AEDA – Cartão Amigo Museus de Portugal

Trabalho 1 – Realizado por:
João de Jesus Costa - up201806560
João Lucas Silva Martins – up201806436
Tiago Duarte da Silva – up201806516

Descrição do Problema

Tema 4 – Cartão Museus de Portugal (Parte 1)

O Cartão Amigo dos Museus de Portugal é um cartão anual que oferece entrada gratuita e ilimitada na Rede Portuguesa de Museus composta por 156 museus nacionais, bem como descontos em eventos selecionados de várias salas de espetáculo aderentes à iniciativa.

O departamento de *marketing* da Direcção-geral do Património Cultural identificou como prioritário o desenvolvimento de uma aplicação para a compra de bilhetes para os museus e salas de espetáculo nacionais dedicada aos aderentes dos Cartões Amigo.

Os cartões têm a validade de 12 meses e podem ser de 3 tipos: Cartão Amigo Uni - com o custo de 32.45€/ano reservado a estudantes e professores universitários; Cartão Amigo Silver - com o custo de 30€/ano dedicado a cidadãos com mais de 65 anos; e o Cartão Amigo Individual - com o custo de 54.90€/ano para os restantes associados. Para a emissão do cartão, será necessária a informação sobre nome, data de nascimento, morada e contacto móvel do associado, bem como a data de aquisição do cartão.

Independentemente do tipo de cartão, os detentores de Cartão Amigo dos Museus de Portugal terão sempre um desconto de 25% em eventos selecionados que ocorram em salas de espetáculo aderentes à iniciativa.

Ainda na recente estratégia de *marketing*, foi decidido avançar com a opção de permitir que os detentores do cartão Silver assistam sem qualquer custo adicional a eventos a realizar-se num museu ou sala de espetáculos abrangida pela Rede e pertencente à área de residência do associado, caso, faltando menos do que 8 horas para o início do evento ainda não tenham sido vendidos mais do que 50% dos bilhetes disponíveis para o mesmo. Nesse caso, os detentores de cartão Silver irão receber uma mensagem com esta informação e com toda a informação necessária para reservar bilhete para o referido evento.

O sistema deve incluir e gerir informação relativa aos museus da Rede Portuguesa de Museus, sobre os detentores dos Cartões Amigo, sobre os tipos de cartão, bem como dos eventos promovidos pelas instituições aderentes à iniciativa. Deve ainda permitir reservar bilhetes para as iniciativas promovidas por estas instituições.

Implemente também outras funcionalidades que considere relevantes, para além dos requisitos globais enunciados na página inicial.

- **Informação/Membros-Dado/Classes**
  - Cartões
  - Rede Portuguesa de Museus
  - Museus
  - Eventos
  - Salas de espetáculo
  - Contacto e Nome (*strings*)
  - Data
  - Morada
  - Descontos e Custos (*floats*)
- **Ações/Métodos**
  - Desconto em eventos
  - Compra de bilhetes
  - Emissão de cartões
  - Cartões Silver podem assistir a eventos com uma seleção específica de filtros sem custos
  - Gerir museus, cartões e eventos de instituições
  - Reservar bilhetes para iniciativas para instituições

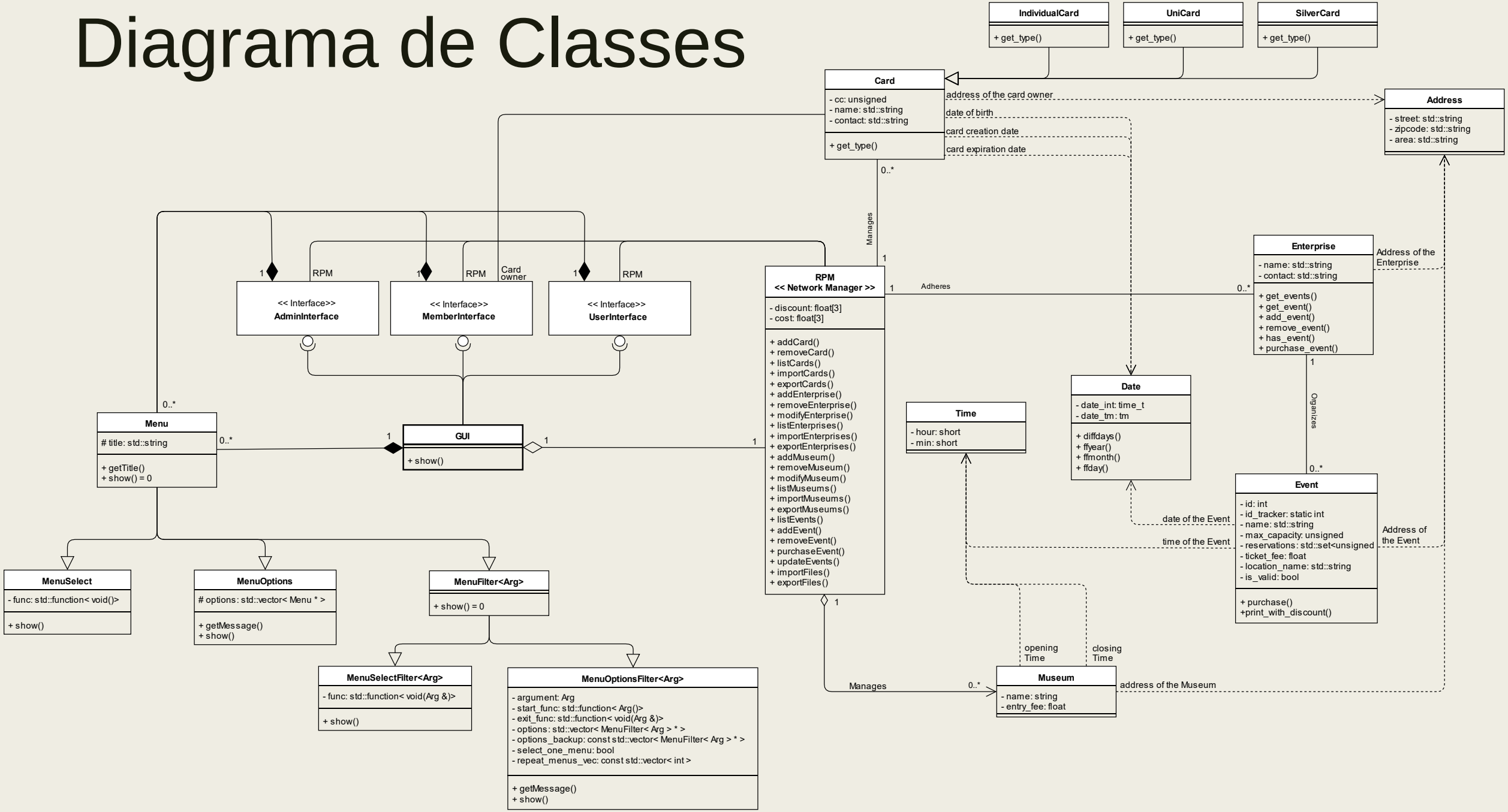
Solução para o problema

- Com base no enunciado o grupo decidiu implementar:
  - **Classes complementares (de uso geral):**
    - Classe de Morada *Address* (rua, código postal e região, ou apenas região)
    - Classe de Data *Date* (ano, mês, dia, dia da semana e ano bissexto)
    - Classe de Tempo *Time* (em horas e minutos)
    - Classe de Menus
  - **Classe cartões/utilizadores:**
    - Possui informação sobre aderentes e os tipos de subscrição
    - Controla validade de cartões
  - **Classe museus**
  - **Classe eventos**
  - **Classe empresas:**
    - Gere os seus eventos
  - **Classe rede de museus:**
    - Gere os seus museus, cartões e empresas
    - Contém informação sobre os custos de subscrição e descontos associados a cada cartão
  - **Classe (ativa) de GUI:**
    - Faz a ligação com a Rede dos Museus usando Interfaces

Algumas notas relevantes

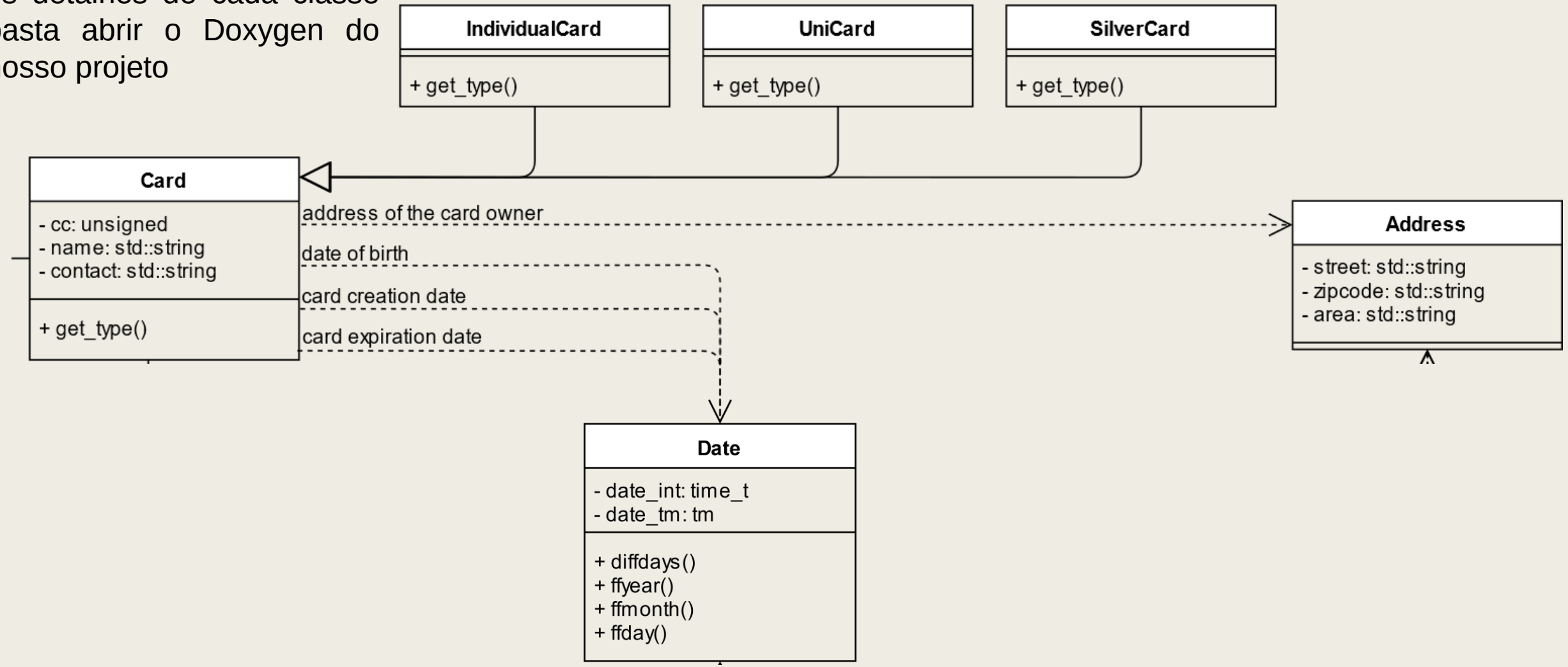
- Algoritmos de pesquisa e filtragem foram adicionados de maneira a facilitar operações CRUD
- É possível selecionar vários elementos da rede. Tal permite a visualização e/ou remoção de vários elementos simultaneamente
- A nossa implementação visou facilitar a implementação de novos tipos de cartão no futuro, tal como cada tipo de cartão ter o seu próprio desconto, preço e perído de renovação, que podem ser lidos do nosso ficheiro de configuração
- Integração de Google Tests no projeto para garantir a estabilidade da aplicação durante o desenvolvimento
- A identificação de eventos é feita por um ID estático único, incrementado por cada novo Evento associado à Rede
- A identificação de aderentes é feita pelo seu cartão de cidadão (cc)
- Aquando da leitura de ficheiros, a informação atual da rede (não salva em ficheiros) não é preservada. Se algum ficheiro não for lido com sucesso, a informação dos restantes é utilizada
- A instanciação de um objecto do tipo Date é seguida de uma tentativa automática de correção de erros (e.g.: 32 Janeiro de 2000 é convertido para 01 de Fevereiro de 2000)

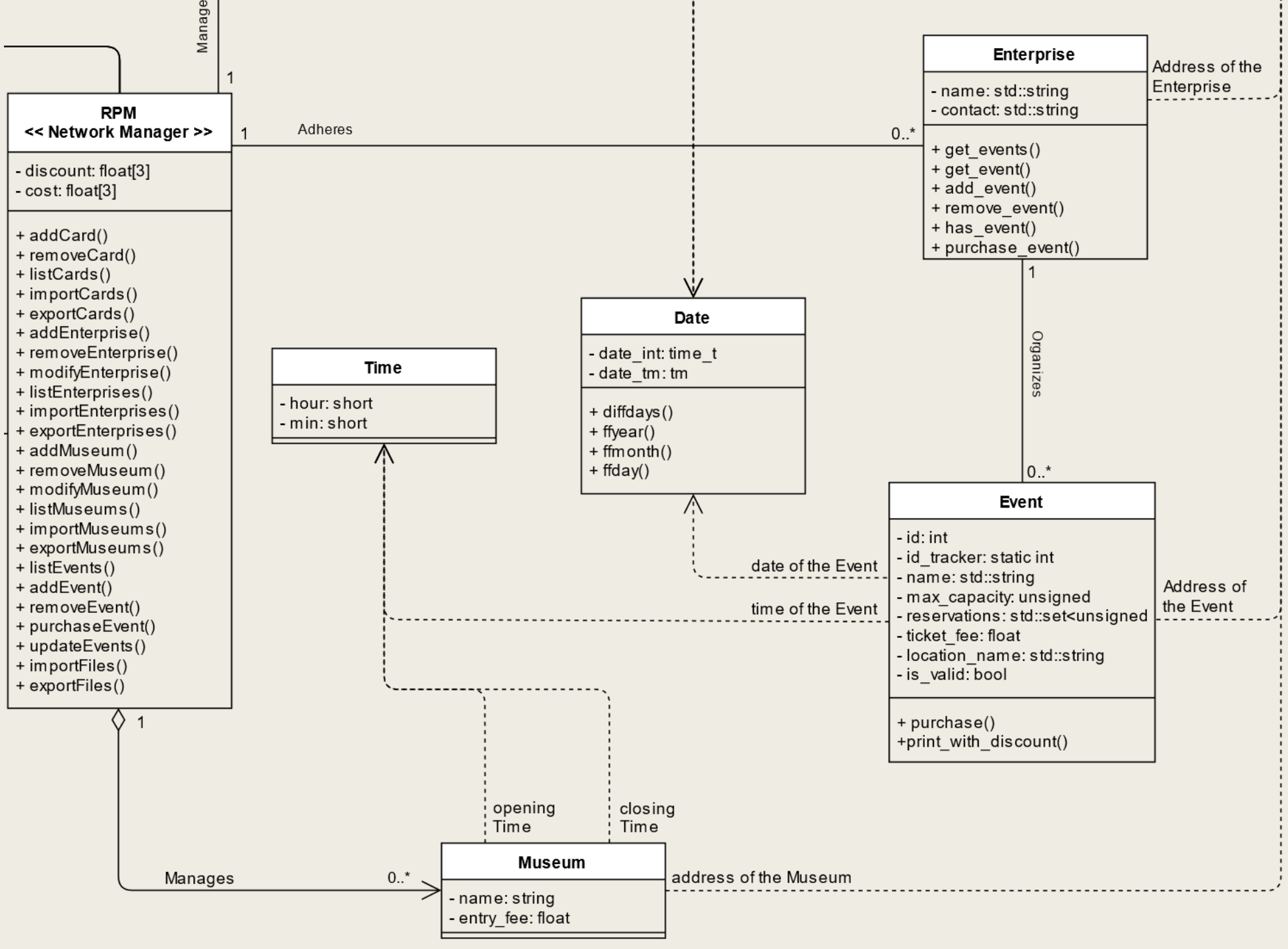
Diagrama de Classes

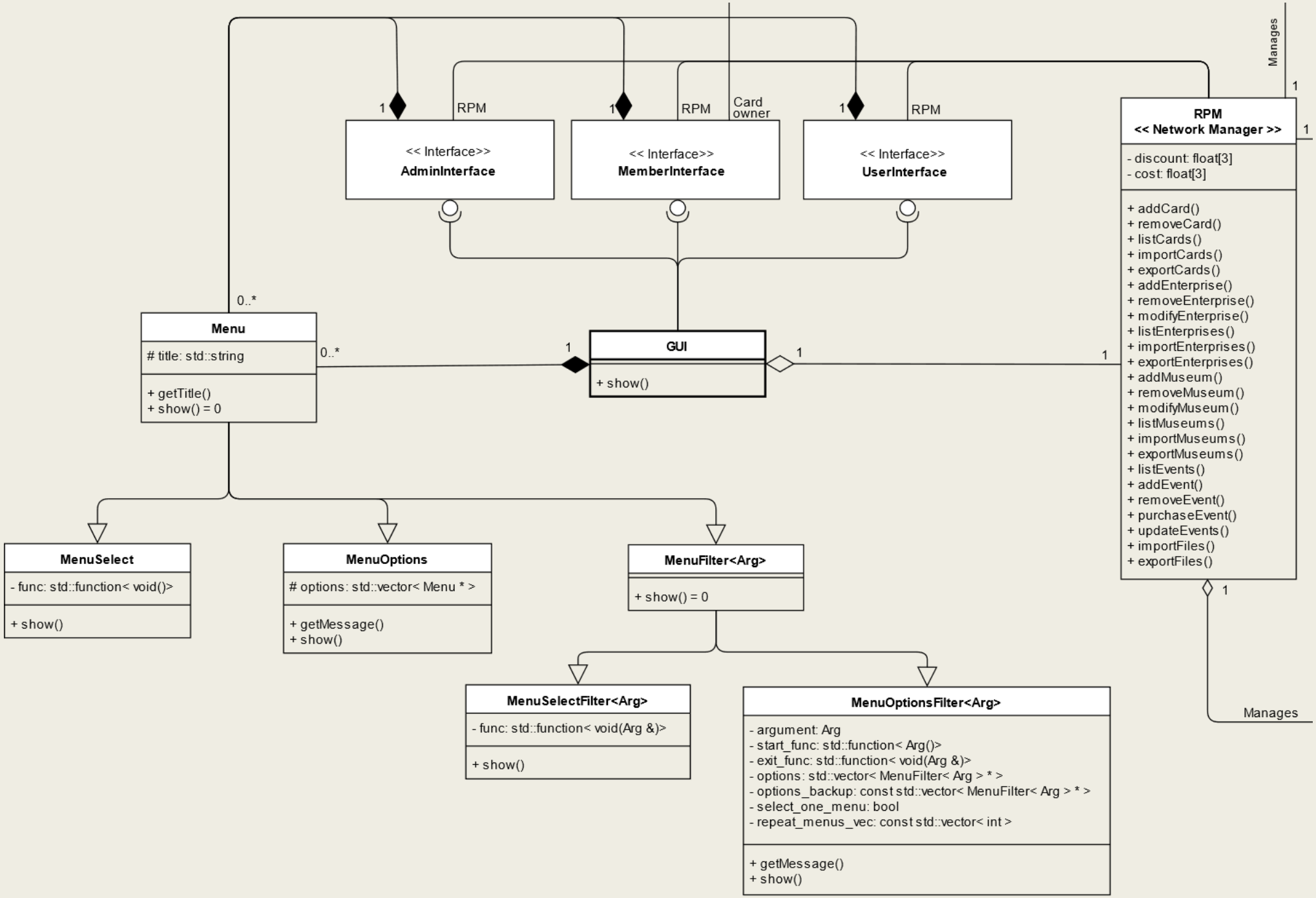


Nota:

O nosso UML encontra-se simplificado, mas para ver os detalhes de cada classe basta abrir o Doxygen do nosso projeto




Manages
1
RPM
<< Network Manager >>
- discount: float[3]
- cost: float[3]
+ addCard()
+ removeCard()
+ listCards()
+ importCards()
+ exportCards()
+ addEnterprise()
+ removeEnterprise()
+ modifyEnterprise()
+ listEnterprises()
+ importEnterprises()
+ exportEnterprises()
+ addMuseum()
+ removeMuseum()
+ modifyMuseum()
+ listMuseums()
+ importMuseums()
+ exportMuseums()
+ listEvents()
+ addEvent()
+ removeEvent()
+ purchaseEvent()
+ updateEvents()
+ importFiles()
+ exportFiles()
1
Adheres
0..*
Enterprise
- name: std::string
- contact: std::string
+ get_events()
+ get_event()
+ add_event()
+ remove_event()
+ has_event()
+ purchase_event()
Address of the Enterprise
1
Organizes
0..*
Date
- date_int: time_t
- date_tm: tm
+ diffdays()
+ ffyear()
+ ffmonth()
+ ffday()
Time
- hour: short
- min: short
Event
- id: int
- id_tracker: static int
- name: std::string
- max_capacity: unsigned
- reservations: std::set<unsigned
- ticket_fee: float
- location_name: std::string
- is_valid: bool
+ purchase()
+print_with_discount()
date of the Event
time of the Event
Address of the Event
opening Time
closing Time
Manages
0..*
Museum
- name: string
- entry_fee: float
address of the Museum


Manages
1
RPM
1
RPM
Card owner
1
RPM
<< Interface>>
AdminInterface
<< Interface>>
MemberInterface
<< Interface>>
UserInterface
RPM
<< Network Manager >>
1
- discount: float[3]
- cost: float[3]
+ addCard()
+ removeCard()
+ listCards()
+ importCards()
+ exportCards()
+ addEnterprise()
+ removeEnterprise()
+ modifyEnterprise()
+ listEnterprises()
+ importEnterprises()
+ exportEnterprises()
+ addMuseum()
+ removeMuseum()
+ modifyMuseum()
+ listMuseums()
+ importMuseums()
+ exportMuseums()
+ listEvents()
+ addEvent()
+ removeEvent()
+ purchaseEvent()
+ updateEvents()
+ importFiles()
+ exportFiles()
0..*
Menu
title: std::string
+ getTitle()
+ show() = 0
0..*
1
GUI
+ show()
1
1
1
Manages
MenuSelect
- func: std::function< void()>
+ show()
MenuOptions
options: std::vector< Menu * >
+ getMessage()
+ show()
MenuFilter<Arg>
+ show() = 0
MenuSelectFilter<Arg>
- func: std::function< void(Arg &)>
+ show()
MenuOptionsFilter<Arg>
- argument: Arg
- start_func: std::function< Arg()>
- exit_func: std::function< void(Arg &)>
- options: std::vector< MenuFilter< Arg > * >
- options_backup: const std::vector< MenuFilter< Arg > * >
- select_one_menu: bool
- repeat_menus_vec: const std::vector< int >
+ getMessage()
+ show()

Estrutura de ficheiros

A exportação de informação é feita em 4 ficheiros: um para a configuração da rede, um para museus, um para cartões e um para empresas (contendo os seus eventos).

O grupo decidiu que a informação dos mesmos não deve ser modificada pelo utilizador diretamente mas com o uso da aplicação desenvolvida, com exceção do ficheiro de configuração da Rede de Museus.

```
1 cards_file_name: files/cards.txt
2 museum_file_name: files/museums.txt
3 enterprise_file_name: files/enterprises.txt
4 individual::cost: 54.9
5 individual::discount: 0.25
6 silver::cost: 30
7 silver::discount: 0.25
8 uni::cost: 32.45
9 uni::discount: 0.25
```

e.g. do conteúdo de um ficheiro de configuracão da Rede

```
 1 3                                          -> Número de Empresas
 2 Pinheiro Lda                               -> Nome da Empresa
 3 general@pinheirolda.sa.pt                  -> Contacto da Empresa
 4 3                                          -> Número de Eventos
 5 Concerto Ornatos Violeta                   -> Nome do Evento
 6 23.000000                                  -> Custo do Evento
 7 2                                          -> Número de vezes Comprado
 8 27593658 73559306                          -> CC's de quem comprou Evento
 9 60                                         -> Lotação Máxima
10 Pavilhao Rosa Mota                         -> Nome do Recinto do Evento
11 Rua D. Manuel II :::::: 4050-346 :::::: Porto -> Morada do Recinto do Evento
12 1031702400                                 -> Data do Evento
13 22:0                                       -> Hora do Evento
```

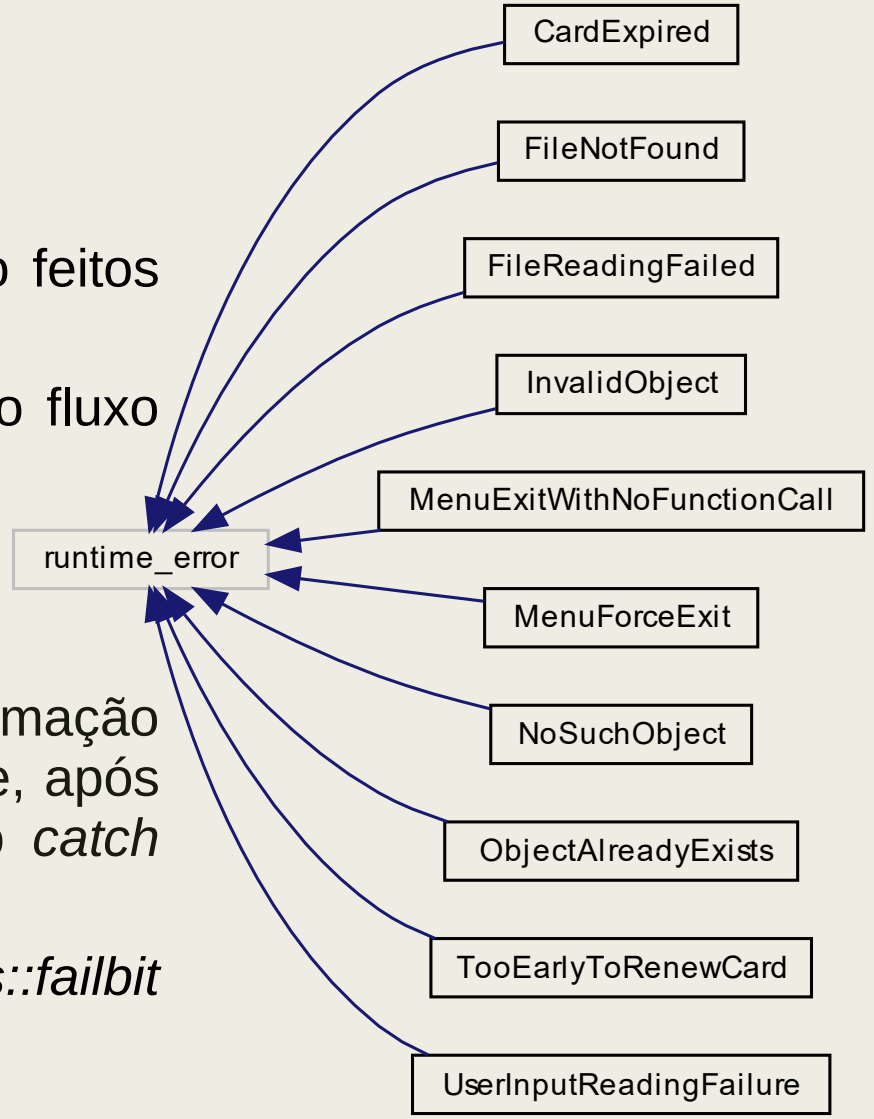
e.g. do conteúdo de um ficheiro de informação de Empresas

Como tal, a tentativa de uma leitura direta de ficheiros é difícil. Por exemplo, a classe Date é representada como o número de segundos desde o *Epoch* definido pelo compilador.

Tratamento de exceções

- O grupo decidiu agrupar as exceções em 3 tipos:
  - Exceções relativas a erros de *input*
  - Exceções da Rede de Museus: *thrown* quando são feitos pedidos inválidos à Rede
  - Exceções relativas a Menus: usadas no controlo do fluxo da seleção de menus

- Notas:
  - Todos os métodos que envolvem a extração de informação do *std::cin* fazem o throw de exceções internamente, após a inserção de um valor considerado inválido, cujo *catch* coloca o *std::ios::failbit* a **1**
  - As interfaces apenas necessitam de avaliar o *std::ios::failbit* para saber se o *input* foi bem sucedido



As exceções que definimos derivam todas da std::runtime_error

Funcionalidades Implementadas

- As funcionalidades de CRUD foram completamente implementadas para Museus, Empresas, Cartões e Eventos
- Interface de Admin:
  - Tem permissão para efetuar todas as operações CRUD, à exceção da atualização da informação nos Cartões (só pode ser efetuado pelo proprietário deste)
- Interface de Membro (aderente da Rede):
  - Listar todos os Eventos, Museus e a sua própria informação (não pode listar informacão de outros Membros nem de Empresas)
  - Renovar o seu Cartão (so é possível a partir de, no máximo, 2 meses antes de este expirar)
  - Comprar Eventos
  - Atualizar parte da sua informação (não pode alterar o seu CC, a data de criação e expiração do seu cartão)
  - Apagar a sua conta (direito a ser esquecido do RGPD)

Funcionalidades Implementadas

- Interface de Membro (continuação):
  - Membros do Cartão Silver (idosos), caso existam, são notificados de Eventos a decorrer nas próximas 8 horas, na sua área de residência (com lotação inferior a 50%) aquando do *login* na Rede e na compra de bilhetes para esses Eventos selecionados. Estes serão gratuitos para o utilizador em questão
  - Durante a listagem de Eventos para um determinado tipo de Cartão, é tanto listado o preço do evento como o preço do evento, com o desconto, associado ao Membro, aplicado
- Interface de Utilizador não registado (sem Cartão criado):
  - Listar todos os Eventos e Museus
  - Criar uma conta (cartão) para si (apresentando informação sobre o preço total da compra, dependendo do tipo de cartão que lhe pode ser atribuído)
- Exportação e Importação da informação para ficheiros

Funcionalidades Implementadas - Filtros

Todas as operações de listagem e seleção de aglomerados de informação têm filtros distintos para ajudar a navegar pela grande quantidade de dados apresentados pela Rede

- **Filtro de Eventos:**
  - Por Id
  - Por Nome do Local
  - Por Endereço
  - Por Nome
  - Nas próximas horas (definidas pelo utilizador)
  - Por Datas:
    - Entre Datas
    - Numa Data
- **Filtro de Museus:**
  - Por Endereço
  - Por Nome
- **Filtro de Utilizadores** (apenas para admin)
  - Por Nome
  - Por Validade do Cartão:
    - Cartões Válidos
    - Cartões Inválidos
- **Filtro de Empresas** (apenas para admin)
  - Por Endereço
  - Por Nome
  - Por Id de um dos seus Eventos

Destaque de funcionalidade - Menu

- Para facilitar a implementação e aumentar a legibilidade das interfaces, foi criada uma classe Menu (abstrata), usada para a instanciação, escolha e listagem de menus e as suas opções
- No início da realização do projeto foram concebidos dois tipos de menus, derivados da classe Menu:
  - **MenuOptions** - Menu que possui uma lista de opções em que cada uma representa um (outro) Menu
  - **MenuSelect** - Menu que quando instanciado chama uma função (guardada num membro-dado do tipo *std::function*)
- No entanto, para simplificar a seleção de elementos da Rede (eventos, museus, etc...) foram definidas mais duas classes de menus (usadas para filtragem de elementos):
  - **MenuOptionsFilter<Arg>** - Menu que inicializa (quando esta não estiver inicializada) uma variável Arg com uma função dada. Possui uma lista de opções que quando selecionadas podem modificar Arg
  - **MenuSelectFilter<Arg>** - Menu que altera a variável Arg, passada por referência, usando seu membro-dado do tipo *std::function*

Principais dificuldades encontradas e esforço de cada elemento do grupo

- O trabalho foi distribuído de maneira homogénea por todos os elementos do grupo
- As principais dificuldades encontradas foram:
  - A implementação de templates na classe Menu
  - Indecisão sobre a estrutura e design das interfaces e dos diferentes filtros
  - Handling de exceções (onde dar *throw* e onde dar *catch*)
  - Definição de exceções gerais o suficiente para o nosso uso mas não demasiado gerais
  - Documentação do código usando *Doxygen* e a configuração do mesmo
  - Atualização do design inicial do *UML* para melhor refletir o resultado final do nosso código
  - Tradução de todas as ideas, features implementados e decisões relativamente design para o relatório

Features Extra que gostaríamos de Implementar

- Interface especificamente só para Empresas que querem aderir/já aderiram à iniciativa: Todos os métodos *back-end* estão implementados. Resta criar a Interface dedicada e dividir as funcionalidades entre esta e a do Admin da Rede
- Eventos Recorrentes: e.g.: Eventos que se repetem todas as semanas
- Dias de funcionamento dos museus: e.g.: não está aberto aos fins de semana
- Horário de funcionamento dos museus para cada dia da semana em específico
- Museus conseguirem declarar eventos diretamente (a ocorrer no seu espaço) em vez de se fazerem passar por uma empresa para o fazer
- *Passwords* para membros durante o seu login (utilizando métodos de *hashing* por exemplo)