

Escola de Engenharia

Departamento de Informática

Universidade do Minho

Licenciatura em Engenharia Informática

Projecto de Laboratórios de Informática III

“Transistários LEI — Projecto 2: Parte II”

Bruno Ferreira, A61055
Daniel Carvalho, A61008

Grupo 42

Braga, Maio de 2012

Conteúdo

1	Resumo	3
2	Introdução	4
3	Conteúdo	6
3.1	Tomada de Decisões: Estruturas de Dados	6
3.1.1	Utilizadores	7
3.1.2	Localidades	7
3.2	Interface gráfica	8
3.3	Camada de Persistência . .	17
3.3.1	Streams de Objectos	18
3.3.2	Streams de Texto . .	22
3.3.3	Comparação	26
4	Conclusão	29
5	Fotos	31

Lista de Figuras

1	Utilizadores importados . .	10
2	Utilizadores: Pesquisa . . .	11
3	Utilizadores: Listar Todos .	12
4	Utilizadores: Eliminar . . .	13
5	Nova Ligação	14
6	Listar Ligações	15
7	Calcular Caminho mais Curto	16
8	Streams de Objectos (escrever)	20
9	Streams de Objectos (ler) .	21
10	Streams de Texto (escrever)	24
11	Streams de Texto (ler) . . .	25
12	Comparação (escrever) . . .	27
13	Comparação (ler)	28

1 Resumo

Neste relatório encontram-se explicitadas de forma detalhada as decisões e implementações feitas neste projecto tendo em conta as análises de performance das estruturas de dados do *Java Collections Framework* feita na etapa anterior. É também abordada a construção de uma interface gráfica em *Swing* de forma a ser possível efectuar mudanças no estado da aplicação por parte do utilizador e garantir a usabilidade e acessibilidade na sua utilização.

Relativamente à aplicação em si, esta tem de garantir o manuseio de estruturas para o armazenamento de dados relativamente a utilizadores do serviço de transportes e das localidades onde existe possibilidade de efectuar serviço, a escolha da estrutura a usar recaiu sobre o *HashMap*.

No seguimento deste relatório é explicado o porquê dessa escolha assim como in-

formações da implementação do código com a interface gráfica e as opções tomadas relativamente a ambas, assim como outras decisões relevantes para o funcionamento da aplicação.

2 Introdução

O objectivo desta etapa é a construção de uma camada interactiva para a manipulação da estrutura dos utilizadores e de localidades e ligações. Torna-se então necessário desenvolver uma interface gráfica dotada de acessibilidade e opções de utilização satisfatórias de acordo com os princípios base do padrão arquitectural MVC (Model-View-Controller). Dessa forma é pretendido que:

1. seja feita uma escolha mediante as estruturas de dados estudadas na primeira parte do projecto. A restante implementação será desenvolvida com base nessa estrutura de dados.
2. se desenvolva a interface com o utilizador, em Java/Swing, que permita ter acesso às seguintes operações:
 - carregar base de dados de utilizadores, localidades e ligações a partir de um ficheiro
 - inserir o registo de um novo utilizador;
 - procurar um utilizador por nome;
 - procurar um utilizador por nif;
 - apagar um registo por nome;
 - apagar um registo por nif;
 - listar todos os utilizador por palavra chave (ex: primeiro nome);
 - inserir uma localidade e respectivas ligações directas;
 - criar uma nova ligação entre duas localidades;
 - visualizar as ligações de uma localidade;

- determinar a quantas localidades de distância está determinada localidade.

Pretende-se também que seja feito um estudo ao desempenho de duas soluções distintas para assegurar persistência em Java: streams de texto e streams de objectos. Dessa forma, o programa deve medir os tempos para escrita e leitura (sempre dos patamares 5000, 10000, 15000 e 18000 itens) utilizando as seguintes configurações:

1. streams de texto, para leitura (BufferedReader) e para escrita (PrintWriter)
2. streams de objecto, para leitura (ObjectInputStream) e para escrita (ObjectOutputStream)

3 Conteúdo

3.1 Tomada de Decisões: Estruturas de Dados

De forma a chegar a uma tomada de decisão sobre a melhor configuração em termos de estruturas de dados a usar na aplicação recorreu-se aos resultados obtidos na etapa anterior que reflectiam a performance das implementações das mesmas para a utilização e execução de operações sobre utilizadores e localidades. Abaixo seguem as configurações de estruturas de dados testadas. Configurações para localidades:

- uma estrutura baseada em ArrayList para fazer a gestão das localidades e um ArrayList para as localidades relacionadas;

- uma estrutura baseada em ArrayList para fazer a gestão das localidades e um HashSet para as localidades relacionadas;
- uma estrutura baseada em HashMap para localidades com um HashMap para as localidades relacionadas;
- uma estrutura baseada em TreeMap para localidades com um TreeMap para as localidades relacionadas.

Configurações para utilizadores:

- duas estruturas baseadas em ArrayList para fazer a ordenação pelos dois critérios: nome e nif;
- uma estrutura baseada em ArrayList para fazer a ordenação por nome e uma estrutura auxiliar em LinkedList para a ordenação por nif;

- duas estruturas baseadas em HashMap, para guardar dados ordenados dos utilizadores;
- duas estruturas baseadas em TreeMap, para guardar dados ordenados dos utilizadores;

3.1.1 Utilizadores

Na etapa anterior deste projecto, chegou-se à conclusão, através de análises à performance das estruturas de dados do Java Collections Framework, que a melhor forma de implementar a base de dados de utilizadores na aplicação seria feita recorrendo a duas estruturas HashMap, uma usando o nome do utilizador como chave e a outra usando o NIF. A vantagem do HashMap relativamente a outras configurações nota-se sobretudo em termos de inserção e procura. Tanto o ArrayList como o Tre-

eMap tornam-se soluções menos óptimos, um pela sua falta de performance e a outra pela complexidade de inserção/remoção.

3.1.2 Localidades

De forma equivalente à decisão tomada sobre a estrutura para utilizadores, a escolha para a manipulação de dados relativos a localidades recai também sobre duas estruturas HashMap, sendo uma delas para localidades e a outra para as suas ligações. Isto deve-se igualmente à rapidez apresentada por esta estrutura na implementação sobre este tipo concreto de dados.

3.2 Interface gráfica

A interface gráfica foi feita para ser concisa e simples de utilizar.

Descrevem-se agora algumas funcionalidades da interface gráfica.

- Existe um menu com opções que permitem guardar e carregar o estado da aplicação ou importar dados de um ficheiro.
- As várias operações sobre utilizadores ou localidades estão distribuídas por separadores. No primeiro separador, *Gerir Utilizadores*, pode-se adicionar novos utilizadores, listar utilizadores existentes e remover utilizadores. No segundo separador, *Adicionar Localidades*, pode-se adicionar localidades e ligações. No terceiro separador, *Listar Ligações*, apenas de listam as ligações da localidade seleccionada. No quarto e último separador, *Calcular Caminhos*, calcula-se o caminho mais curto entre duas localidades.
- Todas as caixas de texto de pesquisa de utilizadores ou localidades funcionam num esquema de pesquisa instantânea, isto é, assim que o utilizador começa a escrever o nome de um utilizador ou localidade, os resultados começam a aparecer.
- As caixas de texto de pesquisa de utilizadores ou localidades são acompanhadas de um botão que lista todos os itens.
- Na tabela de utilizadores, existe um botão que permite seleccionar todos os utilizadores e um botão que apaga os utilizadores seleccionados. Antes de apagar utilizadores, é pedido ao utilizador que confirme que quer remover

os utilizadores seleccionados.

- No separador *Adicionar Localidades*, para adicionar uma nova localidade, pode-se carregar na tecla ENTER em vez de carregar no botão *Adicionar Localidade*.

Seguem-se algumas imagens ilustrativas dos pontos acima, assim como do resto da interface gráfica.

Figura 1: Utilizadores importados

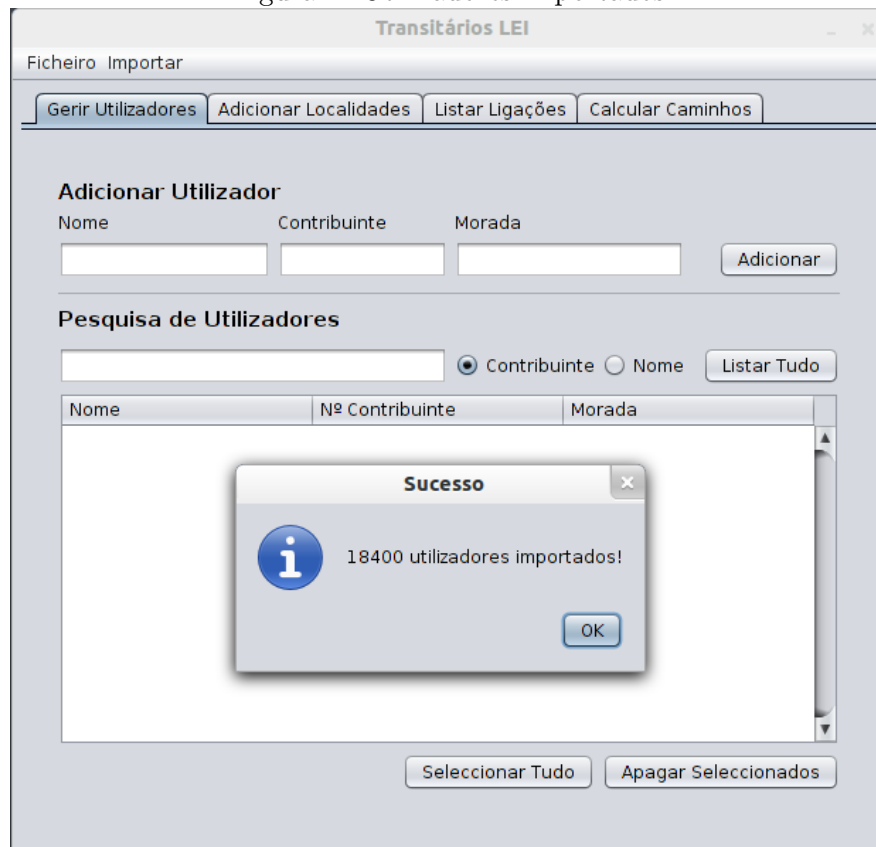
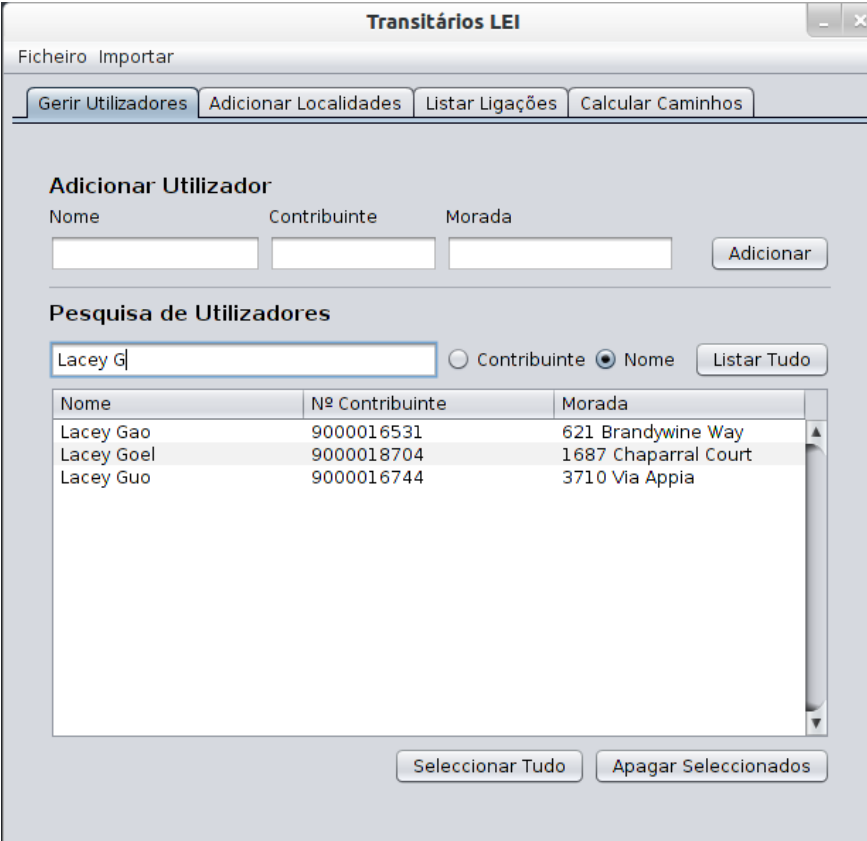


Figura 2: Pesquisa de Utilizadores



Transitários LEI

Ficheiro Importar

Gerir Utilizadores Adicionar Localidades Listar Ligações Calcular Caminhos

Adicionar Utilizador

Nome Contribuinte Morada

Adicionar

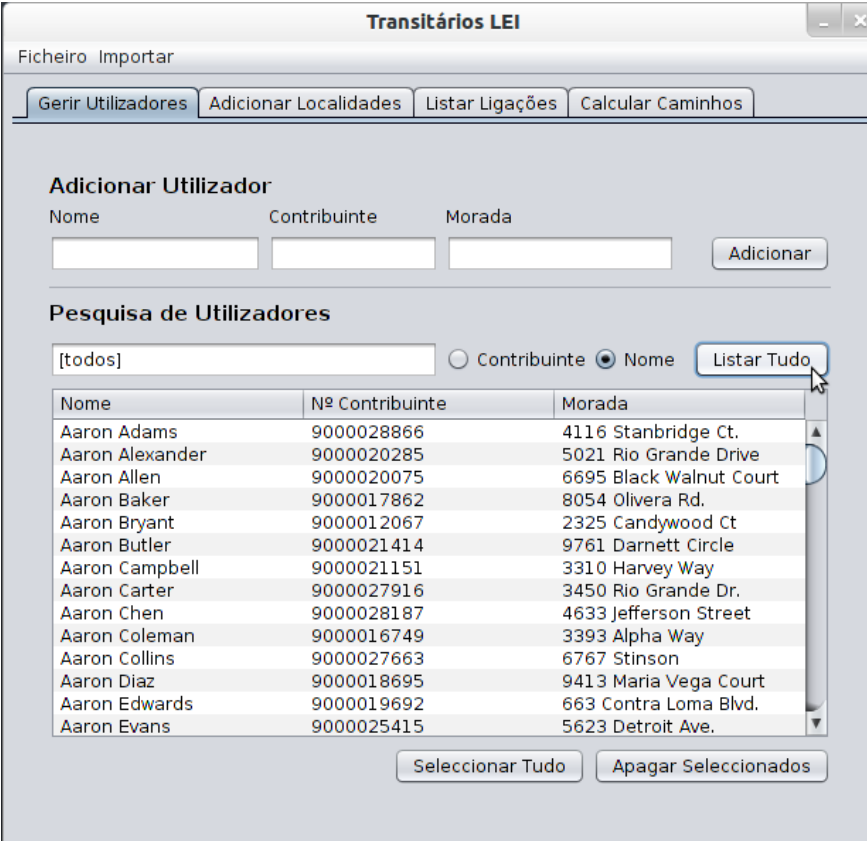
Pesquisa de Utilizadores

Lacey G ☐ Contribuinte ☒ Nome Listar Tudo

Nome	Nº Contribuinte	Morada
Lacey Gao	9000016531	621 Brandywine Way
Lacey Goel	9000018704	1687 Chaparral Court
Lacey Guo	9000016744	3710 Via Appia

Seleccionar Tudo Apagar Seleccionados

Figura 3: Listar todos os Utilizadores



The screenshot shows a software window titled "Transitários LEI". At the top, there is a menu bar with "Ficheiro" and "Importar". Below the menu bar is a tabbed interface with four tabs: "Gerir Utilizadores" (selected), "Adicionar Localidades", "Listar Ligações", and "Calcular Caminhos".

Under the "Gerir Utilizadores" tab, there is a section titled "Adicionar Utilizador" with three input fields labeled "Nome", "Contribuinte", and "Morada", followed by an "Adicionar" button.

Below this is a section titled "Pesquisa de Utilizadores". It contains a search input field with the text "[todos]", two radio buttons labeled "Contribuinte" and "Nome" (with "Nome" selected), and a "Listar Tudo" button. A mouse cursor is shown clicking the "Listar Tudo" button.

Below the search section is a table with three columns: "Nome", "Nº Contribuinte", and "Morada". The table lists 18 users, all with the name "Aaron".

Nome	Nº Contribuinte	Morada
Aaron Adams	9000028866	4116 Stanbridge Ct.
Aaron Alexander	9000020285	5021 Rio Grande Drive
Aaron Allen	9000020075	6695 Black Walnut Court
Aaron Baker	9000017862	8054 Olivera Rd.
Aaron Bryant	9000012067	2325 Candywood Ct
Aaron Butler	9000021414	9761 Darnett Circle
Aaron Campbell	9000021151	3310 Harvey Way
Aaron Carter	9000027916	3450 Rio Grande Dr.
Aaron Chen	9000028187	4633 Jefferson Street
Aaron Coleman	9000016749	3393 Alpha Way
Aaron Collins	9000027663	6767 Stinson
Aaron Diaz	9000018695	9413 Maria Vega Court
Aaron Edwards	9000019692	663 Contra Loma Blvd.
Aaron Evans	9000025415	5623 Detroit Ave.

At the bottom of the window, there are two buttons: "Seleccionar Tudo" and "Apagar Seleccionados".

Figura 4: Remover Utilizadores

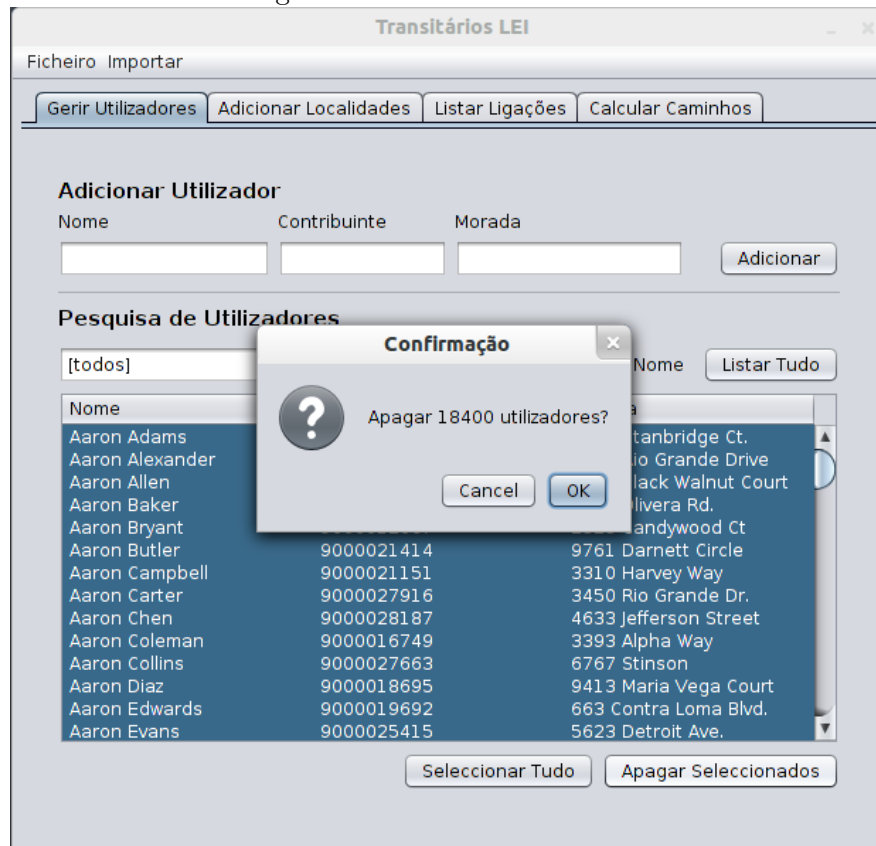


Figura 5: Adicionar nova ligação

Transitários LEI

Ficheiro Importar

Gerir Utilizadores Adicionar Localidades Listar Ligações Calcular Caminhos

Adicionar Localidade


Adicionar Localidade

Adicionar Ligações a Localidades Listar Todas

Pesquisar

Origem	Destino
Barranco das Taipas	Bairro Silva Braga
Caldas das Taipas	Braga
Cerro das Taipas	Braga Airport
Taipa	Braga Mediumwave Transmitter
Taipadas	Bragado
Taipas	
Taipinhas	
Vale de Taipas	

Sucesso

 Ligação criada!

OK

Distância Custo Adicionar Ligação

Figura 6: Listar Ligações de Localidades

Transitários LEI

Ficheiro Importar

Gerir Utilizadores Adicionar Localidades **Listar Ligações** Calcular Caminhos

Pesquisar

Tai

Listar Todas as Origens

Origem

Destinos (ligações directas)

Nome	Distância	Custo
Alcaria do P...	500.0	37.0
Braga	85.0	60.0
Cerro do Se...	386.0	28.0
Lagoa dos ...	803.0	60.0
Monte das ...	742.0	55.0
Ribeira da P...	733.0	54.0
Ribeiro da A...	46.0	3.0
Ribeiro do V...	758.0	56.0
Sorvel Cimeiro	620.0	46.0
Vale de Asn...	542.0	40.0

Origem list:

- Barranco das Taipas
- Barroças e Taiais
- Caldas das Taipas
- Cerro das Taipas
- Ribeira das Tainhas
- Taide
- Taipa**
- Taipadas
- Taipas
- Taipinhas
- Taião
- Vale de Taipa

Figura 7: Calcular o caminho mais curto entre duas localidades

The screenshot shows a software window titled "Transitários LEI". At the top, there is a menu bar with "Ficheiro" and "Importar". Below the menu bar is a tabbed interface with four tabs: "Gerir Utilizadores", "Adicionar Localidades", "Listar Ligações", and "Calcular Caminhos". The "Calcular Caminhos" tab is active.

Inside the "Calcular Caminhos" tab, there are two main sections. The left section is titled "Origem" and has a button "Listar Todas as Origens" above it. It contains a text input field with "Taipa" and a list of locations: "Barranco das Taipas", "Caldas das Taipas", "Cerro das Taipas", "Taipa" (highlighted), "Taipadas", "Taipas", "Taipinhas", and "Vale de Taipa". The right section is titled "Destino" and has a button "Listar Todos os Destinos" above it. It contains a text input field with "Corte P" and a list of locations: "Corte Paral" (highlighted), "Corte Pequena", "Corte Pereiro", "Corte Pinheiro", and "Ribeira da Corte Pequena".

At the bottom of the window, there is a button "Calcular Caminho Mais Curto" and the text "2 localidades de distância".

3.3 Camada de Persistência

Relativamente à camada de persistência, foi pedido que se estudasse o desempenho de duas soluções distintas: streams de texto e streams de objectos.

Assegurar a persistência com steams de objectos consiste em implementar Serializável em todas as classes cujos objectos irão ser guardados. Por outro lado, para assegurar persistência usando streams de texto é necessário formatar toda a informação sobre a forma de Strings, para que essa mesma informação possa ser guardada e recuperada posteriormente consoante determinado formato.

De forma a uniformizar os resultados, todos os testes de desempenho foram efectuados na mesma máquina. As especificações técnicas dessa máquina são as seguintes:

Sistema Operativo:	Ubuntu 12.04
Modelo:	Asus G1S
Processador:	Intel Centrino T7700 Core2Duo @ 2.4GHz
Cache:	2x4096Kb
RAM:	2x1024MB DDR2 667MHz

Para obter resultados consistentes, foram feitas 100 escritas e 100 leituras para cada um dos patamares pedidos em ambos os tipos de ficheiro.

Organizaram-se os resultados de escrever/ler de ambos os formatos em tabelas e gráficos. Por norma, todos os tempos se encontram em milisegundos.

De notar também que todos os gráficos obtidos aproximam a norma linear.

3.3.1 Streams de Objectos

As streams de objectos habilitam a persistência de dados com alterações mínimas no código da aplicação, mas, em contrapartida, guardam uma boa quantidade de informações relativa às classes, e não aos seus dados. Isto permite que sejam feitas uma data de verificações de consistência no momento da importação, mas também torna o processo mais demorado e o ficheiro maior. Nos testes, o ficheiro que continha 18000 utilizadores e 18000 localidades, cada uma com 8 ligações, ocupava 9.3MB.

Tabela 1: Tempos da utilização de streams de objectos

Num. Dados	escrever em ficheiro		ler de ficheiro	
	μ	σ	μ	σ
5000	1182.88	122.67	398.6	43.55
10000	3347.88	233.58	1026.92	92.12
15000	6560.54	356.50	1976.44	234.84
18000	9128.06	428.50	2674.32	357.76

Figura 8: Streams de Objectos (escrever)

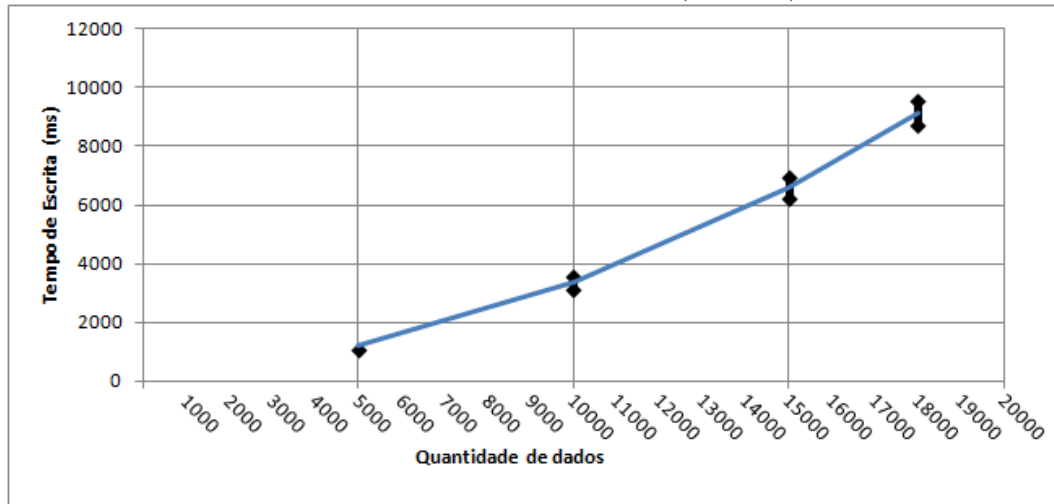
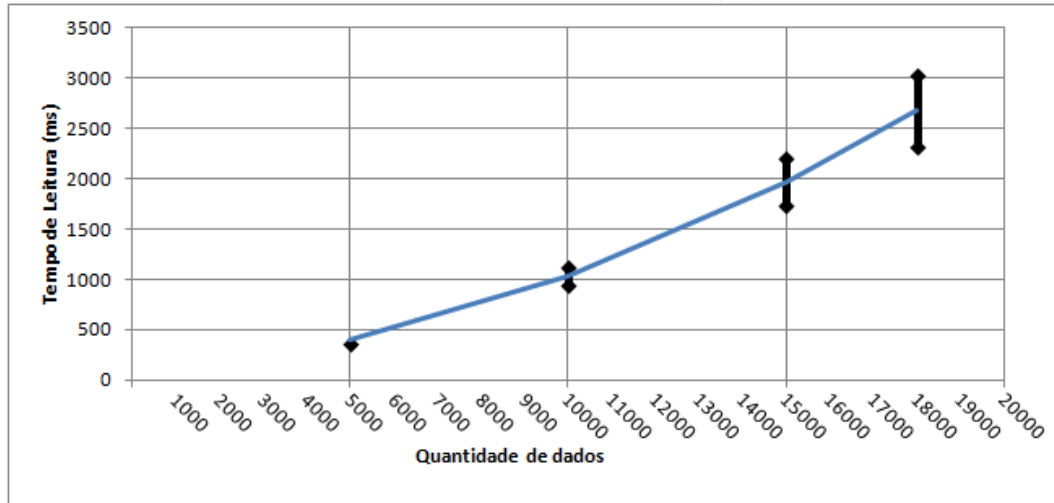


Figura 9: Streams de Objectos (ler)



3.3.2 Streams de Texto

As streams de texto obrigam a que o código seja modificado de forma a guardar a informação formatada que permite a sua posterior recuperação. Na recuperação dessa informação apenas podem ser feitas verificações básicas, como a verificação de tipos de dados numéricos. Isto obriga a que, na inserção, todos os dados sejam introduzidos novamente no sistema. Nos testes, o ficheiro que continha 18000 utilizadores e 18000 localidades, cada uma com 8 ligações, ocupava 7.5MB.

Tabela 2: Tempos da utilização de streams de texto

Num. Dados	escrever em ficheiro		ler de ficheiro	
	μ	σ	μ	σ
5000	31.46	28.47	27.2	10.89
10000	91.74	20.38	91.32	20.18
15000	181.16	21.31	270.72	24.60
18000	290.7	36.56	274.06	44.35

Figura 10: Streams de Texto (escrever)

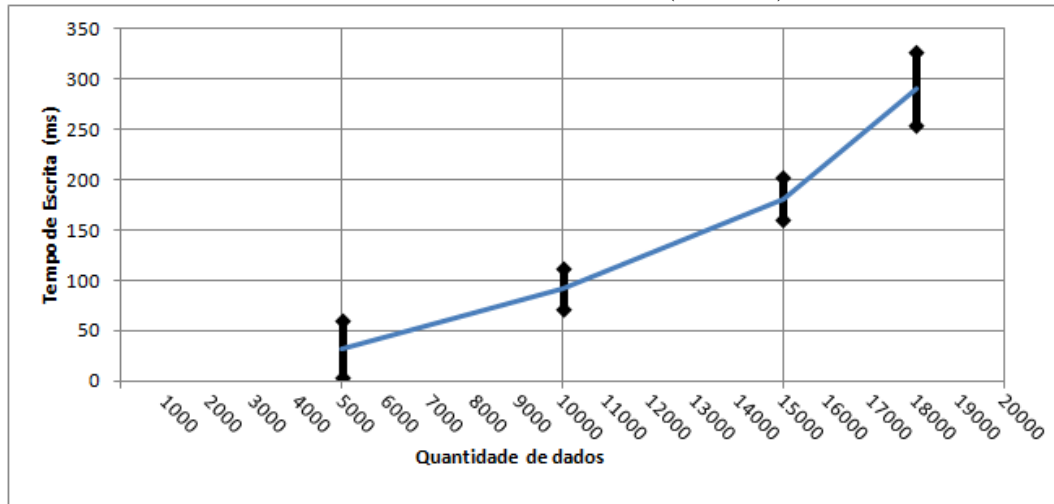
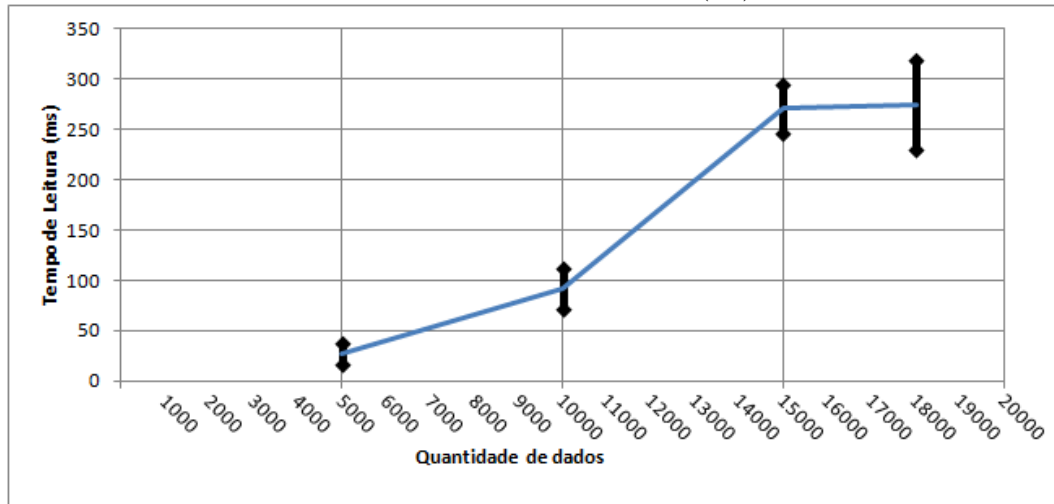


Figura 11: Streams de Texto (ler)



3.3.3 Comparação

De ambas as configurações testadas, usar streams de texto para assegurar persistência é a mais rápida, como se pode verificar nos gráficos abaixo.

Figura 12: Comparação (escrever)

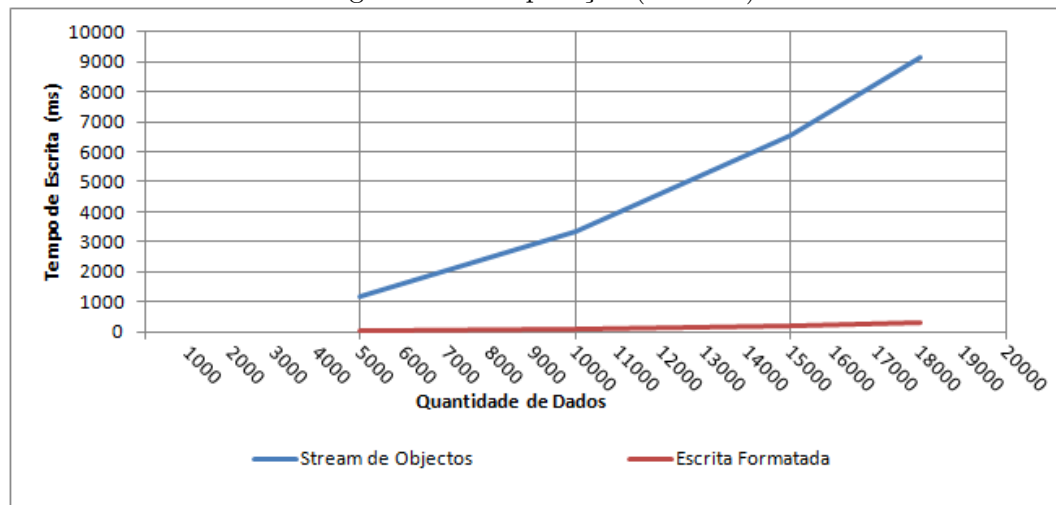
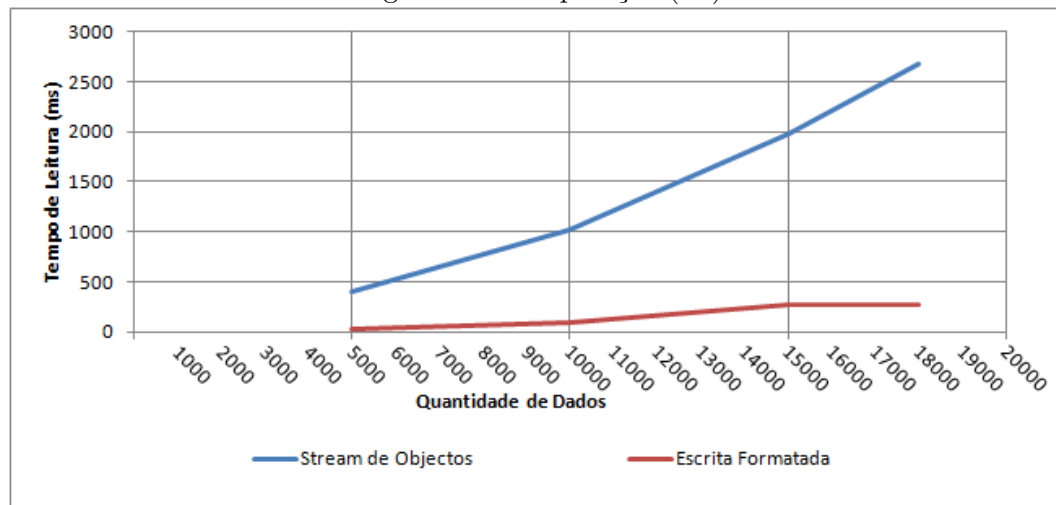


Figura 13: Comparação (ler)



4 Conclusão

Depois da criação desta aplicação aprendemos a manusear a biblioteca Swing do java. Também concluimos pelos testes de persistência de dados que fizemos, que utilizar a serialização fornecida pelo java será, na grande maioria das vezes, mais lento que assegurar persistência usando streams de texto.

Com a chegada do fim deste relatório chegou a altura de se tecerem as conclusões finais. A necessidade de criação de uma interface gráfica para a aplicação foi um novo desafio, pois esta apresenta-se como a primeira oportunidade para realizar algo no que diz respeito à programação gráfica. O Swing, ferramenta primária para criação de interfaces gráficas (GUI) do Java, mostrou ser uma ferramenta de uso bastante completo, tornando necessário durante o seu uso a recorrência a pesquisas. Esta ferra-

menta tornou possível a criação de uma interface de uso intuitivo e completo. Tendo isso em conta pensamos ter atingido as metas deste projecto no que diz respeito ao desenvolvimento da interface gráfica.

Em termos de implementação estrutural de dados da aplicação, o Java é uma linguagem que, por ter já bibliotecas muito completas, facilita bastante a tarefa. O uso de estruturas de dados do Java Collection Framework, nomeadamente o uso de Hash-Map, foi bastante útil e mostrou ser a melhor forma de garantir o manuseamento da informação pedida no enunciado.

Relativamente à camada de persistência, foi-nos possível chegar à conclusão que as streams de objectos são bastante ineficientes comparativamente às streams de texto, maioritariamente pelo facto das streams de objectos guardarem dados relativos às classes que depois invocam verificações de

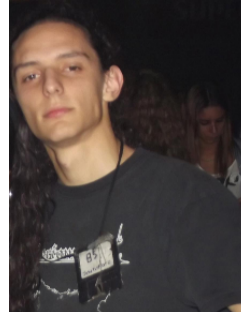
consistência aquando da importação da informação.

Em termos gerais, consideramos que a aplicação que por nós foi construída cumpre todos os objectivos propostos para este projecto.

5 Fotos



Bruno Ferreira
A61055



Daniel Carvalho
A61008