Sistema web para geração de diagramas de atividade utilizando um padrão de linguagem formal e inteligência artificial.

Emanuel Fagan Bissacotti
Universidade Franciscana
Ciência da Computação
Santa Maria, RS
Email: e.bissacotti@ufn.edu.br

Claython da Silva Ludovico
Universidade Franciscana
Ciência da Computação
Santa Maria, RS
Email: claython.silva@ufn.edu.br

Fernando Torres Moreira
Universidade Franciscana
Ciência da Computação
Santa Maria, RS
Email: fernando.moreira@ufn.edu.br

Rian Beskow Friederich
Universidade Franciscana
Ciência da Computação
Santa Maria, RS
Email: rian.friedrich@ufn.edu.br

Herysson Rodrigues Figueiredo
Universidade Franciscana
Ciência da Computação
Santa Maria, RS
Email: herysson.figueiredo@ufn.edu.br

***Abstract*—Este artigo traz a criação de um sistema web utilizando java com o framework Spring Boot, para a criação de diagramas de atividade utilizando o Gemini, para que o mesmo retorne um código com sintaxe PlantUML que por fim cria uma imagem do diagrama de atividade, relatando testes realizados, erros encontrados.**

1. Introdução

Este artigo aborda o desenvolvimento de uma aplicação web para a criação de diagramas de atividades com o auxílio de inteligência artificial. O sistema utiliza as APIs do Gemini e do PlantUML para comunicação integrada.

1.1. Objetivo

Facilitar o processo de criação de diagramas de atividade gerados automaticamente a partir de um texto padronizado fornecido pelo usuário.

1.2. Estrutura

O artigo está estruturado da seguinte forma: iniciando com o resumo do mesmo, após, sua introdução trazendo seus objetivos, revisão bibliográfica mostrando dois trabalhos correlatos e citando ferramentas que serão utilizadas no seu desenvolvimento, a metodologia de desenvolvimento do artigo e a engenharia do projeto, resultados obtidos de como deve ser estabelecida a linguagem formal para evitar possiveis erros e a conclusão do artigo.

2. Revisão Bibliográfica

Esta seção trata exclusivamente de 2 trabalhos correlatos e da apresentação das ferramentas que serão utilizadas neste trabalho, dentre eles: diagrama de atividade, API, Gemini e PlantUML. Procurando dar uma breve explicada sobre cada uma delas.

2.1. Trabalhos Correlatos

O trabalho ”Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT”, desenvolvido por F. Hans-Georg, ET all. fala sobre a padronização dos retornos do chat GPT baseado no que o mesmo aprendeu à partir de informações dadas por usuários. Tal artigo define que são necessários milhares de dados para a geração de saída.

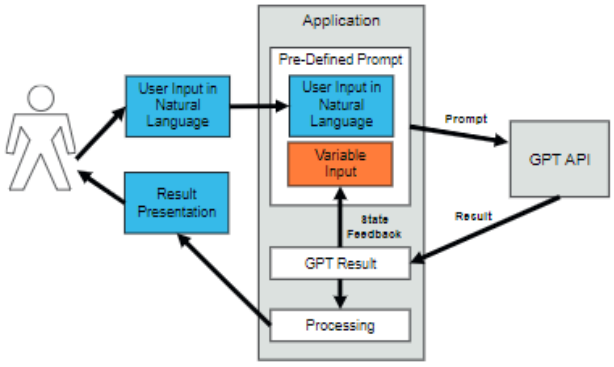

Figura 1 - Exemplo como sistema acima funciona[1].

O trabalho ”On Codex Prompt Engineering for OCL Generation: An Empirical Study”, desenvolvido por S. Abukhalaf, ET all. discute os avanços recentes em aprendizado profundo e processamento de linguagem natural e como eles levaram ao desenvolvimento de grandes modelos de linguagem destacando-o para a geração de código e

explorando o uso desses modelos no contexto da Linguagem de Restrição de Objetos (OCL) para melhorar o desenvolvimento de software orientado a modelos.

2.2. Ferramentas

Nessa subseção são discutidas as ferramentas que foram utilizadas para o desenvolvimento do sistema de geração de diagramas de atividade utilizando inteligencia artificial.

2.2.1. Diagrama de atividade. O diagrama de atividade preocupa-se em descrever os passos a serem percorridos para a conclusão de uma atividade especíica, podendo esta ser representada por um método com certo grau de complexidade, um algoritmo, ou mesmo por um processo completo.[3]

Um diagrama de atividade fornece uma visualização do comportamento de um fluxo de atividade descrevendo a sequência de ações em um processo.[4]

2.2.2. API. Application Programming Interface que, traduzida para o português, pode ser compreendida como uma interface de programação de aplicação. Ou seja, API é um conjunto de normas que possibilita a comunicação entre plataformas por meio de uma série de padrões e protocolos.[5]

APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos. Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. A aplicação para a previsão do tempo em seu telefone "fala" com esse sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.[5]

2.2.3. Gemini. Segundo a criadora do gemini a google, o Gemini dá a você acesso a modelos de linguagem grande, o modelos de linguagem(gemini) aprende com a "leitura" de trilhões de palavras que ajudam na escolha de padrões que formam a linguagem, o que permite uma resposta em padrões comuns[6]

2.2.4. PlantUML. PlantUML é uma linguagem e ferramenta de modelagem de código aberto que permite aos usuários criar diagramas a partir de uma linguagem de texto simples.[7] Ele é uma ferramenta usada em vários campos, incluindo cursos de Análise e Design de Sistemas (SAaD), para criar diagramas visuais como diagramas de Linguagem de Modelagem Unificada (UML).[10]

2.2.5. Java. Java é uma linguagem de programação orientada a objetos criada pela Sun Microsystems em 1995 e atualmente mantida pela Oracle Corporation[9]. É uma das linguagens mais populares do mundo, utilizada para desenvolver uma ampla variedade de aplicações, desde aplicativos móveis até sistemas web.[9]

3. Metodologia

Utilizamos o método SCRUM para o desenvolvimento deste projeto, devido à sua adaptabilidade, iteratividade, rapidez, flexibilidade e eficiência, características que proporcionam um valor significativo ao longo de todo o processo[11]. Os seguintes processos foram detalhados:

Backlog: Foram realizadas reuniões semanais para desenvolver novas ideias inserindo em uma lista de backlog com prazos para desenvolvimento.

Sprint backlog: Para uma maior assertividade no desenvolvimento completo da ideia de backlog, a mesma foi divida em pequenas tarefas, fazendo com que cada membro do grupo pudesse se responsabilizar por uma parte, cada uma possuindo um prazo pré-determinado.

Incremento: À partir da conclusão das pequenas tarefas, as mesmas foram incorporadas como parte de um todo, ou seja, a divisão em pequenas tarefas serviu como incremento da ideia maior de backlog.

Definição de pronto: Quando finalizada a ideia de backlog, foi analisado o desempenho de cada integrante em determinadas partes do projeto e se os objetivos foram concluídos da forma programada anteriormente pela equipe.

3.1. Engenharia do projeto

Apresentação dos dois diagramas criados para o melhor entendimento do projeto ao solicitante e ajudar o programador na codificação.

3.1.1. Diagrama de atividade. Este diagrama representa o comportamento ideal do início ao fim do software, sem que acontece qualquer imprevisto durante a sua execução, trazendo o que cada agente fará dentro do mesmo.

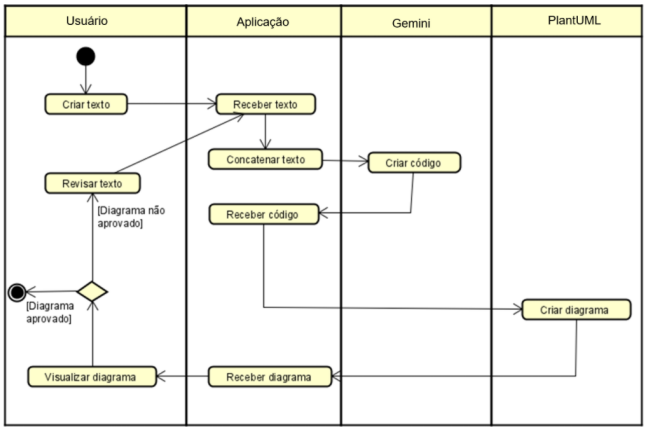

Figura 2 - Diagrama de atividade do projeto.

3.1.2. Diagrama de casos de uso. Este diagrama apresenta as funcionalidades do sistema e seu usuário.

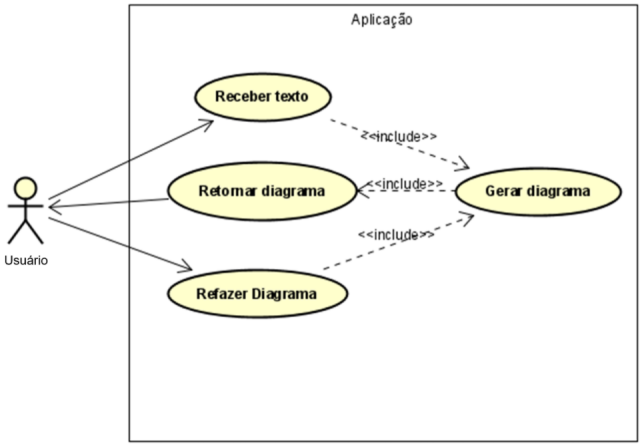

Figura 3 - Diagrama de casos de uso do projeto.

3.2. Modelagem do prompt para a geração dos diagramas

Para a criação dos diagramas de forma automatica precisamos escolher uma linguagem formal que iriamos utilizar, e para isso teria que ser a que a Inteligencia artificial mais se adeque, então com isso foi escolhido o formato de Linguagem formal em forma de frases, pois com ela podemos definir melhor as condicionais do texto e deixar mais explicito quem são os atores como mostra no exemplo a seguir:

```
Cenário: Processo de Acionamento de
    Seguro e Reparação de Veículo

Descrição do Processo:

O segurado aciona o seguro.
A seguradora recolhe o veículo.
A oficina avalia os danos do veículo.
Se for constatada perda total, a
    seguradora deposita o valor segurado
    e o processo éfinalizado.
Caso contrário, a seguradora cobra a
    franquia.
O segurado paga a franquia.
A oficina conserta o veículo.
O processo éfinalizado após o conserto
    do veículo.
Atores e Ações: Segurado: Acionar
    Seguro, Pagar Franquia. Seguradora:
    Recolher Veículo, Depositar Valor
    Segurado, Cobrar Franquia. Oficina:
    Avaliar Danos, Consertar Veículo.
    Atividades Paralelas:

As atividades de "Cobrar Franquia",
    "Pagar Franquia" e "Consertar
    Veículo" são realizadas em paralelo.
```

Como pode-se ver desta forma fica bem definido cada bloco do diagrama, as condicionais e atores...

4. Desenvolvimento

Para o desenvolvimento do projeto, como citado anteriormente, foi utilizado a linguagem de programação Java com o framework spring boot e o maven para gerenciar as dependências do projeto. Algumas dependencias utilizadas para o desenvolvimento:

Thymeleaf: O thymeleaf foi utilizado dentro do projeto para o desenvolvimento das telas utilizando HTML, com ele é possivel desenvolver uma aplicação sem precisar separar a camada do frontend e backend.

mysql-connector: O mysql connector é utilizado para gerenciar o banco de dados da aplicação, com ele podemos lidar com o banco de dados direto no codigo, com anotações para que ele crie e gerencie ele automaticamente.

google-cloud-vertexai: O google cloud vertexai foi utilizado para fazer a comunicação com o gemini 1.5-pro, com ele temos mais facilidade para fazer a comunicação da API do mesmo pois não lidamos diretamente com ela, apenas é passado algumas informações.

plantuml: O plantuml foi utilizado para a comunicação com a API do PlantUML para a geração dos diagramas.

4.1. Páginas criadas

Três páginas foram criadas nesta aplicação: a página inicial, a página do formulário e a página de saída da imagem. Todas possuem um cabeçalho que facilita a navegação entre as páginas e para o site da Universidade Franciscana (UFN).

4.1.1. Página Home. A página Home contém uma explicação de como funciona o gerador de diagramas, como usa-lo e um exemplo de uso com a saida dele (Figura 4).

Figura 4 - Página home.

4.1.2. Página para digitar o texto. A página do formulário contém um prompt para digitar o texto e um botão para submeter o envio deste prompt (Figura 5).

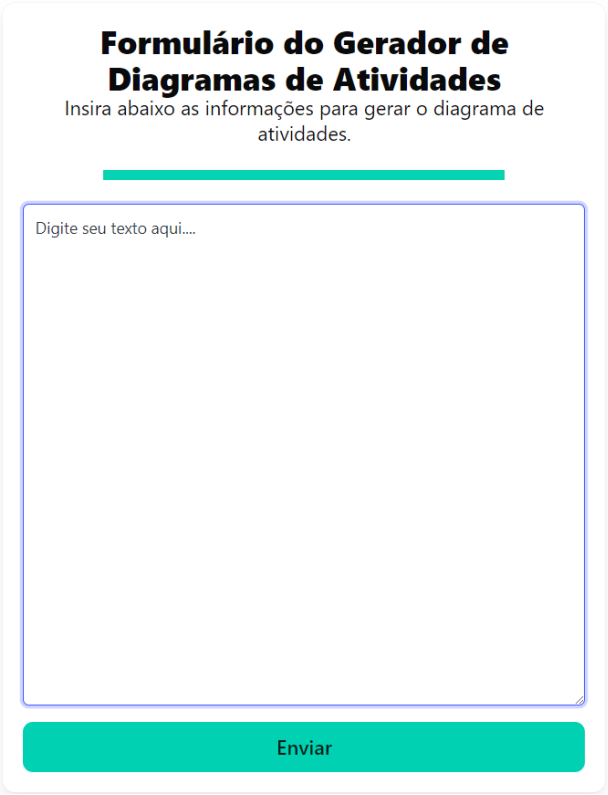

Figura 5 - Página de formulario.

4.1.3. Página onde o texto será gerado. A página onde a imagem será gerada, possui no corpo apenas um título e um texto que é sobreposto quando o diagrama é gerado (Figura 6).

Figura 6 - Página para saída da imagem.

5. Conclusão

Neste estudo, utilizamos o Gemini com o PlantUML para automatizar a geração de diagramas de atividade. Realizamos dez testes para avaliar a eficácia do modelo, dos quais seis apresentaram saídas satisfatórias. Observamos que, em alguns casos, as saídas geradas não corresponderam exatamente ao esperado, especialmente na identificação correta de partições e na precisão da sintaxe. No entanto, os resultados são promissores e indicam um potencial significativo para melhorias futuras. Com ajustes no modelo e na especificação dos prompts, esperamos aumentar a precisão e utilidade dos diagramas gerados. Em suma, este estudo demonstra a viabilidade do uso de IA para a criação automatizada de diagramas de atividade, abrindo caminho para novas inovações na engenharia de software.

5.1. Comparação dos resultados

Dos dez testes realizados para avaliar a eficácia do modelo na criação de diagramas seis resultaram em uma saída satisfatória, correspondendo a uma taxa de sucesso de 60%. Dentre os 40% dos testes que não obtivermos algo minimamente valido estavam: sintaxe inválida, ou seja o Gemini nos retornava um codigo com sinaxe incorreta e diagramas que não estavam corretos. Para ilustrar os resultados, comparamos o formato esperado de um diagrama de atividade (Figura 7) com um dos diagramas gerados pela aplicação (Figura 8). O diagrama gerado apresenta algumas discrepâncias em relação ao modelo ideal, especialmente na identificação correta de partições e na precisão da sintaxe. Observamos que o modelo teve dificuldades em inserir corretamente os nós de decisão e em nomear as ações de forma apropriada, resultando em redundâncias e inconsistências.

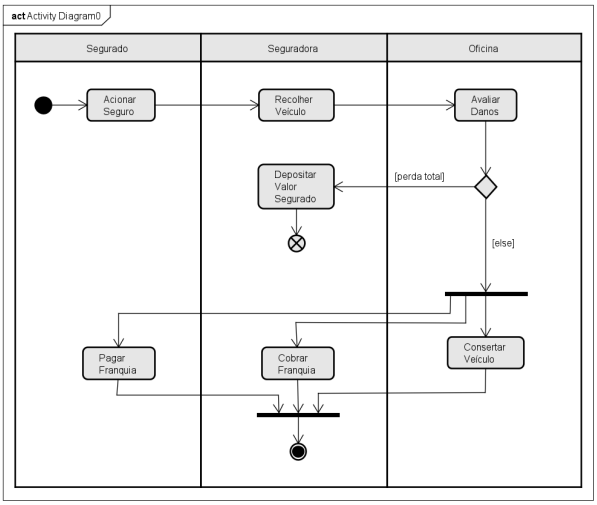

Figura 7 - Formato esperado do diagrama de atividade.

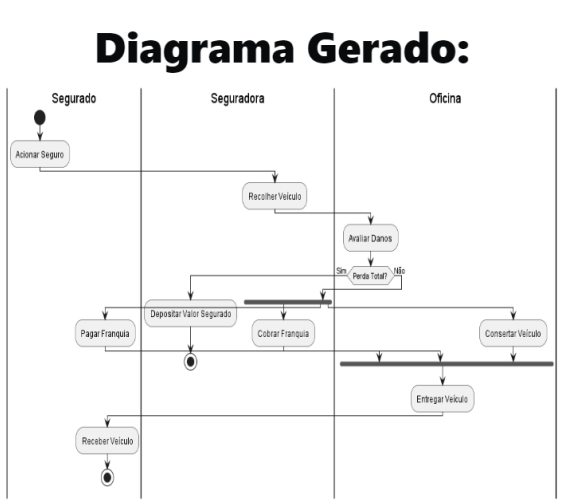

Figura 8 - Diagrama gerado pela aplicação.

Esta saida que nossa aplicação (Figura 7) nos retornou foi a melhor e a ”mais parecida” com a que estavamos esperando (Figura 8).

Apesar desses desafios, os resultados obtidos são encorajadores. A aplicação foi capaz de gerar diagramas utilizáveis em 60% dos casos, indicando que há um potencial significativo para aprimoramentos futuros. Melhorias na especificação dos prompts e ajustes no modelo podem aumentar a precisão e a utilidade dos diagramas gerados.

5.2. Perspectivas Futuras

O estudo demonstrou que é viável utilizar inteligência artificial para automatizar a criação de diagramas de atividade. No entanto, há necessidade de mais pesquisas para resolver os problemas de precisão e sintaxe identificados. Futuros desenvolvimentos podem focar em melhorar a compreensão do modelo em relação à linguagem natural, para isso poderiam ser testadas outras inteligências artificiais que tem esse mesmo proposito como exemplo do chatGPT ou até mesmo desenvolver um treinamento do gemini já que a própria google disponibiliza essa função em seu site de APIs.

Em suma, embora ainda haja desafios a serem superados, os avanços apresentados por esta pesquisa abrem caminho para inovações na engenharia de software, utilizando IA para facilitar e aprimorar o desenvolvimento de diagramas de atividade.

References

[1] F. Hans-Georg, F. Peter and K. Julius. Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. Acessado dia 19 de junho de 2023. Disponível em: https://folia.unifr.ch/global/documents/324646

[2] S. Abukhalaf, M. Hamdaqa and F. Khomh. On Codex Prompt Engineering for OCL Generation: An Empirical Study. Acessado dia 19 de junho de 2023. Disponível em: https://arxiv.org/abs/2303.16244

[3] Gilleanes T. A. Guedes, *UML 2 - Uma Abordagem Prática*, Acessado dia 04 de julho de 2024. Disponível em: https://www.academia.edu/6202541/UML_2_uma_abordagem_prtica

[4] IBM. Diagramas de Atividades. Acessado dia 19 de junho de 2023. Disponível em: https://www.ibm.com/docs/pt-br/rational-soft-arch/9.7.0?topic=diagrams-activity

[5] C. Fabro. O que é API e para que serve? Cinco perguntas e respostas. Acessado dia 19 de junho de 2023. Disponível em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml

[6] Perguntas... . O que é o Gemini? Acessado dia 02 de julho de 2024. Disponível em: https://gemini.google.com/faq?hl=pt-BR

[7] Open Risk Manual. PlantUML. Acessado dia 22 de junho de 2023. Disponível em: https://www.openriskmanual.org/wiki/PlantUML

[8] AWS. O que é uma API?. Acessado dia 26 de junho de 2023. Disponível em: https://aws.amazon.com/pt/what-is/api/

[9] A. Bessa. Java: O que é, linguagem e Guia para iniciar na tecnologia — Alura”. Acessado dia 02 de julho de 2024. Disponível em: https://www.alura.com.br/artigos/java

[10] S. Carruthers, A.Thomas, et all. Growing an Accessible and Inclusive Systems Design Course with PlantUML. Acessado dia 04 de julho de 2024 Disponivel em: https://dl.acm.org/doi/abs/10.1145/3545945.3569786

[11] Guia Sbok. Um Guia para o Conhecimento em Scrum. Acessado dia 04 de julho de 2024 Disponivel em: https://portaldaobmep.impa.br/uploads/material_teorico/d9gycv8klfcwc.pdf