

PUC

INF2102 - PROJETO FINAL DE PROGRAMACAO - 2022.1 - 3WA

Simulador de diagrama MoLIC (Documentação)

Pedro Henrique Bof Gericó

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

RF5	O usuário deve ser capaz de visualizar cada um dos caminhos resultantes entre os elementos escolhidos pelo usuário
RF6	O sistema deve ser capaz de exibir um sub-diagrama da rota resultante
RF7	O sistema deve numerar cada elemento do sub-diagrama com seu identificador
RF8	O sistema deve demarcar os elementos que foram revisitados
RF9	O usuário deve ser capaz de regular o tamanho da tela que exibe os sub-diagramas

2.3. Requisitos Não Funcionais

ID	TIPO	DESCRIÇÃO
RNF1	Organizacional (Implementação)	O sistema deve ser implementado em C# e .NET
RNF2	Produto (Usabilidade)	Os usuários do sistema devem poder aproveitar os recursos sem treinamento
RNF3	Produto (Desempenho)	O sistema deve carregar o XML em menos de 2 segundos
RNF4	Produto (Desempenho)	O sistema deve ser capaz de obter caminhos possíveis em menos de 2 segundos
RNF5	Produto (Portabilidade)	O sistema deve ser responsivo para renderização em dispositivos móveis

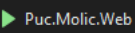
3. Projeto do Programa

3.1. Instruções para execução

Requisitos:

- Visual Studio 2022 Community
(<https://visualstudio.microsoft.com/vs/community/>)
- Windows 10 versão 1909 ou maior: Home, Professional, Education ou Enterprise.

Instruções:

1. Clonar repositório Github ou efetuar download do projeto no site GitHub
(<https://github.com/pedrohbg/puc-molic-simulator.git>)
2. Instalar Visual Studio 2022 Community
3. Abrir Visual Studio 2022 Community em modo Administrador
4. Dentro do Visual Studio 2022, abrir projeto (File > Open > Project/Solution...)
5. Selecionar dentro da raiz do diretório clonado do GitHub o arquivo "Puc.Molic.sln"
6. Com o projeto aberto, clicar em  ou pressionar F5.

3.2. Arquitetura

O sistema usa a tecnologia Microsoft .NET 6.0, lançada em novembro de 2021, que é gratuita e de código aberto para os sistemas operacionais Windows, Linux e macOS. O framework foi desenvolvido pela Microsoft por meio da .NET Foundation e lançada sob a licença do MIT. O .NET 6 oferece suporte ao Visual Studio 2022 e ao Visual Studio 2022 para Mac.

A linguagem de programação de backend utilizada no desenvolvimento do sistema é C#, que é uma linguagem multiparadigma de propósito geral. C# abrange tipagem estática, tipagem forte, escopo léxico, imperativo, declarativo, funcional, genérico, orientado a objetos (baseado em classe) e programação orientada a componentes. Para o frontend foram utilizados HTML e CSS e Javascript com os frameworks: bootstrap, JQuery, JQuerySlider. Para testes unitários, o framework escolhido foi o xUnit.Net, uma ferramenta de teste unitário gratuita e de código aberto para .NET.

O sistema é dividido em 3 projetos ou camadas: Puc.Molic.Web, Puc.Molic e Puc.Molic.Model. A camada Puc.Molic.Web é responsável pela apresentação e pelos controladores no padrão MVC (Model-View-Controller), que permite a comunicação da visão com os controladores. No caso deste projeto, a visão principal está no arquivo “Molic.cshtml”, e seu controlador é o MolicController, que, por sua vez, faz as requisições em conjunto com o DiagramService para a camada principal que concentra as regras de negócio (Puc.Molic).

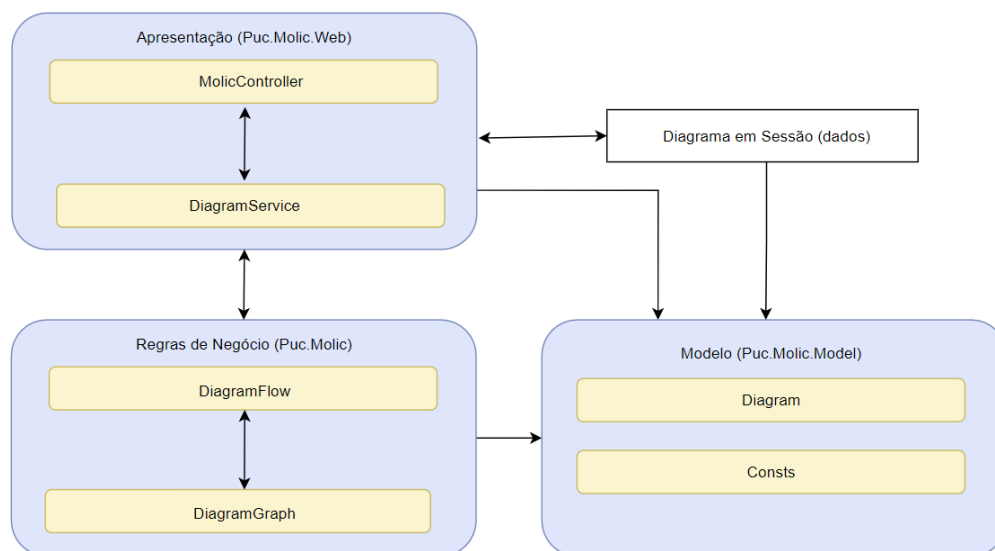


Figura 1 - Arquitetura da Aplicação

A camada Puc.Molic possui um módulo de algoritmo de grafos (**DiagramGraph**), que é operado com uma instância do objeto “**Diagram**” estabelecendo que cada signo e seta sejam mapeados como vértices e arestas de um grafo. Ao transformarmos o diagrama em grafo, um algoritmo de busca de caminhos fica pronto para obtenção de todos os possíveis rotas entre os pontos inicial e final, mesmo que passem por pontos repetidos.

Como fonte de dados, o usuário carrega um diagrama XML que é transformado em objeto de Diagrama que fica armazenado em sessão. A sessão com o Diagrama serve como banco de dados MoLIC a ser operado pelo usuário. Esta sessão é sempre consultada durante a execução do sistema.

```

4 <Nodes>
5 <Node Id="1" Type="Opening">
6 <Geometry>
7 <PositionX>216</PositionX>
8 <PositionY>122</PositionY>
9 </Geometry>
10 <Dialogs/>
11 <Precond/>
12 </Node>
13 <Node Id="2" Type="Scene" Topic="Identificar conta bancária">
14 <Geometry>
15 <PositionX>345</PositionX>
16 <PositionY>532</PositionY>
17 </Geometry>
18 <Dialogs>
19 <Dialog Subject="informar conta">
20 <Signs>
21 <Sign Type="d+u">agência</Sign>
22 <Sign Type="d+u">conta</Sign>
23 </Signs>
24 </Dialog>
25 <Dialog Subject="informar senha">
26 <Signs>
27 <Sign Type="d+u">senha</Sign>
28 </Signs>
29 </Dialog>
30 </Dialogs>
31 <Precond/>
32 </Node>

```

Figura 2 – Parte do conteúdo de um XML MoLIC

A camada Puc.Molic.Model possui o modelo MoLIC definido pela classe “Diagram” com subclasses e strings constantes que definem cada tipo de elemento MoLIC existente.

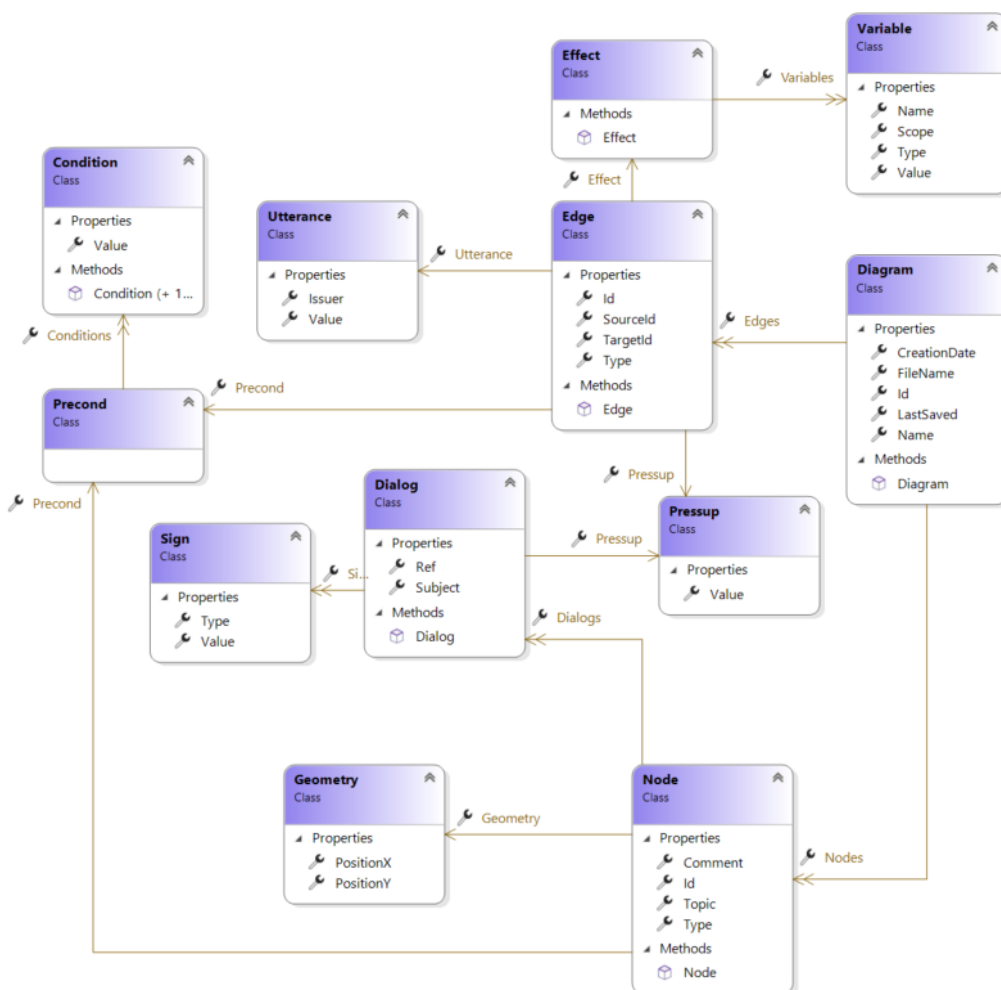


Figura 3 - Diagrama de Classes MoLIC (Classe “Diagram”)

4. Roteiro de Teste

4.1. Testes unitários

O projeto possui testes unitários em xUnit.Net que cobrem os dois principais módulos do backend, o módulo de conversação e o módulo de obtenção de caminhos

Test	Duration	Traits	Error Message
✓ Puc.Molic.Test (6)	102 ms		
✓ Puc.Molic.Test (6)	102 ms		
✓ ConversionTest (2)	95 ms		
✓ TestDiagramToXML	12 ms		
✓ TestXMLToDiagram	83 ms		
✓ TestGraph (4)	7 ms		
✓ TestPathAlgorithm1	7 ms		
✓ TestPathAlgorithm2	< 1 ms		
✓ TestPathAlgorithm3	< 1 ms		
✓ TestPathAlgorithm4	< 1 ms		

Group Summary
Puc.Molic.Test
Tests in group: 6
⌚ Total Duration: 102 ms
Outcomes
✓ 6 Passed

Figura 4 - Resultado dos testes unitários

4.2. Testes dos requisitos funcionais

- **RF1: O usuário deve ser capaz de fazer upload do diagrama MoLiC no formato XML (Passa)**



Figura 5 - Ao clicar em “Carregar diagrama XML”, exibe pop-up para upload do arquivo XML

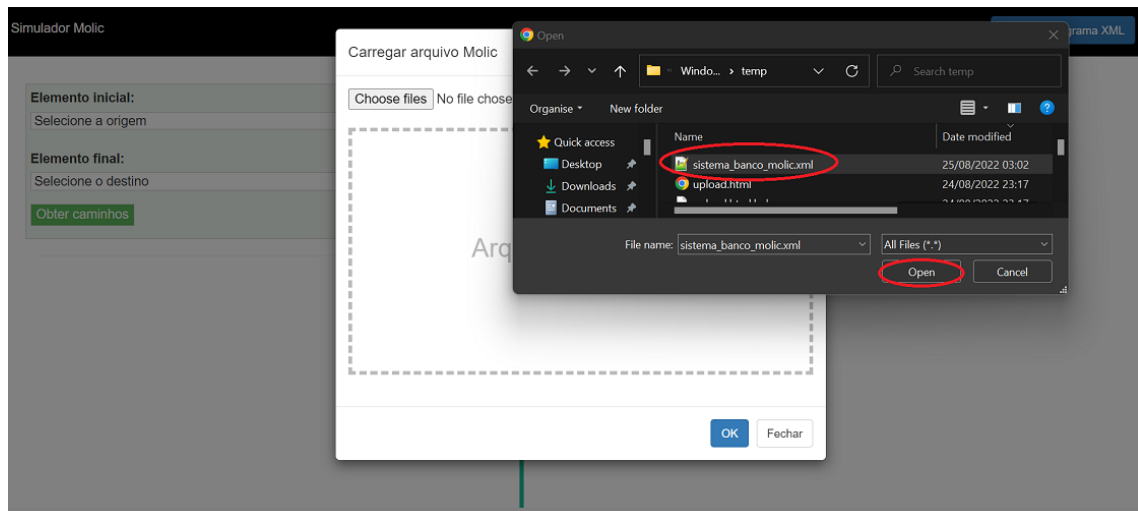


Figura 6 - Seleção de XML MoLIC



Figura 7 – Ao clicar em OK, o arquivo XML é transformado em objeto “Diagram” e armazenado em sessão

- **RF2: O usuário deve ser capaz de selecionar o elemento inicial desejado no diagrama MoLIC (Passa)**

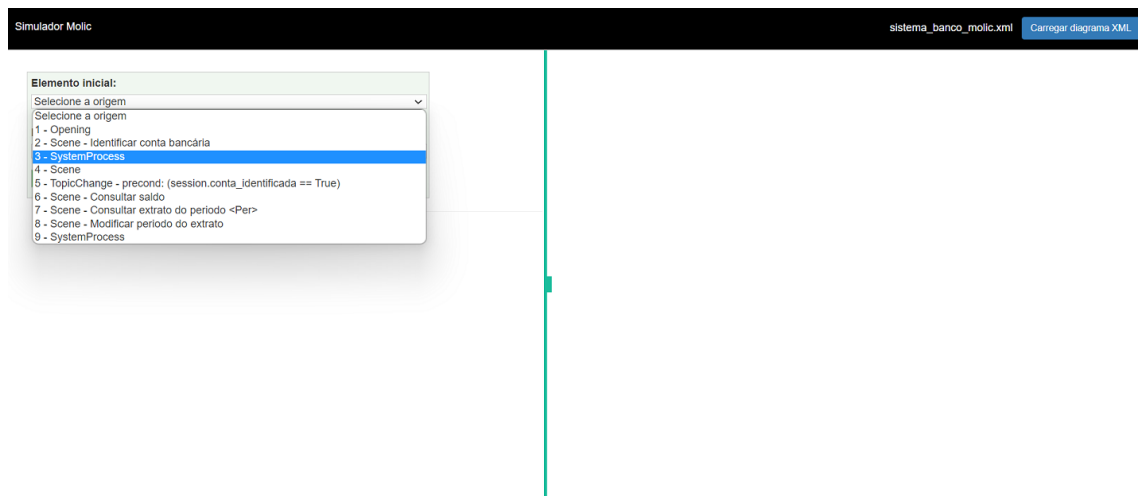


Figura 8 – Seleção do elemento inicial

- **RF3: O usuário deve ser capaz de selecionar o elemento inicial desejado no diagrama MoLIC (Passa)**

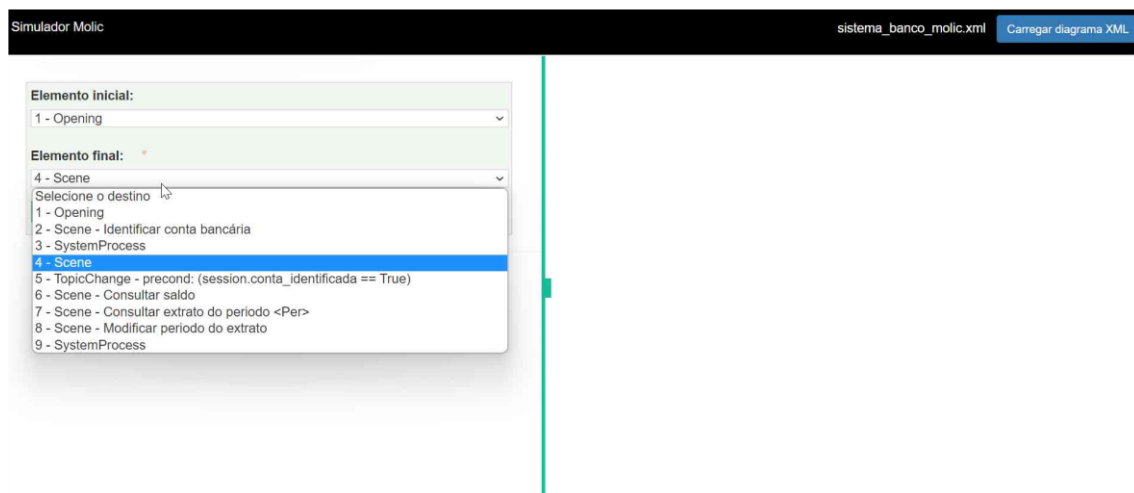


Figura 9 – Seleção do elemento final

- **RF5: O usuário deve ser capaz de visualizar cada um dos caminhos resultantes entre os elementos escolhidos pelo usuário (Passa)**

Elemento inicial:

1 - Opening

Elemento final:

4 - Scene

Obter caminhos

Caminho #1 (1,2,3,4) ←

Caminho #2 (1,2,3,2,3,4) ←

Figura 10 – Caminhos possíveis apresentados ao usuário

- **RF6: O sistema deve ser capaz de exibir um sub-diagrama da rota resultante (Passa)**

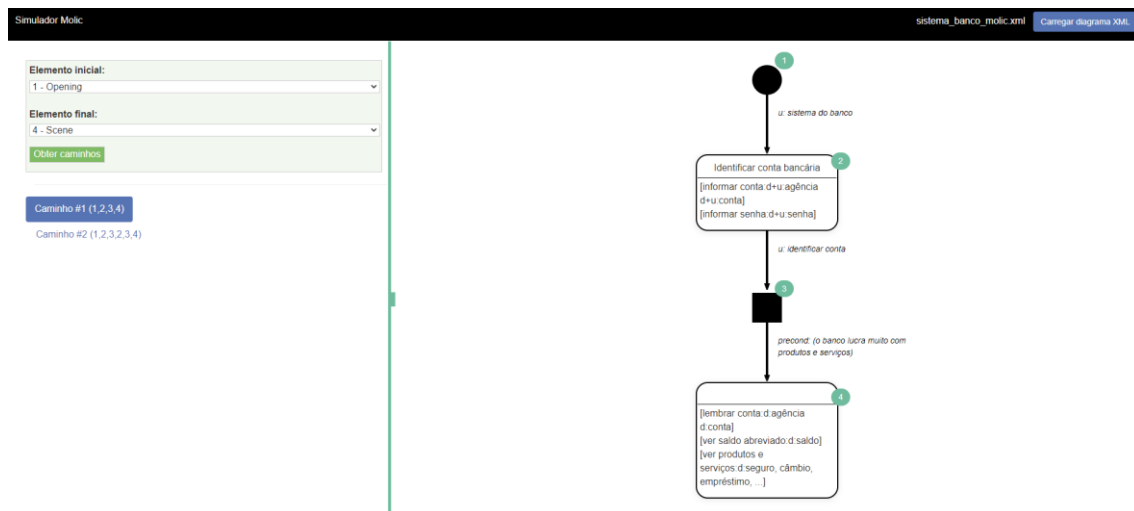


Figura 11 – Sub-diagrama exibido de uma das rotas possíveis

- **RF7: O sistema deve numerar cada elemento do sub-diagrama com seu identificador (Passa)**

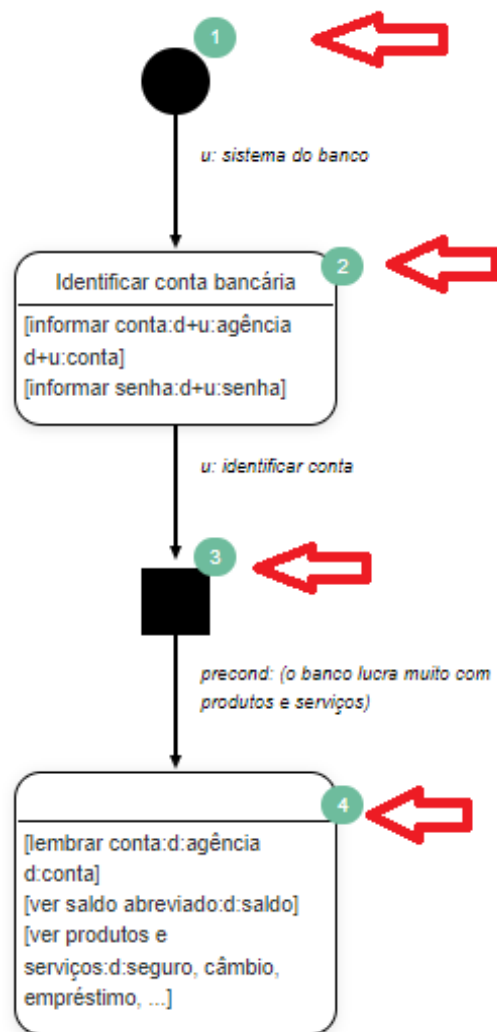


Figura 12 – Elementos MoLIC com identificação

- RF8: O sistema deve demarcar os elementos que foram revisitados (**Passa**)

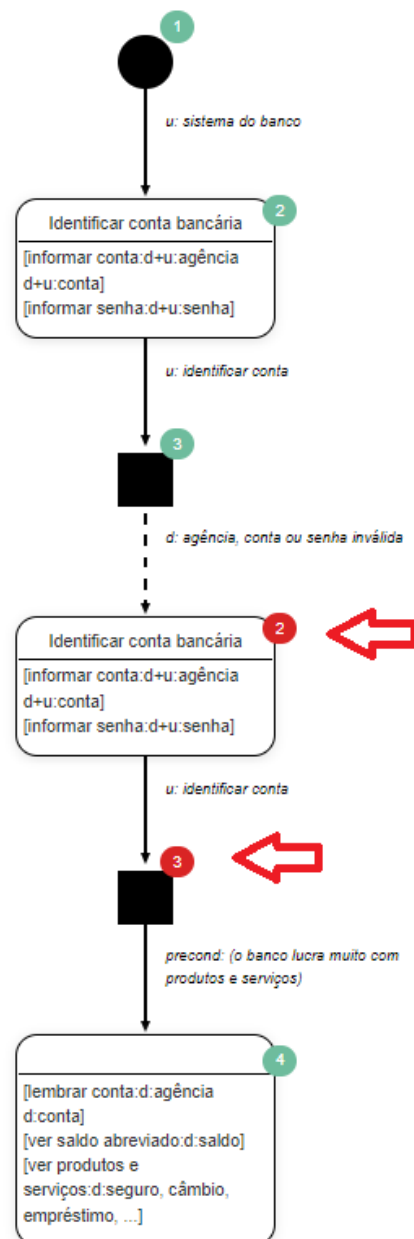


Figura 13 – Elementos repetidos demarcados em vermelho

- **RF9: O usuário deve ser capaz de regular o tamanho da tela que exibe os sub-diagramas (Passa)**

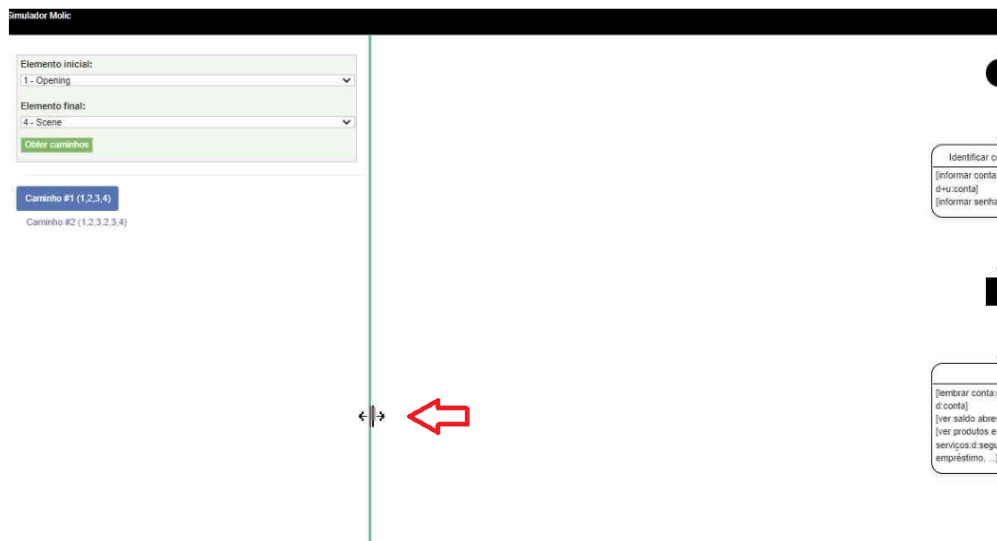
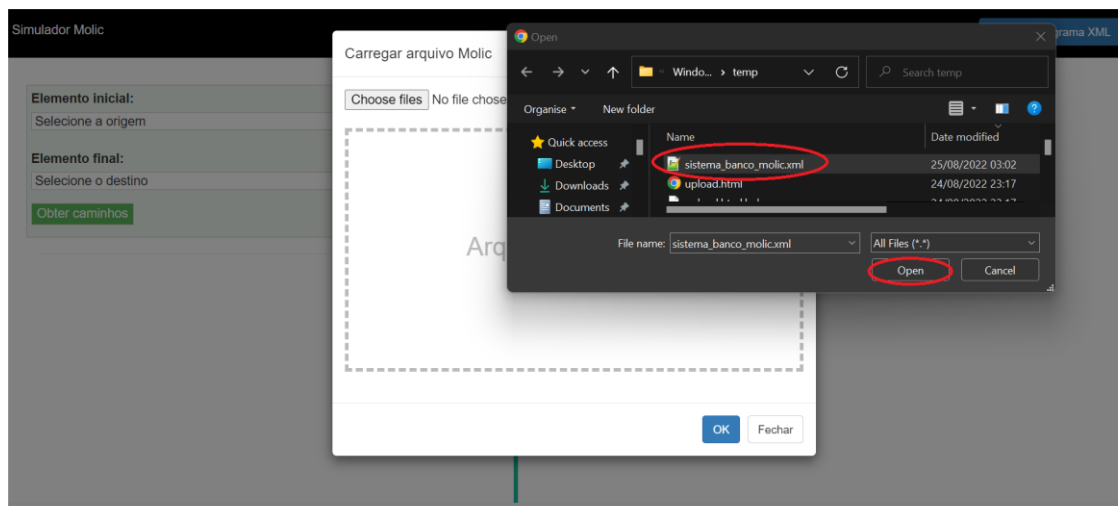
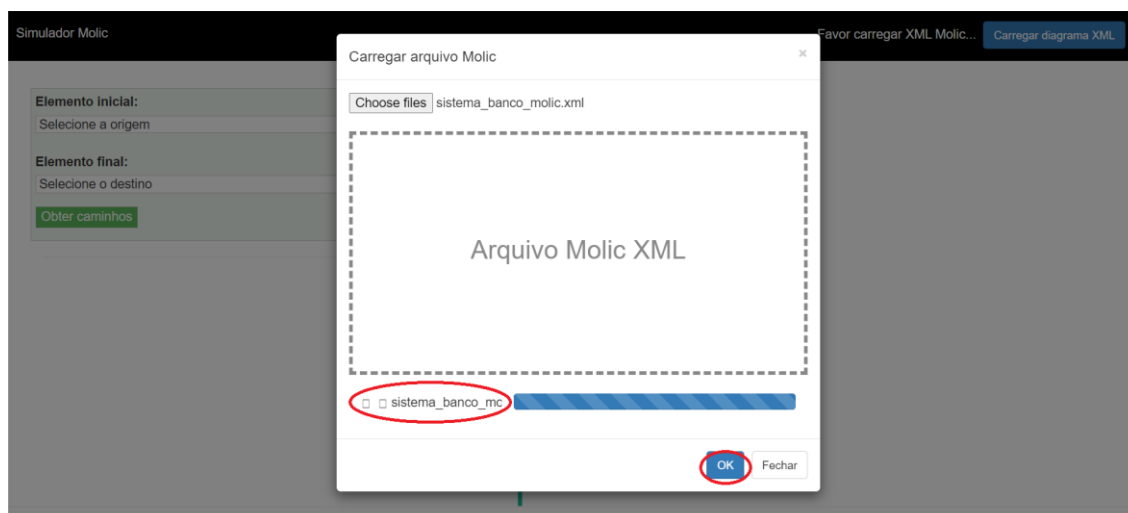


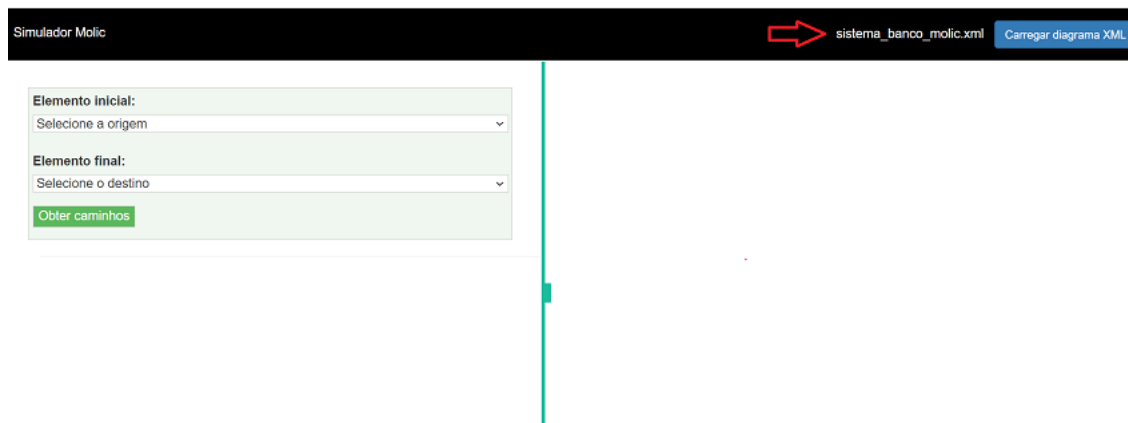
Figura 14 – Barra vertical ajustável



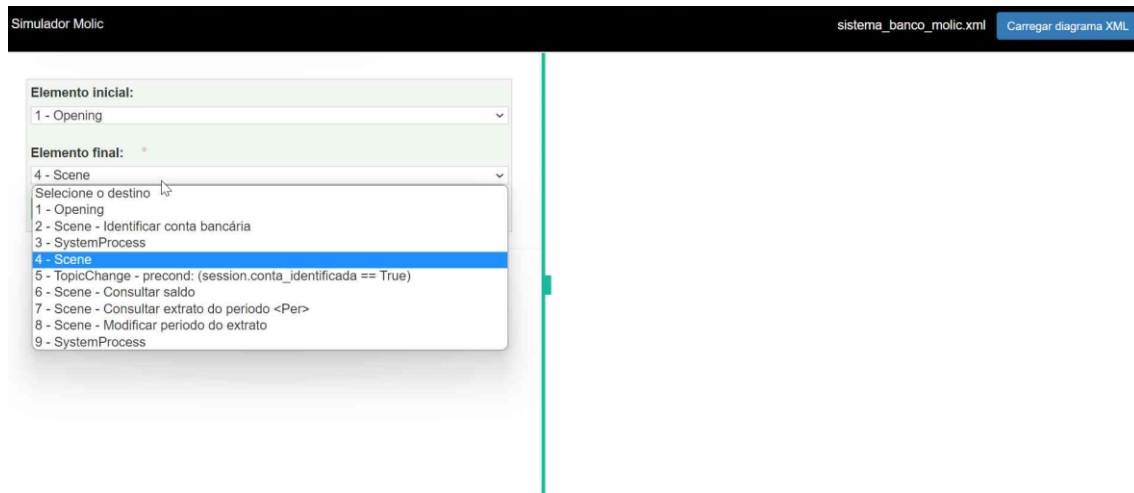
Carregado o diagrama XML, clique em OK para que o diagrama selecionado seja salvo na sessão do sistema consultas durante toda a execução da aplicação:



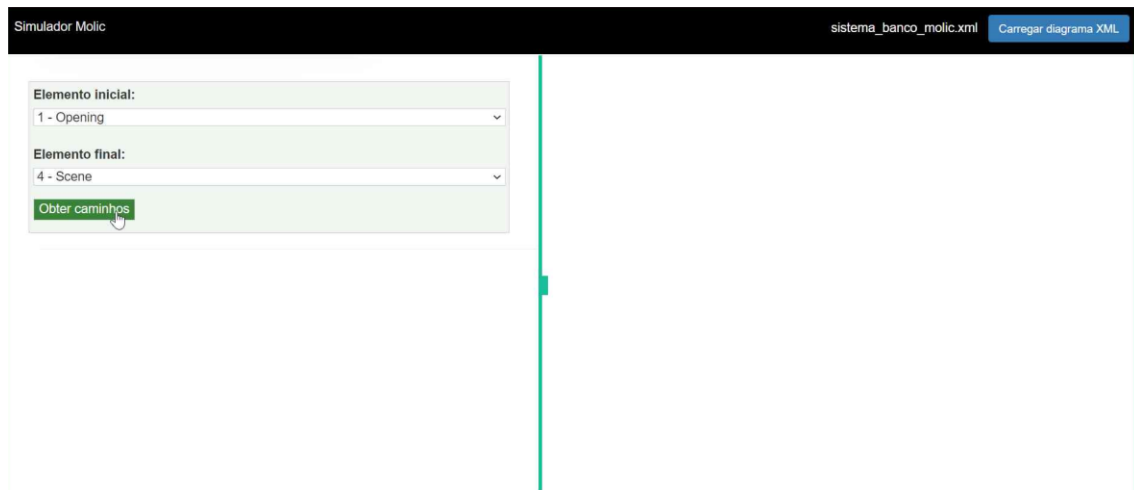
Feito o upload, o sistema aponta na barra horizontal superior o nome do arquivo cujo diagrama será consultado durante a sessão.



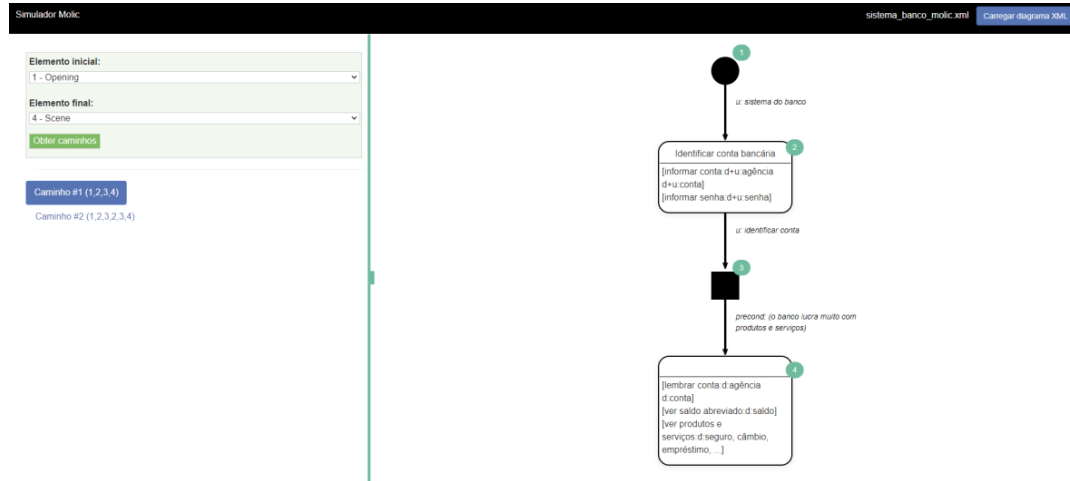
O sistema então apresenta todos os elementos do diagrama na lista de seleção de elemento inicial e na lista de seleção de elemento final. Aqui o usuário determina a origem e o destino da rota que deseja simular. Observe que não há a necessidade de ser em sequência, ou seja, o usuário pode selecionar um elemento inicial que vem depois do elemento final. Isto é possível quando existe retornos definidos no diagrama.



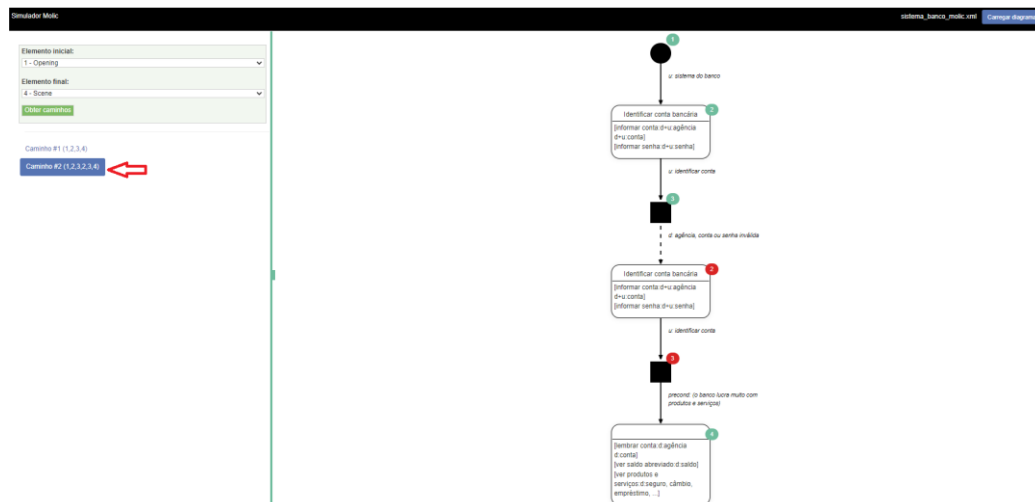
Selecionados o início e fim, o usuário pressiona o botão “Obter caminhos”:



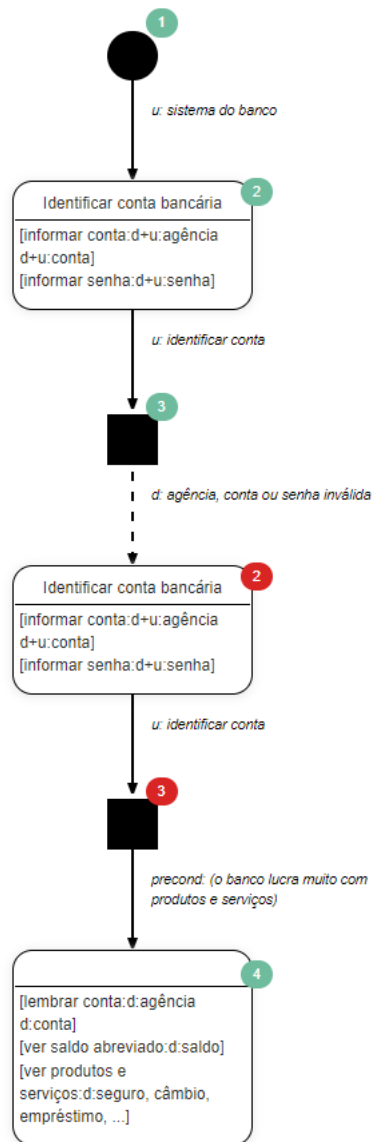
O sistema processa a solicitacao e cria sub-diagramas com base no diagrama completo, contendo as possíveis rotas entre o início e fim em ordem de complexidade. O sub-diagrama apresenta também o ID de cada elemento no canto superior direito em verde, que corresponde o ID presente nos caminhos entre parênteses exibidos no lado esquerdo da tela.



O usuário ao selecionar um outro caminho exibido, o sistema exibe o sub-diagrama no lado direito mantendo o caminho destacado em azul em formato de aba no lado esquerdo.



Os elementos que foram revisitados têm seus identificadores destacados em vermelho:



6. Links

Repositório público GitHub com código fonte completo:

<https://github.com/pedrohbg/puc-molic-simulator.git>

Visual Studio Community 2022:

<https://visualstudio.microsoft.com/vs/community/>