Logotipo

Descrição gerada automaticamente 

Licenciatura em Engenharia Informática (BLEI)

Unidade Curricular de Computação Móvel

3.º Ano

**TG1 (Trabalho de Grupo 1)**

Docente Responsável

Prof. Dr. Célio Carvalho

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Trabalho elaborado por:

**João Madail e Joaquim Ferreira**

Vila Nova de Gaia, 4 de Dezembro de 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Índice**

[Índice de Figuras 4](#_Toc121075099)

[Introdução 5](#_Toc121075100)

[Descrição do Problema e Sua Resolução 6](#_Toc121075101)

[Exercício 1 6](#_Toc121075102)

[Exercício 2 7](#_Toc121075103)

[Exercício 3 9](#_Toc121075104)

[Conclusões 15](#_Toc121075105)

[Ferramentas 16](#_Toc121075106)

[Bibliografia 16](#_Toc121075107)

# Índice de Figuras

[Figura 1 - Print Ex1 6](#_Toc121073094)

[Figura 2 - Output Ex1 6](#_Toc121073095)

[Figura 3 - Menu ex2 7](#_Toc121073096)

[Figura 4 - ex2, caso 1 7](#_Toc121073097)

[Figura 5 - ex2, caso 2 8](#_Toc121073098)

[Figura 6 - ex2, caso 3 8](#_Toc121073099)

[Figura 7 - ex2, caso 4 8](#_Toc121073100)

[Figura 8 - Menu ex3 9](#_Toc121073101)

[Figura 9 - Organização dos ficheiros 10](file:///C:\Users\joaom\Desktop\RelatorioCMO.docx#_Toc121073102)

[Figura 10 - Dashboard 11](#_Toc121073103)

[Figura 11 - Relatório de Apólices 11](#_Toc121073104)

[Figura 12 - Relatório de Entidades 12](#_Toc121073105)

[Figura 13 - Menu de Manutenção 12](#_Toc121073106)

[Figura 14 - Manutenção de Entidades 12](#_Toc121073107)

[Figura 15 - Diagrama de Classes 13](#_Toc121073108)

[Figura 16 - Diagrama de Casos de Uso 14](#_Toc121073109)

# Introdução

O presente relatório, descreve as tarefas desenvolvidas ao longo da resolução dos 3 exercícios propostos para o Trabalho de Grupo 1 (TG1).

TG1 teve como principais objetivos consolidar os conhecimentos obtidos nas aulas relacionados com Programação Orientada a Objetos (POO), praticar regras de semântica da linguagem de programação Dart, explorando as suas funcionalidades e relembrar conceitos de algoritmia, pretendendo assim, também promover competências autodidatas.

A par disto, de modo a facilitar o trabalho em grupo, criou-se e utilizou-se um repositório git

# Descrição do Problema e Sua Resolução

## Exercício 1

O primeiro exercício consiste no seguinte Enunciado:

“Proponha um programa que teste os números inteiros do intervalo ]100; 200[ e que apresente na consola a soma e média dos primos existentes naquele intervalo.”

Após uma breve análise do enunciado, conclui-se que o mesmo seria resolvido da seguinte forma. Criar um método para testar se um número é primo, chamado isPrime, e invoca-lo dentro de um ciclo for. Esse ciclo tem como condição de repetição, enquanto a variável ‘i’ for menor igual a 200, o ciclo repete e adiciona mais um ao valor de ‘i’, sendo que o valor inicial de ‘i’ é 100. Por cada repetição do ciclo, método isPrime é invocado para testar a variável ‘i’ se o valor do mesmo é um número primo, caso o mesmo se confirme, ‘i’ é guardada num vetor chamado primos. Para cada repetição é também somado a uma variável chamada ‘soma’ de valor inicial 0 o valor de ‘i’.

Assim, apresentamos o valor da soma dos números primos e a média dos mesmos, fora do ciclo ‘for’, recorrendo a um ‘print’ da variável ‘soma’ e da divisão do valor de ‘soma’ pelo tamanho (length) do vetor ‘primos’ respetivamente.



Figura - Print Ex1

Os resultados obtidos foram os seguintes:

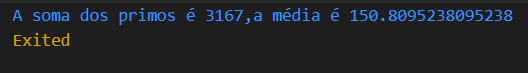


Figura - Output Ex1

## Exercício 2

O segundo exercício consiste no seguinte enunciado:

“Desenvolva um programa capaz de gerar de forma automática e aleatória números inteiros

positivos, em que a quantidade de números a gerar seja sorteada entre [90; 150].

Disponibilize um menu simples com opções para extrair a seguinte informação do conjunto

gerado:

a. O maior e menor elemento do conjunto;

b. A amplitude do conjunto;

c. Uma listagem dos números ímpares com ordenação decrescente;

d. Visualização do conjunto;”

A resolução do segundo exercício passou pela criação de um menu usando ‘switch case’ com quatro casos possíveis como é possível verificar na imagem abaixo.

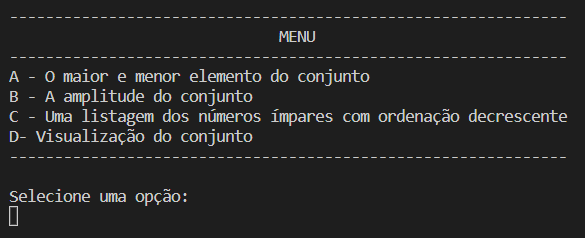


Figura - Menu ex2

No primeiro caso teve como resolução os seguintes procedimentos:

É gerado um numero entre 90 e 150 que, de seguida, é usado como a quantidade de números gerados aleatoriamente que populam a lista 'numeros'.

Depois, a lista é ordenada crescentemente. Utilizando o número de posição 0 obtemos o número mais pequeno, o maior é obtido a partir do número resultante da subtração do tamanho da lista por 1 (numeros.length - 1).

Para finalizar, recorremos a um ‘print’ para apresentar o menor número e o maior como é possível verificar na imagem abaixo

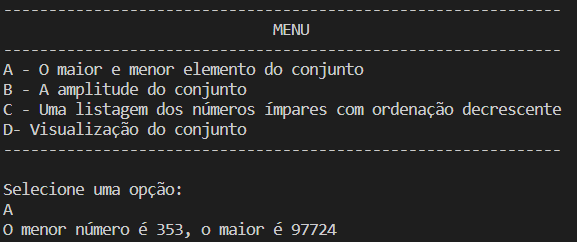


Figura - ex2, caso 1

Recorrendo aos procedimentos anterior, para a resolução do segundo caso bastou recorrer a um ‘print’ para apresentar a subtração do maior número pelo menor.

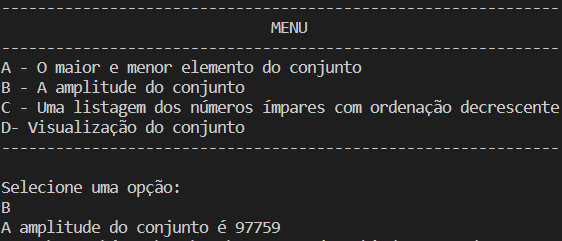


Figura - ex2, caso 2

De seguida, para resolver o terceiro caso, bastou verificar e separar os números pares dos ímpares da lista ‘numeros’ e separa-los para lista diferentes. Logo depois, a lista dos números ímpares foi ordenada decrescentemente e, recorrendo a um ‘print’, apresentamos os resultados como é possível verificar, mais uma vez, na imagem abaixo.

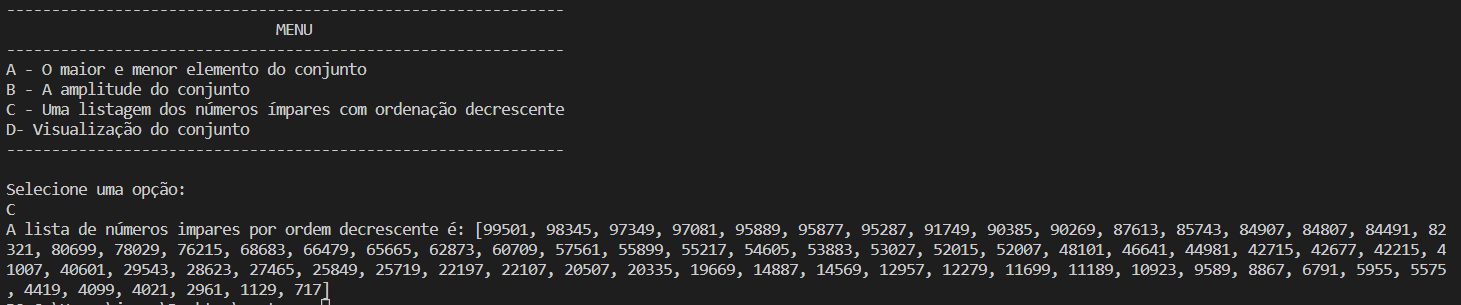


Figura - ex2, caso 3

Para finalizar o exercício, bastou recorrer a um print para apresentar a lista ‘numeros’ que previamente foi ordenada por ordem crescente, sendo que os resultados obtidos estão na imagem abaixo.

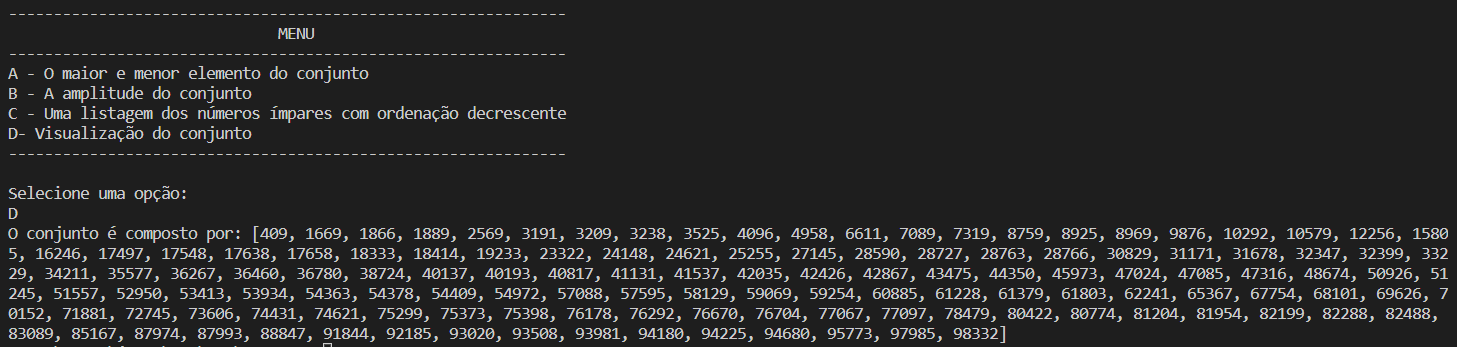


Figura - ex2, caso 4

## Exercício 3

O terceiro exercício consiste no seguinte enunciado:

“O mediador de seguros “Surpresas existem!” necessita de uma aplicação para gerir a sua

carteira de seguros. De entre os dados a guardar de cada apólice salienta-se a seguradora,

tomador, segurado, tipo de seguro (e.g. vida, acidentes de trabalho, etc.) valor segurado,

valor do prémio a pagar mensal ou anualmente, etc.

Desenvolva uma aplicação que ajude o mediador a guardar a informação da sua atividade.

Deve disponibilizar as operações habituais para manutenção de dados também para as

tabelas auxiliares (e.g. seguradoras). Não esqueça de garantir a integridade das referências

(e.g. uma seguradora não pode ser removida se existirem apólices relacionadas). Além das

funcionalidades necessárias (criatividade), apontam-se os seguintes requisitos como

mínimos da aplicação:

• Apresentar um menu com as diferentes funcionalidades da aplicação, que

apresente uma pequena dashboard com a seguinte informação: quantidade de

apólices ativas e inativas; quantidade de apólices e valor médio segurado por

seguradora (apenas apólices ativas); quantidade de apólices e valor médio segurado

por tipo de seguro (apenas apólices ativas);

• Relatório de apólices ativas por tipo de seguro e seguradora, que indique o valor de

cada prémio. Como o objetivo é analisar o valor anual de prémios, no final do

relatório deve surgir o somatório da coluna dos prémios das apólices.

• Relatório de entidades (i.e. tomadores e segurados) com apólices ativas, que mostre

a idade e a morada de cada um”

Na resolução do terceiro exercício para alem do pretendido, incorporou-se ao máximo os conteúdos lecionados nas aulas com pesquisa externa. Como é possível identificar comparando as imagens do terminal do exercício anterior com o presente, utilizou-se a biblioteca ‘dart\_console’ para abrilhantar o menu criado também com o ‘switch case’ utilizado no exercício anterior. Como demonstrou ser um exercício maior que os anteriores, houve uma especial atenção às convenções e à reutilização de código tentando sempre reduzir código.

Criaram-se classes para as apólices, entidades, seguradores e para o tipo de seguro, listas para essas mesmas classes onde guardamos toda a informação que poderia ser alterada, eliminada recorrendo a métodos de manutenção de tabelas criados.

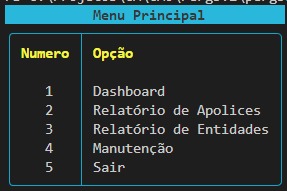


Figura - Menu ex3

Dado o volume de ficheiros, certificou-se que os mesmos se mantiveram organizados.

O ficheiro ‘global\_variables’ foi criado para reduzir o número de ‘imports’ em cada ficheiro, sendo que o mesmo tinha todos os ‘imports’ imperativos para os restantes ficheiros.

Na pasta data. Na pasta manutencao, os procedimentos usados a manutenção das tabelas.

A pasta models tem os ficheiros das classes dos objetos de cada tabela, na pasta data, as classes das listas desses mesmo objetos.

Na pasta relatórios, o tratamento de dados presentes nas listas.



Figura - Organização dos ficheiros

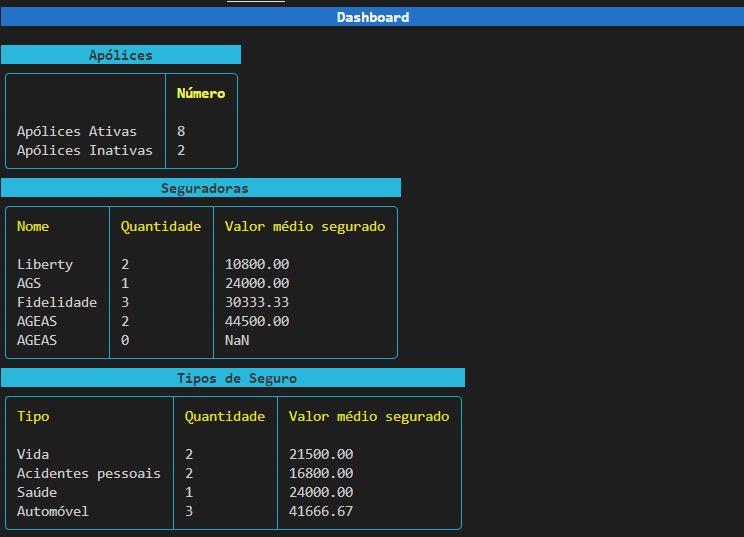


Figura - Dashboard

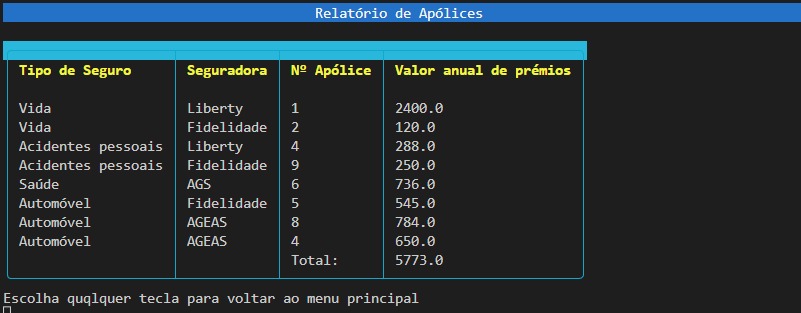


Figura - Relatório de Apólices



Figura - Relatório de Entidades

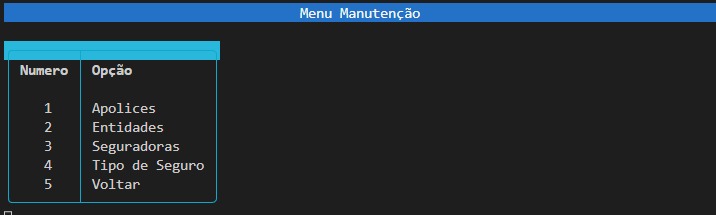


Figura - Menu de Manutenção



Figura - Manutenção de Entidades

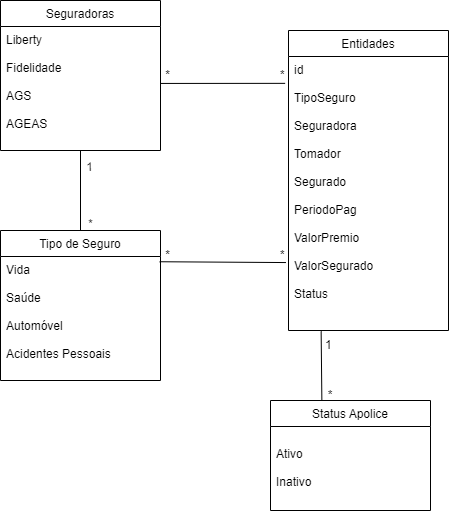


Figura - Diagrama de Classes

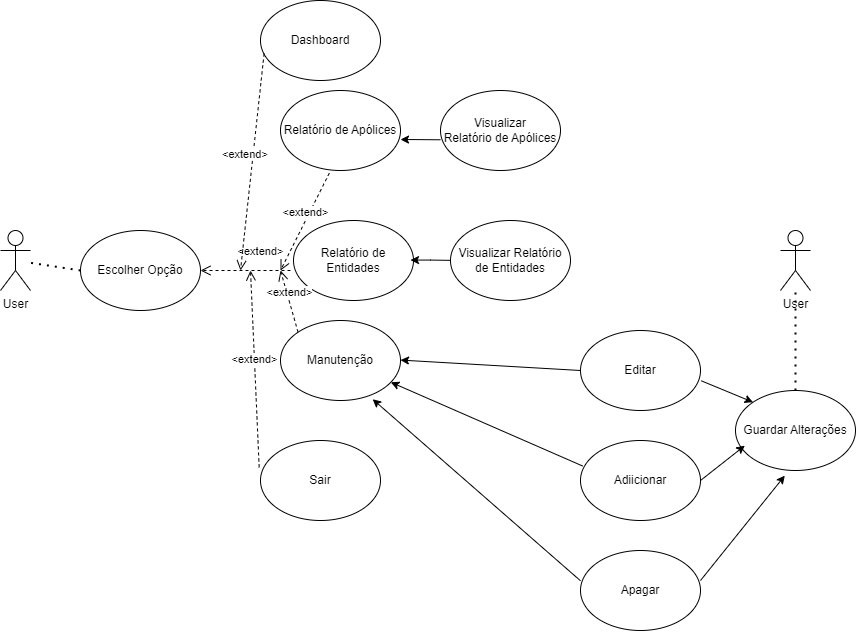


Figura - Diagrama de Casos de Uso

# Conclusões

Aprofundando e aperfeiçoando os conhecimentos de Dart, foi possível identificar uma certa familiaridade com Java, linguagem bastante presente na generalidade das unidades curriculares do curso de Engenharia Informática de Instituto Superior Politécnico Gaya.

Os resultados obtidos nos primeiros exercícios foram os desejados, no primeiro os resultados seriam o mesmo para todos os grupos, sendo que a proposta apresentada foi simples e eficaz.

No segundo, a proposta também se manteve simples, embora não muito maior que o anterior, tentamos reduzir sempre ao máximo o código, aproveitando também algum.

Para finalizar, no terceiro exercício, o verdadeiro desafio do TG1, claramente superior, tanto em dimensão, mas também na dificuldade em relação aos outros dois exercícios, conseguimos concluir praticamente todos os requisitos com exceção dos casos de manutenção que se encontram incompletos.

# Ferramentas

* Repositório Git : https://github.com/joaomcordeiro/CMO
* Visual Studio Code
* Draw.io

# Bibliografia

Pub Dev (Abril 2014) console 4.1.0

https://pub.dev/packages/console

api.dart.dev (Junho 2021) remove method

https://api.dart.dev/be/180791/dart-core/List/remove.html

api.dart.dev (Junho 2021) updateAll method

https://api.dart.dev/stable/2.5.0/dart-core/Map/updateAll.html

Dart/Flutter List Tutorial with Examples. Bezkoder. (Abril 2021)

https://www.bezkoder.com/dart-list/

Dart.Dev (Continuous Editing as of 2022)

Dart/Flutter List Tutorial with Examples