

**GameOn   
Trabalho Prático – 1ª Fase**

|  |  |
| --- | --- |
|  | Daniel Caseiro |
|  | José Gomes  Henrique Fonte |

|  |  |
| --- | --- |
| Orientadores | Afonso Remédios |
|  | Nuno Leite  Walter Vieira |

Relatório GameOn (1ª Fase) realizado no âmbito de Sistemas de Informação,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/2023

Maio de 2023

**Instituto Superior de Engenharia de Lisboa**  
Licenciatura em Engenharia Informática e de Computadores

**GameOn   
Trabalho Prático – 1ª Fase**

|  |  |
| --- | --- |
| 46052 | Daniel André Caseiro |
| TODO | José Gomes |
| 48295 | Henrique Fontes |

|  |  |
| --- | --- |
| Orientadores: | Afonso Remédios |
|  | Nuno Leite  Walter Vieira |

Relatório GameOn (1ª fase) realizado no âmbito de Sistemas de Informação,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/2023

Maio de 2023

# Resumo

O projeto consiste na criação de um sistema de gestão de jogos, onde é possível criar jogadores, jogos, partidas, atribuir crachás, iniciar conversas e outras funcionalidades relacionadas.

Os resultados mais importantes foram a implementação do modelo de dados, incluindo todas as restrições de integridade, e a criação do código PL/pgSQL que permite criar o modelo físico, remover o modelo físico e preencher a base de dados. Também foram criadas funções e procedimentos armazenados que permitem manipular os dados da base de dados de forma eficiente e intuitiva.

Além disso, foi criada uma vista que permite aceder à informação sobre identificador, estado, email, username, número total de jogos em que participou, número total de partidas em que participou e número total de pontos que já obteve de todos os jogadores cujo estado seja diferente de “Banido”. E também foram implementados mecanismos para a atribuição automática de crachás e para colocar jogadores no estado “Banido”.

Conclui-se que a criação de um sistema de gestão de jogos é uma tarefa complexa, mas com a implementação adequada do modelo de dados e do código PL/pgSQL, é possível criar um sistema eficiente e escalável. Além disso, a criação de funções e procedimentos armazenados torna a manipulação dos dados da base de dados mais fácil e intuitiva. Por fim, a implementação de mecanismos automáticos ajuda a garantir a integridade dos dados da base de dados.

**Palavras-chave:** funções; base de dados; chat; crachá; integridade; jogador; PL/pgSQL; pontos; procedimento armazenado; restrições; SQL; vista.

# Abstract

The project consists of creating a game management system, where it is possible to create players, games, matches, assign badges, start conversations and other related functionalities.

The most important results were the implementation of the data model, including all integrity constraints, and the creation of PL/pgSQL code that allows creating the physical model, removing the physical model and populating the database. Functions and stored procedures were also created to manipulate the database data efficiently and intuitively.

In addition, a view was created that allows accessing information about the identifier, state, email, username, total number of games played, total number of matches played, and total number of points obtained from all players whose state is different from "Banned". Mechanisms were also implemented for the automatic assignment of badges and for placing players in the "Banned" state.

It is concluded that creating a game management system is a complex task, but with the appropriate implementation of the data model and PL/pgSQL code, it is possible to create an efficient and scalable system. In addition, the creation of functions and stored procedures makes manipulating the database data easier and more intuitive. Finally, the implementation of automatic mechanisms helps to ensure the integrity of the database data.

**Keywords:** database; chat; badge; integrity; player; PL/pgSQL; points; stored procedure; constraints; SQL; view.

# Agradecimentos

**Índice**

[Resumo v](#_Toc417484088)

[Abstract vii](#_Toc417484089)

[Agradecimentos ix](#_Toc417484090)

[Lista de Figuras xiii](#_Toc417484091)

[Lista de Tabelas xv](#_Toc417484092)

[1. Introdução 1](#_Toc417484093)

[1.1 Modelo de Dados 1](#_Toc417484094)

[1.2 A segunda secção deste capítulo 1](#_Toc417484095)

[1.2.1 A primeira sub-secção desta secção 2](#_Toc417484096)

[1.2.2 A segunda sub-secção desta secção 2](#_Toc417484097)

[1.3 Organização do documento 2](#_Toc417484098)

[2. Formulação do Problema 3](#_Toc417484099)

[2.1 Nome da secção deste capítulo 3](#_Toc417484100)

[2.2 Análise do problema - enumeração 3](#_Toc417484101)

[2.3 Outro problema - tabela 4](#_Toc417484102)

[2.4 Expressões matemáticas 4](#_Toc417484103)

[2.5 Figuras de grande dimensão 4](#_Toc417484104)

[3. Solução Proposta - Grandes Ideias 7](#_Toc417484105)

[3.1 Nome da primeira secção deste capítulo 7](#_Toc417484106)

[3.2 A segunda secção deste capítulo 8](#_Toc417484107)

[3.2.1 A primeira sub-secção desta secção 8](#_Toc417484108)

[3.2.2 A segunda sub-secção desta secção 8](#_Toc417484109)

[3.3 Descrição detalhada da solução 8](#_Toc417484110)

[4. Avaliação Experimental 11](#_Toc417484111)

[4.1 Nome da primeira secção deste capítulo 11](#_Toc417484112)

[4.2 A segunda secção deste capítulo 11](#_Toc417484113)

[4.2.1 A primeira sub-secção desta secção 11](#_Toc417484114)

[4.2.2 A segunda sub-secção desta secção 11](#_Toc417484115)

[4.3 Análise de resultados 11](#_Toc417484116)

[5. Conclusões 13](#_Toc417484117)

[Referências 14](#_Toc417484118)

[A.1 Diagramas da Aplicação 15](#_Toc417484119)

[A.2 Modelos de dados 17](#_Toc417484120)

# 

# Lista de Figuras

[Figura 1 - Legenda da figura com o logotipo do ISEL. 1](#_Toc416101905)

[Figura 2 - Legenda da figura com o logotipo do ISEL – versão 2. 5](#_Toc416101906)

[Figura 3 – Diagrama de casos de utilização. 15](#_Toc416101907)

# Lista de Tabelas

[Tabela 1 -Um exemplo de legenda de tabela. Prazos de entrega de Projecto e Seminário, 4](#_Toc416101908)

# Introdução

## Este documento apresenta o projeto de criação de um sistema de gestão de jogos, que tem como objetivo facilitar a organização e administração de jogos e partidas. O sistema foi desenvolvido utilizando a linguagem PL/pgSQL e a base de dados PostgreSQL, e inclui funcionalidades como a criação de jogadores, jogos e partidas, a atribuição de crachás, a gestão de conversas e outras funcionalidades relacionadas. O projeto inclui a implementação do modelo de dados, a criação do código PL/pgSQL que permite criar, remover e preencher a base de dados, bem como a criação de funções e procedimentos armazenados que facilitam a manipulação dos dados da base de dados. Este documento apresentará o modelo de dados, o código PL/pgSQL, as funções e procedimentos armazenados, bem como os resultados obtidos durante o desenvolvimento do projeto.

## 1.1 Modelo de Dados

Um modelo de dados é uma representação lógica e estruturada de como os dados são organizados e armazenados em uma base de dados. Ele é usado para descrever os dados e as relações entre eles, de forma a garantir que os dados sejam armazenados de maneira eficiente e consistente.

### 1.1.1 Modelo de Dados - Conceptual

O modelo de dados conceptual é uma representação abstrata de alto nível dos dados e das relações entre eles, independentemente da implementação física. Ele é útil para entender as necessidades de negócios e para a comunicação entre os utilizadores finais e os projetistas de banco de dados.

### 1.1.2 Modelo de Dados - Relacional

O modelo de dados relacional é uma implementação física do modelo de dados conceptual que descreve como as tabelas, os campos e as relações entre as tabelas são criados e armazenados em um banco de dados relacional. Ele utiliza a linguagem SQL para criar, modificar e consultar a base de dados. Além disso, as restrições de integridade são regras definidas na base de dados para garantir que os dados sejam consistentes e precisos. Elas incluem restrições de chave primária, chave estrangeira, integridade referencial, restrições de domínio, entre outras.

## 1.2 Por definir

Na segunda secção deste capítulo, vamos abordar o enquadramento, o contexto e as funcionalidades.

### 1.2.1 A primeira sub-secção desta secção

As sub-secções são úteis para mostrar determinados conteúdos de forma organizada. Contudo, o seu uso excessivo também não contribui para a facilidade de leitura do documento.

### 1.2.2 A segunda sub-secção desta secção

Esta é a segunda sub-secção desta secção, a qual termina aqui.

## 1.3 Organização do documento

O restante relatório encontra-se organizado da seguinte forma.

# Formulação do Problema

A formulação do problema é um passo crucial na elaboração deste projeto. É importante definir claramente qual é o objetivo do sistema de gestão de jogos e quais são os desafios que devem ser enfrentados para atingir esse objetivo.

O problema principal que o sistema deve resolver é a gestão de jogadores, jogos e partidas. Isso inclui criar, editar e remover informações dos jogadores, jogos e partidas, bem como atribuir crachás a jogadores e iniciar conversas entre jogadores. Além disso, é importante garantir a integridade dos dados da base de dados, impedindo que informações incorretas ou incompletas sejam inseridas.

Ao definir claramente os desafios a serem enfrentados, podemos trabalhar em soluções eficazes para resolver esses problemas e desenvolver um sistema de gestão de jogos eficiente, seguro e escalável.

## 2.1 Construção do Modelo Físico

A construção do modelo físico é uma fase crucial do projeto de um sistema de gestão de base de dados. O modelo físico é a representação da estrutura da base de dados em termos de tabelas, colunas, tipos de dados e relacionamentos entre tabelas.

Para começar, foram analisados os requisitos do sistema e definidas as entidades e atributos necessários para suportar as funcionalidades do sistema. Com base nesta análise, foi criado o modelo de dados conceptual.

A partir do modelo de dados conceptual, foi criado o modelo de dados relacional, onde as entidades e os relacionamentos foram transformados em tabelas e chaves estrangeiras, respetivamente. Além disso, foram definidas as restrições de integridade necessárias para garantir a consistência e a validade dos dados na base de dados.

Para a implementação do modelo físico, foi utilizada a linguagem SQL para criar as tabelas, definir as chaves primárias e estrangeiras, bem como as restrições de integridade. Foi também criado o código PL/pgSQL para automatizar a criação, remoção e preenchimento da base de dados com dados fictícios para fins de teste.

Por fim, foi realizada uma verificação da integridade dos dados e da performance do sistema através da execução de testes e consultas à base de dados. Qualquer problema detetado foi resolvido através de ajustes no modelo físico e no código PL/pgSQL.

Em resumo, a construção do modelo físico foi uma etapa crítica do projeto, pois permitiu definir a estrutura da base de dados e garantir a integridade dos dados.

## 2.2 Análise do problema

Ao longo do desenvolvimento deste projeto, foram identificados diversos problemas e desafios que exigiram uma abordagem cuidadosa e rigorosa. Desde a modelação do esquema de base de dados, passando pela implementação de funções e procedimentos armazenados, até à criação de mecanismos automáticos de atribuição de crachás e banimento de jogadores, o trabalho enfrentou diversas dificuldades que exigiram soluções criativas e bem fundamentadas. Além disso, a criação de um script autónomo de testes, capaz de verificar o correto funcionamento de todas as funcionalidades em cenários normais e de erro, também se revelou um desafio importante. Neste contexto, este projeto representa um caso concreto de aplicação prática de conceitos de modelação de dados e programação em SQL, ilustrando de forma realista os desafios que se colocam a equipas de desenvolvimento em projetos deste tipo.

## 2.2.1 Análise do problema - Funções

Neste capítulo do relatório, será apresentada uma análise das funções desenvolvidas para o sistema de gestão de jogos. As funções em questão são a totalPontosJogador, totalJogosJogador e pontosJogoPorJogador.

A função totalPontosJogador recebe o identificador de um jogador como parâmetro e retorna o número total de pontos obtidos por esse jogador em todos os jogos em que participou. Essa função é importante para obter informações sobre o desempenho do jogador e para classificar os jogadores por pontos.

A função totalJogosJogador, por sua vez, recebe o identificador de um jogador como parâmetro e retorna o número total de jogos diferentes em que o jogador participou. Essa função é útil para saber quantos jogos um jogador já jogou e para avaliar a experiência do jogador no sistema.

Por fim, a função pontosJogoPorJogador recebe a referência de um jogo como parâmetro e retorna uma tabela com duas colunas: o identificador do jogador e o total de pontos que o jogador obteve no jogo. Essa função é importante para obter informações detalhadas sobre o desempenho dos jogadores em jogos específicos e para comparar o desempenho de diferentes jogadores em um mesmo jogo.

Nas funções (e), (f) e (g) foi estabelecido que não era permitido utilizar a informação das tabelas Estatistica\_jogador e Estatistica\_jogo. Esta restrição adicionou uma camada extra de desafio na resolução do problema, uma vez que a informação necessária para calcular os resultados esperados estava distribuída por várias tabelas. Para solucionar esse desafio, foi necessário possuir um conhecimento mais profundo do modelo de dados e das relações entre as diferentes entidades. Apesar disso, todas as funções foram implementadas com sucesso e fornecem resultados precisos e eficientes.

## 2.2.2 Análise do problema - Procedimentos

Este capítulo do relatório aborda a análise dos procedimentos armazenados criados para o projeto. Foram desenvolvidos quatro procedimentos com diferentes funcionalidades e características. O objetivo destes procedimentos é oferecer aos utilizadores uma forma fácil e eficiente de interagir com o sistema, executando operações complexas e interdependentes de forma transparente.

O procedimento armazenado "associarCrachá" permite atribuir um crachá a um jogador, desde que este cumpra as condições necessárias para o obter. O procedimento recebe três parâmetros, o identificador do jogador, a referência do jogo e o nome do crachá desse jogo.

O "iniciarConversa" é um procedimento que permite iniciar uma nova conversa no sistema, associando automaticamente o jogador à conversa e criando uma mensagem a informar que o jogador criou a conversa. O procedimento recebe como parâmetros o identificador do jogador e o nome da conversa e devolve o identificador da conversa criada num parâmetro de saída.

O "juntarConversa" é um procedimento que permite que um jogador se junte a uma conversa existente no sistema, criando uma mensagem a informar que o jogador entrou na conversa. O procedimento recebe como parâmetros o identificador do jogador e o identificador da conversa.

Por último, o procedimento "enviarMensagem" permite que um jogador envie uma mensagem para uma conversa existente no sistema, associando-a ao jogador também indicado. O procedimento recebe como parâmetros o identificador do jogador, o identificador da conversa e o texto da mensagem.

É importante destacar que cada procedimento foi implementado com o nível de isolamento transacional mais adequado e com um tratamento de exceções único para cada um dos casos. Todos eles implementados com sucesso e fornecem resultados precisos e eficientes.

## 2.2.3 Análise do problema - Vista

O objetivo desta alínea é criar uma vista denominada "jogadorTotalInfo" que permita aceder à informação sobre os jogadores, incluindo o seu identificador, estado, email, nome de utilizador, número total de jogos em que participou, número total de partidas em que participou e o número total de pontos que já obteve. A vista deve incluir apenas informações de jogadores cujo estado seja diferente de "Banido". Além disso, deve ser implementada a lógica para cálculo desses campos diretamente na vista, sem a necessidade de aceder às tabelas de estatísticas. Isso significa que a vista deve ser capaz de calcular o número total de jogos e partidas em que cada jogador participou e o número total de pontos que cada jogador já obteve, com base nas informações das outras tabelas do modelo de dados. A utilização de vistas permite uma visualização mais clara e simplificada dos dados e pode ser útil em várias situações, como na criação de relatórios e análises.

## 2.2.4 Análise do problema - Triggers

A atribuição de crachás a jogadores após a conclusão de uma partida é uma tarefa que pode ser complexa de gerir manualmente, especialmente em jogos com muitos jogadores e crachás diferentes. Por esta razão, foi criado um mecanismo automático para atribuição de crachás. Este mecanismo é acionado após o término de cada partida e, se o jogador cumprir as condições necessárias para receber o crachá, o mesmo é atribuído automaticamente.

A funcionalidade de banimento de jogadores foi implementada para permitir que jogadores que violem as regras sejam imediatamente banidos do sistema, sem a necessidade de intervenção manual por parte do administrador. A execução da instrução DELETE na vista jogadorTotalInfo é a ação que aciona este mecanismo. Quando um jogador é banido, ele é automaticamente colocado no estado "Banido" e não poderá mais participar em jogos ou conversas.

A implementação desses mecanismos foi realizada com atenção aos níveis de isolamento transacional adequados e ao tratamento de exceções para garantir a integridade e segurança do sistema. Além disso, foram realizados testes para verificar a eficácia e robustez dos mecanismos em diferentes cenários de uso.

## 2.2.4 Análise do problema - Testes

O script de testes é uma ferramenta muito importante na validação do sistema desenvolvido, pois permite verificar se as funcionalidades implementadas funcionam corretamente em cenários normais e também em situações de erro.

Os testes foram realizados para as funcionalidades de 2d a 2n, de acordo com as especificações do projeto. Foram considerados casos normais, onde as funções e procedimentos deveriam funcionar corretamente, e casos de erro, onde situações inesperadas poderiam ocorrer.

Cada teste foi projetado para abordar todos os cenários possíveis, de forma a garantir a robustez e confiabilidade do sistema. Para isso, foram utilizados dados de teste representativos, bem como dados inválidos que poderiam levar a erros ou exceções.

Ao ser executado, o script de testes lista o nome de cada teste e indica se ele foi executado com sucesso ou não. Dessa forma, é possível identificar rapidamente quais as funcionalidades que não estão funcionando corretamente e corrigir quaisquer problemas encontrados.

## 2.3 Problemas Adicionais

OPCIONAL PODE SER RETIRADO / POR PREENCHER

# Solução Proposta - Grandes Ideias

A nossa proposta de solução para este projeto foi orientada para alcançar a máxima eficiência e eficácia possível na resolução dos diversos desafios colocados. Para isso, utilizamos um conjunto de boas práticas e metodologias adequadas, tendo em conta as particularidades de cada problema. Dessa forma, conseguimos desenvolver um modelo de dados robusto e coerente, que nos permitiu implementar as funcionalidades solicitadas com segurança e escalabilidade. Além disso, trabalhamos de forma colaborativa e iterativa, com foco na identificação e resolução rápida dos problemas encontrados ao longo do desenvolvimento.

## 3.1 Modelo de Dados – Modelo Físico

### 3.1.1 Criação - createTables.sql

O modelo físico adotado é composto por diversas tabelas que representam as diferentes entidades e relações presentes no sistema. A tabela "Jogador" armazena informações dos jogadores, como email, nome de usuário, estado e região. A tabela "Região" guarda as informações das regiões do jogo. A tabela "Jogo" armazena as informações dos jogos, incluindo o ID do jogo, o nome e a URL do jogo.

A tabela "Compra" regista todas as compras realizadas pelos jogadores, armazenando o ID do jogador, o ID do jogo, o preço e a data da compra. A tabela "Partida" armazena informações sobre cada partida, como o ID do jogo, a data de início, a data de fim, a região da partida, entre outras. As tabelas "Partida\_normal" e "Partida\_multijogador" representam diferentes tipos de partidas e suas características específicas.

A tabela "Pontuação" armazena informações sobre as pontuações de cada jogador em cada partida, incluindo o ID da partida, o ID do jogador e a pontuação. A tabela "Cracha" armazena informações dos crachás do jogo, incluindo o ID do jogo, o nome, a pontuação e a URL da imagem do crachá. A tabela "Crachas\_jogador" relaciona os crachás aos jogadores que os possuem.

A tabela "Amizade" representa as amizades entre os jogadores. A tabela "Conversa" armazena informações sobre cada conversa entre os jogadores, enquanto a tabela "Conversa\_jogador" relaciona os jogadores presentes em cada conversa. A tabela "Mensagem" armazena as mensagens trocadas entre os jogadores de cada conversa.

Por fim, as tabelas "Estatisticas\_jogador" e "Estatisticas\_jogo" armazenam informações estatísticas sobre os jogadores e os jogos, respectivamente. O modelo físico escolhido foi projetado para garantir a integridade dos dados, minimizar a redundância e otimizar o desempenho do sistema. Além disso, ele permite a criação de consultas complexas para análise dos dados.

### 3.1.2 Restrições de Integridade

O modelo físico adotado inclui diversas restrições de integridade para garantir a consistência dos dados e evitar inconsistências ou informações inválidas na base de dados. Algumas das restrições incluem:

* Restrição UNIQUE nas colunas "email" e "username" da tabela "Jogador", garantindo que nenhum jogador possa utilizar o mesmo e-mail ou nome de usuário;
* Restrição CHECK na coluna "estado" da tabela "Jogador", permite apenas valores específicos ("Ativo", "Inativo" ou "Banido");
* Restrição FOREIGN KEY nas tabelas "Compra", "Partida", "Partida\_normal", "Partida\_multijogador", "Pontuação", "Cracha\_jogador", "Amizade", "Conversa\_jogador", "Mensagem", "Estatisticas\_jogador" e "Estatisticas\_jogo", garantindo que as referências a outras tabelas sejam válidas;
* Restrição CHECK na coluna "estado" da tabela "Partida\_multijogador", permitindo apenas valores específicos ("Por iniciar", "A aguardar jogadores", "Em curso" ou "Terminada");
* Restrição CHECK na coluna "grau\_dificuldade" da tabela "Partida\_normal", permite apenas valores de 1 a 3;
* Restrição CHECK na coluna "data\_fim" da tabela "Partida", garante que a data de término da partida não seja anterior à data de início;
* Restrição PRIMARY KEY nas tabelas "Jogador", "Regiao", "Jogo", "Compra", "Partida", "Partida\_normal", "Partida\_multijogador", "Pontuacao", "Cracha", "Crachas\_jogador", "Amizade", "Conversa", "Conversa\_jogador", "Mensagem", "Estatisticas\_jogador" e "Estatisticas\_jogo", garante que cada inserção numa dada tabela tenha um identificador exclusivo;
* Restrição FOREIGN KEY nas tabelas "Compra", "Partida", "Pontuacao", "Crachas\_jogador", "Conversa\_jogador", "Mensagem", "Estatisticas\_jogador" e "Estatisticas\_jogo", garante que as referências a outras tabelas sejam válidas.

Essas restrições ajudam a garantir a integridade dos dados e a consistência do modelo físico adotado.

### 3.1.3 Inserção e Remoção – insertTables.sql e dropTables.sql

A operação de inserção e remoção são fundamentais para a manipulação de dados em um sistema. A sua correta utilização é essencial para garantir a integridade dos dados e a eficiência do sistema.

Para a operação de inserção, é necessário garantir que os dados inseridos respeitem as regras definidas pelo esquema da tabela. Isso inclui o tipo de dado a ser inserido em cada coluna e as restrições de chave primária e chave estrangeira. Além disso, a inserção deve ser realizada de forma consistente e segura, evitando erros de dados e violações de integridade. Para isso, é importante utilizar transações, que garantem a execução completa e correta da operação ou a reversão caso ocorra algum erro.

Já para a operação de remoção, é necessário ter cuidado para evitar a exclusão de dados importantes e garantir que não haja violação de integridade referencial. Isso significa que é preciso remover os dados em uma ordem específica, de forma a respeitar as dependências entre as tabelas. Além disso, é importante ter em mente que a remoção de dados pode ter impacto em outras partes do sistema, como em relatórios e em outros processos que dependem desses dados. Portanto, é preciso avaliar cuidadosamente as consequências antes de realizar uma operação de remoção.

Em resumo, a correta utilização das operações de inserção e remoção em um modelo relacional de dados requer atenção aos detalhes e cuidado na execução. É importante seguir as regras definidas pelo esquema da tabela, utilizar transações para garantir a consistência e segurança dos dados e avaliar cuidadosamente as consequências de uma operação de remoção antes de executá-la.

## 3.2 Operações sobre o Modelo de Dados

### 3.2.2 Alínea (d) – criarJogador

FONTES

### 3.2.3 Alínea (e) – totalPontosJogador

A função tem como objetivo retornar o total de pontos acumulados por um jogador específico, cujo identificador é passado como parâmetro "jogador\_id".

Antes de fazer a consulta na tabela "Pontuacao", a função faz uma verificação de existência do jogador na tabela "Jogador". Se o jogador não existir, a função dispara uma exceção com uma mensagem indicando o id do jogador inexistente.

Caso contrário, a função usa a cláusula "SUM" para calcular a soma total dos pontos na tabela "Pontuacao" associados ao jogador especificado e retorna esse valor como resultado da função.

Dessa forma, a utilização de tabelas e verificações nessa função é fundamental para garantir que o resultado retornado pela função seja válido e consistente com os dados armazenados no banco de dados. Além disso, a função também fornece um mecanismo de proteção contra erros e tentativas de acesso a dados inválidos ou inexistentes.

### 3.2.4 Alínea (f) – totalJogosJogador

A função tem como objetivo retornar o total de jogos disputados por um jogador específico, cujo identificador é passado como parâmetro "jogador\_id".

Antes de fazer a consulta na tabela "Pontuacao" para obter o total de jogos, a função verifica a existência do jogador na tabela "Jogador". Se o jogador não existir, a função dispara uma exceção com uma mensagem indicando o id do jogador inexistente.

Caso contrário, a função usa a cláusula "COUNT DISTINCT" para contar o número de jogos únicos na tabela "Partida" associados ao jogador especificado na tabela "Pontuacao". Isso é feito através de joins entre as tabelas "Pontuacao", "Partida" e "Jogo". Caso nenhum jogo seja encontrado, a função define o total de jogos como zero.

### 3.2.5 Alínea (g) – PontosJogoPorJogador

A função tem como objetivo retornar uma tabela com o total de pontos que cada jogador fez em um jogo específico, cujo identificador é passado como parâmetro "jogo\_referencia".

Antes de fazer a consulta na tabela "Pontuacao" para obter o total de pontos, a função verifica a existência do jogo na tabela "Jogo". Se o jogo não existir, a função dispara uma exceção com uma mensagem indicando o id do jogo inexistente.

Caso contrário, a função usa a cláusula "SUM" para calcular a soma total dos pontos na tabela "Pontuacao" associados a cada jogador em partidas do jogo especificado. Isso é feito através de joins entre as tabelas "Pontuacao" e "Partida", utilizando o identificador do jogo especificado como filtro na cláusula WHERE.

Em seguida, a função agrupa os resultados por jogador utilizando a cláusula "GROUP BY". O resultado é uma tabela com o identificador do jogador e o total de pontos obtidos em cada partida do jogo especificado.

### 3.2.6 Alínea (h) – associarCracha

A função "associarCracha" tem como objetivo associar um crachá a um jogador em um determinado jogo, desde que o jogador atenda aos requisitos de pontuação necessários para obter o crachá.

Antes de realizar a associação, a função verifica se o jogador possui o jogo em questão e se o crachá fornecido é válido para o jogo. Em seguida, a função busca o ID do crachá, a pontuação necessária para obtê-lo e a pontuação atual do jogador no jogo. Se o jogador tiver pontuação suficiente, a função verifica se ele já possui o crachá e, em caso negativo, realiza a associação.

Para garantir a integridade dos dados, a função utiliza transações com nível de isolamento READ COMMITTED e chama outros procedimentos armazenados para a lógica de negócios, tratamento de erros e controle transacional. As tabelas utilizadas incluem "Compra" (para verificar se o jogador possui o jogo), "Jogo" e "Cracha" (para verificar se o crachá é válido para o jogo), "Crachas\_jogador" (para associar o crachá ao jogador) e "Pontuacao" e "Partida" (para obter a pontuação atual do jogador no jogo).

### 3.2.7 Alínea (i) – iniciarConversa

A função "iniciarConversa" é responsável por iniciar uma nova conversa associando um jogador a ela. Para isso, são utilizadas três tabelas: "Jogador", "Conversa" e "Conversa\_jogador".

A tabela "Jogador" contém informações sobre os jogadores, incluindo o seu ID e o estado (Ativo ou Inativo). É verificado se o jogador é válido e ativo antes de continuar com a criação da conversa.

A tabela "Conversa" armazena as informações das conversas, incluindo o seu ID e o nome. É verificado se já existe uma conversa com o mesmo nome antes de criar uma nova.

A tabela "Conversa\_jogador" associa um jogador a uma conversa, usando os IDs correspondentes. É utilizado para registrar a participação dos jogadores nas conversas.

Além disso, são utilizados três procedimentos armazenados para garantir a consistência dos dados e o controle transacional. O procedimento "iniciarConversa\_logic" é responsável pela lógica da criação da conversa e da associação do jogador a ela. O procedimento "iniciarConversa\_handler" trata exceções e controla transações. Finalmente, o procedimento "iniciarConversa" define o nível de isolamento da transação e chama o procedimento "iniciarConversa\_handler".

Em resumo, a função "iniciarConversa" utiliza tabelas para armazenar informações sobre jogadores e conversas e procedimentos armazenados para garantir a consistência dos dados e o controle transacional.

### 3.2.8 Alínea (j) – juntarConversa

FONTES

### 3.2.8 Alínea (k) – enviarMensagem

JOAO

### 3.2.8 Alínea (l) – jogadorTotalInfo

A vista "jogadorTotalInfo" é criada a partir de um conjunto de tabelas do banco de dados, com o objetivo de apresentar informações relevantes dos jogadores que não estão banidos. A vista utiliza a tabela "jogador" como base, juntamente com outras tabelas como "Pontuacao" e "Compra", fazendo uso de funções de agregação como COUNT e SUM para contabilizar o número de partidas, número de jogos e a pontuação de cada jogador.

A cláusula "LEFT JOIN" é utilizada para garantir que todas as linhas da tabela "jogador" sejam mantidas na visualização, mesmo que não haja correspondência na tabela "Pontuacao" ou "Compra". A cláusula "COALESCE" é utilizada para tratar casos em que não há registro de jogos na tabela "Compra".

Por fim, a cláusula "WHERE" é utilizada para filtrar jogadores banidos da vista. Com a criação desta vista, é possível ter uma visão geral da performance dos jogadores em termos de partidas jogadas, número de jogos adquiridos e pontuação acumulada, facilitando a análise e o gerenciamento dos jogadores do sistema.

### 3.2.8 Alínea (m) – ...

# Avaliação Experimental

A avaliação da nossa solução é apresentada neste capítulo. Aqui mostramos

como as nossas grandes ideias funcionaram

Exemplo de indentação do segundo parágrafo.

## 4.1 Nome da primeira secção deste capítulo

Texto da secção.

Continuação do texto noutro parágrafo.

## 4.2 A segunda secção deste capítulo

Na segunda secção deste capítulo, vamos abordar o enquadramento, o contexto e as funcionalidades.

### 4.2.1 A primeira sub-secção desta secção

As sub-secções são úteis para mostrar determinados conteúdos de forma

organizada. Contudo, o seu uso excessivo também não contribui para a facilidade

de leitura do documento[[1]](#footnote-1).

### 4.2.2 A segunda sub-secção desta secção

Esta é a segunda sub-secção desta secção, a qual termina aqui.

## 4.3 Análise de resultados

A análise de resultados segue aqui, nos próximos parágrafos de forma detalhada.

A data limite de entrega da versão final em 19 de Setembro de 2015 tem subjacente a inscrição em época especial. Não se verificando esta situação, a data limite de entrega é em 25 de Julho de 2015. O júri de cada projecto é constituído por indicação do respectivo orientador, até 25 de Maio de 2015. A avaliação da versão beta será realizada até 29 de Junho de 2015.

A discussão da versão final do projecto é pública e inclui até 30 minutos de apresentação e demonstração seguidos de discussão até ao máximo de 120 minutos de duração total da prova (40 e 150 minutos, respectivamente, quando o grupo tem três estudantes, e, 20 e 90 minutos quando o trabalho é individual). O júri da prova é proposto pelo orientador do projecto, tem pelo menos três elementos e inclui o arguente, o orientador e um docente de Projecto e Seminário (podendo este, em caso de impedimento, delegar num docente da área departamental). As deliberações do júri são tomadas por maioria simples.

A melhoria de classificação, se realizada no semestre de Inverno seguinte, terá por base trabalho adicional e discussão oral avaliados pelo mesmo júri. Quando realizada noutro semestre, envolve a realização de novo projecto.

A coordenação global dos projectos e dos seminários é feita pelos docentes de Projecto e Seminário, de acordo com as orientações definidas pela comissão coordenadora do curso. No sítio desta unidade curricular, é mantida a informação relevante, incluindo prazos, calendário dos seminários, estado dos projectos e estudantes e orientadores envolvidos. No final de cada ano lectivo, o regente elabora e apresenta à comissão coordenadora do curso um relatório sucinto sobre o funcionamento da unidade curricular. Em cada ano lectivo são identificados os melhores projectos para promover a sua divulgação pública.

# Conclusões

Neste trabalho tratou-se o problema. Foi formulada a solução que assenta nos princípios de boas práticas aprendidos ao longo do curso.

A solução obtida atingiu resultados satisfatórios.

# Referências

|  |  |
| --- | --- |
| [1] | Wikipedia, “Big data --- Wikipedia, The Free Encyclopedia,” http://en.wikipedia.org/w/index.php?title=Big\_data&oldid=648786139, 2015. |
| [2] | X. Ding, X. Zhu e G. Wu, “Data mining with big data,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, n.º 1, pp. 97-107, 2014. |
| [3] | J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong e J. Zhang, “What Will 5G Be?,” *IEEE Journal on Selected Areas in Communications,* vol. 32, n.º 6, pp. 1065-1082, 2014. |
| [4] | L. Boytsov, “Indexing Methods for Approximate Dictionary Searching: Comparative Analysis,” *J. Exp. Algorithmics,* vol. 16, n.º may, p. 1.81, 2011. |
| [5] | T. Jurkiewicz e K. Mehlhorn, “On a Model of Virtual Address Translation,” *J. Exp. Algorithmics,* vol. 19, n.º jan, pp. 1-18, 2015. |
| [6] | J. Neumann, The Computer and the Brain, New Haven, CT, USA: Yale University Press, 1958. |
| [7] | B. Kernighan e P. Plauger, The Elements of Programming Style, New York, NY, USA: McGraw-Hill, Inc., 1982. |

# A.1 Diagramas da Aplicação

Estamos no início do anexo 1. Nalguns casos, é conveniente colocar anexos de forma a complementar os resultados. Por vezes, em casos excepcionais devido à sua dimensão, as figuras têm que ser apresentadas de forma a ocupar toda a página, na forma de paisagem (*landscape*). Podemos fazer isso da forma que se apresenta na figura 3.



Figura 3 – Diagrama de casos de utilização.

# A.2 Modelos de dados

Estamos no início do anexo 2.

O *relatório* é um resumo do projecto global. Apenas como referência, é expectável cerca de 30 a 40 páginas A4 não devendo exceder 50 páginas.. A estrutura deve ser discutida e aceite pelo orientador. Os capítulos apresentados devem ter, em geral, a seguinte organização:

**Capítulo 1** — Introdução

Enquadramento do trabalho, metas, objectivos e especificações do projecto e resumo da solução. Concluir com a descrição breve dos restantes capítulos.

**Capítulo 2** — Formulação do problema

Introdução dos conhecimentos necessários para entendimento do trabalho, estabelecimento de terminologia e descrição detalhada do problema e do seu contexto. Síntese de abordagens anteriores do problema, caso existam, indicando as razões porque são insatisfatórias.

**Capítulo 3** — Grande ideia 1

Este capítulo pode ser subdividido em secções, designadamente:

1. Introdução: descrição da abordagem do problema e da metodologia utilizada; identificação das tarefas;
2. Elenco das características / Análise do problema: especificações, constrições, ferramenta de análise, etc*.*
3. Projecto: modelo para resolução do problema;
4. Implementação: a implementação do modelo como sistema computacional; descrição concisa do *hardware* e do *software*; dificuldades e contradições encontradas e sua resolução;
5. Avaliação: testes realizados e resultados experimentais (quando possível, o objectivo, a montagem e o método usado devem ser brevemente descritos); análise crítica dos resultados.

…

**Capítulo k+2** — Grande ideia k

**Capítulo k+3** — Conclusões

Recapitulação do trabalho desenvolvido. Referir claramente as observações e conclusões importantes. Discussão de ideias e recomendações para trabalho futuro.

**Referências**

Elenco dos livros e artigos citados no relatório. As referências são numeradas consecutivamente ao longo do relatório. O número da referência deve estar entre parêntesis rectos: [1].

**Anexos**

Os anexos devem incluir as partes importantes do *dossier do projecto*. O seu conteúdo depende da natureza do projecto, mas, em geral, pode incluir: listagem de programas, resultados de testes de *software*, exemplos de ecrãs de interface com o utilizador, esquemas dos circuitos, listagem de componentes, *data sheets* críticos, resultados de testes de *hardware*, desenhos mecânicos, análise económica, etc. (quando realizável, o *relatório* deve ser acompanhado da cópia do código, bibliotecas, etc. em suporte electrónico).

**Mais algumas dicas…**

O júri para avaliação do projecto final de curso apreciará o projecto, a sua demonstração e o respectivo relatório final (valorizando a escrita enquanto forma de divulgação de conhecimento). O relatório, depois de aceite e discutido, ficará disponível na biblioteca do departamento, para consulta.

O relatório deve ser auto-suficiente, isto é, o professor ou qualquer aluno finalista deve ficar apto a perceber o trabalho que realizou sem ter de ir à biblioteca ler os artigos originais.

Não escreva para o orientador, conhecedor de todo o detalhe, ou para um principiante. Tente escrever para uma audiência constituída por estudantes finalistas. Mantenha em mente o nível de conhecimentos do leitor a que se dirige. O relatório será uma base de trabalho para estudantes em circunstâncias semelhantes. Não dificulte o trabalho do leitor nem o faça estúpido (obviamente, …). Também é impossível ser totalmente claro. Evite afirmações dogmáticas (exemplo: "O *software* é a parte mais importante do computador.").

O relatório técnico não é uma história: usualmente não segue a cronologia das actividades realizadas. Também não é um romance (atenção à adjectivação). O relatório é um documento formal, feito para descrever os aspectos importantes do trabalho realizado.

Não tente descrever a função de cada componente, por exemplo a frase "O circuito IC2 e os componentes a ele associados formam um amplificador inversor …" é adequada. Contudo, descreva detalhadamente a função de cada componente ou circuito invulgar ou crítico.

As ilustrações (figuras, tabelas, gráficos e exemplos) são auxiliares preciosos para a explicação, mas envolvem muito trabalho. As figuras e as tabelas devem ser legíveis, instrutivas, legendadas e ter título. Os exemplos devem ser suficientemente detalhados para ilustrar o conceito.

O texto deve, pelo menos, ser analisado por um corrector ortográfico: os erros de ortografia são inadmissíveis.

Recomenda-se a leitura de alguns artigos e ou livros bem escritos para adquirir sensibilidade para a arte de escrever. Os artigos premiados em conferência são, normalmente, bons exemplos de escrita.

A escrita do relatório demora sempre mais tempo do que o inicialmente previsto.

**Resumo**

No essencial, a ideia é que tem algo para vender e o “Resumo” é a montra: a mensagem deve ser suficientemente clara e encorajar o cliente a entrar — se ele não a perceber passará ao lado. O resumo inclui: a motivação para o trabalho, como o fez e os resultados principais. Devem ser evitados chavões e palavras longas, as referências são proibidas e não deve utilizar acrónimos. Tenha em conta que o leitor será influenciado quer pela informação contida no resumo quer pela maneira como este está escrito. Não há desculpas para frases curtas ou desligadas, erros de gramática ou erros de sintaxe.

Não é fácil escrever um bom resumo.

**Introdução**

Procure dar resposta às seguintes questões: qual é o problema? porque é importante? o que é que outros já fizeram? quais as ideias base da solução apresentada? como está organizado o resto do relatório?

**Formulação do problema**

Defina o problema. Introduza a terminologia. Discuta as propriedades básicas.

**Solução do problema**

Enumere as coisas que fez e que considere importantes. Não seja modesto mas também não exagere.

A correcta avaliação do projecto é um aspecto crítico.

**Conclusões**

Procure dar resposta às seguintes questões: quais, se for caso disso, as lições aprendidas? o que, se algo, foi explicado? em que medida os objectivos foram atingidos? se existe algo que agora faria de forma diferente? quais as vantagens e desvantagens do trabalho realizado face a exemplos da literatura? que ideias para trabalho futuro?

**Referências**

A ideia subjacente à referência é que esta poupa papel e que o leitor poderá obter o documento em qualquer biblioteca científica razoável. Assim, é critério essencial referir revistas disponíveis em bibliotecas de instituições de ensino superior ou outras instituições profissionais. Em geral, não é razoável a referência a actas de conferências, dado que estas raramente estão acessíveis em bibliotecas pelo que, para todos os efeitos, estão indisponíveis. As referências a “Comunicações Privadas” são inaceitáveis. A informação dada deve ser sempre suficientemente detalhada por forma a que o leitor possa adquirir a publicação ou consultá-la numa biblioteca. Referências a dissertações de doutoramento ou outras devem indicar a instituição e o seu endereço. Sendo a referência essencial para o trabalho, no caso desta ser difícil de obter, dever-se-á incluir no documento, ou em apêndice, os seus pontos essenciais.

Cite uma referência sempre que está a incluir algo que não é contribuição sua ou quer indicar um conjunto de referências que o leitor pode consultar, mas cujo conteúdo não pode ser descrito adequadamente no relatório.

1. Este é um exemplo de nota de rodapé. Devem ser usadas com moderação, pois retiram a atenção ao texto principal. [↑](#footnote-ref-1)