JULIANO EXPEDITO DE ANDRADE GODINHO

RELATÓRIO TÉCNICO BITKA ANALYTICS:

DESENVOLVIMENTO DE SOFTWARE

LAVRAS - MG

2024

JULIANO EXPEDITO DE ANDRADE GODINHO

RELATÓRIO TÉCNICO BITKA ANALYTICS:
DESENVOLVIMENTO DE SOFTWARE

Relatório técnico apresentado ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação

Profa. Juliana Galvani Greghi
Orientadora

LAVRAS - MG

2024

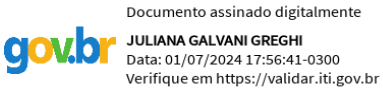
JULIANO EXPEDITO DE ANDRADE GODINHO

RELATÓRIO TÉCNICO BITKA ANALYTICS: DESENVOLVIMENTO DE SOFTWARE

Relatório técnico apresentado ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação

APROVADA em 07 de Maio de 2024.

Profa. Juliana Galvani Greghi	UFLA
Prof. Maurício Ronny de Almeida Souza	UFLA
Profa. Renata Teles Moreira	UFLA



Profa. Juliana Galvani Greghi
Orientadora

LAVRAS - MG
2024

Dedico este trabalho à minha esposa Giovanna, à minha filha Helena, aos meus pais Cássio e Girlene e à minha irmã Ana Luísa por estarem sempre ao meu lado

AGRADECIMENTOS

À minha família, em especial à minha esposa, minha mãe, meu pai e minha irmã, que foram responsáveis por estarem sempre ao meu lado me auxiliando e me aconselhando nos momentos difíceis, além de me motivarem a continuar minha caminhada até o fim. Aos meus amigos por tornarem a trajetória menos árdua. Aos meus professores por todo conhecimento compartilhado. E aos meus companheiros de trabalho, por terem me treinado e ensinado muito nesse tempo de BITKA.

Se não queres sofrer, não ames; porém, se não amas, para que queres viver.
(Santo Agostinho de Hipona)

RESUMO

O presente trabalho visa apresentar as atividades realizadas na empresa BITKA Analytics, onde o aluno desempenhou diversas atividades dos mais variados níveis e conceitos. A experiência do aluno na área era limitada no início, especialmente nas tarefas de maior complexidade, porém foi crescendo a medida que os meses iam passando. O objetivo deste trabalho é, portanto, apresentar tudo que fora desempenhado, desde as tarefas até as tecnologias, além de também enfatizar o crescente conhecimento adquirido que teve como base o aprendizado obtido nas disciplinas da universidade.

Palavras-chave: Trabalho; Experiência; Tecnologias

ABSTRACT

This present work aims to present the activities carried out at the company BITKA Analytics, where the student performed various activities of different levels and concepts. The student's experience in the field was limited at the beginning, especially in the tasks of higher complexity, but it grew as the months passed. The objective of this work is to present everything that was performed, from the tasks to the technologies, and also to emphasize the increasing knowledge acquired, which was based on the learning obtained in the university courses.

Keywords: Work; Experience; Technologies

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Muitas vezes o estudante encontra dificuldades para se inserir no mercado de trabalho. Conseguir uma oportunidade que possibilite a busca pelo conhecimento e crescimento, com a contratação de pessoas com menor experiência no mercado permite, portanto, o desenvolvimento e possibilita o crescimento a partir de seus resultados dentro da própria empresa.

E foi essa oportunidade encontrada na BITKA Analytics, empresa em que o trabalho aqui apresentado foi desenvolvido. O relatório técnico apresentado neste documento é fruto da disciplina PRG610 - Estágio Supervisionado/TCC visando a conclusão do curso de Ciência da Computação na Universidade Federal de Lavras (UFLA) e gira em torno do trabalho empregado, à distância, na área de Desenvolvimento de Software da empresa.

O trabalho desenvolvido na empresa foi responsável por permitir e incentivar a migração do estudante da área de Ciência de Dados para a de Desenvolvimento de Software, ampliando assim toda uma vertente de conhecimento e gerando uma nova visão sobre a computação.

1.1 Motivação

Este início de carreira é de suma importância para o bom futuro do estudante, pois a experiência que o mercado de trabalho propicia possibilita a consolidação do que o estudante aprendeu em sua graduação, visto que o conteúdo das disciplinas é correlacionado às atividades exercidas no emprego. A aquisição de conhecimento através de experiência prática no mercado de trabalho consegue amplificar tudo que é adquirido durante a graduação para que, então, se possa compreender o real intuito de tudo anteriormente aprendido.

Adicionalmente, quando o conhecimento é adquirido através da experiência, é possível notar maior efetividade, visto que é mais fácil para que os estudantes mantenham o que fora recebido pela atividades práticas do mercado, que pelas realizadas em sala de aula, de forma teórica. Por conta disso, este trabalho foi motivado pela relevância em divulgar aos estudantes a importância em dedicar-se com determinação e responsabilidade tanto às aulas, quanto às posições em estágio ou emprego que venham a ocupar.

1.2 Objetivos

O propósito principal deste relatório é fornecer uma apresentação técnica das tarefas executadas durante o período de trabalho na empresa BITKA, bem como uma descrição detalhada das principais tecnologias, metodologias e desafios propostos. As atividades desempenhadas variam desde a criação de novas telas e funcionalidades, tanto na parte do *Front-End* quanto do *Back-End*, até a correção de inúmeros problemas e a otimização de métodos e funções para melhorar a performance do sistema. Vale salientar aqui que, devido aos sigilos contratuais do projeto ao qual o discente atua, não é permitido incluir informações sobre quem é o cliente, qual a finalidade do sistema em análise, ou incluir imagens das telas implementadas ou do código desenvolvido. Dessa forma, as atividades serão descritas de forma técnica sem citar nomes de entidades, telas ou qualquer coisa que possa comprometer o sigilo das informações.

Ademais, o relatório incluirá uma análise de como o conhecimento teórico e técnico adquirido ao longo dos anos de graduação foi útil para as atividades práticas realizadas durante o trabalho, além de descrever como essas atividades contribuíram para consolidar esse conhecimento. Pode-se, brevemente, exemplificar as disciplinas de Introdução aos Algoritmos e Engenharia de Software que fundaram uma sólida base para permitir que o estudante pudesse adquirir conhecimentos ainda mais profundos com os desafios enfrentados na empresa.

Por fim, o objetivo deste trabalho é descrever o trabalho realizado, apresentar os conhecimentos adquiridos pelo aluno e as conclusões a respeito de como a combinação entre o conhecimento adquirido na graduação e a prática de mercado inicial proporcionada pelo trabalho pode ser benéfica para a formação profissional do estudante.

1.2.1 Estrutura do Documento

O documento deste trabalho será dividido em diferentes seções, com início a partir do referencial teórico que apresentará todas as tecnologias utilizadas, desde as linguagens de programação e *frameworks* até as ferramentas (Seção 2). Em seguida, tem-se a descrição das atividades desenvolvidas desde o início do projeto como treinamento até a utilização da plataforma Azure DevOps com o Scrum e o Kanban (Seção 3) e, por fim, será relatada a ligação entre as disciplinas cursadas

e o conhecimento adquirido na universidade com o fora aprendido e executado na empresa (Seção 4).

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentadas informações sobre a empresa em que o trabalho foi desenvolvido e sobre as tecnologias e ferramentas utilizadas durante o período de 01 de Setembro de 2022 até 01 de Fevereiro de 2024.

2.1 A BITKA

A BITKA é uma empresa de *Advanced Analytics* formada por profissionais com mais de 16 anos de experiência no desenvolvimento e implementação de soluções analíticas que auxiliam seus clientes em seus processos de tomada de decisão. As informações a seguir foram retiradas do site da empresa[1].

- **Missão:** Liderar com inovação, experiência e leveza, focando na qualidade de vida das nossas pessoas. Desenvolver soluções inteligentes para diversas indústrias, com o objetivo de causar impactos positivos nos negócios e na sociedade, sempre com um ambiente acolhedor para nossos clientes e equipe.

- **Visão:** Sermos líderes em Soluções Analíticas, impulsionando a inovação em diversos setores. Criar um ambiente onde a paixão se alia ao sucesso, moldando um futuro onde a tecnologia e a criatividade caminham juntas.

- **Valores:** Nosso compromisso é com a equidade e diálogo aberto. Incentivamos a colaboração e o aprendizado, promovendo uma cultura de compartilhamento de conhecimento e tecnologia. Unidos, fortalecemos nossa capacidade de inovar e crescer.

Na Figura 2.1 tem se a estrutura geral da formação da empresa dos funcionários e a experiência dos fundadores. Atualmente ela conta com mais de 130 funcionários e está em constante crescimento, adquirindo novos clientes e funcionários.

[1] https://www.bitkaanalytics.com.br

Figura 2.1 – A Empresa

Fonte: BITKA Analytics

2.1.1 A Equipe

A Figura 2.2 ilustra como é a estrutura de funcionários dentro da empresa, representando o conceito de torres que será descrito a seguir.

A BITKA divide-se num conceito chamado de torres. No total são sete, sendo as três superiores (TI, Preditiva e Prescritiva) ligadas ao desenvolvimento de atividades e soluções. A parte de TI é a que o estudante faz parte e ela é responsável não só por sistemas *WEB*, mas também outros sistemas num geral. Já a parte preditiva é responsável, como o nome indica, pela predição com o uso de inteligência artificial, buscando prever soluções melhores. Enquanto que a parte prescritiva é responsável pela otimização das soluções dos clientes.

Já as quatro torres presentes na base da imagem representam a outra parcela de funcionários. Muitos deles não estão inseridos numa equipe, como os de marketing e infraestrutura, mas alguns, como os da parte de negócios e administrativo, geralmente estão ligados a liderança de muitas equipes.

Figura 2.2 – Composição da equipe

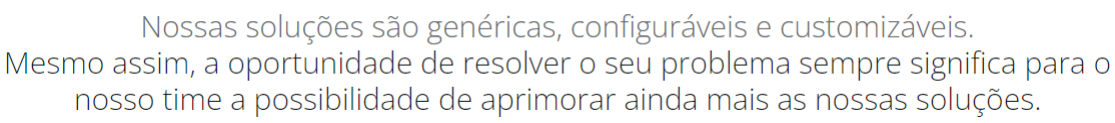
Figura 2.3 – Sobre a BITKA

- Na BITKA é constante a busca por criar alternativas que melhorem a vida das pessoas e organizações. É pura curiosidade, querer pegar, testar, falhar e não ter medo de fazer.
- Acreditamos que se não há diversão, também não há paixão. Sem esses dois fatores é praticamente impossível construir algo realmente transformador, que mova pessoas e mude o mundo.
- Não criamos nada sozinhos. Ideias são muito melhores quando nascem de conflito de argumentos, propostas e objetivos. Buscamos um relacionamento diverso e inclusivo, para fomentar uma verdadeira comunidade que cria e dissemina conhecimento e tecnologia da forma mais abrangente possível.

Fonte: BITKA Analytics

2.1.2 O Trabalho Desenvolvido na Empresa

Figura 2.4 – Soluções



Fonte: BITKA Analytics

A empresa preza por diversificação, inovação e flexibilidade. Seus preceitos estão resumidos na Figura 2.3:

- **Diversificação:** Soluções que tratam desafios de *Supply Chain*, Planejamento e Controle Logístico, Planejamento e Controle de Produção, Otimização Tributária, Planejamento de Transporte, Alocação de Tarefas, Dimensionamento e Alocação de Equipamentos, *Trading* de Mercadorias, Predição de Eventos e muito mais.
- **Pensamento Fora da Caixa:** Capacidade de unir todas as técnicas de *Advanced Analytics* em soluções únicas e disruptivas. Tendo experiências no desenvolvimento de sistemas que são capazes de apresentar a situação atual (Análise Descritiva), prever situações futuras (Análise Preditiva), e propor de maneira otimizada alternativas para obter o melhor resultado (Análise Prescritiva).
- **No Tempo Certo:** Prática em implementação de soluções de Planejamento de Longo, Médio, Curto, Curtíssimo Prazo, bem como de Controle Operacional.
- **Flexibilidade:** O desafio não é formatar o seu problema em uma de nossas soluções analíticas e sim combiná-las da melhor maneira possível para atender as especificidades do seu negócio.

A Figura 2.4 apresenta um convite aos recursos de solução empregados pela BITKA, de forma a chamar a atenção de possíveis futuros clientes.

2.2 Tecnologias e ferramentas

Durante o desenvolvimento das atividades, o discente teve contato com várias tecnologias, linguagens de programação e ferramentas diferentes, e estas serão descritas a seguir.

2.2.1 Front-End

O Front-End (ALURA, 2023) é, como o próprio nome indica, a parte frontal de um sistema, a interface de interação entre usuário e sistema, em muitas vees, sistemas Web. Ele é composto por tecnologias como o HTML, CSS e o JavaScript que serão descritas a seguir e é responsável pelo contato com o usuário, recebendo os dados de entrada e apresentando o resultado na tela, além de fazer a conexão com o servidor, enviando requisições. Além disso, neste emprego, o estudante utilizou a ferramenta Angular com a linguagem TypeScript. Estas duas tecnologias também serão descritas logo abaixo.

O HTML (MOZILLA, 2023b) é a sigla para *HyperText Markup Language* e significa, traduzindo do inglês, Linguagem de Marcação de Hipertexto (BERNES-LEE, 2018) e é a estrutura mais básica dentro do desenvolvimento Web. É uma linguagem que utiliza marcações para definir o conteúdo da página, como botões, imagens, título, campos para inserção de texto. Com esta linguagem, os desenvolvedores conseguem criar páginas melhor organizadas, visto que o HTML é composto por *tags* que serão interpretadas pelo navegador para uma exibição correta e portanto, gera uma interação, para o usuário, mais eficiente.

O CSS (MOZILLA, 2023a) (*Cascading Style Sheets*) é uma linguagem de estilização que formata o conteúdo de uma página, moldando os elementos HTML. Com o CSS é possível definir fontes, cores, espaçamentos, posicionamentos de elementos, entre muitos outros pontos, definindo, assim, como o conteúdo HTML deve ser apresentado na página. O SCSS (KAVINDAK, 2022) (Sassy CSS), por sua vez, é uma extensão que adiciona novas funcionalidades à linguagem CSS. Com o SCSS é possível agrupar melhor seu conteúdo, deixando o código mais curto, limpo e legível, uma vez que é possível aninhar as classes e os identificadores.

O JavaScript (MOZILLA, 2023c) é uma linguagem de programação orientada a objetos, de alto nível e interpretada. Ela é focada para o desenvolvimento Web, permite programação assíncrona e uma boa interação com o HTML, possibilitando, portanto, a criação de páginas dinâmicas, flexíveis, mais rápidas e responsivas. Apesar do seu amplo uso nos navegadores, o JavaScript também é utilizado em outros ambientes sem navegador, como o Node.js (NODE, 2023) e o Adobe Acrobat (ADOBE, 2023).

O TypeScript (LANG, 2023) é uma linguagem de programação de código aberto desenvolvido pela Microsoft. É um *superset* do JavaScript, ou seja, é uma extensão que adiciona recurso

de tipagem estática. O código em TypeScript é convertido para JavaScript, portanto permite rodar em variados ambientes, assim como o JavaScript. Além disso, esses recursos extras proporcionam uma segurança ao código, permitindo que os desenvolvedores encontrem problemas antes mesmo da execução dos ambientes.

O Angular (ANGULAR, 2023) é um *framework* de desenvolvimento Web, de código aberto, mantido pela Equipe Angular do Google. Atualmente é amplamente utilizado para a construção de aplicativos Web modernos e escaláveis. O Angular permite que os desenvolvedores criem interfaces de usuário interativas e responsivas, além de oferecer recursos poderosos, como gerenciamento de estado, injeção de dependência e roteamento.

2.2.2 Back-End

O Back-End (ALURA, 2023) pode ser entendido como tudo o que sustenta a interação entre o usuário e o sistema, como os módulos que recebem as requisições e manuseiam os dados da aplicação até que sejam retornadas as informações necessárias. Em geral, esses módulos trabalham lado a lado com um servidor de banco de dados de onde serão coletados e inseridos dados. Neste emprego o estudante fez o uso da linguagem Java com o *SpringBoot*, além de um banco de dados relacional Oracle utilizando a linguagem SQL. Todos estes conceitos também serão abordados logo abaixo.

O Java (W3SCHOOLS, 2023a) é uma linguagem de programação orientada a objetos, implicando que tudo nesta linguagem é um objeto de uma classe, com exceção dos tipos primitivos (*int*, *float*, *boolean*). Um objeto é a instância da classe a qual pertence, tendo seus atributos e métodos definidos por ela. A classe serve como um molde para os objetos. Por exemplo: uma classe denominada *Pessoa* terá como atributos *nome, idade, cpf*. Cada objeto desta classe terá, portanto, os mesmos atributos nome, idade e CPF, mas não serão, contudo, iguais. Eles têm a mesma forma, todavia o conteúdo é diferente.

O *SpringBoot* (DEVMEDIA, 2023) é uma coleção de módulos, um *framework*, baseado em Java que visa facilitar o desenvolvimento de aplicações empresariais robustas e escaláveis. Ele traz uma abordagem modular, utilizando persistência de dados, segurança, testes, injeção de dependências, entre muitas outras funcionalidades que visam aprimorar a comunicação com o banco de dados, além de gerar uma aplicação mais segura e responsiva.

SQL (W3SCHOOLS, 2023b), ou *Structured Query Language*, é uma linguagem que permite gerenciar e manipular bancos de dados relacionais, possibilitando criar, atualizar, excluir e consultar dados armazenados. A linguagem possui uma série de comandos para a realização destas ações em um banco de dados.

2.2.3 Scrum

O Scrum (SUTHERLAND, 2020) é um *framework*, uma estrutura de gestão com princípios, valores e práticas, que visa agilizar os processos de trabalho, em especial os de desenvolvimento de *software*, melhorando a gestão de projetos e ajudando as equipes a se estruturarem e a trabalharem com maior eficiência.

O componente central do Scrum é a *Sprint* (SUTHERLAND, 2020), um período de cerca de duas semanas, em que ocorre uma iteração, ou pequena entrega, das atividades. É uma boa estratégia de organização, visto que, com entregas contínuas, é possível perceber o trabalho se desenvolvendo já em curto prazo e permite melhorias e correções bem antes do estágio avançado de um *software*.

2.2.3.1 Papéis no Scrum

O Scrum é composto por três papéis essenciais:

- **Product Owner:** é o responsável por compreender e coletar os requisitos do cliente, do mercado, gerando a prioridade para as atividades da equipe.
- **Scrum Master:** é o responsável por orientar a equipe e verificar se a equipe está executando corretamente o Scrum, ajudando na otimização dos processos e recursos.
- **Equipe de Desenvolvimento:** são os responsáveis por desenvolver e completar as atividades. Uma equipe de desenvolvimento eficaz trabalha unida e compartilha, muitas vezes, o mesmo local.

2.2.3.2 Artefatos do Scrum

- **Product Backlog:** é uma lista dos trabalhos a serem desenvolvidos. Pode englobar novos recursos, requisitos, aprimoramentos e correções. O *Product Backlog* deve ser revisado, repriorizado e mantido pelo *Product Owner*.
- **Sprint Backlog:** é uma lista dos trabalho que estão sendo realizados na *Sprint* atual.
- **Incremento:** é um avanço, muitas vezes não muito grande, no produto final, proveniente de uma *Sprint* e que ao longo do tempo, evoluirá até completar uma entrega definitiva e mais completa.

2.2.3.3 Eventos do Scrum

- **Sprint Planning:** é uma reunião com intuito de programar o que será trabalhado na *Sprint*, selecionando os itens diretamente do *backlog*. Além disso, é necessário salientar que o tempo de desenvolvimento das tarefas é levado em consideração, visto que é importante que todos os itens sejam concluídos para se ter uma boa conclusão da *Sprint*. Caso contrário, houve algum problema neste organização, geralmente atrelado a erros de estimativa de tempo de desenvolvimento. finalizados. E é ai que entra o conceito de *Planning Poker* para se estimar o tempo das atividades. Que consiste, durante a *Planning*, de cada desenvolvedor estimar o tempo que ele acredita ser necessário para realização da tarefa. Portanto, cada um informa este tempo simultaneamente, de modo que não haja influência dos demais. Após isso uma discussão é realizada, cada um dando sua opinião sobre sua escolha até que se chegue a um consenso.
- **Daily Scrum:** é uma reunião diária de curta duração que tem o intuito de que todos os membros da equipe sejam atualizados sobre os desenvolvimento de seus colegas de equipe. Para isso, cada membro do time deve responder à três perguntas:

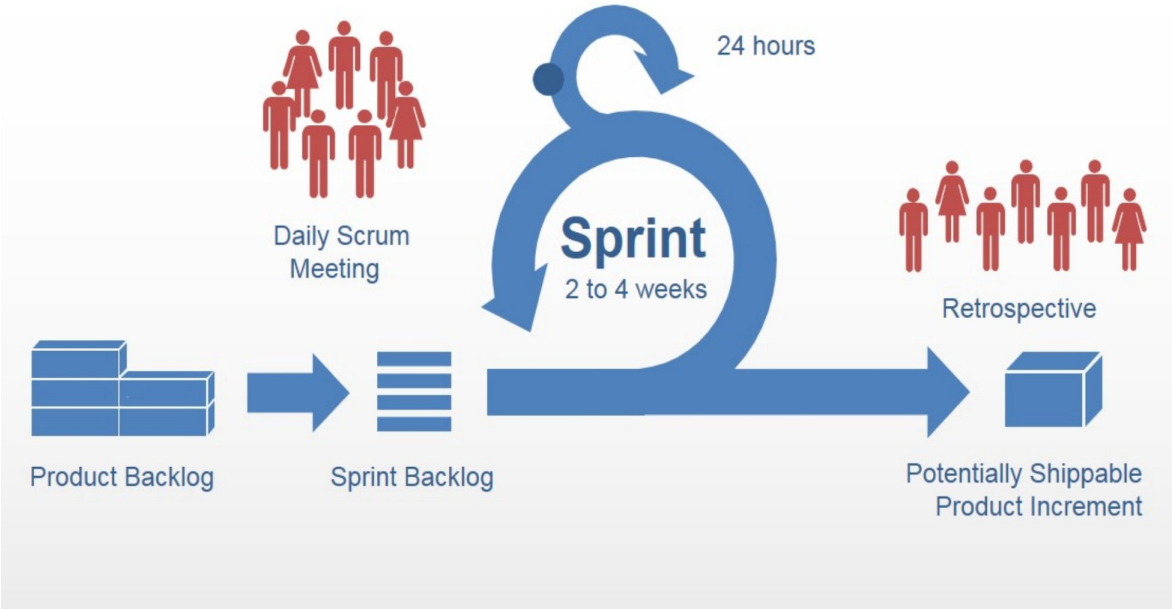
    1. O que eu fiz ontem?
    2. O que eu planejo fazer hoje?
    3. Há algum impedimento?

- **Sprint Review:** é uma reunião que é feita após o fim da *Sprint* e cujo intuito é mostrar, a todas as partes interessadas, o que fora entregue, a fim de que elas deem o *feedback* necessário.
- **Sprint Retrospective:** é uma reunião em que a equipe se reúne para documentar, analisar e revisar tudo que fora feito na *Sprint*. Para isso, muitas vezes se utiliza um quadro em que todos os membros devem anotar três tipos de informações em cartões:

    1. O que deu certo
    2. O que deu errado
    3. Parabenizações a algum membro da equipe

  Tudo isso permite que o time se mantenha em constante crescimento e desenvolvimento, visto que é possível observar os pontos a se melhorarem, além também de manter o que foi.

Figura 2.5 – Scrum



Fonte: Webipedia[2]

A Figura 2.5 ilustra de forma resumida todo o processo do Scrum, desde a reunião de planejamento até a reunião de retrospectiva, contando com o processo da *Sprint* e das *dailies*.

[2] https://webipedia.es/consejos/scrum-tutorial-que-es/

2.2.4 Kanban

O sistema Kanban é um método que utiliza um quadro com diferentes cartões de variadas cores para designar e especificar tarefas a serem realizadas. Dessa forma fica mais fácil visualizar o que precisa ser feito, o que está sendo feito e o que já foi feito. No desenvolvimento de software, devido às variadas entregas em diferentes ambientes, após a tarefa estar concluída é preciso implementá-la nos ambientes de teste e no sistema final, por isso pode se ter mais etapas para indicar em qual etapa do desenvolvimento cada tarefa está.

2.2.5 Azure DevOps

O Azure DevOps (MICROSOFT, 2023) é uma plataforma da Microsoft voltada para a colaboração. Ela conta com diversas funcionalidades, permitindo que uma equipe tenha acesso a versionamento de código, gerenciamento de tarefas, automação da compilação e implantação, segurança do projeto e até mesmo testes automatizados.

2.2.5.1 GIT

O GIT é um popular sistema para controle de versionamento de código para o desenvolvimento de software. Ou seja, ele possibilita o controle de qualquer alteração nos arquivos de um projeto, permitindo um bom rastreio das alterações.

A *Branch* (PEDRO, 2023) é um conceito de ramificação na engenharia de software para controle e gerenciamento de versão de um software, criando um objeto duplicado gerado para desenvolvimento em que se tem uma nova versão, sem que se altere a original.

O *Pull Request* (GITHUB, 2023) é uma solicitação para se mesclar o conteúdo de uma *branch* em outra. Essa solicitação é vista e revisada por outros desenvolvedores para garantir um melhor controle do código a ser implantado.

2.2.6 Docker

O Docker é um programa que trabalha com a containeres e cada um desses containeres contém um ambiente, uma imagem de tudo que é necessário para rodar uma aplicação como código, *frameworks*, programas. É muito útil, especialmente em equipes de desenvolvimento, pois essa

imagem pode ser compartilhada, facilitando que os desenvolvedores tenham um mesmo ambiente e caso uma alteração seja necessária, é mais fácil aplicar para os demais.

2.2.7 DevOps

O DevOps como o nome indica Dev (Desenvolvimento) e Ops (Operations) é um conjunto de ideias, pessoas, ferramentas, práticas e procedimentos que visam integrar todos os processos e operações de desenvolvimento de software com fim de acelerar e otimizar as entregas ao cliente.

3 DESCRIÇÃO DAS ATIVIDADES REALIZADAS

Esta seção descreve as atividades exercidas pelo estudante, desde o seu início, em setembro de 2022 até o começo do ano de 2024. Tendo como foco o crescimento do estudante durante o período, a descrição mostra as diferentes áreas e caminhos percorridos durante o tempo exercido.

3.1 Processo Seletivo

O processo seletivo na BITKA ocorreu em Julho de 2022 e foi um curto período que trouxe bons aprendizados, permitindo que o estudante apresentasse sua trajetória, além, é claro, de poder também realizar uma prova técnica com três questões de programação. Era permitido a escolha dentre três linguagens de programação: C++, C# e Java. A linguagem Java foi escolhida devido ao maior conhecimento e familiaridade.

3.2 Introdução ao Sistema e à Equipe

A princípio, antes de qualquer atividade de desenvolvimento prática, o estudante foi apresentado ao projeto pelo qual atuaria, tendo reuniões com as apresentações iniciais das regras de negócio e do sistema para que assim pudesse familiarizar-se com o ambiente novo.

Este projeto faz parte de um sistema de lógística, responsável desde o processo de extração até a venda do produto. O projeto do qual o estudante faz parte é um subsistema voltado para a otimização, com o intuito de buscar melhores soluções de alocações do produto, transporte e até de processamento dos materiais. Inclusive, esse projeto também é capaz de encontrar compradores para a carga durante o transporte. Portanto, é um projeto grandioso e com um valor altíssimo.

Além disso, a equipe do estudante também atua num processo de migração de um antigo sistema do cliente que se comportava de forma parecida com o atual. Contudo, devido a alta demanda pela utilização desse sistema WEB, esse processo de migração se iniciou e o conteúdo do sistema antiga está sendo transferido para o aplicativo de navegador.

Por conta disso, a atuação da equipe do estudante é dividida em duas frentes, uma de suporte para o sistema atual, focando na correção de *bugs*, utilizando o modelo Kanban e a outra utilizando o Scrum para desenvolvimento de novos conteúdos advindos desta migração.

Vale notar que devido ao tamanho e complexidade deste sistema, há interseções de telas, tabelas e usuários com outros sistemas do próprio cliente e por conta disso há outras empresas prestando serviços para este mesmo cliente, além de empregados do próprio cliente que também atuam. Tudo isso necessita que haja um trabalho conjunto com estas equipes de outras empresas para que uma não dificulte o trabalho da outra.

3.2.1 Configuração do Ambiente

A configuração do ambiente foi o primeiro passo técnico pelo qual o estudante passou, etapa necessária para iniciação das primeiras atividades.

Conforme dito anteriormente, devido ao tamanho do sistema, o projeto que o estudante atua conta com dois repositórios de dois subsistemas que conversam entre si para garantir o funcionamento do sistema. Por conta disso, a configuração de cada subsistema é feita individualmente.

Além da instalação de ambientes de desenvolvimento como IntelliJ[1] e o Visual Studio Code[2], houve também a clonagem dos dois repositórios que o aluno iria atuar utilizando o Git[3]. Mais adiante também foi preciso configurar todo o processo de compilação desses subsistemas, já que um depende do outro para ser executado.

O Docker[4] foi instalado para armazenamento do banco de dados Oracle. A configuração do banco de dados é a etapa que geralmente toma mais tempo devido não só as adversidades, mas também ao processo em si, visto que a forma como é feita esta configuração é executar um *script* SQL para criar todas as tabelas e adicionar todos os dados, linha a linha. Devido ao tamanho do projeto, há mais de 240 tabelas e milhares de linhas de dados, o que toma um tempo considerável.

Além disso, o projeto utiliza o *Weblogic*, um servidor da Oracle para armazenamento da aplicação e toda sua configuração toma um tempo considerável devido a utilização de uma versão mais antiga.

Um exemplo que ilustra bem o problema de contratação de desenvolvedores para agilizar o desenvolvimento na alta demanda, ocorreu aqui, caso em que um novo desenvolvedor foi alocado por cerca de um mês para a equipe. Contudo devido ao alto tempo de configuração do ambiente

[1] https://www.jetbrains.com/pt-br/idea/
[2] https://code.visualstudio.com/
[3] https://git-scm.com/
[4] https://www.docker.com/

e todos os problemas que surgem, este novo integrante gastou quase um mês para configurar o ambiente e o pouco tempo que lhe restava para executar atividades, ele precisava de treinamento e suporte dos outros membros. Ou seja, neste mês em que ele atuou na equipe, a sua alocação mais atrasou o desenvolvimento do que ajudou e não por culpa dele, mas sim devido a essa ideia de urgência em contratar novos desenvolvedores na alta demanda e esquecer que há a necessidade do tempo devido ao treinamento.

3.2.2 Angular - Aplicação Tour of Heroes e Tutorial

O estudante iniciou o *Tour of Heroes* do Angular, um tutorial gratuito disponível pela equipe do Angular, já no começo de suas atividades, sendo esta sua primeira tarefa, um tutorial inicial de Angular responsável por introduzir novos desenvolvedores a este *framework*. Ele auxilia na configuração do ambiente de desenvolvimento do Angular, além de usar o Angular CLI (*command-line interface tool*) para desenvolver uma aplicação. A aplicação final deve apresentar as seguintes funcionalidades básicas, conforme ilustrado pela Figura 3.1:

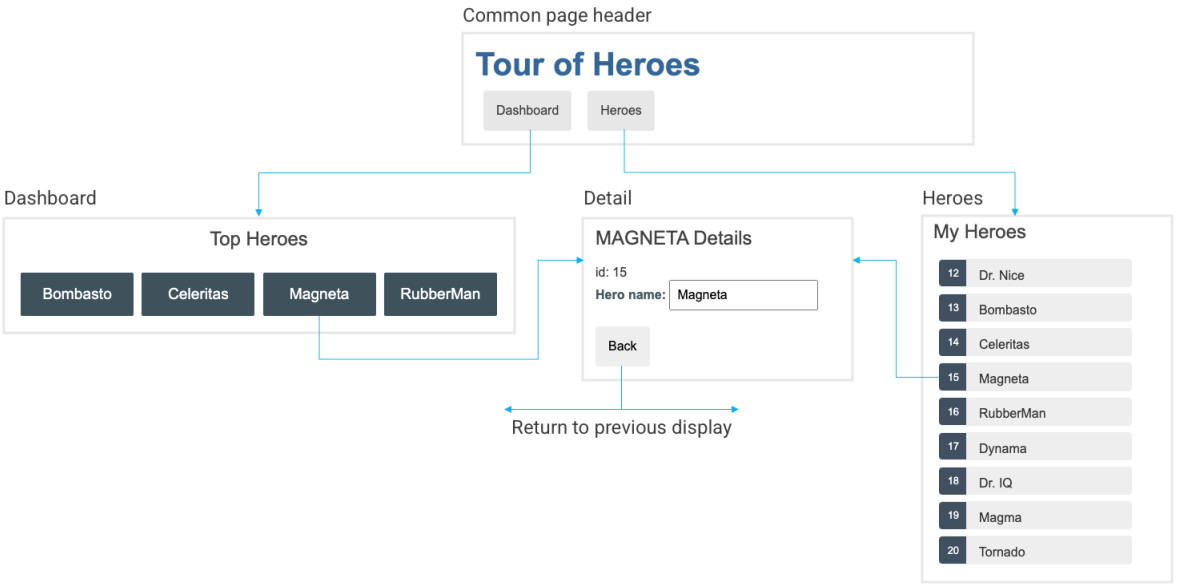
- Retornar uma lista de heróis
- Exibir os heróis em uma lista
- Possibilitar a edição dos detalhes de um heróis selecionado
- Navegar entre diferentes *views* de dados

Com isso, o aluno pôde aprender a execução de alguns comandos no Angular CLI como criação de projetos, execução do projeto, além também da criação dos componentes que são a base da estrutura do Angular, responsáveis por exibir os dados na tela, receber dados do usuários e, inclusive, tomar ações baseadas nas informações recebidas.

O primeiro componente a ser criado foi o *HeroesComponent*, responsável por disponibilizar uma interface em HTML de exibição da lista, conforme o Código 3.15. Já o Código 3.16 é responsável por receber estes dados que serão apresentados na interface HTML, disponibilizando métodos de chamada de serviço como *getHeroes*.

Código 3.1 – heroes.component.html

Figura 3.1 – Relações das Telas




```
<h2>My Heroes</h2>

<ul class="heroes">
  <li *ngFor="let hero of heroes">
  <span class="badge">{{hero.id}}</span> {{hero.name}}
  </li>
</ul>
```

Autor: Juliano Godinho

Código 3.2 – heroes.component.ts

```
import { Component, OnInit } from '@angular/core';

import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
```

```
7    selector: 'app-heroes',
8    templateUrl: './heroes.component.html',
9    styleUrls: ['./heroes.component.css']
10 })
11 export class HeroesComponent implements OnInit {
12   heroes: Hero[] = [];
13
14   constructor(private heroService: HeroService) { }
15
16   ngOnInit(): void {
17     this.getHeroes();
18   }
19
20   getHeroes(): void {
21     this.heroService.getHeroes()
22     .subscribe(heroes => this.heroes = heroes);
23   }
24
25 }
```

Autor: Juliano Godinho

O serviço utilizado no *HeroesComponent* é o presente no Código 3.3. Este serviço de heróis também conta com um serviço de mensagens presente no Código 3.4 que é injetado em seu construtor e é capaz de adicionar mensagens ao *MessageComponent*, funcionando como um *log*, conforme o código 3.5. Além disso, o *HeroService* possui também um método responsável por retornar a lista de heróis para o *HeroesComponent*. Esta lista é proveniente do código apresentado na Figura 3.6.

Código 3.3 – hero.service.ts

```
1 import { Injectable } from '@angular/core';
```

```
2
3  import { Observable, of } from 'rxjs';
4
5  import { Hero } from './hero';
6  import { HEROES } from './mock-heroes';
7  import { MessageService } from './message.service';
8
9  @Injectable({
10   providedIn: 'root',
11 })
12 export class HeroService {
13
14   constructor(private messageService: MessageService) { }
15
16   getHeroes(): Observable<Hero[]> {
17     const heroes = of(HEROES);
18     this.messageService.add('HeroService: fetched heroes');
19     return heroes;
20   }
21 }
```

Autor: Juliano Godinho

Código 3.4 – message.service.ts

```
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class MessageService {
```

```
  messages: string[] = [];

  add(message: string) {
    this.messages.push(message);
  }

  clear() {
    this.messages = [];
  }

  constructor() { }
}
```

Autor: Juliano Godinho

Código 3.5 – messages.component.html

```
<h2>Messages</h2>
<div *ngIf="messageService.messages.length">
    <button type="button" class="clear"
        (click)="messageService.clear()">Clear messages</button>
    <div *ngFor="let message of messageService.messages"> {{message}} </di
</div>
```

Autor: Juliano Godinho

Código 3.6 – mock-heroes.ts

```
import { Hero } from './hero';

export const HEROES: Hero[] = [
  { id: 12, name: 'Dr. Nice' },
```

```
5     { id: 13, name: 'Bombasto' },
6     { id: 14, name: 'Celeritas' },
7     { id: 15, name: 'Magneta' },
8     { id: 16, name: 'RubberMan' },
9     { id: 17, name: 'Dynama' },
10    { id: 18, name: 'Dr. IQ' },
11    { id: 19, name: 'Magma' },
12    { id: 20, name: 'Tornado' }
13  ];
```

Autor: Juliano Godinho

A continuação deste tutorial adicionaria mais funcionalidades ao *HeroService*, além de permitir também a adição, exclusão e edição de heróis, contudo o estudante não pôde concluir esta próxima etapa devido a continuação da configuração do ambiente, com o início da sua primeira atividade. Esta atividade exigiu foco em outra tecnologia, o Spring. Entretanto, vale ressaltar aqui que apesar da não finalização deste tutorial, ele ainda foi responsável por apresentar ao aluno com riqueza de detalhes o contexto do *Frond-End* e também do Angular, ferramenta amplamente utilizada no projeto.

3.2.3 Desenvolvimento das Atividades

A equipe de desenvolvimento se divide entre suporte e migração ao longo de cada *Sprint*. E apesar dessa divisão da atuação, todas pessoas da equipe, independente da frente de atuação, participam dos ritos do Scrum. O que diferencia o suporte é que os cartões de tarefas não são incluídos no *Sprint*, mas sim no quadro Kanban.

As pessoas responsáveis por estas tarefas atuam dando suporte tanto em código antigo quanto em código novo, implementado pela migração. O formato de organização do quadro é o Kanban, lugar em que os cartões tem o tempo mensurado após cada dia de trabalho, ou seja, após a pessoa passar um dia trabalhando em algum cartão, ela acrescentará um ponto a ele, indicando o tempo trabalhado.

Já as pessoas atuantes no processo de Migração trabalham utilizando o Scrum, seguindo todos os ritos desde o *Planning* até a *Retrospective*. Por conta disso, estimativas são feitas através da *Planning Poker* por parte dos desenvolvedores para tentar mensurar o tempo a ser gasto em cada atividade.

O quadro da*Sprint* conta com um gráfico de linhas que ilustra o avanço do *Sprint* de acordo com o esforço executado pelos desenvolvedores dia após dia. O gráfico também possui uma linha para a projeção, indicando quanto de esforço diário precisa ser cumprido para que a *Sprint* seja concluída por completo.

3.2.4 Primeira Atividade Após o Treinamento

Após a finalização do treinamento do *Front-End*, o estudante esperava que um tutorial semelhante fosse realizado para a parte do *Back-End*. Entretanto, chegou-se a conclusão de que uma tarefa real era melhor para desenvolvimento e compreensão dos *frameworks* e do projeto.

A primeira atividade resumiu-se na criação de uma nova tela com uma tabela que recebe dados de duas entidades. Por conta disso, o estudante precisou criar uma tabela no *Front-End*, duas entidades no Spring e duas tabelas no banco de dados.

O projeto possui um biblioteca para o Java que visa facilitar a criação de tabelas no *Front-End*. Contudo esta atividade possui uma tabela dinâmica, ou seja, dependendo dos dados inseridos, as colunas variam. Devido a isso, foi necessário realizar a criação da tela através do Angular, o que se tornou uma tarefa de maior complexidade, tomando um bom tempo do estudante, mas gerando bons aprendizados.

3.2.5 Azure DevOps

O Azure DevOps é a plataforma da Microsoft para DevOps. Ele conta com uma seção de Repositórios, o que permite uma boa segurança para os códigos das aplicações. O estudante pôde aqui ampliar seu conhecimento em *GIT*, resultando em um grande crescimento, devido ao contato com outros integrantes da equipe.

A estrutura do projeto é composta por três ambientes: Desenvolvimento, Homologação e Produção. As *branches* respectivas à estes ambientes contavam com proteção, o que acarretou em um bom desenvolvimento e na utilização de *Pull Requests* por parte do estudante.

Pela tradução, os *Pipelines* são a tubulação que transita o fluxo de implantação de melhorias para o sistema. É através dele, pelo próprio *Azure* em que serão construídas as mudanças de código e então implantadas nos devidos ambientes, tudo isso de forma rápida e eficaz, permitindo entregas contínuas.

O Azure também conta com uma seção chamada Quadros com inúmeras funcionalidades permitindo trabalhar com o uso de cartões de tarefas e quadros tanto no modelo do Kanban quanto do Scrum. Além disso, há uma seção para a *Sprint* e *Retrospectives*.

3.2.6 Cotidiano

A rotina do estudante é repleta de atividades diversas, desde reuniões sobre negócios e de planejamento, até atividades de desenvolvimento ou de resolução de problemas.

A cada duas semanas, é feita a *planning* para se decidir o que será feito na próxima *Sprint*, além de também decidir qual membro da equipe de desenvolvimento será alocado para o suporte, trabalhando no modelo de Kanban. Lembrando também que esta alocação não é sempre íntegra, portanto quem está no suporte pode ainda ter um pouco alocado para trabalhar nas atividades da *Sprint*.

Todos os dias, às 09h30, é realizada a reunião diária e tem duração em média de 30 minutos, cujo intuito é a equipe se manter informada das tarefas desenvolvidas.

No final da *Sprint*, há uma reunião de retrospectiva. Nela, se utiliza um quadro com três colunas. Na primeira se coloca os cartões de tudo que ocorreu bem, já segunda, tudo que ocorreu mal e por fim as parabenizações aos membros da equipe. Aqui se pontua tudo, desde problemas no desenvolvimento de tarefas até dificuldades em contato com os clientes, por isso é uma reunião de extrema importância que acresce em muito no progresso da equipe.

Uma das atividades que o aluno melhor se destacou foram as relacionadas a otimização e reestruturação de código, especialmente de trechos que consumiam muito mais tempo do que o esperado. Um destes exemplos era uma requisição que era executada ao abrir uma tela e demorava cerca de 15 minutos. O aluno após identificar o problema, cuja causa era um trecho de código lento e que não era nem sequer utilizado nesta tela, pôde reduzir o tempo para menos de 30 segundos.

Devido a enorme complexidade de todo o sistema do qual o aluno atua, havia muitos dados advindos de outros sistemas. Por conta disso várias tarefas estavam ligadas a busca de dados vindos

de fontes externas, o que permitiu ao estudante um avanço em seu conhecimento em diferentes espaços.

3.2.7 Cenários

Nesse projeto em que o estudante trabalha, os usuários trabalham em uma parte do sistema chamada de cenário que é uma cópia do sistema com dados salvos em memória. De forma que há telas específicas para esses dados copiados e até novas telas para se trabalhar também. Tudo isso com intuito de que ali ele possua um ambiente em que ele possa fazer as alterações que desejar e verificar possíveis casos e cenários para diversas ocorrências. Inclusive o usuário exportar esse ambiente de trabalho em um arquivo para ser importado futuramente. Tudo isso é feito sem afetar os dados do banco.

Por conta do tamanho do sistema, há um filtro na criação destes cenários, de modo que apenas alguns dados de algumas tabelas são selecionados para serem salvos em memória na sessão do usuário. Já que se copiasse tudo existente no banco, os servidores seriam ocupados com dados desnecessários gastando recursos em vão. Tendo isso em mente, o estudante ficou responsável pela criação de um novo tipo de cenário, cujo filtro seria diferente. Este novo possuiria todos os dados do já existe mais alguns dados definidos pelos usuários. A grande dificuldade desta tarefa era que este sistema de criação de cenários não foi desenvolvido de maneira dinâmica, ou seja, não foi feito com intuito de criar novos cenários no futuro, portanto, o estudante precisou refatorar o código para não só criar este novo cenário, mas também facilitar a criação de outros no futuro.

3.2.8 Experiência Adquirida

Com o passar do tempo, o aluno adquiriu bastante experiência no desenvolvimento *Front-End*, em parte, pela utilização do Angular tanto nas atividades desenvolvidas, quanto pelos conhecimentos passados por colegas de trabalho.

As atividades iniciais foram muito relacionadas a implementações com a utilização da biblioteca AG Grid do Angular, voltada inteiramente para criação de tabelas dinâmicas, flexíveis e responsivas, proporcionando um bom começo e uma boa base para os conhecimentos subsequentes.

Na sequência, o estudante entrou em contato com inúmeros processos mais avançados do *Front-End*, com a criação de inúmeros componentes e telas, serviços diversos para chamadas de

requisições do *Back-End* e até mesmo programação assíncrona. Tudo isso convergiu para uma boa formação neste área, o que permitiu um grande crescimento profissional.

O estudante já possuía uma boa base na parte do *Back-End*, já que houve um grande contato com a linguagem Java e com a linguagem PHP durante a disciplina de Engenharia de Software. O que havia de novo era o *framework* Spring, que se mostrou um grande diferencial e uma grande ferramenta para desenvolvimento. Toda estrutura MVC, aprendida durante o curso, foi rearranjada para um novo patamar, permitindo, assim, um progresso interessante.

Por fim, com relação a Banco de Dados, também pôde-se notar uma boa evolução no nível das atividades. A princípio se tinha a criação de Tabelas e Sequências, enquanto que no final já se trabalhava com *Queries* mais complexas que permitem consultas mais eficazes e um melhor uso do banco.

4 CONCLUSÃO

Durante todo o período de trabalho, o estudante pôde contar com o auxílio de inúmeros profissionais, de diversas áreas e níveis de experiência. Com isso em vista, foi proporcionado uma alta gama de conhecimento, aprendizado e experiência tanto na parte de desenvolvimento quanto na parte de negócios.

A equipe toda sempre foi responsável por ajudar e a ensinar. Desde o início, sempre houve a preocupação no aprendizado a ser adquirido e por conta disso, os colegas se dedicavam a estarem presentes e a aconselharem o estudante.

Outro ponto a ressaltar é de que durante o período exercido, também houve mudanças na equipe, com a entrada de novos membros e possibilitando assim o contato com novos desenvolvedores, acarretando na aquisição de novas experiências e conhecimentos.

Observa-se, portanto, uma boa aplicação dos conhecimentos adquiridos ao longo do curso de Ciência da Computação. Vale ressaltar aqui algumas das disciplinas de maior contribuição para o sucesso neste trabalho:

- **Introdução aos Algoritmos (GAC124):** esta disciplina é a base de toda programação vista no curso e se mostrou essencial para um bom aprendizado e para uma boa aplicação durante o trabalho do estudante.

- **Estrutura de Dados (GAC108):** a continuação da disciplina anterior, esta já introduz o conceito de orientação a objetos, por meio das classes, um ponto alto trabalhado durante este trabalho. Também auxiliou quanto as atividades realizadas durante o processo seletivo, já que estas se assemelhavam bastante as questões realizadas na disciplina.

- **Introdução a Sistemas de Banco de Dados (GCC214):** a primeira disciplina a tratar dos bancos de dados, envolvendo desde a criação de diagramas até a de um banco.

- **Sistemas Gerenciadores de Banco de Dados (GCC175):** a sequência da disciplina anterior, trouxe muitos pontos positivos a serem tratados durante este trabalho.

- **Práticas de Programação Orientada a Objetos (GAC106):** uma sequência a Estrutura de Dados que além de focar na parte de orientação a objetos, também utiliza a linguagem Java, a mesma do Spring, causando um grande impacto na execução das atividades deste trabalho

- **Engenharia de software (GCC188):** é a disciplina que mais gerou frutos durante este trabalho. Visto que ela aborda toda a estruturação do software, desde uma documentação completa até a parte de banco dados, desenvolvimento e testes. Foi esta disciplina, o sustentáculo para todo o conhecimento trabalhado e adquirido ao longo deste trabalho.

- **Metodologia de Pesquisa (GCC220):** responsável por auxiliar na criação deste relatório, sendo, portanto, de suma importância para tal.

Nota-se, consequentemente, que o curso, por completo, desde os primeiros períodos até os últimos, consolidaram toda uma extensa base de conhecimento que foi aplicada pelo estudante durante todo o período de trabalho na BITKA.

Cabe aqui dizer que um ponto a ser abordado no curso futuramente, é o princípio do Código Limpo. Segundo (MARTIN, 2009), um código limpo pode economizar várias horas para uma equipe de desenvolvimento. É uma coisa que, a princípio se parece simples, entretanto se mostra um verdadeiro desafio para os desenvolvedores, convidando-os a tentar sempre empreender uma boa estruturação em seu código. Por isso, é importante mencionar que seria um ótimo avanço ao curso de Ciência da Computação da Universidade Federal de Lavras, a adição, mesmo que concisa, dos princípios do Código Limpo, especialmente nas disciplinas iniciais, para que o aluno já tenha uma base consolidada quanto a escrita de seu código.

Por fim, vale aqui ressaltar que todo o período trabalhado, até os dias de hoje, na BITKA Analytics foram intensos, completos e de indescritíveis aprendizados. O estudante pôde aplicar, de forma completa, tudo que fora aprendido ao longo do curso e agora pode trilhar este novo caminho, iniciado dentro da universidade, em rumo a um desenvolvimento profissional promissor.

REFERÊNCIAS

ADOBE. **Adobe Acrobat**. 2023. (Acessado em 03/07/2023). Disponível em: <https://www.adobe.com/br/acrobat.html>.

ALURA. **O que e Front-End e Back-End**. 2023. (Acessado em 03/07/2023). Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>.

ANGULAR. **Angular**. 2023. (Acessado em 03/07/2023). Disponível em: <https://angular.io/>.

BERNES-LEE, T. **DevDocs HTML Documentation**. 2018. (Acessado em 03/07/2023). Disponível em: <https://devdocs.io/html/>.

DEVMEDIA. **Como começar com Spring?** 2023. (Acessado em 03/07/2023). Disponível em: <https://www.devmedia.com.br/exemplo/como-comecar-com-spring/73>.

GITHUB. **Sobre solicitação de pull**. 2023. (Acessado em 03/07/2023). Disponível em: <https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests>.

KAVINDAK, S. **The definitive guide to SCSS**. 2022. (Acessado em 03/07/2023). Disponível em: <https://blog.logrocket.com/the-definitive-guide-to-scss/>.

LANG, T. **What is TypeScript?** 2023. (Acessado em 03/07/2023). Disponível em: <https://www.typescriptlang.org>.

MARTIN, R. C. **Código Limpo: Habilidades Práticas do Agile Software**. [S.l.]: São Paulo: Alta Books, 2009. ISBN 978-8576082675.

MICROSOFT. **Azure DevOps**. 2023. (Acessado em 03/07/2023). Disponível em: <https://azure.microsoft.com/pt-br/products/devops/>.

MOZILLA. **CSS**. 2023. (Acessado em 03/07/2023). Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.

MOZILLA. **HTML: Linguagem de Marcação de Hipertexto**. 2023. (Acessado em 03/07/2023). Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.

MOZILLA. **JavaScript**. 2023. (Acessado em 03/07/2023). Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>.

NODE. **Node**. 2023. (Acessado em 03/07/2023). Disponível em: <https://nodejs.org/en>.

PEDRO, W. **O que é branch em programação?** 2023. (Acessado em 03/07/2023). Disponível em: <https://tecnoblog.net/responde/o-que-e-branch-em-programacao/>.

SUTHERLAND, K. S. . J. **The Scrum Guide**: The definitive guide to scrum: The rules of the game. [S.l.: s.n.], 2020.

W3SCHOOLS. **Java Tutorial**. 2023. (Acessado em 03/07/2023). Disponível em: <https://www.w3schools.com/java/>.

W3SCHOOLS. **SQL Tutorial**. 2023. (Acessado em 03/07/2023). Disponível em: <https://www.w3schools.com/sql/>.