EDUARDO MIRANDA PEDROSA FILHO

ESTUDO E IMPLEMENTAÇÃO DE MELHORIAS EM UM SISTEMA DE STREAMING PARA VÍDEOS DE EDUCAÇÃO CRISTÃ: IMPLEMENTAÇÃO DA ESTRUTURA DE CURSOS NO SISTEMA BLESSS

LAVRAS – MG

2021

EDUARDO MIRANDA PEDROSA FILHO

ESTUDO E IMPLEMENTAÇÃO DE MELHORIAS EM UM SISTEMA DE STREAMING PARA VÍDEOS DE EDUCAÇÃO CRISTÃ:
IMPLEMENTAÇÃO DA ESTRUTURA DE CURSOS NO SISTEMA BLESSS

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. Eric Fernandes de Mello Araújo
Orientador

LAVRAS – MG

2021

EDUARDO MIRANDA PEDROSA FILHO

ESTUDO E IMPLEMENTAÇÃO DE MELHORIAS EM UM SISTEMA DE STREAMING PARA VÍDEOS DE EDUCAÇÃO CRISTÃ: IMPLEMENTAÇÃO DA ESTRUTURA DE CURSOS NO SISTEMA BLESSS

STUDY AND IMPLEMENTATION OF IMPROVEMENTS IN A STREAMING SYSTEM FOR CHRISTIAN EDUCATION VIDEOS: IMPLEMENTATION OF THE COURSE STRUCTURE IN THE BLESSS SYSTEM

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 9 de novembro de 2021.

Prof. Dr. Raphael Winckler de Bettio
Prof. Dr. Maurício Ronny de Almeida Souza

Prof. Dr. Eric Fernandes de Mello Araújo
Orientador

LAVRAS – MG
2021

À minha mãe, meu pai, minha irmã e toda minha família, que foram pacientes, acreditaram em mim e me apoiaram em todos os momentos; e sempre me deram a certeza de que na companhia deles eu tenho um porto seguro onde encontro os melhores abraços.

AGRADECIMENTOS

Ao Único e Soberano Deus, meu Castelo Forte, que todos os dias derramou sobre mim seu amparo e me deu a certeza de que sem Ele nada posso fazer.

Aos meus pais, à minha irmã e a toda a minha família, que apoiaram em todos os momentos, incentivaram os meus estudos, foram pacientes e onde eu sempre encontrei segurança e amor.

Ao meu supervisor e grande amigo que Deus me deu Thiago Alves Brum, que me deu a oportunidade de realizar o estágio na Zeester e me ofereceu todo suporte necessário nessa jornada.

Aos meus colegas de equipe na Zeester e amigos Fábio, Guilherme, João Pedro e Kaio, e à toda a equipe da Zeester, que me acompanharam no desenvolvimento do projeto.

Ao meu orientador Prof. Dr. Eric Fernandes de Mello Araújo, amigo e irmão na fé, que contribuiu muito para a escrita deste trabalho, sempre estando disponível para ajudar.

Aos meus irmãos em Cristo da Igreja Presbiteriana de Lagoa da Prata e da 1ª Igreja Presbiteriana de Lavras, em especial aos jovens, que estiveram comigo nos momentos difíceis e sempre em oração.

RESUMO

Este trabalho é um relatório de estágio realizado em uma empresa que trabalha no desenvolvimento e manutenção de um sistema para educação cristã. O objetivo do estágio foi desenvolver uma ferramenta de acesso e acompanhamento de vários cursos, com conteúdos oferecidos em formato de vídeo, apostila e questionário, permitindo a interação dos usuários com a plataforma. No desenvolvimento, foi utilizada a linguagem de programação JavaScript, a biblioteca React e outras bibliotecas para gerenciamento de estado global e o acesso de APIs. Como resultado, o sistema foi atualizado e disponibilizado para os usuários com as novas funcionalidades criadas. Ao longo do estágio, desafios foram enfrentados, dentre eles o aprendizado de novas tecnologias, implementação de boas práticas de programação e aplicação de conceitos vistos ao longo do curso de Ciência da Computação na rotina de desenvolvimento do sistema. Além disso foi possível obter experiência no trabalhar em equipe, por meio do uso de metodologias ágeis e da interação constante na discussão e tomada de decisões para o projeto com outros membros da equipe. O presente texto apresentará os avanços obtidos na construção do sistema Blesss durante o estágio.

Palavras-chave: Curso. Front-end. React.

ABSTRACT

This work is an internship report carried out in a company working on the development and maintenance of a system for Christian education. The objective of the internship was to develop a tool for accessing and monitoring various courses, with content offered in video, handout and questionnaire formats, allowing users to interact with the platform. During development, the JavaScript programming language, the React library and other libraries for global state management and API access were used. As a result, the system was updated and made available to users with the new features created. During the internship, challenges were faced, among them the learning of new technologies, implementation of good programming practices and application of concepts seen throughout the Computer Science course in the system development routine. In addition, it was possible to gain experience in working as a team, through the use of agile methodologies and constant interaction in the discussion and decision-making for the project with other team members. This text will present the advances obtained in the construction of the Blesss system during the internship.

Keywords: Course. Front-end. React.

LISTA DE FIGURAS

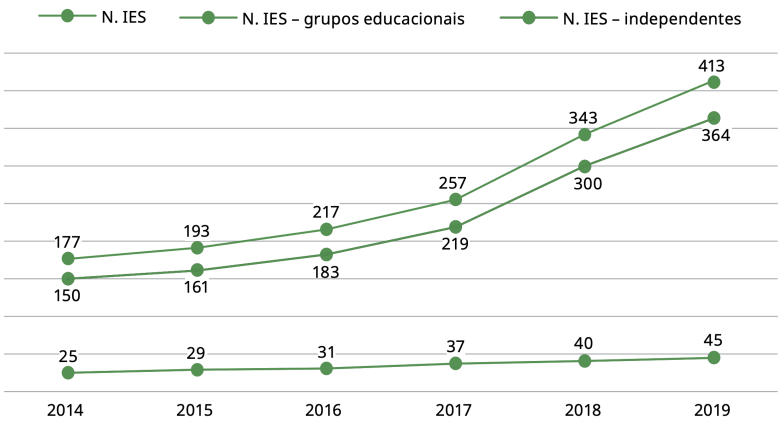
SUMÁRIO

1 INTRODUÇÃO

O uso de vídeos para educação na web vem crescendo nos últimos anos. Várias ferramentas vêm sendo desenvolvidas com funcionalidades diversas buscando antender à crescente demanda educacional por parte de empresas públicas e privadas, no intuito de permitir que seus clientes tenham acesso aos conteúdos produzidos quando e onde quiserem. O aumento na capacidade de banda de internet facilitou o processo de criação de cursos online, bem como a popularização da internet[1]. A pandemia da COVID-19 veio ainda acentuar essa demanda, tendo em conta as limitações físicas impostas para contenção do vírus.

Apesar da motivação inicial de muitas empresas na criação de ferramentas online de educação estar vinculada à necessidade causada pela pandemia, observa-se um crescimento na quantidade de cursos e IES credenciadas para oferta de cursos EAD (Figura 1.1) nos últimos anos pré-pandemia.

Figura 1.1 – IES credenciadas para oferta EAD de 2014 a 2019



Fonte: CensoEAD.BR - 2019/2020[2]

Instituições religiosas cristãs também vêm desempenhando um papel na educação desde os primórdios de seu surgimento, sendo o primeiro material edu-

[1] Em 2019, no Brasil, mais de 80% dos domicílios já contavam com acesso à internet, segundo Pesquisa Nacional por Amostra de Domicílios (PNAD) de 2019.

cacional criado conhecido como Didaquê, escritos datados do primeiro século, contendo os ensinamentos nas igrejas estruturadas ainda no primeiro século e direcionadas pelos apóstolos de Jesus Cristo (BERARDINO et al., 2002). Também é notório o cuidado com a educação dispensado por reformadores do século XVI, em especial Martinho Lutero, que formulou o sistema de ensino público que deu origem ao modelo de escola moderna do Ocidente, a partir da preocupação com o acesso da população aos textos bíblicos, anteriormente acessíveis apenas a pessoas da igreja ou com altos postos políticos e sociais (MANACORDA; NOSELLA; OLIVEIRA, 2002). Com o aumento de evangélicos no Brasil[3], não é de se surpreender que haja também um aumento da demanda na área de educação cristã, especialmente a educação cristã que converge para o uso de ferramentas tecnológicas como a internet para sua implantação.

A empresa Zeester[4] desenvolve ferramentas de ensino religioso, assim como soluções para igrejas cristãs e comunidades que visam a divulgação de estudos teológicos e de prática de vida cristã. Este trabalho parte de um projeto desenvolvido como estágio dentro da empresa Zeester pelo aluno de graduação, autor deste texto, onde foi possível trabalhar no desenvolvimento do Front-end da ferramenta Blesss, que se propõe a fornecer vídeos, artigos, apostilas, livros e questionários para pessoas interessadas em aprofundar seus conhecimentos em disciplinas relacionadas à fé cristã. A empresa conta com uma equipe de desenvolvimento de 5 pessoas. O autor deste trabalho ocupou a função de desenvolvedor Front-end durante o projeto, que teve duração de 7 meses, de 13 de outubro de 2020 até 14 de maio de 2021.

A ferramenta de curso desenvolvida na plataforma Blesss tem como objetivo o aprendizado através de aulas em vídeo, apostila para complementar o conhecimento e questionários para testar o que foi aprendido. Com a criação da

[3] Em 2010, a porcentagem de evangélicos no Brasil atingiu mais de 22% da população, passando para 31% em 2020 segundo dados do IBGE.

[4] <https://zeester.com.br/>

ferramenta, houveram 370 inscrições durante um período de 8 dias após o lançamento. Além disso, houve um aumento de 122 assinantes na plataforma, o que representa um crescimento de 9% em relação ao número total antes de sua implementação, demonstrando a importância de sua criação para o modelo de negócios proposto.

O desenvolvimento da ferramenta de cursos teve a duração de dois meses durante o período de estágio. Este trabalho irá apresentar os principais conceitos e tecnologias voltados a informar o leitor sobre as ferramentas utilizadas durante a implementação do projeto (Capítulo 2). O Capítulo 3 apresentará a metodologia do trabalho desenvolvido, bem como a organização da empresa. No Capítulo 4 serão mostrados os resultados, bem como a interface desenvolvida para cada módulo do curso. Dentre os resultados, se destacarão também as decisões tomadas pela equipe para atender os requisitos do cliente. Finalmente, no Capítulo 5 iremos discutir o aprendizado obtido por meio do estágio, futuras melhorias na ferramenta bem como resultados já obtidos a partir do lançamento do produto.

2 CONCEITOS E TECNOLOGIAS

Este capítulo tem como objetivo apresentar conceitos e tecnologias que foram utilizados durante o desenvolvimento da ferramenta na plataforma Blesss[1]. Primeiramente, será apresentado a linguagem utilizada no projeto, em seguida conceitos referentes ao desenvolvimento e, por fim, princípios utilizados na gestão e organização da equipe de trabalho.

2.1 JavaScript

JavaScript é uma linguagem de programação de alto nível, dinâmica, interpretada e não tipada, conveniente para desenvolvimento orientado à objetos ou funcional (FLANAGAN, 2013). O Javascript é utilizado como linguagem de programação para o lado do cliente por mais de 97% dos sites (Q-SUCCESS, 2021) e tem suporte por todos os navegadores modernos, por exemplo Google Chrome [2] e Mozilla Firefox [3] (W3SCHOOLS, 2021).

O desenvolvimento web tem como principal objetivo definir o comportamento das páginas. O Javascript possui algumas alternativas para desenvolvimento de páginas, como o Dart [4] por exemplo, mas se mantém como o mais utilizado por ter algumas vantagens como: alta compatibilidade com plataformas, o fato de os navegadores interpretarem a linguagem, sem necessidade de uso de um compilador e por ser mais leve e rápido que outras linguagens de programação.

Uma vantagem do Javascript em relação às linguagens mais tradicionais é que ele é uma linguagem multiparadigma, isso quer dizer que ele suporta mais de um paradigma de programação. A linguagem permite utilizar o paradigma estrutural ou orientado a objetos e também imperativo ou funcional. Cada um dos paradigmas possui suas características, como será explicado a seguir:

[1] <https://blesss.org/>
[2] Disponível em <https://www.google.pt/intl/pt-PT/chrome>.
[3] Disponível em <https://www.mozilla.org/pt-BR/firefox/new>.
[4] Disponível em <https://dart.dev>.

- Paradigma Estruturado - É o paradigma tradicional de programação que permite a criação de programas por meio de sequências de instruções e subrotinas.

- Paradigma Orientado a Objetos - Permite encapsular os atributos e funções dentro de seu contexto de uso, que é definido por meio de abstrações de objetos reais.

- Paradigma Imperativo - Descreve como as ações são feitas, por meio de uma sequência de passos.

- Paradigma Funcional - Descreve o que deve ser feito, sem especificar qual a sequência de passos para alcançar o resultado.

Além disso, o Javascript é uma linguagem fracamente tipada, isso quer dizer que os tipos das variáveis podem mudar durante a execução do programa sem a necessidade de uma conversão explícita.

Um exemplo simples de código em Javascript pode ser visto a seguir:

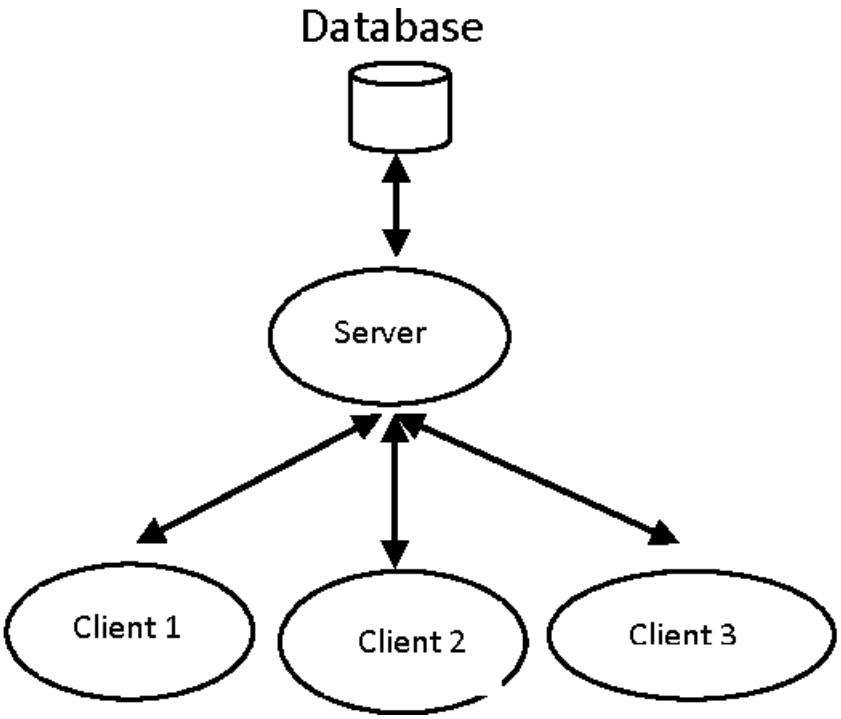
```
// Exemplo de programa simples demonstrando a sintaxe da
   linguagem e os paradigmas estrutural, imperativo e
   funcional
// O programa calcula a soma de dois arrays

function soma(a, b) {
  //Subrotina simples que calcula apenas a soma entre dois
     numeros
  return a + b;
}

const array1 = [0, 1, 2, 3]; // Declaracao de um array, nao
    e necessario especificar o tipo
const array2 = [3, 2, 1, 0];
```

```
const arrayResultado = [];

for (let i = 0; i < 4; i++) {
  // Exemplo de codigo de repeticao imperativo
  arrayResultado.push(soma(array1[i], array2[i])); //
      Funcao para adicionar elemento ao array
}

arrayResultado.forEach((elemento) => {
  // Exemplo de codigo de repeticao funcional que percorre
      o array e recebe cada elemento
  console.log(elemento); // Imprime cada elemento
});
```

2.2 Modelo Cliente-Servidor

O Modelo Cliente-Servidor é um modelo de software que se divide entre duas partes, cliente e servidor. O cliente é a parte do sistema que é executado do lado do usuário, nos dispositivos ou computadores, e faz requisições ao servidor. O servidor, lida com o armazenamento dos dados e o processamento das requisições para retornar o que foi solicitado pelo cliente. O pilar deste modelo é a comunicação entre as duas partes (OLUWATOSIN, 2014). Essa comunicação pode ser feita por meio do protocolo HTTP (Hypertext Transfer Protocol). Uma representação do modelo pode ser visto na figura 2.1, onde vários clientes se conectam a um servidor que por sua vez se conecta com o banco de dados.

Figura 2.1 – Modelo Cliente-Servidor



Fonte: (OLUWATOSIN, 2014)

Front-end é um termo da Engenharia de Software que se refere à camada de apresentação em aplicações que utilizam o modelo cliente-servidor. O Front-end também pode ser entendido como a interface apresentada para o usuário e com o qual ele interage. O desenvolvimento do Front-end geralmente envolve a criação de páginas web, comumente implementadas usando as linguagens HTML (Hyper-text Markup Language), CSS (Cascading Style Sheets) e JavaScript. As páginas web são armazenadas e disponibilizadas por servidores e podem ser acessadas a partir de navegadores em qualquer dispositivo. Ao construir interfaces é importante levar em consideração o desempenho e a experiência do usuário (DINH; WANG, 2020).

As métricas para avaliar se um site tem um bom desempenho geralmente estão ligadas à velocidade. Algumas métricas que devem ser levadas em conta para o desenvolvimento do Front-end são: tempo de carregamento, tempo para

aparecer o primeiro conteúdo na tela e latência ao interagir com o site (KROGH, 2020).

Ao longo do tempo, surgiram várias bibliotecas para melhorar o fluxo de desenvolvimento e manutenção de projetos Front-end, como as bibliotecas React, Next.js e Redux, explicadas na subseção 2.2.3.

A expêriencia do usuário, termo conhecido como *User Experience* ou apenas UX, leva em conta tanto o design quanto os sentimentos que o usuário tem ao usar a ferramenta. Para melhorar a experiência do usuário, é necessário considerar algumas convenções já estabelecidas, como por exemplo as Heurísticas de Nielsen [5]. Além disso, é importante fazer testes diretamente com os usuários do sistema para fazer com que seja cada vez mais confortável para se usar (L., 2021).

O Back-end, por sua vez, se refere à construção de API's (*Application Programming Interface*, ou Interface de Programação de Aplicativos, em tradução), ou seja, recursos que são disponibilizados para serem consumidos por meio de requisições. API's são geralmente compostas de uma base de dados em banco de dados, de onde as respostas às requisições dos usuários são obtidas e retornadas conforme esperado. Alguns exemplos de tecnologias que podem ser usadas para construir uma API são: Node.js, visto na subseção 2.2.1, ASP.NET, Rails, Django, etc.

Dessa forma, uma API consiste em rotinas e padrões para acesso de uma aplicação por meio da web (CANALTECH, 2021). Na subseção 2.2.2 será apresentado uma arquitetura de construção de API's e na 2.2.3 a ferramenta utilizada no projeto para fazer o acesso do Front-End para o Back-end.

2.2.1 Node.js

Node.js é um ambiente multi-plataforma *open-source* que permite executar JavaScript em tempo real fora do navegador do usuário. Node.js é muito uti-

[5] Disponível em <https://www.nngroup.com/articles/ten-usability-heuristics/>

lizado para a construção do serviços Back-end, ou API's, explicadas acima. Isso pode ser visto em aplicativos (ou *apps*) para a web rodando dentro de um navegador ou em dispositivos móveis. Sendo assim, só é retornado para o usuário o resultado do que é feito no servidor de Back-end, onde os dados são armazenados e as requisições atendidas (PEREIRA, 2014).

O Node.js possui um bom desempenho principalmente em sistemas com alta carga de processamento e é base para construção de diversas outras bibliotecas, como por exemplo o React, apresentado na subseção 2.2.3.

A escolha do Node.js como tecnologia de Back-end do projeto se deve ao fato de que é amplamente utilizado, por exemplo, por empresas como PayPal, Uber e Netflix e, por isso, possui uma grande comunidade de desenvolvedores. Portanto, é fácil encontrar bibliotecas open-source e gratuitas para resolver a maioria dos problemas que forem enfrentados durante o desenvolvimento além de permitir implementações em baixo nível para casos mais específicos. Como é possível utilizar o JavaScript no ambiente Node.js, a escolha também se torna interessante porque permite que o código fonte fique consistente com o Front-end.

2.2.2 REST

REST é um estilo de arquitetura muito utilizado que tem como objetivo construir serviços web e facilitar a integração de sistemas. As requisições são feitas em HTML e os dados são geralmente retornados em JSON[6] ou XML[7] . Um sistema RESTful, é um sistema que segue os princípios do REST (LECHETA, 2015). Basicamente, a arquitetura consiste em manter a semântica de acordo com o método utilizado em cada requisição, como será apresentado a seguir:

- GET é utilizado para consultas;
- POST é utilizado para inserir;

[6] Disponível em <https://www.json.org>
[7] Disponível em <https://www.w3.org/TR/xml/>

- PUT é utilizado para atualizar;
- DELETE é utilizado para excluir;

2.2.3 React e outras bibliotecas

O **React** é uma biblioteca JavaScript de código aberto desenvolvida e mantida principalmente pelo Facebook que permite a criação de interfaces de usuário. A biblioteca surgiu com o objetivo de facilitar o desenvolvimento do Front-end, oferecendo maneiras mais simples de tratar os estados da aplicação, em relação ao desenvolvimento tradicional. Além disso, permite uma organização baseada em componentes, em que cada componente tem suas características e estados e que podem se combinar para construir componentes mais complexos. Os componentes utilizam de uma sintaxe chamada JSX, que é similar ao XML e ao HTML, como pode ser visto abaixo (REACT, 2021).

```
// Exemplo de componente em React para demonstrar a
   semelhanca com o HTML

import React from "react"; // Importa a biblioteca React

function Example() {
  // O Componente pode ser uma funcao ou uma classe, cada
     um com suas caracteristicas
  return (
    // O retorno representa o que vai ser mostrado na tela
    // Tags div, h1 e p semelhantes ao HTML
    <div>
      <h1 style={{ color: "red", fontSize: 30 }}>Titulo</h1
         >
      {/* Declaracao de estilos inline */}
      <p>Texto</p>
    </div>
```

```
  );
}

export default Example;
```

O React, por padrão, constrói uma aplicação de página única, isto é, implementa uma arquitetura que o código JavaScript é todo baixado para o cliente e depois qualquer atualização de estado é feita diretamente no navegador. Isso significa que o servidor não é responsável pela renderização das páginas (KONSHIN, 2018).

O **Axios** é um cliente HTTP para Node.js e para navegadores. É uma ferramenta utilizada para realizar requisições HTTP e automaticamente transformar a resposta em JSON (AXIOS, 2021). Pode ser usado dentro de um sistema React para acessar uma API externa que utiliza da arquitetura REST.

O **Next.js** é um framework baseado em React que permite a renderização do lado do servidor, geração de páginas estáticas, além de oferecer diversos recursos para melhorar a experiência de desenvolvimento. (VERCEL, 2021)

Ao utilizar a funcionalidade de renderização pelo lado do servidor, é possível que a aplicação não seja mais toda baixada no primeiro acesso. Assim, cada requisição ao servidor é retornado uma página já pré-renderizada para o cliente, o que permite melhoras de desempenho e otimização para algoritmos de indexação de páginas. (KONSHIN, 2018)

O **Redux** é uma biblioteca e um padrão para gerenciar o estado da aplicação. Ele oferece uma maneira de armazenar o estado global da aplicação de modo que pode ser acessado e alterado por todos os componentes. A alteração dos estados é feita através de eventos chamados *actions*. É normalmente utilizado em conjunto com o React. (REDUX, 2021)

O **Material-ui** é a biblioteca de componentes utilizada no projeto. A biblioteca oferece componentes robustos e customizáveis permitindo a criação de

interfaces bonitas com menos tempo. É utilizada por grandes empresas como Spotify, Amazon, NASA e Netflix. (UI, 2021)

2.3 Scrum

Scrum é um framework ágil utilizado para gestão de desenvolvimento de produtos. Utiliza de abordagem iterativa e incremental e propõe diminuir os riscos do projeto entregando as partes do sistema aos poucos (SABBAGH, 2014). Segue os ideais do Manifesto para Desenvolvimento Ágil de Software, que possui os seguintes itens (BECK et al., 2001):

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano;

Em relação aos papéis que cada um exerce na implementação do Scrum, podemos encontrar principalmente os desenvolvedores, o Product Owner, e o Scrum Master. Suas funções serão apresentadas a seguir (EUAX, 2021):

- **Desenvolvedores:** São os responsáveis por implementar e dar manutenção nas funcionalidades.
- **Product Owner:** É o representante do produto. Decide quais as funcionalidades e a ordem em que serão implementadas.
- **Scrum Master:** É o responsável por defender a metodologia aplicada, com os princípios e práticas Scrum.

Para a organização do que deve ser feito, é criado um *Product Backlog*, que é uma lista de tarefas ordenadas pela prioridade em que serão feitas, que descreve o que deverá ser feito até a finalização ou lançamento da versão do produto.

Uma *Sprint* são ciclos curtos de desenvolvimento, que duram geralmente de 2 até quatro semanas, que possui as próprias tarefas definidas. A cada *Sprint*, tarefas são obtidas do *Product Backlog* e colocadas no *Sprint Backlog*, que consiste nas tarefas distribuídas para serem concluídas neste período. Durante a execução da *Sprint* acontecem alguns eventos que serão explicados a seguir (EUAX, 2021):

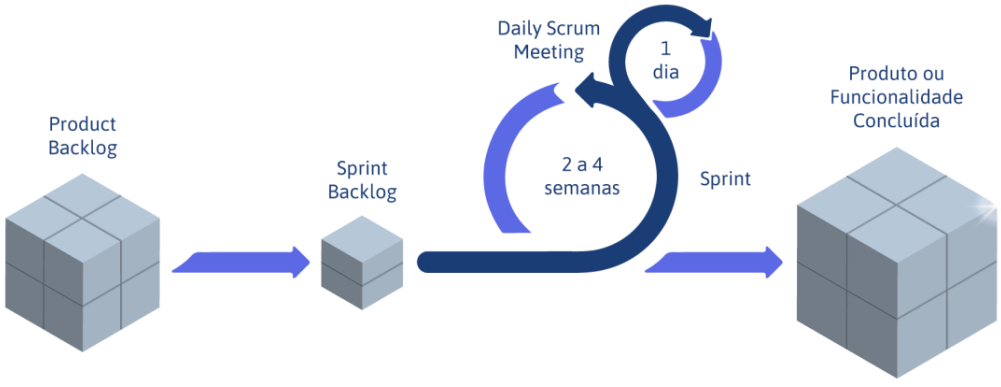
- **Sprint Planning:** É uma reunião que define o que será feito dentro da *Sprint*.

- **Daily Scrum:** É uma reunião diária que não pode durar mais de 15 minutos e que tem como objetivo responder 3 perguntas: o que foi feito no dia anterior, o que será feito hoje e o que pode atrapalhar durante a realização do trabalho planejado.

- **Sprint Review:** É uma reunião realizada ao final da *Sprint* para revisar as funcionalidades implementadas.

A figura 2.2 é uma representação da dinâmica do Scrum.

Figura 2.2 – Dinâmica do Scrum



Fonte: (EUAX, 2021)

2.4 Demais conceitos utilizados no projeto

SEO ou **Search Engine Optimization**, é o processo de melhorar o site para aumentar sua visibilidade para os mecanismos de buscas da web. Tem sua importância porque quanto mais visível a página está nas buscas, mais fácil é atrair clientes para o negócio. (SEARCH ENGINE LAND, 2021)

Git é um sistema de controle de versões distribuído gratuito e de código aberto. É utilizado pelas equipes de desenvolvimento para gerenciar as versões do sistema. Permite separar o fluxo de trabalho em ambientes de produção, teste e desenvolvimento além de salvar o estado do projeto caso seja necessário voltar atrás. Além disso, é possível trabalhar várias pessoas no mesmo projeto ao mesmo tempo, cada um em seu ambiente, até que o código fonte possa ser juntado para lançar uma nova versão (GIT, 2021).

3 METODOLOGIA

O trabalho realizado durante o estágio envolveu a implementação da ferramenta de cursos online dentro da plataforma Blesss. Os cursos têm como objetivo oferecer uma experiência de aprendizado sobre assuntos relacionados à fé cristã. Dessa forma, foi criado um plano de curso onde as pessoas se inscreveriam para depois ter acesso aos módulos do programa. Cada módulo apresenta os seus conteúdos em vídeos, questionários e apostilas, sendo todas essas ferramentas disponíveis dentro do ambiente de aprendizado do aluno.

Este capítulo será dedicado a explicar o funcionamento da organização Zeester (seção 3.1) e o fluxo de desenvolvimento adotado dentro da empresa (seção 3.2). Para a implementação da ferramenta de cursos, todo o processo padrão da empresa foi seguido.

3.1 Funcionamento da organização Zeester

Esta seção tem como objetivo explicar o funcionamento da organização Zeester, descrevendo seus processos e equipes, além de qual a sua área de atuação.

A organização originalmente se encontra na cidade de Lavras, MG, mas atualmente atua totalmente com trabalho remoto. O objetivo da empresa é oferecer soluções tecnológicas para negócios terceiros. Atualmente o principal produto mantido é a plataforma de aprendizado Blesss, que oferece recursos como vídeos, séries, lives, artigos e livros.

A organização conta com cinco equipes, sendo elas a de liderança, design, desenvolvimento, marketing e comunicação. As funções de cada equipe serão apresentadas a seguir:

- **A equipe de liderança** é responsável por tomar decisões sobre os produtos, em especial da plataforma Blesss.

- **A equipe de design** trabalha em conjunto com a de desenvolvimento e tem como objetivo analisar os requisitos e criar protótipos para melhor entendimento das funcionalidades.

- **A equipe de desenvolvimento** tem como objetivo implementar as novas funcionalidades, respeitando os requisitos obtidos.

- **A equipe de marketing** trabalha para tornar a plataforma conhecida, além de divulgar os lançamentos.

- **A equipe de comunicação** mantém contato com os usuários da plataforma para resolver problemas e tirar dúvidas.

A plataforma Blesss possui mais de 1000 horas de conteúdo para estudo e está presente na versão web, para aplicativo Android, versão de testes para o IOS e para as TVs LG e AndroidTV. Além disso, para o gerenciamento do conteúdo da plataforma, foi criado uma ferramenta chamada Blesss Studio. Essa ferramenta oferece aos administradores da plataforma funcionalidades como postar, retirar e destacar conteúdos, gerenciar usuários e assinaturas, responder comentários e acompanhar o sucesso de cada conteúdo por meio do número de visualizações e curtidas.

O atual trabalho apresenta o que foi desenvolvido durante o estágio na equipe de desenvolvimento, especificamente na plataforma em sua versão web. A função desempenhada foi de desenvolvedor Front-end.

3.2 Fluxo de desenvolvimento

Nessa seção será explicado como é o fluxo de trabalho da empresa Zeester utilizando o método Scrum. Será apresentado cada etapa de desenvolvimento, desde a definição de requisitos até a de produção, quando as novas funcionalidades já estão disponíveis para o usuário.

O fluxo de desenvolvimento das ferramentas na plataforma começa com o diretor da empresa, que exerce o papel de *Product Owner*. O diretor se comunica com o cliente e juntos decidem como se dará o desenvolvimento da plataforma. Após a idealização da nova ferramenta, a liderança de cada equipe é comunicada. O objetivo ao comunicar com a liderança de desenvolvimento é viabilizar e entender as possíveis dificuldades de implementação além de definir a prioridade das tarefas.

Depois desse primeiro passo, as tarefas a longo prazo definidas pela liderança são encaminhadas para a equipe de desenvolvimento. Uma vez entendidas as tarefas, adiciona-se ao *Product Backlog* as novas funcionalidades a serem implementadas. Essas funcionalidades são utilizadas nas reuniões de planejamento (*Planning*).

Na empresa, cada *Sprint* tem a duração de uma semana, tendo uma reunião semanal que une os conceitos de revisão (*Review*) e de *Planning*. No início dessa reunião é feita uma retrospectiva das tarefas implementadas na última semana, avaliando se a divisão foi correta, se as tarefas foram bem definidas e lidando com possíveis problemas. Além disso, é feita uma avaliação do que melhorou em relação à *Sprint* anterior e o que ainda pode melhorar, incluindo questões pessoais, de organização e questões de toda a equipe. Por fim, é realizado o *Planning*, que utiliza do *Product Backlog* para obter as próximas tarefas e as dividem para se adequarem ao padrão adotado, sendo delegadas para os desenvolvedores implementarem durante a *Sprint*.

Cada tarefa definida e delegada é também votada por toda a equipe usando o conceito de *Planning Poker*, onde cada um dos desenvolvedores tentam prever a dificuldade da tarefa, classificando de acordo com um nível de dificuldade definido pela equipe. As tarefas recebem um peso, chamado de *Story Points*, que varia de valores baixos como 1, quando a tarefa é considerada muito fácil, até 5 que seria difícil. Acima de 5 seriam tarefas que certamente não seriam terminadas no tempo

de uma *Sprint*. Para essas tarefas a melhor solução é dividir a tarefa em outras menores para se adequar aos pesos mais utilizados. Quando a votação é finalizada, acontece uma avaliação, que verifica se os pesos esperados estão de acordo com o histórico de conclusão da equipe e individual, para evitar excesso ou falta de trabalho aos desenvolvedores.

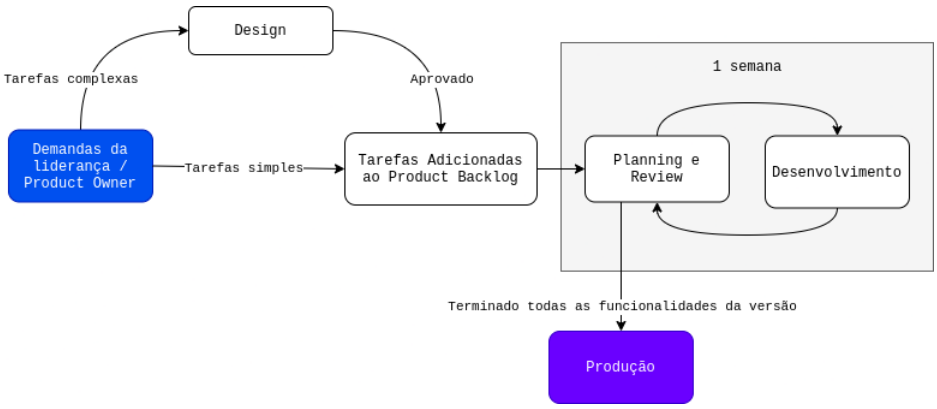
Algumas vezes, enquanto a liderança realiza a definição das tarefas, percebe-se a necessidade de implementar antes do desenvolvimento em si, um protótipo que aborda tudo aquilo que será feito. Esse protótipo serve como um intermediador daquilo que a liderança e a equipe de desenvolvimento entendem, evitando retrabalho, já que antes da implementação, entra-se em acordo sobre cada parte da funcionalidade.

Quando o protótipo está pronto, ele é apresentado para a liderança (em especial para o *Product Owner*) e para a equipe de desenvolvimento. Quando é aprovado, passa-se à criação do *Product Backlog* para ser usado nas *Sprints*.

O resultado das implementações é apresentado para o *Product Owner* quando todas as tarefas propostas no *Product Backlog* são finalizadas. Isso pode ocorrer dentro de uma ou mais *Sprints*. Caso haja entendimento de que não é necessário nenhuma mudança ou adição ao sistema, coloca-se o código fonte em produção. Todo o fluxo de desenvolvimento pode ser resumido na figura 3.1.

Durante o desenvolvimento, é usado o Git para controle de versões. A estratégia usada é de uma *branch*, ou ramificação, para cada tarefa a ser realizada. Desta maneira quando a tarefa é iniciada, cria-se uma ramificação onde cada desenvolvedor irá trabalhar. Ao fim da tarefa, o código é combinado com o código principal, que será disponibilizado para os usuários.

Figura 3.1 – Fluxo de desenvolvimento



Fonte: do autor

4 RESULTADOS

Nesse capítulo será apresentado os resultados de todas as etapas de produção do sistema Blesss para educação cristã online. A seção 4.1 mostrará os resultados da etapa de definição dos requisitos. A seção 4.2 mostrará as decisões tomadas após a definição de requisitos e a implementação da ferramenta. A subseção 4.2.1 mostrará os resultados obtidos ao implementar técnicas de SEO. A subseção 4.2.2 mostrará os resultados de cada página final do sistema.

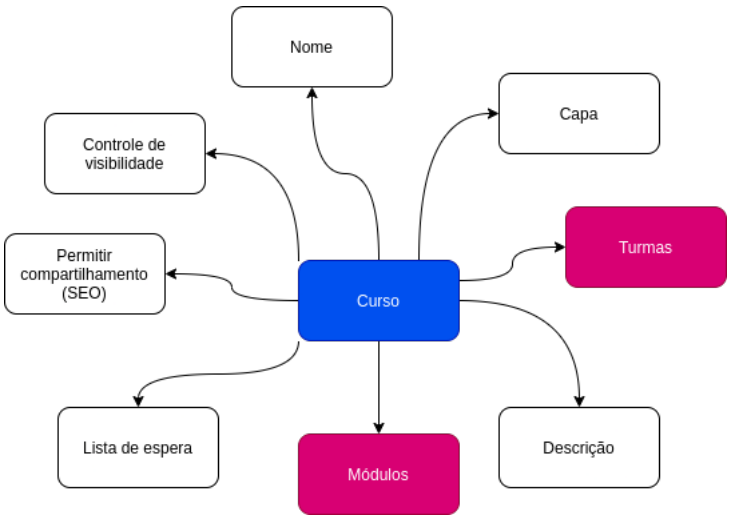
4.1 Definição dos requisitos

A primeira fase para o desenvolvimento da ferramenta de cursos foi a definição dos requisitos. Para isso, reuniu-se o *Product Owner* com a equipe de desenvolvimento e de design para discutir cada funcionalidade que estaria disponível. Foram várias reuniões seguidas com o objetivo de adequar a ferramenta para o modelo utilizado por outras plataformas semelhantes, mas tendo foco em trazer inovações, além de ser incorporada na plataforma já existente.

Na parte de definição dos requisitos, dividiu-se o novo sistema em três principais entidades, sendo elas: **(1) curso**, **(2) módulo** e **(3) conteúdo**.

A entidade **curso** (1) é a estrutura mais externa, que possui vários módulos, além de ter toda informação referente ao curso, como, por exemplo, o nome, a descrição e uma imagem de capa (Figura 4.1). O curso também é o que deve ser usado para se divulgar tanto dentro da plataforma, nas páginas iniciais, como em mídias sociais.

Figura 4.1 – Diagrama da entidade Curso



Fonte: do autor (2021)

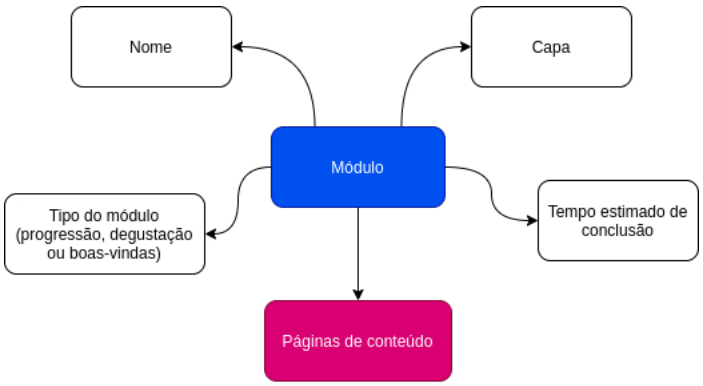
A entidade **módulo** (2), por sua vez, se encontra dentro do curso, e que possui um conjunto de páginas de conteúdo (Figura 4.2). Cada módulo tem informações sobre o que será abordado, além de uma previsão do tempo demorado para sua conclusão. Foram definidos três tipos de módulo:

- **Módulo de progressão:** módulo contendo o conteúdo em si, sendo seu conteúdo liberado de acordo com a progressão da turma. É o principal módulo da ferramenta.

- **Módulo de degustação:** módulo para todas as pessoas, assinantes ou não assinantes. Mostra como é o conteúdo do curso para motivar as pessoas a se inscreverem nele.

- **Módulo de boas-vindas:** módulo que explica como funcionará o curso para o usuário e fica disponível logo após o usuário se inscrever no curso.

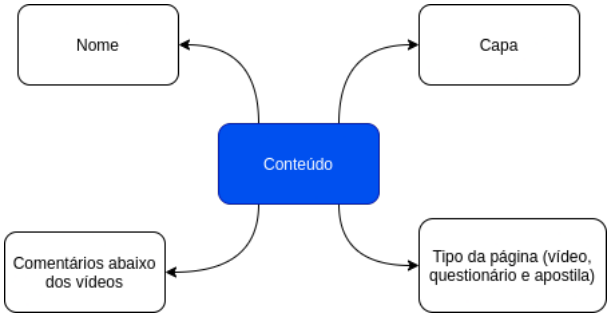
Figura 4.2 – Diagrama da entidade Módulo



Fonte: do autor (2021)

Por fim, foi desenvolvida a entidade **conteúdo** (3), que compreende vídeos, questionários e apostilas. Caso o conteúdo seja um vídeo, a ferramenta permite aos usuários comentarem e interagirem entre si (Figura 4.3). Para os questionários, a cada resposta marcada o usuário recebe o resultado com uma explicação para cada alternativa. Ao final é apresentado um resumo de quantos acertos e erros, sendo possível reiniciar o questionário. Por último, a apostila é um leitor que permite o usuário ler o conteúdo disponibilizado pelos professores.

Figura 4.3 – Diagrama da entidade Conteúdo



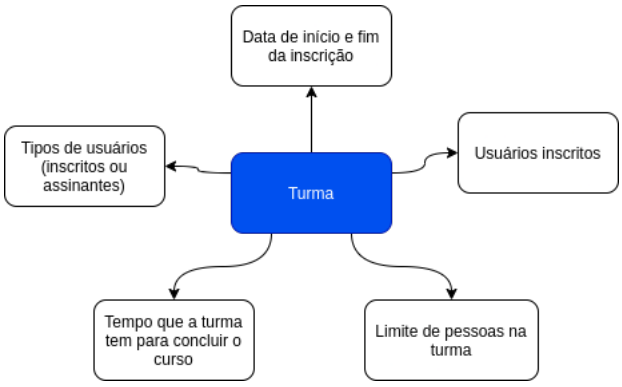
Fonte: do autor

Para que o usuário consiga acessar o conteúdo disponível no curso, é necessário que ele possua uma inscrição à turma. Portanto, foi definido uma entidade chamada *turma* (Figura 4.4), que possui valores como: data de início e fim da ins-

crição, data de início da turma, os usuários que se inscreveram na turma e quanto tempo a turma tem para finalizar o curso. A data de lançamento dos módulos é contada a partir do início de cada turma, sendo possível manter para cada turma uma progressão independente .

Além disso, quando a inscrição está fechada, existe a possibilidade do usuário entrar na lista de espera, para quando uma nova inscrição abrir ele ser notificado. Para cada turma também existe a configuração de quem pode se inscrever, que pode ser para todos os usuários do sistema ou apenas para os assinantes.

Figura 4.4 – Diagrama da entidade Turma



Fonte: do autor

Os diagramas mostrados nas Figuras 4.1, 4.2, 4.3 e 4.4 são o resultado do levantamento de requisitos, e mostram as principais funcionalidades da ferramenta de cursos.

4.2 Desenvolvimento da ferramenta

O protótipo foi avaliado e adaptado para melhorias e ajustes, e o resultado final foi transferido para a equipe de liderança para aprovação. Após a aprovação da equipe de design, passou-se à equipe de desenvolvimento para implementação.

O acesso à API desenvolvida é feito por meio das rotas do Back-end, ou seja, quais URLs estariam acessíveis para obter o conteúdo. Para atender às neces-

sidades do Front-end, viu-se a necessidade de uma rota para obter dados básicos do curso, para ser usado no SEO das páginas.

Foi também necessário criar uma rota para obter a situação do curso para cada usuário, isto é, se a inscrição está aberta, se o usuário está inscrito, se o curso já começou ou se apenas é possível entrar na lista de espera.

Para as páginas mais internas, foi necessário criar rotas que retornam cada entidade definida. A primeira para todos os cursos, a segunda para o curso em si, que retorna a situação de cada módulo, e depois mais duas para obter o módulo e o conteúdo, cada um com seus detalhes. A implementação dessas rotas da API não são apresentadas neste trabalho porque o foco está no Front-end da aplicação.

O desenvolvimento dos componentes que compõem a parte visual da aplicação foram desenvolvidos por meio da biblioteca React (Seção 2.2.3). Para agilizar o processo de desenvolvimento, optou-se por usar o Material-UI como biblioteca de componentes. Para uma melhor manutenção de código, cada componente do sistema é responsável por tratar suas próprias funcionalidades, com seus estados e propriedades. Além disso, a interface foi construída para se adequar à diferentes dispositivos, usando técnicas do próprio Material-UI de responsividade.

A comunicação com o Back-end em todas as telas foi feita através da biblioteca Axios, que é chamada dentro de funções que invocam as actions. As actions são ações enviadas que tem como objetivo alterar o estado global da aplicação, armazenado através do Redux. O fluxo para buscar e obter uma informação é o seguinte:

1. Uma função é invocada dentro do componente que precisa da informação.

2. A função é responsável por mandar a requisição para o Back-end utilizando o Axios.

3. O resultado dispara uma Action com um tipo específico e com o resultado da consulta.

4. O *reducer* é o responsável por identificar pelo tipo da *action* qual é o estado que vai atualizar.

5. Depois que o estado global é atualizado com as informações vindas do Back-end, o componente obtém os dados e mostra na tela.

O ambiente de desenvolvimento utilizado foi o VSCode [1] e para o gerenciamento de repositório git o GitLab [2]. Nas subseções a seguir será apresentado como foi desenvolvido cada parte que compõe a ferramenta de cursos. Todo o código escrito seguiu os padrões definidos acima.

Os resultados mostrados neste capítulo usam como exemplo um curso disponível na plataforma Blesss, chamado Teologia na Prática[3]

4.2.1 SEO

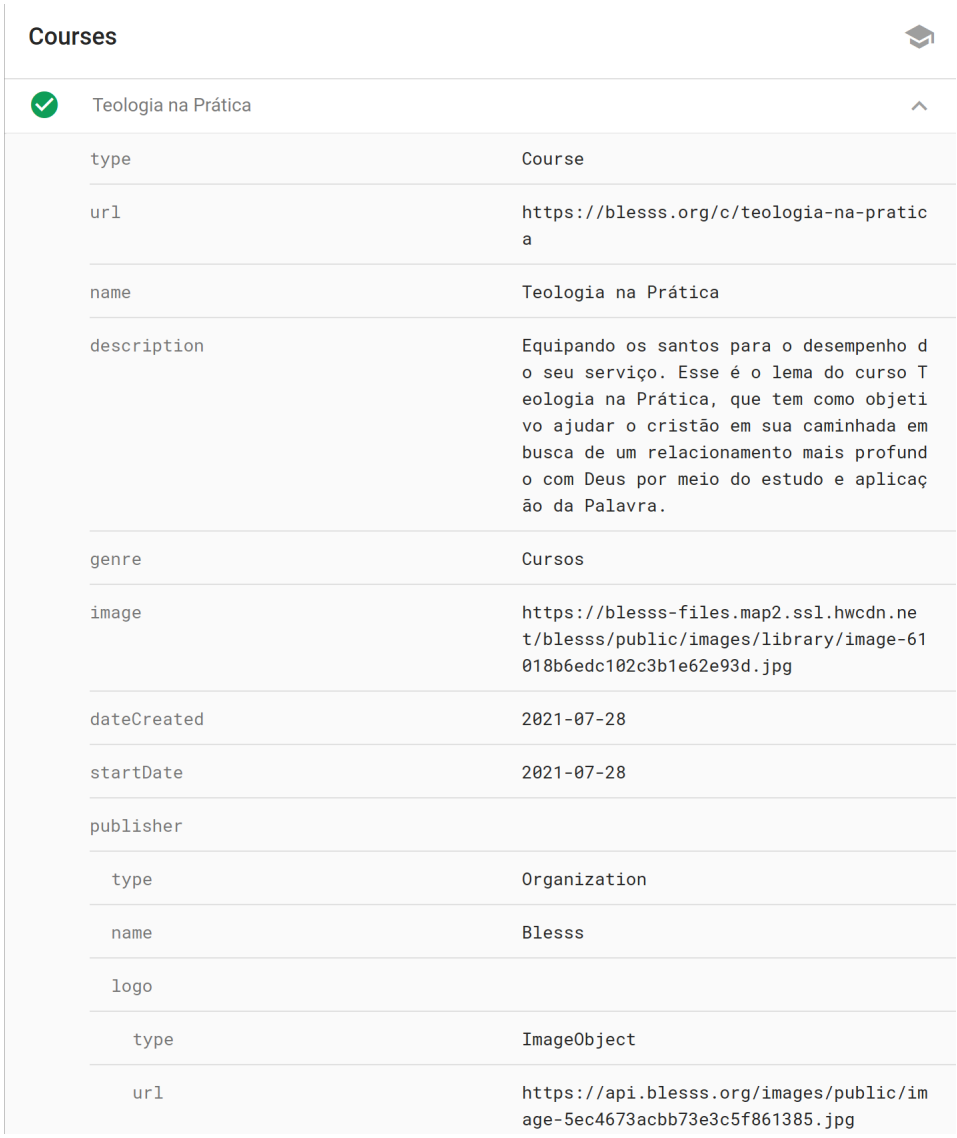
Para a implementação do SEO, foi utilizada a tecnologia do Next.js que permite a página ser pré-renderizada pelo servidor. O Next.js permite por meio de uma função pré-definida, executar código logo depois da página ser requisitada pelo usuário, ainda dentro do servidor. Esta funcionalidade foi utilizada para retornar as informações básicas do curso já contidas na página, por meio de tags HTML que definem informações básicas do conteúdo da tela. Isso permite que a página seja melhor reconhecida pelos mecanismos de busca.

Para a implementação desta funcionalidade, foi utilizada a biblioteca Axios, que do lado do servidor executa uma consulta diretamente à API e retorna os dados referentes ao curso dentro da página.

Ao utilizar uma ferramenta de testes de resultado do Google [4] foi obtido um bom resultado em relação ao reconhecimento do conteúdo da página pelos mecanismos de busca, como pode ser visto na figura 4.5.

[1] Disponível em <https://code.visualstudio.com/>.
[2] Disponível em <https://gitlab.com/>.
[3] Disponível em <https://blesss.org/c/teologia-na-pratica>.
[4] Disponível em <https://search.google.com/test/rich-results>.

Figura 4.5 – Resultado do teste pela ferramenta do Google



Fonte: do autor (2021)

Outro benefício obtido por meio da utilização de requisições pelo lado do servidor e na implementação do SEO foi no compartilhamento de links nas redes sociais. A figura 4.6 apresenta como o link é apresentado ao ser compartilhado no

Facebook. Essa melhor visualização permite que o link chame mais atenção, o que é muito importante para o marketing do produto.

Figura 4.6 – Compartilhamento do link no Facebook

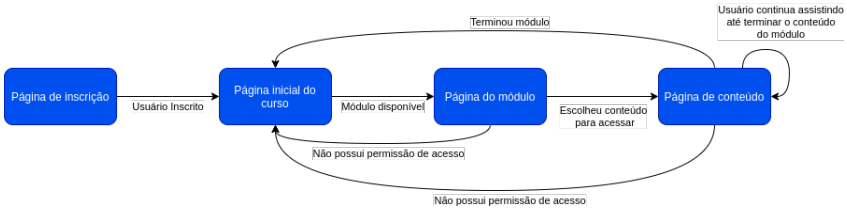
Fonte: do autor

Além disso, como o curso é exclusivo apenas para os inscritos, já nesta mesma requisição é feito a verificação se o usuário pode ou não acessar a página. Caso o usuário esteja em uma página que não pode acessar, já é feito imediatamente um redirecionamento para a página de inscrição do curso. Os dados de login do usuário são passados por *cookies* para o servidor.

4.2.2 Páginas

Nessa seção será mostrado os resultados da equipe de desenvolvimento para as páginas que compõem a ferramenta de cursos no sistema Blesss. O fluxo esperado do usuário entre as páginas é apresentado no diagrama da Figura 4.7.

Figura 4.7 – Fluxo entre as páginas do sistema



Fonte: do autor

4.2.2.1 Página de Inscrição

A página de inscrição conta com informações básicas do curso, como a capa, título, uma breve apresentação, uma apresentação mais detalhada e os professores presentes dentro do curso. As informações foram apresentadas na tela de acordo com o design proposto.

Além disso, através da conexão feita com o Back-end, é consultada qual a situação do curso, ou seja, se a inscrição está aberta e se o curso já começou. Para esta verificação, também é enviado através da biblioteca Axios o *token* de login do usuário, que o identifica dentro da API. Este *token* é utilizado para verificar a situação do usuário em relação ao curso, que envolve verificações como: se o usuário está inscrito, se ele pode se inscrever e até mesmo se ele entrou com sua conta na plataforma.

Para cada caso, é feito um tratamento no Front-end. Quando o usuário pode se inscrever, é apresentado um botão de inscrição no curso (Figura 4.8). Caso o usuário não possa se inscrever por não ser assinante, ou por não ter entrado com sua conta, é apresentado um botão para assinar a plataforma ou realizar login (Figura 4.9). Quando o usuário já está inscrito e o curso começou, é apresentado o botão de entrar dentro do curso (Figura 4.10). Por fim, quando nenhuma inscrição está aberta e o usuário ainda não se inscreveu no curso, é apresentado um botão de entrar na lista de espera (Figura 4.11).

Figura 4.8 – Página de inscrições quando o usuário pode se inscrever

Fonte: do autor

Figura 4.9 – Página de inscrições quando o usuário não pode se inscrever

Fonte: do autor

Figura 4.10 – Página de inscrições quando o usuário está pronto para acessar o conteúdo do curso

Fonte: do autor

Figura 4.11 – Página de inscrições quando o usuário pode apenas entrar na lista de espera

Fonte: do autor

4.2.2.2 Página inicial do curso

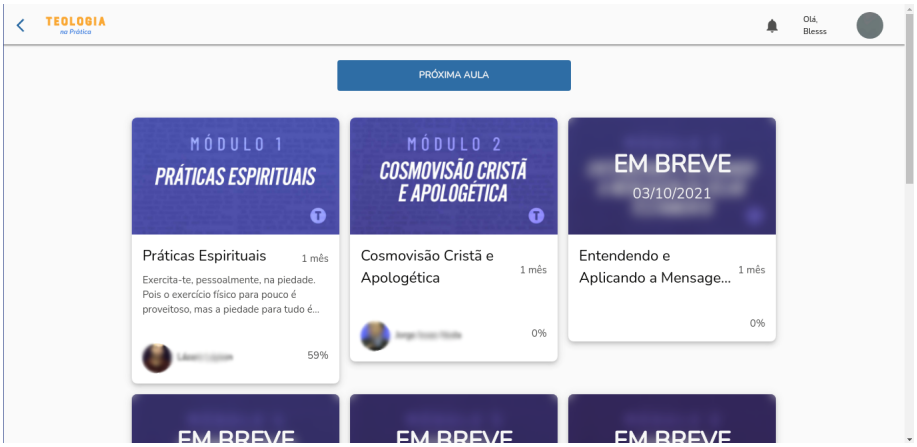
A página inicial do curso traz informações sobre o conteúdo que está dentro do curso. Este conteúdo é apresentado por meio dos módulos presentes no curso. Nesta tela, existe uma lista de módulos com a situação de cada um e o progresso do usuário. A situação do módulo pode ser:

- **Disponível:** O usuário pode acessar o conteúdo. Apresentado nos primeiros dois itens da Figura 4.12.

- **Em breve:** O módulo ainda não foi disponibilizado, mas é apresentado a data em que estará disponível. Um exemplo pode ser encontrado no terceiro item da Figura 4.12.

- **Disponível para inscritos:** Este módulo só pode ser acessado pelos inscritos no curso. Este caso só aparece para os usuários que entram nesta página sem antes se inscrever no curso (Figura 4.13).

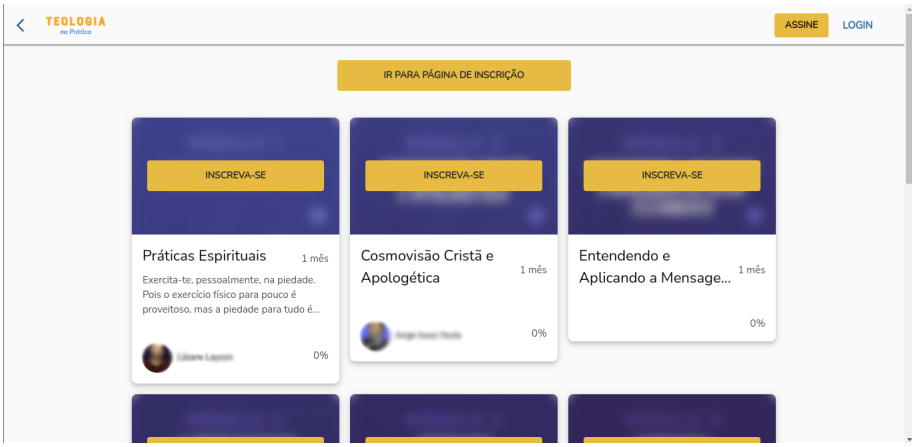
Figura 4.12 – Página inicial do curso para quando o usuário possui acesso ao curso



Fonte: do autor

O progresso em cada módulo é a porcentagem que o usuário já concluiu, útil para o usuário continuar de onde parou no curso e acompanhar seu progresso pessoal. No topo da página também existe um botão que permite o usuário continuar o progresso do curso de onde parou, que redireciona o usuário para a página de conteúdo correta. As informações dos módulos junto com o progresso do usuário são retornadas por meio da consulta feita à API.

Figura 4.13 – Página inicial do curso para quando o usuário não possui acesso ao curso

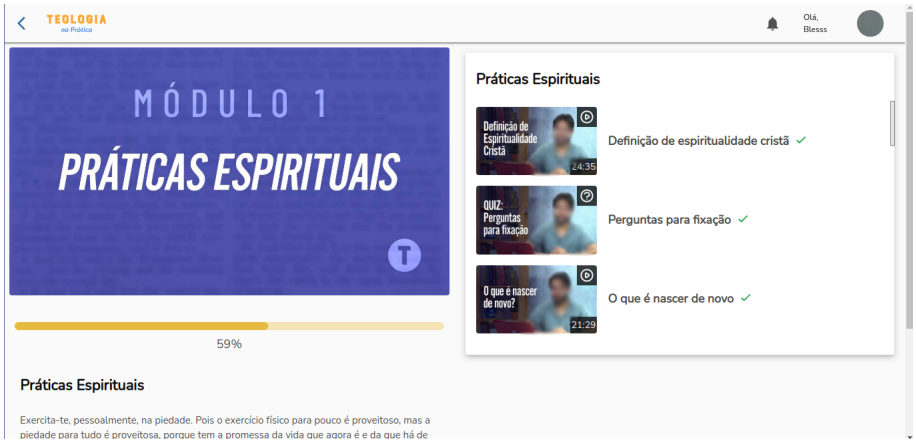


Fonte: do autor

4.2.2.3 Página do módulo

A página do módulo tem como objetivo mostrar o conteúdo disponível dentro do módulo (Figura 4.14). Ela possui informações como o título do módulo, capa, os professores que estão presentes nas aulas e por fim as páginas de conteúdo. Além disso é mostrado nesta página o progresso do aluno dentro do módulo e um botão para continuar de onde o usuário parou. Vale ressaltar que esta página é exclusiva para quem tem acesso ao módulo, isto quer dizer que para módulos do tipo progressão ou boas-vindas, somente os inscritos no curso conseguem acessar.

Figura 4.14 – Página do módulo



Fonte: do autor

O componente utilizado para mostrar todas as páginas de conteúdo do módulo foi planejado para ser reutilizado tanto nesta página quanto nas próprias páginas de conteúdo, o que permite uma melhor manutenção de código e evita repetição. Para que isso fosse possível, o componente recebe propriedades que identificam qual o contexto em que ele está sendo usado para adaptar seu comportamento.

4.2.2.4 Página de conteúdo

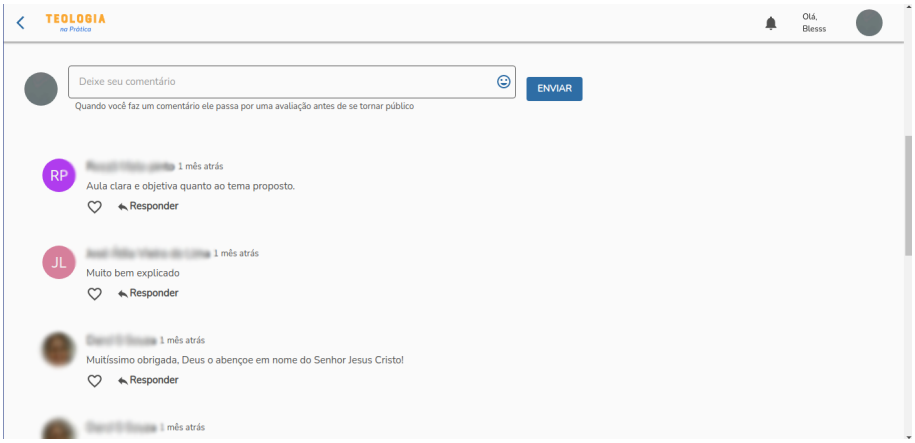
Por fim, foram construídas as páginas de conteúdo. O design destas páginas foi planejado para manter o mínimo de repetição de código possível. Para isso, o que muda entre os tipos de página é apenas o conteúdo em si, apresentado no canto superior esquerdo da página, todos os outros componentes foram reutilizados.

É nesta página que o componente de mostrar todo o conteúdo do módulo foi reutilizado, permitindo o usuário navegar e acessar o conteúdo desejado mais facilmente.

Embaixo do conteúdo principal, também foram apresentadas informações básicas do conteúdo, como nome, descrição e os professores envolvidos. No caso

da página ser de vídeo, abaixo destas informações ainda está disponível uma seção de comentários, onde é possível comentar, ver e responder comentários de outros usuários (Figura 4.15). Os comentários já estavam implementados na plataforma, sendo necessário apenas utilizar o componente pronto.

Figura 4.15 – Seção de comentários abaixo do vídeo



Fonte: do autor (2021)

Para o conteúdo de vídeo, foi utilizado como *player* de vídeo o Clappr[5]. A rota disponibilizada pela API retorna a URL do vídeo, que é passado para o *player* mostrar o conteúdo. O *player* permite controles de tocar e pausar, de volume, tela cheia e também controle de qualidade do vídeo. Quando o vídeo acaba o conteúdo é marcado automaticamente como concluído através de uma requisição à API, para contar no progresso do usuário. Além disso, de tempo em tempo é enviado uma requisição ao Back-end para informar onde o usuário está no vídeo, para que, caso o usuário pare de assistir antes do término, possa voltar de onde parou. Também é permitido ao usuário pular o conteúdo através do botão de Concluir e Continuar. O resultado da página pode ser visto na Figura 4.16.

[5] Disponível em <https://github.com/clappr/clappr>.

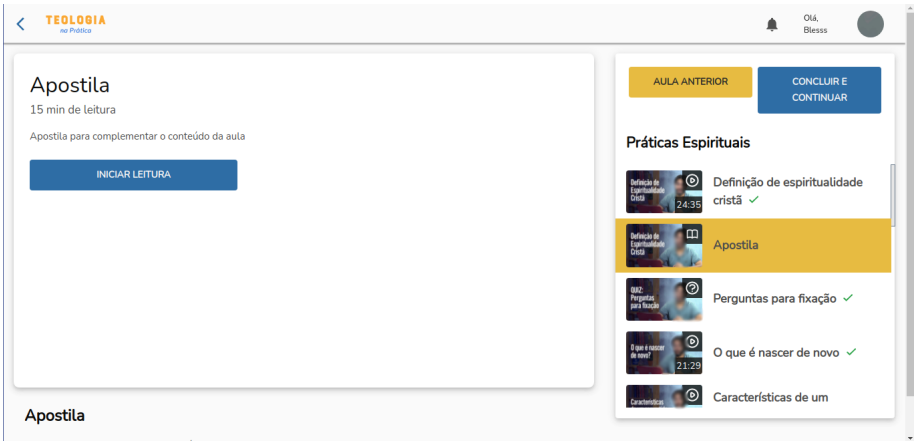
Figura 4.16 – Página do conteúdo de vídeo

Fonte: do autor (2021)

Para o conteúdo de apostila, o fluxo foi definido primeiramente com uma página inicial (Figura 4.17), dando informações iniciais sobre a apostila, e em seguida a leitura em si (Figura 4.18). Foi utilizado um leitor de PDF através da biblioteca React-pdf [6]. A API retorna a URL do arquivo PDF que é passado para o leitor. Na implementação, foi necessário construir componentes para o controle do leitor, como para aumentar e diminuir o zoom da página, navegar entre as páginas e também para mostrar quando o documento ainda está sendo carregado. Esses componentes foram construídos por meio de propriedades que o próprio React-pdf oferece, sendo preciso apenas atualizar os estados por meio de botões na tela.

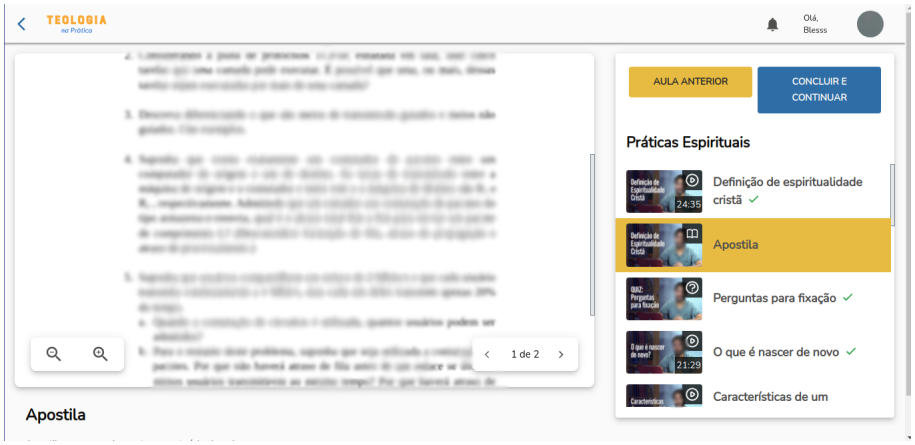
[6] Disponível em <https://react-pdf.org/>.

Figura 4.17 – Página inicial do conteúdo de apostila



Fonte: do autor

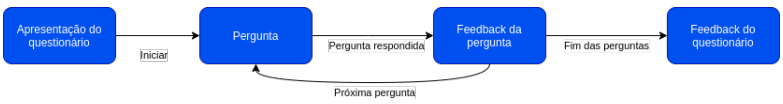
Figura 4.18 – Página de leitura do conteúdo de apostila



Fonte: do autor

Por fim, o conteúdo de questionário foi implementado por meio de requisições à API. O fluxo do usuário ao acessar um questionário pode ser compreendido através do diagrama da Figura 4.19.

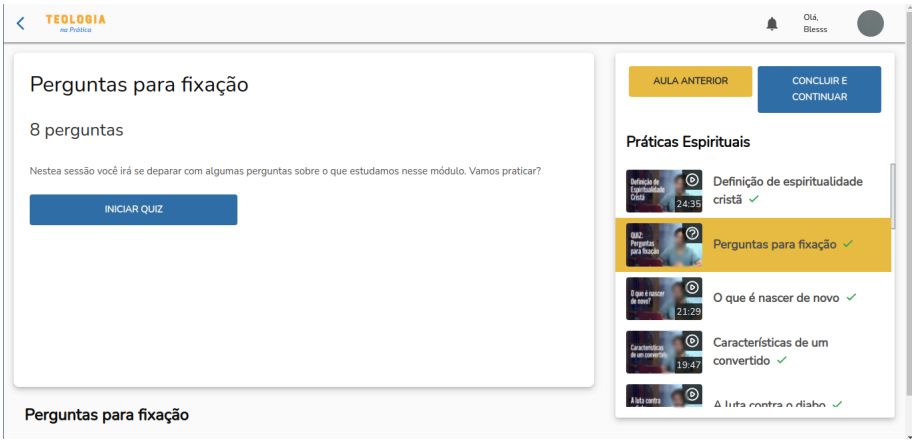
Figura 4.19 – Fluxo do usuário ao acessar um questionário



Fonte: do autor (2021)

Primeiro é apresentada uma página inicial com informações sobre o questionário (Figura 4.20). Todas as perguntas são retornadas na primeira requisição, cada uma com suas alternativas, mas sem as respostas. A pergunta é apresentada ao usuário de acordo com a Figura 4.21.

Figura 4.20 – Página inicial do questionário



Fonte: do autor (2021)

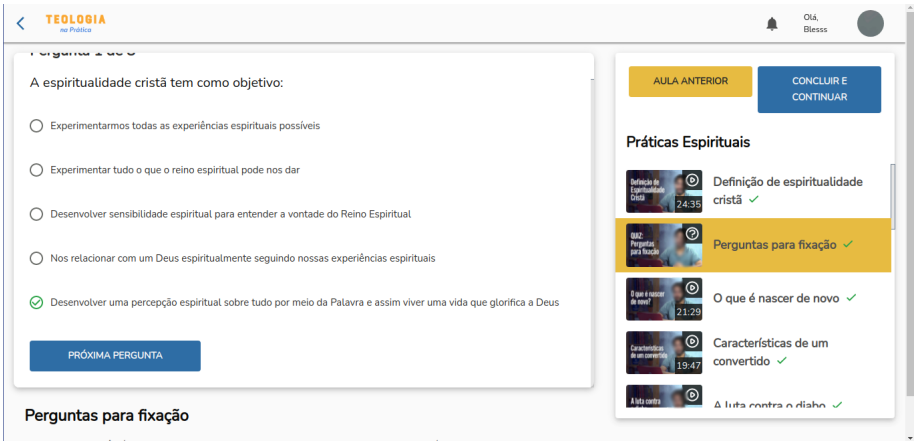
Figura 4.21 – Página de uma pergunta do questionário

Fonte: do autor (2021)

Quando o usuário responde uma pergunta, é enviado uma requisição para o Back-end para verificar a resposta, retornando se está correto (Figura 4.22) ou

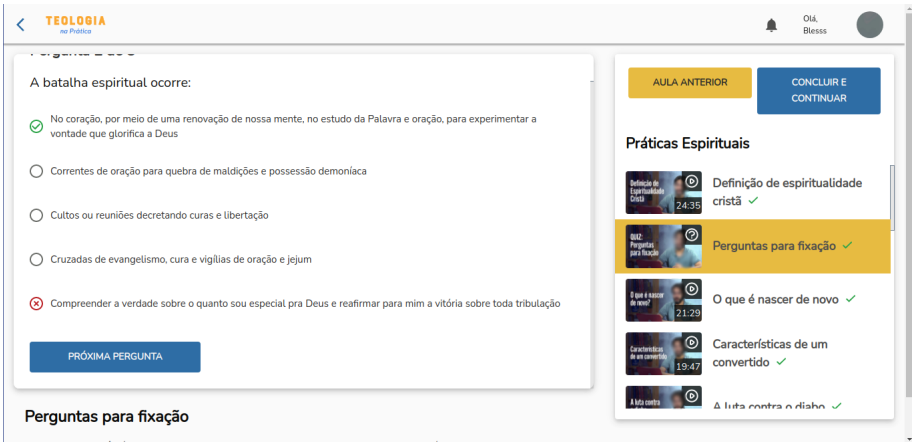
errado (Figura 4.23), além da explicação de cada alternativa. Após ler as explicações, o usuário consegue passar para o próximo passo e continuar respondendo.

Figura 4.22 – Página quando o usuário acerta a resposta



Fonte: do autor

Figura 4.23 – Página quando o usuário erra a resposta

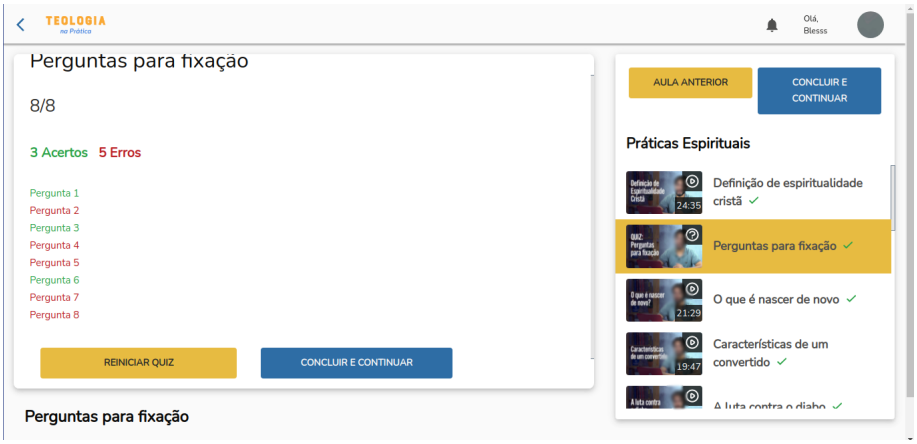


Fonte: do autor

Ao final do questionário é feito uma nova consulta à API para apresentar o resultado do questionário ao usuário, que mostra a quantidade de acertos e erros (Figura 4.24). O usuário decide então se vai refazer o teste ou passar para o próximo conteúdo. Caso o usuário saia no meio do questionário, ele consegue

voltar de onde parou, já que a API retorna o número de questões respondidas e o Front-end redireciona para a pergunta correta.

Figura 4.24 – Página de feedback do questionário



Fonte: do autor

4.3 Testes e Deploy

Quando o desenvolvimento foi finalizado, passou-se para a fase de testes. Como a equipe não possui desenvolvedores dedicados aos testes, toda a equipe ajudou nesta fase. Para isso, houve uma reunião onde as informações eram alteradas tanto no Back-end quanto no Front-end para ver se a aplicação estava lidando de maneira correta em todos os casos. Dessa maneira, diversos *bugs* foram descobertos e resolvidos antes do primeiro lançamento.

Após a fase de testes terminar, a ferramenta foi apresentada para o *Product Owner*, que aprovou a implementação e sugeriu acréscimos para as próximas versões. Depois disso, foi feita a implantação (*deploy*) no servidor.

As atualizações foram enviadas para o repositório remoto do git e logo depois atualizado no servidor através de scripts de construção. Após esse processo toda a ferramenta já estava disponível para os usuários utilizarem.

5 CONCLUSÃO

Sobre a realização do estágio, é possível concluir que foi um período de crescimento profissional, útil para aplicar conceitos aprendidos durante o curso de Ciência da Computação. Disciplinas como Sistemas Distribuídos, Programação WEB, Banco de Dados, Estrutura de Dados, Paradigmas de Linguagem de Programação e Engenharia de Software contribuíram para a realização de diversas tarefas, como as apresentadas neste trabalho.

Durante o desenvolvimento da plataforma Blesss também foi possível aprofundar em conhecimentos e aprender novas tecnologias. Além disso, foi estimulado o trabalho em equipe e a resolução de problemas reais, que contribuem muito para a formação profissional. Esta experiência desenvolveu habilidades que tornam o autor deste trabalho mais seguro e preparado para servir a sociedade enquanto profissional e cidadão.

Durante o desenvolvimento surgiram algumas dificuldades em relação aos processos da empresa. Como os processos de aprovação da definição de requisitos e prototipação não estavam bem definidos, algumas funcionalidades foram implementadas pela equipe de desenvolvimento, mas tiveram que ser alteradas posteriormente. Além disso, durante o processo de ingresso na empresa não houve um acompanhamento formal, o que não garantiu a comunicação com os desenvolvedores mais experientes.

O curso de Ciência da Computação ofereceu uma base para o aprendizado das tecnologias, mas para o início do desenvolvimento foi necessário dedicar cerca de um mês e meio à capacitação. Esse tempo poderia ser minimizado caso as disciplinas apresentassem tecnologias mais atuais e usadas no mercado. Além disso, na área de Engenharia de Software, houve dificuldade em implementar na prática modelos de gestão vistos em aula, devido à falta de prática de trabalhar com equipes maiores e de usar ferramentas que auxiliam estas equipes.

Em relação aos impactos para a plataforma Blesss, observamos um crescimento na quantidade de clientes após o lançamento da ferramenta. O primeiro curso lançado foi o curso Teologia na Prática, usado para apresentar os resultados no Capítulo 4. Para este curso, houveram 370 inscrições durante um período de 8 (oito) dias. Além disso, como o curso é exclusivo para assinantes, aconteceu um aumento de 122 assinantes na plataforma, um crescimento de 9% em relação ao número total anterior. Outros cursos estão sendo preparados para o lançamento, e espera-se que a plataforma ainda vai obter mais crescimento a partir deles.

Por fim, como trabalhos futuros, é possível aplicar melhorias à ferramenta criada. Um exemplo, é permitir a criação de transmissões ao vivo para os alunos do curso, o que contribui para o aprendizado e se encaixa no modelo de negócio proposto. Além disso, é possível aprimorar o sistema de questionários, que atualmente conta apenas com perguntas de múltipla escolha, adicionando perguntas discursivas e permitindo a correção pelos professores.

REFERÊNCIAS

AXIOS. **Getting Started**. 2021. Disponível em: <https://axios-http.com/docs/intro>. Acesso em: 2 set. 2021.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 2 set. 2021.

BERARDINO, A. D. et al. Dicionário patrístico e de antiguidades cristãs. **São Paulo: Paulus**, 2002.

CANALTECH. **O que é API?** 2021. Disponível em: <https://canaltech.com.br/software/o-que-e-api/>. Acesso em: 2 set. 2021.

DINH, D.; WANG, Z. **Modern front-end web development : how libraries and frameworks transform everything**. Dissertação (Bachelor's Thesis) — Turku University of Applied Sciences, 2020.

EUAX, A. S. J. consultora da. **Metodologia Scrum: confira um resumo completo sobre o assunto**. 2021. Disponível em: <https://www.euax.com.br/2017/05/metodologia-scrum-o-que-voce-precisa-saber-2/>. Acesso em: 2 set. 2021.

FLANAGAN, D. **JavaScript: O Guia Definitivo**. Bookman Editora, 2013. ISBN 9788565837484. Disponível em: <https://books.google.com.br/books?id=zWNyDgAAQBAJ>.

GIT. **About**. Git, 2021. Disponível em: <https://git-scm.com/about>. Acesso em: 2 set. 2021.

KONSHIN, K. **Next.js Quick Start Guide: Server-side rendering done right**. Packt Publishing, 2018. ISBN 9781788995849. Disponível em: <https://books.google.com.br/books?id=-rBmDwAAQBAJ>.

KROGH, H. G. **Frontend Performance: Measuring & KPIs**. 2020. Disponível em: <https://crystallize.com/blog/frontend-performance-measuring-and-kpis>. Acesso em: 9 set. 2021.

L., A. **O que é e qual a importância do User Experience**. 2021. Disponível em: <https://www.hostinger.com.br/tutoriais/ux-o-que-e-user-experience>. Acesso em: 9 set. 2021.

LECHETA, R. **Web services RESTful: Aprenda a criar web services RESTful em Java na nuvem do Google**. Novatec Editora, 2015. ISBN 9788575224540. Disponível em: <https://books.google.com.br/books?id=AyyLCgAAQBAJ>.

MANACORDA, M. A.; NOSELLA, P.; OLIVEIRA, R. dos A. **História da educação: da antiguidade aos nossos dias**. [S.l.]: Cortez, 2002.

OLUWATOSIN, S. Client-server model haroon. In: . [S.l.: s.n.], 2014.

PEREIRA, C. **Aplicações web real-time com Node.js**. Casa do Código, 2014. ISBN 9788566250930. Disponível em: <https://books.google.com.br/books?id=Wm-CCwAAQBAJ>.

Q-SUCCESS. **Usage statistics of JavaScript as client-side programming language on websites**. 2021. Disponível em: <https://w3techs.com/technologies/details/cp-javascript/>. Acesso em: 8 set. 2021.

REACT. **React**. Facebook, 2021. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 2 set. 2021.

REDUX. **Redux Essentials, Part 1: Redux Overview and Concepts**. Redux, 2021. Disponível em: <https://redux.js.org/tutorials/essentials/part-1-overview-concepts>. Acesso em: 2 set. 2021.

SABBAGH, R. **Scrum: Gestão ágil para projetos de sucesso**. Casa do Código, 2014. ISBN 9788566250954. Disponível em: <https://books.google.com.br/books?id=pG-CCwAAQBAJ>.

SEARCH ENGINE LAND. **What Is SEO / Search Engine Optimization?** 2021. Disponível em: <https://searchengineland.com/guide/what-is-seo>. Acesso em: 2 set. 2021.

UI, M. 2021. Disponível em: <https://material-ui.com/pt/>. Acesso em: 16 set. 2021.

VERCEL. **Next.js**. Vercel, 2021. Disponível em: <https://nextjs.org/learn/basics/create-nextjs-app>. Acesso em: 2 set. 2021.

W3SCHOOLS. 2021. Disponível em: <https://www.w3schools.com/js/js_versions.asp>. Acesso em: 8 set. 2021.