INSTITUTO FEDERAL
Minas Gerais
Campus
Ouro Branco

Instituto Federal de Minas Gerais - Campus Avançado Ouro Branco
Curso: Bacharelado em Sistemas de Informação
Disciplina: Banco de Dados
Professor: Suelen Mapa de Paula (suelen.mapa@ifmg.edu.br)
Alunos:
Amanda Nelva Almeida Martins (0018739@academico.ifmg.edu.br)
Annah Louise Farias de Andrade (0078858@academico.ifmg.edu.br)
Glauber Vinicius Campos Batista (0078859@academico.ifmg.edu.br)
Monique Evelin Miranda Domingos (0076892@academico.ifmg.edu.br)

## Introdução

**Estrutura Geral do Jogo:** O trabalho apresentado implementa um jogo no formato quiz de perguntas e respostas, tendo como inspiração o programa "Show do Milhão" do SBT.
Foi utilizado o framework Flask para a criação da aplicação web e o MySQL como banco de dados local. A aplicação permite aos usuários se cadastrarem, fazerem login, participarem de partidas, consultarem estatísticas e muito mais.

## Parte 1

**Estrutura de Perguntas:**

- **Perguntas:** O jogo é composto por 10 perguntas. Cada pergunta correta leva o jogador a um prêmio em dinheiro acumulativo. As perguntas são de múltipla escolha, com quatro alternativas e abordam temas diversos como história, geografia, ciência, cultura geral, esportes, etc.
- **Fases:** A cada pergunta respondida corretamente, o jogador passa de fase. A cada fase concluída, a pontuação aumenta.

**Perguntas e Respostas:**

- **Nível das Perguntas:** As perguntas são aleatórias, com níveis de dificuldade variados.
- **Partida:** Durante a partida o jogador pode escolher parar caso não saiba a resposta, assim não perde a pontuação conquistada.

**Prêmios e Progresso:**

- **Valores:** A cada pergunta correta, o jogador acumula uma quantidade de dinheiro. O valor é acumulado a cada rodada concluída.

**Tabela geral dos valores por pergunta:**

**1ª Pergunta:** R$ 100.000

**2ª Pergunta:** R$ 100.000

**3ª Pergunta:** R$ 100.000

**4ª Pergunta:** R$ 100.000

**5ª Pergunta:** R$ 100.000

**6ª Pergunta:** R$ 100.000

**7ª Pergunta:** R$ 100.000

**8ª Pergunta:** R$ 100.000

**9ª Pergunta:** R$ 100.000

**10ª Pergunta (Prêmio Máximo):** R$ 1.000.000

**Regras de Eliminação e Avanço:**

- **Eliminação:** Se o participante responder incorretamente, ele é eliminado e sua pontuação é zerada.
- **Avanço:** Se o participante responder corretamente, ele avança para a próxima fase e tem a chance de ganhar prêmios maiores. O jogo continua até o participante responder a 10 perguntas corretamente, ou até decidir parar ou errar uma resposta. Caso o jogador decida parar antes de responder a pergunta da rodada, ele recebe a pontuação acumulada até o momento.

**6. Regras Gerais:**

- **Participantes:** o jogo se joga individualmente.

**7. Questões Especiais:**

- **Desistência:** O participante pode optar por desistir do jogo e levar os pontos acumulados até o momento. Embora não seja uma ajuda formal, o participante pode escolher parar o jogo a qualquer momento e levar o valor acumulado até a última pergunta respondida corretamente.
  - **Uso:** É uma decisão estratégica, especialmente se o jogador se sentir inseguro para continuar, evitando assim perder dinheiro. Essa ajuda só pode ser usada uma vez durante o jogo pois a partir do momento que o jogador escolhe esta opção, a partida é automaticamente concluída.

**8. Tecnologias e Frameworks Utilizados:**

- **Flask:** Framework Python leve, flexível e de fácil entendimento para o desenvolvimento de aplicações web.
- **PyMySQL:** Conector Python para interagir com bancos de dados MySQL.
- **Pandas:** Biblioteca Python para manipulação e análise de dados (não utilizada diretamente no código principal, mas no script de inserção de dados em massa).

- **CSV:** Módulo padrão do Python para trabalhar com arquivos CSV (formato utilizado para armazenamento dos registros fictícios posteriormente inseridos no banco).
- **Chardet:** Biblioteca para detectar automaticamente o encoding de arquivos, utilizada para verificar se o formato dos caracteres do arquivo CSV era UTF-8 e prevenir que caracteres especiais ficassem desformatados no banco de dados).

**9. Requisitos de Sistema**

- **Python:** Versão compatível com as bibliotecas utilizadas (Flask, PyMySQL, Pandas, CSV, Chardet).
    - No arquivo requirements.txt, que está dentro da aplicação, temos a lista de instalações necessárias.
- **MySQL:** Banco de dados MySQL instalado e configurado, conforme o Script de geração do banco.
    - No arquivo BancodeDados, que está dentro da aplicação, temos o script de criação do banco.
- **Um servidor web:** Para hospedar a aplicação foi utilizado o próprio servidor de desenvolvimento do Flask).

**10. Estrutura do Código**

- **Configuração do banco de dados:** Estabelece a conexão com o banco de dados MySQL.
- **Rotas:** Definem as diferentes páginas da aplicação (login, cadastro, menu, jogo, estatísticas).
- **Lógica do jogo:** Implementa as regras do jogo, como verificar respostas, atualizar pontuações e rodadas.
- **Interação com o banco de dados:** Realiza consultas e atualizações no banco de dados para gerenciar usuários, partidas, perguntas e respostas.
- **Templates:** Utilizados para renderizar as páginas HTML com os dados dinâmicos da aplicação.

**11. Descrição das Funcionalidades**

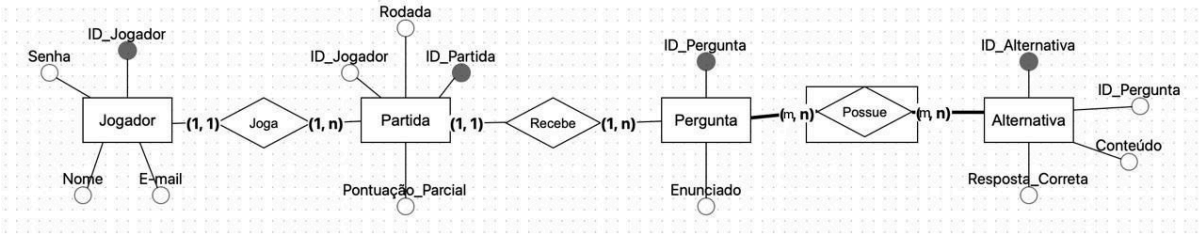
- **Cadastro de usuários:** Permite que novos jogadores se cadastrem no sistema.
- **Login de usuários:** Permite que usuários autenticados acessem suas áreas restritas.
- **Início de partidas:** Inicia uma nova partida para um jogador, associando-a ao seu cadastro.
- **Mecânica do jogo:** Apresenta perguntas, verifica respostas e atualiza a pontuação e a rodada.
- **Estatísticas:** Permite que os jogadores visualizem suas estatísticas, como número total de partidas e média de pontuação.
- **Esqueci minha senha:** Permite que os usuários recuperem suas senhas.

### 12. Fluxo da Aplicação

1. **Usuário acessa a página inicial:** O usuário é apresentado com um formulário de login ou cadastro.
2. **Cadastro:** Se o usuário é novo, ele preenche o formulário de cadastro e suas informações são armazenadas no banco de dados.
3. **Login:** Após o cadastro ou login, o usuário é direcionado para o menu principal.
4. **Iniciar partida:** Ao clicar em "Iniciar Partida", uma nova partida é criada no banco de dados e o usuário começa a responder perguntas.
5. **Jogar:** A cada pergunta, o usuário seleciona uma resposta e o sistema verifica se está correta.
6. **Finalizar partida:** A partida termina quando o usuário erra, acerta todas as perguntas ou decide parar.
7. **Estatísticas:** O usuário pode visualizar suas estatísticas e as estatísticas gerais através do menu, a qualquer momento.

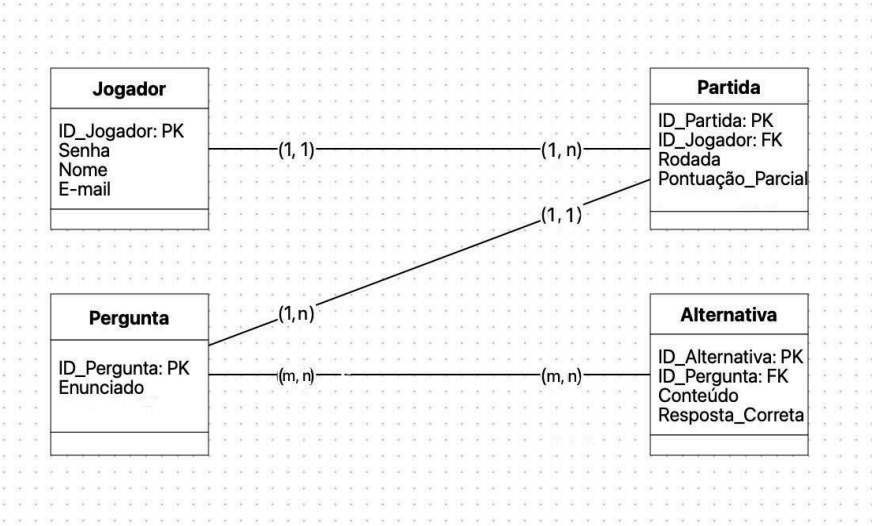
## Parte 2

**Representação Gráfica (DER):**




## Parte 3

**Modelo Lógico:**



Parte 4

**Script de Geração do Banco de Dados:**

```
-- Script de Criação do Banco

create database showdomilhao
default character set utf8;
use showdomilhao;

 create table pergunta(
idpergunta INTEGER NOT NULL AUTO_INCREMENT UNIQUE,
enunciado VARCHAR(255) not null,
 PRIMARY KEY (idpergunta));

create table jogador(
idjogador INTEGER NOT NULL AUTO_INCREMENT UNIQUE,
nome VARCHAR(30) NOT NULL,
email VARCHAR(30) NOT NULL,
senha VARCHAR(12) NOT NULL,
PRIMARY KEY (idjogador));

create table alternativa(
idalternativa INTEGER NOT NULL AUTO_INCREMENT UNIQUE,
idpergunta INTEGER NOT NULL,
conteudo VARCHAR(255) NOT NULL,
alternativacorreta BOOL NOT NULL,
 PRIMARY KEY (idalternativa),
 FOREIGN KEY (idpergunta)
 REFERENCES pergunta (idpergunta));

create table partida(
idpartida INTEGER NOT NULL AUTO_INCREMENT UNIQUE,
PRIMARY KEY (idpartida),
idjogador INTEGER NOT NULL,
pontuacaoparcial INTEGER,
rodada INTEGER NOT NULL,
FOREIGN KEY (idjogador)
REFERENCES jogador(idjogador));
```

**Script de inserção de registros nas tabelas:**

Para inserção em massa dos registros nas tabelas, devemos executar um script de cada vez. Não foi possível inserir os dados em todas as tabelas de uma única vez.

Primeiro precisamos realizar a conexão com o banco de dados, através das configurações abaixo:

```
import mysql
import csv
import pymysql

# Conectar ao banco de dados
conn = pymysql.connect(
  host="localhost",
  user="root",
  password="Minas@0202",
  database="Showdomilhao",
  charset="utf8mb4"
)
```

Depois realizamos a inserção através dos scripts abaixo, lembrando que em todas as inserções o código deve conter o trecho acima para conexão ao banco:

Script de inserção dos registros na tabela pergunta:

```
cursor = conn.cursor()

# Passamos o caminho absoluto e o nome do arquivo CSV
arquivo_csv =
'C:/Users/AmandaeLuiz04/Downloads/Trabalho_Banco_De_Dados/Trabalho_Banco_De_Da
dos/BackEnd/pergunta.csv'
tabela = 'pergunta'  # inserção de dados da tabela pergunta

# Abrir o arquivo CSV
with open(arquivo_csv, 'r', encoding='utf-8') as csvfile:
    csvreader = csv.reader(csvfile)
    next(csvreader)  # Ignorar o cabeçalho

    for row in csvreader:
        sql = f"INSERT INTO {tabela} (enunciado) VALUES (%s)"
        cursor.execute(sql, row)

# Confirmar as alterações
conn.commit()

# Fechar a conexão
conn.close()
```

Script de inserção dos registros na tabela alternativa:

```
cursor = conn.cursor()
```

```
# Passamos o caminho absoluto e o nome do arquivo CSV
arquivo_csv =
'C:/Users/AmandaeLuiz04/Downloads/Trabalho_Banco_De_Dados/Trabalho_Banco_De_Da
dos/BackEnd/alternativa.csv'
tabela = 'alternativa'  # Registros da tabela alternativa

# Abrir o arquivo CSV
with open(arquivo_csv, 'r', encoding='utf-8') as csvfile:
    csvreader = csv.reader(csvfile)
    next(csvreader)  # Ignorar o cabeçalho

    for row in csvreader:
        idpergunta, conteudo, alternativacorreta = row
        sql = f"INSERT INTO {tabela} (idpergunta, conteudo, alternativacorreta) VALUES (%s,
%s, %s)"
        cursor.execute(sql, (idpergunta, conteudo, alternativacorreta))

# Confirmar as alterações
conn.commit()

# Fechar a conexão
conn.close()
```

Script de inserção dos registros na tabela jogador:

```
cursor = conn.cursor()

# Passamos o caminho absoluto e o nome do arquivo CSV
arquivo_csv =
'C:/Users/AmandaeLuiz04/Downloads/Trabalho_Banco_De_Dados/Trabalho_Banco_De_Da
dos/BackEnd/jogador.csv'
tabela = 'jogador'  # Registros da tabela jogador

# Abrir o arquivo CSV
with open(arquivo_csv, 'r', encoding='utf-8') as csvfile:
    csvreader = csv.reader(csvfile)
    next(csvreader)  # Ignorar o cabeçalho

    for row in csvreader:
        sql = f"INSERT INTO {tabela} (nome, email, senha) VALUES (%s, %s, %s)"
        cursor.execute(sql, row)

# Confirmar as alterações
conn.commit()

# Fechar a conexão
conn.close()
```

Script de inserção dos registros na tabela partida:

```
cursor = conn.cursor()

# Nome do arquivo CSV e tabela
arquivo_csv =
'C:/Users/AmandaeLuiz04/Downloads/Trabalho_Banco_De_Dados/Trabalho_Banco_De_Da
dos/BackEnd/partida.csv'
tabela = 'partida'  # inserção de dados da tabela partida

# Abrir o arquivo CSV
with open(arquivo_csv, 'r', encoding='utf-8') as csvfile:
    csvreader = csv.reader(csvfile)
    next(csvreader)  # Ignorar o cabeçalho

    for row in csvreader:
        sql = f"INSERT INTO {tabela} (idjogador, pontuacaoparcial, rodada) VALUES (%s, %s,
%s)"
        cursor.execute(sql, row)

# Confirmar as alterações
conn.commit()

# Fechar a conexão
conn.close()
```

Parte 5

**Operações de seleção simples:**

**1.**

```
SELECT j.nome, p.idpartida, p.pontuacaoparcial
FROM jogador j
INNER JOIN partida p ON j.idjogador = p.idjogador;
```

Funcionamento: Retorna o nome dos jogadores, id da partida e a pontuação nessa partida:

**2.**

```
SELECT j.nome, COUNT(p.idpartida) AS total_partidas
FROM jogador j
INNER JOIN partida p ON j.idjogador = p.idjogador
GROUP BY j.nome
ORDER BY total_partidas DESC;
```

Funcionamento: Conseguimos ver o somatório de quantas partidas já foram jogadas por cada jogador.

**Junção de duas relações:**

**3.**

```
SELECT pe.idpergunta, pe.enunciado, a.conteudo
FROM pergunta AS pe JOIN alternativa AS a
ON pe.idpergunta = a.idpergunta
WHERE a.alternativacorreta = 1;
```

Funcionamento: Retorna o gabarito geral do jogo, das 100 perguntas cadastradas

**4.**

```
SELECT j.nome, p.idpartida, p.pontuacaoparcial
FROM jogador j
INNER JOIN partida p ON j.idjogador = p.idjogador
WHERE p.pontuacaoparcial = 1000000;
```

Funcionamento: Retorna o nome dos jogadores que já ganharam o prêmio máximo e em qual partida isso ocorreu.

**Funções de agregação sobre o resultado da junção de duas ou mais relações:**

**5.**

```
SELECT j.nome, COUNT(p.idpartida) AS total_partidas
FROM jogador j
INNER JOIN partida p ON j.idjogador = p.idjogador
GROUP BY j.nome
ORDER BY total_partidas DESC;
```

Funcionamento: Conseguimos ver o somatório de quantas partidas já foram jogadas por cada jogador.

**6.**

```
SELECT j.nome, ROUND(AVG(p.pontuacaoparcial), 2) AS 'media_pontuacao'
FROM jogador j
INNER JOIN partida p ON j.idjogador = p.idjogador
GROUP BY j.nome
ORDER BY media_pontuacao DESC;
```

Funcionamento: Retorna a média geral de pontuação de cada jogador,

onde somamos a pontuação total de todas as partidas e dividimos pelo total de partidas jogadas.

Conclusão

Durante o desenvolvimento deste trabalho, a decisão do uso das tecnologias e frameworks descritos anteriormente baseou-se nos seguintes critérios:

**Simplicidade:** Tanto o Flask quanto o MySQL são relativamente fáceis de aprender e usar, o que agiliza o desenvolvimento.

**Flexibilidade:** Permitem criar uma aplicação web dinâmica e personalizada, adaptando-se às necessidades do jogo.

**Custo-benefício:** São tecnologias open-source, o que reduz os custos de desenvolvimento e licenciamento.

**Adequação ao escopo do projeto:** Para um jogo como o "Show do Milhão", as funcionalidades oferecidas pelo Flask e MySQL são suficientes.

Encontramos várias dificuldades, devido a nenhum dos integrantes do grupo ter desenvolvido anteriormente um sistema completo com front end, back end e integração com banco de dados, então dividimos de acordo com as habilidades de cada um.
Por isso escolhemos o Python como linguagem de desenvolvimento devido a facilidade de aprendizado e também a grande quantidade de conteúdo de apoio disponível online.
Seguem abaixo algumas das dificuldades e limitações do sistema:

**Escalabilidade:** No momento o jogo está disponível apenas através de um servidor local, com um jogador de cada vez. Caso quisessemos ter um jogo com grande número de usuários simultâneos, o Flask e o MySQL poderiam precisar de otimizações ou até mesmo a migração para tecnologias mais robustas, como Node.js ou um banco de dados NoSQL.

**Segurança:** A proteção dos dados dos usuários, como senhas e informações pessoais, é fundamental. No momento não temos medidas para evitar a quebra de senhas. É preciso implementar medidas de segurança adequadas, como criptografia das senhas e validação de entradas.

**Manutenção:** À medida que o jogo evolui, a manutenção do código pode se tornar mais complexa. Não conseguimos otimizar de maneira eficaz a divisão de métodos e funções utilizadas e acabamos ficando com uma função (buscar_pergunta) que

faz várias coisas dentro do jogo. É importante adotar boas práticas de programação e utilizar ferramentas de versionamento para facilitar a gestão do código.

**Desempenho:** Para garantir uma boa experiência do usuário, é necessário otimizar o desempenho da aplicação, caso formos implementar consultas mais complexas ao banco de dados.

**Escalabilidade do banco de dados:** Conforme a quantidade de dados cresce, pode ser necessário otimizar o banco de dados, criar índices e ajustar a configuração do servidor.

**Gerenciamento de sessões:** A gestão de sessões dos usuários é necessária para garantir a segurança e a experiência do usuário. Caso fôssemos implementar sessões para mais de um jogador ao mesmo tempo seria preciso escolher uma solução adequada para armazenar as informações da sessão e protegê-las contra ataques.

Bibliografia

**Documentação oficial do Flask:** https://flask.palletsprojects.com/en/2.2.x/
**Documentação oficial do MySQL:** https://dev.mysql.com/doc/
**Documentação oficial do SQLAlchemy:**
https://docs.sqlalchemy.org/en/14/core/connections.html#
**Curso de banco de Dados (Gustavo Guanabara):**
https://www.youtube.com/playlist?list=PLHz_AreHm4dkBs-795Dsgvau_ekxg8g1r
**Curso de Flask Alura:** https://cursos.alura.com.br/formacao-flask
**Repositório do Github:**
https://github.com/Amanda0711Martins/Trabalho_Banco_De_Dados