Nome da Empresa: JCJ Systems
Projeto: Editor Ladder

Visão Geral: Editor Ladder com recurso de importação e exportação para ML4.0

Requisitos Funcionais:

- O sistema deve permitir a edição de um programa em LADDER utilizando simplesmente sinais digitais. O editor deve permitir inserir, alterar e deletar sinais posicionados em série ou em paralelo;
- Cada sinal digital deverá ser identificado por um "label";
- Ao realizar a função "COPIAR" em um trecho selecionado do programa LADDER, o sistema deve enviar o código correspondente em ML4.0 para a área de transferência;
- Ao realizar a função "COLAR" dentro do editor, o sistema deve converter o código ML4.0 presente na área de transferência para LADDER;
- O sistema deve contemplar as seguintes instruções ML4.0: SEL, SED, E, EN, OU, OUN, MEMO.

Requisitos Não Funcionais:

- Cada label poderá possuir no máximo 50 caracteres;
- Deve suportar pelo menos 500 linhas de código ML4.0;
- A lógica em LADDER não deve possuir limite máximo de colunas.

Casos de Uso:

Nome: Editar lógica.
Autor: Usuário.
Descrição: O usuário cria lógicas LADDER a partir das instruções disponíveis no editor onde ele pode incluir, alterar e excluir qualquer instrução em suas lógicas.

Nome: Copiar para a área de transferência.

Autor: Usuário.

Descrição: O usuário seleciona a área de instruções que deseja e copia o código em ladder como código ML4.0.

Nome: Colar da área de transferência.

Autor: Usuário.

Descrição: O usuário seleciona uma posição na área de trabalho e cola o código ML4.0 como código ladder.

Planejamento do desenvolvimento:

Classificação dos casos de uso:

	DIFICULDADE	IMPORTANCIA
Editar lógica	ALTA	ALTA
Copiar para a área de transferência	MÉDIA	ALTA
Colar da área de transferência	MÉDIA	ALTA

Escalonamento Final:

- Editar lógica;
- Copiar para a área de transferência;
- Colar da área de transferência.

Caso de usos expandidos:

Nome: Editar lógica

Autor: Usuário

Fluxo Principal de Sucesso:

1 – O usuário abre o sistema;

2 – O sistema apresenta uma tela em branco com as instruções ladder disponíveis;

3 – O usuário e escreve o nome da instrução e seleciona uma das instruções LADDER para inserir;

4 – O sistema guarda a instrução e apresenta ela na linha selecionada pelo usuário;

5 – O usuário seleciona a instrução e seleciona a opção alterar;

6 – O usuário altera o nome e o tipo de instrução da instrução que tinha sido selecionada;

7 – O sistema registra a alteração e altera a apresentação da instrução na tela;

8 – O usuário seleciona uma instrução e seleciona a opção excluir;

9 – O sistema remove a instrução da lógica e da tela de apresentação.

Fluxo alternativo:

3a. O usuário não dá um nome para a instrução;

1 – O sistema mostra um aviso para o usuário e cancela a operação.

3b. O usuário tenta inserir uma instrução diferente de SED ou SEL no começo;

1 – O sistema mostra um aviso para o usuário e cancela a operação.

Nome: Copiar para Área de Transferência

Autor: Usuário

Fluxo Principal de Sucesso:

1 – O usuário cria sua lógica no programa;

2 – O usuário aperta o botão copiar;

3 – O sistema pega as instruções guardadas na lógica atual;

4 – O sistema passa as instruções guardadas na memória como instruções ML 4.0 para a área de transferência.

Fluxo alternativo:

2a. Não existe logica criada;

1 - O sistema indica que não há nada para copiar e cancela a operação.

Nome: Colar na Área de Trabalho

Autor: Usuário

Fluxo Principal de Sucesso:

1 – O usuário copia um código ML para sua área de transferência;

2 – O usuário aperta o botão colar;

3 – O sistema pega o código na área de transferência;

4 – O sistema analisa os dados recebidos e guarda a lógica na memória e apresenta ela na área de trabalho.

Fluxo alternativo:

2a. O código ML copiado não uma instrução de início;

1 - O sistema indica que não pode copiar sem instrução de início e cancela.

Modelo Conceitual:

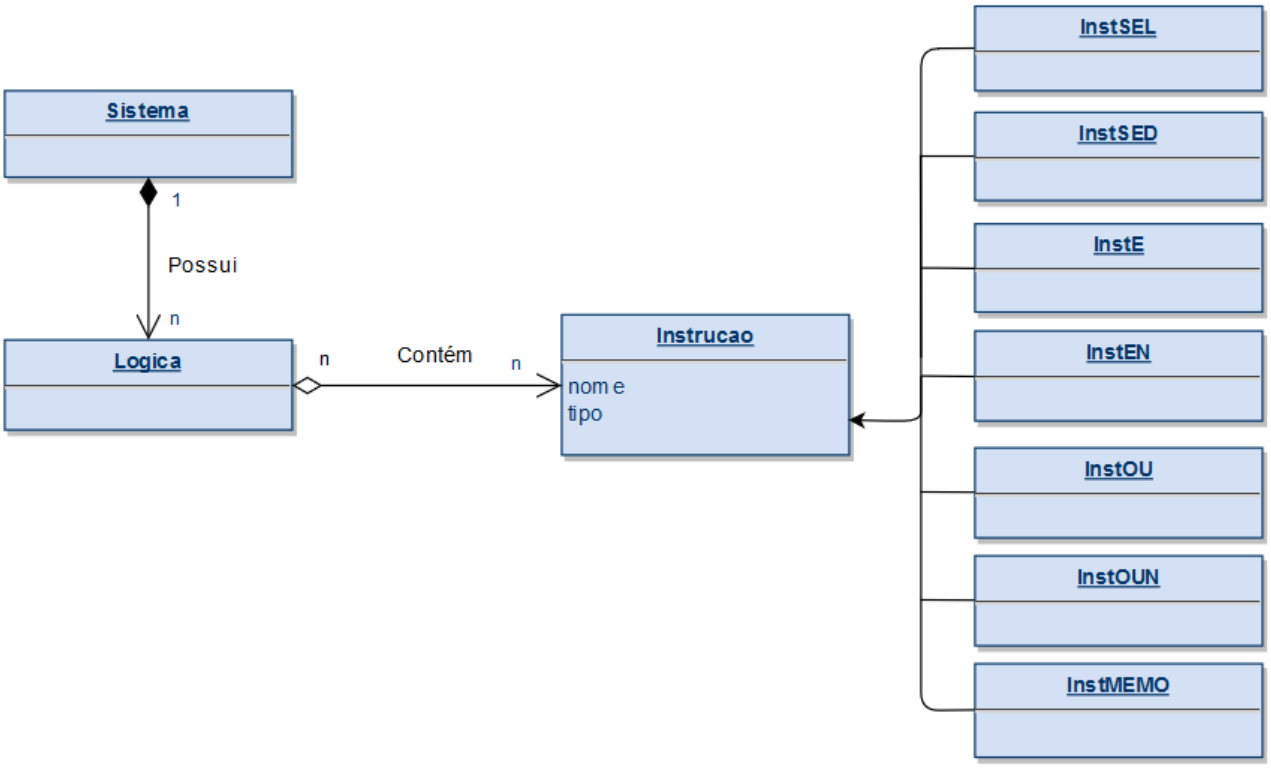
Definição de Classes:

Sistema[];

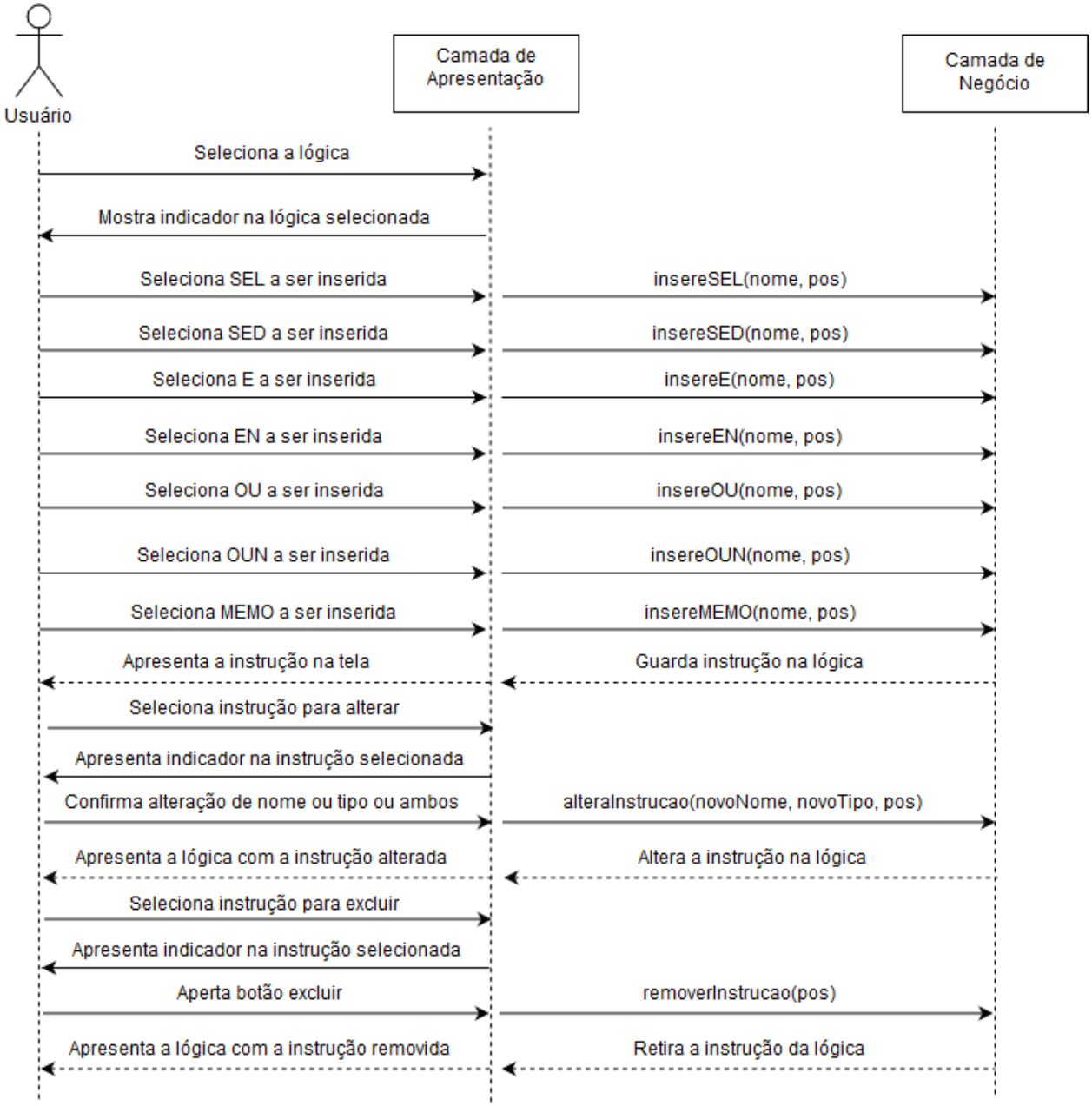
Instrução[nome, tipo];

Lógica[].

Diagrama de Classes:



Eventos – Editar lógica:



Contratos:

Nome: insereSEL

Responsabilidade: Insere a instrução SEL em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereSED

Responsabilidade: Insere a instrução SED em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereE

Responsabilidade: Insere a instrução E em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereEN

Responsabilidade: Insere a instrução EN em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereOU

Responsabilidade: Insere a instrução OU em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereOUN

Responsabilidade: Insere a instrução OUN em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: insereMEMO

Responsabilidade: Insere a instrução MEMO em uma lógica.

Pós-condições: Adiciona um novo objeto em uma lógica.

Nome: alterarInstrucao

Responsabilidade: Pega uma instrução de uma lógica e altera seu nome ou seu tipo de acordo com as informações dadas pelo usuário.

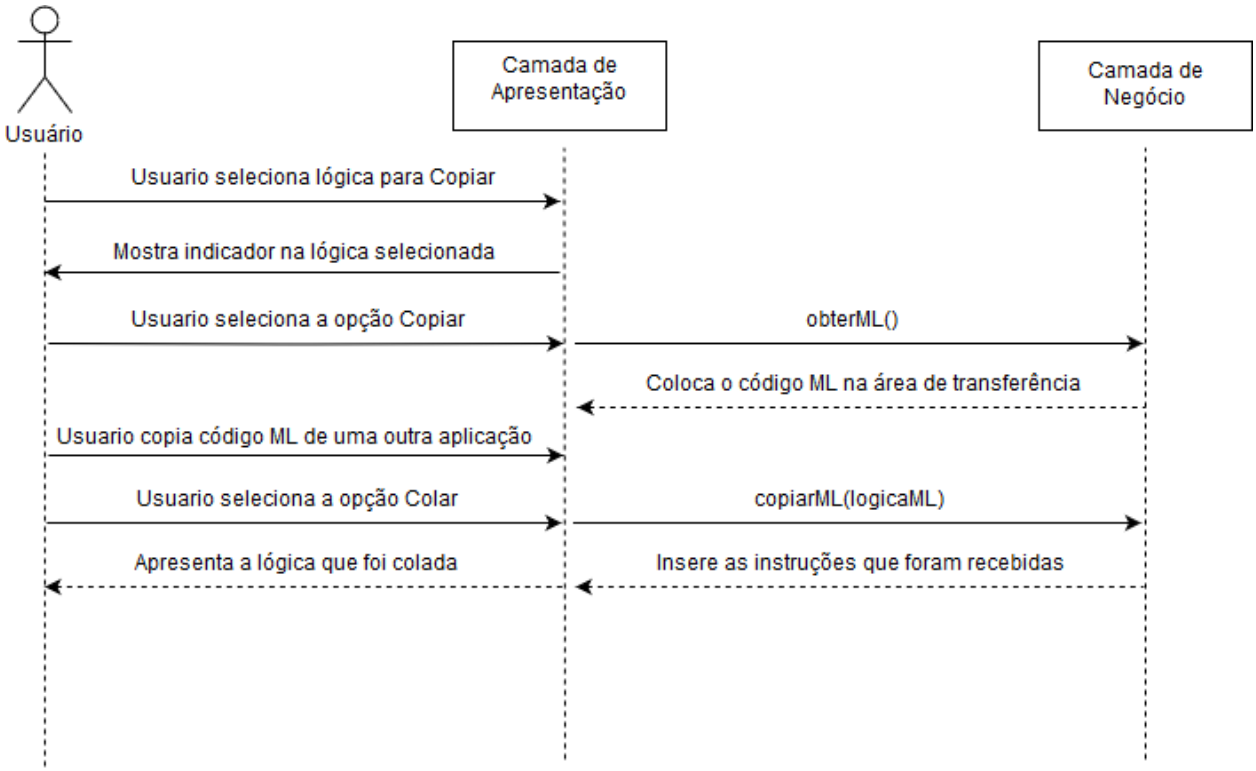
Pós-condições: O objeto que existia é alterado.

Nome: excluirInstrucao

Responsabilidade: Pega uma instrução de uma lógica e remove ela.

Pós-condições: O objeto que existia é removido da lógica.

Eventos – Copiar e Colar código ML:



Nome: obterML

Responsabilidade: Pega a lógica que estava selecionada e coloca as instruções em código ML na área de transferência.

Nome: copiarML

Responsabilidade: Recebe o texto que representa uma logica em código ML faz uma análise das instruções nela e insere elas no sistema.

Pós-condições: Uma nova lógica com objetos é criada na base de dados.

Análise e deploy do código:

O código do programa está localizado dentro da pasta “Editor Ladder” dentro do ficheiro zippado.

O código pode ser analisado em qualquer editor de texto, porém recomendo a utilização do Visual Studio 2015 para poder a abrir o ficheiro “JCJprojeto.sln” que permitirá fazer um debug do código em tempo de execução.

O .exe do programa pode ser encontrado na pasta “Editor Ladder\teste1\bin\Debug” com o nome “teste1.exe”. Que permite a execução direta do programa.