CENTRO DE INFORMÁTICA

UNIVERSIDADE FEDERAL DA PARAÍBA

PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: ESTUDO EXPLORATÓRIO APLICADO A UM SISTEMA DE CONTA DIGITAL COM ÊNFASE NO BACKEND

LETÍCIA SOUSA LEITE

JOÃO PESSOA-PB

2024

LETÍCIA SOUSA LEITE

PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: ESTUDO EXPLORATÓRIO APLICADO A UM SISTEMA DE CONTA DIGITAL COM ÊNFASE NO BACKEND

Trabalho de conclusão de curso apresentado ao curso de Engenharia da Computação da Universidade Federal da Paraíba - UFPB, como pré-requisito para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Alisson Vasconcelos de Brito

JOÃO PESSOA-PB

2024

Catalogação na publicação
Seção de Catalogação e Classificação

L533p Leite, Leticia Sousa.
Processo de desenvolvimento de software: estudo exploratório aplicado a um sistema de conta digital com ênfase no backend / Leticia Sousa Leite. - João Pessoa, 2024.
36 f. : il.

Orientação: Alisson Brito.
TCC (Graduação) - UFPB/CI.

1. Desenvolvimento. 2. Engenharia de Software. 3. Ferramenta. I. Brito, Alisson. II. Título.

UFPB/CI CDU 004.4

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA

UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Engenharia da Computação intitulado **Processo de desenvolvimento de software: Estudo exploratório aplicado a um sistema de conta digital com ênfase no backend** de autoria de Letícia Sousa Leite, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Allison Brito

Universidade Federal da Paraíba - UFPB - Orientador

Prof. XXXX XXXXX XXXXX

Universidade Federal da Paraíba - UFPB

Prof. Dr. XXXXXX XXXXXX XXXXX

Universidade Federal da Paraíba - UFPB

João Pessoa, XX de XXXXX de 2024

AGRADECIMENTOS

Antes de tudo agradeço a Deus, por ter sido a minha base e minha força durante toda essa jornada até aqui, toda honra e glória sejam dadas a Ele. Quero agradecer a minha família, em especial minha mãe, que colocou todas suas orações em minha vida e não mediu esforços para que eu concluísse minha graduação. Quero agradecer também a minha tia, Estelita Sousa, que durante muitos momentos de dificuldade foi o alicerce na minha vida e da minha mãe.

Quero agradecer também ao meu noivo, Guilherme Mendes, que esteve comigo desde a decisão por iniciar o curso de Engenharia da Computação, e me apoiou durante todo o percurso, em especial nessa reta final. Agradeço também a minha amiga Viviane Maria, que juntamente com sua família, foram e são uma segunda família que Deus enviou para cuidar de mim aqui em João Pessoa.

Não poderia deixar de agradecer aos meus colegas de curso e amigos Stenio Ellison e Elaine Cristina, por terem sido tão parceiros do início ao fim do curso, enfrentando diversas dificuldades juntos e conquistando diversos objetivos juntos também.

Agradeço ao meu professor e orientador Alisson Brito, por todo o suporte e orientação, e por ter sido o responsável pela disciplina que inspirou meu tema de trabalho de conclusão de curso.

Por fim, agradeço a todos os professores e colegas que contribuíram para minha formação acadêmica.

RESUMO

O processo de desenvolvimento de um software, desde sua idealização até sua concretização, não é uma tarefa fácil, e exige diversos conhecimentos em várias áreas para que o sistema a ser desenvolvido seja entregue de forma satisfatória. A Engenharia de Software possui metodologias para auxiliar no desenvolvimento de sistemas, como o modelo em cascata, um dos métodos mais simples de planejamento de um software, que consiste em etapas sequenciais, hierárquicas e bem definidas. Diante deste contexto, este trabalho tem como objetivo demonstrar o processo de desenvolvimento de um software, aplicado a uma pequena parte de uma conta digital, utilizando ferramentas que contribuem para que as etapas se tornem ainda mais simples de serem executadas e visualizadas, focando no planejamento e levantamento de recursos e infraestrutura do backend. Para alcançar este objetivo foram seguidas as fases do modelo de desenvolvimento em cascata, fazendo pesquisas bibliográficas em cada uma dessas fases para trazer boas opções de ferramentas e serviços que poderiam ser aplicados a uma conta digital. Por fim, o trabalho faz uma análise dos resultados obtidos durante o planejamento do sistema seguindo os processos de desenvolvimento, e conclui que utilizar as principais etapas do processo de desenvolvimento de um software e unir isso a boas ferramentas e serviços é uma ótima maneira de planejar um sistema. Com isso, este trabalho contribui para comunidade acadêmica e profissionais de TI ainda com pouca experiência na área de Engenharia de Software, como um guia de desenvolvimento de software simplificado.

PALAVRAS-CHAVE: Desenvolvimento, Engenharia de Software, Ferramentas, Software, Sistema, Conta Digital.

ABSTRACT

The process of developing software, from conception to completion, is not an easy task and requires a wide range of knowledge in different areas to ensure that the system to be developed is delivered satisfactorily. Software Engineering has methodologies to help develop systems, such as the waterfall model, one of the simplest methods for planning software, which consists of sequential, hierarchical and well-defined stages. Given this context, this work aims to demonstrate the software development process, applied to a small part of a digital account, using tools that help make the stages even easier to execute and visualize, focusing on planning and gathering resources and infrastructure for the backend. To achieve this goal, the phases of the waterfall development model were followed, and bibliographic research was carried out on each of these phases to provide good options for tools and services that could be applied to a digital account. Finally, the work analyzes the results obtained during the planning of the system following the development processes, and concludes that using the main stages of the software development process and combining this with good tools and services is a good way to plan a system. Therefore, this work contributes to the academic community and IT professionals who still have little experience in the area of software engineering, as a simplified software development guide.

KEYWORDS: Development, Software Engineering, Tools, Software, System, Digital Account.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE SÍMBOLOS E ABREVIATURAS

IEEE	Institute of Electrical and Electronics Engineers
AWS	Amazon Web Services
SGBDs	Sistemas Gerenciadores de Bancos de Dados
API	Application Programming Interface
UML	Unified Modeling Language
LGPD	Lei Geral de Proteção de Dados Pessoais
RDS	Relational Database Service
TI	Tecnologia da Informação
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

O processo de desenvolvimento de um software é um tema que faz parte da rotina de profissionais de tecnologia da informação de qualquer área, pois trata-se de todas as etapas que serão necessárias para consolidar um projeto de software, desde sua ideia até sua entrega final e manutenção. Para que um sistema seja desenvolvido de forma categórica, as principais etapas do processo precisam ser seguidas, sendo algumas delas o levantamento de requisitos, o projeto, a implementação, os testes e a entrega. Com o passar dos anos esses processos têm passado por diversas mudanças que vêm para melhorar a qualidade dos softwares, se adaptando aos diferentes tipos de sistema e visando garantir, dentre muitas coisas, eficiência, diminuição de custos e qualidade para os produtos (ENGHOLM JUNIOR, 2010).

A Engenharia de Software é a área responsável por estudar esse conteúdo, ela aplica princípios da engenharia no desenvolvimento de software, e fornece metodologias que visam garantir que os sistemas sejam projetados de maneira ágil, eficiente e segura (WAZLAWICK, 2019). Ela abrange uma grande variedade de tópicos, como a gestão e manutenção de projetos, e vai além de simples programas de computadores, pelo contrário, atua em sistemas de informações complexos, sendo eles empresariais, bancários e etc. Este trabalho será focado na área de atuação da indústria de tecnologia, contudo, a Engenharia de Software não se limita a esta área apenas, atuando também em diversos setores da sociedade moderna, como governamentais, financeiros, educacionais e de saúde.

A vista disso, consegue-se entender que sintetizar todo o trabalho necessário para criar um projeto de software do início ao fim não é uma tarefa fácil e nem rápida, os processos de desenvolvimento ajudam a saber por onde começar, quais ações serão necessárias durante o desenvolvimento e como finalizar. Mas pode-se ir além disso e explorar, durante essas etapas, ferramentas e recursos que tornem o projeto como um todo uma tarefa menos complexa, mais fluida e consequentemente, mais rápida.

Um bom exemplo de sistema complexo e atualmente crescente no mercado da tecnologia, são as contas digitais. O mercado financeiro está passando por uma etapa significativa de adaptação, bancos físicos são cada vez menos procurados e as contas online vêm ganhando espaço. As Fintechs, empresas com tecnologias voltadas a soluções para serviços financeiros, estão trazendo uma nova cultura financeira, transformando a relação entre bancos e clientes, tornando possível maior inclusão no meio e nos colocando diante de novos conceitos, produtos e serviços digitais, que fazem parte do nosso dia a dia, redefinindo

aspectos relacionados à eficiência, experiência de usuário e muitos paradigmas que estávamos acostumados a usar (DINIZ, 2021). Contudo, contas digitais são sistemas que exigem uma grande qualidade de segurança, disponibilidade, eficiência e rastreabilidade, pois carregam dados sensíveis dos seus usuários e precisam estar sempre disponíveis para utilização. Dessa forma, empresas do ramo financeiro, precisam investir em processos de desenvolvimento bem consolidados, para que consigam entregar todas as exigências que pedem as contas digitais.

Diante disso, é possível entender que seguir as principais práticas de processos de desenvolvimento de software e usufruir de ferramentas para auxiliar suas etapas, além de garantir a qualidade, podem elevar o grau de produtividade dos projetos e ainda tornar tarefas que pareciam muito complexas, mais simples.

Nesse sentido, o presente estudo tem como principal objetivo demonstrar os principais processos de planejamento e desenvolvimento de um software, utilizando como exemplo uma parcela do que seria um sistema de conta digital, e usufruindo de ferramentas que irão facilitar as suas etapas, servindo como um guia de planejamento e desenvolvimento de software e trazendo sugestões de recursos que torne o processo mais prático e ágil. De forma mais específica, será apresentada uma descrição do o que é e como funciona uma conta digital e porque é um bom objeto de estudo para o tema tratado neste trabalho. Neste trabalho, será focada em alguns dos principais requisitos que tem uma conta digital, pois trata-se de um sistema muito complexo, e o objetivo aqui é entregar um guia de como planejar projetos de software. Também será focado apenas no planejamento do backend do serviço, umas vez que o desenvolvimento de frontend é também um tópico bastante abrangente e cabe um estudo voltado apenas a este tema, levantando questões como design acessível e objetivo. Além disso, serão discutidos os principais processos de desenvolvimento, segundo a Engenharia de Software, de forma mais detalhada. Serão também demonstradas, por meio de exemplos aplicados ao projeto de conta digital, a aplicação das principais etapas do processo de desenvolvimento, antes da fase de implementação, trazendo durante essas demonstrações, aplicativos e recursos que auxiliaram cada etapa. As etapas que envolvem a programação de fato do sistema não serão aplicadas neste trabalho, mas serão apresentadas ferramentas que podem auxiliar essas fases também.

A metodologia utilizada compreendeu fazer pesquisa de trabalhos e artigos relacionados ao tema, e usar livros e materiais a respeito da Engenharia de Software para aplicar as principais etapas do processo de desenvolvimento ao projeto de conta digital. Para cada etapa foi apresentada a ferramenta que foi utilizada para auxiliar o desenvolvimento,

mostrando assim, uma visão geral de como seria planejar um projeto de software desde o início.

1.1 MOTIVAÇÃO

A motivação para este trabalho foi conseguir trazer um guia de planejamento e desenvolvimento de um software, tendo como objeto de estudo uma conta digital, por se tratar de um sistema completo e atual, onde cada parte do sistema deve ser pensada cautelosamente. Elaborar um sistema do zero envolve uma série de processos e áreas de estudo, e a quantidade de informações a serem estudadas vem aumentando cada dia mais, dessa forma, profissionais com pouca experiência podem enfrentar dificuldades ao projetar novos sistemas. Este trabalho vem então como um guia prático, reunindo os principais processos de desenvolvimento e trazendo ferramentas e recursos que auxiliam nesses processos, sendo estes recursos em até certo ponto, disponíveis gratuitamente.

1.2 OBJETIVOS

1.2.1 Geral

Demonstrar o processo de desenvolvimento de um sistema de conta digital utilizando ferramentas que irão facilitar suas etapas, servindo como um guia de planejamento de software.

1.2.2 Específicos

1) Descrever o que é e como funciona uma conta digital e porque é um bom objeto de estudo para o tema tratado neste trabalho;
2) Descrever o que é Engenharia de Software e quais os principais processos de desenvolvimento segundo ela;
3) Demonstrar como funciona cada umas das principais etapas do processo de planejamento de um software;
4) Usar tecnologias e ferramentas que simplifiquem os processos;

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido em cinco capítulos. O primeiro capítulo contextualiza o tema apresentado, bem como o que motivou a sua escolha e os objetivos a serem alcançados.

O segundo capítulo traz todo o referencial teórico necessário para pleno entendimento do estudo. O capítulo três, por sua vez, descreve as atividades decorridas, desde o levantamento da análise de requisitos até o planejamento de implantação do sistema, enquanto que o capítulo quatro apresenta os resultados obtidos nele.

Por fim, o quinto e último capítulo, apresenta uma visão geral dos resultados obtidos e perspectivas de desenvolvimentos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta de forma resumida conteúdos que fundamentam este estudo, trazendo o que outros autores levantaram sobre o tema, dessa forma, o leitor conseguirá entender claramente o contexto.

2.1 CONTA DIGITAL

A história dos meios de pagamentos é extensa e sofreu diversas evoluções com o passar dos anos, antes das contas digitais, os bancos possuíam um sistema burocrático, com uma necessidade de um grande número de funcionários e a dependência de uma grande quantidade de recursos materiais e financeiros. Além disso, as transações demandavam um tempo de duração alto e os sistemas eram muito inflexíveis. Os clientes só podiam realizar transações nas agências bancárias onde possuíam conta, apenas nos horários de atendimento disponibilizados pelos bancos e sempre por meio de algum funcionário da agência (DO NASCIMENTO, 2020).

Os primeiros avanços tecnológicos nos bancos vieram então através dos caixas eletrônicos, sistemas computadorizados que permitem que sejam realizados saques, depósitos e consultas de saldo sem necessidade de um funcionário, sendo estes demandados apenas no período de adaptação dos clientes. Os caixas eletrônicos foram um sucesso e são amplamente utilizados ainda atualmente, principalmente os de funcionamento permanente, conhecidos como caixas 24 horas (DO NASCIMENTO apud LINDGREN JR 2001).

Ultimamente a sociedade tem vivenciado uma transição nos sistemas bancários, que agora deixam de se tornarem atividades presenciais e passam a serem totalmente virtuais, realizadas através de celulares, tablets e computadores, de forma muito rápida, bastando apenas ter uma conexão com a internet. Funcionários também podem agora atender seus clientes através de telefonemas e “chats”, podendo atender um maior número de pessoas. Já as

instituições financeiras economizam instalações, funcionários, materiais e custos de manutenção.

Diante do exposto, podemos dizer que as contas digitais são a nova tendência do mercado financeiro, pois estas além de oferecer serviços mais rápidos e práticos, também possuem menores tarifas quando comparadas a alguns tipos de transações bancárias. Com o sucesso das contas digitais, houve um aumento nos investimentos em modernização bancária, expandiu-se o uso de internet banking e o surgimento das Fintechs.

As Fintechs, por sua vez, disponibilizam os meios de pagamento digitais, como transferências, pagamentos e consultas de saldo por intermédio de APIs, e tudo isso é apenas algumas das partes do grande ecossistema de uma conta digital, que por esse motivo, é o objeto de estudo deste trabalho, onde serão aplicadas as técnicas de desenvolvimento de software, tema que será explorado no próximo tópico deste capítulo.

2.2 PROCESSO DE DESENVOLVIMENTO E ENGENHARIA DE SOFTWARE

Softwares de computador são artefatos que consistem em instruções executáveis pelos computadores, onde descrevemos a lógica que queremos aplicar a algum problema. As pessoas desenvolvedoras de software são os responsáveis por criar e dar suporte a esses conjuntos de instruções. Além disso, diferentemente do hardware, o software não passa por desgaste. Enquanto o hardware pode passar por degradação de componentes, onde estes passam por uma substituição, o software pode sofrer apenas alterações, pois um problema no funcionamento de um software implica em erro em seu projeto, o que leva a um replanejamento do sistema e necessidade de suporte de desenvolvimento (PRESSMAN; MAXIM, 2021).

Os softwares de computadores passaram por um grande crescimento e com isso ocorrências de problemas relacionados a manutenções, correções e adaptações passaram a ser mais frequentes (LESSA, 2009). Devido a isso, a utilização de técnicas de Engenharia de Software, para aperfeiçoar os sistemas passou a se tornar cada vez mais necessária. Segundo o IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos), a Engenharia de Software é “A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software” (PRESSMAN; MAXIM, 2021).

De acordo com Pressman e Maxim (2021), a base da Engenharia de Software é constituída por etapas de processos que possibilitam o desenvolvimento racional e dentro do prazo estabelecido.

O processo de software pode ser definido como:

> "O processo de software constitui a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos artefatos (modelos, documentos, dados, relatórios, formulários, etc.), são estabelecidos marcos, a qualidade é garantida e as mudanças são geridas de forma apropriada" (PRESSMAN; MAXIM, 2021).

A engenharia de software tem alguns métodos de desenvolvimento que, quando aplicados de maneira eficiente, nos ajudam a projetar sistemas complexos. Dentre esses métodos, existem o modelo em cascata, prototipação, Scrum, Extreme Programming (XP), e outros. Os modelos cascata e prototipação são metodologias tradicionais do desenvolvimento de software. Enquanto os modelos Scrum e XP tratam-se de metodologias ágeis.

O método de desenvolvimento em cascata tem como objetivo um desenvolvimento linear e sequencial, onde uma etapa é iniciada após o fim da outra. Uma das maiores vantagens desse modelo é o alto grau de gerenciamento em cima do projeto, onde prazos podem ser estabelecidos de forma concisa, seguindo a ordem de cada procedimento estritamente. As principais etapas que este método de desenvolvimento traz incluem o levantamento de requisitos, modelagem do projeto, implementação, testes e manutenção. O método de prototipação, por sua vez, é um modelo evolucionário, onde são criados protótipos de uma versão inicial do sistema, para que seja usada para demonstração, experimentos e descobertas de novos requisitos. Esse modelo pode ser utilizado quando o cliente sabe as funcionalidades que deseja para o sistema, contudo, não sabe ainda os requisitos específicos que serão necessários para seu funcionamento (RIBEIRO, 2020).

O Scrum é uma metodologia ágil, e uma das mais utilizadas atualmente, é um processo multifuncional, onde o processo de desenvolvimento é dividido em pequenas etapas, chamadas de sprints, e a equipe é dividida em papéis de responsabilidade. Alguns dos papéis no Scrum são o de Product Owner, que tem a responsabilidade de ter uma visão geral sobre o produtos, gerenciar backlogs e ser o representante do cliente no produto. O Scrum Master, que é o responsável por gerenciar o ciclo do Scrum e o time de desenvolvimento, que é formado pelos profissionais que atuaram na produção do sistema. Por fim, o método XP tem como

objetivo levar ao extremo práticas reconhecidas como boas na Engenharia de Software e foca em comunicação, cooperatividade e feedbacks de clientes, se mantendo à disposição para alterar requisitos conforme o sistema muda com o tempo (RIBEIRO, 2020).

Neste trabalho, estaremos focando apenas no modelo em cascata, que também pode ser chamado de modelo linear ou modelo clássico. A escolha do modelo se deu pelo fato deste trabalho focar em um processo de desenvolvimento simplificado, e o modelo em cascata, quando se trata de restrição de tempo e orçamento, pode ser um facilitador, uma vez que logo no início do planejamento do sistema podem ser definido seus limites de prazo e custo financeiro. Além disso, apesar de ser um método burocrático, esse modelo divide o projeto em etapas bem estruturadas, tornando melhor o foco em cada etapa do processo.

A fase de levantamento e análise dos requisitos é a primeira etapa do desenvolvimento do modelo em cascata, e nela devem ser documentados os requisitos e funcionalidades que devem ser entregues ao usuário final. A segunda etapa é a modelagem do projeto, que envolve todo o planejamento de estrutura de dados, arquitetura de software, definição de interfaces e detalhamento dos procedimentos. Em seguida, vem a fase de implementação ou codificação, que é onde será escolhida uma linguagem de programação, frameworks que auxiliarão no desenvolvimento, e onde de fato é posto em prática todo o planejamento feito nas etapas de levantamento de requisitos e modelagem (AUDY, 2007). Mais detalhes a respeito de linguagens de programação e frameworks serão explicados no tópico “Interface de Programa de Aplicação (API)” deste mesmo capítulo.

Após a implementação do projeto, vem as etapas de testes e implantação, onde o sistema passa por uma série de testes a procura de possíveis defeitos e também é nesse momento que é verificado se o sistema atende todos os requisitos levantados. Com os testes realizados e validados com sucesso, o sistema está pronto para ser implantado, ou seja, entregue ao cliente. Por fim, mas não menos importante, mesmo após a entrega, não acabam as fases do desenvolvimento, todo sistema precisa de manutenção, e essa é a última etapa do processo, pois inevitavelmente, todo software passa por mudanças, como por exemplo, novos requisitos solicitados pelos clientes, entre outras evoluções, mantendo o sistema funcionando de forma a condizer com o cenário ao qual está inserido no momento (AUDY, 2007).

Além dos métodos de desenvolvimento, também existem as ferramentas da engenharia de software que fornecem suporte na aplicação das etapas do processo de desenvolvimento

(PRESSMAN; MAXIM, 2021). Durante a metodologia deste trabalhos, serão citadas as ferramentas que foram utilizadas em cada etapa.

2.3 CLOUD COMO AMBIENTE DE OPERAÇÃO

A computação em nuvem é um termo consideravelmente novo, e diz respeito a um modelo de operação, onde sistemas são hospedados em recursos computacionais como redes, servidores, armazenamento, aplicativos, e etc, disponibilizados de forma remota e compartilhada. Sendo estes facilmente configuráveis e provisionados de forma rápida e com pouco esforço, além disso, totalmente gerenciados pela empresa fornecedora do serviço de cloud computing (MACHADO, 2019). De forma mais simplificada, podemos dizer que a cloud computing, ou computação em nuvem, em português, é a disponibilização sob demanda de recursos computacionais em servidores de terceiros, sem gerenciamento ativo do usuário.

Algumas das principais vantagens das nuvens como ambiente de operação de sistemas, além do autogerenciamento, são sua alta disponibilidade de recursos, rápida escalabilidade, monitoramento de serviços, otimização de custos e análises de desempenho.

Atualmente, já existem muitos fornecedores de serviços de cloud, como a Microsoft Azure[1], Google Cloud Platform[2], e a Amazon Web Services[3], conhecida como AWS, que será a plataforma utilizada na modelagem do sistema neste trabalho.

A AWS é uma das maiores plataformas de serviços de cloud, ela oferece quase todos os recursos computacionais necessários para diversos tipos de sistemas e permite criar soluções que integram seus serviços, como bem descrito no texto abaixo:

> "A AWS disponibiliza uma ampla gama de serviços como poder computacional, armazenamento de dados, gestão e base de dados, networking, analytics, balanceamento de carga e escalonamento automático. Para além destes, são ainda fornecidos serviços que permitem aumentar a produtividade e eficiência nomeadamente ferramentas de desenvolvimento, ferramentas de gestão, serviços de proteção de identidade e segurança" (MACHADO, 2019).

Diante do exposto, e considerando que uma conta digital, como já mencionado, é um sistema complexo, que demanda uma boa gestão de risco e de segurança, e levando em

[1] https://azure.microsoft.com/pt-br/
[2] https://cloud.google.com/
[3] https://aws.amazon.com/

consideração o fato de que podemos reduzir custos no seu desenvolvimento, optar pela cloud como ambiente de operação de um sistema de conta digital pode ser promissor.

2.4 BANCO DE DADOS

Contas digitais precisam de algumas informações para que funcionem de forma correta, como dados pessoais de seus usuários, número de contas, histórico de transações e etc. Para isso, o sistema deve conter um Banco de Dados, que é onde ficam armazenados e organizados os conjuntos de dados do sistema. Eles guardam as informações que representam o mundo real, e qualquer alteração feita pelo usuário, no sistema, deve refletir neles (ALVES, 2014).

Os softwares responsáveis por administrar os bancos de dados são os SGBDs (Sistema Gerenciador de Banco de Dados), que permitem que o CRUD (create, read, update and delete) seja realizado, isto é, que seja possível criar, ler, editar e deletar dados (GARCIA, 2019). Alguns SGBDs que podemos citar são o PostgreSQL, MySQL, Oracle, MongoDB, entre outros.

Além disso, existem dois tipos de bancos de dados, os relacionais e os não relacionais. O banco de dados relacional refere-se a um modelo onde os dados de uma tabela se relacionam com os dados de outra tabela. Essa conexão entre os dados é realizada através das chamadas chaves primárias e chaves estrangeiras. Uma chave primária é o identificador de um dado em uma tabela, que por sua vez, aparece como chave estrangeira em outra tabela ao qual está relacionado (ROB, 2011). O PostgreSQL e o MySQL são exemplos de sistemas gerenciadores de bancos de dados relacionais. O banco de dados não relacional, por sua vez, diz respeito ao modelo de dados onde as informações entre as tabelas não se cruzam, dessa forma, em comparação ao modelo relacional, esses tipos de bancos de dados não precisam de um sistema gerenciador robusto, pois tem uma escalabilidade mais barata e menos complexa (DE OLIVEIRA, 2014). Um exemplo de sistema que gerencia esse modelo, é o DynamoDB da Amazon Web Services.

2.5 INTERFACE DE PROGRAMAÇÃO DE APLICAÇÃO (API)

Os sistemas de softwares são entregues aos usuários finais por meio de interfaces gráficas, onde as funcionalidades serão objetivas e realizadas por meio de cliques em botões, imagens ou textos. Por trás das funcionalidades disponibilizadas por essas interfaces gráficas estão as APIs, sigla do inglês para *Application Programing Interface*. As APIs fazem parte do

backend dos sistemas, e podem ser definidas, segundo Lenz, 2023, como um conjunto de regras e protocolos que permitem que diferentes sistemas de software se comuniquem e interajam uns com os outros.

As APIs fazem parte do backend, que é a parte do sistema onde são implementadas as regras de negócio do sistema, e que não pode ser acessada diretamente pelos usuários. A maior parte de um software é desenvolvida, armazenada e acessada no backend (SOUZA, 2022). Essas APIs são desenvolvidas utilizando linguagens de programação, que de forma resumida, são linguagens interpretadas e/ou compiladas pelos computadores, que contém conjuntos de instruções e regras de um sistema. As linguagens de programação costumam ter melhor desempenho em algumas atividades do que em outras, como por exemplo, para criação de interfaces de programação de aplicações, algumas das linguagens de programação mais utilizadas são o Java, o JavaScript e Ruby. Isso se deve ao fato de existirem *frameworks* de criação de APIs dentro dessas linguagens e de muitas outras.

Os frameworks são conjuntos de códigos que fornecem ferramentas como bibliotecas e métodos voltados para funcionalidades específicas. No Javascript, por exemplo, existe o framework Express.js, que tem como finalidade facilitar o desenvolvimento de aplicações web nesta linguagem de programação (ERSE, 2021). No tópico “Implementação” do capítulo de metodologia deste trabalho será mencionado outras linguagens de programação que podem ser utilizadas para desenvolver o projeto de backend do objeto de estudo deste trabalho: as contas digitais.

3 METODOLOGIA

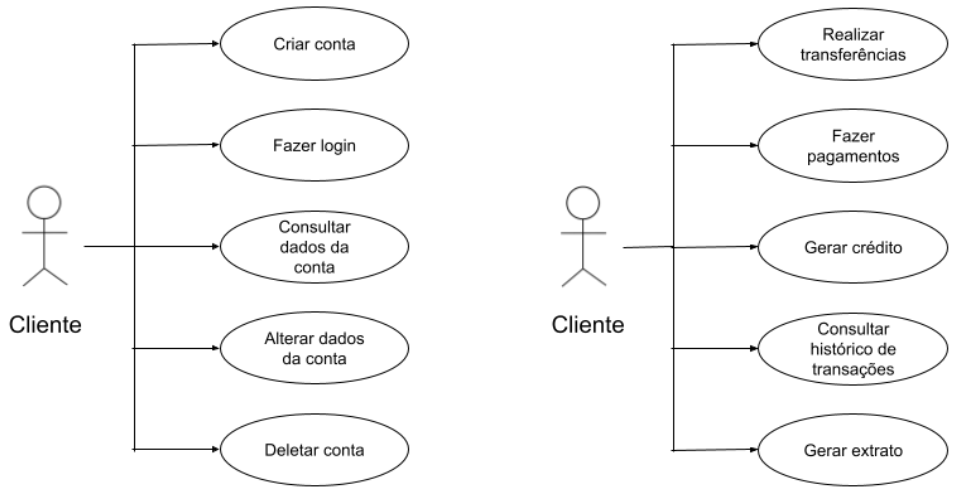
Neste capítulo serão apresentados os procedimentos metodológicos realizados neste trabalho, descrevendo-se os procedimentos que foram utilizados para demonstrar o processo de desenvolvimento de um sistema de conta digital utilizando ferramentas que facilitam suas etapas.

Foi usada a abordagem qualitativa para alcançar os objetivos propostos, e para maior conhecimento da problemática, foi realizada uma pesquisa descritiva e exploratória, onde foram demonstrados os processos de desenvolvimento do projeto de uma conta digital e foram exploradas algumas das ferramentas de Engenharia de Software durante esses processos.

3.1 LEVANTAMENTO E ANÁLISE DE REQUISITOS

Essa fase de levantamento e análise de requisitos se dá no início do processo de criação de um software e é primordial para a aplicação das demais fases do processo. Muitas técnicas podem ser utilizadas nesta etapa, sendo algumas delas a prototipagem e os casos de uso. Neste trabalho será utilizada a técnica de casos de uso, por se tratar de uma boa maneira de levantar informações sobre as interações entre os atores (usuários) e o sistema. E para desenhar esses casos de uso, foi utilizada a ferramenta Draw.io[4], que é um software de desenho multiplataforma online e gratuito, que auxilia na criação de fluxogramas, diagramas UML, wireframes, entre outros. Além disso, essa ferramenta pode ser conectada a sua conta no Google Drive, salvando seus diagramas para serem alterados sempre que for preciso. Outra ferramenta de criação de fluxogramas, muito completa, é o Miro[5], que também é online e gratuita, e possui diversos recursos para criar visualizações de projetos. A imagem abaixo mostra o diagrama de casos de uso levantados neste trabalho. Como o objetivo aqui é termos um exemplo de processo de construção de um software com ênfase no backend, será focada apenas em alguns principais requisitos de uma conta digital.

Figura 1 - Diagrama de casos de uso



Fonte: Acervo Pessoal, 2024.

Levando em consideração os casos de uso levantados, temos então os seguintes requisitos funcionais do sistema:

[4] https://app.diagrams.net/
[5] https://miro.com/app/dashboard/

- Criação de conta: Capacidade de registrar novas contas, com dados essenciais como nome, número de documento de identificação, endereço, email e senha. Esse registro deve gerar um número de conta único.
- Login/Logout de conta: Os usuários registrados devem ter a possibilidade de fazer login e logout de suas contas sempre que acharem necessário, fazendo isso de forma segura por meio de credenciais únicas.
- Consultar e alterar dados: O sistema deve permitir que os usuários consultem seus dados e possam editá-los sempre que necessário.
- Cancelar conta: Capacidade de permitir o cancelamento de conta, essa ação deve bloquear as demais operações que o usuário tem acesso, porém, deve-se manter o histórico de dados e transações que existiram para a conta em questão, por um determinado tempo.
- Realizar transferências: Possibilitar a transferência de saldo entre contas. O débito em uma conta, deve gerar crédito em outra.
- Fazer pagamentos: Possibilitar a realização de pagamentos de contas. Uma operação que gera débito em conta.
- Gerar crédito: Capacidade de receber saldo, uma operação que gera crédito em conta.
- Consultar histórico de transações e gerar extrato: Permitir que os usuários possam consultar suas transações e gerar extratos de saldo.

Ainda na etapa de levantamento e análise de requisitos, precisamos também considerar os requisitos não funcionais. Não é o intuito deste trabalho focar neste ponto, mas como faz parte do processo, podemos listar alguns como:

- Segurança e privacidade: Capacidade de criptografar dados e evitar ataques digitais, protegendo a privacidade dos dados, assim como determinado pela Lei Geral da Proteção de Dados (LGPD).
- Experiência do usuário: Garantir acessibilidade e objetividade na interface do usuário.

- Desempenho e escalabilidade: Garantir um bom funcionamento do sistema mesmo em meio ao crescimento do número de usuários.

3.2 ARQUITETURA E MODELAGEM

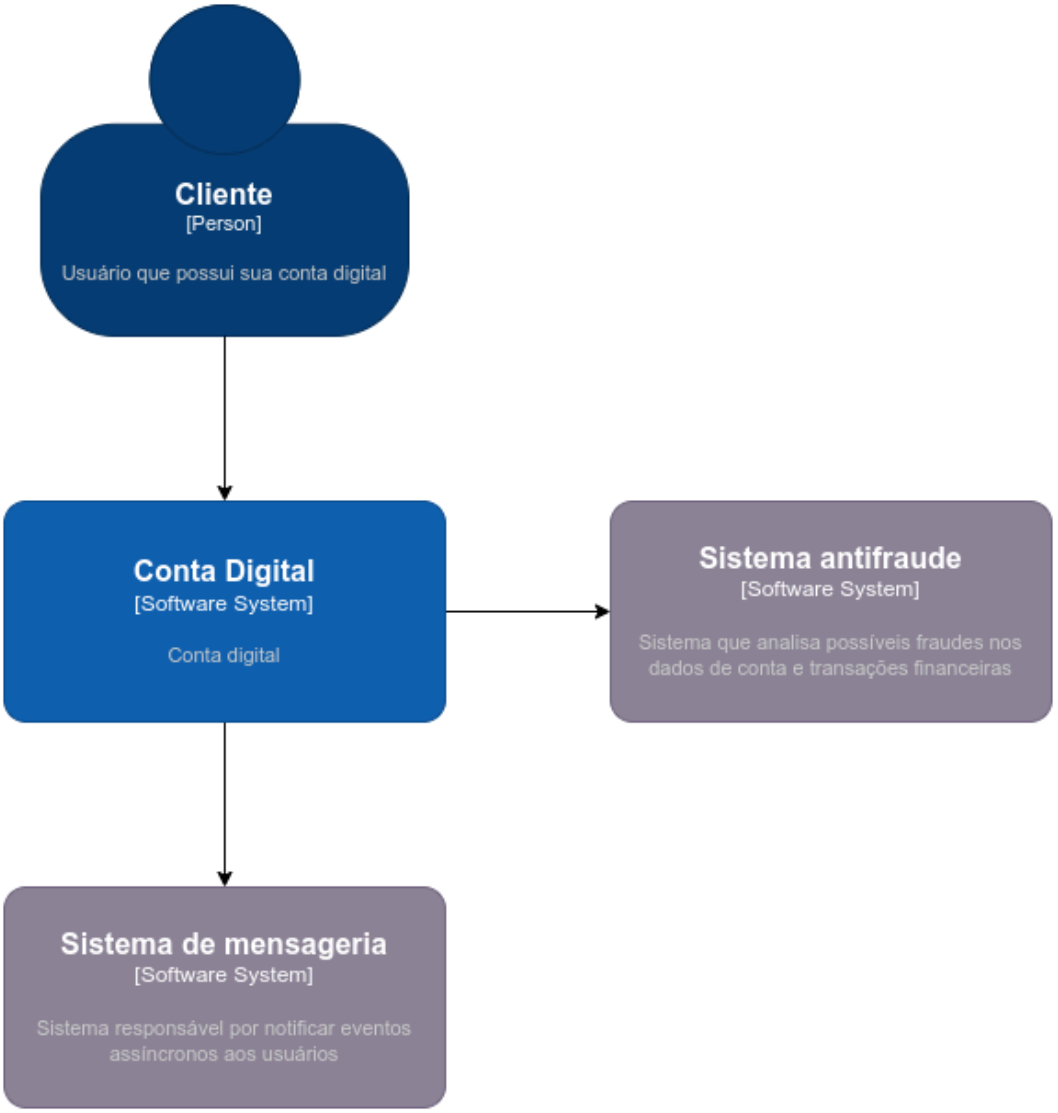
Como mencionado anteriormente, a etapa de arquitetura e modelagem do software consiste em planejar toda a estrutura do sistema, como modelagem de dados, arquitetura do software, definição de interfaces, detalhamento dos procedimentos, e muitos outros pontos que podem ser feitos nesta fase do processo.

Para este trabalho foi usado para a modelagem o C4 Model[6], que é um modelo de documentação de diagramas de arquitetura de software que organiza as camadas de forma hierárquica. Cada camada do C4 Model será explicada e demonstrada a seguir com os exemplos aplicados aos requisitos que foram levantados anteriormente para a conta digital.

A primeira camada do C4 Model traz o contexto do sistema, é o ponto inicial e uma visão geral do produto, onde é demonstrado o sistema como uma única caixa de representação, e as conexões externas com as quais ele interage. A imagem a seguir mostra o exemplo de como seria a camada de contexto do sistema de conta digital que está sendo considerada neste trabalho, onde na caixa retangular azul representa o sistema principal e as caixas na cor cinza representam os serviços externos.

[6] https://c4model.com/

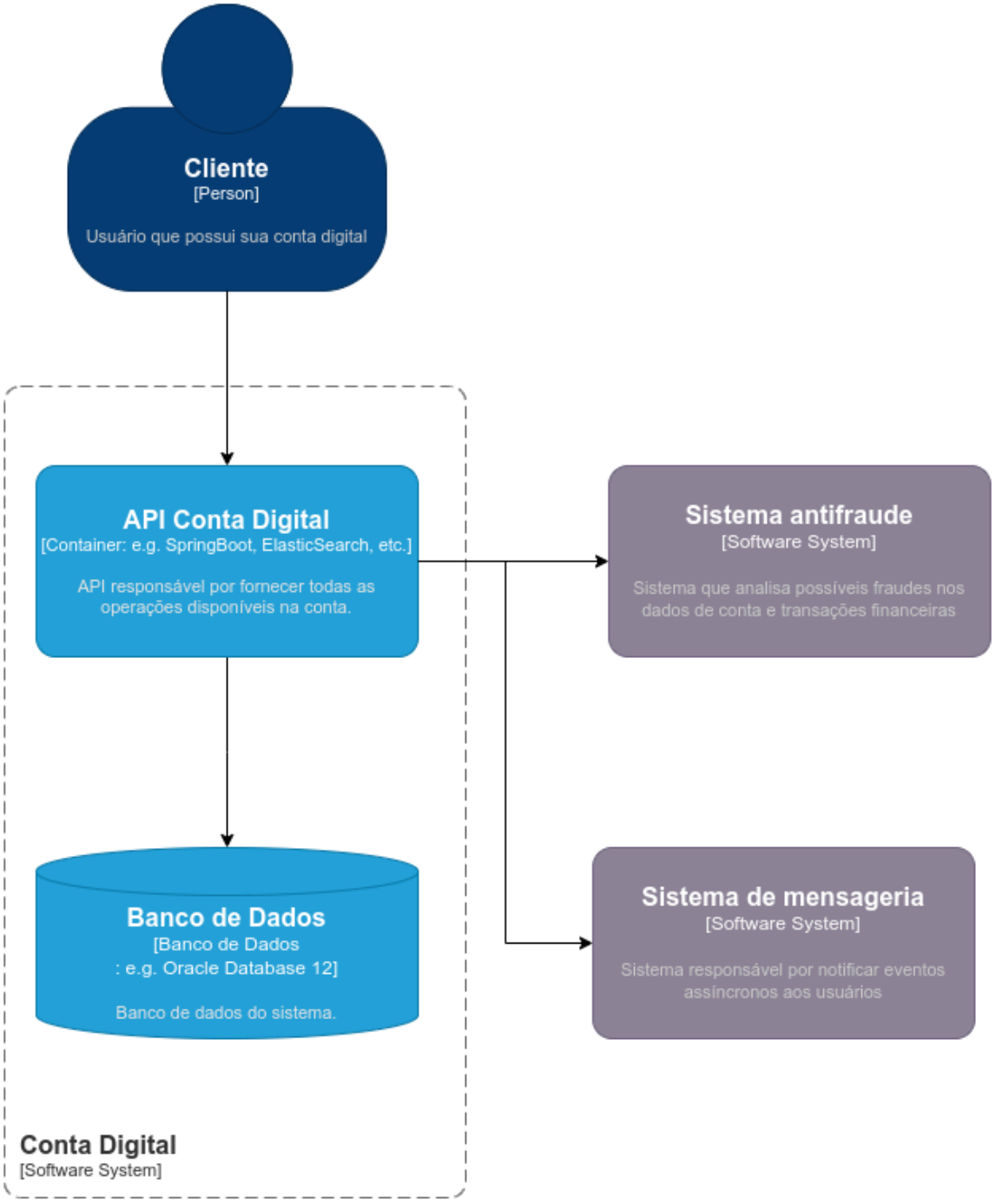
Figura 2 - Contexto do sistema



Fonte: Acervo Pessoal, 2024.

A segunda camada deste modelo é o diagrama de *container*, onde dessa vez é aberta a caixa de representação do sistema, e podem então ser representadas algumas de suas estruturas internas, como APIs, Bancos de Dados e outros. Ainda é um diagrama simples, que mostra em alto nível a arquitetura que será utilizada no projeto. Como foi aplicado na imagem a seguir, mostrando que o sistema de conta digital proposto será composto por uma API e um Banco de Dados:

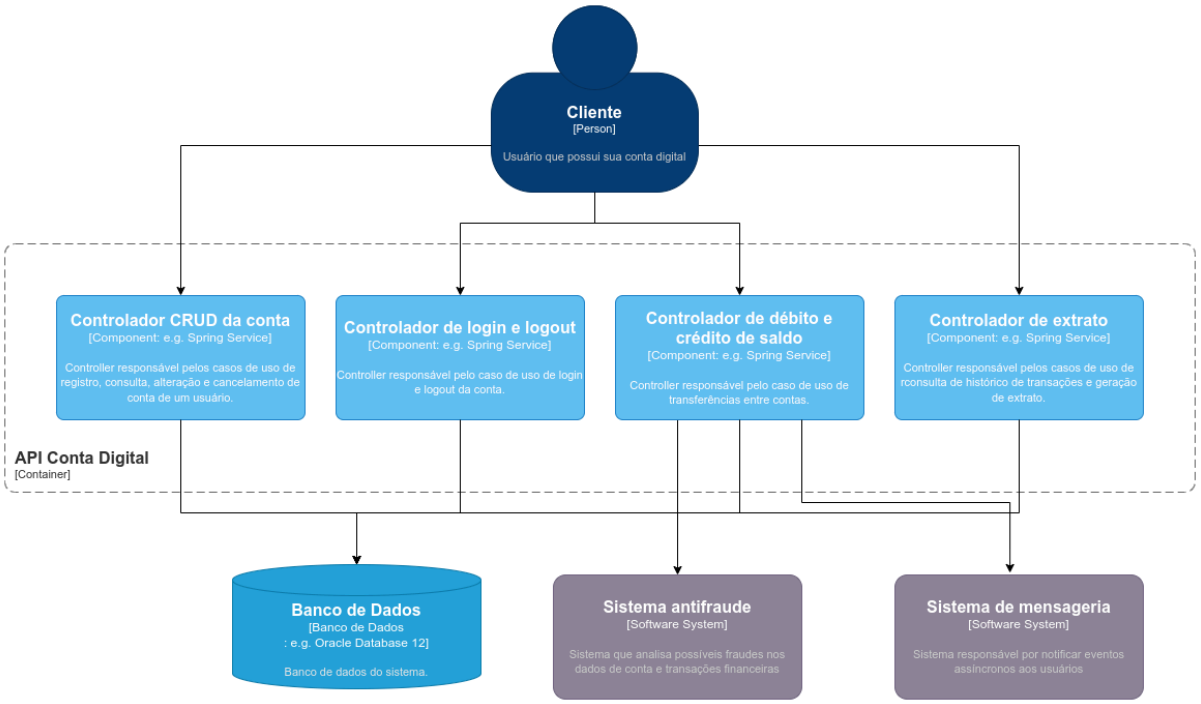
Figura 3 - Camada de containers do sistema



Fonte: Acervo Pessoal, 2024.

Na terceira camada, chamada de componentes, são ampliados os containers, e neles são mais especificados os principais blocos, estruturas e interações que farão parte do sistema. Na imagem abaixo mostra como poderia ser feita essa camada para o projeto de conta digital, especificando melhor como seria a estrutura da API Conta Digital:

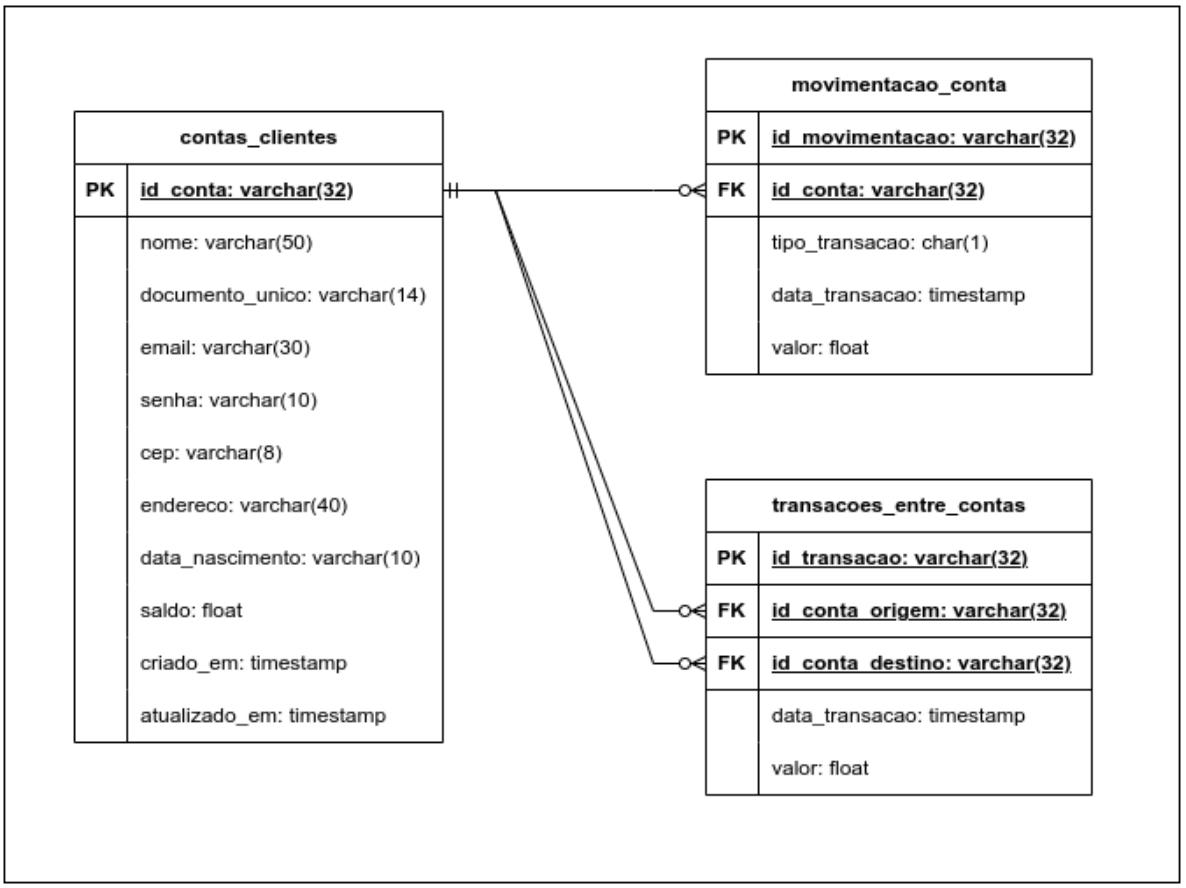
Figura 4 - Camada de Componentes do sistema



Fonte: Acervo Pessoal, 2024.

Na quarta e última camada do C4 Model, chamada de diagrama de código, podemos ampliar cada componente em diferentes blocos, e para isso podemos usar diferentes modelos, como diagramas UML, diagrama entidade-relacionamento, e etc. Foi aplicado nesta quarta camada do modelo, para o sistema de conta digital aqui discutido, o diagrama de entidade relacionamento do banco de dados, como podemos ver na imagem a seguir, um diagrama composto por 3 tabelas, onde a tabela *contas_clientes* têm relação um para vários tanto com a tabela *movimentacao_conta* quanto com a *transacoes_entre_contas*:

Figura 5 - Diagrama de Entidade Relacionamento do sistema



Fonte: Acervo Pessoal, 2024.

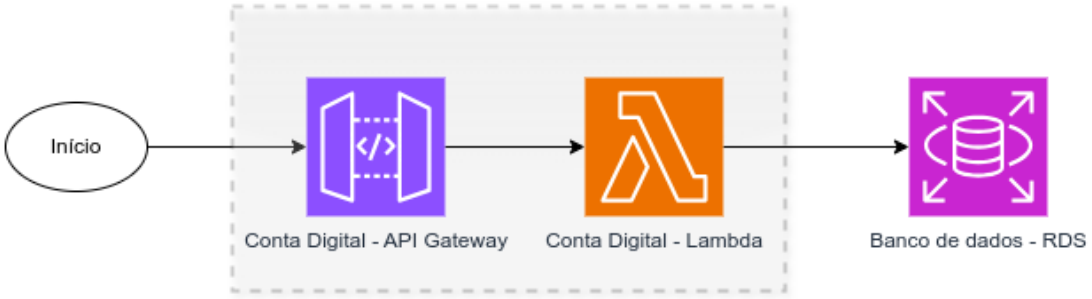
Por fim, ainda nesta etapa de arquitetura e modelagem, também são decididas as ferramentas de sistema de software que serão utilizadas. Neste trabalho foi optado por uma aplicação em *cloud*, especificamente, na cloud da Amazon Web Services, pois é uma plataforma autogerenciada que possui todos os componentes necessários para desenvolvimento de um sistema como o planejado até aqui, além de oferecer a modalidade de serviço *on-demand*, que consiste em cobrar apenas pelo que é utilizado, sem necessidade de contratos a longo prazo. Existe ainda, para alguns recursos a modalidade gratuita, que até certo volume de consumo não há cobrança de taxas..

Na Figura 6, podemos ver os componentes escolhidos para o desenvolvimento da conta digital, onde poderia ser utilizado como porta de entrada um API Gateway[7], serviço responsável por criar, publicar e monitorar APIs, além de possuir um ótimo desempenho, ser escalável e seguro. Esse API Gateway se comunicaria com um serviço AWS Lambda[8], que é

[7] https://aws.amazon.com/pt/api-gateway/
[8] https://aws.amazon.com/pt/lambda/

um componente *serverless*, ou seja, onde o programador não precisa se preocupar em gerenciar ou provisionar servidores, nem dimensionar a capacidade de processamento necessária, a AWS fica responsável por essa tarefa por meio de um *Lambda*. Por fim, esse Lambda se comunicaria com um RDS (Relational Database Service), que por sua vez, é um serviço responsável por gerenciar bancos de dados relacionais, e suporta uma variedades de SGBDs como PostgreSQL, MySQL, Oracle, Amazon Aurora, e etc, logo, qualquer um desses poderia ser utilizado para o desenvolvimento do sistema proposto.

Figura 6 - Diagrama de arquitetura em cloud do sistema



Fonte: Acervo Pessoal, 2024.

3.3 IMPLEMENTAÇÃO

Com a modelagem concluída, pode-se então iniciar a codificação do sistema, trazendo para o código toda a lógica e todos os requisitos levantados no início do projeto. É importante que seja bem analisada a linguagem de programação que será escolhida para o desenvolvimento, pois existem diversas linguagens de programação, mas muitas delas têm desempenho melhor em algumas tarefas que em outras.

Como na modelagem do sistema foi definida a utilização de um AWS Lambda, esse projeto já estaria limitado às linguagens Java, Go[9], JavaScript, C#, Python e Ruby, que são hoje compatíveis com esse serviço. Contudo, isso não é uma problema, essas linguagens são algumas das mais utilizadas no mercado atualmente, pelo seu desempenho, suporte e consolidação de mercado. Além disso, essas linguagens possuem frameworks que auxiliam no desenvolvimento de APIs, para o Java tem o Spring Boot, o Go possui o Gin e o Echo, para JavaScript temos o famoso NodeJs, e muitos outros para as demais linguagens. Dentre essas linguagens, considerando o sistema proposto, de uma conta digital, que exige respostas rápidas e seguras, uma boa escolha seria o Go, por se tratar de uma linguagem confiável, relativamente simples de ser manuseada, e principalmente por ser uma linguagem compilada,

[9] https://go.dev/

pois estes tipos de linguagens de programação tem uma execução mais rápida e mais econômicas dentro da AWS (ANDRADE JÚNIOR, 2023).

3.4 TESTES

Após a implementação do código são realizados os testes, que têm responsabilidade de garantir que todos os requisitos foram atendidos e que não há falhas de segurança, mantendo os dados seguros. Esta é a fase de encontrar bugs e corrigi-los, de encontrar possíveis problemas que não haviam sido considerados, e ajustar todo o projeto para que o sistema seja entregue de forma eficiente.

Também existem ferramentas que podem auxiliar nesta etapa, como a Katalon Studio[10] e a Selenium[11]. O Katalon Studio é uma plataforma de testes construída sobre a estrutura de código do Selenium, integra estruturas e recursos que são necessários para criação e execução rápida de casos de teste. O Selenium é uma das estruturas de automação de testes mais consistentes e utilizadas, além de poder ser integrado com outras ferramentas para aprimorar sua capacidade (DE SOUZA, 2023). Segundo De Souza, 2023, não existe uma única ferramenta de teste automatizado que seja adequada para todos os cenários e a escolha da ferramenta de automação de testes deve ser baseada em uma análise criteriosa das necessidades específicas do projeto.

3.5 IMPLANTAÇÃO

Nesta etapa, com os testes já validados, o serviço é disponibilizado aos usuários. Como no planejamento desse projeto foi pensando em uma solução em AWS Lambda, é necessário que o *deploy* seja feito nesse serviço, e para isso, é necessário que sejam realizados alguns passos dentro da Amazon Web Services, para que a aplicação possa ser disponibilizada, como cadastro de usuário, configurações de segurança e autenticação, criação de credenciais de desenvolvimento e de recursos que serão utilizados pela aplicação (BRIENZE JR, 2022).

Esses recursos podem ser gerenciados por meio de um [12]AWS CloudFormation, serviço que fornece aos profissionais de TI a capacidade de provisionar os modelos de arquitetura necessários para seu sistema. Além disso, pode-se usar ainda o AWS

[10] https://katalon.com/
[11] https://www.selenium.dev/
[12] https://aws.amazon.com/pt/cloudformation/

CodeDeploy[13], como um complemento ao CloudFormation, para gerenciar as implantações e atualizações de recursos (DALBHANJAN, 2015).

3.6 MANUTENÇÃO

Por fim, após a etapa de entrega do software, temos a etapa de manutenção, pois mesmo após o sistema ter sido entregue, podem surgir bugs que só serão notados após o sistema ter começado a ser utilizado pelos clientes, melhorias de desempenho também podem ser necessárias de acordo com o crescimento no número de clientes utilizando o serviço, além de serem necessárias atualizações frequentes de código. O Lambda AWS por exemplo, tem uma lista das versões para cada uma das linguagens de programação que suporta, e essa lista é atualizada frequentemente, dessa forma, sistemas hospedados em Lambdas AWS devem se manter atentos às atualizações do serviço.

4 RESULTADOS E DISCUSSÕES

Durante a revisão de estudos de outros autores, foram identificadas que, de acordo com a Engenharia de Software, as principais etapas do processo de desenvolvimento de um sistema, seguindo o modelo em cascata são:

1. Levantamento e análise de requisitos
2. Arquitetura e modelagem
3. Implementação
4. Testes
5. Implantação (*deploy*)
6. Manutenção

E esses foram o modelo e as etapas seguidos durante este trabalho, onde na primeira etapa, de levantamento e análise de requisitos, foi utilizada a técnica de casos de uso, onde conseguimos colocar de forma visual as requisições do sistema. Para o desenho desses casos foi utilizada a ferramenta Draw.io, este desenho pode ser visto na Figura 1. Após isso, foi feita a modelagem do sistema, que também teve como ferramenta de apoio o Draw.io, e como recurso de apoio o C4 Model, que ajudou a criar um levantamento gradual de como seria a estrutura do software. Na Figura 4 podemos ter uma noção de como seriam estruturadas as

[13] https://aws.amazon.com/pt/codedeploy/

controllers da API do sistema, bem como quais seriam suas integrações internas (o banco de dados) e externas (Sistema Antifraude e Sistema de Mensageria). Já na Figura 5, podemos ver a modelagem do banco de dados, que também confirma os requisitos levantados na primeira etapa do planejamento desses sistemas. Por fim, ainda na fase de modelagem, foi desenhada a arquitetura do software baseado em um ambiente em *cloud*, utilizando serviços das AWS, o resultado dessa arquitetura pode ser visto na Figura 6.

Na etapa de implementação foram trazidas as linguagens de programação que podem ser utilizadas considerando o recurso de cloud que foi planejado durante a fase de arquitetura, o Lambda AWS. Dentre essas linguagens que podem ser utilizadas para o desenvolvimento, foi mencionada que o Go seria uma boa escolha, por se tratar de uma linguagem de programação compilada e ter uma curva de aprendizado menor comparado ao Java e ao C#, as outras linguagens compiladas suportadas pelo AWS Lambda.

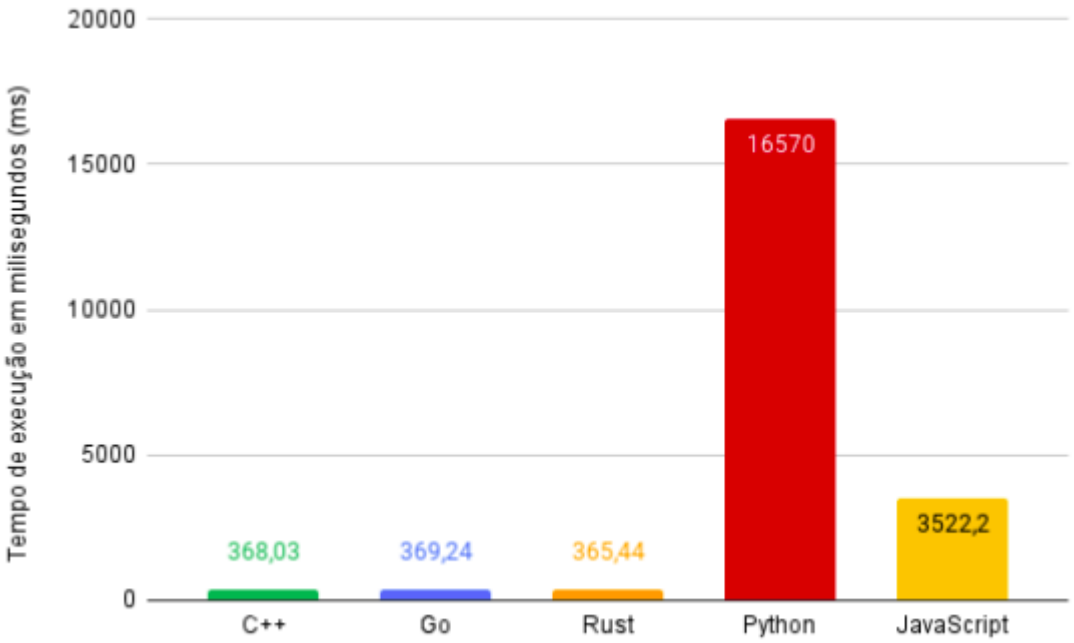
Para a fase de testes foram citadas duas ferramentas de acordo com o estudo feito por De Souza, 2023, sendo elas o Katalon Studio e a Selenium. Já na etapa de implantação foi discutido a respeito do que é necessário para um deploy utilizando os recursos da AWS e foram indicadas duas ferramentas da própria AWS que podem auxiliar nesta etapa, sendo elas o AWS CloudFormation e o AWS CodeDeploy. E por fim, na etapa de manutenção foram descritos alguns dos principais motivos pelos quais sistemas passam por manutenções, trazendo em especial a questão de se atentar a atualizações de compatibilidade de versões de linguagens de programação com o AWS Lambda, uma vez que foi optado pela utilização desse serviço.

A ferramenta Draw.io, utilizada tanto na fase de levantamento de requisitos, quanto na fase de arquitetura e modelagem é uma ótima escolha, pois além de ser totalmente online, sem que haja necessidade de de instalação de softwares pesados, é gratuita, possui suporte para diversos tipos de diagramas, e disponibiliza elementos separados por categoria, o que facilita a criação de desenhos. Além disso, tem compatibilidade com todos os navegadores e funciona também offline (TIRLONI, M., 2018).

O C4 Model, utilizado na fase de arquitetura e modelagem do projeto, por sua vez, também é uma ótima opção para documentar a estrutura de um sistema. A maior vantagem da utilização desse modelo de diagramação está na simplificação do sistema, onde o nível de detalhamento vai aumentando conforme a profundidade das camadas (WAKI, 2021).

O Go, indicada como uma boa opção de linguagem de programação para desenvolvimento de uma API utilizando AWS Lambda, se deu pelo fato dela ser eficiente e simples de se utilizar ao mesmo tempo, oferecendo também um equilíbrio entre performance e segurança de memória (CRESTANI, 2024). Em um dos testes realizados por Crestani, 2024, que comparava a eficiência de algumas linguagens de programação, podemos notar que o Go obtém um ótimo desempenho, mostrado no gráfico abaixo:

Gráfico 1 - Comparativo de média de tempo de execução para um cálculo



Fonte: Uma comparação empírica em velocidade de processamento entre C++, Go, Rust, Python e JavaScript. Crestani, 2024.

Ademais, foram considerados os serviços da AWS tanto na fase de estrutura e implementação, com o AWS API Gateway, AWS Lambda e AWS RDS, quanto na etapa de deploy com o AWS CloudFormation e AWS CodeDeploy pelo motivo da Amazon AWS disponibilizar uma infraestrutura completa para softwares em diversos níveis de processamento (SOUSA, 2009).

Diante disso, podemos observar que seguir o modelo de desenvolvimento de software em cascata, por mais que não seja o modo mais rápido, é uma boa maneira de planejar um sistema, visto que uma etapa depende da outra, sendo assim, a construção da lógica acontece de forma gradual e consequentemente mais simples. Ainda, podemos notar que hoje existem

muitas ferramentas que auxiliam esses processos, tornando ainda menos moroso o desenvolvimento do projeto.

5 CONSIDERAÇÕES FINAIS

Foi demonstrado ao longo deste trabalho os principais processos do desenvolvimento de um software e apresentadas durante as etapas do processo as ferramentas que contribuíram para o planejamento do sistema.

Tendo como principal objetivo demonstrar o processo de desenvolvimento de um sistema de conta digital utilizando ferramentas para facilitar suas etapas, servindo assim, como um guia de planejamento de software, este trabalho conseguiu atingir esse objetivo, seguindo para isso os objetivos específicos como explicar e justificar o objeto de estudo escolhido, descrever os principais processos de desenvolvimento segundo a Engenharia de Software, demonstrar esses processos através do planejamento de uma pequena conta digital e usar e mencionar tecnologias e ferramentas como suporte durante o planejamento.

Diante disso, foi entendido que as ferramentas e práticas de Engenharia de Software auxiliam no planejamento e desenvolvimento de sistemas de forma a torná-los menos complexos, mais fluídos e menos custosos.

Este trabalho pode contribuir para a comunidade acadêmica de tecnologia da informação como um guia inicial de planejamento e desenvolvimento de um software, além de um indicador de ferramentas que contribuem para estes objetivos.

Por fim, para esta pesquisa cabem muitos trabalhos futuros, como a implementação do sistema de conta digital, utilizando os recursos levantados neste trabalho. Cabe também o protótipo e desenvolvimento do frontend do sistema, considerando ferramentas e técnicas voltadas para este tema. Além disso, pode-se adentrar ainda mais nas contas digitais como objetos de estudo, e explorar a área de segurança de dados.

REFERÊNCIAS

ENGHOLM JUNIOR, Hélio. **Engenharia de software na Prática**. São Paulo: Novatec, 2010.

WAZLAWICK, Raul. **Engenharia de software: conceitos e práticas**. Elsevier Editora Ltda., 2019.

DINIZ, Bruno. **A nova lógica financeira: Como as soluções financeiras digitais estão impactando todos os mercados e o que fazer para sobreviver nesse cenário**. Gente Autoridade, 2021.

DO NASCIMENTO, Hérica Henrique. CONTAS DIGITAIS: A REVOLUÇÃO DO SISTEMA BANCÁRIO E A PERCEPÇÃO DOS SERVIÇOS PELA SOCIEDADE. **Revista Valore**, v. 5, p. 282-293, 2020.

LINDGREN JR, J. H. Marketing na Internet. In: CZINKOTA, M. R. **Marketing: As Melhores Práticas.** Porto Alegre:Bookman, 2001.

LESSA, Rafael Orivaldo; LESSA JUNIOR, Edson Orivaldo. **Modelos de processos de engenharia de software.** Link para o PDF: http://xps-project. googlecode. com/svn-history/r43/trunk/outros/02_Artigo. pdf, 2009.

PRESSMAN, Roger S.; MAXIM, Bruce R.. **Engenharia de software: uma abordagem profissional**. 9. ed. Porto Alegre: Amgh, 2021.

AUDY, Jorge Luis Nicolas. **Desenvolvimento distribuído de software**. Elsevier, 2007.

MACHADO, Diogo Alexandre Gonçalves. **Otimização dos custos de operação de aplicações web em cloud**. 2019. Tese de Doutorado. Universidade do Minho (Portugal).

VERAS, Manoel. **Cloud Computing: nova arquitetura da TI**. Brasport, 2012.

GARCIA, Vinícius Salles; SOTTO, Eder Carlos Salazar. Comparativo Entre os Modelos de Banco de Dados Relacional e Não-Relacional. **Revista Interface Tecnológica**, v. 16, n. 2, p. 12-24, 2019.

ALVES, WILLIAM PEREIRA. **Banco de dados**. Saraiva Educação SA, 2014.

ROB, Peter; CORONEL, Carlos. Sistemas de banco de dados. **Projeto, implementação e**, 2011.

DE OLIVEIRA, Samuel Silva. Bancos de dados Não-Relacionais: um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo. **Revista da Escola de Administração Pública do Amapá**, v. 2, n. 1, p. 184-194, 2014.

LENZ, Gabriel Henrique; FRANCISCATTO, Roberto. APIs na Era Digital: Um Enfoque na Segurança e Conscientização. **Anais do Encontro Anual de Tecnologia da Informação**, v. 12, n. 1, p. 97-97, 2023.

ERSE, Alan Vasconcellos. Desenvolvimento e análise do backend do projeto CuidaIdoso. 2021.

SOUZA, Francisco Moreira Calado; LIMA, Edilson Carlos Silva; DE SENA CARIDADE, Elda Regina. CRIANDO SISTEMA ESCALÁVEL DE AGENDAMENTOS UTILIZANDO TYPESCRIPT COM NESTJS NO BACKEND E NEXTJS NO FRONTEND. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 8, n. 12, p. 43-57, 2022.

OLIVEIRA, Márcio Lucas Rezende de et al. Desenvolvimento do sistema web chemistery ênfase no back-end. 2019.

CANTOR, Murray. Software Leadership: A Guide to successful Software Development. 1º ed. Indianapolis: Addison-Wesley 2001.

ANDRADE JÚNIOR, Edilson Alves de. **Uma análise do impacto das linguagens de programação nos custos de execução no AWS Lambda em cenários de cold start e warm start.** 2023. Trabalho de Conclusão de Curso. Brasil.

DE SOUZA, Guilherme Guimarães et al. AUTOMAÇÃO DE TESTES: uma abordagem comparativa entre ferramentas. **Revista Interface Tecnológica**, v. 20, n. 2, p. 100-111, 2023.

BRIENZE JR, Luis Felipe Sabadoto. Análise da implementação de tecnologias da nuvem Amazon Web Services para aplicação backend em Java. 2022.

DALBHANJAN, Peter. Overview of deployment options on aws. **Amazon Whitepapers**, 2015.

TIRLONI, M.; MACHADO, C. C. Uma proposta para auxiliar pessoas com deficiência visual e daltonismo a identificar cores e suas possíveis combinações. **Simpósio de Ciência, Inovação e Tecnologia**, p. 9, 2018.

WAKI, L. H. et al. Projeto Arquitetural da Plataforma de Trabalho Decente. 2021.

CRESTANI, Alexandre de Rosso. Uma comparação empírica em velocidade de processamento entre C++, Go, Rust, Python e JavaScript. 2024.

SOUSA, Flávio RC; MOREIRA, Leonardo O.; MACHADO, Javam C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. **II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)**, p. 150-175, 2009.

RIBEIRO, MATHEUS BARON. CENÁRIOS DE APLICAÇÃO DAS METODOLOGIAS TRADICIONAIS E ÁGEIS NO DESENVOLVIMENTO DE SOFTWARE. 2020.