Entrega II - Trabalho Final

Engenharia de Software

Felipe de Sousa Rodrigues (536984), Iara da Silva Lima (539429), Mariana Oliveira Felipe (539078), Mariana Pereira da Silva (542644), Yagho Miguel da Silva (539460)

Scribble

Sistema de Criação de Diagrama MoLIC

SUMÁRIO

1. Modelos de Arquitetura de Software..... 2
1.1 Diagrama de Classes..... 2
1.2 Diagramas de Atividade..... 2
Diagrama 1:..... 3
Diagrama 2:..... 4
Diagrama 3:..... 5
Diagrama 4:..... 6
Diagrama 5:..... 7
Diagrama 6:..... 8
Diagrama 7:..... 8
Diagrama 8:..... 9
2. Arquitetura do Sistema..... 9
2.1 Principais Componentes e Módulos..... 9
2.1.2 Base de Código e Interdependências..... 10
2.1.3 Componentes Principais..... 10
2.2 Desenvolvimento e Manutenção..... 10
2.3 Desafios e Soluções..... 10
2.3.1 Problemas encontrados durante o desenvolvimento..... 10
2.3.2 Soluções Implementadas..... 10
2.4 Resumo da Arquitetura e Suas Vantagens..... 11

1. Modelos de Arquitetura de Software

1.1 Diagrama de Classes

A seguir, link de acesso ao diagrama de classe do sistema Scribble: https://drive.google.com/file/d/1vi_Zb18OYTZdRvMjaMyeBcm9u75QYmi-/view?usp=sharing (Por questão de visualização, optou-se pelo envio do link de acesso ao documento do diagrama, uma vez que este não seria de fácil visualização no documento atual).
A classe principal **BasicGraphEditor**, que é responsável por gerenciar o editor gráfico. Ela inclui diversas funcionalidades, como a manipulação de desfazer/refazer ações através de um undoManager e a exibição de menus de contexto (popups). Dentro dessa classe, há várias classes internas que desempenham papéis específicos, como responder a eventos e ajustar a interface conforme o usuário interage com o editor. Mostra-se como as várias classes internas e ações definidas no **BasicGraphEditor** interagem para fornecer funcionalidades complexas de edição gráfica, manipulando eventos de interface e ajustes visuais de maneira detalhada e interativa.

BasicGraphEditor, possui as seguintes classes internas:

- **BasicGraphEditor$1**: classe interna que mantém uma referência ao BasicGraphEditor e responde a eventos através do método invoke.
- **BasicGraphEditor$2, $3, $4**: estendem ComponentAdapter e ajustam a interface quando o editor gráfico é redimensionado. Por exemplo, ajustam a localização do divisor em um JSplitPane ou a largura preferida de componentes como EditorPalette com base no tamanho do JScrollPane.
- **BasicGraphEditor$5 e $6**: estendem MouseAdapter para gerenciar cliques do mouse e exibir menus de contexto (popups). O $5 exibe um popup genérico e o $6 exibe um popup específico para o gráfico.
- **BasicGraphEditor$7**: implementa MouseMotionListener para lidar com eventos de movimento do mouse, como atualizar a localização do cursor.
- **BasicGraphEditor$8**: estende AbstractAction e modifica o comportamento das ações disparadas dentro do BasicGraphEditor.

Demais classes:

- **DefaultFileFilter**: estende FileFilter e inclui métodos para aceitar arquivos com extensões específicas. Ela também tem subclasses, como:
    - **DefaultFileFilter$EditorFileFilter**: aceita arquivos XML e XML comprimidos, além de diretórios.
    - **DefaultFileFilter$ImageFileFilter**: filtra arquivos baseados nos formatos de imagem suportados pelo ImageIO.

Ações da classe **EditorActions:**

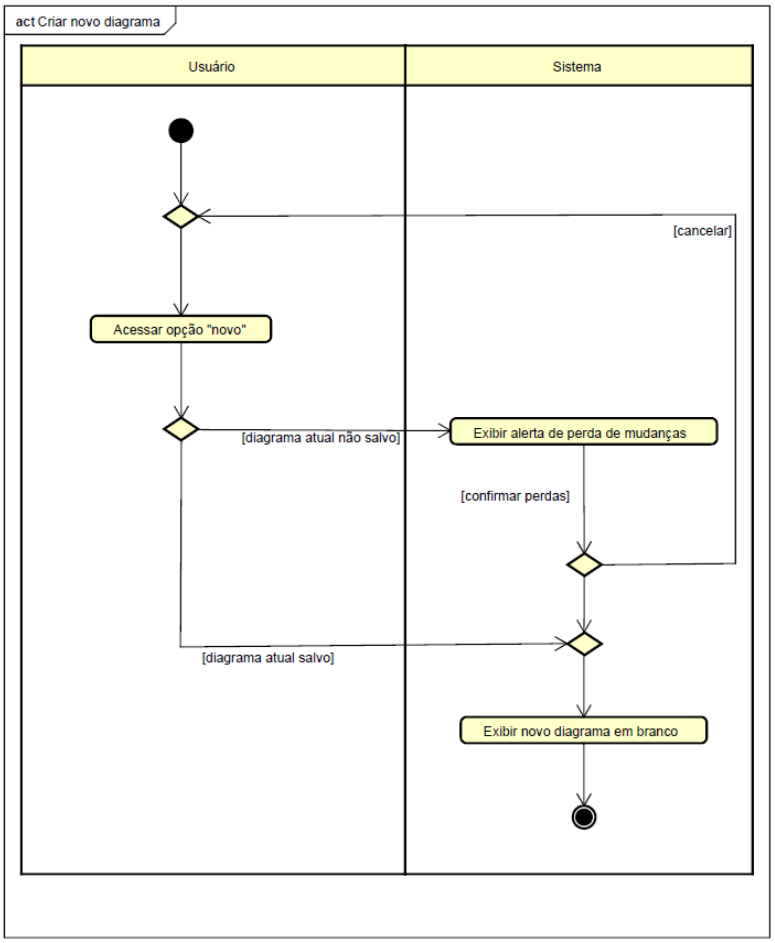
Várias ações relacionadas ao gerenciamento gráfico são descritas, estendendo **AbstractAction:**

- **AlignCellsAction:** alinha células no gráfico do framework JGraphX com base em eventos de ação.
- **AutosizeAction:** ajusta automaticamente o tamanho das células no gráfico quando disparado por um evento.

- **BackgroundAction:** atualiza a cor de fundo do componente gráfico.
- **BackgroundImageAction:** exibe um diálogo para o usuário inserir uma URL de imagem de fundo, e, se a URL estiver vazia, remove a imagem de fundo.
- **ColorAction:** atualiza o estilo das células selecionadas no gráfico com base na cor escolhida.
- **ExitAction:** encerra a aplicação ou finaliza a edição, obtendo o editor através do método getEditor() e chamando exit().
- **FontStyleAction:** modifica o estilo de fonte (negrito ou itálico) das células no gráfico.
- **GridColorAction:** atualiza a cor da grade no componente gráfico e repinta para aplicar as mudanças.
- **GridStyleAction:** altera o estilo da grade exibida no componente gráfico e repinta a interface.

1.2 Diagramas de Atividade

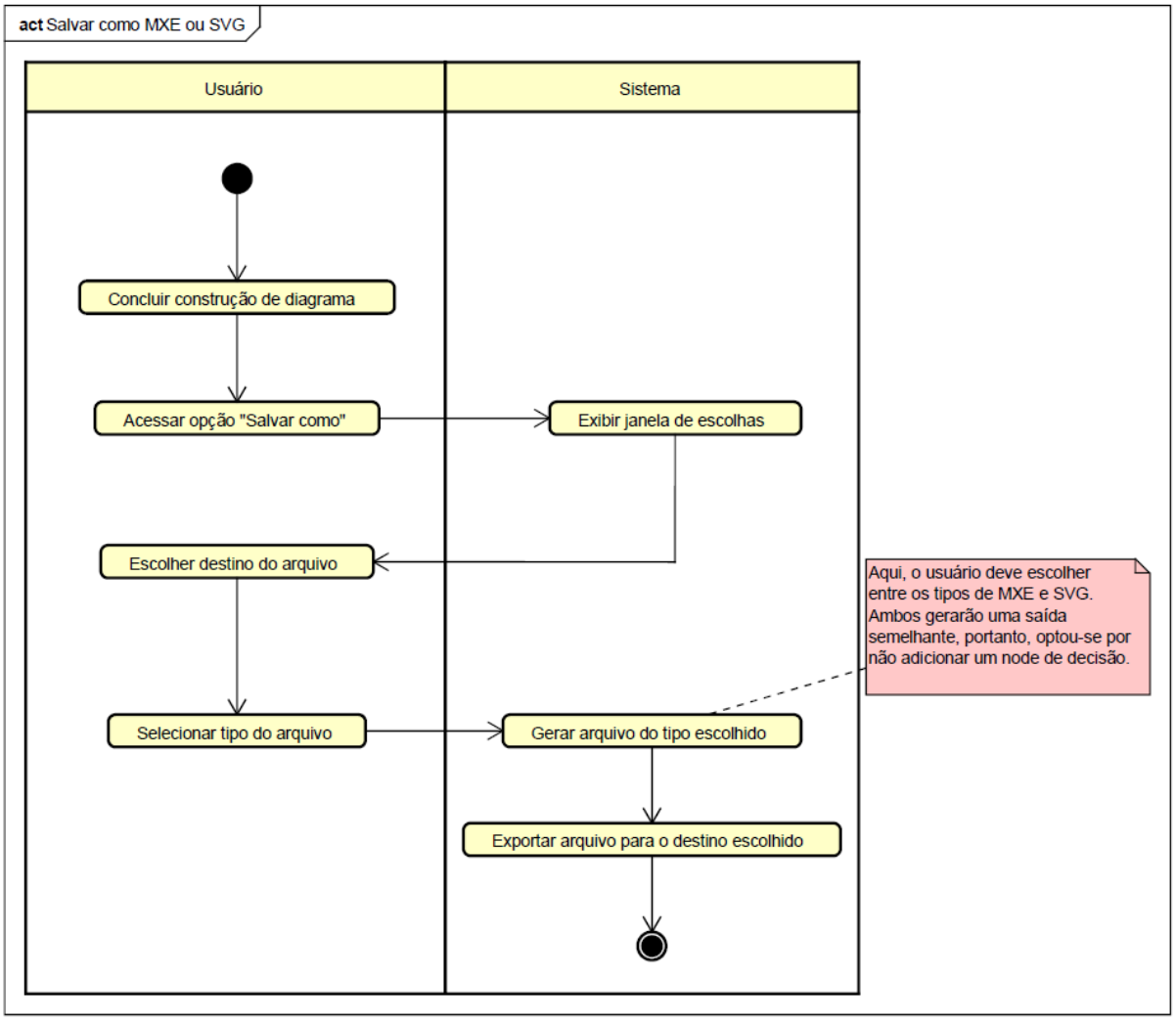
Diagrama 1:



Este diagrama representa o fluxo de ações envolvidas no processo de criação de um diagrama. Aqui, se parte do pressuposto de que o usuário deseja criar um diagrama diferente do diagrama em branco que já é aberto por padrão ao iniciar o Scribble. Na raia do Usuário inicia-se o processo na atividade “Acessar opção “novo””, a partir dessa ação, há uma bifurcação nas atividades representado por um nó de decisão: Se o usuário já dispôs elementos na tela do diagrama atual, ele deverá escolher entre cancelar a operação para que o diagrama seja salvo ou iniciar a criação de um novo. Em resumo, o usuário tem a flexibilidade de criar um novo diagrama com a segurança de que seu diagrama atual não será

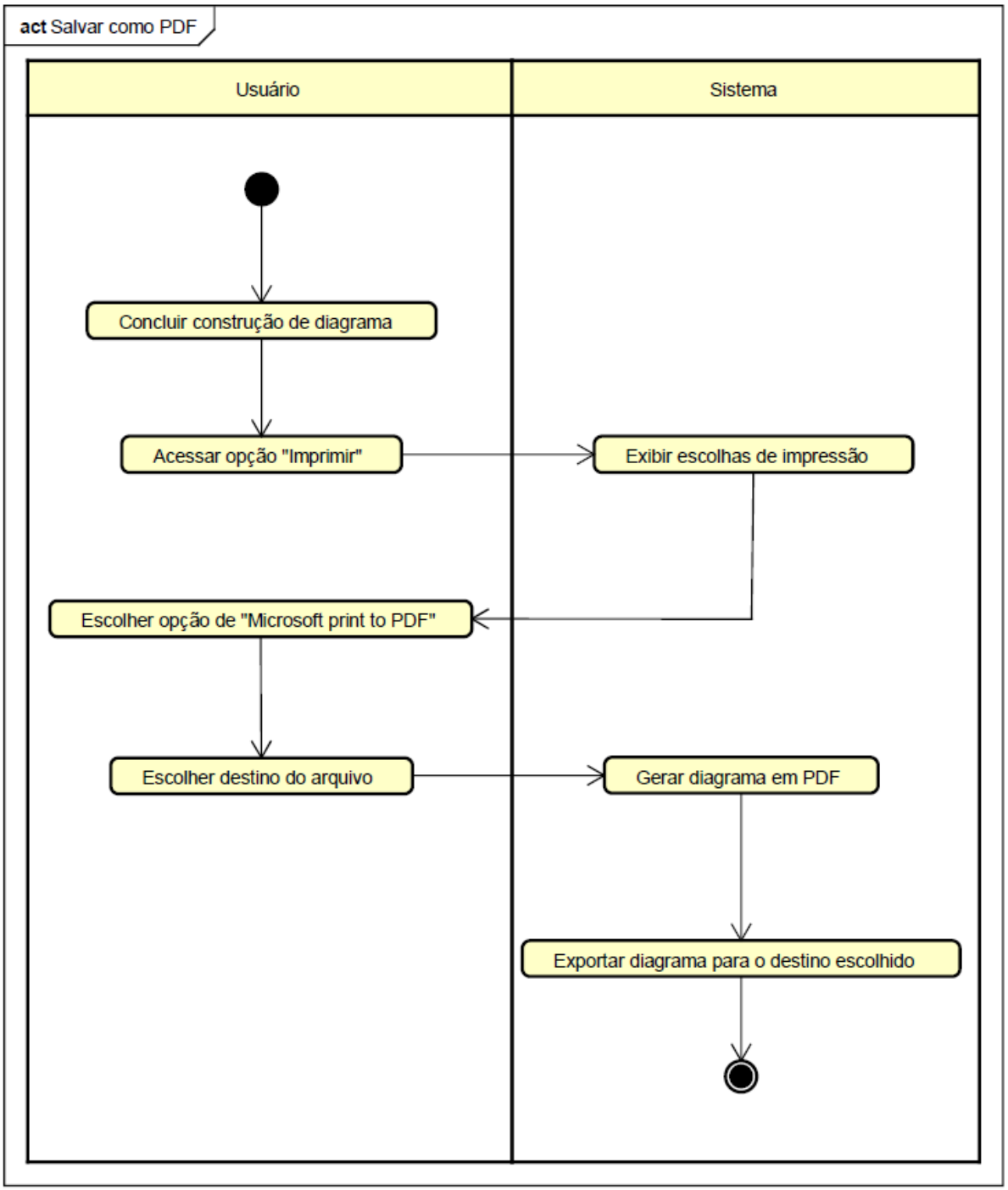
perdido. O sistema responde conforme a escolha do usuário, facilitando o processo de criação e posteriormente edição de diagramas.

Diagrama 2:



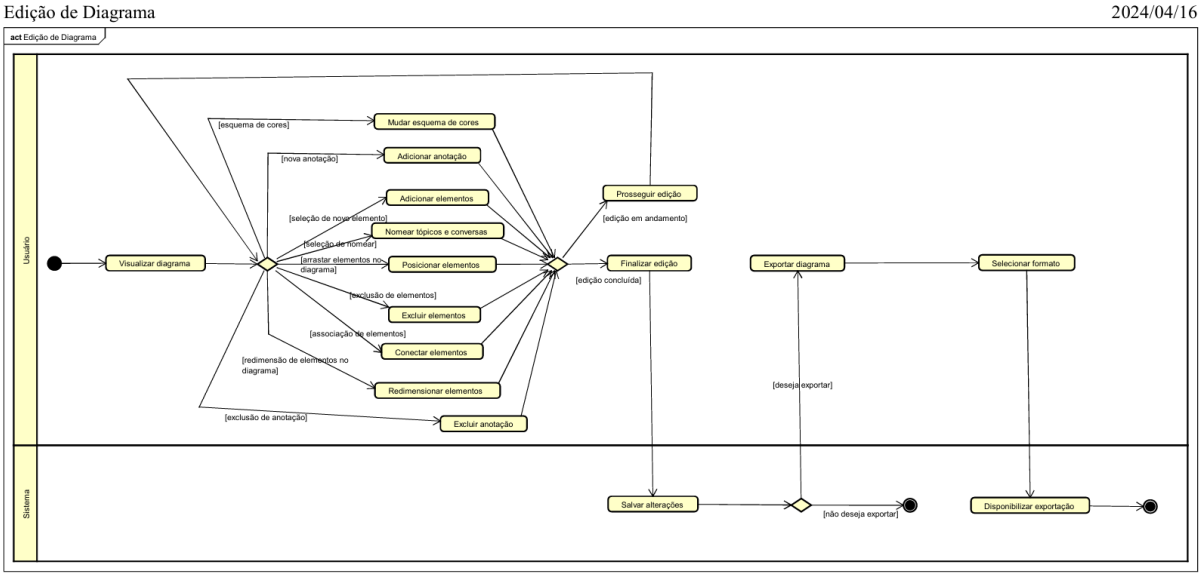
O diagrama acima descreve a atividade de salvar o diagrama MoLIC nos formatos MXE ou SVG. O usuário, ao concluir a construção de seu diagrama, deve acessar a opção de “Salvar como” para que posteriormente escolha tanto o destino do arquivo quanto o seu tipo. Nesta janela, ele irá dispor das opções de tipo MXE (MxGraph Editor) ou SVG para que o diagrama possa ser acessado localmente através de um navegador web. Após este passo, o sistema irá gerar o arquivo no formato selecionado e o exportará para o destino desejado.

Diagrama 3:



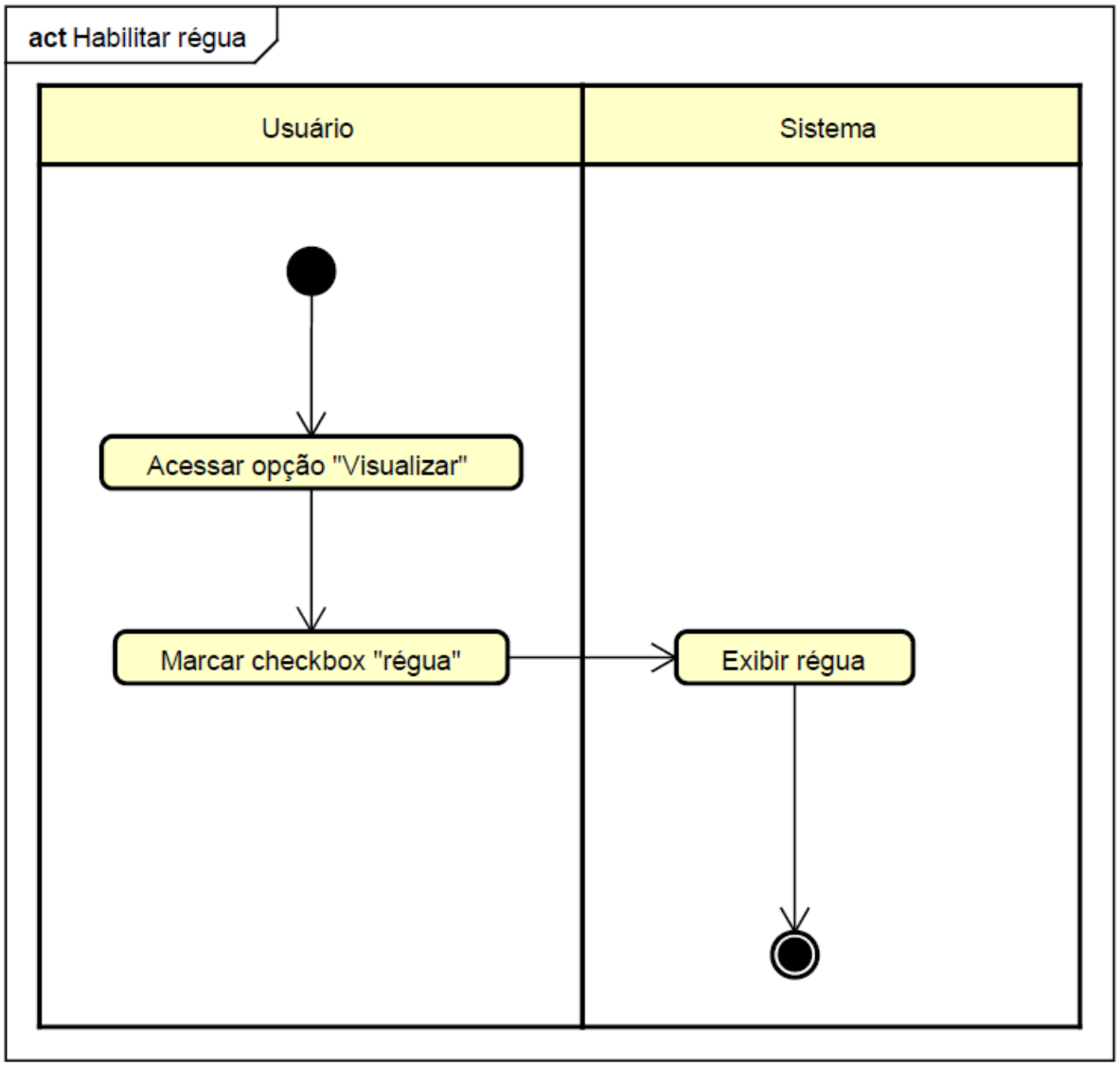
Esta atividade se assemelha à anterior. No entanto, para que o diagrama construído seja exportado no formato de PDF, o usuário deve acessar a opção de “imprimir”, onde, entre as opções de impressão, será disposta a possibilidade de exportação em PDF. Após selecionar “Microsoft print to PDF”, o usuário deverá escolher o destino do arquivo para que, por fim, o sistema gere o diagrama MoLIC em PDF e o exporte para o destino selecionado.

Diagrama 4:



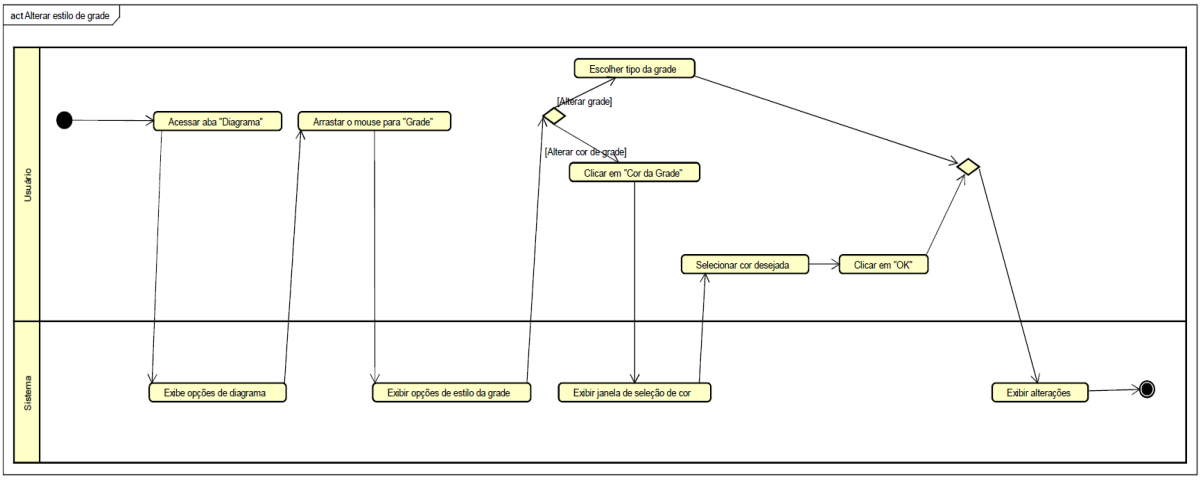
No diagrama de Edição, o Usuário inicia com a ação de “Visualizar diagrama”, onde um nó de decisão o leva a todas as opções disponíveis para edição daquele diagrama, sendo elas as atividades que representam as ações de “Mudar esquema de cores”, “Adicionar anotação”, “Adicionar elementos”, “Nomear tópicos e conversas”, “Posicionar elementos”, “Excluir elementos”, “Conectar elementos”, “Redimensionar elementos” e “Excluir anotações”. Ao fechamento do nó de decisão, têm-se duas saídas: se a edição estiver em andamento, a atividade “Prosseguir edição” volta para o nó de decisão inicial, se edição concluída: “Finalizar edição”, passando o controle então para o Sistema com a atividade “Salvar alterações”, levando novamente a um nó de decisão de duas possíveis saídas, se não há o desejo de exportar o diagrama, finalizado o processo; se há o desejo de exportar, segue-se a atividade “Exportar diagrama”, seguido da atividade de “Selecionar formato” e fica a cargo do Sistema a ação de “Disponibilizar exportação” e assim conclui-se o processo.

Diagrama 5:



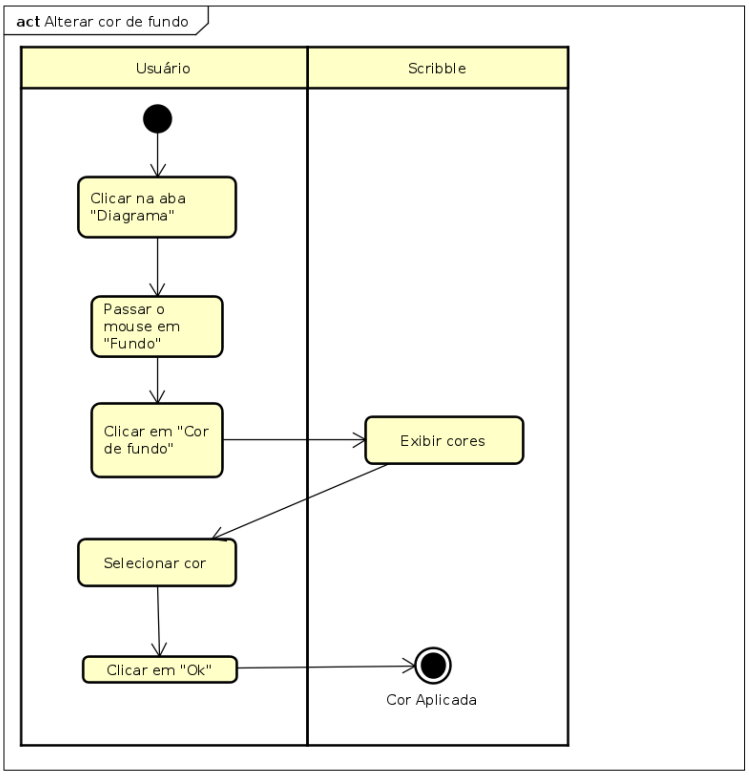
No intuito de obter uma visualização de métricas, o usuário pode desejar utilizar a régua nativa do Scribble. Para isso, na barra superior de ferramentas, deve acessar a opção “Visualizar”, onde poderá habilitar a régua, que será exibida na vertical esquerda e na horizontal superior em forma de plano cartesiano.

Diagrama 6:



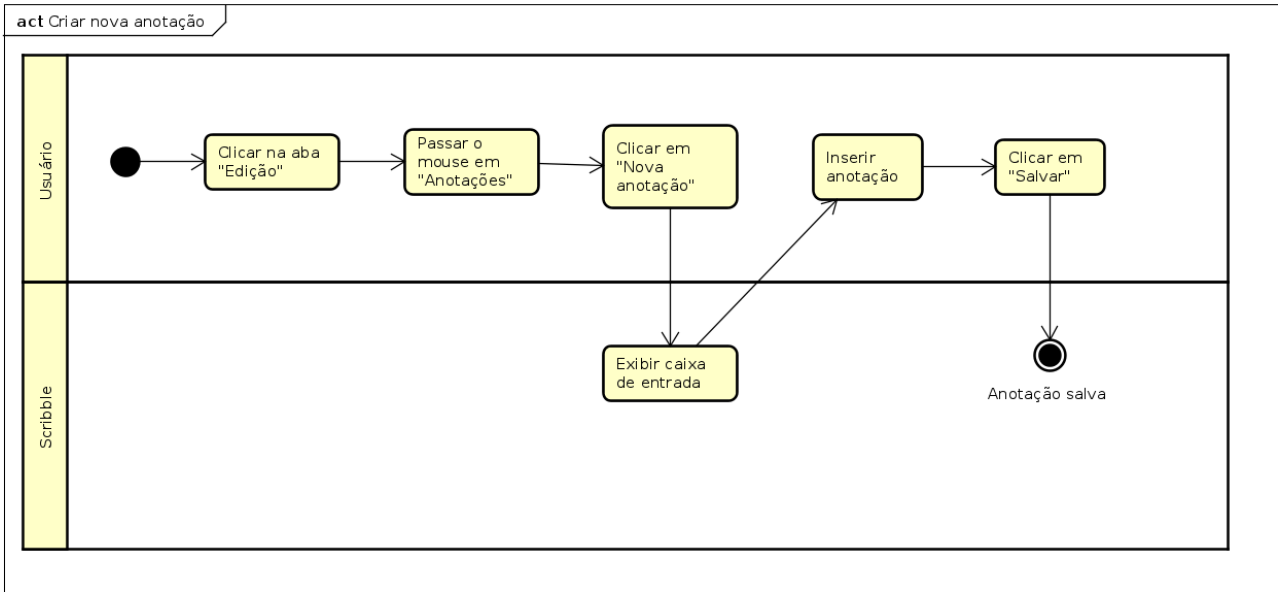
O diagrama acima descreve a ação de alteração de estilo de grade. A princípio, o usuário deve acessar a opção “Diagramas” presente na barra superior de ferramentas, arrastar o mouse para a opção “Grade” para serem exibidas as opções de estilo. Há um node de decisão que representa a escolha que o usuário deverá realizar: alterar o tipo da grade ou alterar a cor dela. A escolha de tipo o leva a uma conclusão automática da atividade, pois o estilo escolhido será aplicado uma vez que o usuário clicar nele. Se ele desejar alterar a cor, o Scribble exibirá uma janela com diversas opções de cores para que o usuário selecione a cor que deseja usar. Depois disso, ele deve clicar em “Ok” para que o sistema exiba as alterações e conclua a atividade.

Diagrama 7:



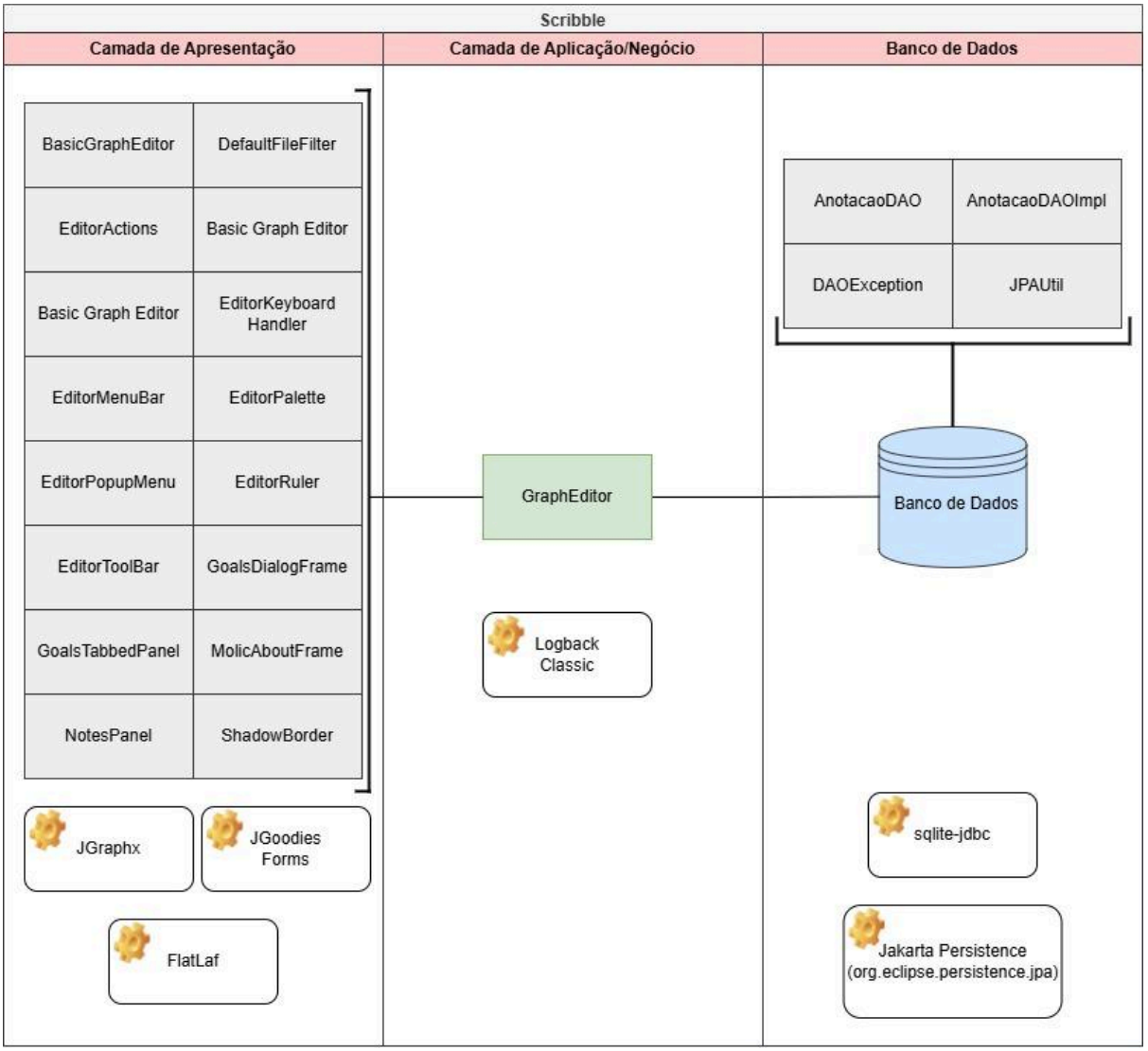
Neste diagrama, o usuário tem o controle de “Alterar cor de fundo”, a princípio, o usuário poderá mudar a cor do fundo do seu diagrama. Onde, na aba “Diagrama”, o usuário poderá escolher a cor que deseja e o Scribble irá aplicar.

Diagrama 8:



No Diagrama de Atividades “Criar nova anotação”, o usuário inicia esse processo na aba “Edição”, onde poderá adicionar novas anotações, inserindo-as em uma caixa de entrada, logo após, o Scribble salva suas anotações.

2. Arquitetura do Sistema



O Scribble é uma aplicação desktop desenvolvida em Java, baseada em uma arquitetura em camadas. Essa abordagem organiza o sistema em módulos separados, onde cada camada tem uma responsabilidade específica, promovendo a independência entre componentes, maior manutenibilidade e escalabilidade do sistema. A aplicação se comunica com o banco de dados SQLite e a persistência de dados é gerenciada pelo Jakarta Persistence (JPA).

2.1 Principais Componentes e Módulos

O sistema é composto por vários componentes e módulos principais que interagem para fornecer as funcionalidades necessárias para a criação de diagramas. Estes componentes incluem bibliotecas específicas para diagramação, interface gráfica e aparência da aplicação, registros de log e persistência de dados.

2.1.2 Base de Código e Interdependências

A aplicação utiliza uma base de código dividida em módulos independentes. Este modelo em camadas facilita a manutenção e permite a alteração em uma camada sem que a outra seja afetada.

2.1.3 Camadas

- **Camada de Apresentação:**

Responsável pela interação com o usuário. Esta camada utiliza Java AWT (Abstract Window Toolkit), FlatLaf, JGoodies Forms e JGraphx.

- **Camada de Aplicação/Negócio:**

Ela é responsável por coordenar as operações entre a camada de apresentação e o banco de dados, garantindo que a lógica do sistema permaneça independente dos detalhes da interface do usuário e da persistência dos dados. A biblioteca mxGraph é usada aqui para fornecer funcionalidades de manipulação de diagramas.

- **Camada de Banco de Dados:**

Essa camada gerencia a persistência dos dados do sistema. O Scribble utiliza SQLite como banco de dados, e a comunicação com ele é feita por meio do SQLite-JDBC. O Jakarta Persistence (JPA) é empregado para mapear as classes Java às tabelas do banco de dados.

2.2 Desenvolvimento e Manutenção

O desenvolvimento segue um processo iterativo e incremental, com ciclos de planejamento, implementação, teste e revisão. Ferramentas de controle de versão como Git são utilizadas para gerenciar o código-fonte. Manutenção contínua é realizada para corrigir bugs, implementar novas funcionalidades e melhorar a performance. Documentação detalhada e testes automatizados são mantidos para garantir a qualidade do código.

2.3 Desafios e Soluções

2.3.1 Problemas encontrados durante o desenvolvimento

- Integração de diferentes bibliotecas e componentes.
- Garantia de performance adequada para a manipulação de diagramas complexos.
- Manutenção da consistência visual e funcional da interface.

2.3.2 Soluções Implementadas

- Adaptação de bibliotecas para melhor integração.
- Otimização de algoritmos e estruturas de dados.

- Utilização de FlatLaf para estilização consistente.