

ALGORITMOS E ESTRUTURAS DE DADOS

Gestão de Ginásios

AEDA1718_Turma1_G3

24 de novembro de 2017

César Manuel Nobre Medeiros - up201605344@fe.up.pt

Duarte Manuel Marques Mano Menezes Frazão - up201605658@fe.up.pt

Sandro Miguel Tavares Campos - up201605947@fe.up.pt

ÍNDICE

Breve resumo	pág. 3
Solução Implementada	pág. 4
Diagrama de Classes (UML)	pág. 5
Casos de utilização para a aplicação	pág. 6
Dificuldades no desenvolvimento	pág. 10
Método de distribuição de trabalho	pág. 10

Breve resumo

O projeto implementado é um sistema de gestão para ginásios, e como tal, a um funcionário que use o programa é-lhe permitido não só aceder, como alterar e monitorizar o ginásio, desde que esteja devidamente autorizado. Com isto, surgem funcionalidades que fazem sentido terem sido implementadas, por refletirem aspetos inerentes à gestão de um negócio, entre elas a gestão dos seus clientes, funcionários e contabilidade. Consequentemente, cada cliente registado no sistema encontra-se associado a um determinado tipo de subscrição do ginásio que frequenta, e por sua opção, a um professor especialista nas áreas em que mais deseja trabalhar, o designado “personal trainer”. O pagamento por parte dos clientes deve ser realizado mensalmente, e a remuneração aos funcionários é de igual forma distribuída através do sistema.

Solução Implementada

1. Login

2. Main menu

2.1. Gym menu

2.1.1. Finances menu

2.1.1.1. Obtain account extract

2.1.1.2. Deposit amount

2.1.1.3. Make payments

2.1.2. Schedule menu

2.1.3. Capacity menu

2.1.3.1. Change maximum capacity

2.1.3.2. Change maximum number of clients

2.1.4. Subscriptions menu

2.1.4.1. Add subscription

2.1.4.2. Edit subscription

2.1.4.3. Remove subscription

2.1.4.4. Display subscriptions

2.1.5. Display gym information

2.2. Client menu

2.2.1. Add client

2.2.2. Edit client

2.2.3. Remove client

2.2.4. Display client information

2.2.5. List clients

2.3. Staff menu

2.3.1. Personal trainer menu

2.3.1.1. Add personal trainer

2.3.1.2. Remove personal trainer

2.3.1.3. Edit personal trainer

2.3.1.4. Display personal trainer information

2.3.1.5. List personal trainers

2.3.2. Add staff

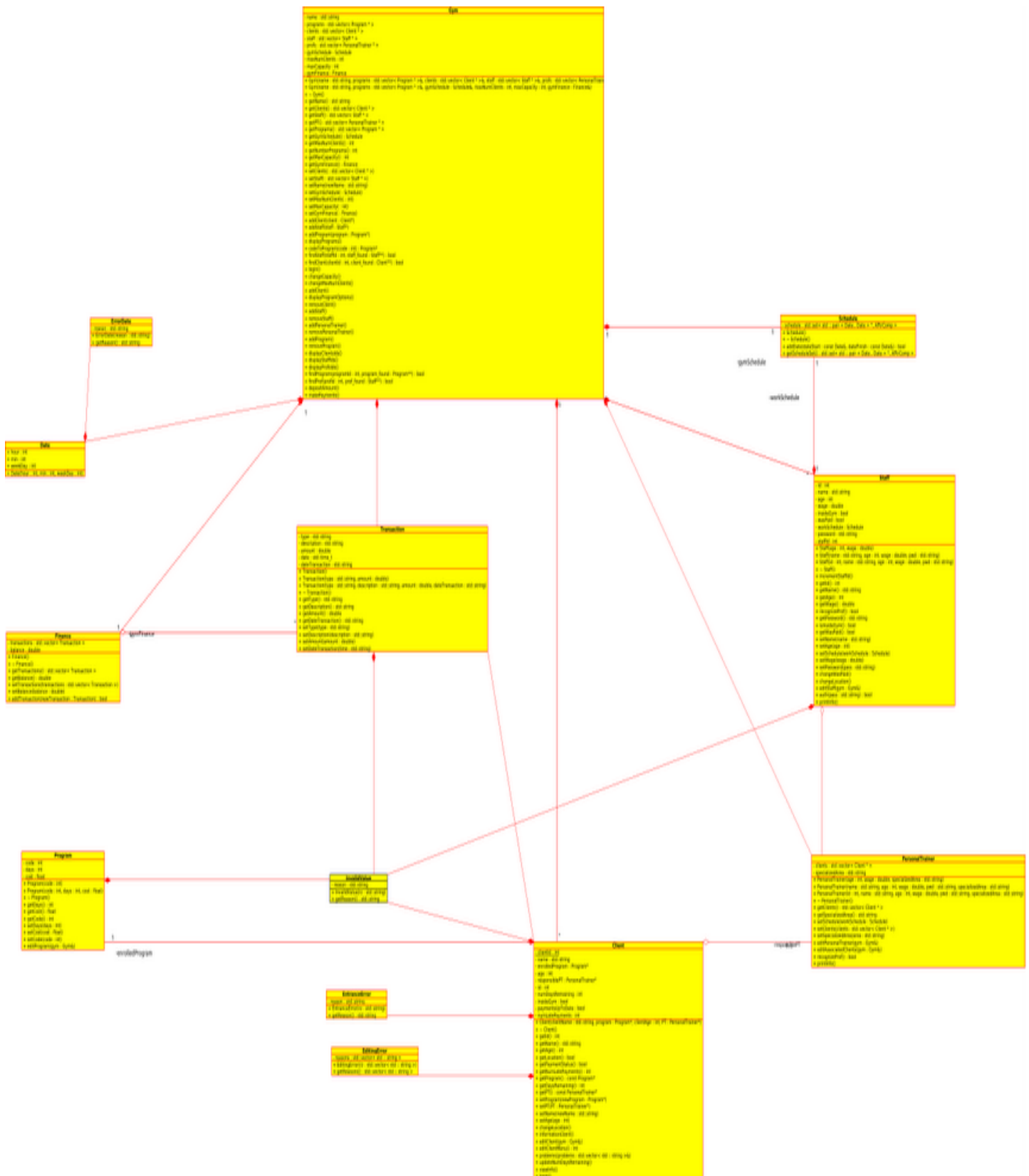
2.3.3. Edit staff

2.3.4. Remove staff

2.3.5. Display staff information

2.3.6. List staff

Diagrama de classes (UML)



Casos de utilização para a aplicação

Classes

Gym

Considerada a classe principal, a classe Gym é a mais extensa, e permite instanciar o objeto ginásio. Os seus atributos estão relacionados com as suas características, e por isso possui diversos vetores que incluem os **programas** para subscrição disponíveis, os seus **clientes**, **funcionários** e “**personal trainers**”. Além disso, possui instâncias das classes **Finance** e **Schedule**, que permitem dar a conhecer o horário de funcionamento e o estado financeiro do mesmo.

Client

A classe Client tem como atributos **id**, **name**, **age** (para a criação de um utilizador do ginásio), **enrolledProgram** e **responsiblePT** que permitem fornecer informação acerca do programa em que o utilizador se encontra inscrito e a que Personal Trainer este se encontra associado. Consequentemente, Client possui ainda atributos **insideGym**, **paymentsUpToDate**, **numDaysRemaining** e **numLatePayments**, que permitem monitorizar e saber em que situação se encontram os pagamentos do cliente, e quantos dias de ginásio lhe são permitidos frequentar com o plano subscrito.

Staff

De forma semelhante à classe anterior, esta classe possui atributos para a criação de um funcionário do ginásio, mas sendo que desta vez surgem **wage** e **wasPaid**, que nos dão informação acerca do estado de pagamento salarial. Um funcionário além disso possui uma password para acesso à aplicação, **password**, e um horário de trabalho, **workSchedule**.

PersonalTrainer: public Staff

A classe PersonalTrainer deriva publicamente da classe Staff, portanto, além de possuir os atributos de Staff, possui um vetor de **clientes** da qual um “personal trainer” é tutor e um parâmetro indicando a área em que este é especializado, **specializedArea**.

Program

Ilustrando os programas para subscrição disponíveis no ginásio, esta classe possui como atributos **code** (código do programa), **days** (número de dias permitidos a frequentar o ginásio) e **cost** (custo do programa).

Transaction

Intervindo na classe Finance, esta classe permite explicitar o tipo de transação a efetuar. Por isso, os atributos desta classe, **type**, **description**, **amount** e **date** são importantes para registar um determinado movimento bancário do ginásio.

Finance

Possuindo como atributos um vetor de **transações** e o saldo presente na conta bancária do ginásio, **balance**, através desta classe um funcionário poderá gerir o pagamento dos trabalhadores e realizar depósitos na conta.

Date

A classe Date com atributos **hour**, **min** e **weekDay**, como os próprios nomes indicam, intervém na criação de uma data, com diversas utilidades nas diferentes classes apresentadas.

Schedule

A classe Schedule tem um **conjunto de par de datas** como atributo, ilustrando a implementação de um horário da qual outras classes necessitam, para se poderem atribuir horários de trabalho e de funcionamento do ginásio.

EntranceError, EditingError, ErrorDate, InvalidValue

Estas classes são dedicadas ao tratamento de erros durante a execução do programa, mais concretamente permitindo o lançamento de exceções. Devido à sua semelhança, todas possuem uma ou mais **razões**, que originaram o erro, como atributos.

Ficheiros de texto

Na implementação, é sempre importante possuímos um sistema de leitura e de escrita para ficheiros de texto, sendo que a partir de um, de input, obtemos os dados predefinidos para iniciar o programa, e no outro, de output, guardamos as alterações feitas ao ginásio durante a execução, para futuro uso.

Assim sendo estes ficheiros possuem informação relativa ao ginásio, como as suas características, o seu horário de funcionamento, a sua lista de clientes e de staff, programas e transações efetuadas. O formato do ficheiro aparece especificado a seguir:

Gym

[nome num_max_clientes capacidade]

Finance

[tipo_transação montante descrição data]

...

;

Schedule

[1 hora_inicio hora_fim]

...

[7 hora_inicio hora_fim]

;

Programs

[código num_dias custo]

...

;

Staff

[ID nome idade salário password]

...

;

PersonalTrainer

[ID nome idade salário password área_especialidade]

...

;

Clients

[nome código_programa idade ID_PersonalTrainer]

...

;

end

Dificuldades no desenvolvimento

As principais dificuldades que o nosso grupo encontrou durante o desenvolvimento deste projeto foram, de um modo geral, mais relacionadas com a conjugação de código na plataforma GitHub e da forma como o desenvolvimento conjunto do projeto em diferentes IDE poder, por vezes, trazer algumas complicações.

Método de distribuição de trabalho

A carga de trabalho do projeto foi distribuída homogeneamente por todos os elementos do grupo, sendo que houve parcelas que foram realizadas por todos um pouco em conjunto, conforme a necessidade de cada um para conseguir implementar a parte individual que lhe competia. Muito resumidamente, as componentes individuais distribuídas são as que se apresentam a seguir:

- **César Medeiros** – implementação das classes de erro, data e horários, leitura / escrita para ficheiros, algoritmos de ordenação e listagens;
- **Duarte Frazão** – implementação dos menus, das classes cliente e programa, criação de documentação Doxygen e do diagrama de classes UML;
- **Sandro Campos** – implementação das classes funcionário, “personal trainer”, finanças e transações, e escrita do relatório.