

**Trabalho Final Módulo 15: “*Valorant*”**

Brena A. P. Picado, Salvador S. Raposo

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

05 de fevereiro de 2024

**Introdução**

Neste relatório, será explicado o código desenvolvido nas aulas de Programação de Sistemas Informáticos (PSI). O tema escolhido para este trabalho é a seleção de agentes do jogo ‘*Valorant’*. O código é composto por vários scripts em *Python* que interagem com uma base de dados *SQLite*. São realizadas tarefas como a criação de tabelas, inserção de dados, eliminação de registos e tabelas e atualização de dados. A base de dados tem como objetivo gerir uma equipa do jogo e os personagens selecionados.

**Estrutura do Projeto**

/src

│

├── create

│ └── create\_tables.py # Criação das tabelas na base de dados

│

├── insert

│ └── insert\_agents.py # Inserção de agentes na base de dados

│

├── select\_unselect

│ ├── select\_agent.py # Seleção de agentes

│ └── unselect\_agent.py # Desmarcação de agentes

│

├── read

│ └── show\_agents.py # Mostrar agentes existentes na base de dados

│

└── main.py # Arquivo principal que executa a aplicação

A base de dados contém duas tabelas principais:

* **Agentes**: Contém as personagens disponíveis para serem selecionadas.
* **Equipa**: Armazena as personagens da tabela **agentes** que fazem parte da equipa

**Create (criar as tabelas)**

Começou-se o projeto pela criação dessas mesmas tabelas:

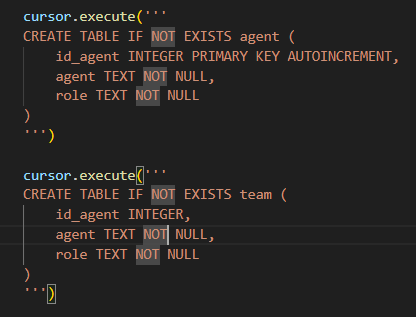
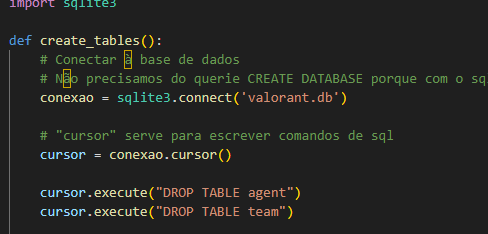
******

Figura 1. Criação das tabelas

A função *create\_tables()* tem como objetivo criar as tabelas necessárias na base de dados *SQLite*, caso ainda não existam. A função estabelece uma conexão com a base de dados chamada *valorant.db* utilizando a função *sqlite3.connect().* Se o ficheiro da base de dados não existir, ele será criado automaticamente. Em seguida, é criado um cursor com o comando *conexao.cursor()*, que permite a execução de comandos SQL na base de dados.

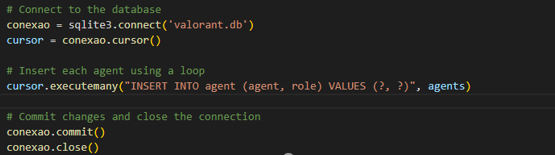
Antes de criar as novas tabelas, o código elimina as tabelas *agent* e *team*, caso estas já existam, com os comandos *cursor.execute("DROP TABLE agent")* e *cursor.execute("DROP TABLE team")*. Isto garante que a base de dados esteja limpa antes de criar as novas tabelas.  


Depois, as tabelas são criadas com os comandos SQL CREATE TABLE IF NOT EXISTS. A tabela *agent* é criada com três colunas: *id\_agent*, *agent* e *role*, o *id\_agent* sendo a chave primária e a coluna *agent* armazena o nome do agente, enquanto role guarda o papel do agente. A tabela *team* é criada com a mesma estrutura, mas com uma coluna extra *id\_agent*, que funciona como uma chave estrangeira.

Por fim, as alterações são confirmadas com *conexao.commit()* e a conexão é fechada com *conexao.close().*

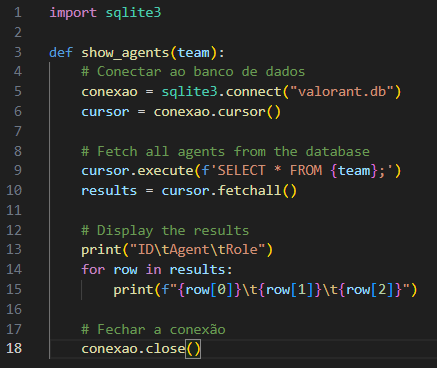
**Insert (inserir dados nas tabelas)**

O código cria a lista *agents*, que contém tuplos, onde cada tuplo representa o nome de um agente e o respetivo papel (role) que desempenha no jogo, como "Controller", "Sentinel", "Duelist", etc.  
  
A seguir, o código estabelece uma conexão com a base de dados utilizando a função *sqlite3.connect().* O cursor é criado para executar operações SQL dentro da base de dados.

Depois, o código utiliza o método *executemany()* para inserir os dados de todos os agentes na tabela *agent*. Este método executa a instrução SQL INSERT INTO agent (agent, role) VALUES (?, ?) para cada tuplo na lista *agents*, onde o ? é um marcador de posição que será substituído pelos valores correspondentes de cada tuplo (nome do agente e papel).  


Após a inserção dos dados, o código faz o "commit" das alterações e a conexão é encerrada com *conexao.close().*

**Read (mostrar agentes no terminal)**

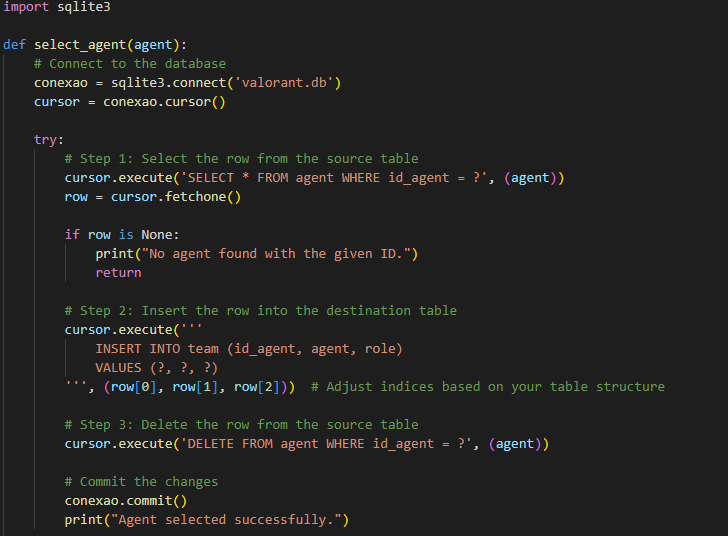
O código seguinte faz uma consulta à tabela *agent* da base de dados *valorant.db*, recuperando todos os dados dos agentes armazenados.

A consulta é realizada com o comando SQL SELECT \* FROM agent, e os resultados são armazenados na variável *results*.

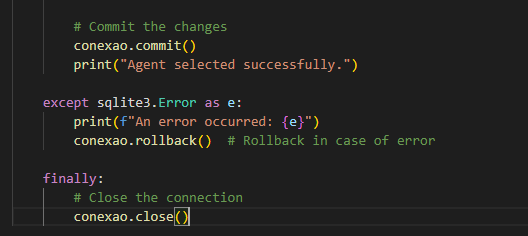
Depois, os dados são apresentados no terminal, onde são exibidos o "ID", o nome do agente ("Agent") e o seu papel ("Role"). Para cada linha de dados obtida, o código imprime o respetivo "ID", nome do agente e papel.

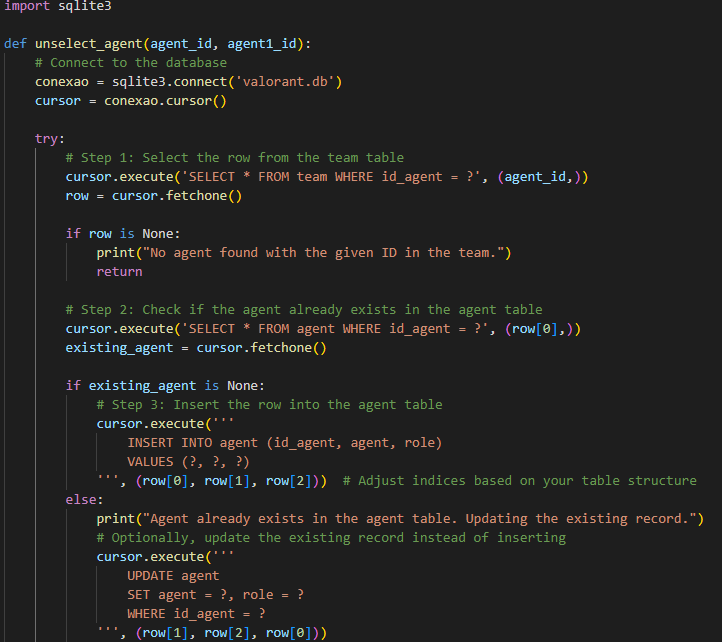
O código não faz nenhuma modificação nos dados da base, apenas os recupera e ilustra os na consola.

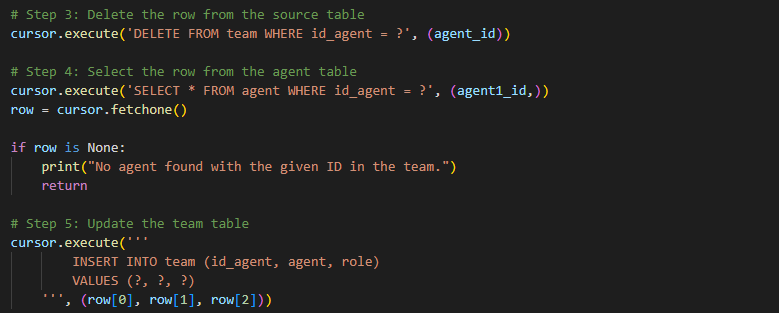
**Selecionar e trocar o agente (mostrar agentes no terminal e atualizar a escolha do utilizador)**

O código define a função *select\_agent*, que conecta à base de dados e executa uma série de operações. Primeiro, ele procura um agente na tabela *agent* com base no *id\_agent* fornecido. Se o agente não for encontrado, a função termina e exibe uma mensagem de erro. Caso o agente exista, o código insere os dados desse agente na tabela *team*, utilizando um comando SQL INSERT. Após isso, o agente é removido da tabela original *agent* com um comando SQL DELETE.  
  


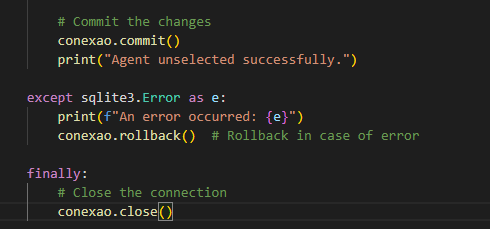
Depois de realizar as operações de inserção e exclusão, o código faz um *commit()* para garantir que as alterações sejam salvas na base de dados. Se ocorrer algum erro durante o processo, um *rollback()* é executado para reverter as alterações. Finalmente, a conexão com a base de dados é fechada com *conexao.close().*



Para atualizar a escolha do agente, o código define a função *unselect\_agent*, que é responsável por remover um agente de uma equipa e adicionar outro no seu lugar. A função começa por procurar o agente com o *agent\_id* na tabela *team*. Se o agente não for encontrado, uma mensagem de erro é exibida e a função termina. Caso o agente exista, o código verifica se o agente já está presente na tabela *agent*. Se não estiver, o agente é inserido na tabela *agent* com um comando SQL INSERT. Se o agente já existir na tabela, o código opta por atualizar o seu registo com um comando SQL UPDATE, modificando o nome e a função do agente na tabela *agent*.  


Após garantir que o agente foi inserido ou atualizado na tabela *agent*, o agente é removido da tabela *team* com um comando SQL DELETE. A seguir, o código procura um outro agente na tabela *agent* com o *agent1\_id* fornecido. Se o novo agente for encontrado, ele é inserido na tabela *team* com um comando SQL INSERT, transferindo-o para a equipa.  


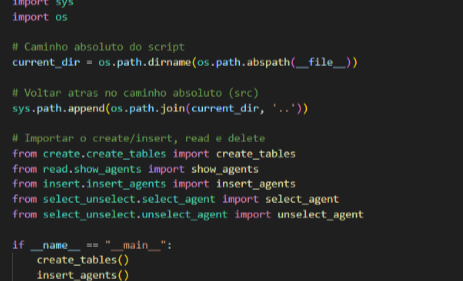
Finalmente, as alterações são confirmadas na base de dados com um *commit().* Se ocorrer algum erro durante o processo, é executado um *rollback()*, revertendo qualquer alteração feita na base de dados. Por fim, a conexão com a base de dados é fechada para liberar os recursos utilizados.



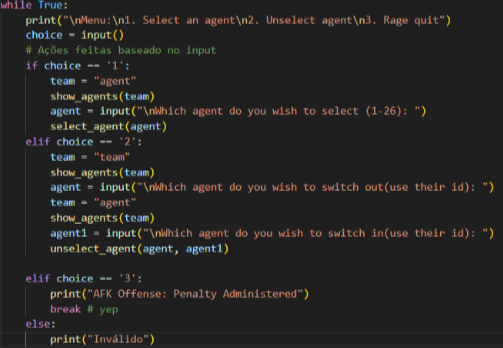
**Main (Menu do código)**

O código importa funções de diferentes módulos para realizar operações específicas. A função **create\_tables** cria as tabelas no banco de dados, **show\_agents** exibe os agentes disponíveis, **insert\_agents** insere agentes no banco de dados, **select\_agent** seleciona um agente, e **unselect\_agent** desmarca ou substitui um agente. Essas funções são organizadas em módulos para manter o código modular e reutilizável.

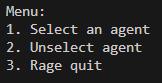
Depois de configurar o ambiente, o código importa as várias funções mencionas anteriormente de diferentes módulos: **create\_tables, show\_agents, insert\_agents, select\_agent, e unselect\_agent**. Estas funções serão usadas mais tarde para interagir com os dados no programa.

No bloco principal de execução (verificado por if \_\_name\_\_ == "\_\_main\_\_":), o código começa por executar as funções **create\_tables()** e **insert\_agents()**, que são responsáveis por criar as tabelas no banco de dados e inserir os agentes no sistema.  


Em seguida, o código entra num *loop*, onde apresenta um menu com três opções para o utilizador:



Se o utilizador introduzir uma opção inválida, o código irá avisar que a opção escolhida é inválida e irá voltar a mostrar o menu.



**Conclusão**

Em conclusão, o projeto foi estruturado de forma modular, com diferentes funções e módulos responsáveis por tarefas específicas, como a criação de tabelas, inserção de dados, visualização dos agentes, e seleção e desmarcação de agentes. A interação com a base de dados é feita através de comandos SQL, permitindo ao utilizador gerir os agentes e as equipas de forma eficiente. A utilização de funções **(create\_tables, insert\_agents, select\_agent e unselect\_agent)** garante que as operações sejam realizadas corretamente.

**Referências**

*APA Style Headings*. <https://apastyle.apa.org/style-grammar-guidelines/paper-format/>