



MIEIC

AEDA

Turma5_G6:

Rui Pinto : up201806441;

Tiago Gomes: up201806658;

Válter Castro: up201706546.



Índice

- National_Team
- Estrutura do Projeto
- UML
- Estrutura de Ficheiros
- Tratamento de exceções
- Funcionalidades implementadas
- Destaque de Funcionalidade
- Exemplos de execução
- Dificuldades
- Exemplos de Execução

National_Team

Este projeto, tem como objetivo, a criação de um modelo de gestão no setor desportivo com relação à seleção nacional.

Uma seleção nacional, tem como âmago da sua existência a vontade de demonstrar o patriotismo , e de promover uma união através do desporto. O seu funcionamento, reside numa hierarquia complexa, desde o pessoal encarregue do tratamento e distribuição dos equipamentos individuais, passando pelos funcionários responsáveis pelo funcionamento da estrutura, seguido dos atletas de nacionalidade referente ao país de origem e finalizando com os órgãos máximos, responsáveis pela gestão da mesma.

Como exemplo atual, temos a seleção nacional de futebol, que representa um desporto nuclear no desenvolvimento da história, mais recente, da humanidade , e que tem não só como objetivo demonstrar a sua habilidade ao mais alto nível, mas também aprazer os seus adeptos a nível mundial.

Estrutura do Projeto

Após uma decisão estruturada e unânime, foram implementadas as seguintes classes:

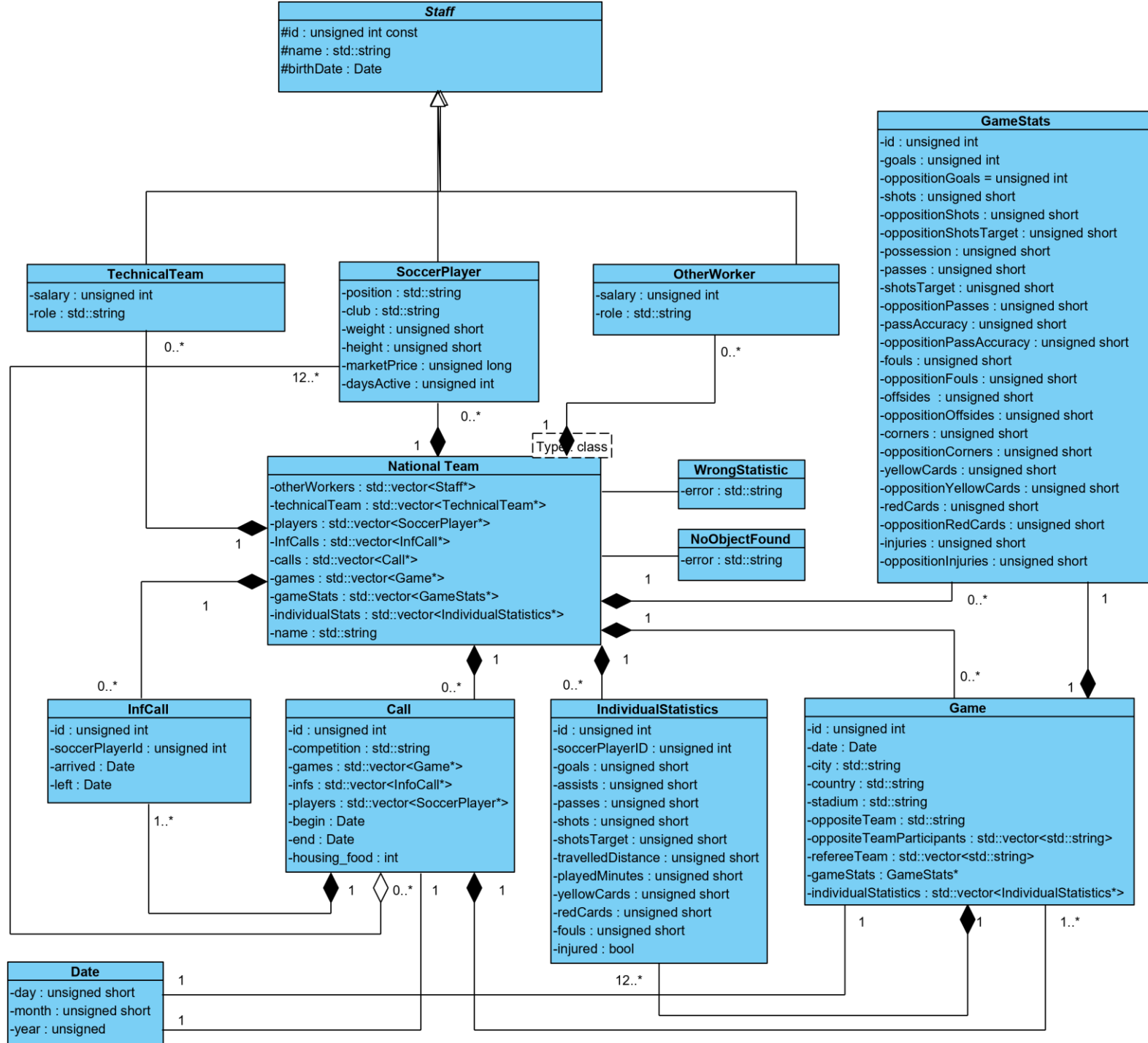
- *Call*
- Date
 - WrongDateFormat
- Game
- GameStats
- IndividualStatistics
- InfCall
- NationalTeam
- Staff
 - SoccerPlayer
 - TechnicalTeam
 - OtherWorker
- NoObjectFound
- WrongStatistic

Foram aplicados todos os conhecimentos adquiridos e exigidos pela disciplina neste projeto, sendo também o número de classes e respectivos atributos e métodos, resultado de uma ponderação sobre a melhor maneira de tornar o código legível e ao mesmo tempo, completo. Para tal efeito foram usadas todas as funções da biblioteca STL necessárias e também o gerador de documentação Doxygen.

Após escrupular o código, é de verificar a existência de três classes no cerne deste projeto:

- NationalTeam, no centro da operação, responsável pela gestão de informação das outras classes, utilizando vetores de apontadores para objetos das diversas classes de modo assegurar a proteção da integridade do seu conteúdo.
- Call, responsável pela logística de todos os jogos.
- Game, que gere todas as estatísticas individuais e globais, bem como todos os presentes nos diversos jogos ocorridos.

UML



National Team

```

- otherWorkers : std::vector<Staff*>
- technicalTeam : std::vector<TechnicalTeam*>
- players : std::vector<SoccerPlayer*>
- InfCalls : std::vector<InfCall*>
- calls : std::vector<Call*>
- games : std::vector<Game*>
- gameStats : std::vector<GameStats*>
- individualStats : std::vector<IndividualStatistics*>
- name : std::string

+ NationalTeam()
+ getName() : string
+ getOtherWorkers() : vector<Staff*>&
+ createOtherWorker() : bool
+ alterOtherWorker() : bool
+ addOtherWorker(oW : Staff*) : void
+ deleteOtherWorker() : bool
+ readOtherWorkersFile(filename : std::string) : bool
+ writeOtherWorkersFile(filename : std::string) : bool
+ displayOtherWorkers() : bool
+ displaySoccerPlayers() : void
+ displayGames() : void
+ displayTechnicalTeamMembers() : void
+ workerLookUp(workers : vector<Type>&) : Type
+ getByID(elements : std::vector<Type>&, id : int) : Type&
+ getLastID(elements : std::vector<Type>&) : unsigned int
+ createSoccerPlayer() : bool
+ alterSoccerPlayer() : bool

```

```

+ addSoccerPlayer(sP : SoccerPlayer*) : void
+ deleteSoccerPlayer() : bool
+ readSoccerPlayersFile(filename : std::string) : bool
+ writeSoccerPlayersFile(filename : std::string) : bool
+ addOtherSoccerPlayer(sP : SoccerPlayer*) : void
+ createGame(game : Game*) : void
+ readGamesFile(filename : std::string) : bool
+ createCall() : bool
+ addCall(call : Call*) : void
+ readCallsFile(filename : std::string filename) : bool
+ createGameStatistics(gameID : unsigned int) : GameSt...
+ addGameStatisticsFile(gStats : GameStats*) : void
+ writeGamesStatisticsFile(filename : std::string) : bool
+ createIndividualStatistics(playerID : unsigned int) : Indivi...
+ addIndividualStatistic(iStat : IndividualStatistics*) : void
+ alterIndividualStatistic() : bool
+ readIndividualStatisticsFile(filename : std::string) : bool
+ writeGamesFile(filename : std::string) : bool
+ writeIndividualStatisticsFile(filename : std::string) : bool
+ readInfCalls(filename : std::string) : bool
+ createInfCalls(sPId : unsigned int) : InfCall*
+ addInfCall(inf : InfCall*) : void
+ addTechnicalTeamMember(tTeam : TechnicalTeam*) : ...
+ readTechnicalTeamFile(filename : std::string) : bool
+ createTechnicalTeamMember() : bool
+ alterTechnicalTeamMember() : bool
+ deleteTechnicalTeamMember() : bool
+ writeTechnicalTeamFile(filename : std::string) : bool

```


Class Diagram1

Call

```

-id : unsigned int
-competition : std::string
-games : std::vector<Game*>
-infs : std::vector<InfoCall*>
-players : std::vector<SoccerPlayer*>
-begin : Date
-end : Date
-housing_food : int

+Call(id : unsigned int, competition : std::string, games : std::vector<Game*>, infs : std::vector<InfoCall*>, players : std::vector<SoccerPlayer*>, begin : Date, end : Date, housing_food : unsigned int)
+getGames() : unsigned int
+getPlayers() : std::vector<Game*>&
+getInfs() : std::vector<InfoCall*>&
+getBeginDate() : Date
+getEndDate() : Date
+getHousingFood() : unsigned int
+getCompetition() : std::string
+setCompetition(competition : std::string) : void
+setGames(games : std::vector<Game*>) : void
+setInfs(infs : std::vector<InfoCall*>) : void
+setPlayers(players : std::vector<InfoCall*>) : void
+setBeginDate(begin : Date) : void
+setEndDate(end : Date) : void
+setHousingFood(housing_food : unsigned int) : void
+header() : static void
+infoPlayers() : void
+infoGames() : void
+infoStats() : void

```

Powered By: Visual Paradigm Community Edition









Class Diagram1

```

classDiagram
    class Game {
        -id : unsigned int
        -date : Date
        -city : std::string
        -country : std::string
        -stadium : std::string
        -oppositeTeam : std::string
        -oppositeTeamParticipants : std::vector<std::string>
        -refereeTeam : std::vector<std::string>
        -gameStats : GameStats*
        -individualStatistics : std::vector<IndividualStatistics*>
        +Game(id : unsigned int, date : Date, city : std::string, country : std::string, stadium : std::string, op...
        +getId() : unsigned int
        +getDate() : Date
        +getCity() : std::string
        +getCountry() : std::string
        +getStadium() : std::string
        +getOppositeTeam() : std::string
        +getOppositeTeamParticipants() : std::vector<std::string>
        +getRefereeTeam() : std::vector<std::string>
        +getGameStats() : GameStats*
        +getIndividualStatistics() : std::vector<IndividualStatistics*>
        +setId(id : unsigned int) : void
        +setDate(date : const Date&) : void
        +setCity(city : const std::string&) : void
        +setCountry(country : const std::string&) : void
        +setStadium(stadium : const std::string&) : void
        +setOppositeTeam(oppositeTeam : const std::string&) : void
        +setOppositeTeamParticipants(oppositeTeamParticipants : const std::vector<std::string>&) : void
        +setRefereeTeam(refereeTeam : const std::vector<std::string>&) : void
        +setGameStats(gameStats : GameStats*) : void
        +setIndividualStatistics(setIndividualStatistics : const std::vector<IndividualStatistics*>) : void
        +info(os : std::ostream&) : void
        +header() : static void
        +info() : void
    }

```

Estrutura de ficheiros

-  Calls.txt
-  Games.txt
-  GameStatistics.txt
-  IndividualStatistics.txt
-  InfCalls.txt
-  OtherWorkers.txt
-  SoccerPlayers.txt
-  TechnicalTeam.txt

ID: 1
Competition: Friendly
Begin Date: 12/03/2010
End Date: 25/03/2010
House-Food: 2000
Games: 1000
PlayersID: 1, 3
InfCalls: 1, 3

:::::::::

ID: 2
Competition: Euro
Begin Date: 12/05/2010
End Date: 25/05/2010
House-Food: 1500
Games: 1000
PlayersID: 1, 2
InfCalls: 1, 2

Calls.txt

ID: 1000
Date: 13/03/2019
City: Lisbon
Country: Portugal
Stadium: Alvalade
Opposite Team: Luxemburgo
Opposite Team Participants: Anthony Moris,
Refereeing Team: Tim Tam Tum, Shium Jhim,
Game Statistics: 1000
Individual Statistics: 2000, 2002

:::::::::

ID: 1001
Date: 23/12/2019
City: Lisbon
Country: Portugal
Stadium: Luz
Opposite Team: Espanha
Opposite Team Participants: Anthony Moris,
Refereeing Team: Tim Tam Tum, Shium Jhim,
Game Statistics: 1001
Individual Statistics: 2001

Games.txt

Game ID: 1000
Goals: 3
Opposing team goals: 0
Shots: 10
Opposing team shots: 3
Shots on target: 6
Opposing team shots on target: 1
Possession: 65
Passes: 530
Opposing team passes: 201
Pass accuracy: 95
Opposing team pass accuracy: 83
Fouls: 15
Opposing team fouls: 27
Yellow cards: 3
Opposing team yellow cards: 6
Red cards: 1
Opposing team red cards: 0
Offsides: 5
Opposing team offsides: 2
Corners: 7
Opposing team corners: 2
Injuries: 0
Opposing team injuries: 1

GameStatistics.txt

ID: 2000
Player ID: 1
Goals: 2
Assists: 1
Shots: 6
Shots on Target: 4
Passes: 54
Travelled Distance: 10
Played Minutes: 90
Yellow Cards: 0
Red Cards: 0
Injured: false
Fouls: 2

::::::::::

ID: 2001
Player ID: 2
Goals: 0
Assists: 1
Shots: 3
Shots on Target: 1
Passes: 57
Travelled Distance: 9
Played Minutes: 83
Yellow Cards: 1
Red Cards: 0
Injured: false
Fouls: 3

ID: 1
Player ID: 1
Arrived Date: 09/11/2019
Left Date: 20/11/2019

::::::::::

ID: 2
Player ID: 2
Arrived Date: 10/11/2019
Left Date: 19/11/2019

::::::::::

ID: 3
Player ID: 3
Arrived Date: 15/11/2019
Left Date: 19/11/2019

::::::::::

ID: 4
Player ID: 1
Arrived Date: 21/09/2000
Left Date: 30/09/2000

ID: 200
Name: Rui Manuel
Birth Date: 01/10/1969
Function: Masseuse
Salary: 1500

::::::::::

ID: 201
Name: Belmiro Miguel
Birth Date: 06/11/1957
Function: Doctor
Salary: 1200

::::::::::

ID: 202
Name: Antonio Aurelio
Birth Date: 14/05/1974
Function: Wardrobe Director
Salary: 3000

IndividualStatistics.txt

InfCalls.txt

OtherWorkers.txt

ID: 1
Name: Cristiano Ronaldo dos Santos Aveiro
Birth Date: 05/02/1985
Position: Forward
Club: Juventus
Weight: 84
Height: 187
Market Price: 100000000
Days Active: 1573

:::::::::

ID: 2
Name: Bernardo Silva
Birth Date: 10/08/1994
Position: Midfielder
Club: Manchester City
Weight: 64
Height: 173
Market Price: 70000000
Days Active: 479

SoccerPlayers.txt

ID: 1500
Name: Fernando Santos
Birth Date: 10/10/1954
Function: Head coach
Salary: 5000

:::::::::

ID: 1501
Name: Ilidio Vale
Birth Date: 11/04/1951
Function: Assistant Coach
Salary: 7500

:::::::::

ID: 1502
Name: Ricardo Santos
Birth Date: 03/12/1957
Function: Assistant Coach
Salary: 2000

TechnicalTeam.txt

Exceções

Neste projeto foram implementadas as seguintes classes direcionadas para exceções:

- **WrongDateFormat**, que indica o caso de uma data estar mal formatada, quer no que diz respeito ao DD/MM/YYYY quer por exemplo se o utilizador inserir 30 de Fevereiro de 2019.
- **NoObjectFound**, que, tal como indica, é utilizada aquando da procura de um objeto de uma determinada classe que não existe. Esta exceção é fundamental em todos os algoritmos de procura de dados, sobretudo para evitar um acréscimo desnecessário de uso de memória.
- **WrongStatistic**, que compara a “soma” das estatísticas de cada jogador com as globais de um jogo e é indicada no caso de esta comparação ser falsa.

FUNCIONALIDADES IMPLEMENTADAS

Tendo em vista um projeto rigoroso, as diversas funcionalidades sugeridas pelo enunciado foram aplicadas e transportadas para o menu com que o utilizador tem contacto, tendo o mesmo as opções de:

- **Display** : tem como função mostrar os dados de empregados, equipa técnica, jogadores, jogos e convocatórias; (OK)
- **Manage**: realizada o *CRUD* da informação sobre um membro da *National Team* ou sobre uma *Call*; (OK)
- **Calls em que um jogador participou** cingindo o método de pesquisa em: ID, Data de nascimento, nome e salário; (OK)
- **Jogos em que um jogador participou** durante uma *Call*; (OK)
- **Estatísticas de uma *Call***; (OK)
- **Estatísticas de um jogador/equipa numa *Call***; (OK)
- **Custos individuais de um indivíduo, da equipa ou de todo o *Staff*** num dado mês ou totais; (OK)
- **Vitórias, derrotas e empates da *National Team***; (OK)
- **Top Scorer**; (OK)
- **Exit**.

DESTAQUE DE FUNCIONALIDADE

A nossa funcionalidade a destacar é o cálculo dos custos. A opção de cálculo de custos, como foi referido, funciona para: um *Soccer Player*, equipa e a *National Team* como um todo, num dado mês, ou desde sempre. Este cálculo de custos é baseado no atributo *daysActive*, presente na classe *SoccerPlayer* e também no seu valor de mercado (*marketPrice*). Considera-se os dias ativos de um jogador como os dias em que já esteve ao serviço da seleção numa convocatória. Ou seja, a cada convocatória em que um jogador participa, o atributo *daysActive* será incrementado do número de dias que dura a convocatória. Para o cálculo dos custos, é feito o processo inverso. Utilizou-se um algoritmo de ordenação que permite pesquisar as convocatórias da mais recente para a mais antiga de modo a que desta forma, como o cálculo do seguro diário é calculado com base no *daysActive* que o jogador tinha no início da convocatória, se faça a subtração do número de dias que dura convocatória para obter o *daysActive* utilizado para o cálculo da seguro diário. Em caso de lesão do jogador na convocatória, esse valor triplicará. Relativamente aos algoritmos, foi também utilizado o algoritmo de pesquisa *binary search*, sobretudo no ato de procura da existência de um dado *Soccer Player* numa determinada convocatória.

Dificuldades

As principais dificuldades encontradas reverterem para o facto de termos enfrentado um projeto com uma enorme quantidade de dados onde dificultava ver algum erro minucioso que possa ter acontecido. Apesar do grau de dificuldade não ser extremo, essa foi o nosso principal entrave.

Exemplos de Execução