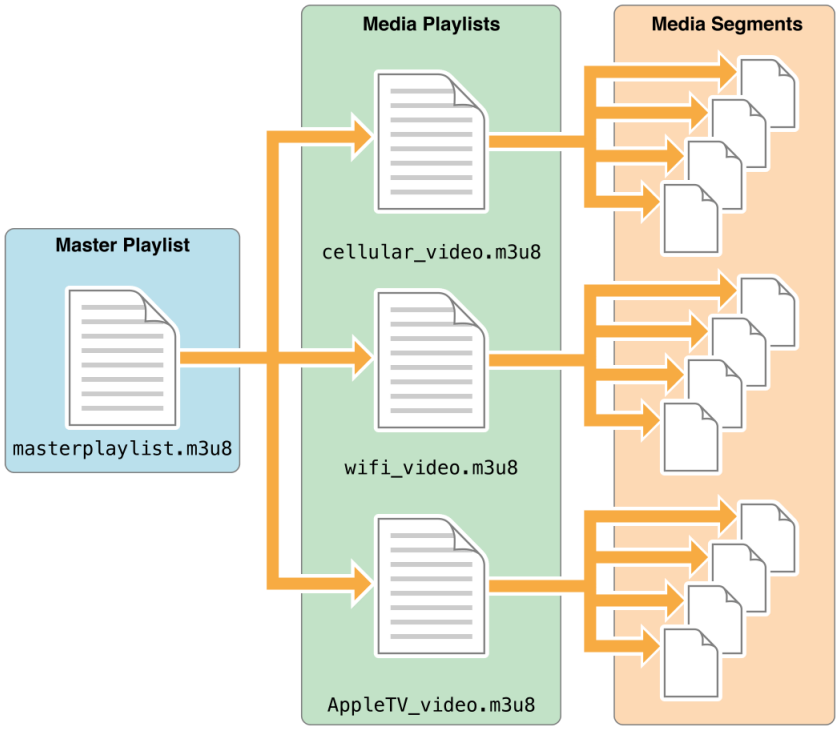


Fonte: (GOMES, 2020)

Existem várias implementações do conceito de streaming de vídeo adaptativo. Um deles é o HLS (Http Live Streaming), ele tem a ideia de realizar todo o processo de streaming de vídeo ao usuário utilizando o protocolo HTTP. Para isso, utiliza um servidor web HTTP que servirá os segmentos do vídeo sendo requisitados pelo cliente (APPLE, 2014).

No HLS, segmentos de vídeo de cada resolução são indexados em um arquivo playlist, assim o player de vídeo consegue saber como acessar cada um deles. Para acessar esses índices existe um arquivo chamado master playlist, que serve como um índice para cada arquivo playlist (APPLE, 2014). O formato de arquivo dos segmentos é o **ts** e o das playlists, o **m3u8**. É possível visualizar essa estrutura de arquivos na Figura 3.5.

Figura 3.5 – Estrutura dos arquivos de um vídeo em HLS



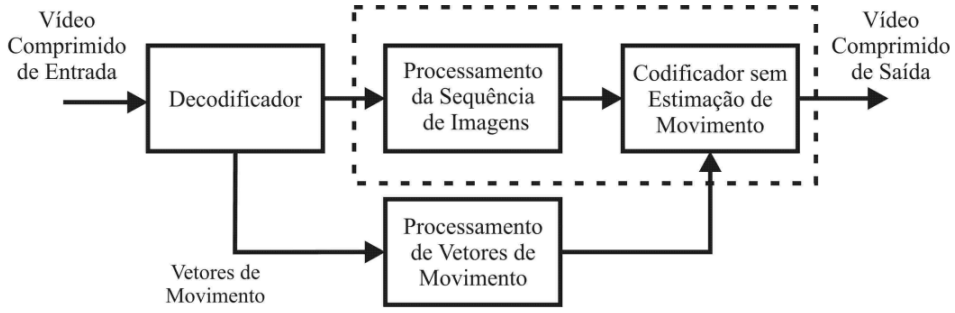
Fonte: (APPLE, 2014)

3.7 Transcodificação de vídeo e FFmpeg

A transcodificação é um processo técnico responsável por transformar um vídeo, áudio ou outro tipo de arquivo, alterando alguma de suas propriedades como, no caso de vídeo, a taxa de bits (CROSSHOST, 2021). Esse processo envolve a decodificação de um arquivo, processamento para atingir as modifica-

ções desejadas e, por fim, a codificação que gerará um novo arquivo em um novo formato na saída. A Figura 3.6 ilustra tal processo para um vídeo.

Figura 3.6 – Modelo de um transcodificador de vídeo.



Fonte: (REGIS et al., 2007)

A transcodificação de vídeo é utilizada, no contexto de streaming de vídeo adaptativo, para transformar um vídeo já produzido e codificado em um determinado formato, para outros formatos, visando atender a clientes com diferentes taxas de bits disponíveis em suas conexões de Internet. Esse processo é responsável por criar diversas cópias de diferentes qualidades do vídeo, de modo a transformá-lo em conteúdo acessível à diversidade de usuários da web (CROSSHOST, 2021).

Dentre as várias modificações que podem ser feitas no processo de transcodificação, uma importante é o codec. Através dele é possível definir uma codificação específica para vídeo ou áudio. Ele realiza tanto o processo de compressão e codificação do conteúdo em questão, quanto o processo de descompressão e decodificação quando o conteúdo for executado (CLAUDIO, 2014).

Para realizar o processo de transcodificação existem ferramentas disponíveis com tais finalidades, como o FFmpeg. O FFmpeg fornece a possibilidade de execução de inúmeros processos de conversão de arquivos multimídia, como transcodificação, decodificação, codificação, aplicação de filtros, entre outros (FFMPEG, 2021).

3.8 NGINX e NGINX-RTMP

NGINX (Engine X) é um servidor web de código livre, com foco em performance, construído em C++. Suas principais aplicações são como proxy reverso e servidor HTTP (NGINX, 2021). Apesar de ter seu foco em tais aplicações, permite configurações e também extensões. A extensão relevante para este trabalho é o NGINX-RTMP.

O NGINX-RTMP é um módulo construído para o NGINX que adiciona ao software funcionalidades relacionadas à comunicação via protocolo Real Time Messaging Protocol (RTMP) e streaming de vídeo ao vivo (LAHN, 2021).

O RTMP é um protocolo baseado em TCP, criado para trabalhar com streaming de vídeo, originalmente para trabalhar com o Adobe Flash Player (EMILY, 2021). Hoje em dia, porém, seu uso mais comum é em conjunto com protocolos como o HLS, onde exerce a função de receber um conteúdo de vídeo para ser servido. Esse processo é chamado de *RTMP ingest*, em português Ingestão RTMP.

Com o NGINX-RTMP é possível utilizar o NGINX para tais finalidades. O NGINX-RTMP também oferece funcionalidades para gravação do vídeo em formato FLV (Flash Video), um formato de vídeo denotado pela extensão ".flv", utilizado também no antigo Adobe Flash Player. (CHISHTI, 2019).

Por fim, o NGINX-RTMP também permite a execução de aplicativos externos como o FFmpeg, para o processo de transcodificação da transmissão. O módulo também possui uma funcionalidade para codificar o vídeo em HLS. Dessa forma, é possível realizar a transcodificação via FFmpeg e enviar a stream novamente para o NGINX-RTMP terminar o processo de codificação para HLS.

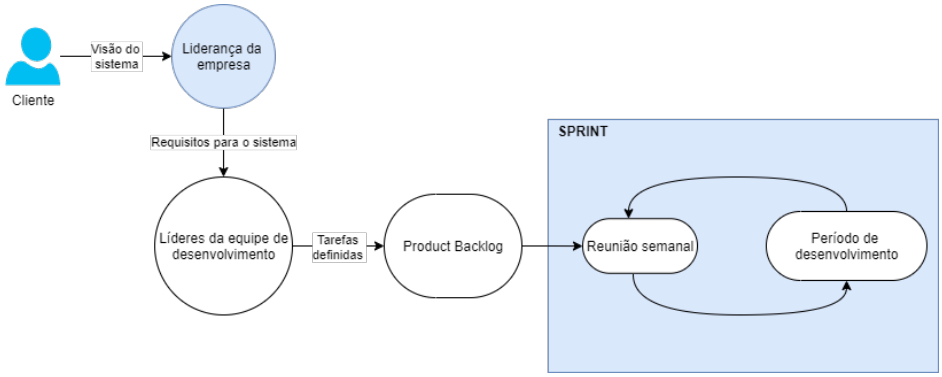
4 METODOLOGIA

O presente capítulo tem por objetivo apresentar a metodologia aplicada durante o desenvolvimento do estágio. Neste capítulo serão abordados o fluxo de tarefas e processos durante o desenvolvimento (Seção 4.1), os requisitos do trabalho e também como foram adquiridos (Seção 4.2) e, por fim, as tecnologias utilizadas para atingir o objetivo do projeto (Seção 4.3).

4.1 Ciclo de desenvolvimento

A metodologia de desenvolvimento da empresa em questão é fortemente baseado no Scrum e no conceito de metodologias ágeis. Nesta sessão serão abordados o ambiente de desenvolvimento do estágio e os processos presentes no desenvolvimento de software da empresa, que pode ser visualizado na Figura 4.1.

Figura 4.1 – Processo de desenvolvimento na empresa



Fonte: do autor.

O fluxo de desenvolvimento tem seu início na liderança da empresa, que decide novos requisitos para a plataforma baseada na visão do cliente, seja esse requisito uma nova funcionalidade ou possíveis melhorias. Esses requisitos são comunicados para a liderança da equipe de desenvolvimento, num processo em que são discutidas as dificuldades, necessidades e a viabilidade dos requisitos para a empresa.

Após isso, é realizado um processo de refinamento dos requisitos pela liderança de desenvolvimento da empresa. Ele serve para definir o que precisa ser feito com mais detalhes e também, adicionar no Product Backlog as tarefas que precisam ser realizadas para a equipe alcançar os objetivos desejados.

Cada Sprint possui duração de uma semana, tendo seu início e fim marcado por uma reunião semanal. Essas reuniões possuem tanto a finalidade de Sprint Review, quanto de Planning. Portanto, nela é o momento onde é decidido quais tarefas do Product Backlog deverão ser realizadas durante a Sprint.

Durante a reunião semanal também é realizado o Planning Poker. Nele, cada uma das tarefas selecionadas para desenvolvimento na Sprint são detalhadas e, são eventualmente atribuídas com uma pontuação que determina a complexidade da tarefa, chamada Story Points.

Eles são determinados por uma votação em que todos os desenvolvedores da equipe participam. Suas possíveis notas seguem uma sequência de Fibonacci modificada [1]. Na empresa é determinado que a maior tarefa possível a ser realizada em uma Sprint possui uma pontuação 5. Assim, quando alguma tarefa ultrapassa essa pontuação ela é quebrada em tarefas menores, para ser possível um progresso dela durante esse ciclo de Sprints.

Ao final da reunião semanal, cada tarefa possui sua responsabilidade delegada a um desenvolvedor, que possui conhecimento e responsabilidade sobre a área de desenvolvimento da mesma (tarefas relacionadas ao Back-end são delegadas a desenvolvedores Back-End, por exemplo).

Com isso, é iniciada uma nova Sprint. Existem diversas categorias de tarefas. Para cada tarefa são necessárias diferentes abordagens. O desenvolvimento delas também não necessariamente é isolado unicamente à pessoa com responsabilidade sobre ela. O desenvolvedor pode realizar reuniões e pedir ajuda aos outros membros da equipe para conseguir desenvolver sua atividade. Existem tarefas

[1] Disponível em <https://agilepink.com/planning-poker/2019/>

ligadas puramente à pesquisa e descoberta de novos conhecimentos para serem aplicados a outras tarefas futuras. Essas tarefas exigem bastante pesquisa e geração de documentos com o conhecimento descoberto, para facilitar seu uso futuro pela equipe.

Tarefas ligadas ao desenvolvimento de uma nova funcionalidade ou correção de falhas a um serviço precisam ser testadas e validadas no ambiente de teste no servidor. Essa prática é necessária para realizar a simulação do serviço funcionando com outros componentes do sistema e ver como se comporta a aplicação no servidor. Esse processo de teste e validação é realizado pelos próprios desenvolvedores.

No caso deste trabalho, o código-fonte é sempre relacionado a configurações do NGINX e está presente num repositório no Git na plataforma Gitlab. Assim, para desenvolvimento das atividades é sempre modificado em um ramo do Git destinado à atividade em questão. Sempre que o código precisa de ser enviado a validações no ambiente de teste, é necessário enviar as modificações realizadas no ramo da atividade ao ramo de teste, para que seja possível atualizar o código no servidor de teste.

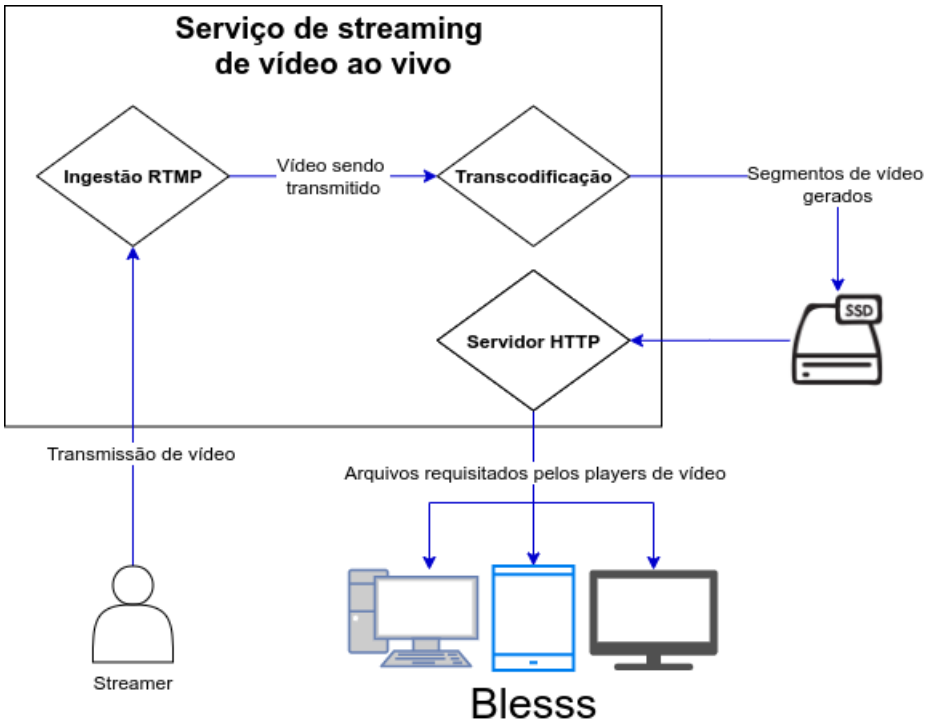
Assim que realizada a validação técnica, são apresentadas para a liderança da equipe as novas funcionalidades finalizadas e assim que todo o processo é aprovado, é realizado o laudo para implantação da nova funcionalidade em ambiente de produção.

Algumas tarefas são definidas como finalizadas quando estão em produção e outras quando estão simplesmente em teste, variando conforme a visão da liderança e a categoria da tarefa. Tarefas de descoberta são definidas como finalizadas quando o conhecimento adquirido no seu processo é gerado em forma de documentos.

4.2 Requisitos para aperfeiçoamentos do serviço

O serviço previamente existente na plataforma já realizava o processo de streaming completo, como pode ser constatado na Figura 4.2. Na figura é possível observar que o serviço possui diversos módulos. O serviço pré-existente já seguia o protocolo HLS para fornecer o streaming de vídeo, assim como descrito na Seção 3.6. As setas indicam o fluxo da transmissão de vídeo no sistema. Inicialmente, o serviço recebia uma transmissão de vídeo via servidor RTMP e realizava sua transcodificação para três diferentes resoluções (180p, 360p e 720p), gerando vários segmentos de vídeo e os servindo via servidor web HTTP.

Figura 4.2 – Serviço de streaming de vídeo ao vivo adaptativo antigo



Fonte: do autor

Apesar do serviço pré-existente realizar o processo de streaming completo, com base na demanda do cliente, havia a necessidade de aprimorá-lo, principal-

mente por questões relacionadas a um evento anual que precisava ser transmitido na plataforma Blesss.

Por isso, as necessidades para o serviço de streaming de vídeo ao vivo foram discutidas entre a liderança da empresa e o cliente. A partir dessa discussão, foram definidos pontos com várias melhorias ao serviço para atender às expectativas do evento. Com base nesses pontos, a liderança da empresa se reuniu com a liderança da equipe de desenvolvimento e foram definidas as melhorias a seguir:

- O serviço deve conseguir disponibilizar o conteúdo da transmissão inteiro ao usuário, enquanto ela estiver sendo transmitida.
- O serviço deve possuir duas rotas para transmissão de vídeo: uma para transmitir vídeos nas resoluções 180p, 360p e 720p e outra para transmitir a 180p, 360p, 720p e 1080p.
- O serviço deve conseguir disponibilizar o vídeo transmitido em formato FLV, após a transmissão ser finalizada.
- O serviço deve permitir transmitir vídeos somente se a Blesss API fornecer autenticação.
- O serviço deve informar à Blesss API quando a transmissão começar ou terminar.

Após isso, houve um processo de refinamento nesses requisitos sendo nele geradas diversas tarefas no Product Backlog. O desenvolvimento do estágio prosseguiu baseando-se nas tarefas relacionadas ao serviço de streaming de vídeo ao vivo.

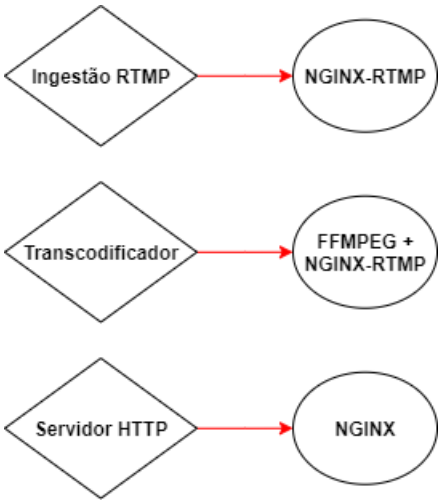
4.3 Uso das tecnologias no desenvolvimento do serviço

O serviço já existente, junto a suas tecnologias, foi utilizado como base para construção das melhorias. Assim, o protocolo de streaming utilizado foi

o HLS, enquanto a tecnologia utilizada para construção do servidor web foi o NGINX.

Na Figura 4.3 é possível observar um mapeamento de cada módulo em relação à tecnologia que ele utiliza. Como o serviço é baseado em NGINX, seu desenvolvimento utiliza o NGINX nativo para servir arquivos de vídeo e também, um módulo extra, NGINX-RTMP, com a finalidade de tratar da comunicação RTMP e realizar o processo de ingestão da transmissão de vídeo.

Figura 4.3 – Tecnologias utilizadas para o trabalho em cada módulo



Fonte: do autor

Por fim, para o processo de transcodificação é utilizado o FFmpeg juntamente ao módulo NGINX-RTMP. O FFmpeg é utilizado para decodificar a transmissão recebida, gerar diferentes resoluções para ela e configurar aspectos técnicos como taxa de bits para cada resolução e o codec do vídeo. O NGINX-RTMP é utilizado para a etapa final de codificação, transformando o formato do vídeo para HLS e salvando no disco.

Assim, foi necessário durante o processo de desenvolvimento, compreender o funcionamento do serviço e de suas tecnologias já existente da forma como

é contextualizada acima. Além disso, foi necessário estudar as documentações de suas tecnologias, para aplicar todo esse conhecimento e evoluir o serviço final.

5 RESULTADOS

Neste capítulo serão apresentados os resultados do processo de desenvolvimento obtidos durante o período de estágio. Temos como resultado um serviço aprimorado conforme os requisitos do cliente da plataforma Blesss.

A Seção 5.1 abordará em detalhes tudo que foi alterado e produzido no serviço e a Seção 5.2 apresentará como o serviço foi utilizado quando integrado à plataforma Blesss.

5.1 Serviço aprimorado e seus módulos

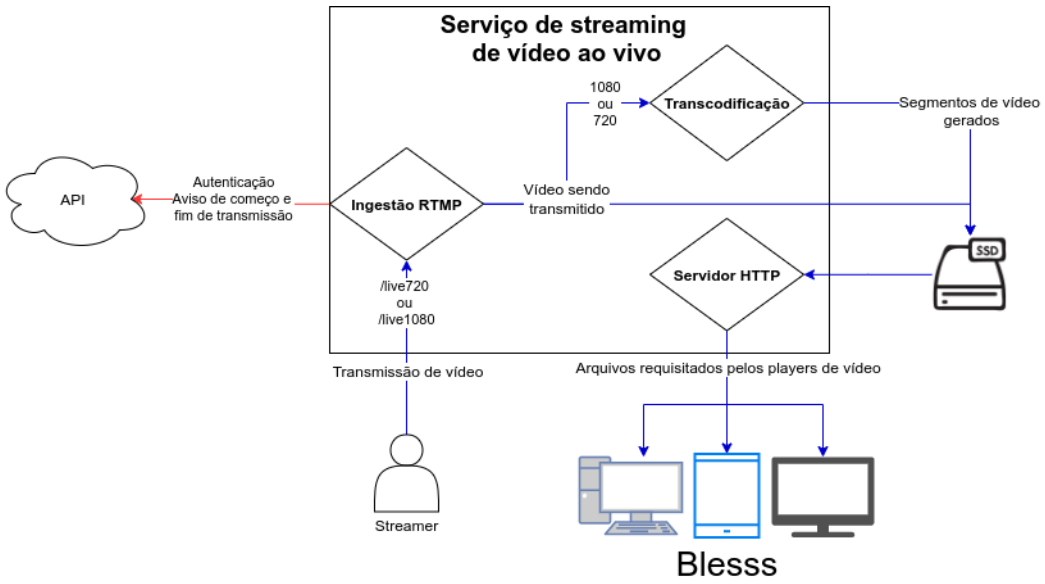
O serviço aprimorado realiza todo o processo de streaming de vídeo em tempo real, desde a recepção da transmissão realizada a ele, até a disponibilização dos arquivos do vídeo final, em várias resoluções. Para transmitir um vídeo pelo serviço é necessário possuir o endereço RTMP do módulo de ingestão do serviço e também uma chave de transmissão. Para auxiliar a transmissão do cliente ao serviço é utilizado um software com a finalidade de realizar transmissões de vídeos. O software utilizado neste trabalho foi o OBS [1].

Após a transmissão, se a chave do usuário possuir autenticação, a conexão estará estabelecida e todos os segmentos do vídeo em diversas resoluções serão disponibilizados por um servidor HTTP. Para um player de vídeo baseado em HTML5 transmitir esse vídeo, é preciso apenas do endereço que representa o arquivo da master playlist do vídeo correspondente à transmissão em questão.

Na Figura 5.1 é possível obter uma ideia geral de como esses módulos se comunicam e de como o serviço se comunica com o usuário final da plataforma Blesss. Assim como na Figura 4.2, o fluxo da transmissão é representado pelas setas azuis, neste caso há também uma seta vermelha que indica a ação de uma requisição HTTP realizada pelo serviço de streaming.

[1] Disponível em <https://obsproject.com/pt-br>

Figura 5.1 – Serviço de streaming de vídeo ao vivo adaptativo após aprimoramentos



Fonte: do autor

Como é possível visualizar numa comparação entre as Figuras 5.1 e 4.2, as mudanças ao fluxo do serviço se iniciam desde o começo da transmissão em que existem duas possíveis rotas do serviço diferentes para realizar a transmissão. O indivíduo que for realizar o streaming deve, portanto, escolher a rota /live720 ou /live1080 para realizar a transmissão. A /live720 irá disponibilizar o vídeo transmitido na resolução máxima 720p, enquanto a /live1080 terá a 1080p como sua resolução máxima.

Além disso, há duas outras diferenças no fluxo adicionadas no módulo de ingestão. A primeira é que há agora requisições HTTP à Blesss API. Elas partem do módulo de ingestão RTMP, são realizadas para autenticar a transmissão e fornecer informação de início e fim da transmissão à API. A segunda é que o sistema de ingestão também salva em disco todo o vídeo sendo transmitido, para ele ser futuramente disponibilizado para processamento por outros serviços do sistema.

Todas as alterações e aprimoramentos descritos brevemente acima serão abordadas em mais detalhes nas seguintes subseções, com cada uma abordando o que foi produzido e como funciona cada módulo do serviço.

5.1.1 Módulo de ingestão RTMP

O módulo de ingestão RTMP tem por objetivo gerenciar a recepção da transmissão de vídeo que ocorre neste caso por RTMP, portanto, sua principal função é receptar a transmissão e enviar para transcodificação.

Nele foram adicionadas várias funcionalidades requisitadas para a melhoria. Primeiramente foi adicionada uma funcionalidade para gravação da transmissão sendo receptada pelo serviço. Assim, é salva toda a transmissão realizada, no formato FLV, em um diretório específico do disco destinado exclusivamente a essa função.

Também foi implementada uma comunicação via requisição HTTP para o serviço se comunicar com a Blesss API. Essa comunicação ocorre em dois momentos no ciclo de uma transmissão: assim que ela inicia e logo quando ela é perdida ou finalizada. A comunicação que ocorre no momento de início da transmissão também serve como meio de autenticação, pois na requisição são enviados os dados da chave da transmissão utilizadas pela Blesss API para gerenciar se ela pode ser transmitida ou não. Se a transmissão for autorizada, a API retornará uma resposta com status (do protocolo HTTP) 200.

A funcionalidade central do módulo de ingestão também sofreu mudanças. Agora o serviço possui duas rotas RTMP para ingestão, /live720 e /live1080. Sendo a /live720 para a transmissão que será disponibilizada nas resoluções 180p, 360p e 720p e a /live1080 para as resoluções 180p, 360p, 720p e 1080p. De acordo com essa regra, cada uma das rotas envia a transmissão para a transcodificação adequada.

5.1.2 Módulo de transcodificação

Este módulo do serviço tem responsabilidade pelo processo de transcodificação. Este processo é realizado em tempo real no serviço, por se tratar de streaming ao vivo.

Nele, foram criados duas categorias de transcodificações diferentes, uma para gerar as resoluções 180, 360 e 720p e outra para produzir o vídeo nas resoluções 180, 360, 720 e 1080p. Dessa forma a transcodificação já existente, que realizava o processo de traduzir o vídeo para as resoluções 180, 360 e 720p foi mantida e, com base nela, foi criado também o outro transcodificador. Ele aborda todos os processos de transcodificação das resoluções já existentes e acrescenta um novo, capaz de gerar segmentos de vídeo em 1080p.

Neste caso da geração de segmentos de vídeo em 1080p, as opções como codec, taxa de bits de aúdio, entre outras, foram mantidas como padrão, apenas alterando a resolução a ser convertido. O número de taxa de bits de vídeo foi decidido com base em testes realizados internamente, em que um valor de 4000kbps se mostrou eficiente em relação a fatores como qualidade e desempenho na entrega do vídeo.

Por fim, para garantir a persistência dos segmentos, de modo a atender o requisito da transmissão ser acessível a qualquer ponto enquanto estiver ao vivo, foi adicionada uma opção ao processo de codificação do HLS. Esta opção garante que os segmentos não sejam voláteis e persistam no disco, sem tempo de expiração, ao contrário do comportamento anterior.

5.1.3 Módulo de servidor HTTP

O servidor HTTP do serviço tem por objetivo servir, por meio de requisições HTTP, os arquivos gerados pela transcodificação. Esse serviço faz um mapeamento do diretório em que os arquivos HLS das transmissões são salvos, para deixá-los disponíveis para o cliente. Sendo assim, o módulo já existente já atendia

às novas configurações de resoluções, com a master playlist de cada transmissão indicando a rota em que cada segmento de cada resolução é acessível. Portanto, neste módulo só foi acrescentada, no processo de melhoria, uma configuração do NGINX para deixar acessível também o arquivo de gravação da transmissão, em formato FLV.

Para deixar a gravação acessível foi necessário restringir seu acesso. Para isso foi utilizado uma restrição de IP, fazendo assim apenas o IP do servidor de APIs do Blesss conseguir acessar a rota que serve os arquivos de gravação.

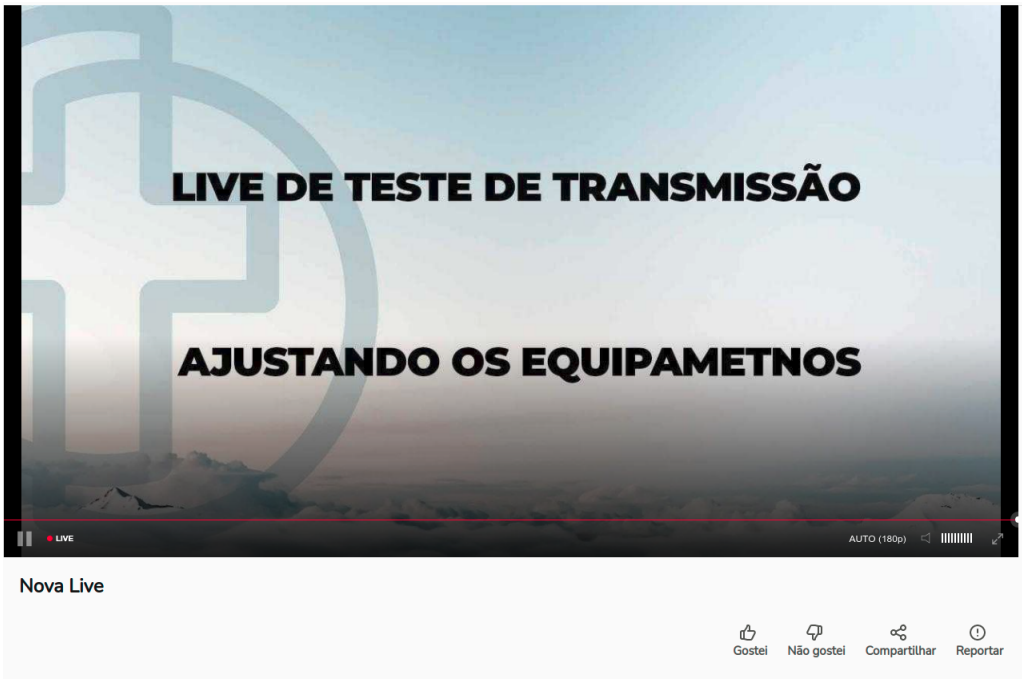
5.2 Uso do serviço após o aprimoramento

Quando finalizado o desenvolvimento, o serviço foi repassado para os desenvolvedores ligados à infraestrutura para instalação em ambiente de produção. Para isso eles realizaram um estudo sobre escala e também necessidade de especificação do servidor com base no planejamento futuro sobre o uso do serviço.

As rotas necessárias foram passadas para outros desenvolvedores Back-end responsáveis por outros serviços do sistema realizarem os devidos controles. Também foi prestado auxilio a desenvolvedores Front-end para integração com os players de vídeo. Na Figura 5.2 é possível de observar uma transmissão ao vivo sendo transmitida na plataforma Blesss web.

Os resultados do serviço foram considerados muito satisfatórios para o sistema, na visão da liderança da empresa. Além disso, o serviço foi consumido por diversos usuários da plataforma, com um destaque para o fato de o serviço ter sido utilizado durante o evento anual do cliente, quando foram disponibilizadas múltiplas transmissões simultâneas, algumas através da rota /live1080 e outras /live720. O pico de uso do serviço ocorreu durante o evento anual do cliente, quando o serviço realizou mais de dez transmissões simultaneamente, com cerca de 1200 usuários simultâneos na plataforma, assistindo alguma dessas transmissões, ao total.

Figura 5.2 – Player de vídeo do Blesss acessando uma transmissão do serviço.



Fonte: do autor

6 CONCLUSÃO

O desenvolvimento das atividades durante o período de estágio resultaram em mudanças críticas e cruciais ao serviço e funcionamento do streaming de vídeo ao vivo da plataforma Blesss, utilizado amplamente pelos usuários da plataforma.

Para conseguir esse resultado, foram envolvidas competências técnicas e habilidades para resoluções de problemas, assim como capacidade de estudo constante, desde os requisitos ao aprofundamento nas tecnologias necessárias. Além disso, o ambiente de trabalho baseado em metodologias ágeis possibilitou um desenvolvimento das atividades de maneira profissional.

Além das habilidades citadas anteriormente, habilidades e conhecimentos teóricos foram cruciais para um melhor desenvolvimento do estágio. A seguir são destacados os conhecimentos abordados no curso de Ciência da Computação mais importantes para a resolução do estágio:

- **Redes e Sistemas distribuídos**: O conhecimento de como funciona a comunicação e processo de um sistema distribuído foi aplicado em toda a comunicação do serviço, tanto no sentido de comunicação com usuários e aplicações Front-end, quanto internamente, entre os serviços.

- **Engenharia e Processos de Software**: Utilizados para uma melhor compreensão sobre práticas e metodologias utilizadas no desenvolvimento de um software.

- **Programação Web**: Utilizado para uma melhor entendimento do sistema trabalhado. Além de fornecer conceitos fundamentais para o entendimento de servidores web e toda a arquitetura de sistemas web.

Apesar do curso ter auxiliado na aquisição de muitos conhecimentos, ele também poderia oferecer mais conhecimentos ligados à experiência profissional de desenvolvimento de software.

Em disciplinas que abrangem conceitos como metodologias de desenvolvimento de software e padrões de projetos, apesar de possuírem uma boa base teórica, elas não refletem exatamente os conhecimentos e forma de se trabalhar no desenvolvimento profissional. Atualmente, no mercado, vários padrões de projetos ou metodologias de desenvolvimento são bem diferentes das ensinadas durante o curso. As próprias metodologias ágeis, presentes neste trabalho, só foram abordadas brevemente durante a disciplina eletiva Processos de Software.

Em disciplinas ligadas ao desenvolvimento web, não são aprendidos solidamente questões muito requisitadas na área. Conceitos como, por exemplo, microsserviços e serviços de computação em nuvem. Além de que, muitas vezes, o desenvolvimento dos trabalhos e da disciplina não é pautado em tecnologias modernas.

Em função do contexto e do período curto do estágio, possíveis melhorias relacionadas à arquitetura do serviço não foram parte do requisito de melhoria do serviço. Dessa forma, uma melhor arquitetura para o serviço pode ser estudada e desenvolvida. Por exemplo, a realização da separação de cada módulo do serviço em diversos microsserviços que se comunicam entre si, poderia ser o foco de trabalhos futuros.

Assim, conclui-se que o período de estágio foi de extrema importância para a aplicação prática de conhecimentos teóricos aprendidos durante o curso. Também ofereceu uma melhor compreensão e expansão do conhecimento pelo estagiário, onde se aprofundou e aprimorou um serviço utilizado por usuários reais, em âmbito profissional.

REFERÊNCIAS

APPLE. **About HTTP Live Streaming**. 2014. Disponível em: <https://developer.apple.com/library/archive/referencelibrary/GettingStarted/AboutHTTPLiveStreaming/about/about.html>.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://agilemanifesto.org/>. Acesso em: 07 set. 2021.

CANALTI. **Arquitetura cliente-servidor**. 2018. Disponível em: <https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor/#:~:text=Definiç~ao,dosdados(osclientes).> Acesso em: 11 set. 2021.

CHISHTI, M. A. 2019. Disponível em: <https://docs.fileformat.com/video/flv/>.

CLAUDIO, Y. **O que são Codecs, quais são os tipos, pra que servem? Saiba mais sobre esse tema**. 2014. Disponível em: <https://canaltech.com.br/software/o-que-sao-codecs-quais-sao-os-tipos-pra-que-servem-saiba-mais-sobre-esse-tema/>.

CROSSHOST. **Transcodificação de vídeo, o que é isso? Streaming na prática**. 2021. Disponível em: <https://www.crosshost.com.br/streaming/transcodificacao-de-video-o-que-e/>.

EMILY, K. **What is RTMP? Real Time Messaging Protocol - What you Need to Know**. 2021. Disponível em: <https://www.dacast.com/blog/rtmp-real-time-messaging-protocol/>.

FELIPE, J. **O que é bitrate e como isso influencia na qualidade dos vídeos?** 2020. Disponível em: <https://canaltech.com.br/internet/o-que-e-bitrate-e-como-isso-influencia-na-qualidade-dos-videos-162423/>.

FFMPEG. **About**. 2021. Disponível em: <https://www.ffmpeg.org/about.html>. Acesso em: 24 out. 2021.

GIT. **About**. 2021. Disponível em: <https://git-scm.com/about>. Acesso em: 11 set. 2021.

GOGONI, R. **O que é bitrate e como ele afeta qualidade do streaming?** 2020. Disponível em: <https://tecnoblog.net/331879/o-que-e-bitrate-e-como-ele-afeta-qualidade-do-streaming/>.

GOMES, D. **O QUE É ADAPTIVE BITRATE STREAMING E PORQUE É TÃO IMPORTANTE**. 2020. Disponível em: <https://sambatech.com/blog/insights/adaptive-bitrate-streaming/>.

HSM. **Front-end e Back-end: conheças as diferenças**. 2020. Disponível em: <https://hsmuniversity.com.br/blog/front-end-e-back-end-diferenca/#:~:text=Odesenvolvedordefront-end,devidamentecertocomoservidor.> Acesso em: 13 set. 2021.

JONES, W. **Adaptive Bitrate Streaming (ABR): Video Streaming Definition**. 2018. Disponível em: <https://www.haivision.com/resources/streaming-video-definitions/adaptive-bitrate-abr-streaming/>.

LAHN, M. **How to Use the NGINX RTMP Module to Setup a Streaming Server**. 2021. Disponível em: <https://www.servermania.com/kb/articles/nginx-rtmp/>.

LAURENT, M. **Resolução, o que é exatamente?** 2020. Disponível em: <http://www.engesat.com.br/resolucao-o-que-e/>.

LEONARD, M.; KURNIAWAN, M. **A beginner's guide to video resolution**. 2021. Disponível em: <https://www.adobe.com/creativecloud/video/discover/video-resolution.html>.

MOZILLA. **O que é um servidor web (web server)? - Aprendendo desenvolvimento web: MDN**. 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/What_is_a_web_server>. Acesso em: 13 set. 2021.

NGINX. **nginx**. 2021. Disponível em: <https://nginx.org/en/>.

PAULA, G. B. **Tudo sobre a Metodologia Scrum: o que é, como usar, exemplos**. 2016. Disponível em: <https://www.treasy.com.br/blog/scrum/>. Acesso em: 11 set. 2021.

REGIS, C. D. M. et al. Transcodificação de vídeo digital para receptores portáteis. **IV SEGeT – Simpósio de Excelência em Gestão e Tecnologia**, 2007. Disponível em: <https://www.researchgate.net/profile/Marcelo-Alencar/publication/228893367_Transcodificacao_de_video_digital_para_receptores_portateis/links/09e41510800b557f26000000/Transcodificacao-de-video-digital-para-receptores-portateis.pdf>. Acesso em: 24 out. 2021.

RIBEIRO, M. **Gitlab: Gerenciamento de Códigos e Projetos**. 2021. Disponível em: <https://spiritsec.zendesk.com/hc/pt-br/articles/360001865777-Gitlab-Gerenciamento-de-Códigos-e-Projetos>. Acesso em: 11 set. 2021.

SOARES, M. d. S. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**,

2004. Disponível em: <http://periodicosibepes.org.br/index.php/reinfo/article/view/146>. Acesso em: 07 set. 2021.

STOPA, G. R.; RACHID, C. L. Scrum: Metodologia Ágil como ferramenta de gerenciamento de projetos. **CES Revista**, v. 33, n. 1, p. 302–323, 2019. Disponível em: <https://seer.cesjf.br/index.php/cesRevista/article/view/2026>. Acesso em: 07 set. 2021.

UFRJ. **O Modelo Cliente Servidor**. 2016. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2016-1/16_1/p2p/modelo.html>. Acesso em: 11 set. 2021.