Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Informática
INF 2102 - Projeto Final de Programação

Nome do Aluno(a): Bruno Castro

Nome do Orientador: Marcus Poggi

Título do Projeto: IRP Viewer

Palavras-Chave: inventory routing, IRP, DIMACS, vehicle routing, VRP

1. Introdução

Irá ocorrer no mês de Fevereiro de 2022 uma competição organizada pela DIMACS para avaliar algoritmos para problemas de roteamento de veículos (VRP). Dentre as várias subdivisões encontra-se o problema do roteamento de inventário (IRP).

Em linhas gerais, o IRP é o problema onde dada uma matriz (depot) de um determinado produto, busca-se saber a melhor quantidade de entrega para n clientes ao longo de T períodos, de forma que cada cliente não ultrapasse seu estoque máximo, e também não fique sem estoque em nenhum período. Além disso, busca-se saber a melhor rota para cada K veículos, de forma que cada um não ultrapasse sua capacidade máxima Q de transporte. A melhor quantidade e rota baseia-se em minimizar o custo de armazenamento em cada cliente, e na distância percorrida por cada veículo.

Para esta competição, será utilizado uma coleção de instâncias de problemas, e espera-se receber de cada participante um arquivo contendo a solução para cada instância.

Todo arquivo de solução deverá respeitar o formato definido pelo documento de regras. Assim como a solução em si deverá respeitar as regras do problema também definidas.

2. Especificação

Escopo

Este sistema tem por finalidade ajudar os competidores do desafio DIMACS-IRP a validarem e visualizarem soluções do problema IRP.

Deverá ser um site web que permita aos usuário o acesso via navegador.

O site deverá permitir o upload de uma solução no formato da competição, realizar a validação com base nas regras da competição e por fim permitir uma visualização detalhada de cada solução.

Espera-se com este sistema facilitar a análise de soluções encontradas, e poder comparar o resultado com o de outros algoritmos.

Aceitação do sistema

Ao final do projeto, será marcada uma reunião online com os usuários para que possa ocorrer a aceitação do sistema. Para isto o sistema deverá estar acessível via internet no momento da reunião.

Inicialmente o sistema será demonstrado com uma breve apresentação e explicação dos principais recursos. Durante esta explicação, todos os Requisitos Não Funcionais deverão ser passados ponto a ponto.

Logo após será realizada uma demostração com três arquivos de solução:

- Arquivo com formato incorreto;
- Arquivo com formato correto e solução válida;
- Arquivo com formato correto e solução inválida;

Com estes três arquivos, será possível a validação e aceite dos Requisitos-Funcionais **RF-1**, **RF-2**, **RF-3**, **RF-4**, **RF-5**, **RF-6**.

Por existir um número grande de regras de validação do arquivo e regras da competição, deseja-se testes unitários com cobertura de todas as regras.

Artefatos (ART)

Ao final do projeto deseja-se:

- **ART-1:** Endereço do repositório GIT com o código fonte da última versão do sistema;
- **ART-2:** Dockerfile com as instruções de construção da imagem Docker do sistema;
- **ART-3:** Documentação explicando como montar um ambiente de desenvolvimento;
- **ART-4:** Arquivo com execução dos testes unitários sem falhas;

Listagem de Requisitos Não-Funcionais (RNF)

- **RNF-1:** O sistema deverá ser capaz de ler e processar uma solução de forma rápida (menos de 1s) desconsiderando a velocidade da internet;
- **RNF-2:** As soluções enviadas por cada competidor não deverá ser submetida aos servidores por razões de ser uma competição, sendo assim todo processamento deverá ser feito de forma local;
- **RNF-3:** O sistema deverá permitir enviar o arquivo de solução de forma fácil e intuitiva;
- **RNF-4:** O sistema deverá permitir o compartilhamento da solução de forma fácil e intuitiva;
- **RNF-5:** O sistema deverá ser de fácil utilização para usuários desktop ou computadores portáteis;
- **RNF-6:** O sistema deverá ser compatível com versões de navegadores recentes;
- **RNF-7:** O sistema deverá ser capaz de ser executado em dispositivos móveis, apesar de não haver requisitos de usabilidade;
- **RNF-8:** O sistema deverá ser compatível com monitores de 14 polegadas ou superior;
- **RNF-9:** O sistema deverá ser desenvolvido com nomenclaturas e comentários em língua inglesa;

- **RNF-10:** Toda mensagem de sistema deverá ser na língua inglesa;

Listagem de Requisitos Funcionais (RF)

- **RF-1:** A tela inicial do sistema deverá conter informações relevantes como uma introdução ao problema, e instruções em como utilizar o sistema;
- **RF-2:** O sistema não deverá aceitar arquivos que não estejam no formato esperado;
- **RF-3:** O sistema deverá aceitar arquivos que estejam no formato esperado e que possua soluções válidas;
- **RF-4:** O sistema deverá aceitar arquivos que estejam no formato esperado e que possua soluções inválidas;
- **RF-5:** O sistema deverá mostrar detalhes da instância do problema;
- **RF-6:** O sistema deverá mostrar detalhes da solução enviada;

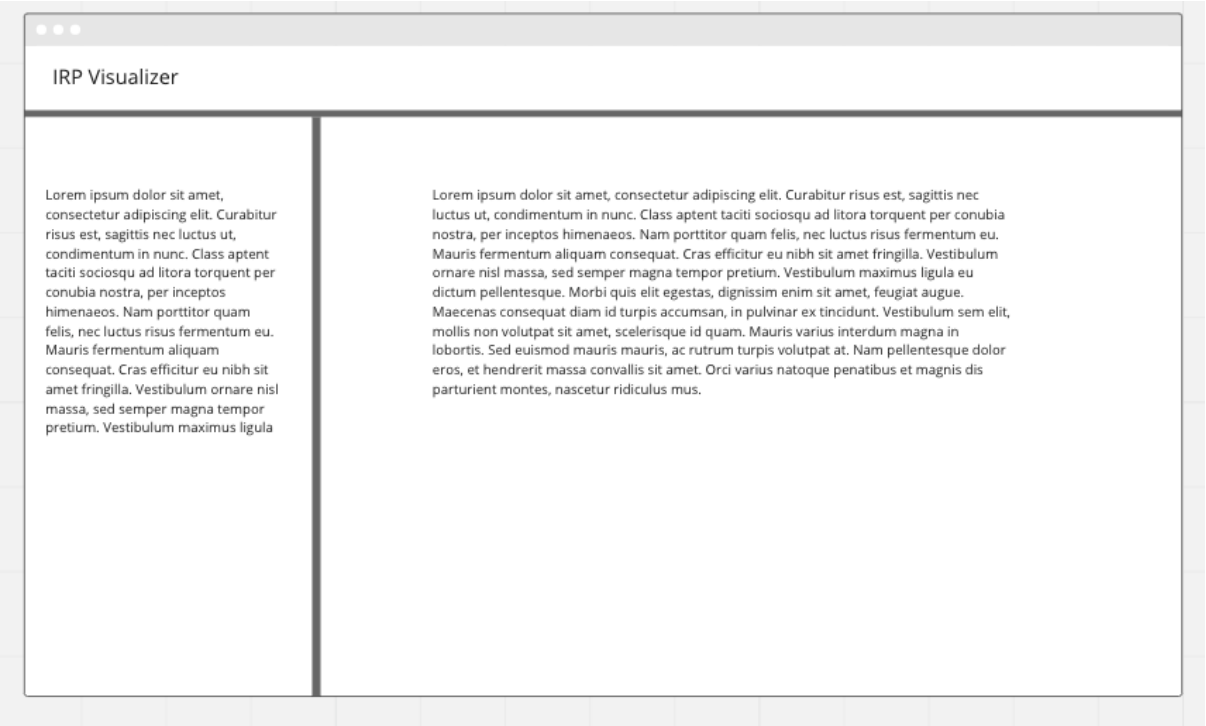
Detalhamento dos Requisitos funcionais

RF-1: A tela inicial do sistema deverá conter informações relevantes como uma introdução ao problema, e instruções em como utilizar o sistema;

Ao abrir o site, o usuário deverá ser apresentado uma tela parecida com a seguinte imagem. Este é o esqueleto da aplicação, com um cabeçalho onde se encontra o título do sistema, uma coluna lateral e uma tela principal.

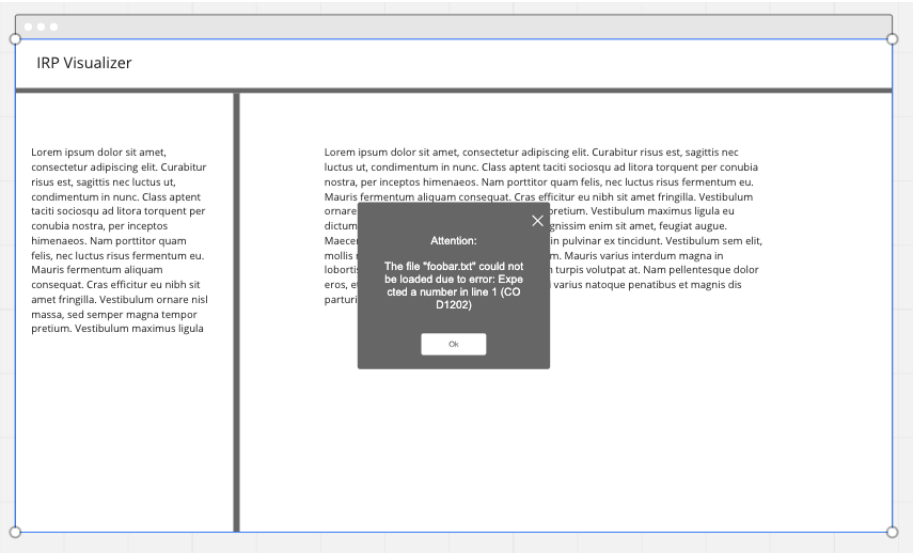
Na coluna lateral, deverá ser apresentado informações sobre o objetivo do sistema.

Na tela principal, deverá ser apresentado as instruções de como utilizar o sistema.



RF-2: O sistema não deverá aceitar arquivos que não estejam no formato esperado;

Após o envio de um arquivo, o sistema deverá validar o formato do arquivo. Em caso de falha será mostrado uma caixa de alerta conforme a imagem a seguir



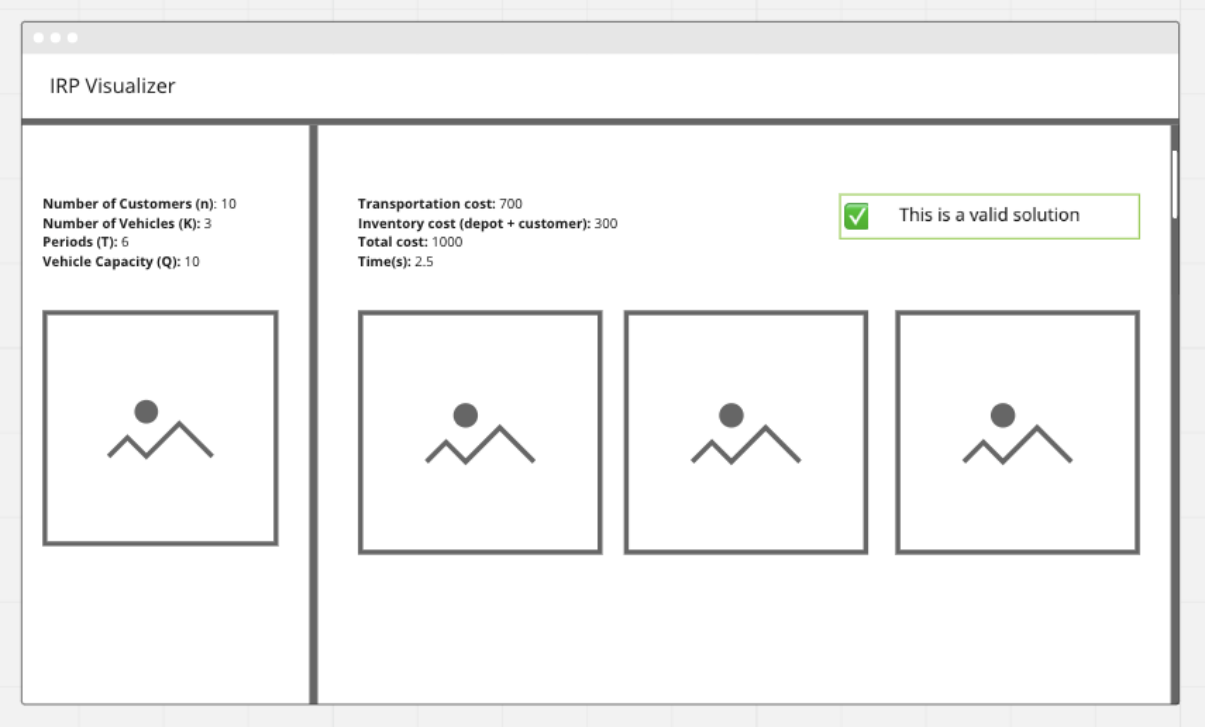
RF-3: O sistema deverá aceitar arquivos que estejam no formato esperado e que possua soluções válidas;

Após o envio e validação do formato do arquivo, o sistema deverá alterar a tela para a seguinte forma, porém mantendo ainda a disposição definida em RF1.

Na coluna lateral, deverão ser apresentadas informações dos detalhes da instância em questão. Para mais detalhe RF5.

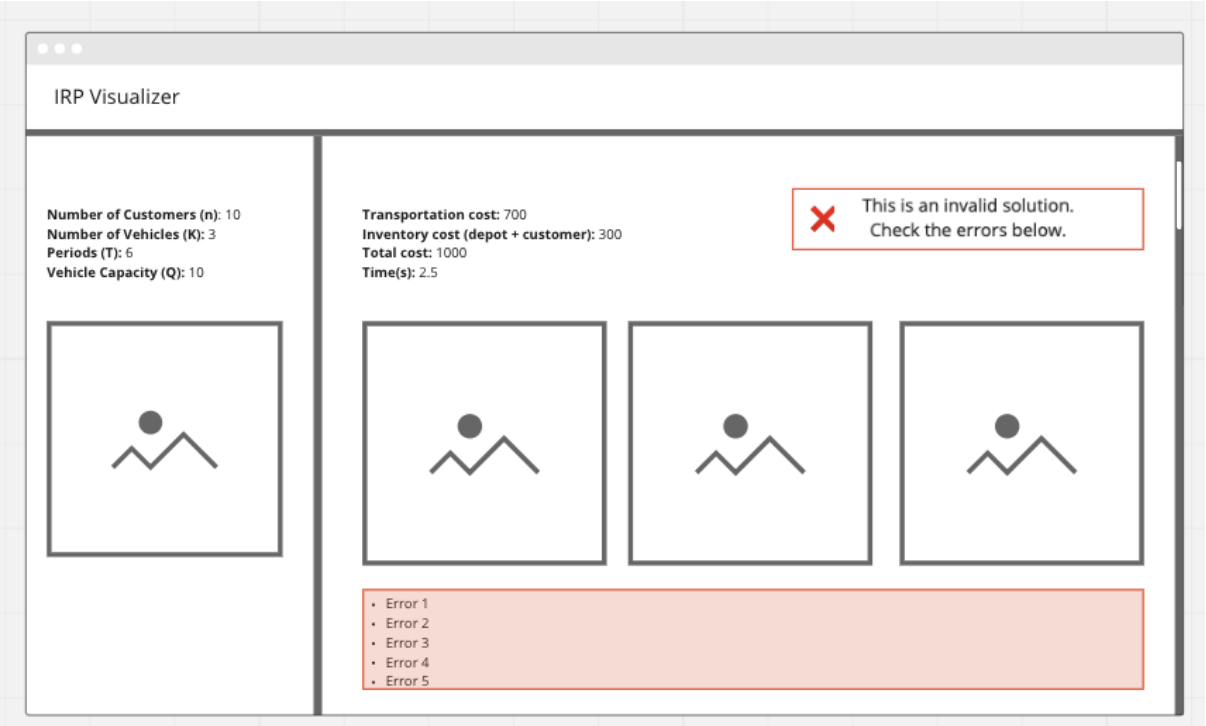
Na tela principal, deverão ser apresentadas informações dos detalhes da solução em questão. Para mais detalhe RF6.

Uma caixa verde deverá ser apresentada no canto superior direito da tela demonstrando que a solução é válida segundo as regras.



RF-4: O sistema deverá aceitar arquivos que estejam no formato esperado e que possua soluções inválidas;

Complementando o RF3, para o caso da solução apresentada não ser válida. Uma mensagem em vermelho deverá ser mostrada no canto superior direito, assim como uma caixa vermelha com uma lista apresentando todos os erros encontrados:



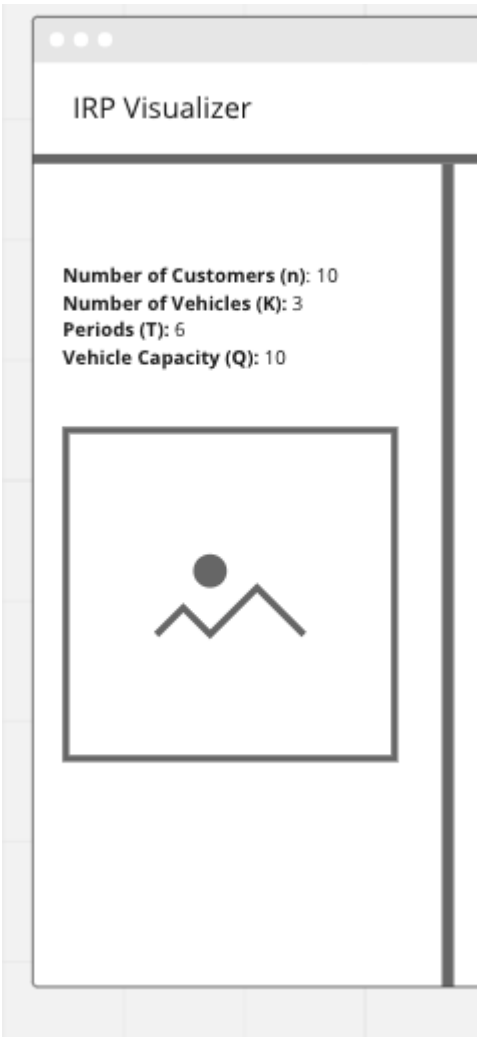
RF-5: O sistema deverá mostrar detalhes da instância do problema;

Os detalhes da instância desejado são:

Título	Formato do valor	Descrição
Number of Customers (n)	Numérico sem precisão decimal.	Número de clientes
Number of Vehicles (K)	Numérico sem precisão	Número de veículos

	decimal.	
Periods (T)	Numérico sem precisão decimal.	Período
Vehicle Capacity (Q)	Numérico sem precisão decimal.	Capacidade do veículo

Deverá ser apresentado um mapa com pontos indicando a posição de cada cliente e a posição do depósito.

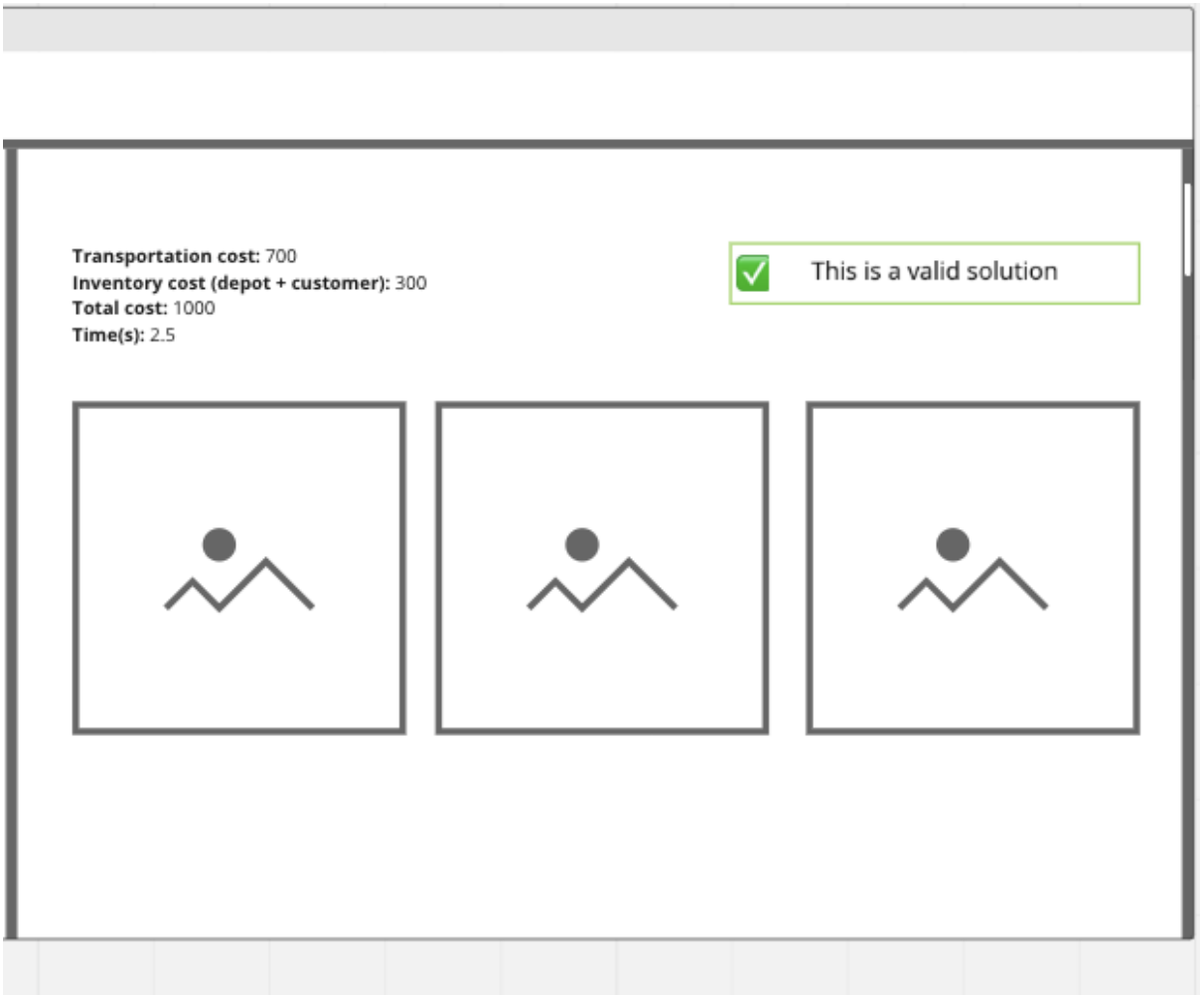


RF-6: O sistema deverá mostrar detalhes da solução enviada;

Os detalhes da instância desejado são:

Título	Formato do valor	Descrição
Transportation cost	Numérico sem precisão decimal.	Custo de transporte extraído do arquivo de solução.
Inventory cost (depot + customer)	Numérico com precisão decimal de duas casas com a seguinte formação: TOTAL (Custo-Depot + Custo-Customer)	Custo de inventário do depósito e dos clientes extraído do arquivo de solução. A soma dos dois valores deverá ser apresentada como soma total.
Total cost	Numérico com precisão decimal de duas casas	Custo total extraído do arquivo de solução.
Time(s)	Numérico com precisão decimal de duas casas	Tempo de execução do algoritmo que gerou a solução. Valor extraído do arquivo de solução.

Deverá ser apresentado um mapa com pontos indicando a posição de cada cliente, a posição do depósito e as rotas para cada veículo em cada período. Ou seja, nas instâncias de três períodos deverão ser mostrados três mapas, um para cada período, e nas instâncias de seis períodos, seis mapas.



Regras de Negócio (RN)

Com relação ao formato da instância

- **RN-1:** Primeira linha com quatro parâmetros: número de nós incluindo o depot (n+1); número de períodos T; capacidade do veículo Q; número de veículos;
- **RN-2:** Segunda linha descreve o depot com: Identificador (no caso, sempre zero); coordenada x; coordenada y; quantidade inicial do inventário; produção diária; custo do inventário;
- **RN-3:** As próximas n linhas descrevem um cliente com: Identificador; coordenada x; coordenada y; quantidade inicial do inventário; máximo de inventário; mínimo do inventário; consumo diária; custo do inventário;

Com relação ao formato da solução

- **RN-4:** O nome do arquivo de solução deve estar no formato "out_INSTANCE.txt"
- **RN-5:** Devem constar todos períodos
- **RN-6:** Rotas para cada período devem estar numeradas de 1 a M
- **RN-7:** Devem existir M rotas. Rotas vazias devem estar com 0 - 0
- **RN-8:** Toda rota deve iniciar e terminar no nó 0 (depot)
- **RN-9:** Devem constar 6 linhas finais com o seguinte valores: custo de transporte; custo de inventário no cliente; custo de inventário no depot; custo total; processador; tempo;
- **RN-10:** Formato das rotas deve seguir: 0 – customer (quantity delivered) – . . . customer (quantity delivered)–0

Com relação a validade da solução

- **RN-11:** Restrição de capacidade do veículo. O somatório de entregas de cada rota deverá ser menor que a capacidade do veículo definido pela instância.
- **RN-12:** Restrição de inventário mínima para depot. O inventário do depot em período t, não pode ser menor que zero.
- **RN-13:** Restrição de inventário mínima para cliente. O inventário do cliente em período t, não pode ser menor que o mínimo definido pela instância.
- **RN-14:** Restrição de inventário máxima para cliente. O inventário do cliente em um período t, não pode ser maior que a soma do inventário em t-1 e a quantidade recebida em t. Regra conhecida como: entrega antes do consumo.
- **RN-15:** Restrição de visitação única por período. Um cliente pode receber no máximo uma visita no período;
- **RN-16:** Cálculo de custo de transporte. É a distância euclidiana dada pela fórmula

$$d_{ij} = int((sqrt((x_i - x_j)^2 + (y_i - y_j)^2) + 0.5)$$

- **RN-17:** Custo de inventário de cliente. É o custo do inventário no período t-1 mais a quantidade recebida no período t menos a quantidade consumida no período t. Deve-se desconsiderar custos do inventário inicial.
- **RN-18:** Custo de inventário no depot. É o custo do inventário no período t-1 menos a quantidade produzida no período t menos a quantidade entregue no período t. Deve-se desconsiderar custos do inventário inicial.
- **RN-19:** Custo de inventário total. É o custo de inventário no depot, mais o custo de inventário no cliente

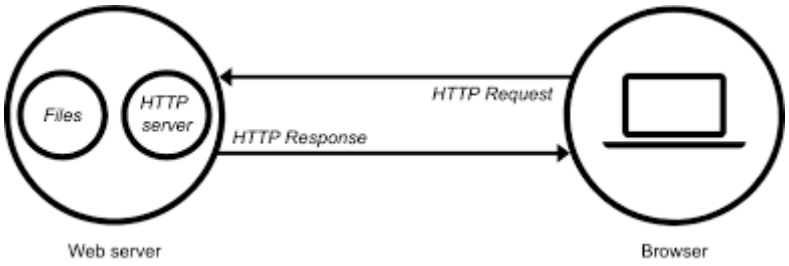
3. Arquitetura e Projeto

Este projeto é um simples sistema web, e como principais componentes temos:

- **Client-Side (navegador web):** Executado no computador dos usuários. Responsável pelo requerimento das páginas web, renderização do código HTML+CSS e execução de códigos JavaScript .
- **Server-Side (servidor web):** Executado no servidores. Responsável pelo armazenamento e resposta aos clientes por este conteúdo. Neste projeto ainda conta com uma API de serviço que será discutida mais à frente.

O protocolo de comunicação entre esses dois componentes é o já conhecido HTTP.

A seguir será apresentado com mais detalhe informações sobre as tecnologias escolhidas para cada componente.



Client-Side

Anonimidade da solução (RNF-1 e RNF-2)

Para satisfazer os RNF-1 e RNF-2, todo processamento em torno de uma uma solução deverá ser realizado no client-side. Isto é, o usuário irá enviar uma solução para o navegador, que este deverá ser capaz de executar os algoritmos de validação da solução.

Usabilidade (RNF-3)

Para satisfazer o RNF-3, será utilizado a interface HTML Drag and Drop. Com este recurso os usuários poderão arrastar um arquivo a partir de seu sistema de arquivo local até o navegador, que será responsável por ler o conteúdo e disponibilizar para acesso via JavaScript. https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API

Compartilhamento de soluções (RNF-4)

Para poder montar uma requisição HTTP para o servidor Web, os navegadores utilizam como entrada as URIs, também conhecido como endereço web. E um dos recursos que os navegadores exploram das URIs para poder realizar navegação dentro de uma própria página é a utilização dos Hashes. Os Hashes são a porção das URIs que aparecem após o caractere '#', e todo conteúdo após este marco é consumido apenas dentro do navegador, não é enviado para o servidor na forma de requisição.

Este recurso será importante para satisfazer o RNF-4. Uma vez que o usuário enviar o arquivo de solução para o navegador, este deverá transformar todo conteúdo recebido em um formato que possa ser anexado no campo de hash das URIs. Assim, uma URI possuirá toda informação de uma solução, e desta forma o compartilhamento poderá ser feito apenas copiando e colando a URI do navegador.

Por exemplo, um arquivo solução com o seguinte conteúdo:

```
Day 1
Route 1: 0 - 1 ( 10 ) - 0
Route 2: 0 - 2 ( 20 ) - 0
10
20
```

Deverá ser transformado utilizando o formato URL-Encode, que terá como resultado:

```
Day%201%0ARoute%201%3A%200%20-%201%20%28%2010%20%29%20-%200%0ARoute%
202%3A%200%20-%202%20%28%2020%20%29%20-%200%0A10%0A20%0A
```

E assim a URI final será:

```
http://irp-viewer.com.br/#content=Day%201%0ARoute%201%3A%200%20-%201
%20%28%2010%20%29%20-%200%0ARoute%202%3A%200%20-%202%20%28%2020%20%2
9%20-%200%0A10%0A20%0A
```

Boas práticas de desenvolvimento web (RNF-5, RNF-6, RNF-7, RNF-8)

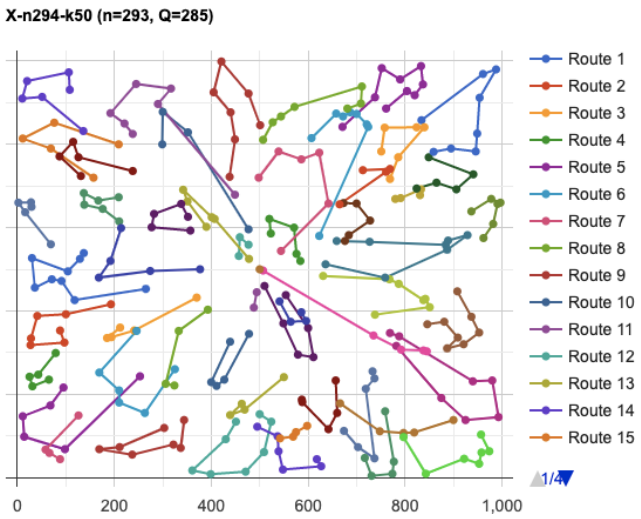
O código executado nos navegadores web seguirá os padrões HTML5, CSS3 e JavaScript ES6. Estas tecnologias já são amplamente difundidas e assim atenderão os RNF-5 e RFN-6.

Algo importante de se dizer que com o JavaScript ES6 é possível utilizar-se de módulos, um recurso muito importante para manutenibilidade do código, uma vez que permite-se a separação de código entre diversos arquivos diferentes (também conhecido por modules) e assim permitindo reutilização de código e também uma melhor organização do projeto.

Devido a natureza do projeto onde o objetivo é ser um apoio aos participantes da competição, foi discutido durante o levantamento de requisitos não ser necessário a boa usabilidade em dispositivos móveis. Sendo assim, para este sistema, não será necessário o desenvolvimento de páginas do tipo responsiva, onde o layout é ajustado de acordo com o tamanho da tela do usuário. Assim atendendo os RNF-7 e RNF-8.

Desenho de rotas

Por fim, para desenho das rotas, como requisitados nos RF-5 e RF-6, deverá ser utilizado a biblioteca Google Charts. Como referência, pode-se verificar o seguinte gráfico retirado da CVRPLIB:



Server Side

O servidor web, por definição, tem a responsabilidade de responder às requisições HTTP com os os arquivos que os navegadores web necessitam. Para esta aplicação serão utilizados os arquivos de HTML, CSS e JavaScript.

Validação da solução

Porém, apesar de todo processamento de validação ser feito no client-side uma informação importante o usuário não é capaz de informar com apenas o conteúdo da solução. É necessário saber informações da solução.

Uma forma de solucionar este impasse seria solicitar ao usuário o envio do arquivo de instância em conjunto do arquivo de solução. Porém isto não é prático, haveria a necessidade de parte do usuário de buscar dois arquivos diferentes em seu sistema de arquivo. Assim o servidor web deverá ter a capacidade de armazenar e informar os dados de uma instância, que será feito através de uma API Rest.

Detalhamento da API

GET /api/instance/<instance-id>

Overview

Recebe o pedido de detalhamento de uma instância de problema e retorna com o resultado

Request

Query Param	Tipo	Descrição	Comentário	Exemplo
<instance-id>	string	Identificador único de instância como definido pelas regras do DIMACS	É um valor obrigatório	L_abs2n100_4_L

Response 200

Resposta com sucesso. Deverá ser retornado um objeto JSON no seguinte modelo:

Campo	Tipo	Descrição	Exemplo
instanceId	string	Identificador único de instância como definido pelas regras do DIMACS	L_abs2n100_4_L
n	numérico	Número de clientes	
T	numérico	Número de períodos	
Q	numérico	Capacidade máxima do veículo	
K	numérico	Número de veículos	
nodes	array<Node>	Lista de nós (clientes e depósito)	

Modelo para Node:

Campo	Tipo	Descrição	Exemplo
id	numérico	Identificador do nó. Zero significa depósito	L_abs2n100_4_L
x	numérico	Coordenada x	
y	numérico	Coordenada y	
startInv	numérico	Inventário inicial	
maxInv	numérico	Máximo inventário	
minInv	numérico	Mínimo inventário	
daily	numérico	Para um nó cliente, é a quantidade consumida diariamente. Para o depósito, é a quantidade produzida diariamente.	
cost	numérico	Custo do inventário	

Diagramas UML

Para facilitar e consolidar entendimento, a seguir será apresentado a modelagem do sistema Client-Side. O sistema Server-Side por ser mais simples foi deixado como um ator externo.

Os dois atores no sistema são

- **Usuário:** O usuário que fará a interação com o navegador
- **Servidor Web:** O Server-Side.

Diagrama de Caso de Uso (UC)

UC-1: Abrir sistema

Ação do usuário de abrir o navegador e entrar com o endereço da página. Este caso de uso implica no UC-2.

UC-2: Busca páginas HTML, CSS e Javascript

Ação tomada automaticamente pelo navegador, que realiza uma chamada HTTP para o Servidor-Web, requisitando os arquivos.

UC-3: Enviar solução

Ação tomada pelo usuário onde é realizado o Drag-and-Drop da solução. Este caso de uso implica no UC-4.

UC-4: Encoding da solução

Ação tomada automaticamente pelo navegador, por meio de código JavaScript. É realizado o enconding da solução e alteração da URI.

UC-5: Verificar formato da solução

Ação tomada automaticamente pelo navegador, são aplicadas todas regras definidas pelo DIMACS. Caso verifique-se um erro, ocorre-se o UC-6.

UC-6: Mostrar erros do formato da solução

Ação tomada automaticamente pelo navegador. É mostrado para o usuário os erros encontrados no formato da solução.

UC-7: Visualizar detalhes da instância

Mostrado para o usuário os detalhes da instância. Para isso, é necessário solicitar ao Servidor-Web os detalhes da instância.

UC-8: Buscar detalhes da instância

Ação tomada automaticamente pelo navegador. Requisição REST à API GET /api/instance/instance-id

UC-9: Visualizar detalhes da solução

Mostrado para o usuário os detalhes da solução.

UC-10: Verificar validade da solução

Ação tomada automaticamente pelo navegador, são aplicadas todas regras definidas pelo DIMACS. Caso verifique-se um erro, ocorre o UC-11.

UC-11: Visualizar validade da solução

É mostrado para o usuário os erros encontrados durante a validação da solução.

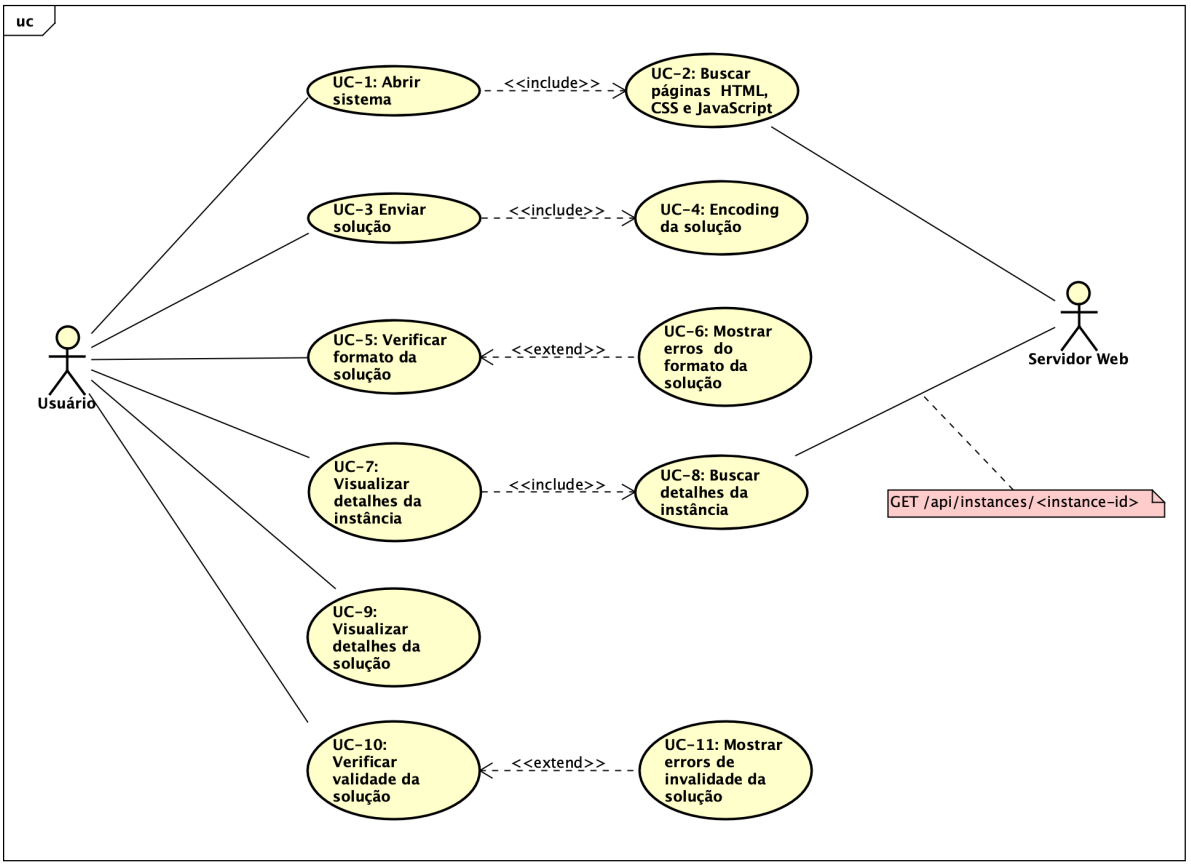
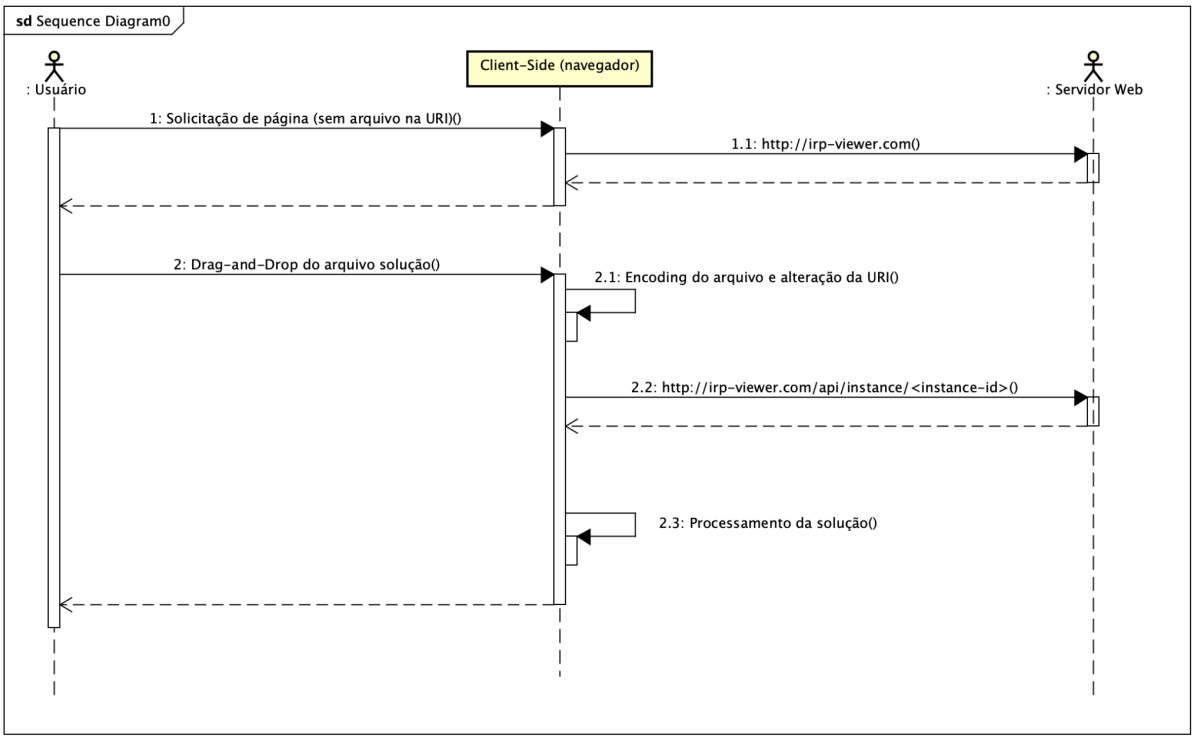


Diagrama de Sequência

O processo inicia-se com o usuário abrindo a página inicial do sistema (Mensagem 1). O navegador processa esse pedido enviando uma solicitação (Mensagem 1.1) ao servidor web. A resposta vem no formato dos arquivos HTML, CSS e JavaScript. Neste momento o usuário é capaz de interagir com o sistema.

A interação inicia-se com o usuário arrastando um arquivo de solução (Mensagem 2) para a tela do navegador. O navegador por sua vez altera a URI com o conteúdo do arquivo (Mensagem 2.1). A alteração dispara um código JavaScript que faz uma chamada REST ao servidor solicitando informações da instância do problema. De posse da solução e da instância do problema, o navegador é capaz de realizar o processamento (Mensagem 2.3) e retornar a resposta para o usuário.



4. Detalhes de implementação

Tipagem de objetos

A linguagem escolhida para o desenvolvimento foi o JavaScript que é uma linguagem dinamicamente tipada, ou seja, o tipo de cada variável é inferido em tempo de execução. Este recurso, apesar de reduzir o *boilerplate* de código, reduz também a legibilidade do código.

Para amenizar este problema, um padrão comumente adotado pela comunidade é de se adicionar a tipagem através de comentários comentar utilizando o padrão JSDoc (https://jsdoc.app/) que é uma API para geração de documentação (similar ao Javadoc). Além de facilitar a geração de documentação, é tido como uma boa prática pela comunidade, uma vez que permite que diversas IDEs realizem a análise sintática do código, ajudando com o auto-complete.

Uma outra boa prática é ter um arquivo de definição que contempla objetos que serão utilizados ao longo do código. Neste projeto você poderá encontrar com o nome de *typedef.js*.

Língua utilizada para o desenvolvimento

Seguindo o RNF-9, todo código (nome de variável, métodos, comentários…) deve ser escrito em inglês.

Modularização de código

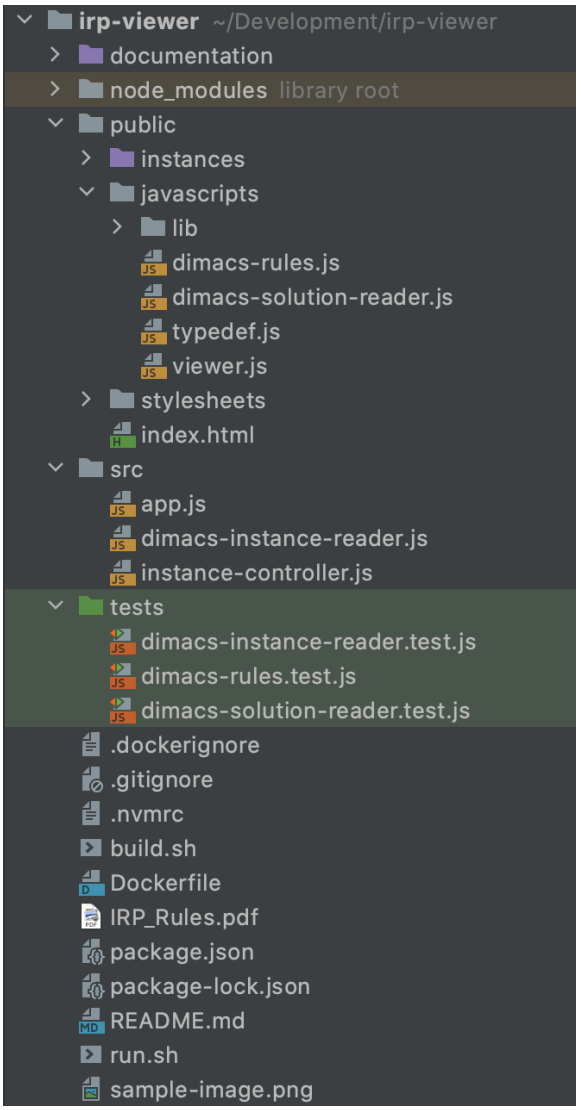
Uma boa prática de desenvolvimento é permitir a separação de códigos em arquivos (ou conjunto de arquivos) conforme sua afinidade.

O JavaScript não é uma linguagem que priorizou este fundamento desde sua concepção, portanto a modularização só foi possível em versões mais recentes da linguagem. Por isso, para a futura continuação do projeto é importante ter em mente que algumas bibliotecas porão não ter compatibilidade com o novo formato.

Estrutura de arquivos

O projeto está organizado da seguinte estrutura de pastas:

- ***documentation***: Possui toda documentação do projeto
- ***public***: Arquivos que serão executados no Client-Side. Em outras palavras, todo arquivo acessível pelo protocolo HTTP (arquivos HTML, CSS e JavaScript)
- ***src***: Arquivos que serão executados no Server-Side.
- ***tests***: Arquivos de teste unitário**.**



5. Controle de Qualidade

Para atestar a confiabilidade do programa, o controle de qualidade se dará por teste manuais, seguindo o plano de teste, e testes unitários que implementam as regras de negócio definidas na especificação.

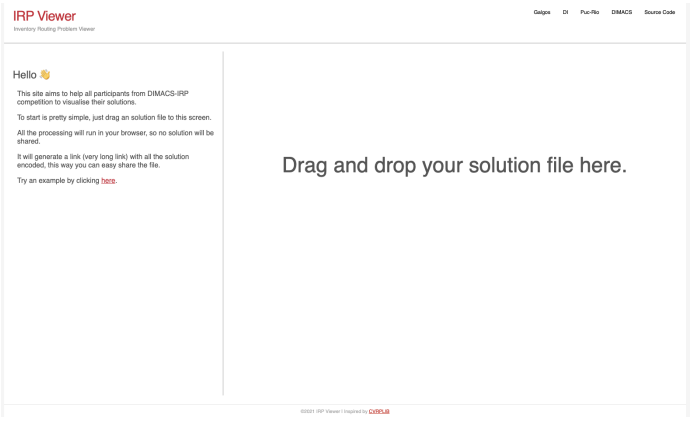
Para testes unitários, será utilizado a biblioteca JestJS (https://jestjs.io/). Sua escolha é devida sua simplicidade por além de ser um framework para teste unitários, conta com APIs para Mocks e Assertions.

Casos de Teste (CT)

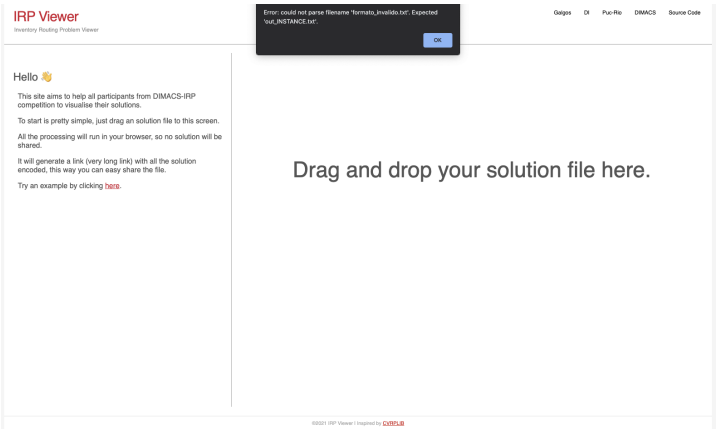
ID	Resumo	Pré-Condição	Entrada	Ação	Resultado esperado
TC-1	Testar a tela inicial do sistema	-	-	Acessar a tela inicial do sistema	Tela definida pela RF-1
TC-2	Testar a validação de um arquivo em formato inválido	TC-1	Arquivo: formato_invalido.txt	Arrastar e soltar arquivo na região indicada	Caixa de alerta com mensagem descritiva. Tela definida pela RF-2.
TC-3	Testar a validação de um arquivo em formato válido e solução válida	TC-1 e TC-2	Arquivo: out_S_abs1n5_2_L3 .txt	Arrastar e soltar arquivo na região indicada	Tela definida pela RF-3
TC-4	Testar a validação de um arquivo em formato válido e solução inválida	TC-1 e TC-2	Arquivo: out_S_abs1n5_3_L3 .txt	Arrastar e soltar arquivo na região indicada	Tela definida pela RF-4

Evidências:

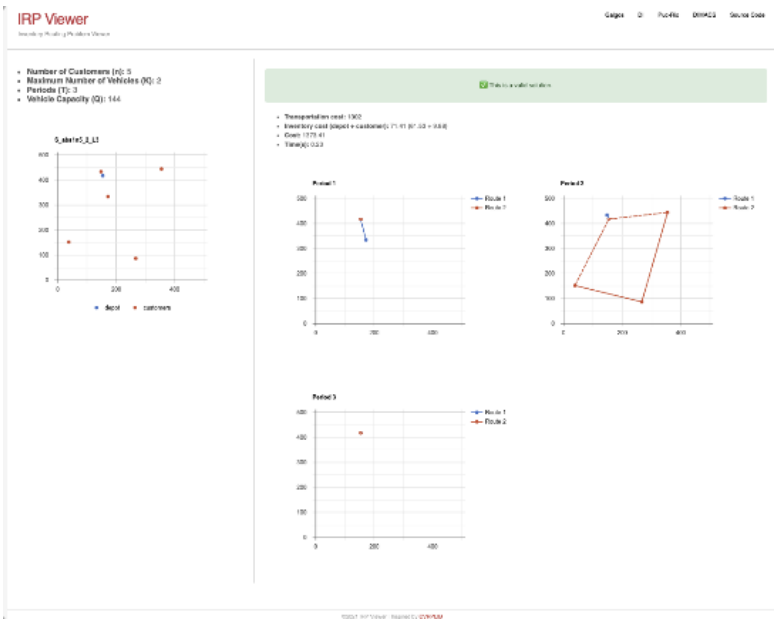
TC-1: Testar a tela inicial do sistema



TC-2: Testar a validação de um arquivo em formato inválido



TC-3: Testar a validação de um arquivo em formato válido e solução válida



TC-4: Testar a validação de um arquivo em formato válido e solução inválida

Testes Unitários

Sendo as Regras de Negócio um componente importante para este sistema e também ao mesmo tempo com vários casos específicos, foi desejado criar-se testes de unidades para cada Regra de Negócio.

Os testes unitários se encontram dentro da pasta tests. e podem ser executados seguindo o comando a partir do diretório raiz:

```
npm test
```

Evidências

```
➜  irp-solution-debug npm test

> irp-solution-debug@1.0.0 test /Users/brunocastro/Development/irp-solution-debug
> node --experimental-vm-modules node_modules/.bin/jest --verbose

(node:16824) ExperimentalWarning: VM Modules is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
 PASS  tests/dimacs-solution-reader.test.js
  ✓ empty filename should raise an error (4 ms)
  ✓ empty content should raise an error
  ✓ filename should follow the pattern out_INSTANCE.txt (case sensitive) (2 ms)
  ✓ instance id should follow the pattern defined by DIMACS (case sensitive) (2 ms)
  ✓ solution file must contains Days starting from 1 (1 ms)
  ✓ solution file must contains Routes starting from 1 (1 ms)
  ✓ solution file must contains all routes
  ✓ route lines must follow pattern defined by DIMACS (with spaces) (1 ms)
  ✓ route lines must start and finishes in node 0 (1 ms)
  ✓ should contains 6 lines in the end of the file (1 ms)
  ✓ should contains 6 numbers in the end of the file
  ✓ parse correct solution routes and deliveries (2 ms)
  ✓ parse correct solution results (1 ms)

 PASS  tests/dimacs-instance-reader.test.js
  ✓ empty instance should raise an error (1 ms)
  ✓ the first line should contains four parameters (1 ms)
  ✓ the second line should contains six parameters
  ✓ the third line should contains eight parameters (1 ms)
  ✓ read the instance S_abs1n5_2_H3 and check for the correct parameters according to session 5 (3 ms)

 PASS  tests/dimacs-rules.test.js
  ✓ sanity check, test the base instance and solution for the next tests (1 ms)
  ✓ vehicle capacity constraint, a route should delivery less than the vehicle capacity (1 ms)
  ✓ depot inventory constraint, the depot never should stock out (1 ms)
  ✓ customer inventory min constraint, the customer never should has less than the minimum
  ✓ customer inventory max constraint, the customer never should has more than the maximum (1 ms)
  ✓ customers should be visited once per period
  ✓ Euclidean distance should round up

Test Suites: 3 passed, 3 total
Tests:       25 passed, 25 total
Snapshots:   0 total
Time:        0.313 s, estimated 1 s
Ran all test suites.
```

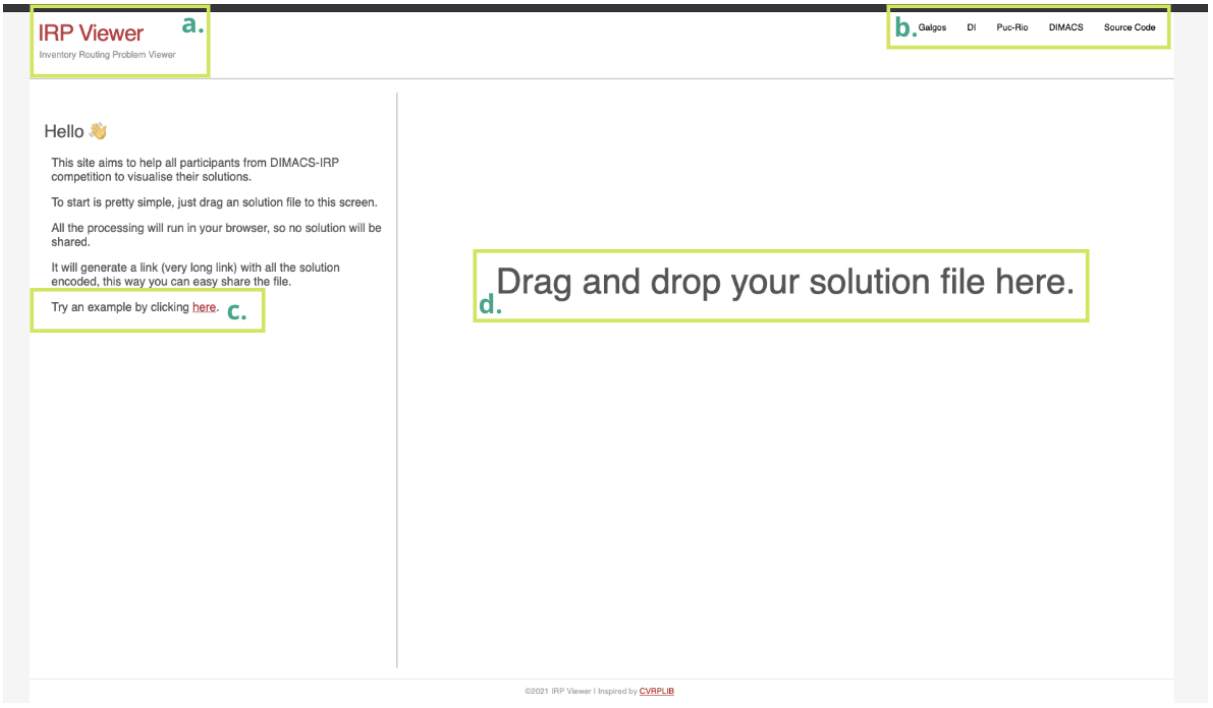
6. Documentação para usuário

Nesta seção será apresentado um passo a passo de como utilizar o sistema IRP-Viewer

Tela Inicial

Abaixo é apresentado a tela inicial do sistema IRP-Viewer.

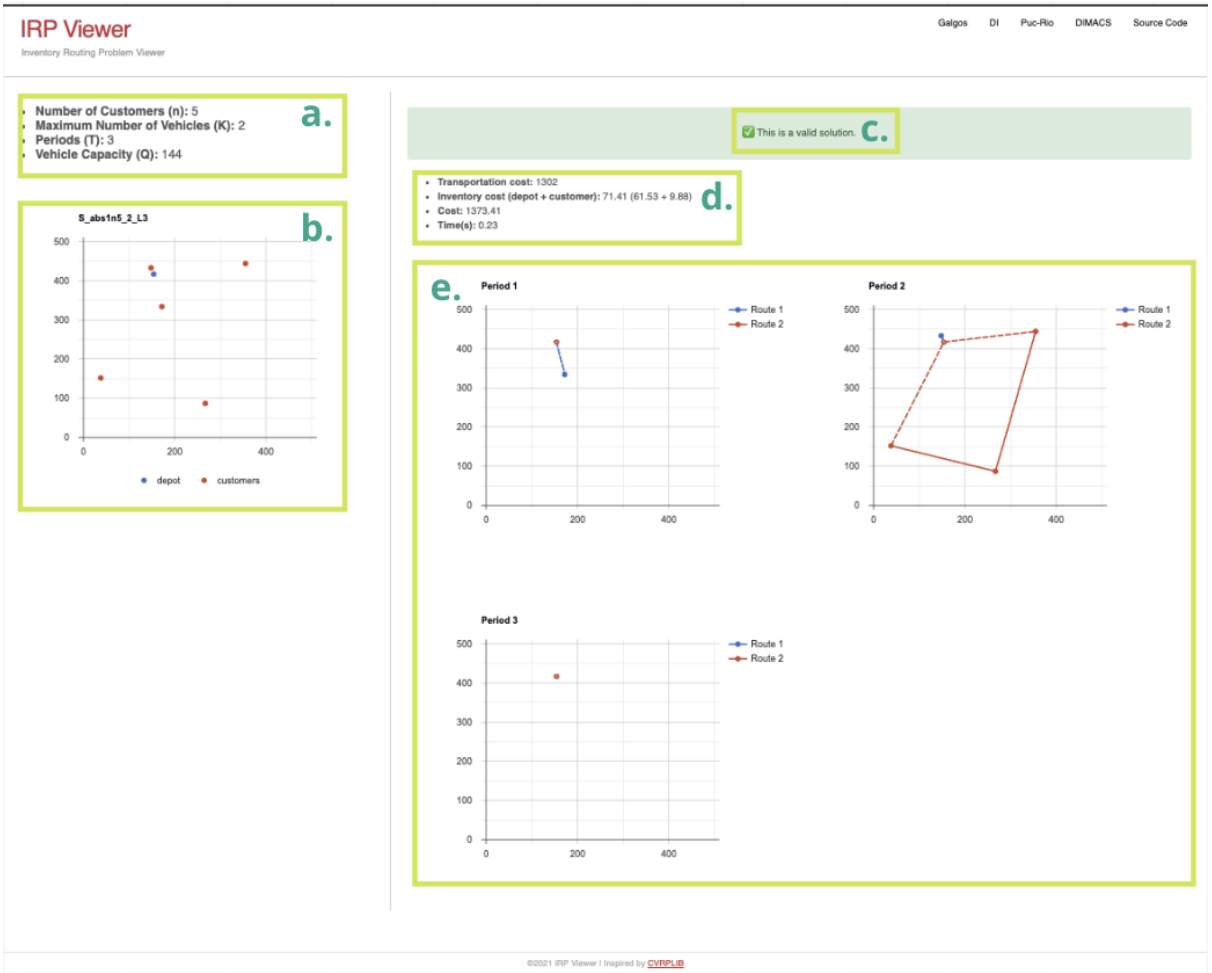
- Sempre que clicar em **a.**, será redirecionado a esta tela.
- No canto superior direito **b.** pode-se verificar links importantes relacionados ao projeto.
- Para iniciar a utilização, basta arrastar um arquivo para qualquer área da tela.
- Caso queira verificar um exemplo de utilização, basta clicar em **c.**.



Tela de visualização de solução (solução válida)

Ao arrastar um arquivo de solução, a tela será alterada para a seguinte disposição:

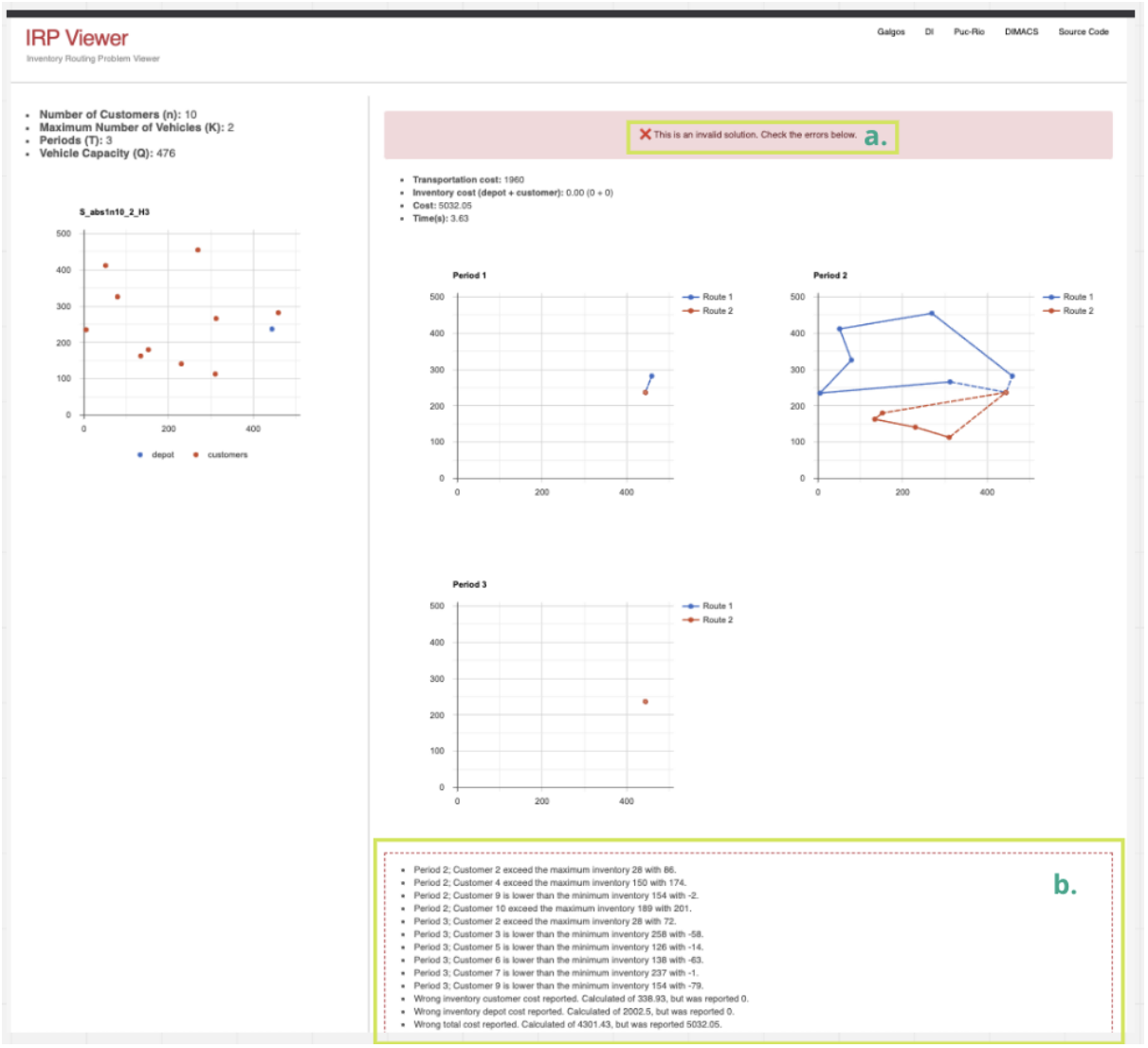
- Em **a.** são apresentados as informações da instância do problema;
- Em **b.** é apresentado o desenho da instância do problema. O círculo azul demonstra um depot, e círculos vermelhos os clientes;
- Em **c.** é apresentado a confirmação de que a solução é válida;
- Em **d.** são apresentados as informações da solução enviada;
- Em **e.** são apresentados para cada período, um desenho com todas as rotas e suas entregas.



Tela de visualização de solução (solução inválida)

Caso uma solução inválida seja enviada, a tela será carregada com a mesmas informações que a tela de visualização de solução válida, porém com os seguintes detalhes

- Em **a.** é apresentado a informação de que a solução é inválida;
- Em **b.** são apresentados cada erro encontrado na solução enviada;



7. Artefatos

Conforme definido anteriormente, segue a lista dos artefatos solicitados:

- **ART-1:** https://github.com/brunoguic/irp-viewer
- **ART-2:** <project-root-dir>/Dockerfile
- **ART-3:** <project-root-dir>/Readme
- **ART-4:** <project-root-dir>/documentation/test-output.txt