PROPOSTA DE UMA ARQUITETURA DISTRIBUÍDA PARA UMA PLATAFORMA DE ENSINO DIGITAL BASEADA EM VÍDEO ON DEMAND

Proposal of a distributed architecture for a digital teaching platform based on video on demand

Crysthian Zanote Armini
Graduando
Engenharia de Computação
czarmini@ucl.br

Felipe de Abreu Ferrarini
Graduando
Engenharia de Computação
felipeferrarini@ucl.br

Marlon Ferrari
Orientador
Faculdade UCL
marlonferrari@ucl.br

A lista completa com informações de autoria está no final do artigo

RESUMO

Objetivo: Implementar uma plataforma de ensino digital interativa em vídeos, escalável e seguindo os princípios de arquitetura distribuída.
Método: Análise de padrões de modelagem atuais segundo a bibliografia, definição de casos de uso, modelagem da arquitetura, implementação da solução proposta e, por final, uma análise das escolhas técnicas definidas.
Resultado: Uma implementação prática da arquitetura proposta, com aplicação em nuvem pública.
Conclusões: Os resultados permitiram explorar possibilidades de serviços e práticas modernas para a construção de aplicações web distribuídas. A definição de arquitetura que contemple os principais casos de uso de uma plataforma de ensino digital baseada em *vídeo on demand* foi desenvolvida de forma prática e se mostrou segura e escalável permitindo com isso que universidades, escolas e empresas com esse foco possam se basear nesse modelo e evoluí-lo de acordo com suas regras de negócio específicas.

PALAVRAS-CHAVE: Arquitetura Distribuída, Engenharia de Software, Ensino Digital.

ABSTRACT

Objective: Implement an interactive video teaching platform, scalable and following the principles of distributed architecture.
Method: Analysis of current modeling patterns according to the bibliography, definition of use cases, architectural modeling, implementation of the proposed solution and, finally, an analysis of the defined technical choices.
Results: A practical implementation of the proposed architecture, with public cloud application.
Conclusions: The results made it possible to explore possibilities of services and modern practices for building distributed web applications. The definition of architecture that includes the main use cases of a digital model platform based on video on demand was developed in a practical way and proved to be safe and scalable, thus allowing universities, schools and companies with this focus to be based on this and specific evolution according to your business rules.

KEYWORDS: Distributed Architecture, Software Engineering, E-learning.

1 INTRODUÇÃO

Desde o início da pandemia de COVID-19, a sociedade foi inserida em um contexto de restrição e fechamento de ambientes, entre eles os acadêmicos, e a demanda por cursos online aumentou de forma significativa, como mostram os dados do *Google Trends*, na Figura 1.

Figura 1. Dados sobre buscas por cursos online, em português e inglês, nos últimos 5 anos (GOOGLE)

Apesar do pico de demanda ocasionada pela pandemia ter se estabilizado posteriormente, o ensino digital, academicamente muito referenciado como *e-learning,* se mostrou como uma alternativa aos métodos de aprendizagem tradicionais, como defende MURPHY (2020), o *e-learning* aumenta o acesso à educação nas comunidades rurais e à indivíduos que não podem frequentar uma escola tradicional presencial em tempo integral devido a circunstâncias pessoais ou financeiras. E mesmo dentro das instituições tradicionais de ensino superior, as formas híbridas ou digitais podem ajudar a melhorar a qualidade do ensino presencial, realizando a entrega de conteúdo on-line e concentrando as sessões presenciais na aprendizagem ativa. CARO (2008) aponta que o *e-learning* oferece vantagens consideráveis para organizações e universidades, bem como para os alunos. Além disso, há um grande número de iniciativas para impulsionar esse modelo visando contribuir para um ensino mais acessível, dinâmico e individualizado.

A partir da identificação desse cenário de constante desenvolvimento, foi observada uma oportunidade acadêmica pouco explorada com relação a propostas de soluções orientadas à arquitetura distribuída para plataformas de ensino digital.

Baseado nas premissas de utilidade pública, comercial e de desenvolvimento acadêmico das tecnologias utilizadas, o presente artigo descreve uma proposta de arquitetura distribuída que contemple tecnicamente os principais casos de uso de uma plataforma genérica de ensino digital baseada em *vídeo on-demand*. Apresenta como resultado a implementação prática da arquitetura proposta, utilizando um Serviço de computação em nuvem e diversas tecnologias de desenvolvimento web e com isso, gera conclusões sobre as escolhas técnicas que foram definidas.

2 REFERENCIAL TEÓRICO

2.1 Sistemas Distribuídos

Conforme COULOURIS (2013), um sistema distribuído é definido como sendo aquele no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas enviando mensagens entre si. Essa definição simples abrange toda a gama de sistemas nos quais computadores interligados em rede podem ser distribuídos de maneira útil.

Esse conceito adotado tem requisitos importantes a serem atendidos. Por exemplo, em uma rede de computadores a execução concorrente de programas é a norma que determina a capacidade de um sistema manipular recursos compartilhados na rede. Outra consequência é a inexistência de relógio global, pois existem limites de precisão na qual computadores podem ser sincronizados em uma rede e por isso a comunicação se dá por meio de mensagens. Uma última consequência citada, são as falhas independentes, pois a falha de um computador ou término inesperado de um programa em algum lugar do sistema não é imediatamente percebido pelos outros componentes com os quais ele se comunica. Ou seja, cada componente do sistema pode falhar independentemente, deixando os outros ainda em funcionamento (COULOURIS, 2013, p. 2).

2.2 Computação em Nuvem

A computação em nuvem vem sendo ressignificada na última década de acordo com os novos avanços das tecnologias que englobam o termo, mas segundo TAURION (2009), é possível descrever a Computação em Nuvem como um ambiente baseado em uma imensa rede de servidores físicos ou virtuais. Ou seja, é um conjunto de recursos com capacidade de processamento, de armazenamento, de conectividade, de plataformas, de aplicações e de serviços disponibilizados na internet.

TAURION (2009) define algumas características intrínsecas ao conceito de computação em nuvem. Dentre elas, a disponibilidade de recursos acessíveis sob demanda, a eliminação da necessidade de adquirir ou provisionar recursos antecipadamente, a elasticidade, permitindo que sejam usados recursos na quantidade que forem necessários, o que aumenta e diminui a capacidade computacional de forma dinâmica e o pagamento dos serviços em nuvem pela quantidade de recursos utilizados (*pay-per-use*).

2.3 Sistemas Web

A Internet introduziu novos hábitos de entretenimento, comportamento, comunicação e consumo no cotidiano das pessoas (MILETTO, 2014, p. 9). O desenvolvimento de sistemas web precisou acompanhar essa mudança de hábitos dos usuários e utilizar métodos, tecnologias e ferramentas específicas para isto.

Como ainda descrito por MILETTO (2014), o desenvolvimento de um sistema web envolve analisar, compreender, projetar e implementar. Quando um software é desenvolvido para a web, vários aspectos são combinados para que ele possa ser acessado de forma segura e remota por meio de um navegador.

2.4 Video On Demand (VOD)

Segundo ZHANG (2010), um sistema de *streaming* baseado em VOD consiste em servidores de vídeo, clientes e a rede que conecta os servidores aos clientes. Conforme explica, quando um cliente seleciona um vídeo, geralmente navegando em um site ou aplicativo, ele envia uma solicitação para um servidor. O servidor de vídeo armazena os arquivos em seu disco rígido, lê o arquivo na memória e entrega o vídeo solicitado ao cliente através da rede.

O servidor pode entregar a uma taxa diferente da taxa de reprodução do vídeo, desde que os pacotes cheguem antes do tempo de reprodução. Cada cliente reproduz o vídeo em seu próprio ritmo e pode realizar operações (por exemplo, buscar novas posições, alterar qualidade) durante a reprodução. Nesse sentido, cada cliente solicitante receberá sua própria réplica do vídeo.

2.5 E-Learning

O *e-learning* é definido por LAURILLARD (2006) como sendo o uso de qualquer tecnologia a serviço do aprendizado ou suporte ao aluno, podendo fazer uma diferença significativa na forma como eles aprendem, na rapidez com que dominam uma habilidade, na facilidade de estudo e no quanto eles gostam de aprender um assunto.

Hoje, o conceito de *e-learning* é o núcleo de vários negócios, sendo um serviço oferecido pela maioria das faculdades e universidades. Pode-se tomar como exemplo, plataformas como Udemy[1] e Alura [2] como negócios diretamente relacionados ao tema, assim como EDx [3]e Coursera [4]como sendo ferramentas oferecidas por grandes universidades.

Em uma perspectiva mais detalhada, o case da Udemy, de acordo com CETINA (2018), é uma das plataformas de aprendizado online mais importante do mundo com mais de 20 milhões de alunos, 65 mil cursos online, 14 categorias e 142 tópicos. Em sua evolução como negócio, a Udemy encontrou condições favoráveis devido ao desenvolvimento e acessibilidade à internet ocorrida nas últimas duas décadas.

Nesse cenário, o que chama atenção é uma tendência relacionada à natureza da nova geração, esta que é formada pelos novos usuários da internet. Os chamados "nativos digitais" ou, às vezes, "geração Z", abordam o trabalho, o aprendizado e o lazer de novas maneiras. DOWNES (2005), defende que eles absorvem informações rapidamente, em imagens e vídeos, bem como em texto, de várias fontes simultaneamente, esperam respostas e feedback instantâneos, preferem acesso aleatório "sob demanda" à mídia.

Na aprendizagem, essas tendências se manifestam no que às vezes é chamado de aprendizagem "centrado no aluno". Isso é mais do que apenas adaptar para diferentes estilos de aprendizagem ou permitir que o usuário altere o tamanho da fonte e a cor de fundo de uma tela, é a colocação do próprio controle da aprendizagem nas mãos do aprendiz (DOWNES, 2005, p. 1).

[1] Udemy: https://www.udemy.com/
[2] Alura: https://www.alura.com.br/
[3] EDx: https://www.edx.org/
[4] Coursera: https://www.coursera.org/

3 METODOLOGIA

A metodologia adotada neste trabalho baseou-se em pesquisas bibliográficas para justificar as definições de arquitetura e por fim, o desenvolvimento de um sistema real para avaliar os resultados da arquitetura proposta.

3.1 Amazon Web Services (AWS)

Para a implantação da arquitetura, foi definido como serviço de nuvem a *Amazon Web Services* que, como descreve VARIA (2014), é uma plataforma abrangente de serviços em nuvem que oferece poder de computação, armazenamento, entrega de conteúdo e outras funcionalidades que as organizações podem usar para implantar aplicativos e serviços de maneira econômica, com flexibilidade, escalabilidade e confiabilidade.

Seguindo essa abordagem, foram utilizados os serviços indicados no Quadro 1, disponíveis na AWS, para modelagem da proposta e posteriormente o desenvolvimento da aplicação.

Quadro 1 – Definição dos serviços da AWS utilizados.

Lambda Functions	É um serviço de computação sem servidor (*serverless*) e orientado a eventos que permite executar códigos para praticamente qualquer tipo de aplicação ou serviço *backend* sem provisionar e gerenciar servidores.
API Gateway	É um serviço que permite que desenvolvedores criem, publiquem, mantenham, monitorem e protejem APIs em qualquer escala com facilidade.
DynamoDB	É um banco de dados de chave-valor NoSQL, sem servidor e totalmente gerenciado, projetado para executar aplicações de alta performance em qualquer escala.
Simple Storage Service (S3)	É um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance para qualquer quantidade de dados.
CloudFront	É um serviço de rede de entrega de conteúdo (CDN) criado para alta performance, segurança e conveniência do desenvolvedor. Pode ser usado para entregar todo o site, incluindo conteúdo dinâmico, estático, de *streaming* e interativo.
CloudWatch	É um serviço de monitoramento que pode ser usado para coletar e rastrear métricas, coletar e monitorar arquivos de *log* e definir alarmes.
Cognito	Serviço que facilita o salvamento de dados do usuário na AWS sem escrever nenhum código de *backend* ou gerenciar qualquer infraestrutura.
Amplify	É um conjunto de recursos que permite criar aplicações completas de forma rápida e fácil na AWS. Configurar o *backend* de uma aplicação e conectá-la em minutos, criar o *frontend* e gerenciar facilmente o conteúdo da aplicação.
Simple Queue Service (SQS)	É um serviço de filas de mensagens gerenciado que permite o desacoplamento e a escalabilidade de sistemas distribuídos.

Fonte: Produtos da nuvem AWS (2022).

3.2 Arquitetura Definida

3.2.1 Casos de Uso

Foram definidos dois casos de uso, limitando as regras de negócio e pensando no uso geral de uma aplicação genérica para ensino online, que são os descritos no Quadro 2 e Quadro 3:

Quadro 2 – [UC-01] Cadastrar um curso

Ator:	Professor.
Descrição:	A plataforma deverá permitir o cadastro de um **Curso** feito por um **Professor**, contendo conteúdo de vídeo ou exercícios.

Fonte: Do Autor (2022).

Quadro 3 – [UC-02] Realizar o curso

Ator:	Estudante.
Descrição:	Um **estudante** deve poder listar **cursos** disponíveis, se inscrever, assistir as aulas e realizar as atividades.

Fonte: Do Autor (2022).

3.2.2 Diagrama de Caso de Uso

Na Figura 2 se encontra o diagrama de caso de uso onde foi definido um processo ponta a ponta que ajuda a identificar ações e atores em um contexto de aplicação prática da proposta.

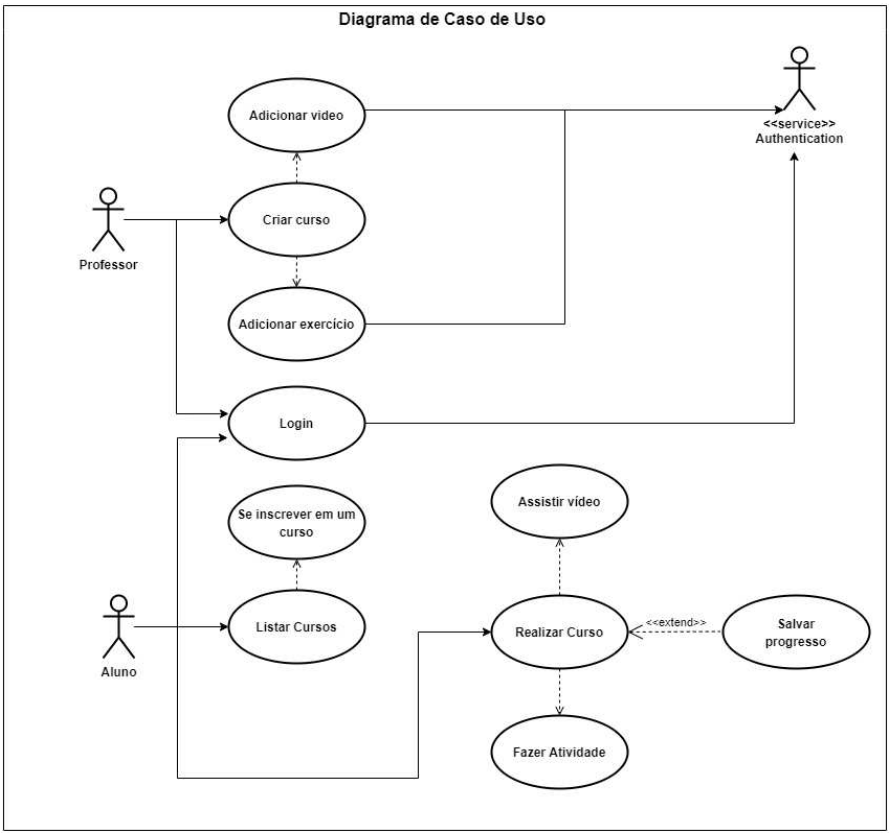

Figura 2. Diagrama de casos de uso gerais (Do autor)

3.2.3 *Modelagem de Arquitetura Geral*

Com base nas definições dos serviços da AWS que foram referenciados nos tópicos acima, foi definido o modelo geral para uma arquitetura distribuída que atende os requisitos gerais de casos de uso para uma plataforma de cursos baseada em *vídeo on demand.* A modelagem da arquitetura se encontra abaixo, na Figura 3.

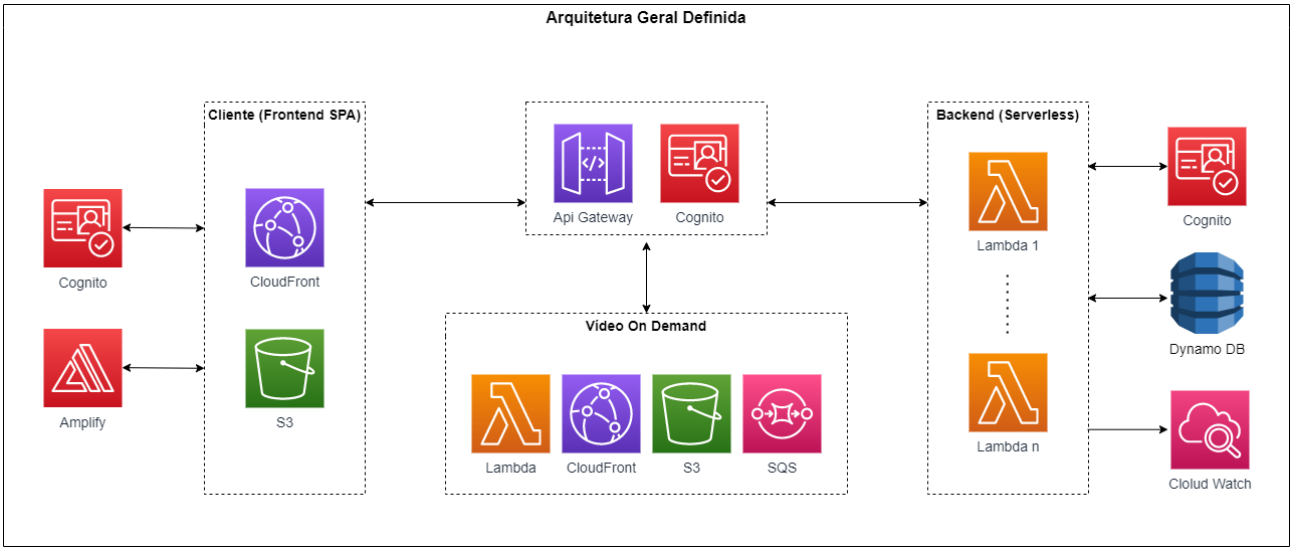

Figura 3. Modelagem de arquitetura geral da proposta definida (Do autor)

A Figura 3 mostra um design ponta a ponta, e referencia todos os serviços usados para o armazenamento de *frontend*, a autenticação, os serviços *backend*, o monitoramento, o armazenamento e a fila. Os tópicos abaixo, contém cada ponto desse modelo detalhado.

3.2.4 *Vídeo On Demand*

O principal desafio técnico de uma aplicação que tem como foco a distribuição de conteúdo de mídia próprio é a parte de VOD, e na AWS existem serviços que fazem isso. Entretanto, com os serviços de *lambda functions*, S3 e SQS é possível criar uma estrutura semelhante, com a diferença de que é possível gerenciar de forma direta, e com menor custo, a parte principal do serviço. Com isso, é possível utilizar bibliotecas de linguagens de programação para realizar os tratamentos necessários, processamento e armazenamento dos vídeos.

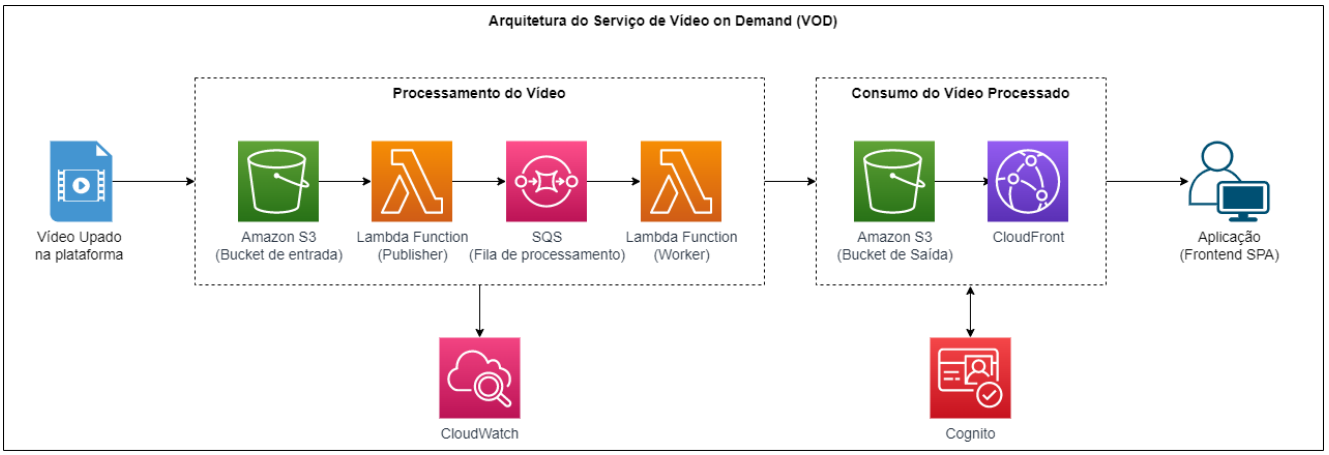

Figura 4. Arquitetura do serviço de *Vídeo On Demand* (Do autor)

Conforme mostra a Figura 4, sempre que um vídeo é enviado para um *bucket* de entrada no S3 é executada uma *lambda function* que foi definida como *publisher*, responsável por coletar as informações do arquivo e enviar para a fila de processamento (SQS) juntamente com as resoluções em que se deseja que o vídeo seja convertido. Para cada resolução será executada uma nova *lambda function* definida como *worker,* que realiza a conversão do vídeo para a resolução especificada. Após o vídeo ser convertido, ele é enviado para um *bucket* de saída. Esse *bucket* pode ser acessado de forma direta pelo *CloudFront*.

3.2.5 Backend Serverless

Foi definida uma estrutura *serverless* para os serviços de *backend* levando em conta as vantagens do uso de *Lambdas* na AWS, como sua capacidade de escalar de acordo com a demanda e o processamento em paralelo das funções. Ou seja, cada requisição é processada individualmente, além das *Lambdas* serem responsáveis por implementar e gerenciar toda a infraestrutura exigida para executar o código. Isso gera confiabilidade entre os serviços e casos de uso dentro da plataforma.

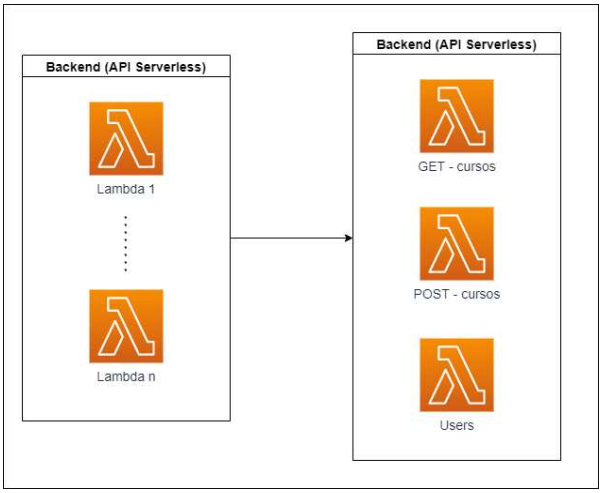

Figura 5. Exemplo de distribuição de *Lambdas* de uma aplicação (Do autor)

A Figura 5 exemplifica a distribuição de serviços dentro de uma aplicação e nesse caso separa as responsabilidades relacionadas a cursos em dois serviços diferentes, *get* e *post*, e um terceiro serviço referente ao gerenciamento de usuários. Neste exemplo, caso uma delas apresente problema ou instabilidade as outras não seriam afetadas devido ao isolamento de responsabilidades em cada *lambda*.

Ao se considerar que é uma proposta de plataforma web, existe um outro serviço da AWS que se mostrou fundamental dentro desse escopo, que é o *API Gateway*. Com ele é possível rotear um *endpoint* para cada *lambda* criada, além de manter e monitorar as APIs geradas. Ou seja, o *API Gateway* é a "porta de entrada" para aplicativos acessarem os dados, lógica de negócios ou funcionalidade de seus serviços de *backend* definidos através das *lambdas*.

3.2.6 Frontend

Foi definida uma estrutura utilizando os serviços S3, *CloudFront* e *Amplify*. Todos os *scripts* e arquivos estáticos ficam armazenados no S3, que serve esses recursos para que o *CloudFront* os utilize na construção de recursos web do site. Isso tudo é configurado e gerenciado com o uso do *Amplify*. Com essa definição, é possível gerar redundância no *frontend* da aplicação, seguindo os princípios de escalabilidade, alta disponibilidade e pouca dependência dos recursos de máquinas, uma vez que não necessita de um servidor local rodando a todo momento.

3.2.7 Banco de Dados

O *DynamoDB* foi escolhido baseado nas premissas de utilização dele dentro da AWS, ou seja, sua facilidade de trabalhar com *Lambdas* e por ser projetado para executar aplicações de alta performance em qualquer escala.

3.2.8 Monitoramento do Sistema

Aplicações modernas geram grandes volumes de dados na forma de métricas, *logs* de informações, erros e eventos. Para gerenciar todo esse fluxo de monitoramento, foi definido o *CloudWatch* como serviço, pois ele coleta, acessa e correlaciona os dados e permite explorar, analisar e visualizar *logs* para solucionar problemas operacionais de uma aplicação com uma maior facilidade.

3.2.9 Armazenamento

Uma solução distribuída como essa necessita de armazenamento em diferentes partes da arquitetura. Entendendo isso, foi definido o serviço S3 da AWS como o padrão para armazenamento de dados. Observe na Figura 3 que o S3 é representado em quase todo o processo.

Inicialmente, foi definido o seu uso para o armazenamento de arquivos para a aplicação web, como arquivos CSS e JavaScript do *frontend*. Além disso, o S3 é utilizado para armazenar os vídeos que serão inseridos na plataforma, assim como fazer parte da etapa de processamento desses vídeos tanto no *upload* quanto no *streaming* do conteúdo.

3.2.10 Autenticação

A autenticação dentro de uma proposta genérica deve permitir adicionar o cadastramento, o login e o controle de acesso de aplicações com rapidez e facilidade, e esses são os pontos que fizeram com que o C*ognito* fosse o serviço de autenticação selecionado. Além de poder ser escalado dinamicamente, ele conta com uma gama de opções de autenticação que atendem os mais diversos casos de uso, permitindo gastar mais tempo no desenvolvimento da ideia, enquanto o *Cognito* cuida e facilita toda complexidade técnica de autenticação.

Observe a Figura 3, foi proposto o uso do *Cognito* na parte de permissões do *frontend*, no momento de acesso aos serviços de VOD, até nas permissões de acesso a rotas gerenciadas pelo *API Gateway.* Sendo assim, o *Cognito* tem grande relevância para a segurança geral do modelo proposto.

3.3 Ferramentas de Desenvolvimento

Para desenvolver um sistema seguindo o modelo proposto é necessário utilizar ferramentas específicas, como IDE, containers e controlador de versão para rodar localmente a aplicação em fase de testes.

3.3.1 Visual Studio Code

O Visual Studio Code é um editor de código de alta produtividade que, quando combinado com serviços de linguagem de programação, oferece o poder de IDE e a velocidade de um editor de texto. É um editor de código-fonte leve, mas poderoso, que é executado em sua área de trabalho e está disponível para Windows, macOS e Linux. (Visual Studio Code Docs, 2022).

3.3.2 Docker

BOETTIGER (2015), define *docker* como uma plataforma de código aberto que executa aplicativos e facilita o processo de desenvolvimento e distribuição. Os aplicativos criados são empacotados com todas as dependências em um formulário padrão chamado contêiner. Esses containers continuam rodando de forma isolada em cima do *kernel* do sistema operacional. A tecnologia de containers já existe há mais de 10 anos, mas *docker*, em geral se destaca atualmente entre as melhores inovações, pois acompanha novas capacidades que as tecnologias anteriores não possuíam.

3.3.3 GIT

Como explica VUORRE (2016), o Git, ao contrário do visualizador de arquivos padrão do sistema operacional, não é um programa de apontar e clicar para navegar em arquivos e pastas de um computador. Em vez disso, adiciona funcionalidade ao sistema de arquivos existente, disponibiliza um conjunto específico de comandos que permite acompanhar os arquivos, seu histórico e, com isso, distribui os arquivos entre vários computadores e usuários.

4 DESENVOLVIMENTO DE UM SISTEMA BASEADO NA ARQUITETURA DEFINIDA

Foi desenvolvido uma aplicação web com regras de negócio simples, mas que demonstram o uso da arquitetura definida de forma clara.

Nessa aplicação, qualquer pessoa pode realizar o seu cadastro através da página inicial do site e ao se cadastrar é possível escolher se deseja ser aluno ou professor na plataforma. Caso escolha ser aluno, você poderá visualizar os cursos disponíveis e fazer a matrícula em cada um deles e em seguida assistir as aulas e realizar o seu progresso. Caso escolha ser professor, terá a possibilidade de criar cursos e formatá-los como desejar, ao adicionar na plataforma várias atividades que podem ser vídeos ou artigos, e ao final o curso, é publicado para que os alunos possam realizar matrículas.

Nas sessões abaixo, é possível entender como as definições de arquitetura foram utilizadas no desenvolvimento técnico dessa aplicação.

4.1 CI /CD

Continuous Integration/Continuous Delivery ou integração e entrega contínuas, tem como principal objetivo, como definido por HILTON (2017), melhorar a qualidade do software, reduzir os riscos e reduzir o esforço humano automatizando tarefas repetitivas.

4.1.1 GitHub Actions

A documentação oficial da tecnologia define o *GitHub Actions* como uma plataforma de integração contínua e entrega contínua (CI/CD) que permite automatizar seu pipeline de compilação, teste e implantação. Com ele foi possível criar fluxos de trabalho que constroem e testam cada *pull request* para seu repositório ou implementam *pull requests* mesclados para produção.

4.1.2 Serverless Framework

É uma ferramenta fácil e acessível para implantar o código e a infraestrutura de nuvem necessária para criar qualquer caso de uso de aplicações *serverless*. É uma estrutura multilíngue compatível com Node.js, Typescript, Python, Go, Java e muitas outras linguagens.

4.2 Serviço de Vídeo On Demand

Para criar o serviço de *vídeo on demand* conforme a Figura 4, foi desenvolvido inicialmente a *lambda publisher*, que é acionada quando ocorre um evento de upload no *bucket* de entrada no S3, e com isso, ela obtém informações do vídeo que acabou de ser inserido e envia para a fila do SQS três requisições, cada uma com qualidade de saída diferente, que nessa aplicação desenvolvida foram definidas como sendo 360p, 720p e 1080p. Após isso, o SQS processa cada chamada e dispara para a *lambda worker* realizar o processamento e conversão do vídeo, e para essa aplicação foi utilizada uma biblioteca chamada FFmpeg. Finalizando a conversão, a *lambda worker* envia o vídeo final para o *bucket* de saída, onde o caminho é criado dentro dele por meio de um identificador da atividade referente ao vídeo em questão, concluindo, assim, a parte de *upload*.

No *frontend*, é possível acessar de forma direta, com as devidas configurações de permissão e autenticação, o caminho desse *bucket* de saída portanto, o consumo desse vídeo, na qualidade desejada é realizada através do *frontend*. Para essa aplicação, não foi desenvolvido a interpretação dinâmica da qualidade do vídeo a ser entregue para o usuário de acordo com a sua rede sendo assim, o usuário deve selecionar manualmente a qualidade do vídeo caso deseje trocar.

4.3 Desenvolvimento do *Backend Serverless*

Utilizou-se o NodeJS para desenvolver o *backend*, que como definido por HELLER (2017), é um ambiente de execução JavaScript enxuto, rápido e multiplataforma que é útil para servidores web e aplicativos *desktop*. Atinge baixa latência e alta taxa de transferência, e adota uma abordagem "*non-blocking*" para atender solicitações. Em outras palavras, o Node.js não perde tempo ou recursos esperando o retorno das solicitações de I/O.

Com isso, foram construídas todas as funções necessárias para sustentar a aplicação, que são descritas no Quadro 2.

Quadro 2 – Funções desenvolvidas para o *backend serverless*

Lambda:	Funcionalidade:
Atividades	Função responsável por buscar e/ou criar as atividades de um determinado curso.
Cursos	Função responsável pela criação, modificação e exclusão de um curso, além de retornar uma lista de cursos ou detalhes de um em específico.
Matrículas	Função responsável por realizar a matrícula de um aluno em um curso e também retornar a lista de cursos que um aluno está matriculado
Usuários	Função responsável por criar um usuário e buscar informações através do ID dele no AWS *Cognito*

Fonte: Do autor.

Todas as funções criadas são roteadas e gerenciadas pelo *API Gateway* e se comunicam com um banco de dados, como demonstrado na Figura 6

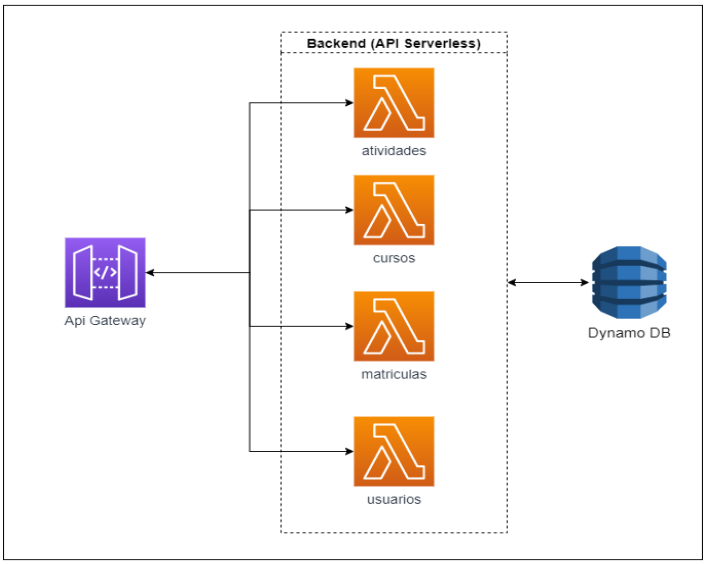

Figura 6. Arquitetura distribuída do *backend serverless* (Do autor)

4.4 Desenvolvimento do *Frontend*

Na camada cliente da aplicação foi utilizado o ReactJS, que é definido por CASPERS (2017) como uma biblioteca Javascript para interface de usuário. Seus principais objetivos são renderizar a interface do usuário e responder a eventos. Também foi utilizado o conceito de *Single Page Application* (SPA) que de acordo com MIKOWSKI (2013), é uma aplicação entregue ao navegador que não recarrega a página durante o uso. Ou seja, ao invés de recarregar a mesma página ou redirecionar o usuário para uma outra página, apenas o conteúdo principal é atualizado de forma assíncrona, mantendo com isso a estrutura padrão da página estática.

4.5 Monitoramento do Sistema

Foi utilizado o *CloudWatch*, como definido, para realizar o monitoramento de logs e erros de toda a aplicação, desde uma conversão de vídeo pelo serviço de VOD até a chamada de uma *lambda* específica pelo *frontend*, tudo isso foi centralizado e gerenciado pelo *CloudWatch*.

4.6 Armazenamento

Em vários momentos do processo de desenvolvimento da aplicação, já mencionados, foram utilizados recursos de armazenamento do S3, seguindo o que foi proposto.

No *frontend,* o S3 foi necessário para armazenar os arquivos estáticos a serem entregues ao *CloudFront*. Já no serviço de VOD, foi utilizado como ponto de partida para o processamento e na alocação final dos vídeos processados e devidamente identificados.

4.7 Autenticação

Toda a autenticação foi realizada pelo *Cognito*, gerenciado diretamente via *frontend* e pelas rotas do *API Gateway*.

O cadastro e login também foram realizados utilizando as abstrações do *Cognito*. Seguindo a regra de negócio proposta, o usuário tem a possibilidade de realizar o cadastro ao definir e-mail e senha ou, também, utilizando a autenticação do Google. Ao se cadastrar, ele escolhe qual tipo de usuário deseja ser, professor ou aluno, e com isso recebe a regra de usuário que define suas permissões de autenticação para determinadas rotas da aplicação.

Algumas rotas não foram autenticadas com restrição por usuários, como a da listagem de cursos. Entretanto, diversas outras são relacionadas ao tipo de usuário e permitem o acesso apenas com a devida autenticação realizada.

4.8 Resultados da Aplicação

As figuras abaixo mostram o resultado da aplicação desenvolvida, tanto na visão do aluno quanto na do professor, ao usar a plataforma.

Figura 7. Página inicial e acesso da plataforma (Do autor)

A Figura 7 mostra o resultado da página inicial e o redirecionamento para realizar o acesso ou cadastro no site, utilizando o *Cognito*.

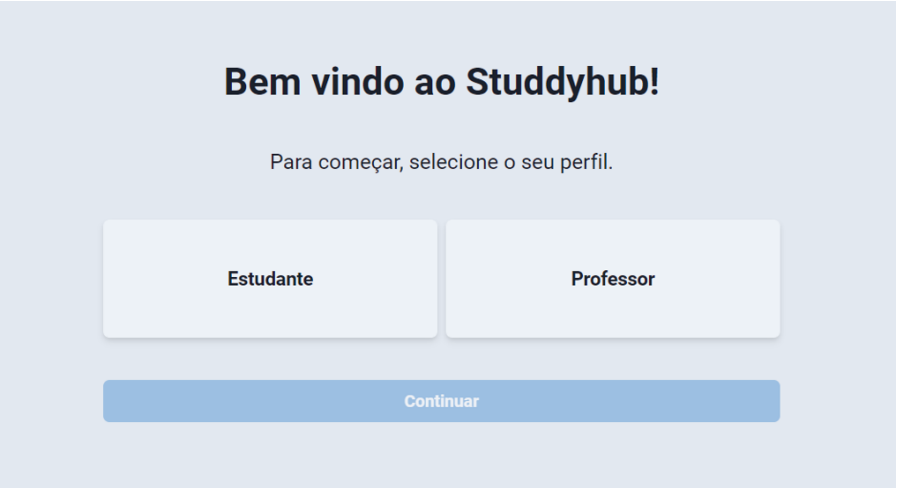

Figura 8. *Onboarding* na plataforma (Do autor)

Após realizar o cadastro o usuário é redirecionado para uma página onde ele define se deseja ser professor ou aluno dentro da plataforma, como mostra a figura Figura 8.

4.8.1 Usuário Professor

O professor, ao realizar seu cadastro, tem a opção de criar um novo curso, visualizar informações de cursos já existentes assim como, excluí-los da plataforma. Além disso, ele consegue buscar por cursos disponíveis na plataforma.

Nas figuras abaixo é possível visualizar as funcionalidades finais desenvolvidas para que um professor utilize a plataforma.

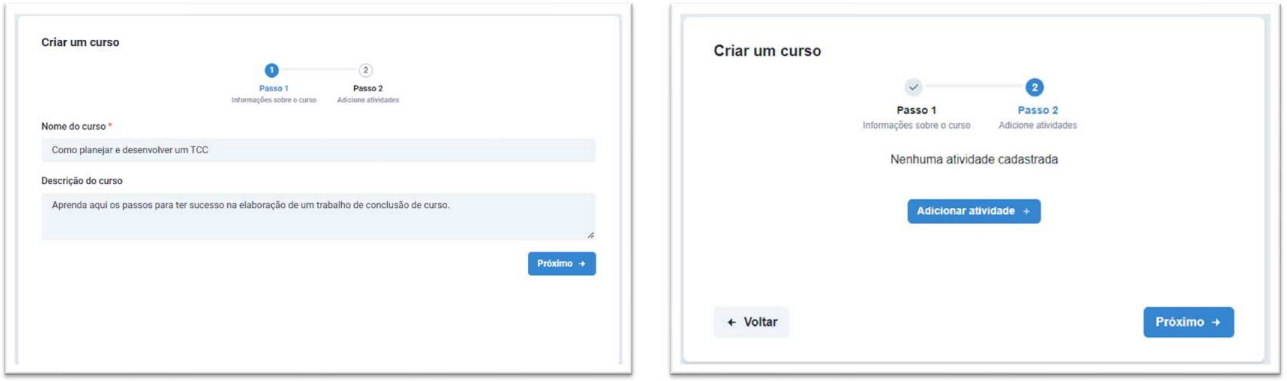

Figura 9. Criação de um novo curso (Do autor)

Durante a criação foram definidos dois passos. O primeiro para informar nome e descrição do curso, mostrado na Figura 9, e o segundo para adicionar atividades ao curso, informando o nome, descrição e tipo da atividade, como mostra a Figura 10, onde foi escolhido o tipo vídeo e realizado o *upload.*

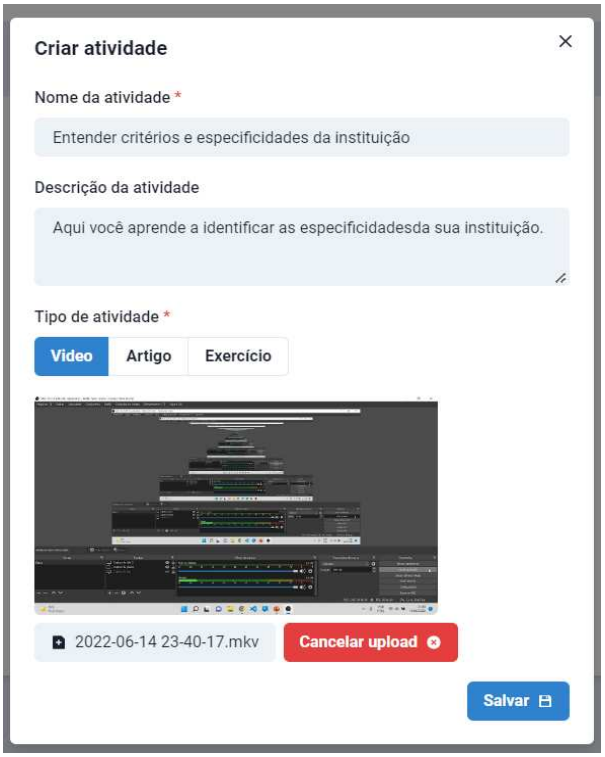

Figura 10. Criação de uma atividade para um curso (Do autor)

Figura 11. Visualização de cursos do professor (Do autor)

O professor tem a opção de visualizar e gerenciar os seus cursos cadastrados, como mostra a Figura 11. Sendo possível publicar, visualizar detalhes, editar e excluir um curso.

4.8.2 Usuário Aluno

Figura 12. Busca por cursos na plataforma (Do autor)

Como aluno, após realizar o acesso, é possível buscar por um curso, conforme a Figura 12, e acessá-lo e realizar a sua matrícula, como mostra a Figura 13.

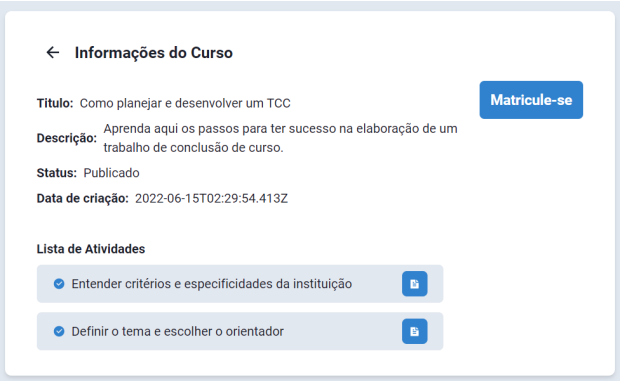

Figura 13. Detalhes e opção de matrícula em um curso (Do autor)

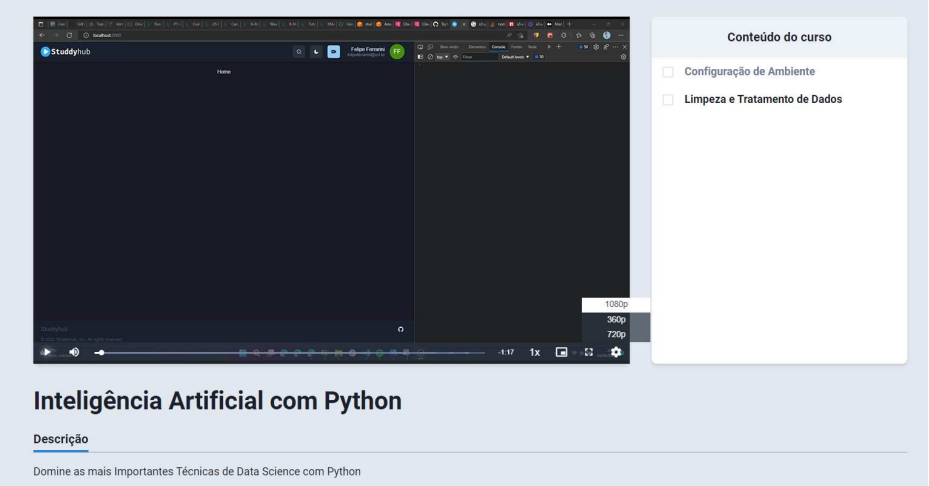

Figura 14. *Player* de reprodução de vídeos e atividades (Do autor)

Após realizar a matrícula, o aluno pode iniciar o curso. A Figura 14 mostra o *player* de execução dos vídeos, com o detalhe para a alternativa de diferentes qualidades de reprodução, além das opções de diferentes velocidades, tela cheia e tela minimizada.

Ao lado do *player,* é possível identificar as atividades referentes ao curso sendo que essas possuem um *checkbox* com a opção de ser preenchido para identificar que o vídeo em questão já foi visualizado, como mostra a Figura 15.

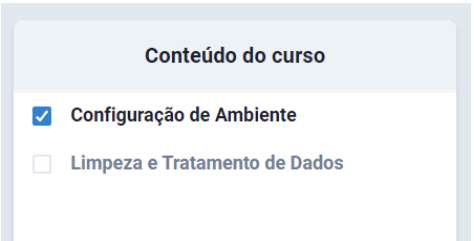

Figura 15. Progresso de execução de atividades (Do autor)

5 CONCLUSÃO

O desenvolvimento do presente artigo possibilitou explorar potenciais de qualidade e performance de serviços e práticas modernas para a construção de aplicações web distribuídas em nuvem. A definição de arquitetura, que contempla os principais casos de uso de uma plataforma de ensino online baseada em *vídeo on demand,* se mostrou segura e aplicável com o desenvolvimento de uma aplicação prática baseada inteiramente nela.

A proposta definida conseguiu cumprir com os conceitos de sistemas distribuídos referenciados, como falhas independentes e execução concorrente de funcionalidades. No decorrer do desenvolvimento da aplicação prática, foi observado, diversas vezes, falhas independentes entre as funcionalidades que não influenciaram na execução geral da aplicação, ressaltando com isso a validação do modelo proposto.

Na perspectiva de mercado, existem vantagens nessa proposta que se tornam mais relevantes quando usadas em cenários como os de *startups* de ensino digital, conhecidas como *Edtechs*. Elas precisam responder de forma rápida às mudanças exigidas pelo mercado e de altas cargas de uso como na pandemia. Com o uso da arquitetura proposta nesse artigo, os requisitos mencionados são supridos.

É importante ressaltar que a solução proposta não é a única capaz de resolver as demandas técnicas de uma plataforma de ensino digital e, até mesmo por meio de outros serviços da AWS, é possível desenvolver diferentes soluções. Para trabalhos futuros, é possível explorar mais o potencial de busca elástica dentro desse tema e realizar uma análise comparativa entre a arquitetura distribuída proposta e uma arquitetura monolítica semelhante.

Entretanto, o propósito deste artigo foi elaborar uma solução geral sobre a possibilidade de se utilizar os conceitos de sistemas distribuídos em uma plataforma que possa ser escalável com pouco gerenciamento, permitindo com isso que universidades, escolas e empresas com esse foco possam se basear nesse modelo e evoluí-lo de acordo com suas regras de negócio específicas e contribuir com o desenvolvimento acadêmico do tema.

REFERÊNCIAS

CARO, Eva Martinez. **E-learning: uma análise do ponto de vista do aluno**. ITEN. Revista Ibero-Americana de Educação a Distância, v. 11, não. 2 P. 151-168, 2008.

CASPERS, Matthias Kevin. **React and redux.** Rich Internet Applications w/HTML and Javascript, 2017.

CETINA, Iuliana; GOLDBACH, Dumitru; MANEA, Natalia. **Udemy: a case study in online education and training.** Revista Economică, v. 70, 2018.

COULOURIS, George *et al*. **Sistemas Distribuídos: Conceitos e Projeto**. Porto Alegre: Bookman, 2013.

DOWNES, Stephen. **E-learning 2.0**. ELearn, v. 2005, n. 10, p. 1, 2005.

GOOGLE TRENDS, 2022. Disponível em: https://bit.ly/3EbHz4i. Acesso em: 12 abr. 2022.

HELLER, Martin. **What is Node. js?** The JavaScript runtime explained. InfoWorld, 2017.

HILTON, Michael et al. **Trade-offs in continuous integration:** assurance, security, and flexibility. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017.

LAURILLARD, Diana. **E-learning in higher education.** Changing higher education: The development of learning and teaching, v. 3, 2006.

MIKOWSKI, Michael; POWELL, Josh. **Single page web applications**: JavaScript end-to-end. Simon and Schuster, 2013.

MILETTO, Evandro *et al*. **Desenvolvimento de Software II.** Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014.

MURPHY, Michael PA. **COVID-19 and emergency eLearning: Consequences of the securitization of higher education for post-pandemic pedagogy**. Contemporary Security Policy, v. 41, n. 3, p. 492-505, 2020.

Produtos da nuvem AWS, 2022. Disponível em: https://aws.amazon.com/pt/products/. Acesso em: 21 mai. 2022.

TAURION, Cezar. **Cloud Computing.** Computação em Nuvem. Rio de Janeiro: Brasport, 2009.

VARIA, Jinesh et al. **Overview of amazon web services**. Amazon Web Services, v. 105, 2014.

Visual Studio Code Docs, 2022. Disponível em: https://code.visualstudio.com/docs. Acesso em: 21 mai. 2022.

VUORRE, Matti; CURLEY, James P. **Curating research assets in behavioral sciences**: A tutorial on the GIT version control system. PsyArXiv Preprints, 2017.

ZHANG, Xiangyang *et al*. **Video on-demand streaming on the Internet.** A survey. 25th Biennial Symposium on Communications, Canada, 2010.

AGRADECIMENTOS
Agradecemos às nossas famílias que estiveram conosco até aqui, ao nosso orientador pelo suporte e a todos que de alguma forma contribuíram com a nossa formação.

CONTRIBUIÇÃO E AUTORIA
Não se aplica

FINANCIAMENTO
Não se aplica.

CONSENTIMENTO DE USO DE IMAGEM
Não se aplica

CURSO
Engenharia de Computação

COORDENADOR DO CURSO
Dayane Corneau Broedel

DATA DE ENTREGA
20/06/2022

BANCA AVALIADORA
Zirlene Effgen
André Ribeiro da Silva

DECLARAÇÃO DE INEXISTÊNCIA DE PLÁGIO
Declaro que o trabalho não contém plágio ou autoplágio total ou parcial. Todo o conteúdo utilizado como citação direta ou indireta foi indicado e referenciado.