Faculdade de Engenharia da Universidade do Porto

Base de dados de um casino

2.ª submissão do projeto

Duarte Souto Assunção up202208319@up.pt

Guilherme Duarte Matos up202208755@up.pt

João Vítor Ferreira up202208393@up.pt

Índice

1. Modelo conceptual aperfeiçoado **1**

2. Esquema relacional **2**
2.1. Solução inicial 2
2.2. Sugestões da IA 2
2.3. Solução final 2

3. Dependências funcionais e formas normais **3**
3.1. Solução inicial 3
3.2. Melhorias com base em IA 4
3.3. Solução final 5

4. Restrições **6**

5. Base de dados SQLite **8**

6. Introdução de dados Feito, mas por rever **9**

7. O uso de IA no projeto Feito, mas por rever **10**

8. Contribuição de cada membro do grupo Feito, mas por rever **11**

1. Modelo conceptual aperfeiçoado

Após a primeira submissão do projeto, as seguintes sugestões foram implementadas:

- As características de cada generalização estão a preto, para contrastar com as restrições;
- Mais uma restrição foi adicionada à classe “Ganho”.

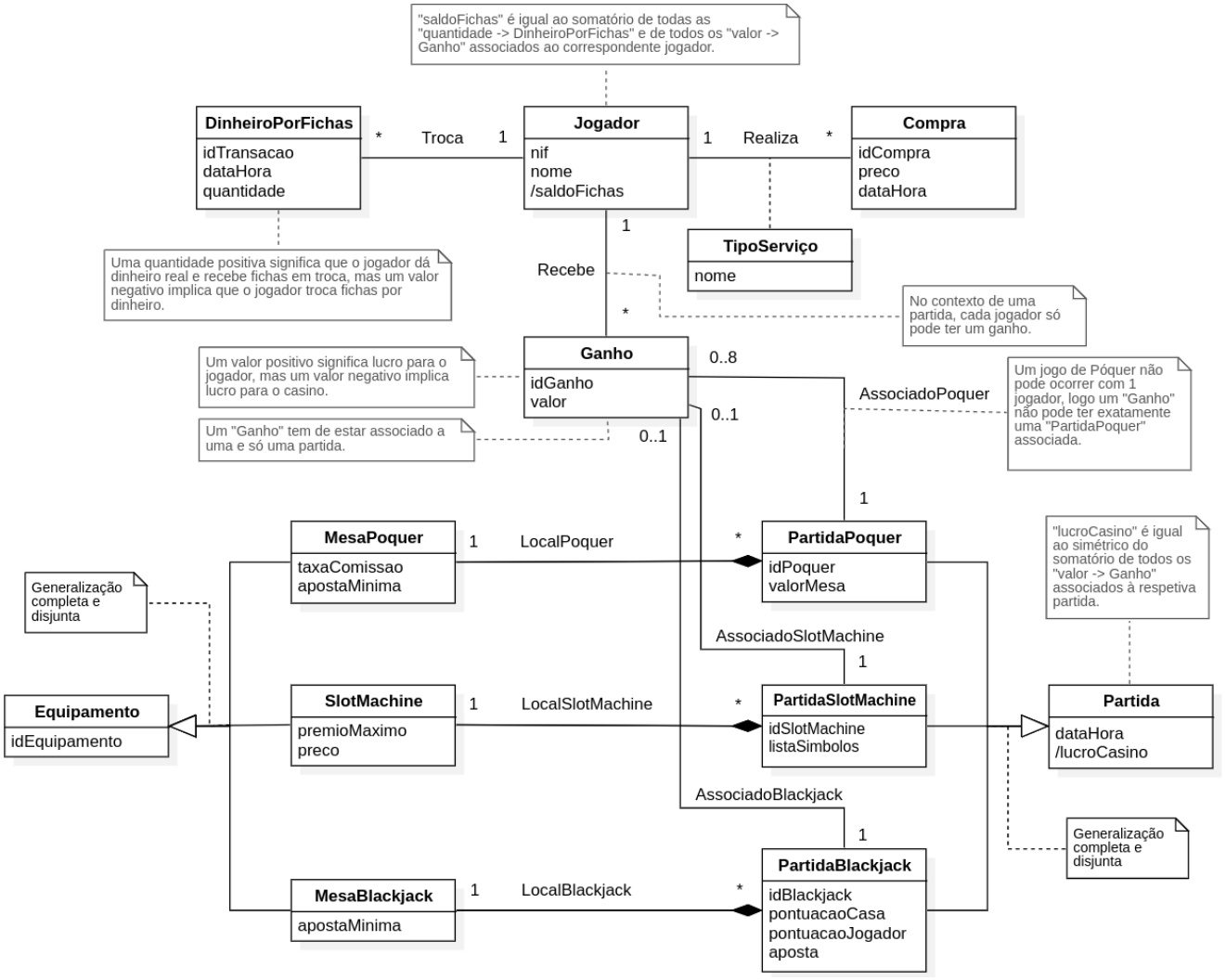

Figura 1. Diagrama UML do modelo conceptual

2. Esquema relacional

2.1. Solução inicial

- Jogador (nif, nome, saldoFichas)
- DinheiroPorFichas (idTransacao, dataHora, quantidade, nif→Jogador)
- Compra (idCompra, preco, dataHora)
- Realiza (nome, idCompra→Compra, nif→Jogador)
- Ganho (idGanho, valor, nif→Jogador, idPoquer→PartidaPoquer, idSlotMachine→PartidaSlotMachine, idBlackjack→PartidaBlackjack)
- PartidaPoquer (idPoquer, dataHora, lucroCasino, valorMesa, idEquipamento→MesaPoquer)
- PartidaSlotMachine (idSlotMachine, dataHora, lucroCasino, listaSimbolos, idEquipamento→SlotMachine)
- PartidaBlackjack (idBlackjack, dataHora, lucroCasino, pontuacaoCasa, pontuacaoJogador, aposta, idEquipamento→MesaBlackjack)
- MesaPoquer (idEquipamento, taxaComissao, apostaMinima)
- SlotMachine (idEquipamento, premioMaximo, preco)
- MesaBlackjack (idEquipamento, apostaMinima)

2.2. Sugestões da IA

Com base no diagrama UML, a inteligência artificial sugeriu o seguinte esquema:

- Jogador(nif, nome, saldoFichas)
- Compra(compraID, nif→Jogador, preco, dataHora, tipoServico)
- DinheiroPorFichas(dinheiroPorFichasID, nif→Jogador, dataHora, quantidade)
- Ganho(ganhoID, nif→Jogador, valor, partidaPoquerID→PartidaPoquer, partidaSlotMachineID→PartidaSlotMachine, partidaBlackjackID→PartidaBlackjack)
- Partida(partidaID, idPartida, dataHora, lucroCasino)
- PartidaPoquer(partidaPoquerID, valorMesa, partidaID→Partida)
- PartidaSlotMachine(partidaSoltMachineID, listaSimbolos, partidaID→Partida)
- PartidaBlackjack(partidaBlackJackID, pontuacaoCasa, pontuacaoJogador, aposta, partidaID→Partida)
- Equipamento(equipamentoID, idEquipamento)
- MesaPoquer(mesaPoquerID, taxaComissao, apostaMinima, equipamentoID→Equipamento)
- SlotMachine(slotMachineID, premioMaximo, preco, equipamentoID→Equipamento)
- MesaBlackjack(mesaBlackjackID, apostaMinima, equipamentoID→Equipamento)

É de notar que muito contexto foi perdido ao longo da conversa, como a hierarquia de equipamentos e partidas, por exemplo.

2.3. Solução final

Nenhuma alteração foi feita com base nas sugestões da inteligência artificial, pelo que a solução inicial mantém-se.

3. Dependências funcionais e formas normais

3.1. Solução inicial

Uma relação *R* está na 3ª Forma Normal se e só se *{A1, A2, …, An}* for uma superchave de *R* ou os membros *B1, B2, …, Bm* que não estão dentro de *{A1, A2, …, An}* forem cada um membros de alguma chave. Isto aplica-se a todas as dependências funcionais não triviais *A1 A2 … An -> B1 B2 … Bm* em *R*.

Uma relação *R* está na Forma Normal de Boyce-Codd se e só se *{A1, A2, …, An}* for uma superchave de *R* para todas as dependências funcionais não triviais *A1 A2 … An -> B1 B2 … Bm* em *R*.

Esta tabela indica para cada tabela as suas dependências funcionais. Isto é importante para analisar se as tabelas violam ou não as formas normais.

Tabela	Dependências Funcionais
Jogador	nif -> nome, saldoFichas
DinheiroPorFichas	idTransacao -> dataHora, quantidade, nif
Compra	idCompra -> preco, dataHora
Realiza	idCompra, nif -> nome
Ganho	idGanho -> nif, valor, idPoquer, idSlotMachine, idBlackjack nif, idPoquer, idSlotMachine, idBlackjack -> idGanho, valor idSlotMachine -> idGanho, nif, valor, idPoquer, idBlackjack idBlackjack -> idGanho, nif, valor, idPoquer, idSlotMachine
PartidaPoquer	idPoquer -> dataHora, lucroCasino, valorMesa, idEquipamento idEquipamento, dataHora -> idPoquer, lucroCasino, valorMesa
PartidaSlotMachine	idSlotMachine -> dataHora, lucroCasino, listaSimbolos, idEquipamento idEquipamento, dataHora -> idSlotMachine, lucroCasino, listaSimbolos
PartidaBlackjack	idBlackjack -> dataHora, lucroCasino, pontuacaoCasa, pontuacaoJogador, aposta, idEquipamento idEquipamento, dataHora -> idBlackjack, lucroCasino, portuacaoCasa, pontuacaoJogador, aposta
MesaPoquer	idEquipamento -> taxaComissao, apostaMinima
SlotMachine	idEquipamento -> premioMaximo, preco
MesaBlackjack	idEquipamento -> apostaMinima

Tendo as definições em mente, conclui-se que todas as dependências funcionais estão na 3ª Forma Normal e na Forma Normal de Boyce-Codd, pelo que não é necessário decompor mais o esquema relacional.

3.2. Melhorias com base em IA

Com base nas relações feitas, a inteligência artificial sugeriu as seguintes dependências funcionais:

Tabela	Dependências Funcionais
Jogador	nif -> nome, saldoFichas
DinheiroPorFichas	idTransacao -> dataHora, quantidade, nif nif -> idTransacao, dataHora, quantidade
Compra	idCompra -> preco, dataHora dataHora -> idCompra, preco
Realiza	nome, idCompra -> nif nif, idCompra -> nome nif, nome -> idCompra
Ganho	idGanho -> valor, nif, idPartida nif -> idGanho, valor, idPartida idPartida -> idGanho, valor, nif idPartida➞PartidaPoquer -> idGanho, valor, nif idPartida➞PartidaSlotMachine -> idGanho, valor, nif idPartida➞PartidaBlackjack -> idGanho, valor, nif
PartidaPoquer	idPartida -> dataHora, lucroCasino, valorMesa, idEquipamento idEquipamento➞MesaPoquer -> idPartida, dataHora, lucroCasino, valorMesa
PartidaSlotMachine	idPartida -> dataHora, lucroCasino, listaSimbolos, idEquipamento idEquipamento➞SlotMachine -> idPartida, dataHora, lucroCasino, listaSimbolos
PartidaBlackjack	idPartida -> dataHora, lucroCasino, pontuacaoCasa, pontuacaoJogador, aposta, idEquipamento idEquipamento➞MesaBlackjack -> idPartida, dataHora, lucroCasino, pontuacaoCasa, pontuacaoJogador, aposta
MesaPoquer	idEquipamento -> taxaComissao, apostaMinima
SlotMachine	idEquipamento -> premioMaximo, preco
MesaBlackjack	idEquipamento -> apostaMinima

3.3. Solução final

Grande parte das dependências funcionais geradas pela inteligência artificial não faziam sentido.

Contudo, a IA apontou que a dataHora iria determinar o idCompra e o preco (dataHora -> idCompra, preco) na tabela Compra, o que está errado, contudo fez-nos aperceber de que a dataHora e o nif determinariam o idTransacao e a quantidade (dataHora, nif -> idTransacao, quantidade) na tabela DinheiroPorFichas.

Tabela	Dependências Funcionais
Jogador	nif -> nome, saldoFichas
DinheiroPorFichas	idTransacao -> dataHora, quantidade, nif dataHora, nif -> idTransacao, quantidade
Compra	idCompra -> preco, dataHora
Realiza	idCompra, nif -> nome
Ganho	idGanho -> nif, valor, idPoquer, idSlotMachine, idBlackjack nif, idPoquer, idSlotMachine, idBlackjack -> idGanho, valor idSlotMachine -> idGanho, nif, valor, idPoquer, idBlackjack idBlackjack -> idGanho, nif, valor, idPoquer, idSlotMachine
PartidaPoquer	idPoquer -> dataHora, lucroCasino, valorMesa, idEquipamento idEquipamento, dataHora -> idPoquer, lucroCasino, valorMesa
PartidaSlotMachine	idSlotMachine -> dataHora, lucroCasino, listaSimbolos, idEquipamento idEquipamento, dataHora -> idSlotMachine, lucroCasino, listaSimbolos
PartidaBlackjack	idBlackjack -> dataHora, lucroCasino, pontuacaoCasa, pontuacaoJogador, aposta, idEquipamento idEquipamento, dataHora -> idBlackjack, lucroCasino, portuacaoCasa, pontuacaoJogador, aposta
MesaPoquer	idEquipamento -> taxaComissao, apostaMinima
SlotMachine	idEquipamento -> premioMaximo, preco
MesaBlackjack	idEquipamento -> apostaMinima

4. Restrições

A seguinte tabela mostra as restrições indicadas nos comentários do diagrama UML, porém numa linguagem mais formal e mais próxima do SQL:

Jogador	● PRIMARY KEY (nif) ● saldoFichas DEFAULT (0)
Dinheiro Por Fichas	● PRIMARY KEY (idTransacao) ● UNIQUE (dataHora, nif) ● FOREIGN KEY (nif) REFERENCES Jogador (nif)
Compra	● PRIMARY KEY (idCompra) ● CONSTRAINT precoPositivo CHECK (preco >= 0)
Realiza	● PRIMARY KEY (idCompra, nif) ● FOREIGN KEY (idCompra) REFERENCES Compra (idCompra) ● FOREIGN KEY (nif) REFERENCES Jogador (nif)
Ganho	● PRIMARY KEY (idGanho) ● UNIQUE idSlotMachine *(Porque uma partida de SlotMachine só está associada a 1 jogador, logo, só 1 ganho.)* ● UNIQUE idBlackJack *(Porque uma partida de BlackJack só está associada a 1 jogador, logo, só 1 ganho.)* ● UNIQUE (nif, idPoquer, idSlotMachine, idBlackjack) *(No contexto de uma partida, cada jogador só pode ter um "Ganho".)* ● FOREIGN KEY (nif) REFERENCES Jogador (nif) ● FOREIGN KEY (idPoquer) REFERENCES PartidaPoquer (idPoquer) ● FOREIGN KEY (idSlotMachine) REFERENCES PartidaSlotMachine (idSlotMachine) ● FOREIGN KEY (idBlackjack) REFERENCES PartidaBlackjack (idBlackjack) ● CONSTRAINT partidaUnica CHECK ((idPoquer != NULL and idSlotMachine == NULL and idBlackjack == NULL) or (idPoquer == NULL and idSlotMachine != NULL and idBlackjack == NULL) or (idPoquer == NULL and idSlotMachine == NULL and idBlackjack != NULL) *(Um "Ganho" tem de estar associado a uma e só uma partida, logo, 2 chaves estrangeiras idPartida terão de ser NULAS 1 não NULA)*
Partida Poquer	● PRIMARY KEY (idPoquer) ● lucroCasino DEFAULT (0) ● UNIQUE (idEquipamento, dataHora) ● FOREIGN KEY (idEquipamento) REFERENCES MesaPoquer (idEquipamento)
Partida Slot Machine	● PRIMARY KEY (idSlotMachine) ● lucroCasino DEFAULT (0) ● UNIQUE (idEquipamento, dataHora) ● FOREIGN KEY (idEquipamento) REFERENCES MesaPoquer (idEquipamento)
Partida Blackjack	● PRIMARY KEY (idBlackjack) ● lucroCasino DEFAULT (0) ● UNIQUE (idEquipamento, dataHora) ● FOREIGN KEY (idEquipamento) REFERENCES MesaPoquer (idEquipamento)
Mesa Poquer	● PRIMARY KEY (idEquipamento)
Slot Machine	● PRIMARY KEY (idEquipamento)

Mesa Blackjack	● PRIMARY KEY (idEquipamento)

Para cada chave estrangeira, as restrições `ON DELETE RESTRICT` e `ON UPDATE CASCADE` são aplicadas.

Os seguintes elementos não estão incluídos nesta tabela e não serão implementados na base de dados porque requerem a implementação de gatilhos:

- Todos os elementos derivados, incluindo os atributos "`saldoFichas`" e "`lucroCasino`";

- A restrição em que um "Ganho" não pode ter exatamente uma "`PartidaPoquer`" associada;

- A limitação em que um "Ganho" não pode ter mais do que 8 "`PartidaPoquer`" associadas.

5. Base de dados *SQLite*

As soluções inicial e final estão disponíveis nos ficheiros enviados em conjunto com este relatório: *create1.sql* e *create2.sql*, respetivamente.

A inteligência artificial sugeriu muitas alterações, em que apenas as seguintes faziam sentido serem implementadas nesta base de dados:

- Incluir valores por omissão nas chaves estrangeiras da tabela *"Ganho"*;

```
Unset
CREATE TABLE Ganho (
  idGanho INT NOT NULL,
  valor INT NOT NULL,
  nif INT NOT NULL,
  idPoquer INT DEFAULT NULL,
  idSlotMachine INT DEFAULT NULL,
  idBlackjack INT DEFAULT NULL,
...
```

- Restringir todos os outros atributos a "NOT NULL", quando aplicável. Por exemplo:

```
Unset
CREATE TABLE Jogador (
  nif INT NOT NULL,
  nome TEXT NOT NULL,
  saldoFichas INT DEFAULT (0),
  PRIMARY KEY (nif)
);
```

6. Introdução de dados

As soluções inicial e final estão disponíveis nos ficheiros enviados em conjunto com este relatório: *populate1.sql* e *populate2.sql*, respetivamente.

Infelizmente não é possível preencher a base de dados com valores verdadeiros, já que tal informação é privada a cada casino. Assim, o ficheiro *populate1.sql* contém jogadores e valores fictícios criados manualmente, mas que se aproximam o mais possível da realidade e que mantêm a coerência que se espera de tais dados.

Inicialmente a inteligência artificial acompanhou o desenvolvimento de ficheiros SQL que gerassem automaticamente dados de forma aleatória, permitindo uma maior quantidade de entradas na base de dados. Após muitas refinações e conversas com a inteligência artificial numa tentativa de gerar estes dados automaticamente com SQL, a IA sugeriu escrever um programa em Python; o código atual permite criar qualquer número de entradas desejadas com um nível de coerência aceitável, cumprindo assim com todas as restrições impostas. O ficheiro *populate2.sql* é, portanto, o resultado de uma iteração aleatória desse mesmo código.

Os nomes usados no ficheiro populate2.sql são provenientes do repositório livre "gerador-nomes" criado por João Antunes. Todas as informações sobre o projeto "Central de Dados" e os nomes encontram-se em https://github.com/centraldedados/gerador-nomes.

7. O uso de IA no projeto

Ao longo da construção da base de dados, foi pedido ao *ChatGPT*, a inteligência artificial da *OpenAI* escolhida para este projeto, para melhorar vários aspetos e confirmar se os resultados das soluções iniciais deveriam ser melhorados e se estavam corretos de todo.

Conclui-se que esta ferramenta foi muito útil para detetar erros de distração e, no caso concreto da modelação de uma base de dados, para recomendar boas práticas, como aplicar restrições, normalizar e regras, por exemplo. No entanto, muitas das sugestões dadas ao longo deste projeto não são aplicáveis de todo ou são melhorias que já estavam presentes no projeto ou estão completamente erradas.

É de notar, portanto, que:

- Aplica-se um trabalho extra em filtrar os resultados destes modelos;
- Estas ferramentas ainda não são confiáveis o suficiente para construir uma base de dados que satisfaça as expectativas que este projeto exige;
- Alguns dos passos deste projeto não beneficiaram do uso de inteligência artificial;

8. Contribuição de cada membro do grupo

Cada um teve uma participação ativa e relevante na elaboração do trabalho:

- O Duarte encarregou-se de definir as dependências funcionais, conversar com a inteligência artificial, de resumir as restrições e de melhorar a base de dados;

- O Guilherme encarregou-se de preencher a base de dados, de definir as dependências funcionais e de desenvolver o "*script"* de Python juntamente com a inteligência artificial;

- O João encarregou-se de desenhar o UML refinado, de criar o esquema relacional, de definir as dependências funcionais e de definir a base de dados em SQLite.

Juntos, colaboramos de forma eficaz, combinando os nossos pontos fortes únicos para desenvolver um modelo abrangente de base de dados de um casino. O esforço coletivo garantiu o sucesso do estabelecimento de relações, a identificação de dependências e a implementação prática em SQLite, contribuindo significativamente para o sucesso do projeto.