INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS (IFMG) - CAMPUS BAMBUÍ
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

Arthur Albino Resende
Daiana Aparecida Garcia Oliveira

TRABALHO PRÁTICO : **Sistema de memorização espaçada**

Bambuí
2024

ARTHUR ALBINO RESENDE
DAIANA APARECIDA GARCIA OLIVEIRA

TRABALHO PRÁTICO : **Sistema de memorização espaçada**

Trabalho Prático apresentado ao Curso Bacharelado em Engenharia da Computação do Instituto Federal de Minas Gerais - *Campus* Bambuí para aprovação na disciplina.
Professor : Felipe Lopes de Melo Faria

Bambuí
2024

SUMÁRIO

INTRODUÇÃO

Este relatório apresenta o desenvolvimento de um sistema para a empresa fictícia Learn, que se especializa no ensino de línguas estrangeiras para pessoas de todas as idades e gêneros. A empresa busca aprimorar a retenção do conteúdo estudado pelos seus alunos através da implementação de um sistema de repetição espaçada.

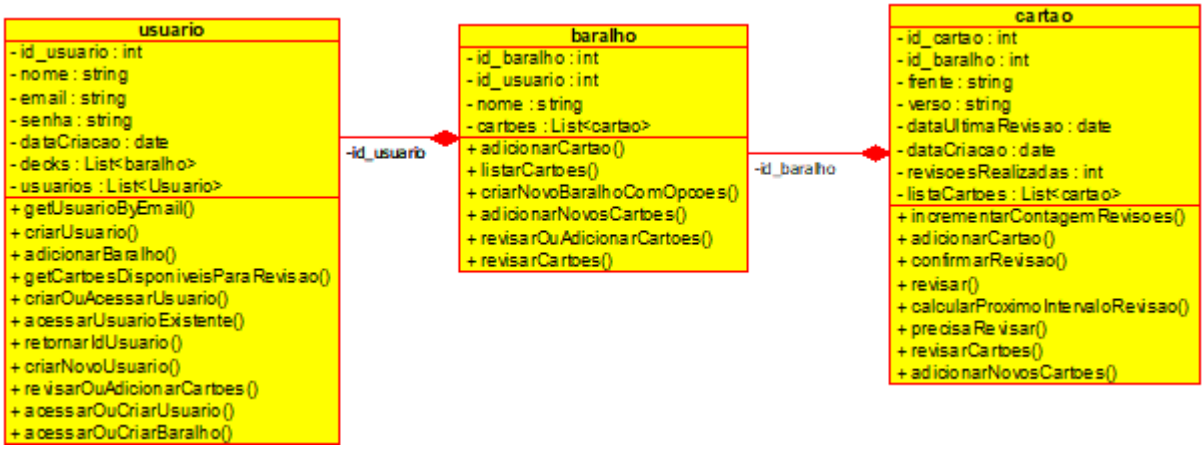
O sistema de repetição espaçada é uma técnica de memorização baseada na curva do esquecimento de Ebbinghaus, amplamente reconhecida como uma das abordagens mais eficazes para o aprendizado de vocabulário. Esse método parte do princípio, proposto por Ebbinghaus, de que a informação precisa ser revisitada em intervalos regulares para ser consolidada de forma duradoura na memória.

O sistema desenvolvido permite que múltiplos usuários se registrem, cada um podendo criar e gerenciar seus próprios baralhos e cartões de estudo. Além disso, o sistema automatiza a revisão dos cartões em intervalos predeterminados, como imediatamente após a criação, e posteriormente em períodos de 1, 7, 30 dias, entre outros. Para garantir a persistência dos dados, foi utilizado o banco de dados PostgreSQL.

DESENVOLVIMENTO

Durante a fase de planejamento, foram estabelecidos os principais objetivos do sistema e identificados os requisitos essenciais, como o cadastro de usuários, a criação e o gerenciamento de baralhos e cartões, além da revisão dos cartões em intervalos programados. Com base nesses requisitos, foi desenvolvido o seguinte diagrama de classes:

Figura 1 : Diagrama de classes



O diagrama de classes é composto por três entidades principais: Usuário, Baralho e Cartão. Ele serve como uma representação visual da estrutura fundamental do sistema, delineando as relações e interações entre essas entidades.

A classe Usuário representa os indivíduos registrados no sistema, cada um com a capacidade de criar e gerenciar seus próprios baralhos de cartões. A classe Baralho está intimamente relacionada ao Usuário por meio de um relacionamento de composição, refletindo que cada usuário pode possuir vários baralhos, cada um contendo diferentes conjuntos de cartões. Dentro de cada baralho, os cartões são representados como instâncias da classe Cartão. Esses cartões podem conter informações como palavras, frases, perguntas ou qualquer outro tipo de conteúdo relevante ao propósito do sistema. A relação entre Baralho e Cartão também é de composição, indicando que cada baralho pode conter múltiplos cartões, exclusivos desse baralho específico.

Além disso, o diagrama de classes inclui outros atributos e métodos relevantes para cada entidade, proporcionando uma base sólida para o desenvolvimento do sistema.

A classe Usuario possui os seguintes atributos: id_usuario, nome, email, senha, dataCriacao e decks. O id_usuario é utilizado como identificador único para cada usuário no sistema. Para autenticação, são utilizados o email e a senha. Cada usuário pode criar e gerenciar seus próprios baralhos (decks).

Entre os principais métodos da classe Usuario, destacam-se:

- criarNovoUsuario(): Permite a criação de um novo usuário. São necessários o nome, o email e a senha desejados. O id_usuario é gerado automaticamente pelo banco de dados.
- acessarUsuarioExistente(): Permite que um usuário existente acesse sua conta. Para isso, o usuário deve inserir seu email e senha. O método realiza a busca no banco de dados e valida o acesso.
- revisarOuAdicionarCartoes(): Permite ao usuário revisar os cartões existentes ou adicionar novos, buscando os baralhos associados no banco de dados.

A classe Baralho possui os atributos id_baralho, nome e cartões. O id_baralho é gerado automaticamente pelo banco de dados, e o id_usuario é obtido da classe Usuário.

Entre os principais métodos da classe Baralho, destacam-se:

- criarNovoBaralhoComOpcoes(): Possibilita a criação de novos baralhos. Requer autenticação do usuário e solicita o nome desejado para o novo baralho. Após a inserção no banco de dados, o id_baralho gerado é obtido.
- revisarCartoes(): Permite ao usuário revisar os cartões de um baralho específico, recebendo o baralho escolhido e a lista de cartões como parâmetros.

A classe Cartao possui atributos essenciais como id_baralho, id_cartao, frente, verso, dataUltimaRevisao, dataCriacao, revisoesRealizadas e listaCartoes. O id_baralho é derivado da classe Baralho, enquanto o id_cartao é gerado automaticamente pelo banco de dados. Os atributos frente e verso armazenam o conteúdo frontal e traseiro do cartão, respectivamente. Os atributos dataUltimaRevisao e revisoesRealizadas são cruciais para calcular a próxima revisão do cartão.

Dentre os principais métodos da classe Cartao, destacam-se:

- incrementarContagemRevisoes(): Incrementa o contador de revisões realizadas no cartão.
- adicionarCartao(): Solicita ao usuário as informações de frente e verso do cartão, obtém o id_baralho do banco de dados e salva o cartão com a ajuda de um método adicional.
- calcularProximoIntervaloRevisao(): Calcula o próximo intervalo de revisão com base na contagem atual de revisões: 1 dia para a primeira revisão, 7 dias para a segunda e 30 dias para três ou mais revisões.
- precisaRevisar(): Verifica se o cartão precisa ser revisado. Se nunca foi revisado ou se o intervalo calculado somado à data da última revisão é igual ou anterior à data atual, o cartão é marcado para revisão.

Além das classes principais, foram desenvolvidas classes auxiliares essenciais para garantir o correto funcionamento do sistema, especialmente no gerenciamento do banco de dados. Essas classes fornecem uma interface eficiente para acessar e manipular os dados armazenados, facilitando operações como inserção, atualização, exclusão e consulta, garantindo a integridade e consistência dos dados em todo o sistema. Com essas classes, tornou-se possível uma interação eficaz entre a lógica do programa e o banco de dados subjacente, assegurando um desempenho otimizado e uma experiência confiável para os usuários.

A classe User001 oferece os seguintes métodos principais para o gerenciamento do banco de dados:

- salvarUsuario(): Recebe um objeto Usuario contendo os atributos nome, email, senha e dataCriacao, e os armazena no banco de dados.
- getUsuarioByEmailSenha(): Recebe um email e senha como entrada e, se encontrados no banco de dados, retorna o id_usuario, a dataCriacao e o nome associados.

A classe Card001 apresenta os seguintes métodos principais:

- buscarCartoes(): Realiza uma busca no banco de dados e retorna uma lista de objetos Cartao contendo os atributos id_cartao, frente, verso, dataUltimaRevisao, dataCriacao e revisoesRealizadas.

- inserirCartaoNoBancoDeDados(): Recebe o id_baralho e um objeto Cartao preenchido com informações como frente, verso, dataCriacao e id_baralho, e os armazena no banco de dados.
- atualizarDataUltimaRevisaoNoBancoDeDados(): Recebe um objeto Cartao e atualiza a dataUltimaRevisao associada a ele.
- buscarCartoesPorBaralho(): Recebe o id_baralho como entrada e retorna os cartões relacionados a ele.

Por fim, a classe Desk001 conta com os seguintes métodos principais de gerenciamento no banco de dados:

- inserirBaralhoNoBancoDeDados(): Recebe o id_usuario e o nome do baralho como entrada, insere as informações no banco de dados e obtém o id_baralho gerado.
- acessarBaralhoExistente(): Recebe um objeto Usuario como entrada e retorna os baralhos associados ao seu id_usuario.

CONCLUSÃO

A abordagem de Orientação a Objetos desempenhou um papel essencial em todas as etapas do projeto. Desde a fase inicial de planejamento, onde foram identificados os requisitos principais e definidos os objetivos do sistema, até a implementação das classes principais, como Usuário, Baralho e Cartão, a orientação a objetos proporcionou uma base sólida e flexível para o desenvolvimento do sistema.

A utilização de classes auxiliares para o gerenciamento do banco de dados, como User001 e Card001, exemplifica como a orientação a objetos facilita a organização e manutenção do código, além de garantir a integridade e consistência dos dados em todo o sistema.

Os métodos principais implementados em cada classe, como salvarUsuario(), buscarCartoes() e inserirBaralhoNoBancoDeDados(), demonstram a eficácia da orientação a objetos na implementação de funcionalidades específicas. Essa abordagem permitiu uma interação eficiente com o banco de dados e proporcionou uma experiência confiável para os usuários.

Em resumo, a orientação a objetos foi fundamental para o sucesso do projeto, permitindo a criação de um sistema de memorização espaçada eficiente, escalável e adaptável.

REFERÊNCIAS

Mosalingua. **O Sistema de Repetição Espaçada (SRS):** memorizar para jamais esquecer. Disponível em: https://www.mosalingua.com/pt/o-sistema-de-repeticao-espacada-memorizar-para-jamais-esquecer/. Acesso em: 16 maio 2024.

Blogdoead. **Conheça a Repetição Espaçada, método que vai turbinar seus estudos**. Disponível em: https://www.blogdoead.com.br/tag/metodologias-de-estudo/repeticao-espacada. Acesso em: 16 maio 2024.

Basmo. **Conheça a Repetição Espaçada, método que vai turbinar seus estudos**. Disponível em: https://www.blogdoead.com.br/tag/metodologias-de-estudo/repeticao-espacada. Acesso em: 16 maio 2024.

Essential. **Sistema de Repetição Espaçada para Ensinar e Aprender um Idioma**. Disponível em: https://essential.com.br/sistema-de-repeticao-espacada/. Acesso em: 16 maio 2024.

APÊNDICE

Link do projeto no GitHub: https://github.com/daianagarcia2805/Programa-o-Orientada-a-Objetos