INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

João Pedro Almeida Costa (1231541)

Roberto Oliveira Valente (1231555)

Samuel Oliveira Lemos (1231557)

Dinis Pinto Faryna (1231530)

Gonçalo De Menezes Pacheco Morais De Abreu (1231536)

Jogos Olímpicos Software (Oporto Olympics)

(Plataforma de Gestão de Eventos, Atletas e Resultados)

Laboratório Projeto III

Santa Maria da Feira

2025

Índice

Conteúdo

Introdução

Os Jogos Olímpicos representam um dos eventos deportivos mais prestigiados do mundo, reunindo atletas de diversas nações para competirem em diversas modalidades. A gestão desses eventos exige um alto nível de organização, desde a administração desses eventos e locais de competição até o registo e acompanhamento dos seus participantes.

Nesse contexto, a empresa Oporto Olympics (OpO) identificou a necessidade de um sistema para auxiliar na gestão dos Jogos Olímpicos que ocorrerão nos próximos anos. O objetivo principal desse sistema é fornecer uma plataforma centralizada para administrar os eventos, incluindo a gestão de desportos, equipas, atletas, inscrições e resultados. O sistema também permitirá a importação de dados através de arquivos XML validados por XSD, garantindo integridade e padronização das informações.

O projeto foi desenvolvido por uma equipa de TI utilizando a metodologia Scrum, com ciclos de desenvolvimento, chamados de sprints, permitindo a evolução contínua da aplicação. O sistema foi construído principalmente em Java com o auxílio da base de dados SQL Server para armazenar todos os tipos de informações.

Entre os principais requisitos do sistema, destaca-se a possibilidade de adicionar, editar e consultar informações sobre eventos, modalidades e participantes. Além disso, é possível definir a calendarização das modalidades, garantindo que os locais e horários de competição não entrem em conflito. O sistema também permite que os atletas consultem seu histórico de participação e resultados, além de garantir que o registo e acesso à plataforma sejam intuitivos.

Além da gestão interna dos Jogos Olímpicos, o sistema integra-se com uma API externa para permitir a venda de bilhetes para eventos. Esta funcionalidade permite aos espetadores registarem-se e adquirirem tickets através de uma interface cliente, sendo utilizado a ferramenta Postman para testes.

Para garantir a integridade e confiabilidade dos dados, o sistema conta com funcionalidades de segurança, incluindo senhas encriptadas. Além disso, os arquivos importados são mantidos num histórico para futura consulta, enquanto as imagens associadas aos atletas e eventos serão atualizadas conforme novas versões forem carregadas.

Levantamento de Requisitos e Metodologia SCRUM

Levantamento de Requisitos

O levantamento de requisitos foi realizado com base nas necessidades apresentadas pela empresa Oporto Olympics (OpO), garantindo que todas as funcionalidades essenciais fossem devidamente identificadas e documentadas. Os requisitos foram categorizados em funcionais e não funcionais, assegurando que o sistema atendesse tanto às operações específicas do negócio quanto aos critérios de qualidade e segurança.

1. Requesitos Funcionais

Os requisitos funcionais representam as funcionalidades que o sistema deve fornecer:

- Gestão de Eventos Olímpicos:
  - Criar, editar e excluir eventos olímpicos.
  - Associar um país anfitrião e locais aos eventos.
  - Fechar eventos somente quando todas as modalidades terminarem.

- Gestão de Locais de Competição:
  - Registar locais como recintos interiores ou exteriores.
  - Definir informações como morada, cidade, capacidade e ano de construção.
  - Garantir que um local não esteja ocupado por mais de uma modalidade no mesmo horário.

- Gestão de Modalidades:
  - Criar e configurar modalidades individuais e coletivas.
  - Definir regras, número mínimo de participantes, medidas de pontuação e histórico de vencedores.

- Gestão de Equipas e Atletas:
  - Criar, editar e excluir equipas e atletas.
  - Importar dados através de ficheiros XML validados por XSD.
  - Garantir que atletas pertençam a equipas do seu país.
  - Controlar participações e medalhas de atletas e equipas.

- Inscrição e Aprovação de Atletas:
  - Permitir que atletas se inscrevam em modalidades.
  - O gestor deve aprovar as inscrições coletivas e definir quais atletas participam.
  - Evitar sobreposição de horários na inscrição de atletas.

- Geração e Gestão de Resultados:
  - Criar automaticamente resultados quando uma modalidade inicia.
  - Armazenar históricos de desempenho de atletas e equipas.

- Gestão de Utilizadores e Segurança:
  - Armazenar utilizadores no sistema.
  - Encriptar passwords.
  - Garantir um processo de registo rápido e intuitivo.
  -
- Integração com API Externa:
  - Implementar suporte para compra de bilhetes de eventos.
  - Integrar sistema com API para gestão de bilhetes.

2. Requisitos Não Funcionais

Os requisitos não funcionais garantem que o sistema seja eficiente, seguro e escalável:

- ***Usabilidade***: Interface intuitiva e acessível para utilizadores.
- ***Performance:*** Resposta eficiente nas operações de gestão e consulta.
- ***Segurança:*** Encriptação de dados sensíveis e controlo de acessos.
- ***Interoperabilidade:*** Uso de Java e integração com sistemas externos via API.
- ***Manutenção:*** Código modular e documentado para facilitar expansões futuras.
- ***Armazenamento:*** Base de dados SQL Server para persistência de dados.

Metodologia Scrum

O desenvolvimento do projeto seguiu a metodologia ágil Scrum, que permite uma abordagem iterativa e incremental. A equipa foi organizada de forma a garantir a divisão eficiente das tarefas e a entrega contínua de funcionalidades ao longo do semestre.

1) Estrutura da Equipa

A equipa foi composta por:

- Team Leader: Responsável pela coordenação e alinhamento com os objetivos do cliente.
- Desenvolvedores: Responsáveis pela implementação das funcionalidades.

2) Ciclo de Desenvolvimento

O desenvolvimento ocorreu em sprints de 1 a 2 semanas, seguindo as seguintes etapas:

- ***Sprint Planning:*** Definição das tarefas e objetivos da sprint.
- ***Daily Scrum:*** Reuniões diárias para acompanhamento do progresso e resolução de impedimentos.
- ***Sprint Review:*** Apresentação das funcionalidades desenvolvidas ao final de cada sprint.
- ***Sprint Retrospective:*** Análise das melhorias no processo de desenvolvimento.

3) Artefatos do Scrum

- Product Backlog: Lista de funcionalidades priorizadas a serem implementadas.
- Sprint Backlog: Conjunto de tarefas a serem desenvolvidas em cada sprint.
- Burndown Chart: Monitorização do progresso e desempenho da equipa.

4) Benefícios do Scrum no Projeto

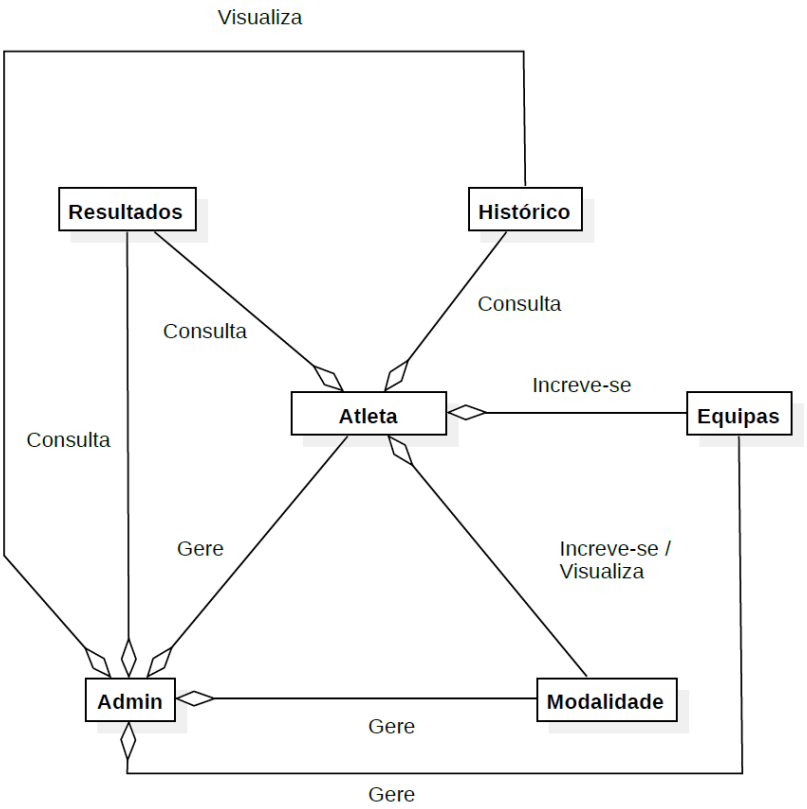
A aplicação da metodologia Scrum proporcionou:

- Melhor adaptação a mudanças nos requisitos.
- Entrega contínua de funcionalidades ao cliente.
- Melhoria na comunicação e colaboração da equipa.
- Acompanhamento constante do progresso e identificação rápida de problemas.

Com esta abordagem, o desenvolvimento da plataforma de gestão dos Jogos Olímpicos ocorreu de forma estruturada e eficiente, garantindo a entrega de um sistema robusto e alinhado às necessidades do cliente.

Análise/Design e comentários da solução

Domain Model



Este diagrama de modelo de domínio representa a estrutura do sistema de gestão dos Jogos Olímpicos, destacando as entidades principais e as relações entre elas.

Entidade e Papéis

- Atleta
  - É a entidade central do sistema.
  - Pode consultar os resultados e o seu histórico.
  - Pode inscrever-se em equipas e em modalidades.

- Resultados
  - Contêm informações sobre as competições que o atleta participou.
  - O atleta pode consultar os resultados das suas participações.
  - O admin pode consultar os resultados gerais.

- Histórico
  - Armazena informações sobre a trajetória do atleta.
  - O atleta pode consultar o seu histórico de participações passadas.

- Equipas
  - Representam grupos de atletas inscritos para competir juntos.
  - O atleta pode inscrever-se em equipas.

- Modalidade
  - Representa os diferentes desportos nos quais os atletas podem competir.
  - O atleta pode inscrever-se e visualizar informações sobre as modalidades.
  - O admin gere as modalidades.
  - Tem um papel de gestão no sistema.
  - Pode gerir: resultados, equipas e modalidades.
  - Também pode consultar os resultados.

Relações e Interações

- **Atleta**
  - Tem uma relação direta com resultados, histórico, equipas e modalidade.
  - Pode realizar consultas e inscrições.
- **Admin**
  - Atua como um gestor do sistema.
  - Pode gerir equipas, modalidades e resultados.
- **Equipas e Modalidades**
  - São entidades associadas à participação dos atletas.
  - Permitem que os atletas se organizem e se inscrevam em competições.

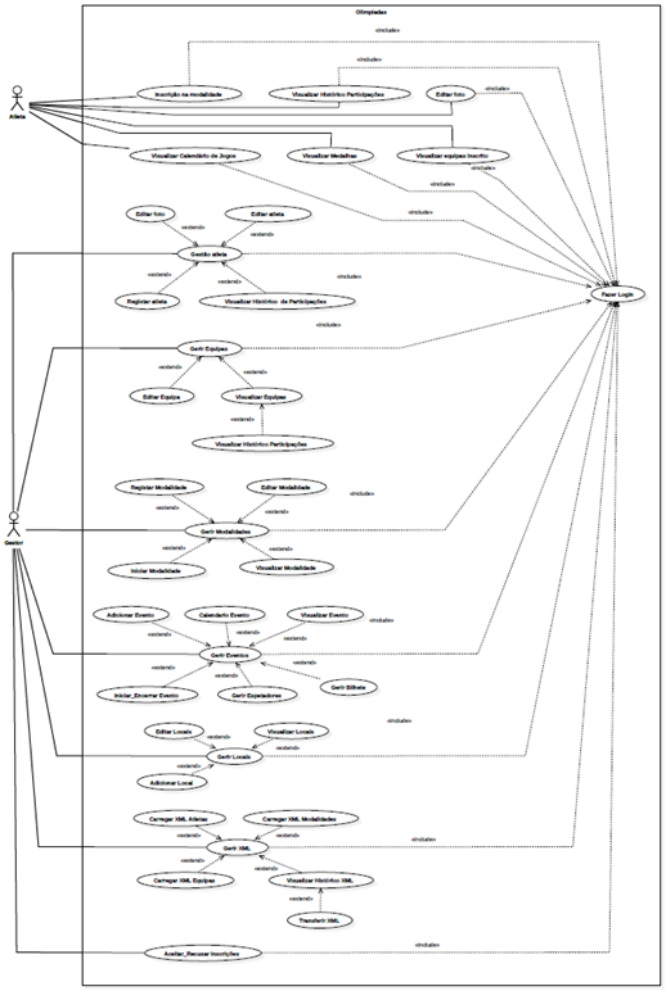
Principais Funcionalidades do Sistema

Consulta de informações: Atletas podem consultar os seus resultados e histórico.

Inscrição em modalidades e equipas: Atletas podem se inscrever em diferentes competições.

Gestão administrativa: O Admin tem controlo sobre modalidades, equipas e resultados.

Use Case Diagram



Atores

- Atleta: Interage com funcionalidades relacionadas ao seu perfil e histórico.
- Gestor: Responsável por gerir atletas, equipas, modalidades, eventos, espectadores, locais e arquivos XML.

Casos de Uso Principais:

- Gestão de Atletas: Registar, editar, visualizar histórico de participações e editar foto do atleta.
- Gestão de Equipas: Editar e visualizar equipas.
- Gestão de Modalidades: Registar, editar, visualizar e iniciar modalidades.
- Gestão de Eventos: Adicionar, visualizar, iniciar/encerrar eventos e gerir calendário de eventos.
- Gestão de Espectadores: Gerir bilhetes.
- Gestão de Locais: Adicionar, editar e visualizar locais.
- Gestão de XML: Carregar e visualizar histórico de arquivos XML relacionados a atletas, modalidades e equipas.

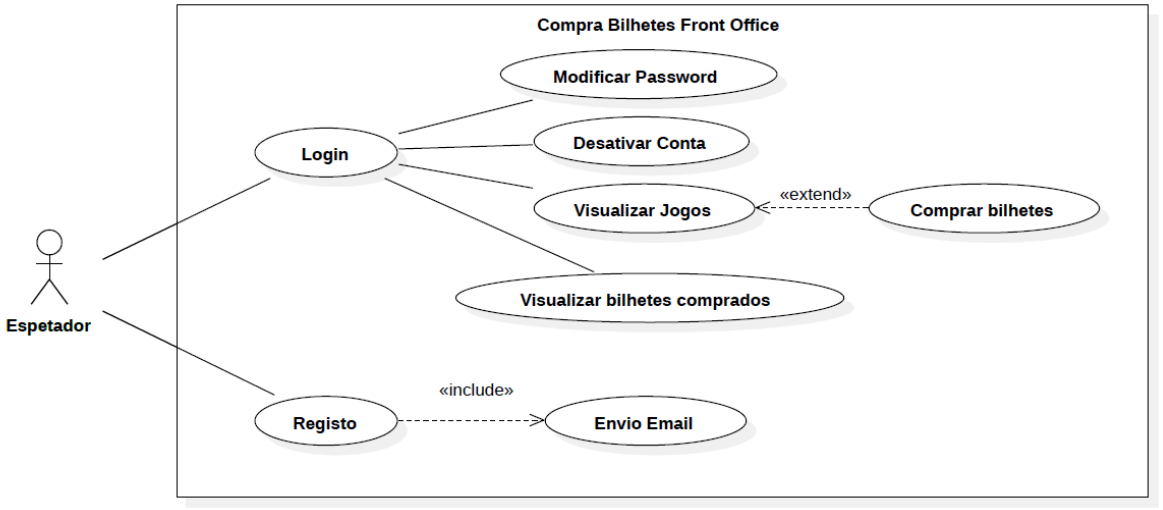
- Visualização de Medalhas: Visualizar medalhas conquistadas.
- Inscrição na Modalidade: Inscrever-se em modalidades e visualizar equipas inscritas.
- Login: Fazer login no sistema.

Relacionamentos:

Include: Indica que um caso de uso inclui a funcionalidade de outro caso de uso.

Extend: Indica que um caso de uso pode ser estendido por outro caso de uso, adicionando funções opcionais.

Use Case Diagram API



Atores

- Espectador: Representa o utilizador principal do sistema, que pode interagir com os casos de uso associados à compra de bilhetes. Interage com os casos de uso de Login, Registo e funcionalidades subsequentes, como Compra de Bilhetes e Visualização de Bilhetes Comprados.

Casos de Uso e suas Relações:

Os casos de uso representam as funcionalidades disponíveis para o espetador dentro do sistema.

- Login
  - Caso de uso central que permite ao utilizador acessar a sua conta.
  - Após o login, o utilizador pode:
  - Modificar Password - Alterar a palavra-passe de acesso.
  - Desativar Conta - Remover sua conta do sistema.
  - Visualizar Jogos - Consultar os eventos disponíveis para compra de bilhetes.
  - Visualizar Bilhetes Comprados - Consultar os seus bilhetes adquiridos para eventos.

- Registo
  - Permite que novos utilizadores criem uma conta no sistema.
  - Inclui a funcionalidade de Envio de Email, ao registar-se, o utilizador recebe um e-mail de confirmação da conta.

- Visualizar Jogos
  - Permite que o espectador veja os jogos disponíveis.
  - Possui uma relação de «extend» com Comprar Bilhetes, o que indica que a compra de bilhetes pode ser uma funcionalidade opcional ao visualizar os jogos.

- Comprar Bilhetes
  - Caso de uso que permite a aquisição de bilhetes para eventos.
  - É acessado diretamente ao visualizar jogos.

Análise das Relações:

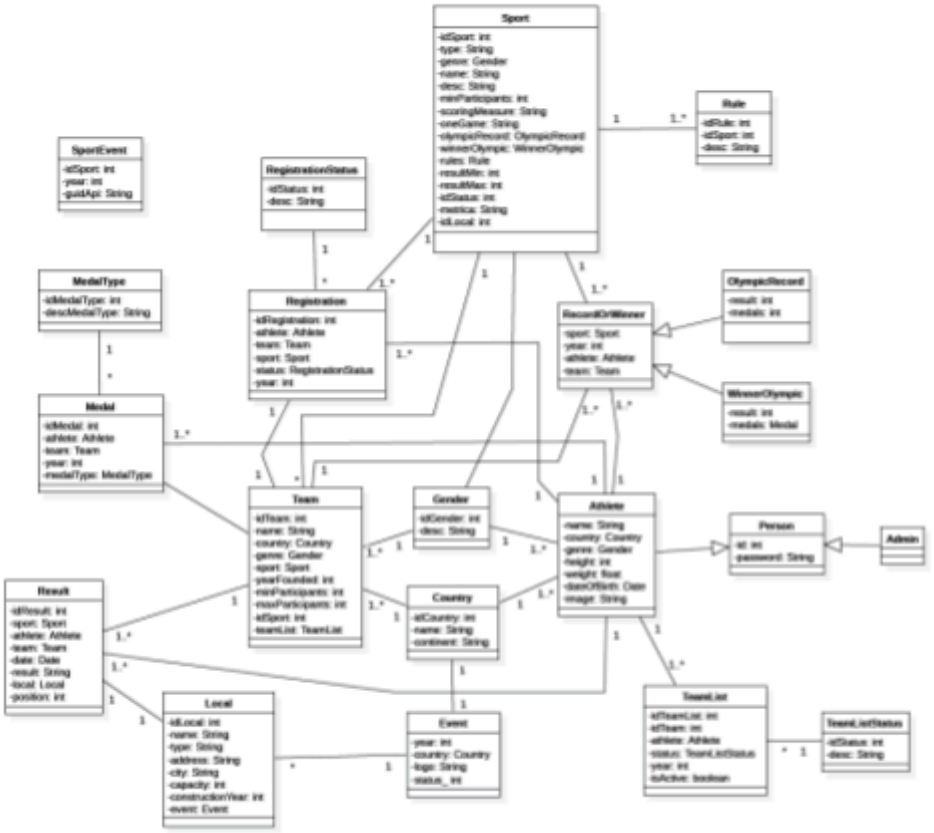
Relação ‘extend’ entre "Visualizar Jogos" e "Comprar Bilhetes"

O uso do relacionamento ‘extend’ significa que "Comprar Bilhetes" é uma funcionalidade opcional, ativada dentro da funcionalidade "Visualizar Jogos", pois primeiro o utilizador deve visualizar os jogos disponíveis antes de comprar bilhetes.

Relação ‘include’ entre "Registo" e "Envio Email"

Indica que o Envio de Email é um passo obrigatório dentro do Registo.

Classes Diagram



Este diagrama de classes modela um sistema para gestão de Jogos Olímpicos, abrangendo entidades como Modalidade (Sport), Atleta (Athlete), Equipa (Team), Inscrição (Registration), Resultado (Result), entre outras.

Principais Classes e seus Atributos

Classes principais

Sport (Modalidade)

Representa um desporto praticado nas Olimpíadas.

Possui atributos como idSport, name, genre (gênero), minParticipants, scoringMeasure e oneGame (tipo de jogo).

Está associada a Rules (Regras), WinnerOlympic (Vencedor Olímpico), OlympicRecord (Recordes Olímpicos).

Athlete (Atleta)

Representa um atleta participante dos jogos (herda de Person(Pessoa)).

Possui atributos como name, country, genre, height, weight, dateOfBirth, image.

Está relacionado a Team (Equipa), Registration (Inscrição) e Result (Resultados).

Team (Equipa)

Representa uma equipa participante.

Possui atributos como idTeam, name, country, genre, yearFounded, minParticipants.

Relaciona-se com Athlete e Sport.

Registration (Inscrição)
Representa a inscrição de um atleta ou equipa num desporto específico.
Relaciona-se com Athlete, Team, Sport e RegistrationStatus (Status da inscrição).

Result (Resultado)
Regista os resultados de uma competição.
Possui atributos como idResult, athlete, team, date, result, position, local.

Classes auxiliares

Rule (Regra)
Define regras específicas para cada modalidade.

OlympicRecord (Recorde Olímpico)
Regista recordes alcançados num desporto.

WinnerOlympic (Vencedor Olímpico)
Relaciona os vencedores olímpicos e suas medalhas.

Medal e MedalType (Medalha e Tipo de Medalha)
Representam as medalhas conquistadas.

Local (Local)
Representa o local onde decorrem os eventos olímpicos.

Event (Evento)
Regista os eventos que ocorrem a cada ano, vinculados a um país.

Country (País)
Representa um país e o seu continente.

Person e Admin
Modela pessoas do sistema, onde Admin herda de Person.

Análise das Relações
Relações entre entidades principais
Um Athlete pode estar associado a uma Team.
Uma Team pode participar de múltiplos Sport.
Um Athlete pode ter múltiplas Registration associadas a diferentes desportos.
Um Result está associado a um Athlete, uma Team e um Local.

Associações entre entidades auxiliares
OlympicRecord e WinnerOlympic registam os recordes e vencedores de cada modalidade.
Medal está associada a Athlete, Team e MedalType.
RegistrationStatus define os estados das inscrições.

Novos diagramas de Sequência (um geral e outro detalhado)

Geral

Registar Atleta em Modalidade

O diagrama de sequência descreve o fluxo de interações entre diferentes componentes do sistema durante o processo de registo de um atleta numa modalidade. Envolve várias entidades, como controladores, DAOs (Data Access Objects), e interfaces.

1. Principais Objetos e Ator

Atleta: Regista-se num desporto.
ViewsController: Gere as telas e interações com a interface gráfica.
LoginController: Responsável pela autenticação do atleta.
AtletaDAO: DAO que acessa os dados do atleta na base de dados.
LoginView: Interface gráfica de login.
AthleteView: Interface do atleta autenticado.
SportRegisterController: Controlador responsável por processar registos de modalidades.
RegistrationDao: DAO que gere o armazenamento dos registos na base de dados.

2. Fluxo do Processo

1. Inserção de Dados: O atleta insere seus dados na interface.
2. Acessa o sistema: A interface encaminha a solicitação ao ViewsController.
3. Evento de Login: handleEntrarButtonAction(ActionEvent event) é acionado no LoginController.
4. Verificação de Login: O LoginController chama loginVerify(id, senha), validando as credenciais.
5. Pesquisa na Base de Dados: getAthleteById(id) é chamado na AtletaDAO.
6. Verificação de Dados: A base de dados processa a busca e retorna os dados do atleta.
7. Autenticação bem-sucedida: O login retorna verdadeiro (Return True).
8. Acesso autorizado: O sistema permite que o atleta continue.
9. Exibição da Interface do Atleta: showAthleteView(ActionEvent event) é acionado para mostrar a interface do atleta autenticado.
10. Solicitação de Inscrição: O atleta solicita a inscrição numa modalidade.
11. Acesso ao sistema de registo: A interface de registo é acionada.
12. Exibição da tela de registo: mostrarRegistaModalidades(ActionEvent event) é chamado para exibir as opções disponíveis.
13. Solicitação de Registo: registerSport(ActionEvent event) inicia o processo de inscrição na modalidade.
14. Armazenamento na BD: addRegistrationSolo(Registration registration) é chamado no RegistrationDao.
15. Validação dos Dados: O DAO valida os dados fornecidos.
16. Retorno dos Dados: Caso os dados sejam válidos, retorna um ID gerado (Return GeneratedID).
17. Confirmação ao Utilizador: O sistema exibe uma mensagem de confirmação.
18. Tratamento de Erros:
    a. Se os dados forem inválidos, retorna -1, indicando falha no registo.
    b. Se ocorrer erro, o sistema exibe uma mensagem de erro ao utilizador.
    c. Retornos como Null ou False indicam falha no processo.

Registar Evento
O diagrama representa o fluxo de execução do processo de adição de um evento no sistema, detalhando a interação entre os diferentes componentes envolvidos.

1. Atores e Classes Envolvidas
Admin: Utilizador que deseja adicionar um evento.
LoginView: Interface gráfica para autenticação do administrador.
ViewsController: Controlador responsável pelas interações com as views.
LoginController: Controlador responsável por validar as credenciais do administrador.
AdminDao: Classe de acesso aos dados do administrador.
DataBase: Base de dados onde as informações são validadas.
Menu: Interface que permite ao administrador selecionar opções, como adicionar um evento.
EventAddController: Controlador responsável pelo processo de adição do evento.
EventDao: Classe de acesso aos dados e inserção de eventos.

2. Fluxo de Execução
Fase 1: Autenticação do Administrador
1. Inserção de Dados - O admin insere as suas credenciais (Id e senha).
2. Acesso à Interface - O LoginView envia os dados ao controlador.
3. Método handleEntrarButtonAction(ActionEvent event) - O evento do botão de login é tratado.
4. Chamada loginVerify(id, senha) - O LoginController verifica as credenciais do utilizador.
5. Busca de Admin getAdminById(id) - O AdminDao consulta a base de dados para validar o utilizador.
6. Validação dos Dados - O sistema valida se o Id e senha correspondem a um administrador válido.
7. Retorno dos Dados - Os dados do administrador são retornados ao controlador.
8. Retorno do Admin - Se o admin for encontrado, retorna os dados.
9. Retorno Verdadeiro - Se a autenticação for bem-sucedida, retorna true.
10. Permissão Garantida - O sistema permite o acesso ao menu.
11. Exibição do Menu - O admin visualiza as opções disponíveis.

Fase 2: Seleção e Redirecionamento
1. Seleção da Opção "Adicionar Evento" - O admin seleciona a opção para adicionar um evento.
2. Redirecionamento - O sistema carrega a interface de adição de evento.

Fase 3: Adição do Evento
1. Inserção de Dados - O admin preenche os dados do evento na interface.
2. Método handleAddEvent(Event event) - O controlador recebe os dados e inicia a validação.
3. Chamada addEvent(Event event) - O controlador chama a DAO para validar os dados.
4. Validação dos Dados - O sistema verifica se os dados do evento são válidos.
5. Retorno dos Dados - Se os dados estiverem corretos, eles são processados.
6. Retorno Verdadeiro - Se a inserção foi bem-sucedida, retorna true.
7. Confirmação - O sistema exibe a mensagem "Evento adicionado com sucesso!".

Fase 4: Tratamento de Erros

1. Retorno Nulo - Se ocorrer erro, o sistema pode não retornar dados.
2. Retorno Falso - Caso os dados sejam inválidos, retorna false.
3. Evento já existe - Se já existir um evento para o mesmo ano, exibe uma mensagem de erro.
4. Retorno Sem Dados - Se o erro for detetado, nenhuma alteração é feita.
5. Retorno Nulo - Se houver inconsistências nos dados, retorna null.
6. Retorno Falso - O sistema informa que a adição falhou.
7. Exibição de Mensagem de Erro - O admin recebe uma notificação sobre o problema.

Detalhado
Visualizar Modalidades
O diagrama de sequência representa o fluxo de execução do processo de visualização de modalidades por um atleta dentro do sistema.

1. Atores e Classes Envolvidas
1.1. Atores e Classes
Atleta: Utilizador que deseja visualizar as modalidades disponíveis.
LoginView: Interface gráfica de login do atleta.
AtletaController: Controlador responsável por autenticar o atleta.
ModalidadeController: Controlador que gere as modalidades.
ModalidadeView: Interface gráfica onde as modalidades são exibidas.
AtletaDAO: DAO que acessa os dados do atleta na base de dados.
ModalidadeDAO: DAO que pesquisa as modalidades da base de dados.

2. Fluxo de Execução
Fase 1: Autenticação do Atleta
1. Inserção de Credenciais – O atleta insere seus dados de login (Id e senha).
2. Envio das Credenciais – O LoginView encaminha as informações ao AtletaController.
3. Solicitação de Autenticação – O controlador solicita a autenticação à AtletaDAO.
4. Acesso à Base de Dados – O AtletaDAO consulta os dados do atleta.
5. Devolução dos Dados – A BD retorna os dados do atleta ao controlador.
6. Resultado do Login – O sistema informa se o login foi bem-sucedido.

Fase 2: Recuperação das Modalidades
1. Solicitação de Dados – O AtletaController solicita ao ModalidadeController os dados das modalidades disponíveis.
2. Acesso aos Dados – O ModalidadeController encaminha a solicitação ao ModalidadeDAO.
3. Consulta das Modalidades – O ModalidadeDAO realiza a busca na base de dados.
4. Verificação dos Dados – A BD verifica a existência de modalidades disponíveis.
5. Retorno dos Dados – O ModalidadeDAO retorna os dados encontrados ao controlador.
6. Fornecimento de Dados – O ModalidadeController encaminha as modalidades para a ModalidadeView.
7. Exibição dos Dados – A ModalidadeView exibe a lista de modalidades.
8. Exibição da Página da Modalidade – O atleta visualiza as modalidades disponíveis.

Fase 3: Tratamento de Erros

- Retorno de Dados Nulos – Se nenhuma modalidade for encontrada, o ModalidadeDAO retorna null.
- Exibição de Mensagem de Erro – O sistema informa ao atleta que não há modalidades disponíveis.
- Exibição da Página – Mesmo sem modalidades disponíveis, a página da interface é exibida.

Comprar Bilhetes API

O diagrama de sequência representa o fluxo de execução do processo de compra de bilhetes através de uma API, desde a autenticação do cliente até a finalização da transação.

1. Atores e Classes Envolvidas

1.1. Ator e Componentes do Sistema

Cliente: Utilizador que deseja comprar um bilhete.
Login.ejs: Página de login do utilizador.
Login.js: Script que gere a autenticação do utilizador.
Home.ejs: Interface do cliente.
Comprar-Bilhete.js: Script responsável por processar a compra do bilhete.

2. Fluxo de Execução

Fase 1: Autenticação do Cliente

1. Acesso à Página – O cliente acessa a interface de login (Login.ejs).
2. Inserção de Credenciais – O utilizador insere os seus dados de autenticação (e-mail e senha).
3. Solicitação de Autenticação – Login.js recebe os dados e os envia para validação.
4. Validação dos Dados – O sistema verifica se as credenciais são corretas (DataBase).
5. Retorno dos Dados – A BD confirma se o login foi bem-sucedido ou não.
6. Permissão de Acesso – Se a autenticação for válida, o sistema permite o acesso ao cliente.
7. Exibição do Menu Inicial – O cliente visualiza as opções disponíveis.

Fase 2: Processo de Compra do Bilhete

1. Seleção da Compra do Bilhete – O cliente escolhe a opção de compra.
2. Solicitação da Compra – Comprar-Bilhete.js recebe a solicitação do cliente.
3. Envio dos Dados – O sistema processa os dados do pedido e os armazena.
4. Armazenamento dos Dados – A base de dados guarda a transação.

Fase 3: Confirmação da Compra

1. Retorno de Sucesso – Se a compra for concluída corretamente, o sistema retorna sucesso.
2. Exibição de Mensagem de Sucesso – O cliente recebe a confirmação da compra.

Fase 4: Tratamento de Erros

Retorno de Insucesso – Se houver falha no processo, o sistema retorna um erro.
Exibição de Mensagem de Erro – O cliente recebe um aviso sobre a falha na compra.
Nenhum Dado Retornado – Se a compra não for processada, o sistema pode não retornar nada.
Mensagem de Erro – O cliente recebe uma notificação informando que a compra falhou.

Modelo Base de Dados

O modelo de base de dados apresentado estrutura a gestão dos Jogos Olímpicos, incluindo eventos, atletas, equipas, modalidades, locais, medalhas e registos de participação.

1. Estrutura Geral e Entidades Principais

A base de dados é composta por várias tabelas relacionadas. As principais entidades incluem:
Eventos Olímpicos (tblEvent): Representa cada edição dos Jogos Olímpicos.
Países (tblCountry): Identifica os países participantes.
Gênero (tblGender): Define os gêneros dos atletas e equipas.
Locais (tblLocal): Representa os locais das competições.
Modalidades (tblSport): Define os desportos e as suas características.
Atletas (tblAthlete): Representa os atletas com os seus dados pessoais e estatísticas.
Equipas (tblTeam): Define as equipas e suas composições.
Inscrições (tblRegistration): Relaciona atletas e equipas às modalidades.
Resultados (tblResult): Armazena os desempenhos dos atletas e equipas.
Recordes Olímpicos (tblOlympicRecord): Mantém registos de recordes nos eventos.
Medalhas (tblMedal e tblMedalType): Regista os medalhistas e o tipo de medalha.

2. Principais Relações

2.1. Relações entre Eventos, Países e Locais

tblEvent (PK: year)
Relacionado com tblCountry (FK1: idCountry), indicando o país anfitrião do evento.
Relacionado com tblLocal (FK1: event), especificando os locais das competições.

2.2. Atletas e Equipas

tblAthlete (PK: idAthlete)
Relacionado a tblCountry (FK1: idCountry), indicando o país de origem.
Relacionado a tblGender (FK2: idGender), especificando o gênero do atleta.

tblTeam (PK: idTeam)
Relacionado a tblCountry (FK1: idCountry), indicando o país da equipa.
Relacionado a tblGender (FK2: idGender), para especificar o gênero das equipas.
Relacionado a tblSport (FK3: idSport), determinando a modalidade praticada.

2.3. Inscrição e Participação

tblRegistration (PK: idRegistration)
Relacionado a tblAthlete (FK1: idAthlete), indicando atletas inscritos.
Relacionado a tblTeam (FK2: idTeam), indicando equipas inscritas.
Relacionado a tblSport (FK3: idSport), especificando a modalidade.
Relacionado a tblEvent (FK5: year), indicando o evento correspondente.

tblRegistrationStatus (PK: idStatus)
Define estados como "Aprovado", "Pendente" ou "Recusado" para as inscrições.

2.4. Resultados e Medalhas

tblResult (PK: idResult)

Relacionado a tblSport (FK1: idSport), especificando a modalidade.
Relacionado a tblAthlete (FK2: idAthlete) e tblTeam (FK3: idTeam), determinando os competidores.
Relacionado a tblLocal (FK4: idLocal), indicando o local da competição.

tblMedal (PK: idMedal)
Relacionado a tblAthlete (FK1: idAthlete) e tblTeam (FK2: idTeam), determinando medalhistas.
Relacionado a tblEvent (FK3: year), associando a edição olímpica.
Relacionado a tblMedalType (FK4: idMedalType), especificando o tipo de medalha.

tblOlympicRecord (PK: idSport, year)
Regista os recordes olímpicos em modalidades específicas.
Relacionado a tblAthlete (FK2: idAthlete) e tblTeam (FK3: idTeam), identificando os recordistas.

2.5. Administração e Regras
tblAdmin (PK: id)
Contém administradores do sistema, com autenticação por senha.

tblRule (PK: idRule)
Relacionado a tblSport (FK1: idSport), armazenando regras específicas de cada modalidade.

Desenvolvimento/Arquitetura/Testes unitários e integração e plano de testes de aceitação de casos de uso

1. Arquitetura do Sistema

1.1 Utilização do Padrão MVC

O sistema foi desenvolvido seguindo o padrão MVC (Model-View-Controller) para garantir a separação de responsabilidades, facilitar a manutenção e permitir escalabilidade futura.

Model (Modelo): Representa os dados da aplicação e as regras de negócio. Aqui foram incluídas as classes principais, como por exemplo Athlete, Sport, Team, Event e OlympicRecord. O modelo interage com a base de dados por meio das classes DAO (Data Access Object), como por exemplo AthleteDao e EventDao, que utilizam JDBC (Java Database Connectivity) para comunicação com o SQL Server.

View (Visão): Responsável pela interface gráfica desenvolvida em JavaFX, garantindo uma experiência intuitiva para os utilizadores.

Controller (Controlador): Atua como intermediário entre a View e o Model, processando entradas recebidas do utilizador e atualizando a interface quando necessário. Classes como MainController e AthleteController foram responsáveis por gerir os eventos da GUI e invocar métodos nos DAOs para manipulação dos dados.
Esta estrutura permite futuras modificações na interface sem impacto na lógica de negócio.

1.2 Interação com a Base de Dados

A base de dados foi modelada seguindo uma estrutura relacional no SQL Server, garantindo integridade e normalização dos dados.
O acesso aos dados foi implementado utilizando DAO (Data Access Object), garantindo uma comunicação eficiente entre a aplicação e a base de dados.
A importação de ficheiros XML foi gerida de forma estruturada, garantindo validação por meio de XSD antes do armazenamento na base de dados.

1.3 Integração com APIs Externas

O sistema foi projetado para consumir uma API que permite a compra de bilhetes para eventos olímpicos.
A integração foi testada utilizando ferramentas como Postman, assegurando que a comunicação entre os sistemas estivesse correta.

2. Desenvolvimento e Metodologia Scrum

O desenvolvimento do projeto seguiu a metodologia Scrum, garantindo um ciclo de desenvolvimento iterativo e incremental.

2.1 Organização da Equipa e Papéis

Scrum Master: Responsável pela remoção de impedimentos e pelo acompanhamento do progresso da equipa.
Product Owner: Definiu os requisitos do cliente e priorizou as funcionalidades no backlog.
Desenvolvedores: Implementaram as funcionalidades conforme definido nos Sprints.

2.2 Sprints e Entregas

Foram realizadas iterações de 1 a 2 semanas, cada uma entregando um incremento funcional do sistema.
O backlog do produto foi constantemente atualizado com user stories priorizadas pelo Product Owner.
As reuniões diárias (Daily Scrums) ajudaram na identificação de impedimentos e ajustes na abordagem do desenvolvimento.

3. Testes Unitários e Integração

A aplicação foi testada em múltiplas camadas, utilizando estratégias de testes unitários. Implementados utilizando JUnit, garantindo que as principais funcionalidades fossem verificadas individualmente.
Foram testadas regras de negócio como:
CRUD para os atletas.
CRUD para as modalidades.
CRUD para as equipas.
Verificação do Login
Geração automática de resultados.

CRUD Atletas

Este teste unitário verifica as operações CRUD (Create, Read, Update, Delete) para a entidade Athlete na base de dados.

Passos do teste:

Criação do Atleta:

- Define os dados do atleta (nome, altura, peso, data de nascimento, etc.).
- Obtém um gênero ("Male") e um país ("Portugal") da base de dados.
- Cria um objeto Athlete e o adiciona à base de dados.
- Recupera o atleta inserido e verifica se os dados estão corretos.

Atualização do Atleta:

- Define novos dados para o atleta, incluindo nome, gênero ("Female") e país ("Angola").
- Atualiza o registo do atleta na BD.
- Recupera o atleta atualizado e verifica se os dados foram modificados corretamente.

Remoção do Atleta:

- Remove o atleta da base de dados e confirma a exclusão.

Método de Comparação (assertEqualsAthlete)

Este método verifica se os atributos do atleta esperado e do atleta recuperado são iguais, comparando ID, nome, altura, peso, data de nascimento, imagem, país e gênero.
Objetivo: Garantir que todas as operações CRUD funcionem corretamente e que os dados do atleta sejam armazenados e recuperados corretamente.

CRUD Modalidades

testCrudSport() - Teste CRUD para Sport

Este teste garante que um desporto pode ser criado, lido, atualizado e removido corretamente na base de dados.

Fluxo do teste:

Criar um desporto(Sport):

- Define os atributos da modalidade (type, gender, name, description, minParticipants, scoringMeasure, oneGame, resultMin, resultMax).
- Adiciona o desporto à base de dados.
- Recupera o desporto pelo Id e verifica se foi inserido corretamente com assertEqualsSport.

Atualizar o desporto:

- Modifica os atributos.
- Atualiza na base de dados.
- Recupera a modalidade atualizada e compara com assertEqualsSport.

Remover a modalidade

- Remove da base de dados.
- Mensagem de sucesso confirma a remoção.

testGetNumberParticipantsSport() - Teste de contagem de participantes

Este teste verifica se o método getNumberParticipantsSport() retorna corretamente o número de atletas registados num desportos específico para um determinado ano.

Fluxo do teste:

Criar um desporto:

- Define os atributos (iguais ao primeiro teste) e insere na base de dados.

Adicionar atletas:

- Obtém um país ("Portugal").
- Cria quatro atletas com diferentes características.
- Adiciona cada atleta na BD.

Registar atletas no desporto:

- Obtém dois tipos de RegistrationStatus:
  - status3: ID 3 ("Aprovado")
  - status4: ID 4 ("Terminado")
- Cria registos de inscrição associando os atletas ao desporto e ao ano.
- Adiciona os registos à BD.

Testar o método getNumberParticipantsSport():

- Define o número esperado.
- Chama sportDao.getNumberParticipantsSport(sportAdded.getIdSport(), year).
- Compara o número retornado com assertEquals(expectedNumber, numberParticipants).

Remover dados para limpeza:

- Remove os registos de participação.
- Remove os atletas criados.
- Remove o desporto.

assertEqualsSport() - Método de comparação entre objetos Sport
Este método auxilia os testes verificando se dois objetos Sport são idênticos após operações na base de dados.
Comparações realizadas:

- ID do desporto
- Tipo (Individual ou Coletivo)
- Nome
- Descrição
- Mínimo de participantes
- Sistema de pontuação
- Estrutura de jogo
- Gênero (Male/Female)

Se qualquer um desses valores for diferente, o teste falha e exibe uma mensagem indicando o erro.

CRUD Equipas
Este teste verifica se uma Team pode ser criada, lida, atualizada e removida corretamente na base de dados, além de verificar a consistência dos dados após cada operação.

Fluxo do teste:
Criar um desporto:

- Define os atributos da modalidade (type, gender, name, description, minParticipants, scoringMeasure, oneGame, resultMin, resultMax).
- Adiciona o desporto à base de dados.
- Recupera o desporto pelo ID e verifica se foi inserido corretamente com assertEqualsSport.

Criar uma equipa:

- Define os atributos da equipa (name, country, gender, sport, yearFounded, minParticipants, maxParticipants).
- Adiciona a equipa à base de dados.
- Recupera a equipa pelo ID e verifica se foi inserida corretamente com assertEqualsTeam.

Atualizar a equipa:

- Modifica os atributos da equipa (name, yearFounded, minParticipants, maxParticipants).
- Atualiza na base de dados.
- Recupera a equipa atualizada e compara com assertEqualsTeam.

Remover a equipa:

- Remove a equipa da base de dados.
- Mensagem de sucesso confirma a remoção.

Remover o desporto:

- Remove a modalidade da base de dados.
- Mensagem de sucesso confirma a remoção.

Validação com assertEqualsTeam(expected, actual)
Este método compara se dois objetos Team são idênticos após operações na BD.
Comparações realizadas:

- idTeam: ID da equipa
- name: Nome da equipa
- yearFounded: Ano de fundação
- minParticipants: Mínimo de participantes
- maxParticipants: Máximo de participantes

Além disso, valida se os objetos relacionados não são nulos e que seus atributos correspondem:

- Country: Verifica se o país da equipa é o mesmo.
- Gender: Confirma se o gênero é o esperado.
- Sport: Garante que a equipa continua associada ao mesmo desporto.

Se alguma dessas verificações falhar, o teste exibe uma mensagem indicando a inconsistência.

Verificação do Login
Estes testes garantem que o sistema de login verifica corretamente as credenciais do atleta.

Teste de Login com Credenciais Válidas (testLoginTrue):
Fluxo:

- Simula um login com um ID de atleta (idAtleta = 1001) e senha correta (password = "1001").
- Chama o método loginVerify do LoginController.
- Verifica se o login foi bem-sucedido com assertTrue(loginSucess).

Teste de Login com Credenciais Inválidas (testLoginFalse):
Fluxo:

- Simula um login com um ID de atleta inexistente ou senha incorreta (idAtleta = 1009, password = "password123").
- Chama o método loginVerify do LoginController.
- Verifica se o login falhou com assertFalse(loginSucess).

Geração Automática de Resultados

Geração Automática de Resultados

testIniciarModalidadesIndividual() - Teste de Início de Modalidades Individuais

Este teste verifica o processo completo de inicialização de uma modalidade individual, desde a criação do desporto e registro de atletas até a atribuição de medalhas e remoção de dados.

Fluxo do Teste:

Criação da Modalidade (Sport):

- Define os atributos do desporto (type, gender, name, description, minParticipants, scoringMeasure, oneGame, resultMin, resultMax, idStatus, metrica, dataInicio, dataFim, local).
- Adiciona o desporto à base de dados.
- Recupera o desporto pelo ID e verifica se foi inserido corretamente usando assertEqualsSport.

Adição de Atletas:
- Obtém o país "Portugal" e o gênero "Male".
- Cria cinco atletas com diferentes características.
- Adiciona cada atleta à base de dados.
- Encripta as senhas dos atletas para garantir consistência na comparação.
- Verifica se os atletas foram inseridos corretamente usando assertEqualsAthlete.

Registo dos Atletas:
- Obtém o status de registro com ID 3 ("Aprovado").
- Cria registros de inscrição associando os atletas à modalidade e ao ano.
- Adiciona os registros à base de dados.
- Verifica se cada atleta possui um registro associado.

Início da Modalidade:
- Inicia a modalidade usando o método sportStart.
- Verifica se o início foi bem-sucedido com assertTrue.

Registo dos Atletas:
- Obtém o status de registro com ID 3 ("Aprovado").
- Cria registros de inscrição associando os atletas à modalidade e ao ano.
- Adiciona os registros à base de dados.
- Verifica se cada atleta possui um registro associado.

Verificações Testes:
- Verifica se os registos foram adicionados
- Verifica se os resultados foram atribuídos

- Verifica se as medalhas foram atribuídas
- Verifica se o WinnerOlympic foi atribuído.

Limpeza dos Dados:

- Remove os resultados da base de dados.
- Remove os registros de inscrição.
- Remove as medalhas atribuídas.
- Remove o registro do vencedor olímpico.
- Remove os atletas criados.
- Remove o desporto criado.

assertEqualsSport() - Método de Comparacão entre Objetos Sport:
- Compara atributos como ID, tipo, nome, descrição, mínimo de participantes, sistema de pontuação, estrutura de jogo e gênero.
- Falha o teste se houver discrepâncias, indicando o erro.

assertEqualsAthlete() - Método de Comparacão entre Objetos Athlete:
- Verifica atributos como ID, nome, país, gênero, altura, peso, data de nascimento e senha encriptada.
- Assegura que os dados inseridos estão corretos após operações na base de dados.

Este teste garante que o ciclo de vida de uma modalidade individual e seus atletas está sendo tratado corretamente pela aplicação, desde a criação até a remoção final dos dados.

testIniciarModalidadesCollective () - Teste de Início de Modalidades Coletivo

Este teste verifica o processo completo de inicialização de uma modalidade coletiva, desde a criação do desporto e registo de atletas e equipas até a atribuição de medalhas, winnerOlympic, resultados e remoção de dados.

Fluxo do Teste:

Criação da Modalidade (Sport):

- Define os atributos do desporto (type, gender, name, description, minParticipants, scoringMeasure, oneGame, resultMin, resultMax, idStatus, metrica, dataInicio, dataFim, local).
- Adiciona o desporto à base de dados.
- Recupera o desporto pelo ID e verifica se foi inserido corretamente usando assertEqualsSport.

Adição de Equipas:
- Obtem os países “Australia”, “Brazil”, “Portugal” e “United States of America” e o gênero “Male”.
- Cria quatro equipas com diferentes características.
- Adiciona cada equipa à base de dados.
- Verifica se as equipas foram inseridas corretamente com getTeamById()

Adição de Atletas:

- Obtém os países “Australia”, “Brazil”, “Portugal” e “United States of America”e o gênero "Male".
- Cria oito atletas com diferentes características (2 para cada país).
- Adiciona cada atleta à base de dados.
- Encripta as senhas dos atletas para garantir consistência na comparação.
- Verifica se os atletas foram inseridos corretamente usando assertEqualsAthlete.

Registo dos Atletas:

- Obtém o status de registro com ID 3 ("Aprovado").
- Cria registos de inscrição associando os atletas à modalidade e ao ano.
- Adiciona os registos à base de dados.
- Verifica se cada atleta possui um registro associado.

Início da Modalidade:

- Inicia a modalidade usando o método sportStart.
- Verifica se o início foi bem-sucedido com assertTrue.

Verificações Testes:

- Verifica se os registos foram adicionados
- Verifica se os resultados foram atribuídos
- Verifica se as medalhas foram atribuídas
- Verifica se o WinnerOlympic foi atribuído.

Limpeza dos Dados:

- Remove os resultados da base de dados.
- Remove os registos de inscrição.
- Remove as medalhas atribuídas.
- Remove o registro do vencedor olímpico.
- Remove os atletas criados.
- Remove o desporto criado.

assertEqualsSport() - Método de Comparacão entre Objetos Sport:

- Compara atributos como ID, tipo, nome, descrição, mínimo de participantes, sistema de pontuação, estrutura de jogo e gênero.
- Falha o teste se houver discrepâncias, indicando o erro.

assertEqualsAthlete() - Método de Comparacão entre Objetos Athlete:

- Verifica atributos como ID, nome, país, gênero, altura, peso, data de nascimento e senha encriptada.
- Assegura que os dados inseridos estão corretos após operações na base de dados.

Este teste garante que o ciclo de vida de uma modalidade coletiva e seus atletas está a ser tratado corretamente pela aplicação, desde a criação até a remoção final dos dados.

Os últimos dois testes baseiam-se neste dois primeiros, apenas serão gerados múltiplos resultados em vez de apenas um por atleta ou equipa.

Conclusão

Desafios e Soluções Encontradas

Durante o desenvolvimento, diversos desafios foram encontrados e superados através de soluções eficientes:

Importação e Validação de XML:

Implementação de validação XSD para garantir a integridade dos ficheiros importados.

Armazenamento dos ficheiros para auditoria e futuras consultas.

Gestão de Conflitos de Horário em Modalidades:

Desenvolvimento de um sistema de verificação que impede inscrições em modalidades com horários coincidentes, incluindo um intervalo de segurança de duas horas antes e depois de cada prova.

Geração Automática de Resultados:

Algoritmo desenvolvido para calcular os resultados conforme a medida de pontuação de cada modalidade (tempo, pontos ou distância), garantindo aleatoriedade e realismo.

Integração com API para Compra de Bilhetes:

Testes extensivos e documentação detalhada para garantir comunicação confiável com a API externa.

Perspectivas Futuras

Embora a aplicação tenha atingido os requisitos iniciais, existem diversas oportunidades de melhoria e expansão:

Desenvolvimento de uma plataforma Web utilizando .Net ou PHP, permitindo que espectadores consultem informações e comprem bilhetes diretamente.

Aprimoramento do Sistema de Estatísticas, permitindo análises detalhadas sobre desempenhos de atletas e países.

Suporte a Notificações e Alertas, para informar atletas sobre novas inscrições, alterações de horários e resultados.

Melhoria na Interface Gráfica (JavaFX), proporcionando uma experiência mais fluida e interativa.

Conclusão Final

O desenvolvimento da solução para a Oporto Olympics (OpO) resultou numa aplicação robusta, modular e escalável, garantindo a gestão eficiente dos Jogos Olímpicos futuros. O uso de Java, SQL Server e MVC proporcionou uma arquitetura bem estruturada, capaz de lidar com grandes volumes de dados e futuras expansões. A abordagem Scrum permitiu um desenvolvimento ágil e colaborativo, assegurando que os requisitos do cliente fossem atendidos de forma incremental e eficaz.

Com as funcionalidades implementadas e testadas, a aplicação está pronta para ser utilizada na administração dos eventos olímpicos, garantindo transparência, organização e eficiência na gestão das competições. A estrutura desenvolvida possibilita a implementação de novos recursos sem impactar a base do sistema, tornando a solução adaptável às futuras necessidades da empresa. Assim, a Oporto Olympics dispõe agora de uma ferramenta inovadora e tecnológica para a organização dos Jogos Olímpicos, alinhada com as melhores práticas da engenharia de software.