UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

THIAGO CORREIA MONTEIRO

ATS: UM *FRAMEWORK* PARA APOIAR A IMPLEMENTAÇÃO DE SOFTWARE ACADÊMICO

SOBRAL

2023

THIAGO CORREIA MONTEIRO

ATS: UM *FRAMEWORK* PARA APOIAR A IMPLEMENTAÇÃO DE SOFTWARE ACADÊMICO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do *Campus* Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Fischer Jônatas Ferreira

SOBRAL

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M78a Monteiro, Thiago Correia.
ATS: : UM FRAMEWORK PARA APOIAR A IMPLEMENTAÇÃO DE SOFTWARE ACADÊMICO / Thiago Correia Monteiro. – 2023.
55 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2023.
Orientação: Prof. Dr. Fischer Jônatas Ferreira.

1. Framework. 2. Software Acadêmico. 3. Desenvolvimento Web. I. Título.

CDD 621.39

THIAGO CORREIA MONTEIRO

ATS: UM *FRAMEWORK* PARA APOIAR A IMPLEMENTAÇÃO DE SOFTWARE ACADÊMICO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do *Campus* Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Fischer Jônatas Ferreira

Aprovado em: 12/07/2023

BANCA EXAMINADORA

_____
Prof. Dr. Fischer Jônatas Ferreira (Orientador)
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Wendley Souza da Silva
Universidade Federal do Ceará (UFC)

_____
Prof. Msc. Erick Aguiar Donato
Universidade Federal do Ceará (UFC)

Dedico este trabalho à minha família, à minha mãe, Rozangela, à minha irmã, Thais, a meu padrasto, Renato, e à minha noiva, Raquel, base incontestável de força e resiliência.

AGRADECIMENTOS

Agradeço em primeiro lugar aos meus familiares e seu apoio, amor e encorajamento, que foi muito necessário durante a graduação.

Agradeço à minha mãe, Rozangela Correia, por ser minha maior fonte de inspiração nessa existência. Sei das dificuldades que passamos, em como foi difícil me ver sair de casa, me manter financeiramente. Obrigado pelos valores que a senhora me passou, por ser essa pessoa maravilhosa a qual eu me orgulho de me espelhar.

Agradeço a minha irmã, Thais Correia, que sempre me ajudou em todos os âmbitos, me dando palavras de conforto, conselhos, tirando boas risadas e tornando sempre minha vida mais leve.

Agradeço também a minha noiva Raquel Santana, sem a qual eu não estaria terminando essa graduação. Você foi o gatilho da minha mudança como ser humano, como estudante, como profissional, me mostrou como eu deveria amadurecer e evoluir para me tornar de fato um homem.

Agradeço também ao meu padrasto Renato Sena, o qual me transmitiu diversos valores, conselhos, ensinamentos, e principalmente, como a vida é só uma e devemos sempre tentar levar ela com leveza, tranquilidade e um sorriso no rosto.

Agradeço, especialmente, ao meu orientador Prof. Fischer Jônatas Ferreira por ser meu guia na execução deste trabalho, pelo conhecimento compartilhado comigo e pela imensa paciência na orientação. Sei que por muitos momentos foi difícil me fazer focar no trabalho e terminar essa graduação, agradeço muito pela insistência e por ter acreditado que eu poderia finalizar esse ciclo.

Agradeço ao corpo docente do curso de Engenharia da Computação pelo conhecimento compartilhado, pelo meu aprendizado, sem o qual eu não conseguiria ter chegado até aqui.

Aos membros da banca examinadora Prof. Dr.Wendley Souza da Silva e Prof. Msc. Erick Aguiar Donato, meus sinceros agradecimentos por aceitarem fazer parte desse momento tão especial para mim.

Gostaria de agradecer também a todos os amigos que fiz nesta graduação, como Carlos Eduardo, Renan Henrique, Rhuan Sousa, Renato Brito, Iago Melo, Daniel Jorge, Rafael José dentre muitos outros que tornaram estes anos muito mais leves e me ajudaram em diversos momentos. Sem eles eu não estaria aqui.

“Technology itself is neither good nor bad.
People are good or bad”.

*(*Naveen Jain)

RESUMO

CONTEXTO: A tecnologia digital tem se tornado ferramenta essencial para a execução e aprimoramento de funções outrora realizadas de forma manual, o que possibilitou mudanças na sociedade em diferentes âmbitos e espaços. No que concerne ao ambiente acadêmico, observa-se uma crescente necessidade de inserir novas estratégias de concepção e organização de diferentes tarefas, tais quais, a utilização de sistemas de software. Tal automatização proposta favorece a interatividade entre setores e a utilização de programas que auxiliem na mecanização de processos dentro das universidades. MOTIVAÇÃO: Nesse contexto, observa-se a necessidade de se construir mecanismos de inovação tecnológica, possível pela utilização de software voltado para o âmbito acadêmico, que promova integração de diferentes setores e facilite a execução de processos burocráticos/manuais, por meio da criação de um *framework* a fim de garantir funcionalidade, rapidez, estabilidade e disponibilidade de adaptação ao desenvolvedor. OBJETIVO: Logo, este trabalho se propõe a desenvolver um *framework* intitulado ATS (*Academic Tool Support*) a ser utilizado como ferramenta facilitadora para implementar sistemas de software, a fim de mecanizar processos em ambientes acadêmicos. METODOLOGIA: A construção deste trabalho se delineou a partir de cinco etapas, a saber: I) Fundamentação e pesquisa por meio de revisão bibliográfica *ad-hoc;* II) Levantamento dos requisitos estruturais para a criação de um *framework* de desenvolvimento *Web*; III) Implementação do *Framework* ATS e de suas funcionalidades; IV) Distribuição do *Framework* ATS para ser utilizado por desenvolvedores; V) Validação das funcionalidades e performance do *Framework* ATS. RESULTADOS: Foi desenvolvido o *Framework* ATS com o intuito de acelerar o desenvolvimento *Web* em espaços acadêmicos. Também foi validado a dificuldade e as funcionalidades do *Framework* ATS através de um formulário online com desenvolvedores atuantes no mercado de trabalho, a fim de atestar a eficiência do projeto desenvolvido. BENEFICIADOS: Em vista do exposto, espera-se que este projeto possa beneficiar outros desenvolvedores por ter um *framework* destinado a apoiar atividades acadêmicas, bem como as instituições de ensino, que poderão usufruir da automatização de processos pelo advento da tecnologia.

Palavras-chave: *Framework*, Software Acadêmico, Desenvolvimento Web.

ABSTRACT

CONTEXT: Digital technology has become an essential tool for the execution and improvement of functions that were once performed manually, which has enabled changes in society in different areas and spaces. Regarding the academic environment, there is a growing need to insert new strategies for conceiving and organizing different tasks, such as the use of software systems. This proposed automation favors the interactivity between sectors and the use of programs that help in the mechanization of processes within universities. MOTIVATION: In this context, there is a need to build technological innovation mechanisms, made possible by the use of software aimed at the academic environment, which promotes the integration of different sectors and facilitates the execution of bureaucratic/manual processes, through the creation of a *framework* to ensure functionality, speed, stability, and adaptability to the developer. OBJECTIVE: Therefore, this work proposes to develop a *framework* called ATS (Academic Tool Support) to be used as a facilitating tool to implement software systems in order to mechanize processes in academic environments. METHODOLOGY: The construction of this work was outlined from five stages, as follows: I) Rationale and research through ad-hoc literature review; II) Survey of the structural requirements for the creation of a Web development *framework*; III) Implementation of the ATS *Framework* and its functionalities; IV) Distribution of the ATS *Framework* to be used by developers; V) Validation of the functionalities and performance of the ATS *Framework*. RESULTS: The ATS *Framework* was developed with the intention of accelerating Web development in academic spaces. It was also validated the difficulty and the functionalities of the ATS *Framework* through an online form with developers working in the market, in order to certify the efficiency of the developed project. BENEFICIADOS: In view of the above, it is expected that this project can benefit other developers by having a *framework* to support academic activities, as well as educational institutions, which can enjoy the automation of processes by the advent of technology.

Keywords: *Framework*, Academic Software, Web Development.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURA E SIGLAS

API: *Application Programming Interface*

ATS: *Academic Tool Support*

DOM: *Documente Object Model*

GUI: *Graphical User Interface*

HTML: *Hypertext Markup Language*

JSX: JavaScript XML

MVC: *Model-View-Controller*

UI: Interface de Usuário

SUMÁRIO

1 INTRODUÇÃO

A área da educação se tornou campo fértil para a implementação de diversas tecnologias de ensino, aprendizagem e mecanização de processos manuais. Tal evolução acabou contribuindo sobremaneira com um ambiente propício para a aplicabilidade de inovações digitais, principalmente no espaço acadêmico, no qual se faz imprescindível a implementação de tecnologias computacionais. Dessa forma, é possível promover a execução de diversas atividades no escopo educacional de maneira automatizada, com o auxílio de sistemas de software, cada vez mais inseridos nas universidades (MOCBEL *et al.,* 2020).

A tecnologia digital tem se tornado ferramenta essencial para a execução e aprimoramento de funções outrora realizadas de forma manual, o que possibilitou mudanças na sociedade em diferentes âmbitos e espaços (LIMA; LEZANA, 2005). No que concerne ao ambiente acadêmico, observa-se uma crescente necessidade de se utilizar novas estratégias de concepção e organização de diferentes tarefas, tais quais, a inserção de sistemas de software. Tal automatização proposta favorece a interatividade entre setores e a utilização de programas que auxiliem na mecanização de processos dentro das universidades (SOUZA, 2021).

Quando se trata do ambiente acadêmico é possível perceber que existem problemas recorrentes, principalmente diante da falta de interoperabilidade entre a educação e os sistemas de informação. Portanto, é necessário fortalecer os métodos para o desenvolvimento se softwares ágeis, estruturados e adaptáveis no contexto educacional (MEDEIROS, 2019). Isso ocorre, principalmente, porque a maioria destes ambientes é construída sobre uma arquitetura integrada de difícil modificação com interesses educacionais mais gerais ou interesses muito específicos (SILVA; MOREIRA, 2004).

Observa-se, no contexto hodierno das universidades, a implementação de sistemas de software no que se refere aos processos de ensino e aprendizagem, focando na relação entre alunos e docentes. Contudo, existem outros setores dentro do contexto acadêmico que também precisam ser automatizados, com o intuito de transformar processos manuais e burocráticos em software de fácil aplicabilidade e utilização (SIEMENS; BAKER, 2012).

Nesse contexto, a utilização de uma estrutura conhecida como *framework* atende bem às necessidades de produção e utilização de softwares automatizados em contextos acadêmicos. Conceitualmente, o *framework* é uma ferramenta que apresenta códigos estruturais já construídos, que servem para servir de molde a configurações pré-prontas, no qual o

desenvolvedor vai preenchendo as lacunas com as funcionalidades que ele deseja adicionar (BITTENCOURT, 2021).

A palavra *framework* advém do inglês e significa estrutura. Na programação, um *framework* é definido como um conjunto de códigos genéricos que possui a finalidade de unir trechos de um projeto e desenvolvimento, possibilitando ao programador maior produtividade na execução de seu trabalho (NEVES; JÚNIOR, 2020).

Levando o contexto supracitado em consideração, a produtividade é uma caraterística bastante valorizada hoje, tendo em vista que a tecnologia disponível e o mercado de trabalho exigem maior agilidade no processo de desenvolvimento, pois o resultado relaciona-se diretamente aos insumos necessários à sua produção, bem como ao tempo investido no processo. Por esta razão, tecnologias como o *framework* potencializam a criação de ferramentas que aceleram a construção de softwares mais completos e robustos (LIMA, 2019).

De acordo com Rosales (2014, p.24), o *framework* fornece "*modelos de contextos e raciocínio genéricos e extensíveis que foram desenvolvidos com o propósito de facilitar a análise de dados que, frequentemente, é realizada pelas aplicações".* Nesse sentido, a criação de framework possibilita a implementação de ferramentas unificadas, que convergem entre si para o propósito estabelecido previamente pelo desenvolvedor, favorecendo a reutilização do código e sua extensa aplicabilidade e funcionalidade em diferentes sistemas de software acadêmico.

A relevância deste projeto consiste em promover mudanças no ambiente acadêmico por meio da inserção da tecnologia no cotidiano, objetivando auxiliar desenvolvedores através da criação de um framework que dê celeridade ao processo de desenvolvimento de software educacionais, garantindo estabilidade ao projeto proposto.

1.1 Justificativa

Diante do exposto, observa-se a necessidade de se construir mecanismos de inovação tecnológica, possível pela utilização de software voltado para o âmbito acadêmico, que promova integração de diferentes setores e facilite a execução de processos burocráticos/manuais, por meio da criação de um framework a fim de garantir funcionalidade, rapidez, estabilidade e disponibilidade de adaptação ao desenvolvedor. Nesse sentido, a realização deste trabalho justifica-se na observação do cotidiano acadêmico, no qual percebeu-se a realização de diversas atividades de forma manual, como a reserva de espaços de acesso às salas e laboratórios por choque de horários ou desorganização na forma de proceder as reservas

necessárias às aulas práticas em um campus de uma universidade no interior do Estado do Ceará.

1.2 Objetivos

Nesta seção, são descritos os objetivos gerais e específicos que nortearam a construção do *Framework* ATS e sua validação a partir da percepção de desenvolvedores de software.

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um *framework* intitulado ATS (*Academic Tool Support*) a ser utilizado como ferramenta facilitadora para implementar sistemas de software, a fim de mecanizar processos em ambientes acadêmicos.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho consistem em:

- Descrever conceitos e concepções associados ao *framework* sob a ótica da literatura;
- Associar, de forma descritiva, o processo de criação de um *framework* e sua funcionalidade para *software* acadêmico;
- Projetar a arquitetura do *framework* proposto;
- Implementar o *framework* proposto;
- Levantar requisitos de um *software* para prova de conceito do *framework* proposto;
- Implementar a validação do *Framework* ATS por meio da aplicação de um formulário;
- Realizar experimentos usando o *Framework* ATS com um grupo de desenvolvedores a fim de medir usabilidade e determinar a funcionalidade e performance.

1.3 Organização do Trabalho

A organização deste trabalho se baseia na divisão e apresentação de capítulos enumerados, a saber: 2, 3, 4, 5 e 6. No Capítulo 2 é apresentada a Fundamentação Teórica que possibilitou a formulação do *Framework* ATS. Este capítulo foi dividido em subtópicos, a fim de detalhar melhor a tecnologia empregada para organizar o fluxo de informações contidas no *Framework* ATS. O Capítulo 3 aborda a metodologia escolhida para a realização deste estudo,

sendo descrita de forma mais detalhada. O Capítulo 4 contém a estrutura criada para a formulação do *framework*, desde a seleção das bibliotecas, da linguagem de programação, padronização do código e arquitetura de pastas. O Capítulo 5 compreende os resultados e discussões gerados a partir da implementação e da validação do *Framework* ATS por meio da aplicação de um formulário com desenvolvedores Web. Por fim, o Capítulo 6 apresenta as considerações finais, incluindo contribuições, limitações identificadas e sugestão de propostas de continuidade para esta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo fornece embasamento aos assuntos relacionados ao contexto teórico envolvido na proposta deste trabalho. Desse modo, tem como objetivo elucidar os seguintes tópicos: Conceitos sobre *framework* (Seção 2.1); Visão geral do funcionamento dos *frameworks* (Seção 2.2); Utilização e aplicabilidade dos *frameworks* (Seção 2.3); *Frameworks* em aplicações educacionais (Seção 2.4); Arquitetura *Model-View*-Controller – MVC (Seção 2.5); React Js (Seção 2.6).

2.1 Conceitos sobre *framework*

Os *framework*s são estruturas criadas para reduzir custos de desenvolvimento, pois são ferramentas que aceleram sobremaneira o tempo de produção devido à técnica de reutilização tanto do código-fonte quanto de análise de projeto (GUPTA, 2018).

Um *framework* é uma estrutura que serve como base para a construção de aplicações Web, sendo possível construir sites, aplicativos e *softwares* a partir de um esqueleto pré-definido, no qual se altera algumas particularidades (KRIGER, 2022).

Hoje existem diversos *framework*s que podem ser utilizados, os quais se diferenciam em alguns aspectos essenciais: linguagem de programação utilizada, arquitetura implementada, modos de gerar arquivos executáveis e estratégia de reutilização de artefatos e ativos (XANTHOPOULOS; XINOGALOS, 2013).

A escolha do *framework* certo para o projeto deve levar em consideração a experiência e conhecimento prévio dos desenvolvedores sobre a linguagem de programação, a curva de aprendizagem caso não haja conhecimento da linguagem, que tipo de projeto será desenvolvido, quais funcionalidades estarão atreladas, bem como outros fatores específicos (LIMA, 2019).

Os *framework*s apresentam alguns atributos que os caracterizam qualitativamente, sendo estes analisados pelos desenvolvedores como critérios de escolha para sua seleção e utilização para os projetos de desenvolvimento *Web*. Para compreender a estrutura de um *framework*, é importante saber a que se refere cada atributo. São eles: Linguagem de Programação, Curva de Aprendizagem, Reutilização e Manutenção (LIMA, 2019).

A Linguagem de Programação é uma das principais características de um *framework*. Pode ser definida como um idioma com regras de escrita que garante a comunicação com o computador. Estruturalmente, elas podem ser divididas em linguagens de alto nível (permitem que os códigos escritos com ela sejam mais simples, permitindo que outros desenvolvedores possam ler, compreender, editar, comentar, compartilhar ou revisar os códigos escritos nessa linguagem); linguagens de baixo nível (por estarem mais próximas da linguagem da máquina, são mais eficientes em termos de desempenho, pois são realizadas menos conversões e as instruções são direcionadas de uma forma mais próxima do hardware) (SANTANA, 2023).

A Curva de Aprendizagem é um conceito idealizado por Hermann Ebbinghaus, em 1885, e se refere à relação entre a proficiência em determinada atividade em função do tempo. Ou seja, a proficiência relaciona-se diretamente à redução nos custos do produto desenvolvido, beneficiando o gerenciamento de tempo (COUTINHO, 2021). Nesse sentido, a curva de aprendizagem serve como parâmetro indicador de desempenho, sendo um atributo importante a ser levado em consideração na escolha do *framework*.

A Reutilização pode ser entendida como um processo de construir sistemas a partir de software ou artefato de software existente, não necessitando construí-lo do absoluto zero. Pode ser compreendida também como a capacidade dos *frameworks* de criar grupamentos de códigos, criar repositórios para estes grupamentos e utilizá-los de forma genérica em vários projetos (BOTELLA *et al.,* 2016).

A Manutenção de um *framework* pode ser medida pela facilidade de os desenvolvedores o manterem atualizado com as tecnologias do mercado e a possibilidade de torná-lo o mais longevo possível, por meio de sua refatoração constante.

As vantagens em se utilizar um *framework* são baseadas, principalmente, no aumento da produtividade, pois os desenvolvedores podem construir seus projetos com mais rapidez e fluidez; o código sem erros é outro fator vantajoso, tendo em vista que os criadores do *framework* já executaram a função de identificar e resolver as falhas.

A fácil manutenção, como supracitado, é outra qualidade quando se opta pela utilização de um *framework*, pois a padronização do código facilita o processo de manutenção. A redução

de custos e de tempo também tornam justificáveis o uso de *frameworks* no desenvolvimento *Web*, bem como a maior segurança devido a menor chance de ocorrer erros de execução (KRIGER, 2022).

2.2 Visão geral sobre o funcionamento dos *frameworks*

Um *framework* funciona semelhantemente a um *template* ou modelo que, quando utilizado, oferece certos recursos e elementos estruturais básicos para a composição de alguma aplicação ou *software*, através do uso de um conjunto de ferramentas de programação, tais quais: o código fonte, compiladores, bibliotecas, classes abstratas, APIs (*Application Programming Interface)*, dentre outros, que oferecem o suporte necessário para a programação de *softwares* no geral (BORGES, 2023).

Ainda de acordo com Borges (2023), existem diversos tipos de *frameworks*, podendo ser classificados tanto pelas suas aplicações quanto pela forma de implementação. Os *frameworks* para aplicação *Web* são aqueles que fornecem suporte para o desenvolvimento de sistemas online e demais recursos disponíveis na internet. Já os *frameworks* para Desenvolvimento Mobile guiam a programação associada à criação de aplicativos e serviços disponíveis para dispositivos móveis, podendo ser em ambientes nativos, híbridos ou multiplataforma.

A forma de funcionamento de um *framework* se baseia em suas funções, que possuem uma grande variedade de parâmetros, o que proporciona ao desenvolvedor a disponibilidade de realizar personalizações a partir das necessidades do projeto que está sendo trabalhado (NOLETO, 2020).

Os *frameworks front-end* são mais voltados para auxiliar o desenvolvimento das partes visuais de um site ou plataforma, ou seja, a parte em que ocorre a interação com o usuário. Em contrapartida, os *frameworks back-end* estão associados à resolutividade de tarefas específicas de funções que não são visíveis pelo usuário (HOSTGATOR, 2023).

2.3 Utilização e Aplicabilidade dos *Frameworks*

Para se pensar na utilização e aplicabilidade dos *frameworks,* é importante primeiro compreender como ocorre sua estruturação e desenvolvimento. De acordo com Silva (2006, p.41), as seguintes fases são identificadas durante a construção de um sistema baseado em *frameworks:*

- Desenvolvimento do *framework:* Essa fase que consome maior tempo do profissional desenvolvedor de software, pois tem como objetivo produzir um produto reutilizável dentro de um domínio que suporte a implementação do projeto estruturado.
- Uso do *framework:* Também denominada fase de instanciação ou fase de desenvolvimento de aplicações, corresponde à instanciação do *framework* para sistemas específicos.
- Evolução e manutenção do *framework:* Evolução do projeto a fim de incorporar novas funcionalidades e especificidades ou modificação para posterior correção de problemas que surjam a partir da implementação e utilização do *framework.*

Algumas fontes encontradas para formular a fundamentação teórica são relativamente antigas, contudo, alguns teóricos são pioneiros em conceitos referentes aos *frameworks* não podendo se desconsiderar a contribuição de tais estudiosos. Dito isto, importante mencionar a contribuição de Bosch et al. (1999), que dividiu a fase de desenvolvimento de um *framework* em seis etapas bem delimitadas, a saber: I) Análise de domínio; II) Modelo de domínio, com requisitos e conceitos; III) Projeto Arquitetural de alto nível do *framework*; IV) Atividade de implementação, utilizando-se uma linguagem de programação específica; V) Atividade de teste, a fim de avaliar tanto a funcionalidade quanto a usabilidade do *framework*; VI) Documentação do *framework,* contendo o manual do usuário, a documentação do projeto e a descrição de como o *framework* funciona.

De acordo com o Developer Survey 2022, relatório anual elaborado pela Stack Overflow, um site de perguntas e respostas para profissionais e entusiastas na área de programação de computadores, no que tange à *frameworks* para aplicações *Web* os mais utilizados no ano de 2022 foram Node.js[1], React.js[2], jQuery[3], Express[4] e Angular[5], nesta ordem.

O Node.js é um ambiente de execução do código JavaScript do lado servidor (*server side),* que na prática se reflete na possibilidade de criar aplicações *standalone* (autossuficientes) em uma máquina servidora, sem a necessidade de navegador.

O React.js é uma biblioteca de *front-end*, possui uma estrutura em JavaScript, é de código aberto e foi criado pelo *Facebook.*

---

[1] Node.js (nodejs.org)

[2] https://react.dev/

[3] https://jquery.com/

[4] https://expressjs.com

[5] https://angular.io/

É usado para construir interfaces de usuário, facilita o processo de atualização e sincronização de atividades simultâneas, permite criar aplicações de grande porte para diversas finalidades na *Web*, oferecendo flexibilidade para fazer alterações.

O *jQuery* também é uma tecnologia de código aberto. Permite a criação de scripts que otimizam a experiência dos usuários e proporcionam mais interatividade no desenvolvimento de páginas *Web*. No entanto, a ideia principal é fazer com que os códigos sejam incrementados sem que se tornem mais complexos, já que a biblioteca possibilita esta simplificação.

O *Express* é baseado em estruturas mínimas, rápidas e que não são postuladas, além de suportar API *Rest*. É leve, escrito em JavaScript, escalável e possui cache integrado. Fornece ainda instrumentos como roteamento, tratamento de erros, depuração e integrações de banco de dados.

O *framework* Angular é de codificação *front-end* baseado em JavaScript. Conta com uma rápida produção de códigos, possui diversas ferramentas e soluções voltadas à aceleração do desenvolvimento *Web* e combinação de *templates.*

2.4 *Frameworks* em aplicações educacionais

O uso de *framework*s tem forte importância comercial e essa forma de desenvolvimento de aplicações também pode ser utilizada em ambientes educacionais, os quais necessitam da adequação de tecnologias digitais para suprir suas demandas de organização e relação entre discentes, docentes e corpo administrativo.

Nesse sentido, de acordo com Silva (2006, p.10) *"muitos dispositivos voltados para os sistemas educacionais são utilizados para auxiliar no gerenciamento de atividades propostas e realizadas pelas entidades envolvidas no processo educacional (...) e para armazenar e organizar materiais didáticos"*. Compreende-se, pela perspectiva do autor, que a integração de tecnologias aos ambientes educacionais já é uma realidade experenciada a algumas décadas, principalmente no que concerne à educação e tecnologia no âmbito acadêmico.

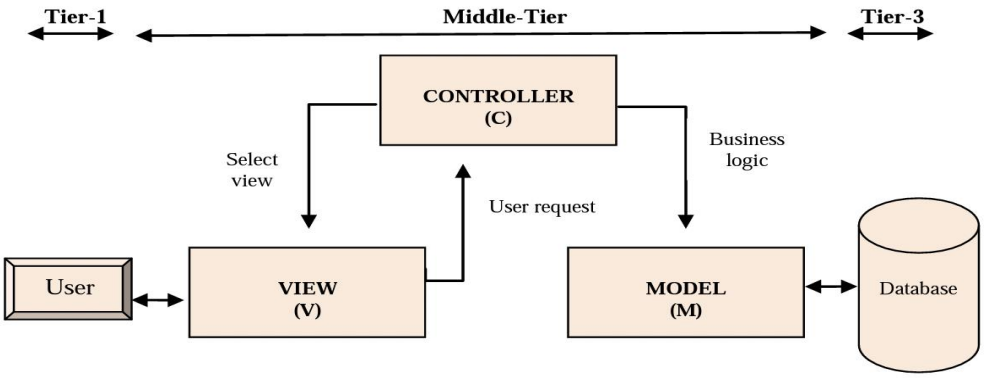
2.5 Arquitetura *Model-View-Controller* (MVC)

É um conceito associado à arquitetura de *software* considerado um padrão de arquitetura no que se refere à engenharia de *software*. É composto por três componentes: *Model, View* e *Controller* (MVC). Em uma estrutura MVC, a visualização e o controlador fazem parte da interface do usuário. Inicialmente, o usuário envia uma solicitação a um controlador através de

uma Interface Gráfica do Usuário (GUI - *Graphical User Interface*). Em seguida, o modelo de acesso do controlador fornece os dados de acordo com a solicitação do usuário. Depois disso, o modelo retorna os dados para o controlador e o controlador apresenta os dados por meio da exibição especificada. Também se utiliza de banco de dados para armazenar dados e fornecer fonte de dados original no sistema (ADAM; ANDOLO, 2019)

Os componentes são resumidos da seguinte maneira: *Model:* Possui a função de gerenciar e controlar a maneira como os dados se comportam por meio das funções, lógicas e regras pré-configuradas, além de deter os dados que recebe as informações do *Controller.* Representa um objeto ou JAVA transportando dados. Ele também pode ter lógica para atualizar o controlador se seus dados mudarem. *View:* representa a visualização dos dados que o modelo contém. Essa camada apresenta as informações visualmente ao usuário, possuindo apenas recursos ligados à aparência, tais como mensagens, telas e botões. *Controller*: atua tanto no *Model* quanto na *View*. Ele controla o fluxo de dados no objeto do modelo e atualiza a exibição sempre que os dados são alterados. Ele mantém *View* e *Model* separados (SARKER; ABU, 2014). A Figura 1 esquematiza visualmente a relação entre os componentes da arquitetura MVC, demonstrando o fluxo de dados, no qual o usuário interage com uma *View*, que faz requisições para o *Controller*. A resposta do *Controller* utiliza a lógica de negócio que está presente no *Model* e os dados dessas lógicas estão salvos no banco de dados.

Figura 1: Relacionamento entre os componentes da arquitetura MVC



Fonte: Sarker; Abu, 2014.

O MVC é um padrão de arquitetura de *software* que contribui na otimização da velocidade entre as requisições feitas pelo comando dos usuários. Possui diversos benefícios que justificam sua utilização no processo de desenvolvimento, tais como (ZUCHER, 2020):

- Segurança: O *controller* funciona como uma espécie de filtro capaz de impedir que qualquer dado incorreto chegue até a camada modelo.
- Organização: Esse método de programação permite que um novo desenvolvedor tenha muito mais facilidade em entender o que foi construído, assim como os erros se tornam mais fácil de serem encontrados e corrigidos.
- Eficiência: Como a arquitetura de *software* é dividida em 3 componentes, sua aplicação fica muito mais leve, permitindo que vários desenvolvedores trabalhem no projeto de forma independente.
- Tempo: Com a dinâmica facilitada pela colaboração entre os profissionais de desenvolvimento, o projeto pode ser concluído com muito mais rapidez, tornando o projeto escalável.
- Transformação: As mudanças que forem necessárias também são mais fluidas, já que não será essencial trabalhar nas regras de negócio e correção de *bugs*.

2.6 *React*

O *React Js* é uma biblioteca JavaScript implantada para desenvolver componentes reutilizáveis da interface do usuário (UI). Nesse sentido, o *React* permite o desenvolvimento de aplicativos grandes e complexos baseados na *Web*, que podem alterar seus dados sem atualizações de páginas subsequentes. É usado como *View* (V) no *Model-View-Controller* (MVC). O *React* abstrai o *Document Object Model* (DOM), oferecendo assim uma experiência de desenvolvimento de aplicativos simples, eficiente e robusta. O *React* implementa o fluxo de dados unidirecional, diminuindo assim o *boilerplate* e, portanto, prova ser muito mais fácil do que a vinculação de dados tradicionais (AGGARWAL *et al.,* 2018).

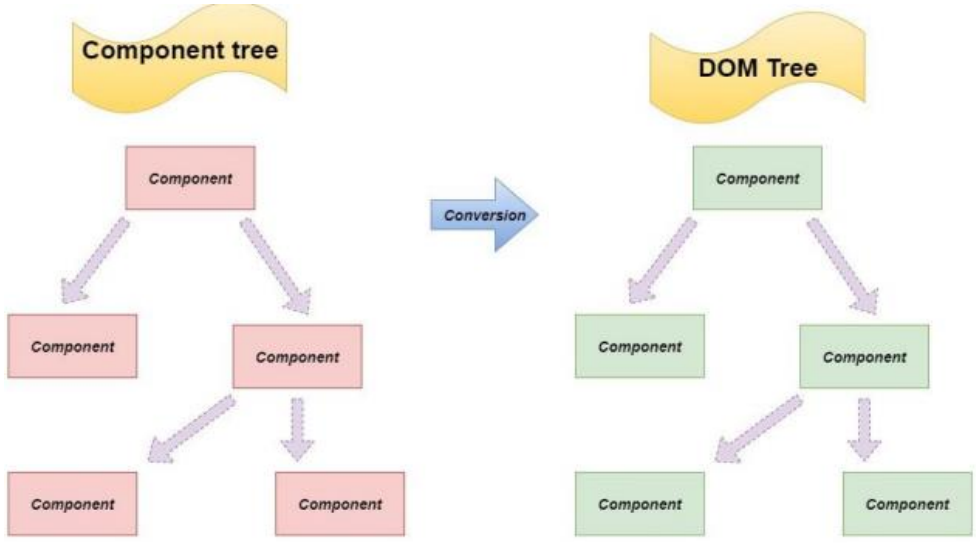
A utilização do *React* baseia-se em quatro pontos principais, que podem ser delimitados da seguinte maneira, a saber (AGGARWAL *et al.,* 2018):

- I) DOM leve (modelo de objeto de documento) para melhor desempenho: a natureza fácil e não complexa do *React Js* permite que alguém se sinta rapidamente confortável com o *framework*. A curva de aprendizagem e a arquitetura são extremamente fáceis para o desenvolvedor.
- II) Curva de aprendizagem fácil: *React* fornece um modelo de objeto de documento muito eficiente e leve. Ele não interage com o DOM gerado pelo navegador, mas reage,

ao modelo de objeto do documento armazenado na memória. Isso resulta em um desempenho incrivelmente rápido e robusto do aplicativo.

- III) JSX: é uma linguagem muito semelhante ao gigante da tecnologia XML. Não é obrigatório usar JSX ao desenvolver um aplicativo baseado em reação, mas é muito popular entre os desenvolvedores, pois facilita o desenvolvimento sempre que eles escrevem marcações para componentes e os eventos de ligação correspondentes.
- IV) Desempenho: a razão para o desempenho altamente eficiente é essencialmente o recurso DOM virtual da estrutura. O *React JS* mantém um modelo de objeto de documento virtual dentro da memória. Sempre que uma alteração deve ser refletida na página da *Web* exibida no momento, em vez de atualizar instantaneamente o DOM de navegação, as primeiras alterações no DOM virtual são feitas. Depois que as alterações no DOM virtual são feitas, um algoritmo diff() é aplicado que compara o reboque, o DOM virtual e o DOM de navegação e apenas os nós relevantes e desejados da árvore DOM do navegador são atualizados, o que resulta em um desempenho extremamente rápido do aplicativo. A Figura 2 representa a manipulação da DOM feita pela biblioteca, onde a árvore de componentes, vista na esquerda da figura, é traduzida para a Árvore DOM tradicional do HTML, como pode ser visto à direita.
-

Figura 2: Conversão de DOM intermediário para HTML DOM



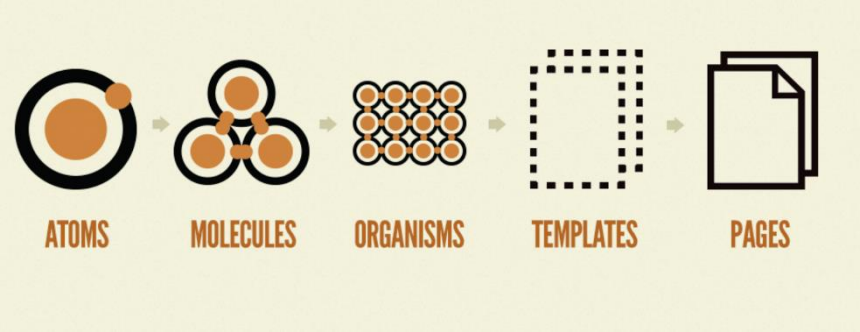
Fonte: Aggarwal *et al.,* 2018.

Uma biblioteca muito importante é a *Zustand*[6], a qual gerencia estados no *react* através de *hooks,* que são definidos como uma nova adição no *React* 16.8 e permitem que o desenvolvedor use o estado e outros recursos do *React* sem escrever uma classe, fornecendo para toda a aplicação estados que serão utilizados para a renderização de toda a interface.

Foi feito um sistema de despachos e ações em que os componentes JSX fizeram despachos para a biblioteca que fez ações baseadas nesses despachos podendo assim atualizar dados, deletá-los, fazer requisições de API e, a depender do resultado dessa requisição, fazer ainda mais modificações.

Também foi adotado o design atômico que é uma metodologia para criar sistemas de design. Essa metodologia separa componentes em cinco partes: Átomos, Moléculas, Organismos, *Templates* e Páginas. Essas partes vão se aglomerando e formando uns aos outros, onde, por exemplo, tem-se um átomo de tipografia e um de botão. Esses átomos juntos podem formar uma molécula de *input*. A Figura 3 demonstra a proposta do design atômico e seu fluxo de componentes, onde átomos compõem moléculas, que juntas constroem organismos, que, por sua vez, constroem *templates*, e esses, por fim, constroem páginas inteiras.

Figura 3: Design Atômico



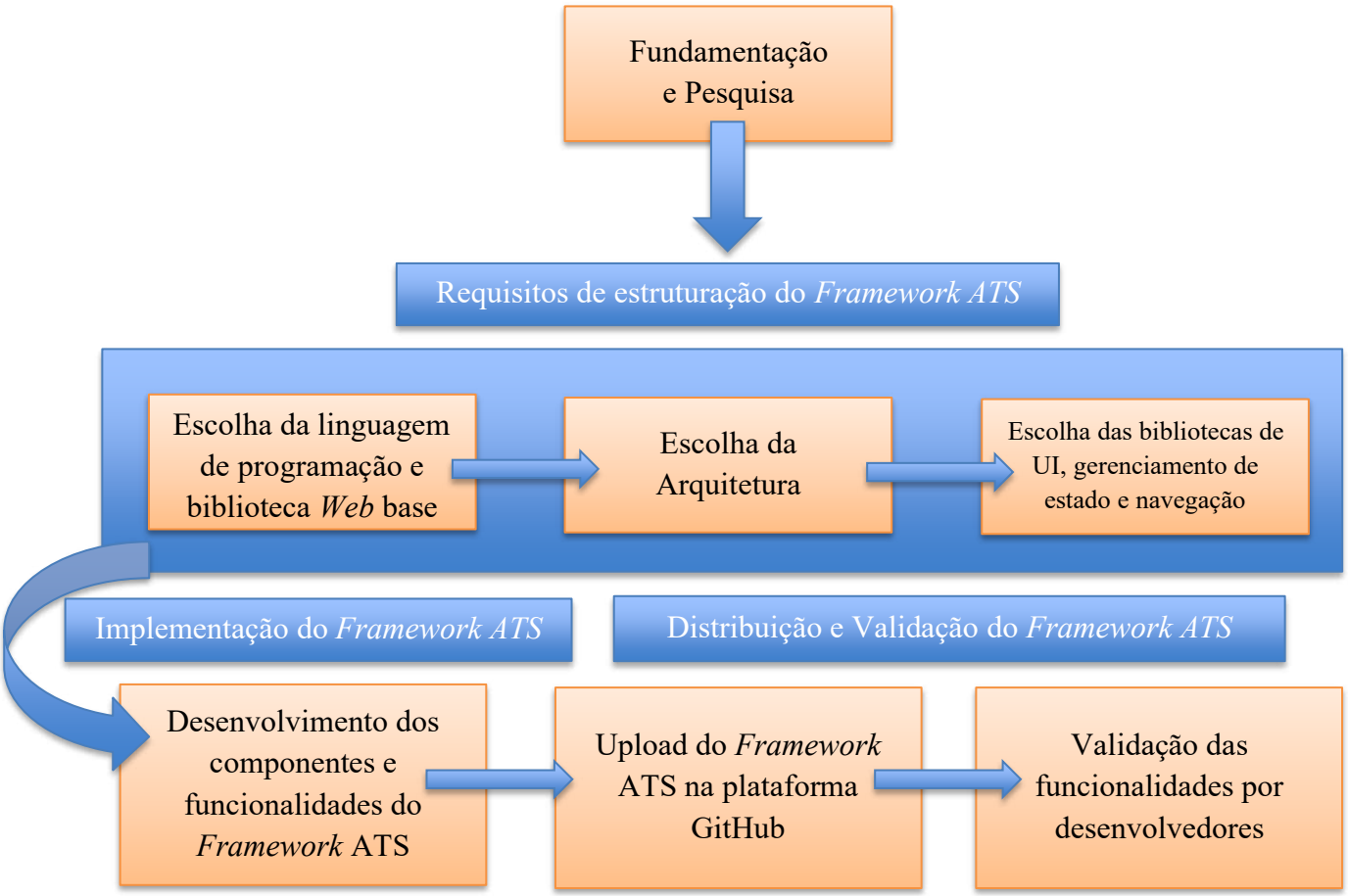
Fonte: Frost, 2013.

[6] https://docs.pmnd.rs/zustand/recipes/recipes

3 METODOLOGIA

Neste capítulo será abordado a metodologia utilizada para a realização deste estudo, que visa fomentar a construção do *Framework* ATS, a ser utilizado como ferramenta facilitadora para implementar sistemas de software, a fim de mecanizar processos em ambientes acadêmicos. Para melhor compreensão da metodologia proposta, serão detalhadas as etapas a seguir: Fundamentação e Pesquisa; Requisitos de estruturação do *Framework* ATS; Implementação do *Framework* ATS; Distribuição e Validação do *Framework* ATS; Resultados e trabalhos futuros. A Figura 4 abrange visualmente a estrutura metodológica realizada bem como destrincha melhor as seções abordadas nos próximos capítulos.

Figura 4: Estrutura metodológica do trabalho



Fonte: Autor, 2023.

3.1 Abordagem e Tipo de Pesquisa

A abordagem de pesquisa realizada neste projeto é qualitativa do tipo exploratório-descritiva por meio da criação de um *framework* a ser implementado em software acadêmico. FNa pesquisa qualitativa, o objetivo do pesquisador está concentrado em caracterizar um agrupamento de informações específicas, de maneira subjetiva, investigando mais profundamente os grupos e seguimentos delimitados e focalizados, por meio de histórias sociais sob a ótica de atores e em relações dos sujeitos envolvidos (MINAYO, 2014).

De acordo com Gil (2017), na pesquisa descritiva o pesquisador atua de forma imparcial, não causando nenhuma interferência em relação aos dados coletados, realizando um estudo a partir da análise e interpretação de fatos, com o intuito de fazer uma observação impessoal, registrando e analisando os fatos selecionados.

3.2 Método de Pesquisa e Procedimentos para construção do *Framework* ATS

A construção deste trabalho se delineou a partir de cinco etapas, a saber:

I) Fundamentação e pesquisa por meio de revisão bibliográfica *ad-hoc*[7];
II) Levantamento dos requisitos estruturais para a criação de um *framework* de desenvolvimento *Web*;
III) Implementação do *Framework* ATS e de suas funcionalidades;
IV) Distribuição do *Framework* ATS para ser utilizado por desenvolvedores;
V) Validação das funcionalidades e performance do *Framework* ATS.

3.3 Fundamentação e Pesquisa

Foi feita uma revisão bibliográfica *ad-hoc*, por meio da busca na literatura sobre os temas propostos, possibilitando a construção de um referencial teórico pertinente e aprofundado sobre o desenvolvimento de um *Framework Web*, seus principais pontos estruturais, seus conceitos e funcionalidades. Abordou-se ainda sobre a arquitetura selecionada para a construção do *Framework* ATS, bem como da escolha pelas bibliotecas de UI (*React Js*).

[7] *ad-hoc:* Termo proveniente do latim que significa “para esta finalidade”.

3.4 Levantamento dos requisitos estruturais

Foram avaliados os principais requisitos estruturais para a criação de um *framework* de desenvolvimento *Web*, dos quais o requisito inicial foi a partir da escolha da biblioteca de desenvolvimento *Web* base, sendo selecionada a biblioteca *React Js*, construída com as linguagens JavaScript e TypeScript, que fornecem aos desenvolvedores inúmeras funcionalidades embutidas, além de ser altamente flexível, ter ótima experiência de desenvolvimento, ter uma grande comunidade e manutenção, ótima performance e ser facilmente testável.

Acerca da utilização do *React Js* para a implementação do ATS, todo o lado do cliente foi feito em cima dessa biblioteca, em razão dos benefícios mencionados anteriormente. A interface dos componentes primários reutilizáveis e das páginas prontas se baseou no ciclo de vida e dos componentes oferecidos pelo *React* para serem feitos de forma otimizada e leve. É importante mencionar que como o *React Js* é uma biblioteca muito difundida e utilizada no mercado, inúmeras outras bibliotecas foram criadas para resolver problemas e facilitar o trabalho dos desenvolvedores que a utilizam.

Com a escolha da biblioteca base, também foi necessária a escolha da linguagem de programação. O *React* dá ao desenvolvedor a possibilidade de uso do JavaScript e do TypeScript. Para o *Framework* ATS foi utilizada a linguagem TypeScript, pois ela oferece uma tipagem aos dados e métodos utilizados no código, o que permite uma função de *autocomplete* nos editores de texto e evita muitos erros na hora de desenvolver o projeto.

Quanto à arquitetura do projeto, foi selecionada a arquitetura MVC, onde seu principal ponto de escolha se justifica por ser uma arquitetura de desenvolvimento rápido. Como ela isola as funcionalidades em três níveis (*Model, View* e *Controller*) é muito simples de se trabalhar em um só componente isolado, acelerando o desenvolvimento. Além disso, este isolamento permite que, caso sejam necessárias modificações em partes do código, essas mudanças não afetarão outras partes.

Em relação à implementação do ATS, a utilização do MVC fez parte da arquitetura do cliente-servidor, onde no *React* foram criadas *Views* que foram disponibilizadas prontas em *templates* já disponíveis (telas que sempre são utilizadas, como telas de login, cadastro de usuário, esqueci minha senha, telas de suporte, chat, etc) como nas telas criadas pelo desenvolvedor. Essas *Views* fizeram requisições para o *Controller* que possui todas as regras de negócio e passa para o *Model* que se comunica com o banco e retorna apenas o que foi requerido pelo usuário.

Após as escolhas estruturais mais significativas, previamente mencionadas, foram escolhidas bibliotecas de funcionalidades do *Framework* ATS, como a Chakra UI[8], que possui a finalidade de fornecer componentes de UI pré-configurados para o *Framework* ATS e, com isso, acelerar bastante o desenvolvimento.

Vale ressaltar que, apesar de os componentes serem fornecidos pela biblioteca, eles são altamente customizáveis, tanto em estilização quanto em controle de lógicas e estados, dando maior flexibilidade ao desenvolvedor.

Um componente bastante complexo em todas as aplicações é o de formulário. Para o controle deste componente no *Framework* ATS, foram usadas duas bibliotecas em conjunto: a *React Hook form*[9] e a *Yup*[10]. Essas duas bibliotecas disponibilizam o acesso aos estados do formulário, validação e erros.

Outro ponto muito importante foi a escolha de biblioteca de controle de estado na aplicação. Foi escolhida a biblioteca Zustand[11] por um motivo em específico: Facilidade. Apesar de ser uma biblioteca robusta que traz muita confiança e funcionalidades ao desenvolvedor, a Zustand é simples de implementar e se assemelha bastante com a funcionalidade de contexto provida pela *React Js.*

Para gerir a navegação entre rotas do projeto foi escolhida a biblioteca *react-router*[12] que é bastante difundida e robusta, fornecendo acesso a todo o ciclo de vida das navegações e possibilita o envio de parâmetros na troca de rotas. Apesar do *Framework* ATS não se comunicar com nenhuma API, foi adicionada a biblioteca Axios[13] para caso de necessidades futuras.

3.5 Implementação do *Framework* ATS

Após escolhidas as bibliotecas, linguagens de programação e arquitetura, o projeto foi iniciado através da ferramenta Vite[14], que inicializa o projeto *React Js* com configurações prontas para uso da linguagem TypeScript e faz a compilação do projeto.

[8] https://chakra-ui.com
[9] https://react-hook-form.com
[10] https://github.com/jquense/yup
[11] https://docs.pmnd.rs/zustand/recipes/recipes
[12] https://reactrouter.com/en/main
[13] https://axios-http.com/docs/intro
[14] https://vitejs.dev

O desenvolvimento se deu primariamente com a construção dos componentes de UI, sendo criados componentes de texto, imagens, *input, link, switch,* botões, *selects,* áreas de texto, dentre outros. Após feita a estilização base dos componentes, seu controle de estado interno e sua organização de pastas respeitando o *Atomic Design,* foi criada a estrutura de navegação, com a rota inicial sendo a tela de Login.

A tela de Login foi construída com um formulário, controlado pelas bibliotecas de controle de formulário citadas acima. Para uma futura comunicação com uma API de autenticação na tela de login, foi criada uma função de autenticação usando a biblioteca Zustand, que guarda um status simulado da autenticação, informando se ela foi um sucesso ou não.

A fim de garantir a estabilidade e baixa manutenção, o *Framework* ATS decidiu adotar uma filosofia minimalista, provendo componentes e funcionalidades base para a criação da majoritária parte dos projetos de desenvolvimento *Web*, no qual o desenvolvedor terá muita flexibilidade no uso do software, se utilizando dos componentes e funcionalidades previamente integrados. Também vale salientar que todo o desenvolvimento do código fonte foi feito em inglês, tendo em vista que é uma linguagem de maior alcance do que o português, possibilitando o maior uso do *Framework* ATS.

3.6 Distribuição do *Framework* ATS

Após finalizado o desenvolvimento do *Framework* ATS, o código fonte foi disponibilizado na plataforma GitHub, onde qualquer desenvolvedor pode baixá-lo e utilizá-lo para quaisquer projetos que achar útil.

3.7 Validação do *Framework* ATS

O método de validação do *Framework* ATS ocorreu através da aplicação de um formulário online onde nove participantes, todos desenvolvedores atuantes no mercado de trabalho, responderam perguntas acerca das dificuldades de instalação, curva de aprendizado, leitura de código, bem como citaram pontos positivos e negativos acerca do *Framework* ATS. Para prover aos participantes um conhecimento parecido, foram também feitas três ordens de execução no formulário onde os respondentes teriam que lidar com funcionalidades chaves do *Framework* ATS, a fim de testar aplicabilidade e funcionalidade.

4 ESTRUTURA DO *FRAMEWORK* ATS

Neste capítulo está descrita toda a estrutura selecionada para a construção do *Framework* ATS, desde as etapas iniciais de seleção das ferramentas e bibliotecas. Foi ainda descrito em detalhes como ocorreu a padronização do código e a instalação de bibliotecas para que o usuário compreenda o motivo da seleção de componentes da estrutura do *Framework* ATS. Por fim, orienta-se sobre a utilização dos elementos que foram criados para o *Framework* ATS e como pode ser feita a instalação para uso.

4.1 Construção inicial do projeto

O projeto iniciou com a instalação da ferramenta Vite, mencionada no capítulo anterior. Segundo as definições da própria plataforma, consiste em uma ferramenta de construção que se destina a oferecer uma experiência de desenvolvimento mais rápida e leve para projetos de Web modernos. É dividido em duas partes:

- Um servidor de desenvolvimento que oferece melhorias de funcionalidades para os módulos de ECMAScript nativo;
- Um comando de construção pré-configurado para otimizar recursos estáticos que empacotam o código para compilação.

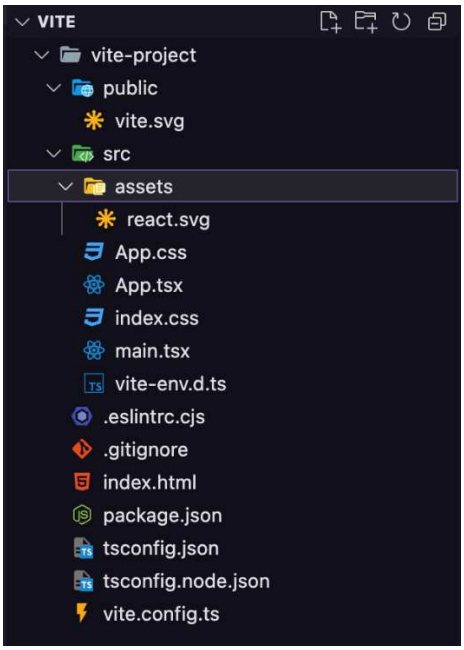
A Figura 5 representa a execução do comando para a criação do projeto através do gerenciador de pacotes npm:

Figura 5: Comando de inicialização de um projeto com a ferramenta Vite

Fonte: Autor (2023).

Após a instalação da ferramenta Vite, o projeto em *React* já está pré-configurado para sua execução. Foi utilizado o modelo *React* com TypeScript fornecido pela ferramenta de desenvolvimento. A Figura 6 mostra a estrutura de pastas formada por um projeto criado com a ferramenta Vite, onde pode-se perceber que os arquivos utilizados pelo desenvolvedor, em sua maioria, se encontram na pasta src.

Figura 6: Estrutura de Pastas criadas na ferramenta Vite



Fonte: Autor (2023).

Também são providos diversos arquivos de configuração. A configuração da linguagem TypeScript, que por padrão é complexa e demanda tempo, é provida ao desenvolvedor pela ferramenta, como pode ser evidenciado na Figura 7, mostrada abaixo. Pode-se ver, ainda, as opções do compilador TypeScript no arquivo, versões alvo e como lidar com diversos tipos de extensão de arquivos, como JSX (JavaScript XML).

Figura 7: Arquivo de configuração da linguagem TypeScript

```
{
  "compilerOptions": {
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx",

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["src"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

Fonte: Autor (2023).

Foi escolhida a linguagem TypeScript, que pode ser definida como um superconjunto sintático da linguagem Javascript que adiciona tipagem estática. O TypeScript, diferentemente do Javascript, permite especificar os tipos de dados que estão sendo transmitidos no código e tem a capacidade de relatar erros quando os tipos não correspondem.

4.2 Padronização do Código

Para garantir menor complexidade ao desenvolvedor que utiliza o Framework, a legibilidade do código-fonte necessita da existência de uma padronização de código no desenvolvimento. Neste trabalho, a padronização foi alcançada por meio da junção de duas ferramentas: ESLint[15] e Prettier[16].

Com o ESLint, é possível definir regras de padronização de código, achar os códigos que fogem da padronização previamente estipulada e identificar erros que serão passados para a ferramenta Prettier, que, por sua vez, irá consertar os erros de forma automática. Pode-se observar na Figura 8 o arquivo de configuração do ESLint usado no *Framework* ATS, onde tem-se variáveis como o padrão de análise que será utilizada, os plugins adicionados como extensão da configuração e pode-se definir e excluir regras que são padronizadas pela ferramenta.

Figura 8: Arquivo de configuração do ESLint

```
{
  "globals": {
    "window": false
  },
  "parser": "@typescript-eslint/parser",
  "env": {
    "node": true,
    "browser": true
  },
  "extends": [
    "plugin:@typescript-eslint/recommended",
    "eslint:recommended",
    "plugin:prettier/recommended"
  ],
  "plugins": ["@typescript-eslint"],
  "rules": {
  }
}
```

Fonte: Autor (2023).

[15] https://eslint.org

[16] https://prettier.io

4.3 Instalação de Bibliotecas

Biblioteca é uma coleção de subprogramas usados no desenvolvimento de softwares. A sua principal função é facilitar a programação, garantindo mais agilidade e menos erros. Para a construção deste *framework*, foram utilizadas cinco bibliotecas com diferentes funções, a saber: interface, gerenciamento de estados e navegação.

A primeira biblioteca selecionada foi a Chakra, já mencionada anteriormente, que é uma biblioteca abrangente de componentes *React* acessíveis, reutilizáveis e combináveis que agiliza o desenvolvimento de aplicativos e sites da *Web* modernos. A biblioteca oferece uma gama diversificada de componentes que podem ser facilmente combinados para criar interfaces de usuário complexas, respeitando as práticas recomendadas de acessibilidade. A biblioteca foi instalada com o comando mostrado na Figura 9.

Figura 9: Comando de instalação da biblioteca Chakra

```
--- ~ » npm i @chakra-ui/react @emotion/react @emotion/styled framer-motion
```

Fonte: Autor (2023).

Após instalada a biblioteca no *Framework* ATS, para se ter acesso aos componentes de UI implementados pela Chakra é preciso informar ao nosso arquivo de entrada, ou seja, o arquivo que é executado quando o *Framework* ATS está em funcionamento, o provedor de componentes da biblioteca. É possível atingir isto como visto na Figura 10 abaixo, onde tem-se nosso componente App que é o componente de entrada do *Framework*, e ele está envolto pelo componente ChakraProvider, que dá acesso a todos os componentes da Biblioteca.

Figura 10: Componente provedor da biblioteca Chakra UI

```
1   import { ChakraProvider } from "@chakra-ui/react";
2   import React from "react";
3   import ReactDOM from "react-dom/client";
4   import App from "./App";
5   import "./index.css";
6
7   ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(
8     <React.StrictMode>
9       <ChakraProvider>
10        <App />
11      </ChakraProvider>
12    </React.StrictMode>
13  );
14
```

Fonte: Autor (2023).

Foram utilizados os componentes base do Chakra e a biblioteca também fornece os tipos de propriedades que os componentes recebem, sejam eles de estilo ou de controle de estado. Exemplificado na Figura 11 pode-se visualizar que da biblioteca Chakra é importado para o arquivo o componente *Text* o qual é renomeado para ChakraText, e suas propriedades, denominadas TextProps. Logo, podemos criar um componente de Texto chamado Text que recebe da biblioteca as tipagens corretas. Assim temos o *autocomplete* fornecido pelo TypeScript para o editor de texto.

Figura 11: Uso dos componentes fornecidos pela biblioteca Chakra UI

```
1  import { Text as ChakraText, TextProps } from "@chakra-ui/react";
2  import React from "react";
3  const Text = ({ children, ...props }: TextProps): React.ReactElement => {
4    return <ChakraText {...props}>{children}</ChakraText>;
5  };
6
7  export default Text;
8
```

Fonte: Autor (2023).

Especificamente, os componentes de entrada (*input*) precisaram de mais configurações que foram alcançadas com as bibliotecas *react-hook-form. A react-hook-form* é uma biblioteca flexível que gerencia formulários, suas validações, controle de estados e dados de usuário. A instalação da biblioteca é feita a partir do comando na figura 12 abaixo:

Figura 12: Comando de instalação da biblioteca *react-hook-form*

```
--- ~ » npm install react-hook-form
```

Fonte: Autor (2023).

Vale ressaltar que a validação de formulários é um processo dispendioso e exaustivo de se fazer manualmente, tornando o código muito repetitivo e cada vez mais sujo. Dessa forma, para melhorar a validação dos campos foi instalado a biblioteca *Yup* que nada mais é que um construtor de esquemas para validação de campos e transformações de valores no JavaScript. Os esquemas de validação são muito flexíveis e permitem modelar complexas validações, correlacionadas ou não, e até mesmo transformação de valores. Pode-se observar na Figura 13 um exemplo de esquema de validação usando *Yup*, referente ao formulário da tela de login.

Nesta tela temos dois campos, um de e-mail e um de senha, como mostrado abaixo, alguns parâmetros são passados para validar ambos os campos.

No campo de e-mail é desejável que ele seja uma *string*, que seja do tipo e-mail (esse tipo é fornecido pela própria biblioteca, que valida se o formato do campo é um formato de e-mail válido) e que ele é necessário. Já no campo de senha, está sendo validado se é uma *string*, se tem no mínimo 8 caracteres e é um campo obrigatório.

Figura 13: Esquema de Validação usando *Yup*

```
const schema = yup.object().shape({
  [Login.email]: yup.string().email().trim().required(),
  [Login.senha]: yup.string().min(8).trim().required(),
});
```

Fonte: Autor (2023).

Para o gerenciamento de estados globais foi utilizada a biblioteca *Zustand* que é uma solução de gerenciamento de estados simples, pequena, rápida e escalável, usando princípios de fluxo simplificado. Com essa biblioteca é possível passar dados entre todos os componentes da aplicação de forma simples. Por padrão, quando implementamos o gerenciamento da Zustand, criamos componentes chamados de *stores*, onde criamos o conjunto de funções de manipulação de estados para um componente. No *Framework* ATS foi implementado uma *store* de autenticação, consumida pela página de login. Apesar do *Framework* ATS não ter implementado ainda um *back-end* para requisições de APIs, já foi deixado configurado uma possível interação através de uma *store* da Zustand, chamada de *userSignInStore,* como podemos ver na Figura 14 abaixo. Na linha 10 foram feitas as tipagens das funções executadas pela *store.* Na linha 19 a *store* é criada através do método *create* fornecido pela biblioteca, e definimos como as funções explicitadas na interface da linha 10 se comportam.

Figura 14: Função de autenticação utilizando a biblioteca Zustand

```tsx
/* eslint-disable no-unused-vars */
import { create } from "zustand";
import AuthRepository from "../../services/auth/authRespository";
import { Status, StatusType } from "./types";

interface ISignInState {
  status: Status;
}

interface ISignInActions {
  signIn: (email: string, password: string) => Promise<void>;
  reset: () => void;
}

const initialState: ISignInState = {
  status: { type: StatusType.IDLE },
};

const useSignInStore = create<ISignInState & ISignInActions>((set) => ({
  ...initialState,
  signIn: async (email: string, password: string): Promise<void> => {
    set({ status: { type: StatusType.LOADING } });

    const response = await AuthRepository.signIn(email, password);

    set({ status: response });
  },
  reset: (): void => set(initialState),
}));

export default useSignInStore;
```

Fonte: Autor (2023).

Para a navegação entre as páginas da aplicação foi utilizada a biblioteca *react-router,* que é uma biblioteca para o *React* criada com o objetivo de resolver problemas de rotas para as páginas de um site e tornar o desenvolvimento muito mais simples e escalável. Com ela, podemos declarar caminhos do site para renderizar componentes e, até mesmo, resolver parâmetros na URL. A Figura 15 mostra como é feita a instalação da biblioteca e suas dependências.

Figura 15: Comando de instalação da biblioteca *react-router*

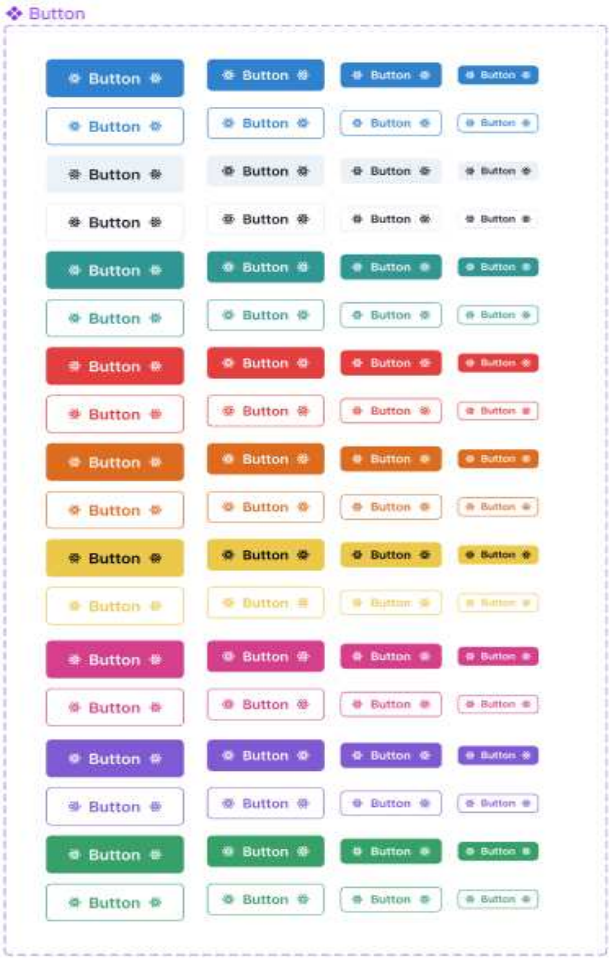
```
--- ~ » npm install react-router-dom localforage match-sorter sort-by
```

Fonte: Autor (2023).

4.4 Elementos criados no *Framework* ATS

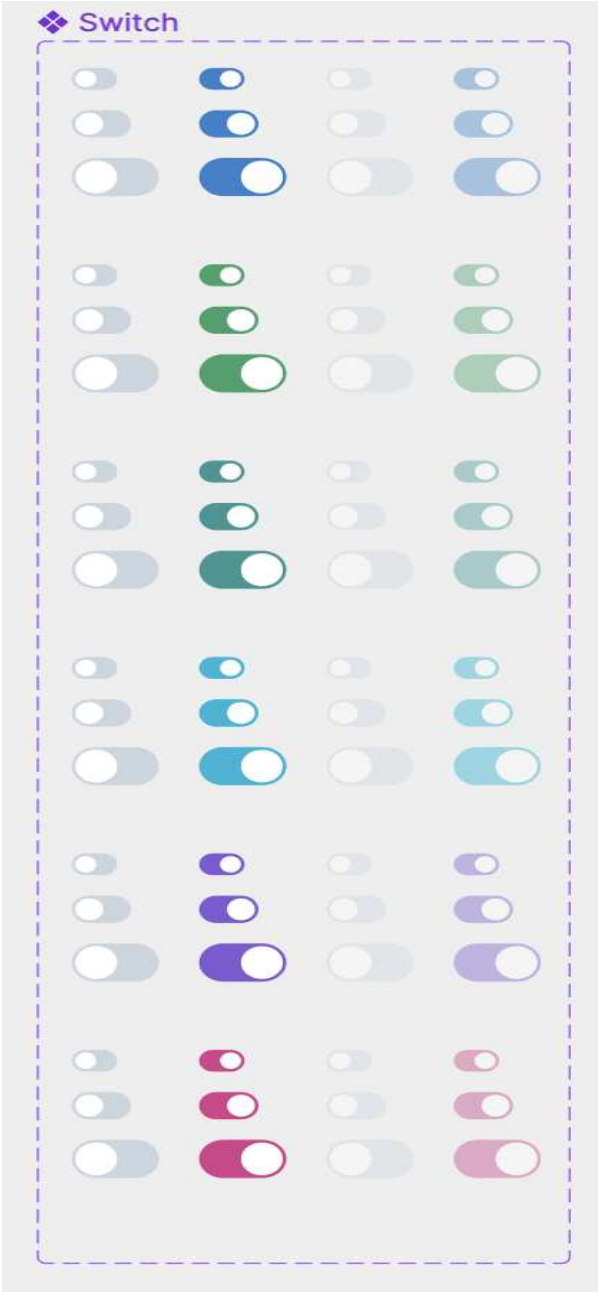
Para auxiliar a aceleração do desenvolvimento de um novo sistema, foram criados cerca de quinze componentes altamente customizáveis para a montagem de qualquer interface que o usuário necessite. Os componentes têm controles de estado base e podem ser modificados de acordo com a documentação da biblioteca Chakra. Os componentes podem ser encontrados no caminho "/src/components". Dentre alguns componentes criados, podemos citar: botões, *inputs*, *selects*, áreas de texto, tipografia, formulários e barras de progresso. As Figuras 16 e 17 demonstram a flexibilidade gerada pelos componentes do *Framework* ATS, onde na Figura 16 temos várias possíveis estilizações de um componente de botão e na Figura 17 possíveis estilizações do componente de *switch*.

Figura 16: Possíveis estilizações de botões do ATS



Fonte: Figma Chakra UI, 2022.

Figura 17: Possíveis estilizações de Switch do ATS

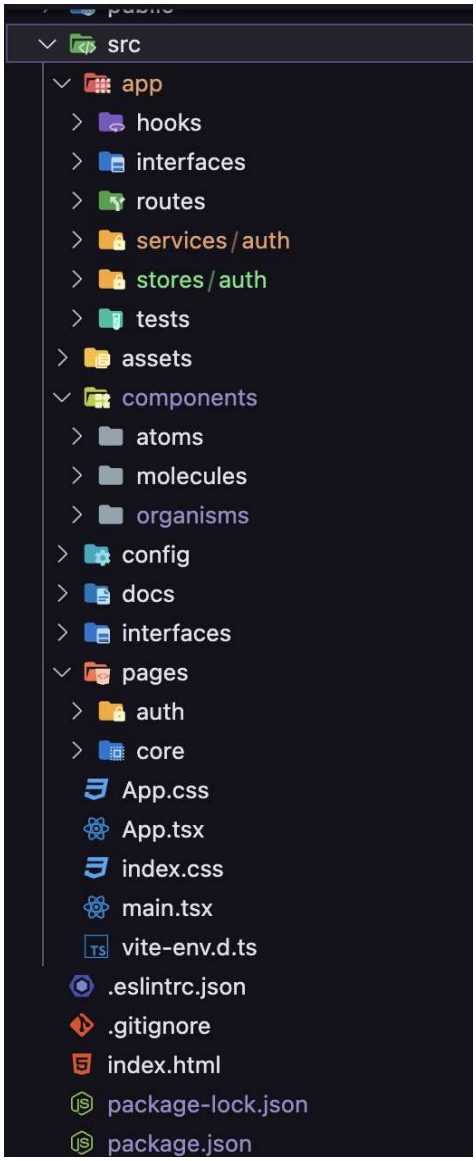


Fonte: Figma Chakra UI, 2022

Foi implementado e configurado toda a navegação e sistemas de rotas com a biblioteca *react-router*, possibilitando que o usuário possa criar suas próprias rotas, sejam elas autenticadas ou não. Foi implementado separadamente no *framework* rotas autenticadas e não autenticadas com o redirecionamento automático. As rotas podem ser modificadas ou alteradas no caminho "/src/app/routes", adicionando as rotas autenticadas à pasta *core* e as não autenticadas à pasta *auth*.

Foi criada uma página de Login utilizando os componentes do *framework* para prover os componentes às rotas já inseridas por padrão. Novas páginas podem ser adicionadas no caminho "/src/pages", adicionando os componentes referentes às rotas autenticadas na pasta *core* e as não autenticadas à pasta *auth*. Essas páginas ainda não estão se comunicando com nenhum *back-end,* porém a integração pode ser feita facilmente com o uso da biblioteca *Axios.* Na Figura 18 podemos ver a estrutura de pastas do *Framework* ATS, no diretório src temos todos os componentes e métodos contidos pela aplicação. Podemos citar alguns diretórios inseridos em src que são muito importantes, como o de componentes, que podemos ver a estruturação baseada em atomic design, a pasta app que contém os arquivos de navegação etc.

Figura 18: Estrutura de pastas do *Framework* ATS



Fonte: Autor (2023).

4.5 Instalação do *Framework* ATS

Para instalar o Framework é necessário ter pré-instalado o Node.JS. Após a instalação do Node, é preciso clonar o repositório no *Github*[17] e executar o comando “npm install”. Esse comando fará a instalação de todas as bibliotecas que o framework depende para o funcionamento correto. Após a instalação das bibliotecas, é necessário executar o comando “npm run dev” que iniciará um servidor local, no qual o desenvolvedor terá acesso à página de login, que foi pré-estabelecida como página inicial.

[17] *GitHub - thiago-cmont/ats*

5 VALIDAÇÃO DO *FRAMEWORK* ATS

5.1 Aplicação de formulário para validação

A validação do *Framework* ATS ocorreu por meio da construção, aplicação e preenchimento de um formulário via *Google Forms*, com o intuito de avaliar o *feedback* de desenvolvedores que atuam na área e que puderam testar o *framework* construído, a fim de melhorar a ferramenta e disponibilizá-la no GitHub para uso em ambientes acadêmicos, após análise das respostas dos participantes.

Participaram desta pesquisa nove desenvolvedores que preencheram completamente o formulário disponibilizado e adicionaram o arquivo da versão executada por eles. É válido ressaltar que todos os participantes que preencheram o formulário atuam na área de desenvolvimento de *software*.

O formulário elaborado dispunha de dez perguntas, sendo cinco delas objetivas e cinco subjetivas; e três ordens de execução para construção de algumas telas do *framework*, a fim de avaliar o nível de entendimento e execução de componentes, navegação e rotas pré-configuradas, a partir das ferramentas disponibilizadas.

A primeira pergunta contida no formulário fazia um questionamento objetivo acerca do nível de dificuldade para instalar e começar a utilizar o *Framework* ATS. Os participantes tinham como opções de resposta os seguintes itens: Muito fácil; Fácil; Mediano; Difícil; Muito difícil. Nessa questão, os programadores poderiam selecionar apenas um item como resposta. O objetivo do autor era compreender, pela perspectiva dos participantes, a dificuldade para iniciar a utilização do framework ATS, e torná-lo o mais intuitivo possível.

Logo após responder à primeira pergunta sobre a dificuldade de instalação e utilização do *Framework* ATS, o formulário apresentava aos desenvolvedores três ordens de execução, ao indicar a utilização do *Framework* ATS para cumprir os passos seguintes.

O intuito do autor era avaliar se o *framework* conseguiria cumprir seu objetivo, que seria acelerar o desenvolvimento de *softwares* ao fornecer componentes (botões, tipografias, *inputs*), navegação e rotas pré-configuradas para o desenvolvedor. Nesse sentido, as ordens de execução conseguiriam validar o *framework* ao analisar a forma como os participantes executariam as funções solicitadas.

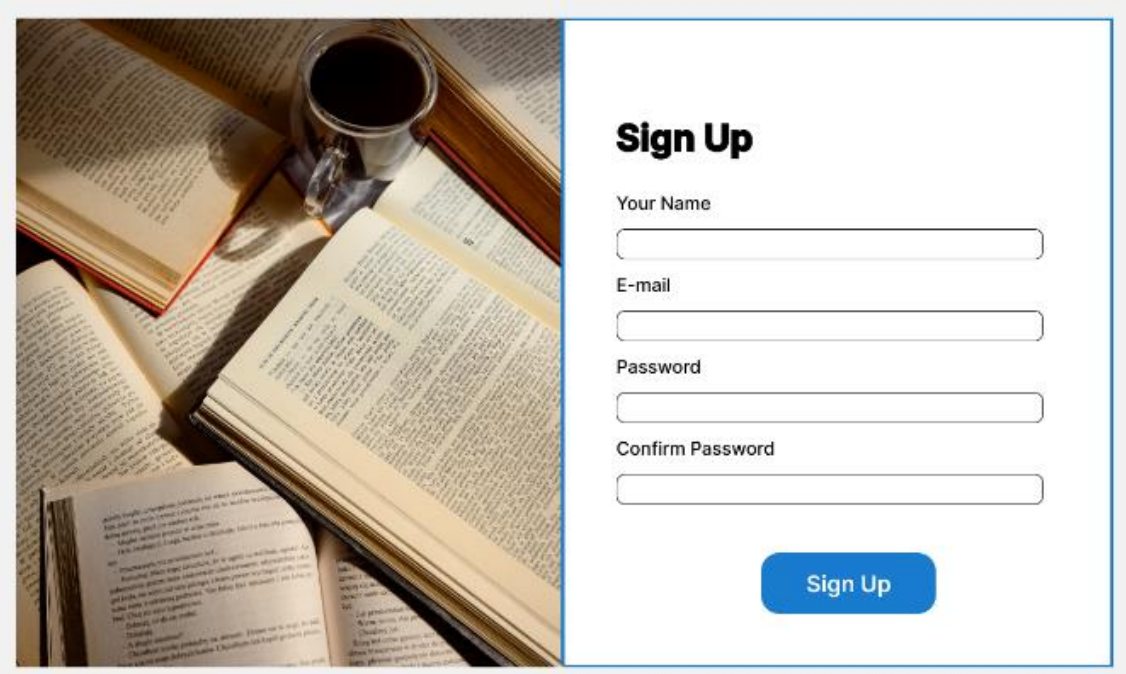
As ordens de execução continham as seguintes instruções:

I) Crie uma rota de navegação nova, nomeie-a de Cadastro, faça a navegação da tela de login fornecida pelo *framework* utilizando um link abaixo do formulário de login;

II) Na tela de Cadastro criada no passo anterior, crie um formulário com os campos de Nome, E-mail, Senha, Confirmar Senha;

III) Estilize a tela para atingir a seguinte aparência:

Figura 19: Tela a ser reproduzida no formulário



Fonte: Autor (2023).

O primeiro item objetivava analisar se os programadores conseguiriam utilizar a estrutura de navegação disponibilizada pelo *framework*, criando uma nova rota e integrando-a com a rota de login fornecida.

A segunda ordem de execução se baseava na lógica do formulário e solicitava ao respondente a criação de um formulário com os componentes fornecidos pelo *framework*, através de critérios de validação, tais como: Nome, E-mail, Senha e Confirmar Senha. O objetivo do autor era analisar se os participantes iriam utilizar a validação de campos previamente estabelecida no formulário da tela de login.

Por fim, a terceira ordem de execução possuía o intuito de analisar se o participante conseguiria utilizar os componentes de UI e estilizá-los, tais como: componente de imagem, de tipografia, de botão e de visualização do formulário. O objetivo seria criar uma tela conforme uma imagem fornecida no formulário, utilizada como modelo a ser reproduzido.

As dez perguntas subsequentes foram baseadas na experiência dos participantes em executar as tarefas solicitadas nas três ordens de execução supracitadas. Foi utilizada a escala

Likert para responder às perguntas objetivas do formulário. Esta escala consiste em um instrumento psicométrico no qual o respondente deve indicar sua concordância ou discordância sobre uma afirmação, feito por meio de uma escala ordenada e unidimensional, utilizando a numeração de 1 a 5 (MATAS, 2018).

O formulário propunha a alternância das respostas pelo uso da escala Likert (o participante deveria selecionar um número de 1 a 5 que respondesse à pergunta de acordo com sua percepção). A mensuração das respostas era baseada da seguinte forma: 1) Muito Fácil; 2) Fácil; 3) Mediano; 4) Difícil; 5) Muito difícil. Dessa forma, as perguntas 1, 3, 5 e 7 deveriam ser respondidas objetivamente pela seleção de um número correspondente à avaliação do participante acerca do framework. Já os itens 2,4,6, e 8 deveriam conter justificativas subjetivas do respondente sobre sua opinião acerca das perguntas precedentes.

A seguir serão demonstradas as perguntas contidas no formulário utilizado para validação do *Framework* ATS:

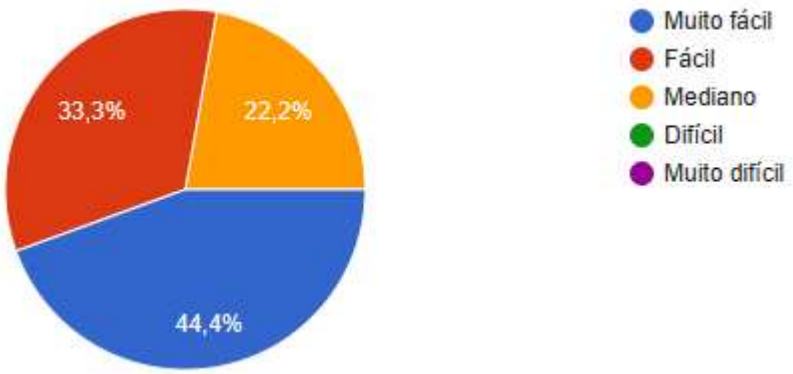
1) Numa escala de 1 a 5, quanto acha que o *Framework* ATS acelera o desenvolvimento inicial de um projeto?
2) Explique um pouco sobre sua opinião em relação à pergunta anterior (Pergunta 1).
3) Numa escala de 1 a 5, qual a dificuldade de entendimento do código do *Framework* ATS?
4) Explique um pouco sobre sua opinião em relação à pergunta anterior (Pergunta 3).
5) Numa escala de 1 a 5, qual foi sua dificuldade em utilizar o *Framework* ATS e as ferramentas fornecidas?
6) Explique um pouco sobre sua opinião em relação à pergunta anterior (Pergunta 5).
7) Numa escala de 1 a 5, qual foi o nível da sua curva de aprendizado ao utilizar o *Framework* ATS?
8) Explique um pouco sobre sua opinião em relação à pergunta anterior (Pergunta 7).
9) Indique, na sua opinião, os pontos negativos do *Framework* ATS.
10) Indique, na sua opinião, os pontos positivos do *Framework* ATS.
11) Faça o upload da pasta do projeto.

5.2 Feedback

Como descrito durante a etapa de validação do *framework*, a primeira pergunta elaborada pelo autor questionava o nível de dificuldade que os participantes tiveram para realizar a instalação e começar a utilizar o *Framework* ATS. As respostas foram dispostas

visualmente em gráficos, para melhor entendimento dos resultados, como mostrado no Gráfico 1, a seguir:

Gráfico 1: Nível de dificuldade para instalação e utilização do *Framework* ATS



Fonte: Autor (2023).

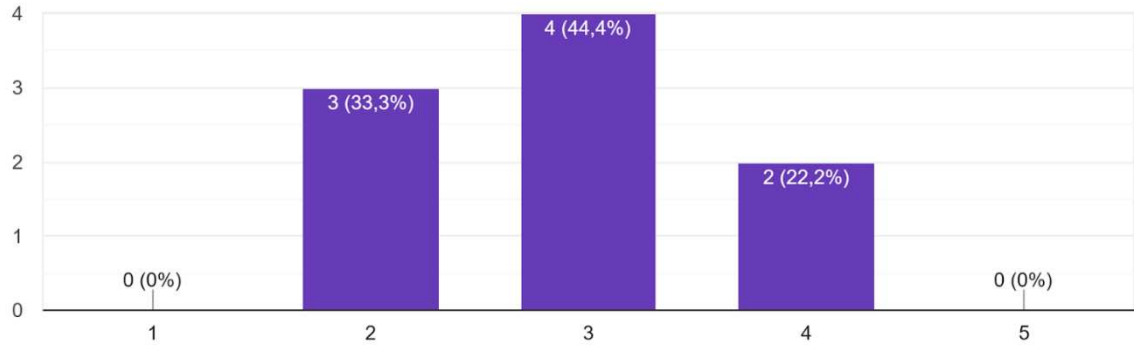
Como percebido pelas informações contidas no gráfico, as respostas dos participantes acerca do nível de dificuldade para instalação e utilização do *Framework* ATS giraram em torno dos itens “muito fácil” e “fácil” (quase 80% das respostas). Tal fato evidencia que a etapa inicial proposta foi facilmente executada pelos desenvolvedores.

Em relação às ordens de execução, os nove participantes conseguiram desenvolver as atividades propostas conforme o esperado pelo autor, tanto no que diz respeito à criação das rotas de navegação novas, como na criação do formulário conforme solicitado na segunda ordem de execução.

Em relação à estilização da tela e utilização dos componentes de UI, os participantes conseguiram estilizar todos os componentes de imagem, de tipografia, de botão e de visualização do formulário.

As perguntas subsequentes (1 a 10) foram baseadas na experiência dos respondentes completando as tarefas propostas pelas três ordens de execução, utilizando a escala Likert como critério de avaliação. A pergunta 1 fazia jus ao aceleramento disponibilizado pelo uso do framework ATS em relação ao desenvolvimento inicial de um projeto. O Gráfico 2 evidencia as respostas dos participantes acerca do nível de aceleração do *Framework* ATS para o desenvolvimento inicial de um projeto, a partir da perspectiva deles.

Gráfico 2: Nível de aceleração do *Framework* ATS para o desenvolvimento inicial de um projeto



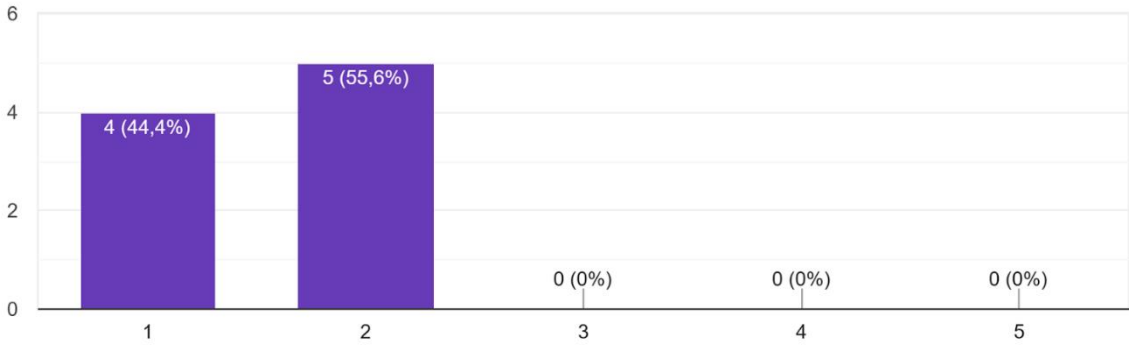
Fonte: Autor (2023).

Em relação às justificativas sobre a avaliação do nível de aceleração do framework ATS para o desenvolvimento inicial de um projeto, os participantes, codificados de P1 a P9, por ordem de resposta, deram as seguintes declarações na questão 2 do formulário:

> *P3: "A utilização de bibliotecas/frameworks que contenham um set de componentes pré-existentes e passíveis de estilização permitem uma maior celeridade no desenvolvimento, e tudo isso o ATS fornece".*
>
> P5: *"Mesmo não sendo um desenvolvedor frontend, pude perceber o valor do ATS e como ele pode impulsionar significativamente a velocidade do desenvolvimento. O framework oferece ferramentas e recursos que simplificam e agilizam a criação de interfaces de usuário, gerenciamento de estado, roteamento e outras tarefas comuns do desenvolvimento frontend".*
>
> P7: *"Entrega components que agilizam o desenvolvimento, mas poderia entregar algumas funcionalidades a mais, como uma navegação pré-configurada".*
>
> P9: *"Como utiliza bons padrões de desenvolvimento é um ótimo começo de projeto, porém é bastante simples e necessita de uma melhor estruturação da navegação.*

A pergunta 3 contida no formulário referia-se à dificuldade de entendimento do código do *Framework* ATS. Todas as respostas obtidas giraram em torno da opção "muito fácil" e fácil", evidenciando boa compreensão do código por parte dos desenvolvedores, como evidenciado no Gráfico 3:

Gráfico 3: Nível de dificuldade de entendimento do código do *Framework* ATS



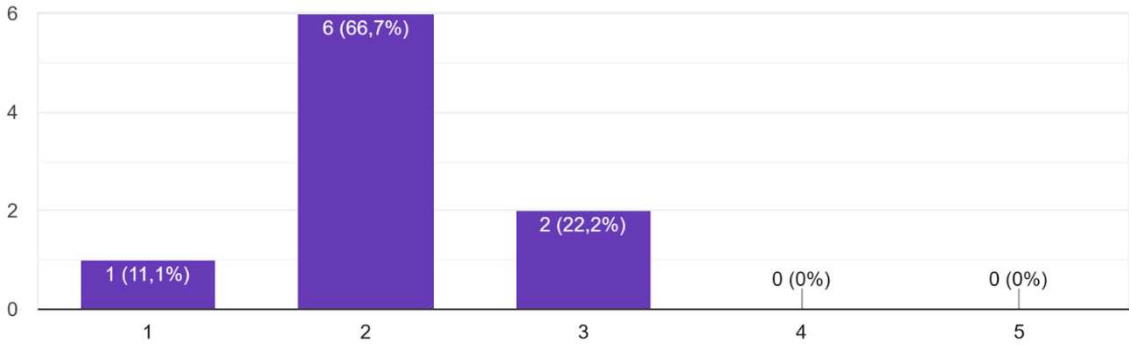
Fonte: Autor (2023).

Em relação às respostas subjetivas que justificaram a escolha dos participantes por classificar o entendimento do código como fácil, obteve-se os seguintes discursos:

> P2: "Código claro e escrito de forma reutilizável e usando os princípios de clean code".
>
> P4: *"O framework tem como linguagem o ECMAScript e por sua vez o superset TypeScript, acrescido de uma biblioteca de componentes pré-estilizados. Essa junção de tecnologias e ferramentas, é amplamente difundida no desenvolvimento Web. Existem documentações e exemplos práticos, que ajudam um desenvolvedor mesmo ele não sendo tão experiente a entender e evoluir o código do framework.*
>
> P5: *"Acredito que o código do framework seja extremamente fácil de entender, com uma ótima clareza e organização. A documentação e a estrutura do código são bem elaboradas, o que facilita o aprendizado e a compreensão das funcionalidades e fluxos do framework".*
>
> P8: *"O código é simples e bem escrito, utiliza as tecnologias mais recentes o que auxilia bastante no entendimento do código.*

A pergunta 5 indagava aos respondentes acerca da dificuldade em utilizar o framework e as ferramentas fornecidas. Sete respostas oscilaram entre os itens "muito fácil" e "fácil" e duas respostas selecionaram o número correspondente à "mediano", como visto no Gráfico 4 demonstrado abaixo:

Gráfico 4: Dificuldade em utilizar o *Framework* ATS e as ferramentas fornecidas



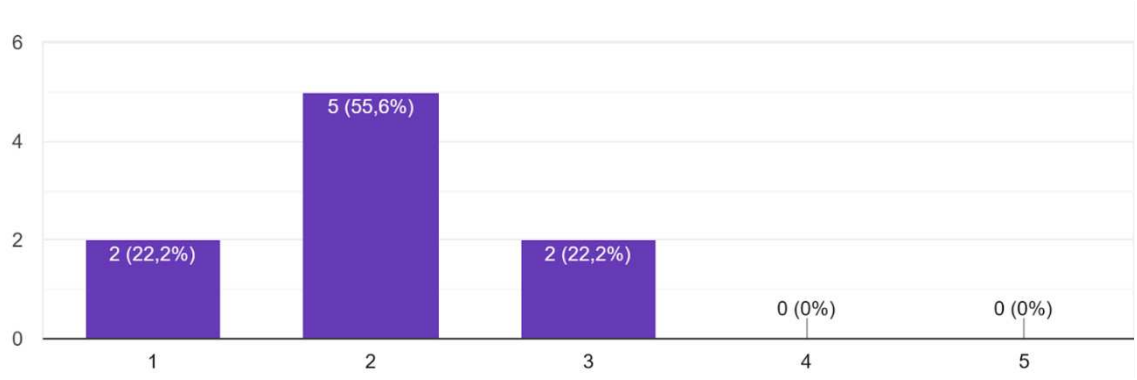
Fonte: Autor (2023).

Acerca das respostas justificando o item anterior, os participantes foram bem sucintos, referindo a facilidade de utilizar a ferramenta, mesmo alguns não tendo experiência como frontend. Contudo, a simplicidade de algumas etapas do framework também se tornou ponto negativo, como a falta de documentação relatada por alguns. Segundo os desenvolvedores:

> P1: *"Não houve dificuldade, a construção do formulário se dá com a importância dos componentes e a estilização dos mesmos, prática comum quando se utiliza ferramentas desse tipo.*
>
> P6: *"Tive uma leve dificuldade pois não sou desenvolvedor frontend, mas mesmo assim o processo foi fácil e simples para entender as ferramentas".*
>
> P7: *"Faltou implementação de certas features para que o tutorial pudesse ser realizado corretamente e também faltou documentação".*
>
> P9: *"Os componentes são de fácil entendimento e estão bem estruturados para achá-los no código, mas uma documentação cairia bem para facilitar ainda mais".*

A sétima pergunta do formulário fazia uma indagação a respeito do nível da curva de aprendizado dos participantes ao utilizarem o *Framework* ATS. As respostas obtidas evidenciaram que a curva de aprendizado variou entre muito fácil e mediano, ou seja, os desenvolvedores tiveram uma aprendizagem relativamente rápida das ferramentas do *framework*. O Gráfico 5 evidencia as respostas sobre a curva de aprendizado em relação ao *Framework* ATS:

Gráfico 5: Curva de aprendizado ao utilizar o *Framework* ATS



Fonte: Autor (2023).

Acerca das respostas escritas sobre o item anterior, os desenvolvedores relataram facilidade no aprendizado das ferramentas fornecidas, tendo em vista a simplicidade do framework, o que pode ser demonstrado abaixo:

> P1: *"A curva de aprendizado foi rápida e fácil. Eu consegui assimilar os conceitos e utilizar as ferramentas do framework de forma ágil, mesmo sem ter experiência prévia com ele".*
>
> P2: "*Embora o framework utiliza-se de tecnologias amplamente difundidas na Web. Essas tecnologias têm por si, sua curva de aprendizado. É necessário muito tempo e esforço para dominar, linguagem de marcação, linguagem de estilização, paradigma de programação, entender os tipos de dados, além de outras especificidades envolvidas".*
>
> P6: *"O framework não tem muitas particularidades que o torne de difícil entendimento. É muito fácil de baixá-lo e já utilizá-lo sem muita dificuldade"*
>
> P8: "*A curva de aprendizado é relativamente baixa".*
>
> P9: *"Como todo framework é necessário olhar mais a fundo o código e entender o propósito de cada componente. Este é um framework com ótima arquitetura e de fácil entendimento".*

Em relação aos pontos negativos observados no framework ATS, muitos participantes citaram a falta de documentação como ponto principal, além da falta de melhor pré-configuração de navegação e falta de interface de comunicação com APIs, como pode ser lido nos discursos abaixo:

> P5: "*Não possuir uma documentação detalhada de como iniciar um novo projeto para iniciantes e todas suas funcionalidades; Não possuir um biblioteca instalada e pré-configurada que lide com o roteamento; Não possuir um template pré-configurado para conseguir se comunicar com uma API externa".*

> P6: "*Necessita de uma melhor estruturação da navegação e roteamento, seria bom ter mais páginas pré-prontas para ter uma navegação completa. Também seria interessante uma interface para interagir com APIs".*
>
> P9: "*Um ponto importante e necessário é a estruturação e componentização da navegação de modo que seja possível utilizar da mesma forma que os outros recursos. Em frameworks assim também é interessante dispor de documentação para consulta".*

A partir do feedback dos desenvolvedores sobre a carência de uma documentação mais detalhada, foi criado um arquivo *markdown*, ou seja, uma linguagem de marcação simples e fácil de usar que servirá para documentar o *Framework* ATS. Nessa documentação foram explicitados os motivos da criação do framework, uma lista de dependências necessárias para sua utilização e um guia de instalação e primeiros passos. Também foram mostrados como o desenvolvedor poderia fazer o uso das ferramentas do framework ATS e um código de exemplo da sua utilização.

Outro ponto negativo citado pelos participantes foi a falta de pré-configuração de navegação. Para melhorá-la, foram criadas mais rotas pré-configuradas (Cadastro e Home) e criadas as interações entre as rotas para servir de melhor embasamento aos utilizadores do framework. Foi utilizada a biblioteca react router, sendo uma solução mais robusta e difundida no mercado do que a biblioteca proposta na versão anterior do framework.

Quanto à falta de interface com APIs, foi implementada uma pré-configuração da biblioteca react query, sendo uma solução mais robusta do que a proposta anteriormente com a biblioteca axios. A react query é bem mais performática e facilita muito com que o desenvolvedor faça ações como *refetch* e cacheamento. A biblioteca também tem embutido um gerenciamento de estado que pode ser configurado a fim de evitar requisições ao mesmo endereço várias vezes consecutivas.

Quanto aos pontos positivos, os desenvolvedores abordaram os seguintes tópicos:

> P2: "*O framework resolve o que se propõe a resolver. É minimalista o que garante versatilidade e confiança, e por utilizar linguagens e sub ferramentas modernas o torna seguro. Além de fácil de compreender e de evoluir para os desenvolvedores mais experientes".*
>
> P5: "*O framework resolve os problemas iniciais e conhecidos de todo projeto e consegue padronizar e acelerar o desenvolvimento de uma aplicação; Estrutura de pastas e arquivos muito bem definida; Utilização de ferramentas modernas e consolidadas pelos desenvolvedores front-end".*
>
> P6: "*É um ótimo começo de projeto, adiantando muitas das configurações iniciais de qualquer projeto. Utiliza ótimas tecnologias como TypeScript, Vite, React que são super atuais e fáceis de dar manutenção. O fato de usar TypeScript também ajuda muito com o autocomplete na hora de programar".*

> P9: "*Framework leve, de fácil utilização, a forma como foi construído dispondo de componentes que vão aumentando de complexidade (atomic design) e que permitem ser combinados de diferentes faz com que seja uma poderosa e versátil ferramenta para o desenvolvimento".*

Por fim, foi solicitado aos desenvolvedores o *upload* dos projetos feitos por eles, a fim de confirmar a execução das ações propostas pelo formulário. Os códigos-fonte podem ser consultados Google Drive[18].

[18]https://drive.google.com/drive/u/4/folders/1Ps1Zk0dZU1q6z898NaAp48qR4G7e6h1sohjEk8q_pXQiCsCa1DuBXyp4j5fvnugUyc3px4kd.

6 CONSIDERAÇÕES FINAIS

A realização deste trabalho permitiu criar uma ferramenta de desenvolvimento que visa acelerar a construção de projetos de outros desenvolvedores de diferentes níveis de conhecimento técnico. Durante o processo, percebeu-se a necessidade de implementar diversas melhorias, tais como a documentação, a navegação e interface com APIs, possibilitando a composição de um framework mais completo e robusto que possa resolver os problemas mais comuns no início de um novo projeto de desenvolvimento Web.

É válido salientar que há extrema relevância em se discutir sobre os padrões que compõem um framework, pois são extremamente vastos e complexos, tendo que solucionar múltiplos problemas em diversas áreas, tais como arquitetura de projeto, padronização de código, criação de componentes funcionais, criação de estruturas de navegação, criação de interface com APIs, validação de formulários, dentre outros.

Para tanto, torna-se indiscutível a necessidade de fomentar a construção do conhecimento acadêmico acerca do desenvolvimento de software e, principalmente, desenvolvimento Web, que é amplamente utilizado por muitas pessoas no contexto atual de um mundo extremamente dependente da tecnologia.

Nesse sentido, buscou-se elencar alguns elementos que pudessem ser aplicados a diferentes problemas enfrentados pelos desenvolvedores. Dessa forma, ter levado em consideração a percepção de profissionais da área foi de extrema importância para implementar melhorias e tornar o framework ATS mais amigável, eficiente e robusto na resolução de problemas comuns no desenvolvimento de software.

Como possibilidade de trabalhos futuros, tem-se a criação do *back-end* do *Framework* ATS, utilizando *NodeJS* com a biblioteca Express para prover funcionalidades base utilizadas em todos os sistemas *Web.* Também pode-se implementar um banco de dados utilizando *Postgres* para comunicar-se com o *back-end* e prover os dados necessários para o funcionamento correto da aplicação.

Portanto, espera-se que, futuramente, o estudo aqui produzido possa servir de referencial teórico para outros profissionais da área criarem ferramentas que resolvam ainda mais problemas no âmbito do desenvolvimento Web, para tornar o cotidiano laboral mais facilitado e despojado.

REFERÊNCIAS

ADAM, S.I.; ANDOLO, S. A New PHP Web Application Development Framework based on MVC Architectural Pattern and Ajax Technology. Aug 2019. **Computer Science**. 2019 1 st International Conference on Cybernetics and Intelligent System (ICORIS). DOI:10.1109/ICORIS.2019.8874912.

AGGARWAL, S. et al. Modern Web-Development using ReactJS. **International Journal of Recent Research Aspects** ISSN: 2349-7688, Vol. 5, Issue 1, March 2018, pp. 133-137.

BITTENCOURT, J. **Framework:** o que é e pra que serve essa ferramenta? Artigos > Programação. [Internet]. Alura. 16 nov 2021. Disponível em Framework: o que é e pra que serve essa ferramenta? | Alura

BORGES, J. **O que são os frameworks e quais os mais utilizados?** Blog [Internet]. Dio.me, 2023. Academia PME Educação e Consultoria em Negócios LTDA. Disponível em O Que São Frameworks e Quais os Mais Utilizados? (dio.me)

BOSCH, J. et al. **Framework Problem and Experiences**. In: FAYAD, M.E., SCHIMIDT, D.C. and JOHNSON, R.E., *Building Application Frameworks: Object-Oriented Foundations of Framework Design,* John Wiley & Sons, 1999.pp.55-86.

BOTELLA, F. et al. Selecting the best mobile framework for developing Web and hybrid mobile apps. In: Proceedings of the XVII International Conference on Human Computer Interaction. New York, NY, USA: ACM, 2016. (**Interacción '16**), p. 40:1–40:4. ISBN 978-1-4503-4119-6. Disponível em: http://doi.acm.org/10.1145/2998626.2998648

COUTINHO, T. **Descubra quais são os pontos cruciais da Curva de Aprendizagem e como mensurá-los**. Blog [Internet]. Voitto, 2021. Disponível em: Entenda o que é Curva de Aprendizagem e como calculá-la (voitto.com.br)

FROST, B. **Atomic Design.** Blog [Internet]. my name is brad frost, 2013. Disponível em https://bradfrost.com/blog/post/atomic-web-design/

GIL, A. C. (2017). **Como elaborar projetos de pesquisa** (6th ed., p. 192).

GUPTA, A. Part 1: **The road to cross platform frameworks**. 2018. Disponível em: https://medium.com/coding-blocks/part-1-the-road-to-cross-platform-frameworks-d6a193b9ce2d

HOSTGATOR. **Framework**: o que é, quais utilizar e como eles funcionam! Blog [Internet]. HostGator, 2023. Disponível em Framework: o que é, quais utilizar e como eles funcionam! (hostgator.com.br)

KRIGER, D. **O que são os frameworks e por que são importantes para os DEVs**. Programação. Blog [Internet]. KENZIE, 2022. Disponível em O que é framework, para que serve, vantagens e desvantagens (kenzie.com.br)

LIMA, E.P.; LEZANA, A.G.R. Desenvolvendo um *framework* para estudar a ação organizacional: das competências ao modelo organizacional. **Rev Gestão e Produção.** v.12, n.2, p.177-190, mai.-ago. 2005.

LIMA, F.F. **Avaliação de frameworks para o desenvolvimento de aplicações híbridas.** Trabalho de Conclusão de Curso. Graduação em Engenharia de Software. Universidade Federal do Pampa. 2019. 119f. Disponível em title (unipampa.edu.br)

MATAS, A. Likert-Type Scale Format Design: State of Art. **Revista Electrónica de Investigación Educativa,** 20(1), 38-47. https://doi.org/10.24320/redie.2018.20.1.134. Disponível em 1607-4041-redie-20-01-38.pdf (scielo.org.mx)

MEDEIROS, A. YPEDUC: Uma adaptação de metodologia ágil para o desenvolvimento de software educativo. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE).** 2019. p.379.

MINAYO, M.C.S. **O desafio do conhecimento**: pesquisa qualitativa em saúde**.** São Paulo; Hucitec; 14. ed; 2014. 407 p.

MOCBEL, M.A.R. et al. Um Framework para Desenvolvimento de Softwares de Suporte ao Ensino para Plataformas Móveis. **RENOTE**, Porto Alegre, v. 18, n. 1, 2020. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/105970

NEVES, J.; JÚNIOR, V.M. **Uma análise comparativa entre Flutter e React Native como frameworks para desenvolvimento híbrido de aplicativos mobile**: um estudo de caso visando produtividade. Curso de Ciências da Computação. Universidade do Sul de Santa Catarina (UNISUL). 2020. 19f. Disponível em Artigo final.pdf (animaeducacao.com.br)

NOLETO, C. **Framework**: o que é, como ele funciona e para que serve? Framework de programação. Blog [Internet]. Trybe, 2020. Disponível em Framework: o que é, como ele funciona e para que serve? (betrybe.com)

ROSALES, G.C.M. **Modelo e framework para o desenvolvimento de ferramentas analíticas de apoio ao ensino, aprendizagem e gestão educacional**. Tese (Doutorado). Universidade Federal de São Carlos, 2014. 208f. Disponível em 6494.pdf (ufscar.br)

SANTANA, A**. Linguagens de Programação:** uma breve introdução contextualizada. Artigos > Programação. Blog [Internet]. Alura, 2023. Disponível em: Linguagem de Programação: O que é e as mais usadas | Alura

SARKER, I.H.; ABU, K.; MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application**. International Journal of Hybrid Information Technology** Vol.7, No.5 (2014), pp.317-322 http://dx.doi.org/10.14257/ijhit.2014.7.5.29

SIEMENS, G.; BAKER, R. S. J. D. Learning Analytics and Educational Data Mining: Towards Communication and Collaboration. In: **INTERNATIONAL CONFERENCE ON LEARNING ANALYTICS AND KNOWLEDGE**, New York, 2012. Proceedings of the 2nd International Conference on Learning Analytics and Knowledge. New York, 2012, p. 252-254.

SILVA, E.Q.; MOREIRA, D.A. **Um framework de componentes para o desenvolvimento de aplicações Web robustas de apoio à educação**. 2004, Anais. Manaus: Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2004.

SILVA, E.Q. **Um framework baseado em componentes para desenvolvimento de aplicações Web e um processo de instanciação associado**. Tese de Doutorado. Instituto de Ciências Matemáticas e de Computação – ICMC-USP. USP São Carlos. 2006. 167p. Disponível em Microsoft Word - tese-junho.doc (usp.br)

SOUZA, A.A. O uso de *softwares* educativos como ferramenta no processo de ensino e aprendizagem para construção de uma autonomia do estudante do ensino médio com intermediação tecnológica da Bahia – EMITEC. **Revista Científica Multidisciplinar Núcleo do Conhecimento**. Ano 06, Ed. 07, Vol.10, pp. 99-110. Julho de 2021. ISSN: 2448-0959. Disponível em: https://www.nucleodoconhecimento.com.br/educacao/softwares-educativos

STACK OVERFLOW. **2022 Developer Survey**. Disponível em Stack Overflow Developer Survey 2022

XANTHOPOULOS, S.; XINOGALOS, S. A comparative analysis of cross platform development approaches for mobile applications. In: Proceedings of the 6th Balkan Conference in Informatics. New York, NY, USA: ACM, 2013. (**BCI '13**), p. 213–220. ISBN 978-1-4503-1851-8. Disponível em: http://doi.acm.org/10.1145/2490257.2490292

ZUCHER, V. **O que é padrão MVC?** Entenda arquitetura de software. Blog [Internet]. Le Wagon. 2020. Disponível em https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc?utm_campaign=&utm_adgroup=&gclid=CjwKCAiAp7GcBhA0EiwA9U0mthV_kvW5YOhBOveiXvVeLoW5Q22HLdMx6-Wnflo_T4PXtjyKP2QKkRoCIxgQAvD_BwE