FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório de Base de Dados - 2.ª Fase

1º Semestre

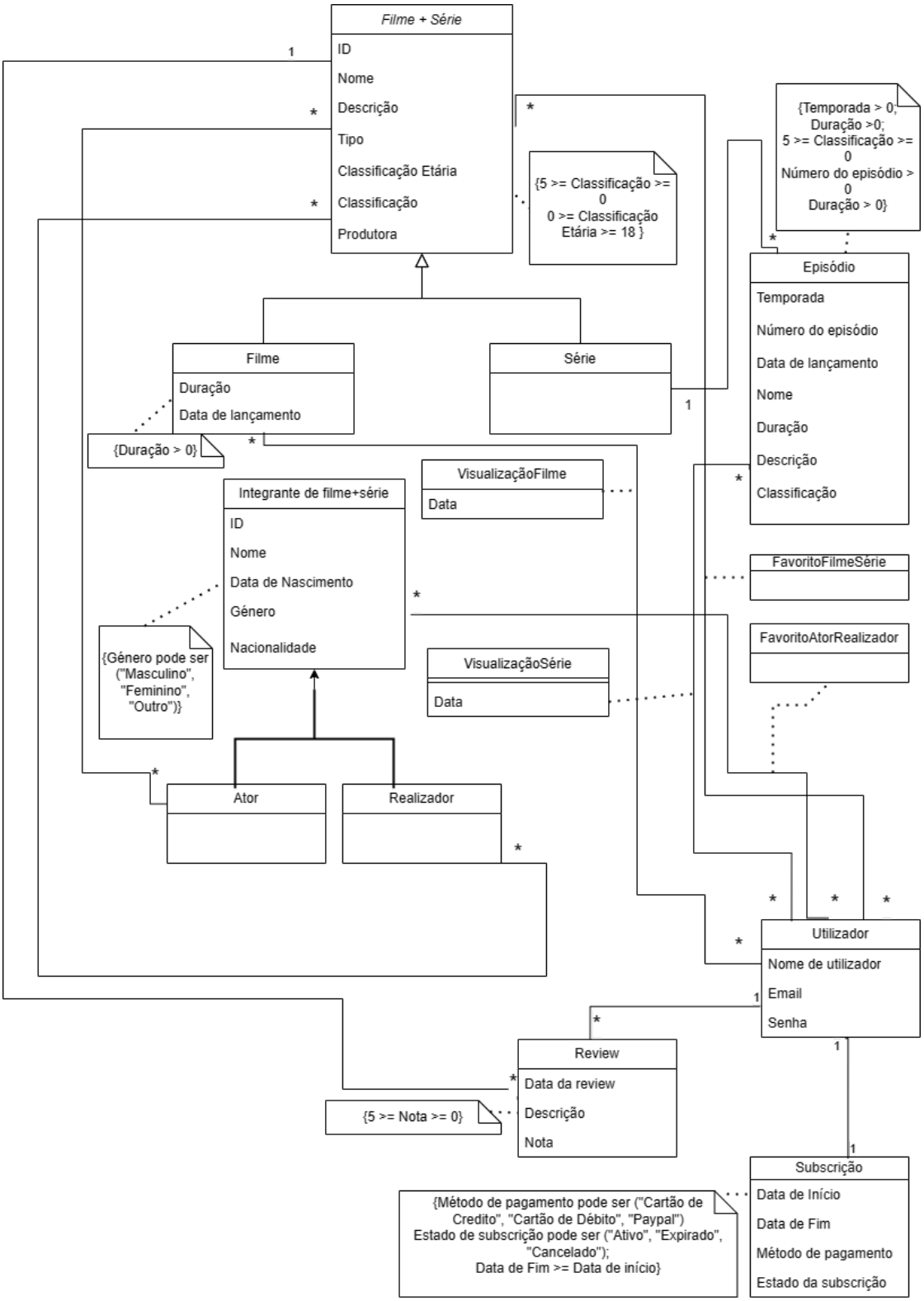
Ano letivo
2023/2024

Turma 4 -Grupo 1
Francisco Magalhães - up202007945
Lucas Faria - up202207540
Rodrigo Sousa - up202207292

Índice

1. Modelo Conceptual Melhorado

Após feedback da primeira entrega do projeto foram feitas melhorias ao modelo conceptual UML:



As melhorias feitas foram:

- Eliminação da classe temporada, tendo esta passado a ser um atributo da classe episódio;
- Adição de várias restrições em diversas classes de forma a controlar os valores dos seus atributos;
- Remoção do atributo Tipo de subscrição e adição do atributo Data de Fim na classe Subscrição;

2. Modelo Relacional

- Utilizador (Nome de utilizador, email, senha)

- Subscrição (SubscriçãoID, data de início, data de fim, método de pagamento, estado da subscrição, nome de utilizador -> Utilizador)

- Review (ReviewID, data da review, descrição, nota, Nome de utilizador-> Utilizador, Filme+SérieID->Filme+Série)

- Filme+Série (Filme+SérieID, nome, descrição, tipo, classificação etária, classificação, produtora)

- Filme (Filme+SérieID->Filme+Série, duração, data de lançamento)

- Série (Filme+SérieID->Filme+Série)

- Episódio (EpisódioID, temporada, número do episódio, data de lançamento, nome, duração, descrição, classificação, Filme+SérieID -> Série)

- VisualizaçãoFilme (VisualizaçãoID,FilmeID -> Filme, Nome de utilizador-> Utilizador, data)

- VisualizaçãoSérie (VisualizaçãoID,SérieID -> Série, Nome de utilizador-> Utilizador, data)

- Integrante de filme+série (integranteID, nome, data de nascimento, género, nacionalidade)

- Ator (integranteID -> integrante de filme+série)

- Realizador(integranteID -> integrante de filme+série)

- ParticipaAtor(integranteID -> Ator, Filme+SérieID -> Filme+Série)

- ParticipaRealizador(integranteID -> Realizador, Filme+SérieID -> Filme+Série)

- FavoritoFilmeSérie (Nome de utilizador -> Utilizador, Filme+SérieID -> Filme+Série)

- FavoritoAtorRealizador (Nome de utilizador -> Utilizador, integranteID -> integrante de filme+série)

3. Dependências Funcionais e as Formas Normais

3.1 Dependências Funcionais

- Utilizador:
    - Nome de utilizador -> {email, senha}
    - email -> {senha, Nome de utilizador}

- Subscrição:
    - SubscriçãoID -> { data de início, data de fim, método de pagamento, estado da subscrição, nome de utilizador}
    - nome de utilizador -> {data de início, data de fim, método de pagamento, nome de utilizador}

- Review:
    - ReviewID -> {data da review, descrição, nota, Nome de utilizador , Filme+SérieID}
    - {Nome de utilizador, Filme+SérieID} -> {ReviewID, data da review, descrição, nota}

- Filme+Série:
    - Filme+SérieID -> {nome, descrição, tipo, classificação etária, classificação, produtora}
    - {nome, descrição} -> {tipo, classificação etária, classificação, produtora, Filme+SérieID}

- Filme:
    - Filme+SérieID -> {duração, data de lançamento}

- Série:
    - Filme+SérieID -> Filme+SérieID (FD trivial)

- Episódio:
    - EpisódioID -> { temporada, número do episódio, data de lançamento, nome, duração, descrição, classificação, Filme+SérieID}

    - {temporada, número do episódio, Filme+SérieID } -> { data de lançamento, nome, duração, descrição, classificação, EpisódioID}

    - {temporada, nome, Filme+SérieID } -> {número do episódio, data de lançamento,duração, descrição, classificação, Filme+SérieID, EpisódioID}

    - {temporada, descrição, Filme+SérieID } -> {número do episódio, data de lançamento, nome, duração, classificação, EpisódioID}

- VisualizaçãoFilme:
    - VisualizaçãoID -> {FilmeID, Nome de utilizador, data}

- VisualizaçãoSérie:
    - VisualizaçãoID -> { SérieID, Nome de utilizador, data}

- Integrante de filme+série:
    - IntegranteID -> {nome, data de nascimento, género, nacionalidade}

- Ator:
    - IntegranteID -> IntegranteID (FD trivial)

- Realizador:
    - IntegranteID -> IntegranteID (FD trivial)

- ParticipaAtor:
    - {integranteID, Filme+SérieID} -> {integranteID, Filme+SérieID} (FD trivial)

- ParticipaRealizador:
    - {integranteID, Filme+SérieID} -> {integranteID, Filme+SérieID} (FD trivial)

- FavoritoFilmeSérie:
    - {Nome de utilizador, Filme+SérieID} -> {Nome de utilizador, Filme+SérieID} (FD trivial)

- FavoritoAtorRealizador:
    - {Nome de utilizador, integranteID} -> {Nome de utilizador, IntegranteID} (FD trivial)

É de notar que as dependências funcionais triviais apenas aparecem nas relações em que não existem outras dependências para além dessas. Nesses casos, as relações aparecem marcadas com a indicação “FD trivial” de forma a serem mais facilmente identificáveis. Também foram excluídas dependências funcionais redundantes ou que não fossem mínimas de forma a simplificar os passos seguintes.

3.2 Violações à Forma Normal de Boyce-Codd e 3ª Forma Normal

Nas dependências funcionais de cada relação, o “lado esquerdo” é apenas composto por keys/superkeys visto que a partir destas foi possível obter todos os outros elementos da relação no “lado direito”. Assim, visto que tanto a forma Normal de Boyce-Codd e a 3ª Forma Normal são verificadas se os lados esquerdos forem superkeys, então não existem violações às duas formas normais.

3.3 Integração de A.I

Após a consulta do A.I chatGPT, foram-nos sugeridas algumas alterações ao nosso modelo relacional, entre as quais:

- Alteração da primary key da relação Utilizadores, de userID para nome de utilizador, uma vez que em grande parte dos sites e apps de streaming existentes, os nomes de utilizadores dos usuários são únicos e suficientes para referenciar todas as informações relativas a uma dada conta.

- Remoção das subclasses ator e realizador, uma vez que estas subclasses não acrescentam qualquer informação que já não esteja presente na superclasse integrantes de filmes+séries. Devido a esta remoção seria também necessário unificar as relações de participação em filmes+séries numa só relação e modificá-la para incluir um atributo “papel desempenhado” que guardasse o papel que cada integrante teve num dado filme ou série(por exemplo ator, realizador,etc). Esta sugestão não foi adotada, porque não nos pareceu que fosse melhorar de qualquer forma o nosso modelo relacional.

- Separação da relação episódio em duas sub relações, uma que associasse uma série, uma temporada e um dado episódio a um ID e outra que associasse o respetivo ID a detalhes específicos sobre um determinado episódio, como a sua data de lançamento, descrição e duração. Esta sugestão não foi aceite, uma vez que nos obrigaria a criar mais uma tabela e assim, ocupar mais espaço de memória com a nossa base de dados, sem grande necessidade.

- Uma sugestão semelhante à anterior foi feita também para a relação review, tendo sido rejeitada pelos mesmos motivos.

- Alteração da estrutura e propósito da relação subscrição, passando esta a guardar o histórico de subscrições dos utilizadores em vez de guardar apenas a sua subscrição mais recente. O AI sugeriu esta mudança por ter considerado que a mesma simplificaria substancialmente o nosso modelo relacional, já que guardar usar a relação subscrição para guardar apenas a subscrição mais recente associada a cada utilizador nos obrigaria a impor restrições bastante mais complicadas ao nosso modelo relacional e nos poderia obrigar, mais tarde depois de implementar o modelo em SQL, a ter que estar permanentemente a atualizar os dados das subscrições de cada utilizador. Esta sugestão foi aceite, passando assim a relação subscrição a ter uma relação many-to-one com utilizador e sendo a dependência funcional “nome de utilizador -> {data de início, data de fim, método de pagamento, nome de utilizador}” eliminada do modelo relacional.

- A remoção do atributo estado da subscrição da tabela subscrições, uma vez que, por um lado, este atributo é algo desnecessário, já que através do atributo data de fim da subscrição, já é possível avaliar o seu estado e por outro, os valores associados a este atributo necessitam sempre de ser modificados e atualizados pelo menos uma vez, dado que o estado de cada subscrição ativa terá de ser manualmente modificado para um estado inativo de cada vez que a subscrição expirar ou for cancelada. Esta sugestão acabou por ser adotada, resultando na remoção do atributo em questão da relação subscrição do nosso modelo relacional.

4. Criação da base de dados em SQL com ajuda de A.I

Após a criação de um ficheiro SQL com todas as tabelas e seus atributos presentes no nosso modelo relacional, pedimos a uma A.I (*Microsoft Copilot* neste tópico) para nos ajudar a melhorar e detetar possíveis erros que possamos ter cometido. Assim, as sugestões obtidas forma as seguintes:

- Usar tipo de dados INTEGER em vez de INT pois, apesar de ambos serem entendidos pelo SQlite, o INTEGER é mais consistente com a documentação oficial e apresenta uma maior compatibilidade entre diferentes sistemas de gestão de bases de dados.

- Foi dada a sugestão de usar o modificador AUTOINCREMENT nas chaves primárias e estrangeiras com id e serem únicas de forma a evitar possíveis conflitos. Contudo,

tal não é necessário pois é pouco provável que conflitos ocorram numa base de dados mais pequena, além de que esta opção aumenta consideravelmente o uso de recursos de Hardware e por isso apenas deve ser usada em casos estritamente necessários.

- Uso de TEXT em vez de VARCHAR de forma a não ter um limite no tamanho das strings armazenadas e, além disso, obter uma maior compatibilidade com diferentes sistemas de gestão de bases de dados.

- Usar REAL em vez de FLOAT, tendo em conta que este é mais curto e consistente com a documentação oficial. Neste caso, a diferença não justifica a mudança na implementação SQL.

Desta forma, foram escolhidas as primeira e terceira sugestões visto que são as que mais poderão beneficiar a implementação SQL.

6. Povoação da base de dados com a ajuda de A.I

Após a criação das tabelas da base de dados em SQL, pedimos à A.I (*Microsoft Copilot* neste tópico) para nos ajudar a povoá-la. Assim, surgiram os seguintes problemas:

- Os dados fictícios fornecidos pela A.I não são muito realistas, por exemplo, os nomes de utilizadores são muito genéricos e as descrições das reviews têm uma estrutura muito parecida umas com as outras.

- O A.I enganou-se muitas vezes, principalmente com os valores que são foreign key de outras tabelas.

- Não é possível ter mais que 30 iterações seguidas com a A.I, sendo que a partir de 20 iterações a qualidade das respostas degradasse ligeiramente. Esta é uma limitação específica da A.I utilizada.

- Após um certo número de palavras escritas pela A.I, esta pára e temos que lhe pedir para continuar a resposta. Esta é outra limitação da A.I que limita bastante o seu uso com quantidades de dados grandes, como é o caso dos episódios de cada série ou os atores e realizadores dos filmes e séries.

- A A.I não conseguiu nos ajudar nas tabelas de participação, pois nestes casos em que é preciso associar duas foreign keys de diferentes tabelas, o A.I não consegue associar os valores corretamente.

Apesar desses problemas devemos destacar que:

- O A.I possibilitou obter os dados para os filmes e séries de uma forma já corretamente formatada e pronta a ser colocada no código SQL. Além disso, as informações estavam todas corretas e na sua resposta a A.I colocou a referência para a fonte dos dados.

- Por outro lado, também foi possível obter dados para os atores e realizadores de uma forma corretamente formatada e pronta a ser colocada no código SQL, sendo que as informações apresentadas pela A.I também estavam corretas e acompanhadas de referências para as fontes.

7.Considerações finais

As ferramentas usadas durante a execução desta fase do projeto foram o *Chat Gpt* na sua versão gratuita, o Microsoft Copilot na sua configuração “criativa”, o drawn.io e o Google Docs. Relativamente à participação dos elementos do grupo, podemos concluir que todos trabalharam de forma empenhada para a conclusão deste trabalho.