

**Relatório do Projeto de Psi**

Ander Bogalho, David Gonçalves

Curso Técnico de Gestão e Programação de Sistemas Informáticos

Escola Profissional Bento de Jesus Caraça, Delegação do Barreiro

P.S.I.: Programação e Sistemas de Informação

Coordenador Marcelo Simão

10 de janeiro de 2024

**Projeto Final de PSI**

O projeto Final de PSI foi realizado a partir de todas as aulas onde foram feitas operações *CRUD*. O tema foi escolhido a partir do gosto pessoal dos desenvolvedores do projeto.

O projeto facilita a jogabilidade de um jogo (League of Legends) a partir das escolhas dos jogadores, de forma a unir indiretamente uma comunidade pouco amigável para novos jogadores. É muito habitual neste tipo de jogo os novos jogadores estarem completamente perdidos, devido a grande escolha de personagens que consegue agradar com vários tipos de jogabilidade completamente diferente, desde dos mais fácies de aprender até os que requerem uma longa curva de aprendizagem, este projeto pretende diminuir essa curva de aprendizagem na parte teórica, ou seja, facilitando as escolhas antes do jogo, como as posições, os itens que podem ser utilizados durante o jogo, os personagens e também com a opção de inserir os registos da última partida jogada onde vai-se calcular as estatísticas do jogador conforme ao seu desempenho. Os desenvolvedores desse projeto pretendem fazer um projeto otimizado.

Com isso os desenvolvedores do projeto querem que o mesmo seja o mais fiel ao jogo a partir dos jogadores.

**Introdução**

Foi solicitado o desenvolvimento de uma aplicação em (Docs Python, s.d.)com ligação a uma base de dados *SQLite* no decorrer de um projeto com a duração de duas semanas. Para iniciar o processo, foram analisados os requisitos necessários, identificando-se as funcionalidades essenciais e os dados a armazenar na base de dados. Com base nessa análise, foram definidos os principais modelos de dados, garantindo que a estrutura da base de dados estivesse alinhada com os objetivos do projeto.

Após a definição dos requisitos, procedeu-se à criação e configuração da base de dados *SQLite*. As tabelas foram planeadas e os seus esquemas estruturados de forma a assegurar a correta organização dos dados e das relações entre diferentes entidades. Em seguida, foi implementada a camada de acesso à base de dados, permitindo a execução das operações fundamentais, como inserção, atualização, remoção e atualização de dados. Para garantir a integridade da informação, foram aplicadas restrições e índices adequados.

Durante a segunda semana do projeto, foi dada especial atenção à integração da base de dados com a interface da aplicação. Foram desenvolvidos mecanismos que facilitassem a interação eficiente entre o utilizador e os dados armazenados, assegurando que todas as operações fossem realizadas de forma segura e otimizada. A validação de entradas foi implementada para prevenir a introdução de dados inválidos ou inconsistentes. Além disso, foram aplicadas otimizações nas consultas *SQL*, melhorando o desempenho e reduzindo o tempo de resposta da aplicação.

No decorrer do projeto, foram conduzidos testes rigorosos para validar o correto funcionamento da aplicação. Diferentes cenários de utilização foram simulados para identificar possíveis erros ou inconsistências na manipulação dos dados. Sempre que problemas foram detetados, ajustes foram realizados na implementação, assegurando que os requisitos previamente estabelecidos fossem cumpridos. Adicionalmente, foi elaborada a documentação do código e da estrutura da base de dados, permitindo que futuras manutenções e melhorias fossem efetuadas de forma eficiente.

Por fim, a aplicação foi submetida a uma revisão final, na qual foram analisados todos os aspetos relacionados com a usabilidade, desempenho e segurança. Após as últimas correções e ajustes, a versão final foi disponibilizada, garantindo que todas as funcionalidades planeadas estavam corretamente implementadas e que a aplicação poderia ser utilizada conforme o esperado.

**Finalidades do projeto**

O projeto foi criado com o intuito de ajudar os jogadores de League of Legends, de forma a orienta-los com builds, personagens e roles.

***Escolha do Projeto***

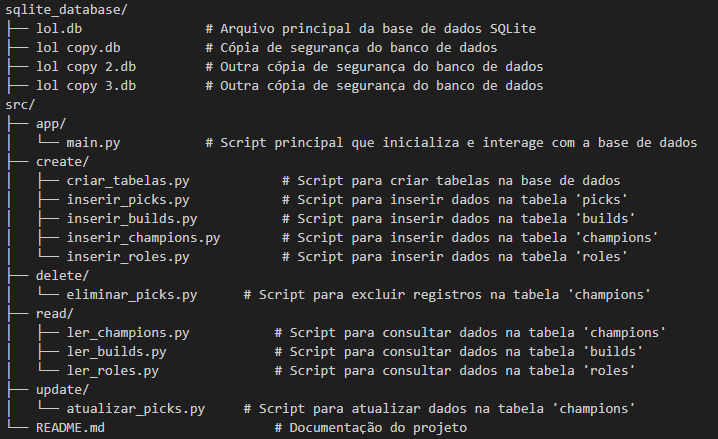
O projeto foi escolhido pelo facto do jogo (League of Legends Wiki, s.d.) ser um jogo de equipa onde a equipa necessita de estar informada dos personagens, builds e roles, como uma forma de orientar a comunidade e ajudar no jogo. Indicando ao jogador como os jogadores estão a jogar com os champions e builds, os mais recentes e menos escolhidos pelos jogadores tudo isso organizado em uma base de dados com várias tabelas.

***Estrutura do Projeto***

A estrutura do projeto foi feita a partir de pastas que usam a logica do CRUD (*Create*, *Read*, *Update* e *Delete*) e o código separado por ficheiros, como pode encontrar baixo:

**Figure 1**

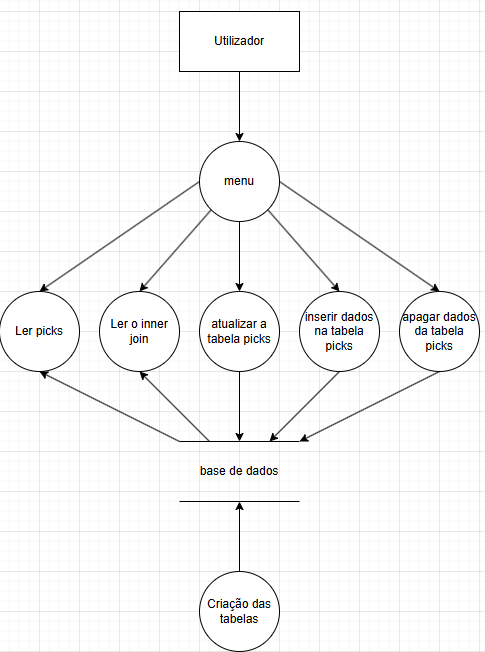
Estrutura do projeto.

**

*Note. A base de dados possui 3 backups, o código está no src(source), esta dividido por pastas utilizando a logica CRUD todo oque seja associado a create esta na pata create, o delete esta na pasta delete, o read esta na pasta read e o update na pasta update, a pasta app possui o main.py que é o script principal que inicia tudo.*

***Diagrama de Fluxo de Dados (DFD)***

**Figure 2**

Diagrama de fluxo de dados do projeto.

*Note. O utilizador interage com o menu, o menu possui 5 opções, ler os picks, ler o inner join, inserir picks, atualizar picks e apagar picks. Todo isso interage com a base de dados. A criação das tabelas cria 4 tabelas, 3 delas predefinidas com os Champions, roles e builds, e a 4ª tabela é a tabela picks que faz uma ligação entra as tabelas como pode-se ver na figura 3.*

***Funcionalidades***

Estas são as funcionalidades disponíveis no menu do projeto:

**Inserir *picks*:** Esta funcionalidade é responsável por fazer a inserção dos registos na tabela *picks,* sendo os registos, o nome do *pick,* o id do *champion*, o id do *role* e o id da *build* que são selecionados a partir dos ids das suas respetivas tabelas mãe (*champions*, *roles*, *builds*).

**Atualizar *picks*:** Esta funcionalidade é responsável por atualizar os registos da tabela *picks*, dando a opção ao utilizador de escolher o *pick* que quere alterar e selecionando o registo específico para alterar o valor.

**Ler *picks*:** Esta funcionalidade é responsável por apresentar todos os registos disponíveis na tabela *picks*.

**Eliminar *picks*:** Esta funcionalidade é responsável por eliminar os registos da tabela *picks*, dando a opção ao utilizador para selecionar o id do *pick* que pretende eliminar.

**Inserir jogo:** Esta funcionalidade é responsável por inserir o registo do jogo ao *pick* que o utilizador escolha, de modo a inserir as estatísticas dos abates, mortes, e assistências do último jogo para o *pick* e assim poder calcular-se nas outras funcionalidades a *winrate* e o *KDA* (*kills*, *deaths*, *assists*).

***Winrate*:** Esta funcionalidade é responsável por calcular a *winrate* (métrica que representa a proporção de vitórias em relação ao total de tentativas ou jogos realizados) do registo de jogos do respetivo pick, selecionado pelo utilizador.

***KDA*:** Esta funcionalidade é responsável por calcular o KDA (Abates+Assistências)/(Mortes+1) do pick selecionado pelo utilizador.

**Conclusão**

Em resumo, este projeto foi útil para a aprendizagem e a aplicação dos conceitos sobre bases de dados, utilizando a biblioteca sqlite3 do *python* que permitiu utilizar os *queries* *SQL* e criar uma base de dados de maneira local. Com este projeto, os alunos poderam aplicar os conteúdos lecionados de uma forma mais prática, com a escolha do tema livre para este projeto.

# **Referências**

*Docs Python*. (s.d.). Obtido de https://docs.python.org/3/library/sqlite3.html

*League of Legends Wiki*. (s.d.). Obtido de https://wiki.leagueoflegends.com/en-us/