CLAUDINEI MOREIRA DA SILVA

ANÁLISE E TRANSFORMAÇÃO DE UM SOFTWARE CUSTOMIZADO EM UM SISTEMA CONFIGURÁVEL COM FOCO EM LOW CODE

LAVRAS-MG
2023

CLAUDINEI MOREIRA DA SILVA

ANÁLISE E TRANSFORMAÇÃO DE UM SOFTWARE CUSTOMIZADO EM UM SISTEMA CONFIGURÁVEL COM FOCO EM LOW CODE

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Antônio Maria Pereira de Resende
Orientador

LAVRAS - MG
2023

CLAUDINEI MOREIRA DA SILVA

ANÁLISE E TRANSFORMAÇÃO DE UM SOFTWARE CUSTOMIZADO EM UM SISTEMA CONFIGURÁVEL COM FOCO EM LOW CODE

ANALYSIS AND TRANSFORMATION OF CUSTOMIZED SOFTWARE INTO A CONFIGURABLE SYSTEM WITH A FOCUS ON LOW CODE

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADO em 07 de dezembro de 2023.

Dr. Antônio Maria Pereira de Resende	UFLA
Dr. Paulo Afonso Parreira Júnior	UFLA
B.el. Wilson Camilo Costa	ZETTA

gov.br
Documento assinado digitalmente
ANTONIO MARIA PEREIRA DE RESENDE
Data: 19/12/2023 19:08:36-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Antônio Maria Pereira de Resende

Orientador

LAVRAS – MG
2023

AGRADECIMENTOS

Desejo manifestar minha gratidão à minha família, que sempre me apoiou durante toda essa trajetória. Às amizades que se tornaram um refúgio nos momentos de dúvida e desafio. E aos meus professores, cuja orientação e conhecimento foram fundamentais para o sucesso deste trabalho. Obrigado a todos que fizeram parte dessa caminhada.

“Software é uma expressão da mente humana, uma manifestação de nossa criatividade. Seja inovador, seja ousado, e crie ferramentas que permitam que outros manifestem sua criatividade.”(Alan Cooper)

RESUMO

O presente trabalho teve o propósito de analisar Softwares existentes com o objetivo de mapear características, regras de negócio e recursos comuns entre eles, ampliando a visão do autor deste trabalho sobre possibilidades e técnicas para aumento da eficiência na produção de software. A escolha deste tema justifica-se pelo fato do trabalho de desenvolvimento de Software frequentemente ser repetitivo. Este fato faz com que os desenvolvedores acabem refazendo os mesmos processos a cada novo projeto. Certos recursos como banco de dados, controle de usuários, CRUDs de entidades, dentre outros são comuns em sistemas de informação. Portanto espera-se que com o levantamento dos recursos comuns a estes Softwares, seja possível prototipar um sistema capaz de gerar automaticamente um sistema genérico. Este Software lerá as configurações desejáveis como esquemas de cores, logotipo, títulos do sistema, informações a serem armazenadas e pacotes a serem utilizados. Tal sistema já disponibiliza todos os recursos básicos para que os mesmos não precisem ser feitos manualmente e poupando tempo de desenvolvimento. Desta forma com este trabalho pretende-se que a ferramenta produza um sistema (low code – Baixa quantidade de código), funcional e pronto para receber novos recursos. Ao final do trabalho foi construído um protótipo de alta fidelidade da ferramenta além da implementação de um pequeno sistema de informações geográficas para representar o tipo de sistema que pode ser construído com a ferramenta.

Palavras-chave: Low Code. Desenvolvimento. Repetitivo. Protótipo. Geoprocessamento.

ABSTRACT

The purpose of this study was to analyze existing software with the aim of mapping features, business rules, and common resources among them, expanding the author's perspective on possibilities and techniques for enhancing software production efficiency. The choice of this topic is justified by the fact that software development work is often repetitive. This leads developers to redo the same processes for each new project. Certain resources such as databases, user control, entity CRUDs, among others, are common in information systems. Therefore, it is expected that by identifying common features in these software, it will be possible to prototype a system capable of automatically generating a generic system. This software will read the desired configurations, such as color schemes, logos, system titles, information to be stored, and packages to be used. This system already provides all the basic resources so that they do not need to be created manually, saving development time. Thus, with this work, the goal is for the tool to generate a functional, ready-to-use system (low code - minimal amount of code) and capable of accommodating new features. At the end of the work, a high-fidelity prototype of the tool was built, along with the implementation of a small geographic information system to represent the type of system that can be built with the tool.

Keywords: Development. Repetitive. Prototype. LowCode. Geoprocessing.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Segundo a ABES (Associação brasileira das empresas de software) e a International Data Corporation (IDC), em novo estudo realizado em 2023 para analisar a atual situação do mercado de desenvolvimento de software, a indústria brasileira de Tecnologia da Informação (TI) ocupa atualmente a 12º posição no ranque mundial na produção de Software. Somente em 2022 o Brasil investiu cerca de US$45.2 bilhões no desenvolvimento de software.

Normalmente, a criação de um software é um esforço colaborativo realizado por equipes de desenvolvedores. “O processo de desenvolvimento consiste em uma série de passos, tarefas, eventos e práticas que devem ser seguidos pelos desenvolvedores para a construção de um sistema." (ENGENHARIA DE SOFTWARE MODERNA, 2022). Normalmente esse processo exige proficiência em diversas linguagens de programação e o uso de várias ferramentas auxiliares.

Por outro lado, existe a teoria de Linha de Produto de Software, cujo princípio baseia-se na criação de uma família de produtos de software em vez de um único produto isolado. Em vez de desenvolver cada software do zero, uma LPS aproveita a reutilização eficiente de componentes, módulos e práticas de engenharia comuns para criar uma variedade de produtos relacionados que compartilham características e funcionalidades comuns.

Considerando: a) o trabalho repetitivo do desenvolvedor; b) débitos técnicos; c) as várias funcionalidades comuns e generalizáveis de um software; e d) a sistemática de Linha de Produto de Software; este trabalho pretende propor a criação de uma ferramenta de geração de sistema para evitar iniciar cada novo projeto a partir do zero. Isto é possível, pois, ao analisar os sistemas produzidos pela Agência Zetta durante o período de realização da bolsa de pesquisa foi possível identificar um escopo em comum entre alguns sistemas desenvolvidos pela empresa. Desta forma utilizando os conceitos da linha de produção de software o desenvolvimento de um novo produto se tornará mais eficiente e com menor probabilidade de erros.

Ao fim deste trabalho todos os objetivos propostos foram alcançados de maneira bem sucedida. Desde o início deste projeto, o principal objetivo era gerar um protótipo de alta fidelidade e um pequeno SIG para demonstrar a viabilidade do trabalho proposto. Ao longo do processo de pesquisa, análise e redação, foi possível aprofundar-me nos temas relacionados ao geoprocessamento e explorar as questões levantadas de maneira abrangente e detalhada.

2 REFERENCIAL TEÓRICO

Neste capítulo, vamos apresentar de forma resumida todos os conceitos necessários para a fundamentação do projeto.

2.1 Low-Code e No-Code

O texto em Projeto Draft (2022) discute duas abordagens no desenvolvimento de software, conhecidas como Low-Code e No-Code. Ambas visam minimizar a necessidade de codificação, permitindo a criação de sistemas por meio de uma espécie de "montagem de blocos".

Enquanto o Low-Code envolve a intervenção da equipe de TI ou de um programador profissional para ajustes finais, as plataformas No-Code são mais avançadas, embora ainda estejam em desenvolvimento.

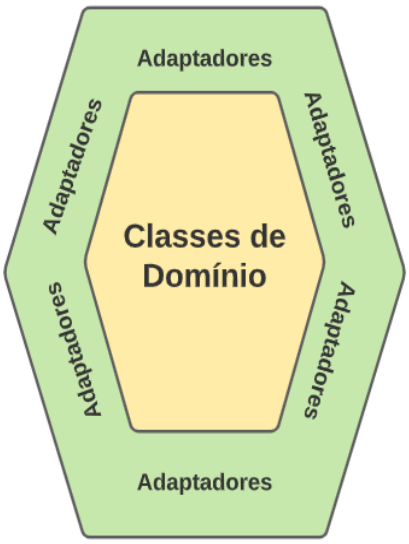
Apesar de dispensarem conhecimento em linguagem de programação, essas ferramentas exigem uma curva de aprendizado. A concepção de plataformas Low-Code tem origens antigas, como o "RAP" (Rapid Application Development), evoluindo ao longo do tempo com os avanços tecnológicos recentes.

2.2 Arquitetura hexagonal

A Arquitetura Hexagonal organiza as classes de um sistema em dois grupos fundamentais, sendo eles a) Classes de domínio: Estas classes ou objetos representam as entidades e regras de negócios essenciais ao núcleo do software; e b) Classes relacionadas à infraestrutura: Encarregam-se da integração com sistemas externos, como bancos de dados, garantindo uma conexão eficaz com o ambiente externo.

> “A arquitetura é ilustrada visualmente por meio de dois hexágonos concêntricos. O hexágono interno abriga as classes do domínio ou de negócios, enquanto o hexágono externo contém os adaptadores que são responsáveis por traduzir as operações ou chamadas provenientes das interfaces externas para operações que podem ser compreendidas pelo núcleo da aplicação.” (ENGENHARIA DE SOFTWARE MODERNA, 2022).

Figura 1 - Arquitetura hexagonal.



Fonte: ENGENHARIA DE SOFTWARE MODERNA (2022).

Na Arquitetura Hexagonal, conforme definido em engenharia de software moderna (2022), destaca-se a premissa crucial de que as classes de domínio não devem depender de classes relacionadas à infraestrutura, tecnologias ou sistemas externos. Essa divisão é estratégica para desacoplar essas classes, promovendo uma arquitetura mais flexível e modular.

De acordo com as diretrizes apresentadas em engenharia de software moderna (2022), na perspectiva da Arquitetura Hexagonal, as classes de domínio são concebidas para manter independência das especificidades tecnológicas, bancos de dados, interfaces do usuário e outras bibliotecas do sistema. Essa abordagem permite não apenas adaptações de tecnologia sem impactar as classes de domínio, mas também facilita o compartilhamento dessas classes entre distintas tecnologias, exemplificado pela coexistência de interfaces web e mobile em um mesmo sistema.

2.2.1 Adaptadores e portas

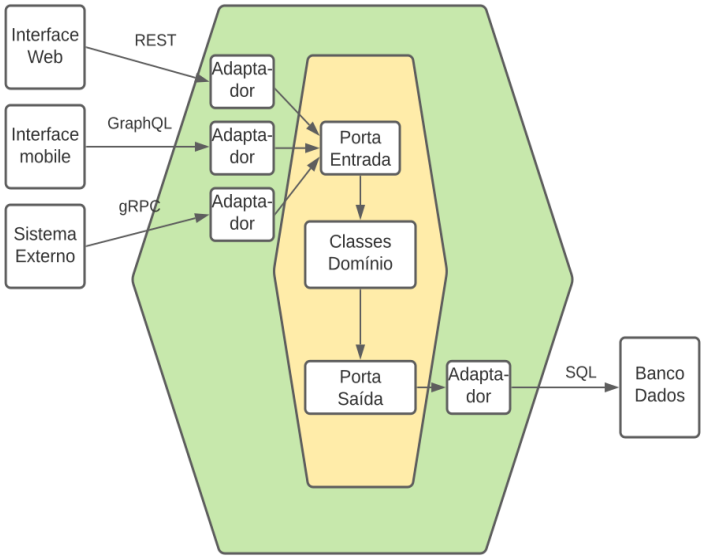
"Em uma Arquitetura Hexagonal, o termo porta designa as interfaces usadas para comunicação com as classes de domínio (veja que interface aqui significa interface de programação; por exemplo, uma interface de Java)" (ENGENHARIA DE SOFTWARE MODERNA, 2022).

Na arquitetura hexagonal, distinguimos dois tipos de portas: portas de entrada e portas

de saída. As portas de entrada facilitam a comunicação de entradas externas para o sistema, enquanto as portas de saída são responsáveis pela comunicação interna do sistema para o exterior. Importante destacar que essas portas são tecnologicamente independentes e estão situadas no hexágono interno.

Já os adaptadores, localizados no hexágono mais externo, desempenham duas funções: encaminhar chamadas vindas de fora para as portas de entrada e direcionar chamadas vindas de dentro do sistema para sistemas externos, como bancos de dados ou sistemas de terceiros.

Figura 2 - Arquitetura hexagonal com adaptadores e portas.



Fonte: ENGENHARIA DE SOFTWARE MODERNA (2022).

2.3 Sistema de informações geográficas

Geoprocessamento é o campo da tecnologia da informação dedicado à aquisição, processamento, análise e visualização de dados espaciais (com localização geográfica). Existem vários programas de geoprocessamento que permitem realizar diferentes tarefas, como criar mapas, analisar dados demográficos, gerar relatórios sobre dados espaciais e muito mais. Alguns exemplos de geoprocessadores incluem *ArcGIS*, *QGIS*, *GRASS GIS*, *ENVI* e *Geomedia*. Esses programas podem ser usados em vários campos, como agricultura, urbanização, conservação da natureza, etc.

2.3.1 O que é SIG ?

Um Sistema de Informação Geográfica (SIG) é uma ferramenta tecnológica que integra dados geográficos com recursos de análise e visualização, capacitando os usuários a desenvolver, armazenar, examinar e apresentar dados espaciais. A aplicação dos SIGs é crucial em várias áreas para orientar a tomada de decisões, proporcionando *insights* sobre padrões espaciais e resolvendo questões associadas a informações geográficas.

2.3.2 *Software* de informações geográficas

Foram estudados três software de geoprocessamento, sendo eles:

a) *o GeoNode* é uma plataforma de código aberto voltada para o compartilhamento e visualização de dados geoespaciais. Essa tecnologia capacita organizações e comunidades a estabelecerem portais dedicados a dados geoespaciais, nos quais os usuários podem publicar, compartilhar e colaborar em iniciativas geográficas. Integrando funcionalidades de visualização de mapas, gestão de metadados e suporte a padrões geoespaciais, o *GeoNode* emerge como uma ferramenta robusta para difundir informações geoespaciais e fomentar a colaboração em projetos de mapeamento;
b) o *MapStore* é uma plataforma de código aberto destinada à criação e compartilhamento de aplicativos interativos e geoespaciais. Essa solução disponibiliza recursos para a elaboração de mapas personalizados, análise de dados geoespaciais e colaboração em projetos cartográficos;
c) o Georadar é construído em cima de 3 outros *softwares* gratuitos fornecidos pela *GeoSolutions*. Os *software* gratuitos são respectivamente *Geonode, MapStore* e o *GeoServer*. Este programa oferece várias ferramentas para se trabalhar com geoprocessamento. O GeoRadar além dos recursos nativos dos *software* bases também oferece diversos outros recursos específicos que são construídos para atender demandas particulares do cliente.

2.4 Linha de Produto de *Software*

De acordo com Silva et al. (2011), uma linha de produtos de *software* (LPS) é um conceito usado na engenharia de *software* para descrever uma metodologia de desenvolvimento destinada a criar um conjunto de produtos de *software* relacionados que compartilham um núcleo comum, mas também possuem flutuações específicas para atender às necessidades de

diferentes clientes ou partes.

A ideia básica por trás de um conjunto de produtos de *software* é que, em vez de desenvolver produtos de *software* individuais do zero, propriedades usuais que podem ser reutilizadas podem ser identificadas e compartilhadas entre os produtos. Estas características gerais formam a base desta linha de produtos. Também conhecido como o núcleo do LPS.

O núcleo LPS foi projetado para ser flexível e extensível, permitindo que você selecione ou configure diferentes combinações de recursos para criar produtos específicos dentro de sua linha de produtos. Essas características podem incluir certas propriedades tais como interface de usuário, requisitos de desempenho, suporte entre plataformas ou qualquer outro aspecto relacionado ao *software*.

Ainda segundo Silva et al. (2011) , os benefícios da linha de produtos de *software* incluem custos e tempo de desenvolvimento reduzidos devido à reutilização de componentes e à personalização de propriedades existentes. Isso elimina a necessidade de criar tudo do zero para cada produto. O LPS também amplia a flexibilidade e a capacidade de resposta às necessidades do cliente facilitando a adição ou alteração de recursos em produtos existentes ou a criação de novos produtos usando componentes existentes.

Consequentemente, a linha de produtos de *software* é uma abordagem estratégica para construir produtos de *software* relacionados com base em um núcleo comum que provê reusabilidade, customização e flexibilidade de desenvolvimento.

2.4.1 Linha de produto de *software* versus reuso tradicional

Com base na publicação de Rafael em DevMedia sobre "Linha de Produto de Software" (2015), observa-se que a linha de produtos de software (LPS) e a reutilização tradicional representam abordagens distintas para a reutilização de software. A fim de entendermos melhor essas abordagens, procederemos à comparação das principais características de cada método.

Em resumo, enquanto a reutilização tradicional se concentra em reutilizar componentes individuais em diferentes contextos, o LPS busca criar uma série de produtos de *software* relacionados que compartilham um núcleo comum e permitem maior customização e agilidade no desenvolvimento. Ambas as abordagens têm suas vantagens e são adequadas para diferentes situações, dependendo do escopo e dos requisitos do projeto.

2.4.2 Variabilidade e gestão de variabilidade

Em Silva et al. (2011) são levantados dois importantes pontos de um (LPS) a primeira delas é a variabilidade. A variabilidade é fundamental, pois permite que diferentes produtos na LPS tenham combinações distintas de características e comportamentos para atender às necessidades específicas dos clientes ou segmentos de mercado. Ela é projetada e gerenciada para lidar com as diferenças entre os produtos, abrangendo diversos aspectos, como funcionalidade, interface do usuário, requisitos de desempenho e suporte multiplataforma. Isso possibilita a personalização dos produtos com base em um conjunto comum de recursos disponíveis.

Já a gestão de variabilidade em uma linha de produto de *software* (LPS) é uma atividade essencial para garantir o sucesso da linha e maximizar os benefícios da reutilização e personalização. Ela envolve o planejamento, o controle e a organização dos recursos relacionados à variabilidade dentro da LPS. Alguns aspectos importantes da gestão de variabilidade em LPS são a identificação, modelagem, gerenciamento, reutilização, entre outros.

Em resumo, a gestão de variabilidade em uma LPS envolve a identificação, a modelagem, o controle e a organização dos recursos relacionados à variabilidade.

2.4.3 *Feature Model*

O *feature model* é o principal mecanismo para modelar a variabilidade em uma linha de produto de *software* (LPS). As *features* representam características do sistema visíveis aos usuários finais e unidades de comportamento usadas para especificar requisitos funcionais e de qualidade relevantes para as partes interessadas. Elas também ajudam a agrupar requisitos relacionados. O objetivo das *features* é reduzir a distância entre usuários finais e desenvolvedores em relação aos requisitos, apresentando as principais características comuns e variáveis em uma LPS em alto nível.

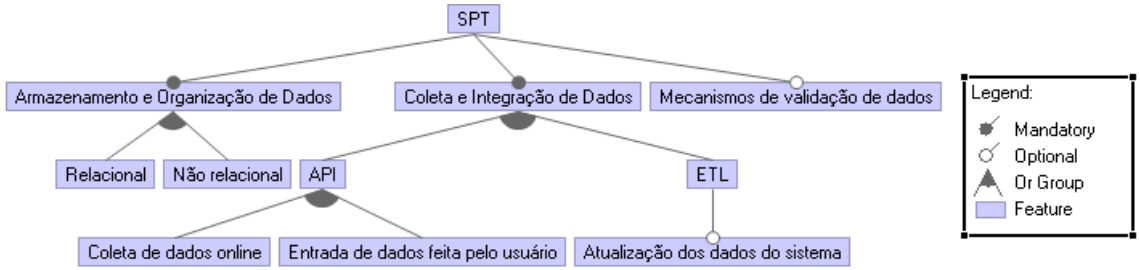
2.4.3.1 Notação F.O.D.A

Como descrito em Silva et al. (2011) a notação F.O.D.A (*Feature Oriented Domain Analysis*) oferece uma maneira estruturada de descrever e organizar as características em uma LPS. Ela é baseada em uma estrutura hierárquica de características, em que cada característica é representada como um nó no modelo. Essas características podem ser organizadas em níveis diferentes, permitindo uma visualização clara das relações entre elas.

Além da estrutura hierárquica, a notação F.O.D.A também utiliza um conjunto de símbolos para representar as características e suas propriedades. Por exemplo, uma característica pode ser representada por um retângulo com o nome da característica dentro dele, enquanto uma característica opcional pode ser representada por um retângulo pontilhado.

A notação F.O.D.A também permite a representação de restrições entre características. Por exemplo, é possível indicar que determinadas características são mutuamente exclusivas, ou que uma característica só pode ser selecionada se outra estiver presente.

Figura 3 - Feature model.



Legenda: Feature model de um sistema de processamento de transações.

Fonte: Do autor (2023).

A figura 3 é um exemplo de um pequeno diagrama criado com a notação F.O.D.A, onde é possível ver a representação de um sistema de processamento de transações com suas características e restrições.

O modelo possui duas *feature* obrigatórias e uma *feature* opcional sendo elas respectivamente armazenamento e organização de dados, colete e integração de dados e mecanismos de validação de dados. Além disso, todas as *features* obrigatórias são compostas por duas *sub-features* numa alternativa OR (uma ou as duas podem ser escolhidas).

2.5 Levantamento de requisitos

Os requisitos são descrições das necessidades dos clientes, indicando o que o sistema deve realizar. A engenharia de requisitos engloba o processo de identificação, análise, documentação e verificação desses requisitos, abrangendo os serviços oferecidos e as restrições operacionais do sistema. Isso assegura que o sistema cumpra com os objetivos pretendidos. O termo "requisito" na indústria de *software* é utilizado de maneiras diversas, podendo abranger desde declarações abstratas de alto nível até definições detalhadas e formais das funções do sistema.

Durante o processo de engenharia de requisitos, é frequente nos depararmos com desafios decorrentes da falta de uma distinta separação entre os diversos níveis de descrição. Para solucionar esse problema, empregamos a terminologia "requisitos de usuário" para denotar às necessidades abstratas e de alto nível, enquanto utilizamos "requisitos de sistema" para detalhar minuciosamente as funcionalidades do sistema. Portanto, os requisitos de usuário apresentam uma visão geral, ao passo que os requisitos de sistema oferecem especificações pormenorizadas das tarefas que o sistema deve desempenhar.

3 METODOLOGIA

Este capítulo apresenta toda a metodologia utilizada para a elaboração do protótipo da ferramenta de construção de *software*, descrevendo passo a passo os estudos feitos e as ferramentas utilizadas. Esta metodologia foi desenvolvida pelo autor em colaboração com o professor orientador do trabalho.

3.1 Preparação

Este passo contém as seguintes atividades:

a) seleção dos *software de entrada:*
   - s*oftware* de geoprocessamento sendo eles o geonode, mapstore e o georadar;
b) identificação das funcionalidades generalizáveis ou seja, levantar todas as características em comum que os *software* selecionados possuem.

3.2 Levantamento de requisitos

Neste passo foi feito o levantamento dos requisitos que a ferramenta deverá atender. Estes requisitos estão separados entre requisitos de sistema e requisitos do SIG (requisitos de usuário).

A fonte dos requisitos são os *software* de geoprocessamento selecionados no primeiro passo. Para levantar os requisitos comuns dos três sistemas foi necessário realizar uma análise detalhada das necessidades específicas de cada sistema, bem como identificar os elementos essenciais que são compartilhados entre eles.

3.3 Diagrama F.O.D.A

Neste passo será criado um diagrama utilizando os conceitos de uma linha de produção de *software* para modelar a relação entre os requisitos encontrados.

3.4 Arquitetura hexagonal

Após o levantamento de requisitos será construído um diagrama utilizando os conceitos

da arquitetura hexagonal para esboçar uma possível arquitetura que a ferramenta de construção de *software* poderia utilizar para construir as funcionalidades do sistema que o usuário escolher.

3.5 Protótipos

3.5.1 Protótipo de baixa fidelidade

Neste passo será produzido um primeiro esboço do sistema. Após a criação do esboço será feita uma análise do mesmo para listar as principais dificuldades enfrentadas.

3.5.2 Protótipo final

Após o levantamento de requisitos e a construção do protótipo de baixa fidelidade será construído o protótipo da ferramenta de construção de *software*. A ferramenta será prototipada em um *software* gratuito de construção de interface de forma a poder demonstrar o seu funcionamento.

3.6 Implementação de um pequeno SIG

Neste passo, será desenvolvido um pequeno Sistema de Informação Geográfica (SIG) utilizando o *OpenLayers*, oferecendo algumas funcionalidades fundamentais típicas de um SIG. Este sistema utiliza o *OpenLayers* como infraestrutura e tem como propósito representar um exemplo de sistema que pode ser automaticamente gerado pela ferramenta de criação de *software*.

4 RESULTADOS E DISCUSSÕES

4.1 Preparação

4.1.1 Seleção dos *software*

Entre os diversos *software* desenvolvidos na zetta tais como Radar, MPF-Análise, PGST entre outros foram selecionados apenas os sistemas que trabalham com geoprocessamento além de outros *software* gratuitos que também trabalham com geoprocessamento, sendo eles o geonode, o *MapStore* e o GeoRadar.

4.1.2 Identificação das funcionalidades generalizáveis

Aqui estão alguns dos pontos em comum entre os 3 *Software*:

a) publicação e compartilhamento de dados:
   - os 3 sistemas permitem publicar, compartilhar e visualizar dados geoespaciais;
b) personalização:
   - ambos os sistemas oferecem opções de personalização, permitindo que você adapte a aparência e funcionalidade dos aplicativos de mapeamento;
c) suporte a padrões geoespaciais:
   - os 3 sistemas atendem aos padrões geoespaciais, tornando-os adequados para ambientes que exigem conformidade com normas específicas;
d) integração de dados:
   - os 3 sistemas suportam a integração de dados de várias fontes e formatos;
e) gerenciamento de metadados:
   - os 3 sistemas oferecem recursos para gerenciar metadados de forma a melhorar a busca e a organização de dados geoespaciais;
f) recursos de colaboração:
   - os 3 sistemas permitem a colaboração em projetos geoespaciais, facilitando o trabalho em equipe;

4.2 Levantamento de requisitos

4.2.1 Requisitos do SIG

Nesta seção vamos levantar os requisitos esperados para um SIG tendo em vista que os futuros usuários do sistema têm as necessidades de capturar, armazenar, manipular, analisar e visualizar dados geográficos.

A maioria dos requisitos levantados abaixo são comuns na maioria dos SIGs e o autor conhece as necessidades comuns do que o usuário precisa no sistema:

a) aquisição de dados geográficos; isto é, o sistema deve ser capaz de adquirir dados geográficos de diferentes fontes, como mapas, imagens de satélite, dados de sensoriamento remoto, levantamentos terrestres, etc;
b) armazenamento de dados; isto é, o sistema deve ser capaz de armazenar dados geográficos em um formato adequado e eficiente, como bancos de dados espaciais, arquivos shapefile, geodatabases, entre outros;
c) análise espacial; isto é, o sistema deve fornecer ferramentas para realizar análises espaciais, como consultas espaciais, análise de proximidade, análise de padrões, análise de redes, interpolação espacial, entre outras;
d) integração de dado; isto é, o sistema deve permitir a integração de diferentes conjuntos de dados geográficos, permitindo a sobreposição e análise conjunta de informações;
e) visualização e representação; isto é, o sistema deve permitir a visualização e representação dos dados geográficos em forma de mapas, gráficos, tabelas e outros formatos visuais;
f) edição e atualização de dados; isto é, o sistema deve fornecer recursos para editar e atualizar os dados geográficos, incluindo a criação, exclusão e modificação de elementos espaciais;
g) compartilhamento de informações; isto é, o sistema deve permitir o compartilhamento de dados geográficos e informações derivadas, possibilitando o acesso e colaboração entre diferentes usuários ou departamentos;
h) integração com outras aplicações; isto é, o sistema deve ser capaz de integrar-se com outras aplicações, como sistemas de gerenciamento de banco de dados, *software* de análise estatística ou seja sistemas de planejamento urbano, entre outros;
i) segurança e controle de acesso; isto é, o sistema deve garantir a segurança dos dados geográficos, incluindo controle de acesso para proteger informações sensíveis e garantir a privacidade;
j) suporte a padrões e interoperabilidade; isto é, o sistema deve aderir a padrões e

protocolos de interoperabilidade para permitir a troca de dados com outros sistemas e facilitar a integração com tecnologias futuras.

4.2.2 Requisitos de sistema

Os requisitos do sistema definem as funcionalidades, características e restrições que o sistema deve atender para satisfazer as necessidades dos usuários. Eles podem ser divididos em diferentes categorias para uma melhor compreensão e organização.

Aqui estão os quatro tópicos comuns para descrever requisitos do sistema:

a) função; isto é, o que o sistema deve fazer?;
b) serviços; isto é, como o sistema deve responder em diferentes situações?;
c) restrições operacionais; isto é, quais são as limitações impostas ao sistema?;
d) tipo; isto é, como esse requisito pode ser implementado ? como um módulo ou funcionalidade ? Sendo que o módulo é uma parte isolada e independente do sistema que desempenha uma função específica e a funcionalidade é o que o sistema é capaz de fazer para atender a determinadas necessidades ou requisitos.

Os requisitos do sistema que estão listados abaixo foram subdivididos em função, serviços, restrições operacionais e tipo, respectivamente:

a) aquisição de dados geográficos:

   - permitir a obtenção de dados geográficos de diversas fontes tais como *OpenStreetMap*, *Google Maps*, Agências governamentais entre outros;

   - incluir suporte para importação de mapas, imagens de satélite, dados de sensoriamento remoto, levantamentos terrestres, entre outros;

   - garantir a compatibilidade dos formatos de dados e capacidade de processar diferentes tipos de dados geográficos;

   - este requisito pode ser construído como um módulo visto que o mesmo precisa tratar a entrada de vários formatos de arquivos diferentes.

b) armazenamento de dados:

   - armazenar e gerenciar os dados geográficos adquiridos utilizando os seguintes formatos *Shapefile, GeoJSON, KML, GeoTIFF e CSV*;

   - incluir opções para armazenar dados em bancos de dados espaciais (*SGBD*), arquivos shapefile, geodatabases, ou outros formatos apropriados;

   - garantir a eficiência de armazenamento, escalabilidade e segurança dos dados;

   - este requisito pode ser construído como um módulo do sistema.

c) integração de dados:

- permitir a combinação e integração de diferentes conjuntos de dados geográficos e oferecer ferramentas para a padronização dos dados;
- oferecer ferramentas para sobreposição, fusão e análise conjunta de dados geográficos;
- garantir a compatibilidade dos formatos de dados, resolução espacial e lidar com problemas de integração, como diferentes projeções e sistemas de coordenadas;
- este requisito pode ser construído como uma funcionalidade do sistema visto que o mesmo precisará agrupar várias funcionalidades diferentes.

d) análise espacial:

- oferecer análise por proximidade (Avalia a relação espacial entre elementos, como a proximidade entre pontos ou a distribuição de recursos ao redor de um ponto de referência), análise por cluster (Identifica agrupamentos ou concentrações espaciais de elementos semelhantes);
- incluir recursos para consultas espaciais, análise de proximidade, análise de padrões espaciais, análise de redes, interpolação espacial, entre outros;
- lidar com grandes volumes de dados, fornecer algoritmos e métodos robustos para análise espacial e garantir o desempenho eficiente das operações;
- este requisito pode ser construído como uma funcionalidade do sistema.

e) visualização e representação:

- apresentar dados geográficos de forma visualmente compreensível sendo elas mapas temáticos, gráficos e diagramas, infográficos, animações, uso de cores e símbolos adequados, utilização de legenda e rótulos;
- oferecer opções para a criação de mapas, gráficos, tabelas e outras representações visuais;
- garantir a capacidade de renderização rápida, suporte a diferentes tipos de símbolos, cores e estilos de visualização, além de permitir a interação com as visualizações;
- este requisito pode ser construído como uma funcionalidade do sistema.

f) edição e atualização de dados:

- permitir a modificação e atualização dos dados geográficos ou seja após selecionar os elementos, é possível realizar alterações nos atributos associados a eles. Os atributos representam informações não espaciais, como nome, tipo, valor, data, entre outros. A edição de atributos pode incluir a adição, exclusão ou atualização de

valores, dependendo das necessidades do usuário e das restrições definidas no banco de dados. Permitir a edição de geometria ou seja além da modificação de atributos, o SIG permite a edição da geometria dos elementos geográficos. Isso envolve a capacidade de mover, redimensionar, adicionar ou excluir pontos, linhas ou polígonos. Isso pode incluir a verificação de restrições de formato, intervalos de valores aceitáveis e outras regras definidas no banco de dados. Se os dados não atenderem a essas regras, o SIG pode sinalizar erros ou avisos para o usuário;

- incluir recursos para criar, editar e excluir elementos espaciais, bem como atributos relacionados;

- fornecer controle de concorrência em ambientes de edição simultânea, garantir a consistência dos dados e oferecer mecanismos de desfazer/refazer;

- este requisito pode ser construído como um módulo pois será necessário implementar várias funcionalidades diferentes para possibilitar o funcionamento do mesmo.

g) compartilhamento de informações:

- permitir o compartilhamento de dados geográficos e informações derivadas;

- incluir opções para compartilhar dados e resultados de análises com outros usuários ou departamentos;

- possíveis restrições operacionais podem ser o compartilhamento eficiente e seguro dos dados geográficos, tamanho de arquivo, segurança e privacidade, licenciamento e direitos autorais, acesso e permissões, interoperabilidade, sincronização e atualização, largura de banda e velocidade de transferência;

- este requisito pode ser implementado como uma funcionalidade.

A definição e elencagem dos requisitos do sistema apresenta desafios significativos que poderiam surgir durante a implementação do sistema. A aquisição de dados geográficos, por exemplo, enfrenta a dificuldade de lidar com diversas fontes, como *OpenStreetMap* e *Google Maps*, exigindo um módulo especializado para processar diferentes formatos de arquivos. O armazenamento de dados também representa um desafio, com a necessidade de garantir eficiência, escalabilidade e segurança, indicando a construção de um módulo dedicado para essa funcionalidade.

A integração de dados, por sua vez, demanda uma abordagem multifacetada, incorporando diversas funcionalidades do sistema para agrupar diferentes conjuntos de dados geográficos. A análise espacial e visualização requerem uma complexidade adicional, envolvendo o desenvolvimento de funcionalidades específicas para consultas, sobreposição de

dados e apresentação visual compreensível.

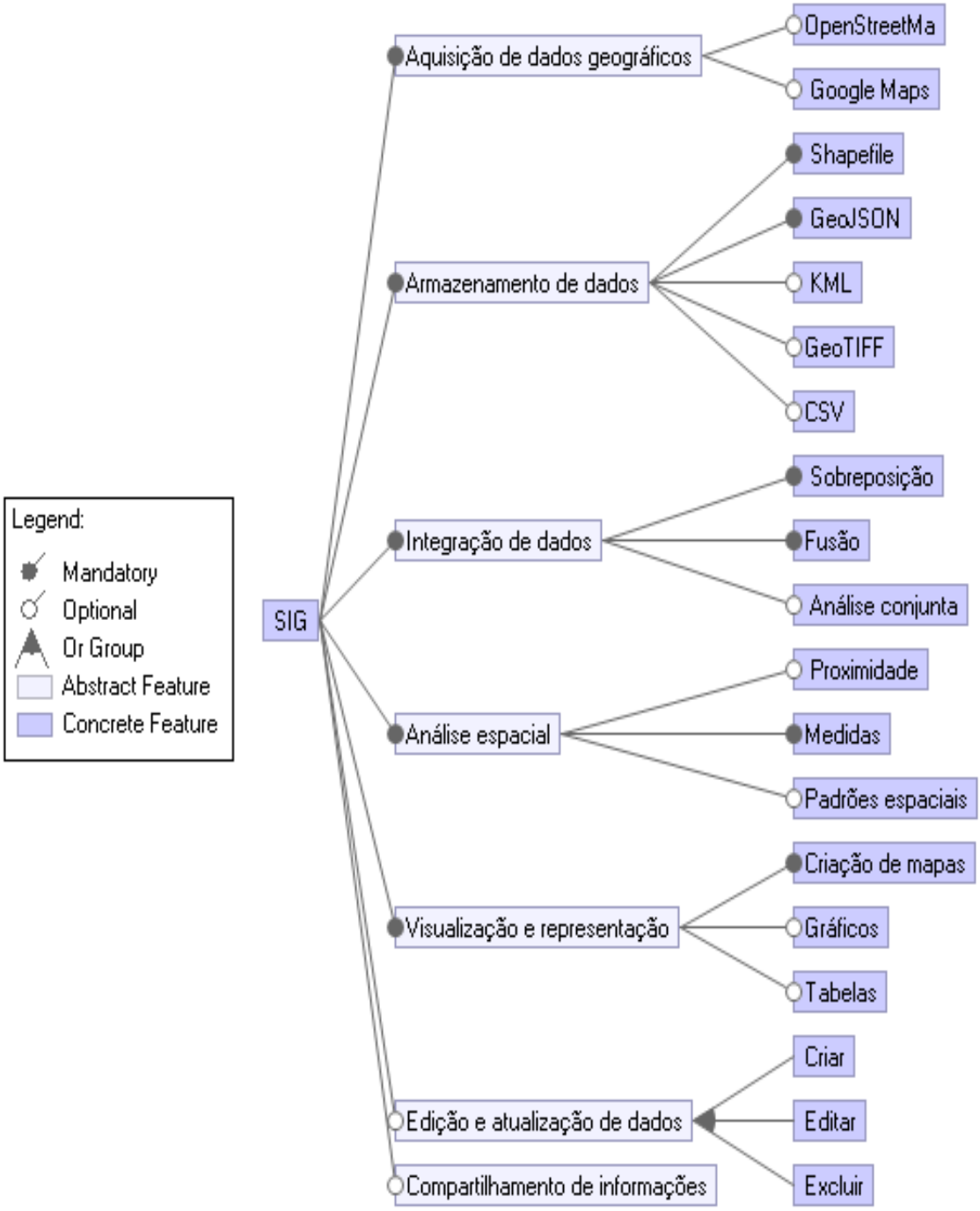
A edição e atualização de dados apresentam um conjunto diversificado de desafios, desde a modificação de atributos até a edição da geometria dos elementos geográficos, indicando a necessidade de um módulo abrangente. O compartilhamento de informações, por fim, destaca a complexidade associada à eficiência, segurança, interoperabilidade, licenciamento e outros aspectos operacionais.

Essas dificuldades ressaltam a importância de uma abordagem cuidadosa na implementação de requisitos do sistema, levando em consideração a natureza multifacetada e interconectada das funcionalidades necessárias para atender às demandas dos usuários e garantir o desempenho eficaz do sistema.

4.3 Diagrama F.O.D.A

Abaixo segue o diagrama construído com a notação F.O.D.A para representar quais atributos levantados são obrigatórios, quais são opcionais, as restrições existentes entre eles e quais deles são características abstratas ou concretas.

Figura 4 – Modelo feature de um SIG.



Legenda: Feature model de um sistema de informações geográficas que modela as relações existentes entre os atributos levantados.

Fonte: Do autor (2023).

Analisando o diagrama foi possível identificar respectivamente as seguintes caracteristicas abstratas e concretas:

a) aquisição de dados geográficos, armazenamento de dados, integração de dados, análise espacial, visualização e representação de dados geográficos, edição e atualização de dados e compartilhamento de informações.

   As características abstratas são elementos que não são facilmente quantificáveis, mas que têm um impacto significativo na dinâmica e na eficácia da linha de produção de *software*. Das características abstratas levantadas apenas duas não são obrigatórias sendo elas a edição e atualização de dados e o compartilhamento de informações. Estas características não são obrigatórias visto que o sistema pode ser apenas para visualização de dados ou os dados armazenados não podem ser alterados por alguma razão. Já o compartilhamento de informações também não é obrigatório visto que os dados podem possuir restrições de acesso que impeçam o compartilhamento;

b) *OpenStrigMap, googleMaps, Shapefile*, *GeoJson*, *KML*, *GeoTIFF, CSV*, sobreposição, fusão, análise conjunta, proximidades, medidas, padrões especiais, criação de mapas, gráficos, tabelas, criar editar e excluir.

   As características concretas na linha de produção de *software* referem-se a elementos tangíveis, mensuráveis e observáveis que impactam diretamente a eficiência. Das características concretas levantadas são obrigatórias somente aquelas que são fundamentais para implementar a funcionalidade já as demais são opcionais.

Por fim podemos concluir que o *feature* model proposto fornece uma visão abrangente das funcionalidades do sistema, destacando as características essenciais para aquisição, armazenamento, análise e visualização de dados geográficos. A clara distinção entre características abstratas e concretas facilita a compreensão e o desenvolvimento do sistema, permitindo uma implementação modular e adaptável.

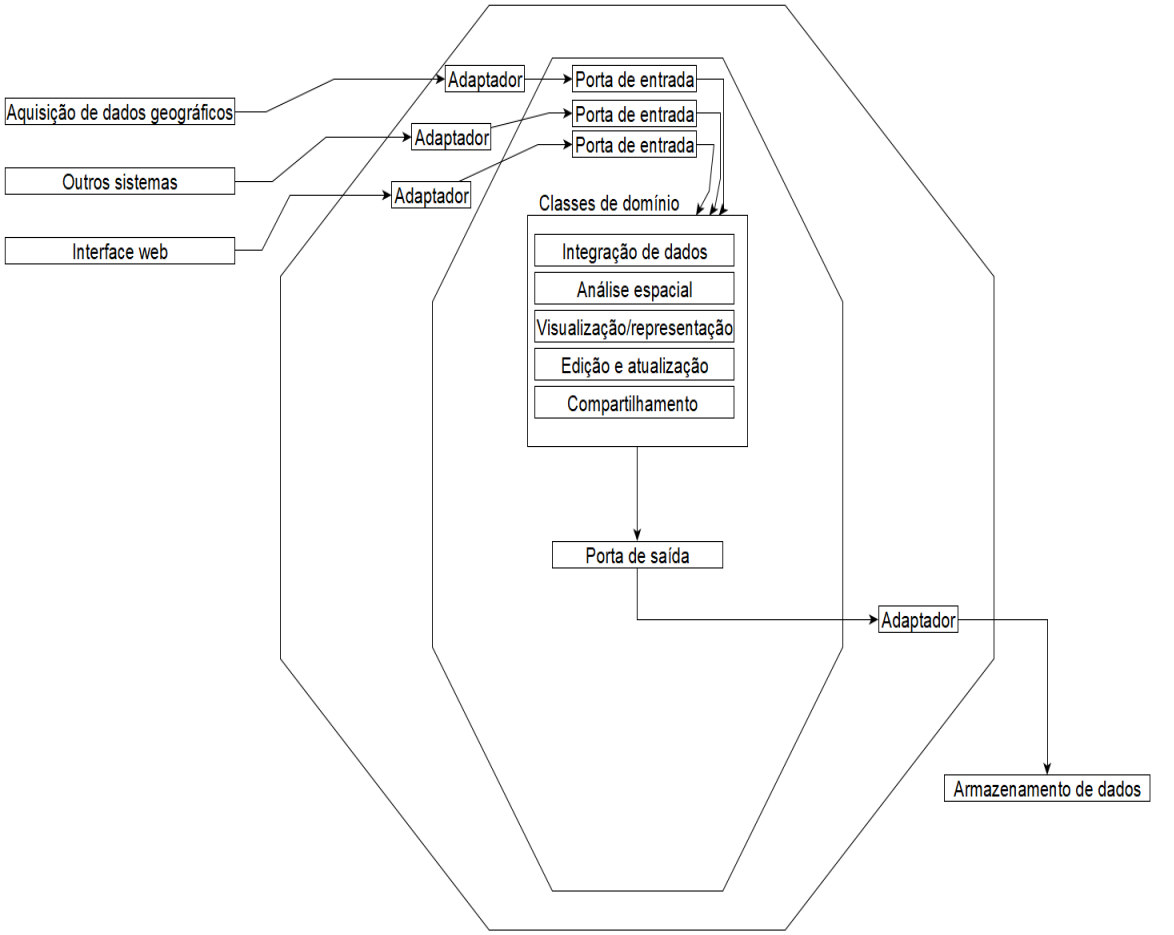
4.4 Arquitetura hexagonal

Após o levantamento de requisitos foi possível a criação de um diagrama utilizando o conceito da arquitetura hexagonal que fornece um modelo conceitual de como este sistema poderia ser implementado.

Todos os módulos ou seja estruturas independentes do sistema que foram identificadas durante o levantamento de requisitos são representados do lado de fora dos hexágonos. Já as

funcionalidades identificadas durante o levantamento de requisitos são representadas como classes de domínio e aparecem no hexágono interno.

Figura 5 - Arquitetura hexagonal.



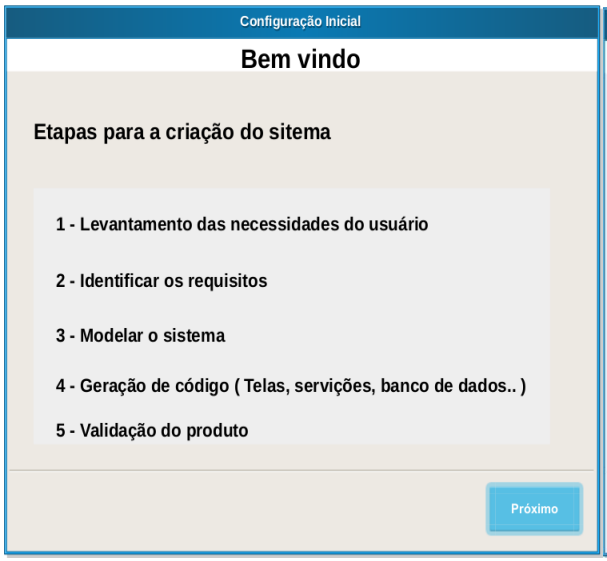
Legenda: Arquitetura que a ferramenta de construção de software irá utilizar para gerar os sistemas.

Fonte: Do autor (2023).

4.5 Criação do protótipo de baixa fidelidade

A criação do protótipo de baixa fidelidade tem como objetivo principal proporcionar uma representação visual rudimentar da futura ferramenta de desenvolvimento de *software*. Este protótipo serve como uma base inicial para avaliar e validar as funcionalidades essenciais, fluxos de trabalho e interações que serão incorporados no protótipo final da ferramenta.

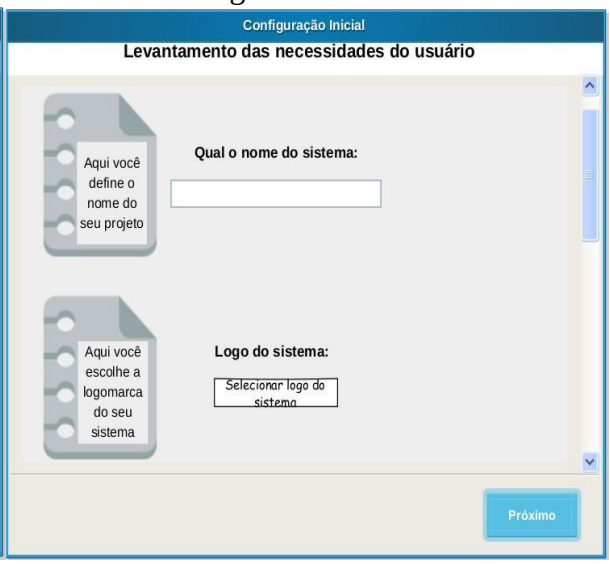
Figura 6 - Tela de apresentação.



Legenda: Tela de apresentação do sistema onde são listadas as etapas que a ferramenta utiliza para criar o software.

Fonte: Do autor (2023).

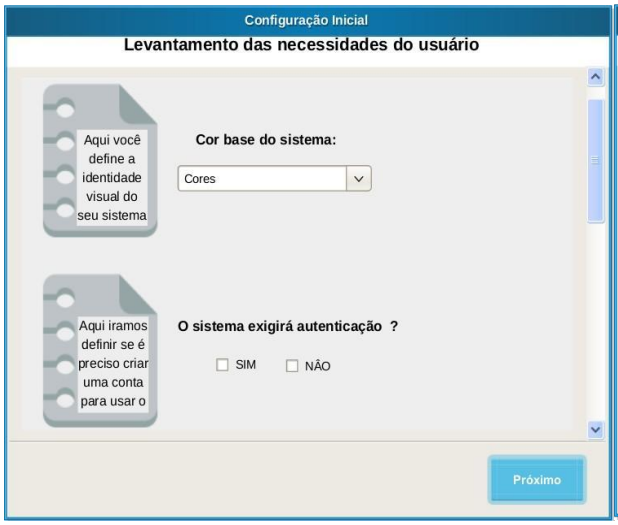
Figura 7 - Início.



Legenda: Leitura do nome e da logo do sistema.

Fonte: Do autor (2023).

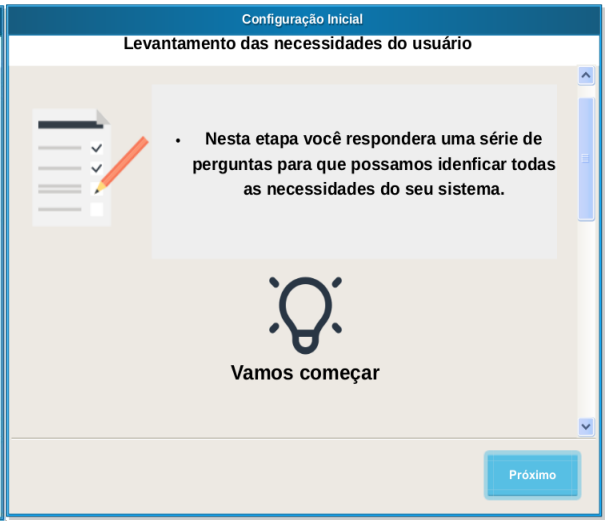
Figura 8 - Personalização.



Legenda: Personalização do sistema.

Fonte: Do autor (2023).

Figura 9 - Aviso.



Legenda: Aviso ao usuário das próximas etapas.

Fonte: Do autor (2023).

Figura 10 - Escolha do tipo do sistema.

Legenda: Escolha do tipo de sistema que a ferramente irá grar.

Fonte: Do autor (2023).

Figura 11- Aviso.

Legenda: Alerta para o usuário sobre a próxima etapa.

Fonte: Do autor (2023).

Figura 12 - Escolha de funcionalidades.

Legenda: Lista de funcionalidades que o SIG poderá oferecer.

Fonte: Do autor (2023).

Figura 13 - Geração do sistema.

Legenda: Enceramento do processo de criação do sistema.

Fonte: Do autor (2023).

O protótipo começa com a tela inicial de apresentação descrevendo os passos que a ferramenta seguirá para gerar o sistema, logo após a tela de apresentação as telas 6 e 7 leram as primeiras configurações básicas do sistema. As telas 8, 10 e 12 possuem apenas alertas para os

usuários sobre as próximas etapas que serão seguidas. As telas principais do sistema são respectivamente as telas 9 e 11 onde será definido o tipo do sistema que o usuário irá trabalhar e a escolha das funcionalidades que o sistema final deverá oferecer.

4.6 Análise do protótipo de baixa fidelidade

Foram encontrados alguns desafios para se construir o protótipo de baixa fidelidade, sendo eles:

a) funcionalidade limitada; protótipos de baixa fidelidade são mais simples e menos detalhados, o que pode prejudicar a representação precisa do produto final;
b) dificuldades no teste; esses protótipos têm desempenho inferior aos de alta fidelidade, dificultando o teste e a validação de ideias de design;
c) visualização complexa; antecipar o produto final a partir de um protótipo de baixa fidelidade é complicado, levando a equívocos e expectativas incorretas;
d) apresentação desafiadora; apresentar protótipos de baixa fidelidade às pessoas não familiarizadas com o processo de design pode ser difícil, prejudicando a obtenção de feedback e a comunicação da visão do produto.

Diante destes desafios foi necessário uma grande pesquisa e análise de ferramentas já existentes de criação de *software* como e GeneXus para fazer o levantamento das primeiras ideias para o projeto. Após esta pesquisa foi possível a criação de um protótipo rudimentar mas que fornecia uma boa ideia do caminho a ser seguido.

4.7 Criação do protótipo final

O protótipo final está disponível no MarvelAPP e pode ser acessado pelo link: https://marvelapp.com/prototype/68ejfjf/screen/90288894

Este protótipo oferece uma abordagem intuitiva e amigável, permitindo que desenvolvedores e profissionais de geotecnologia construam aplicações complexas de forma eficiente e eficaz. Com recursos de arrastar e soltar, os usuários podem facilmente incorporar elementos geoespaciais em suas aplicações, como mapas interativos, marcadores, camadas de dados e ferramentas analíticas.

Além disso, o protótipo apresenta uma biblioteca abrangente de componentes pré-construídos específicos para geoprocessamento, reduzindo significativamente o tempo de

desenvolvimento. Esses componentes incluem widgets para análise espacial, visualização de dados geográficos e interação com serviços de mapas.

4.8 Sistema implementado

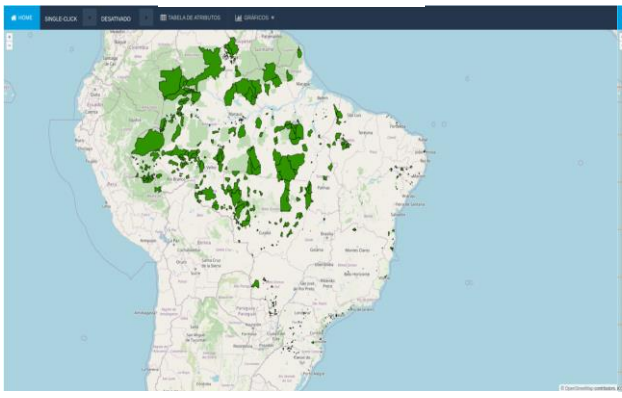
O sistema implementado foi construído com o *OpenLayers* que é uma biblioteca de código aberto em JavaScript que é usada para criar mapas interativos e aplicativos de mapeamento na web. Ele fornece uma estrutura poderosa e flexível para exibir mapas geoespaciais em páginas da web. O sistema implementado possibilita carregar camadas, e consultar os dados da mesma além da geração de gráficos. O Sistema pode ser visto no protótipo de alta fidelidade e está disponível no *GitHub* https://github.com/ClaudineiMS/SIG

O sistema foi implementado em duas semanas e é relativamente pequeno possuindo pouco mais de 1000 linhas de código. O sistema segue uma estrutura de pasta básica separando cada componente em um diretório. Ou seja, dentro de cada diretório está o *HTML*, o *CSS* e o *Js* que faz cada componente funcionar.

As maiores dificuldades enfrentadas para a construção deste sistema foram entender o funcionamento do *OpenLayers*, criar uma interface de usuário interativa e responsiva, fazer um tratamento eficiente de eventos como cliques e arrastos, integrar a aplicação com outras ferramentas de criação de gráficos e por fim a utilização de *Js* puro também trás algumas dificuldades visto que muitas coisas precisam ser implementadas do zero quando um *framework* não é utilizado.

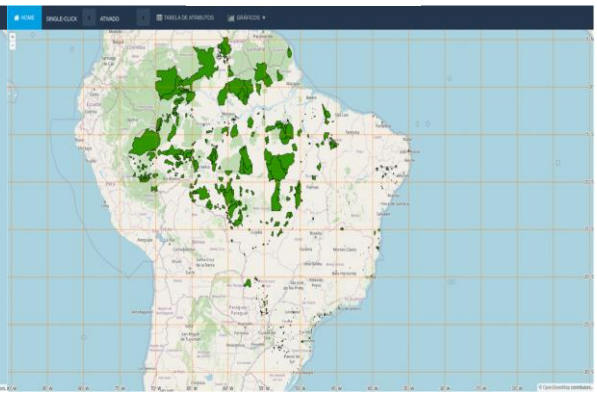
4.8.1 Telas do sistema

Figura 14 - home.



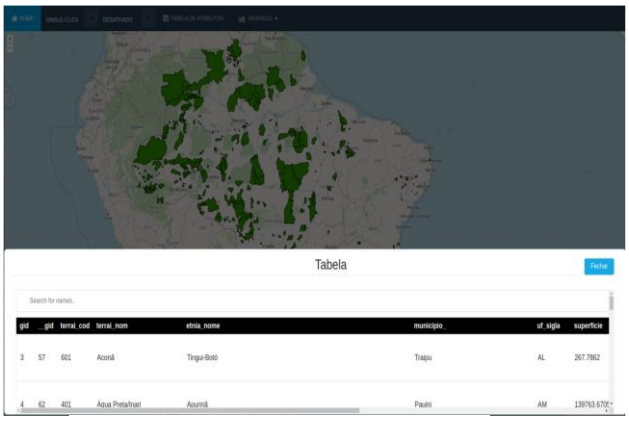
Legenda: Página inicial do sistema com uma camada carregada.
Fonte: Do autor (2023).

Figura 15 - Grid.



Legenda: Grid de coordenadas plotado sobre o mapa.
Fonte: Do autor (2023).

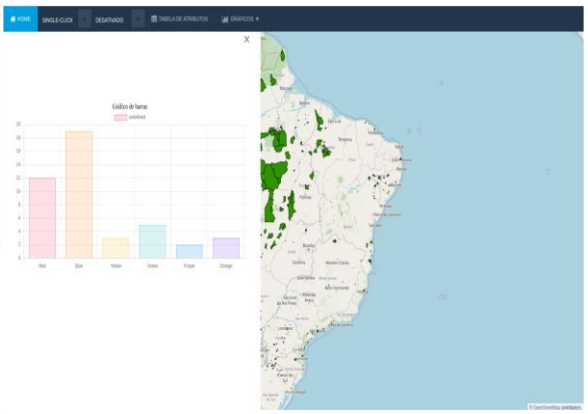
Figura 16 - Tabela de atributos.



Legenda: Tabela com todos os dados da camada carregada no

Fonte: Do autor (2023).

Figura 17 - Gráficos.



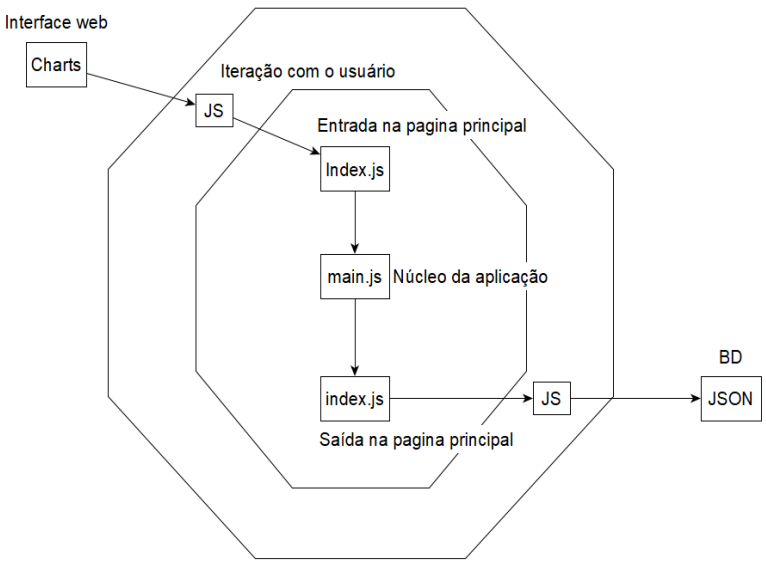
Legenda: Gráficos gerados a partir dos dados da camada.

Fonte: Do autor (2023).

O sistema implementado disponibiliza na sua página principal uma camada pública (Terras indígenas) e algumas ferramentas para que o usuário possa visualizar e trabalhar com os dados dessa camada. As ferramentas disponibilizadas são clicke em um ponto do mapa para retornar as informações daquele ponto, uma tabela de atributos listando todos os dados da camada, um menu de geração de gráficos com os dados da camada e um *grid* de coordenadas plotado sobre o mapa.

4.8.2 Arquitetura do sistema

Figura 18 - Arquitetura hexagonal.



Legenda: Arquitetura utilizada para implementar o SIG.

Fonte: Do autor (2023).

O diretório com nome *Charts* é onde está localizada toda a interface web do sistema, a partir dele são feitos todos os *requests* para o adaptador implementado em *JavaScript* que redireciona essas chamadas para a entrada da aplicação. A partir deste ponto as chamadas são redirecionadas para o núcleo da aplicação para serem resolvidas e devolvidas ao usuário.

4.9 Materiais

Para a criação do diagrama F.O.D.A foi utilizado a IDE eclipse e foi feita a instalação de um plugin chamado *FeatureIDE* que possibilita a criação do diagrama e a exportação do mesmo no formato png.

Para a criação do modelo representando a arquitetura hexagonal foi utilizado uma ferramenta gratuita de criação de diagramas chamada *yEd graph editor*.

Para a criação do protótipo de baixa fidelidade foi utilizado um *software* gratuito disponível para *Linux* chamado *Pencil Project*. Já para a criação do protótipo final foi utilizado uma ferramenta gratuita chamada *Marvel APP* que permite a prototipação de diversos tipos de interface e a simulação de um sistema complexo sem a necessidade de fazer qualquer tipo de implementação ou conhecimento prévio em alguma linguagem de programação.

Para a implementação do SIG foi utilizado o *NodeJs* além de *HTML, CSS e o ChatsJS* para criação dos gráficos.

A principal dificuldade enfrentada para utilização destes materiais foi a curva de aprendizado, visto que cada *software* empregado demanda um certo tempo para que seus recursos sejam conhecidos e utilizados de forma correta.

5 CONCLUSÕES E TRABALHOS FUTUROS

Em resumo, o objetivo principal deste trabalho foi propor a criação de uma ferramenta para o desenvolvimento de *software*, visando simplificar o processo de criação e promover a eficiência na produção.

Durante o desenvolvimento deste TCC, várias dificuldades e facilidades foram encontradas. As principais dificuldades incluíram a complexidade na produção de ferramentas de geoprocessamento e a baixa quantidade de *software* disponíveis na zetta para a análise. Por outro lado, as facilidades encontradas incluíram a abundância de recursos de código aberto de geoprocessamento e a abundância de ferramentas de construção do código com low-code ou no-code.

Além disso, a realização deste TCC nos permitiu explorar e aplicar conceitos de diversas disciplinas, incluindo Engenharia de *Software*, Programação, *Design* de *Interfaces* e Gerenciamento de Projetos. A integração desses conhecimentos foi fundamental para o desenvolvimento bem-sucedido da ferramenta proposta.

Com base em todas as etapas realizadas e nas conclusões tiradas ao longo deste estudo, acredita-se que a criação da ferramenta proposta tem o potencial de simplificar o desenvolvimento de *software* de geoprocessamento, tornando o processo mais acessível e eficiente.

Espera-se que este trabalho tenha contribuído para o avanço na área de desenvolvimento de *software* e para a criação de ferramentas mais eficazes que beneficiem os profissionais da área e as empresas que dependem da criação de *software* de alta qualidade.

Em trabalhos futuros podem ser analisados mais sistemas disponíveis e extrair mais recursos para a ferramenta. Além disso, também pode se disponibilizar suporte para mais tipos de sistemas.

REFERÊNCIAS

ENGENHARIA de Software Moderna. **Engenharia de Software Moderna**, 2022. Disponível em : https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html#:~:text=Uma%20Arquitetura%20Hexagonal%20divide%20as,tais%20como%20bancos%20de%20dados). Acesso em: 24 out. 2023.

KALINOWSKI, M.; SPÍNOLA, R. O. Introdução à inspeção de software. **Engenharia de Software Magazine**, v. 1, n.1, p. 68-74, 2008. Disponível em: https://www-di.inf.puc-rio.br/~kalinowski/publications/KalinowskiS07.pdf. Acesso em: 24 out. 2023.

RAFAEL. LINHA de Produto de Software. **DEVMEDIA**, 2023. Disponível em: https://www.devmedia.com.br/linha-de-produto-de-software/33894. Acesso em: 29 out. 2023.

MERCADO Brasileiro de Software. **ABES**, 2023. Disponível em: https://abes.com.br/dados-do-setor/. Acesso em: 14 nov. 2023.

ROSOLEN, Dani. Verbete Draft: o que são Low-Code e No-Code. **Projeto Draft**, 2021. Disponível em: https://www.projetodraft.com/verbete-draft-o-que-sao-low-code-e-no-code/#:~:text=O%20que%20%C3%A9%3A%20Low%2Dcode,desenvolvimento%20de%20produtos%20e%20servi%C3%A7os. Acesso em: 02 nov. 2023.

SILVA, F. A. P. *et al*. Linhas de Produtos de Software: Uma tendência da indústria. *In*: SANTANA, A. M.; SANTOS NETO, P. A.; MOURA, R. S. **V Escola Regional de Informática Ceará, Maranhão, Piauí:** Livro texto dos minicursos. [*S.l.*]: Sociedade Brasileira de Computação, 2011. Disponível em: https://www.researchgate.net/publication/236784308_Linhas_de_Produtos_de_Software_Uma_tendencia_da_industria..Acesso em: 02 nov. 2023.