Requisitos funcionais (RF)

RF01 - Tela de Login

O aplicativo disponibilizará uma tela de login distinta para o **usuário** e para o **administrador**. Esta tela conterá os seguintes elementos:

- Campo para **e-mail**
- Campo para **senha**
- Opção para **redefinir a senha**
- Botão ou link para **fazer cadastro**

RF02 - Tela de Cadastro

O aplicativo deverá disponibilizar uma tela de cadastro contendo os seguintes campos:

- **Nome**
- **E-mail**
- **Senha**
- **Confirmar Senha**
- **Frase secreta** (utilizada para redefinição de senha)

RF03 - Tela do Administrador

O aplicativo deverá disponibilizar uma tela exclusiva para o **administrador**, contendo:

- Um botão **"Sair"** para encerrar a sessão
- Exibição de uma **mensagem de boas-vindas**, acompanhada do **nome do administrador logado**

Funcionalidades disponíveis:

- **Adicionar novos textos** a serem analisados (sem limite de quantidade)

    - Ao salvar um texto, o sistema deverá exibir:
      "Texto cadastrado com sucesso."

- **Criar quizzes** relacionados aos textos

    - Ao salvar um quiz, o sistema deverá exibir:
      "Quiz cadastrado com sucesso."

- **Alterar** obras e textos já cadastrados

- **Alterar** quizzes já criados

- **Visualizar comentários** feitos nos textos cadastrados

- **Visualizar pontuações** atribuídas pelos usuários ao aplicativo

    - O administrador **não poderá alterar** a pontuação

- **Remover comentários** inapropriados (racistas, preconceituosos, etc.)

    - Antes da exclusão, o sistema deverá perguntar:
      "Deseja realmente excluir esta mensagem?" com as opções **Sim** ou **Não**

    - Se confirmada a exclusão, deverá ser exibida:
      "Mensagem excluída com sucesso."

RF04 - Tela do Usuário

O aplicativo deverá disponibilizar uma tela para o **usuário**, com as seguintes funcionalidades:

- Exibição de uma **mensagem de boas-vindas**, com o **nome do usuário logado**

- Um botão **"Sair"**

    - Se o usuário clicar por engano, deverá aparecer a mensagem:
      "Deseja realmente sair do sistema?" com as opções **Sim** ou **Não**

Funcionalidades disponíveis:

- **Escolher obras** para leitura

- **Selecionar o quiz** referente à obra escolhida
    - Ao clicar no botão do quiz, será exibido um **pop-up** com os níveis de dificuldade: **Fácil**, **Médio**, **Difícil**
    - Após a escolha do nível:
        - O texto da obra **desaparecerá da tela**
        - Iniciará o **quiz correspondente ao nível escolhido**
        - Cada pergunta terá **cinco enunciados ou mais**, com as opções **Verdadeiro** ou **Falso**
        - Se o usuário **errar**, deverá aparecer a **mensagem com a resposta correta**
        - Se **acertar**, uma mensagem confirmará o acerto, e o sistema **avançará automaticamente** para o próximo enunciado
- Após **finalizar o quiz**, atribuir uma **nota** e (opcionalmente) fazer um **comentário**, o usuário poderá:
    - Clicar em **Salvar**:
        - As respostas e comentários serão **salvos**
        - O sistema retornará para a **tela inicial**, onde o usuário poderá escolher outro texto
    - Clicar em **Menu**:
        - Se o quiz **não tiver sido finalizado**, o sistema:
            - Voltará para o menu inicial
            - Exibirá uma mensagem informando que **todas as respostas e dados não salvos serão perdidos**

Observações:

- **Deixar comentário** será **opcional**

- **A nota atribuída ao aplicativo** será **obrigatória**

Requisitos Não Funcionais (RNF)

RNF01 - Acessibilidade

- O sistema deverá ser acessível via navegadores modernos (Chrome, Firefox, Edge, Safari) sem necessidade de plugins adicionais.

- O layout deverá ser responsivo, adaptando-se automaticamente a dispositivos móveis, tablets e desktops.

RNF02 - Desempenho

- O tempo de resposta para carregamento de páginas não deverá ultrapassar 3 segundos em conexões padrão.

- O tempo entre uma resposta correta do quiz e a próxima pergunta deve ser inferior a 1 segundo.

- O sistema deve suportar simultaneamente no mínimo 100 usuários ativos sem degradação perceptível de desempenho.

RNF03 - Segurança

- As senhas dos usuários devem ser armazenadas de forma criptografada.

- O sistema deve conter validação de entrada para evitar ataques como SQL Injection e Cross-Site Scripting (XSS).

- A redefinição de senha deverá ser possível apenas mediante verificação da **frase secreta**.

- Os dados sensíveis (nome e e-mail) deverão ser transmitidos via **HTTPS**.

- A área administrativa deverá ser protegida por autenticação com verificação de permissões.

RNF04 - Usabilidade

- A interface do usuário deve ser simples, clara e intuitiva, com botões e mensagens claras como "Texto cadastrado com sucesso".

- O sistema deve utilizar ícones e cores consistentes, além de mensagens de erro amigáveis e informativas.

- O botão de sair deve solicitar confirmação antes de encerrar a sessão.

RNF06 - Manutenibilidade

- O sistema deverá ser desenvolvido com arquitetura modular para facilitar atualizações e manutenções futuras.

- O código-fonte deverá ser comentado e estruturado seguindo boas práticas de desenvolvimento (ex: MVC ou similar).

- Quizzes, textos e comentários deverão ser gerenciados por meio de painel administrativo com fácil acesso e edição.

RNF07 - Confiabilidade

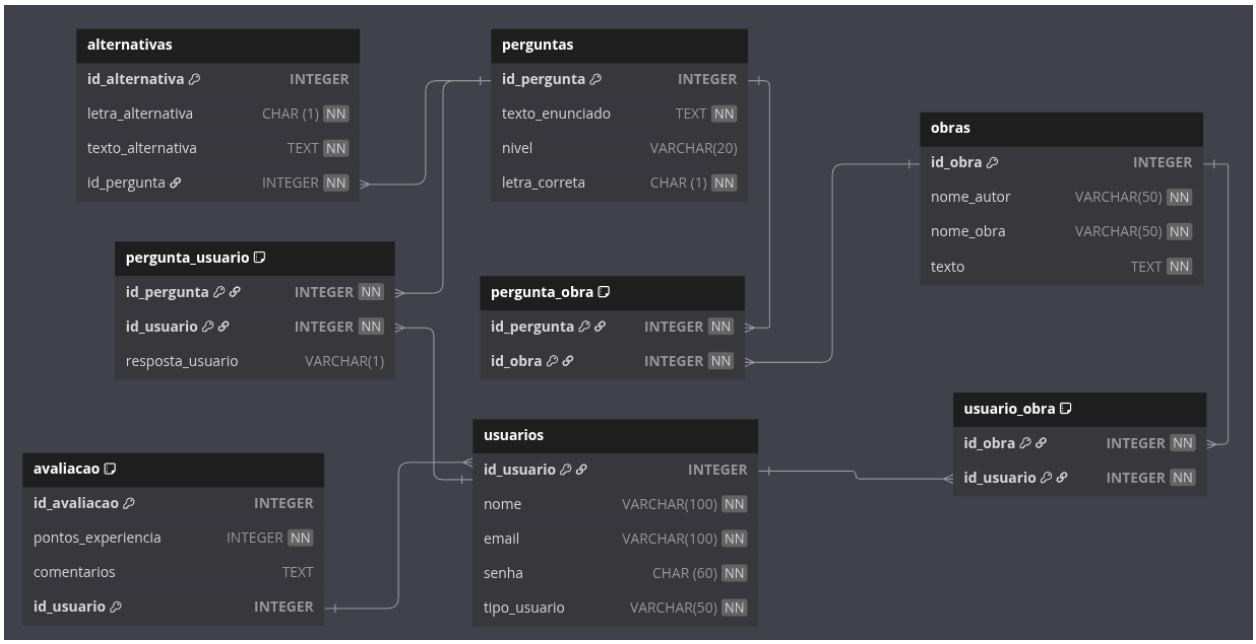
- O sistema deverá garantir que os dados salvos (comentários, notas, textos) sejam persistentes e não se percam em caso de falhas momentâneas.

- Se o usuário tentar sair ou navegar sem salvar, o sistema deverá alertar para possível perda de dados.

RNF08 - Escalabilidade

- O sistema deve estar preparado para ser escalado horizontalmente (em número de usuários, textos, quizzes) com pouco esforço.

Diagrama relacional:



Código no Postgresql

```
-- Tabela de usuários
CREATE TABLE IF NOT EXISTS usuarios (
  id_usuario INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  senha CHAR(60) NOT NULL,  -- Recomendado para hashes
  tipo_usuario VARCHAR(50) NOT NULL
);
-- Tabela de perguntas
CREATE TABLE IF NOT EXISTS perguntas (
  id_pergunta INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  texto_enunciado TEXT NOT NULL,
```

```
  nivel VARCHAR(20),
  letra_correta CHAR(1) NOT NULL -- (A, B, C, D, E)
);
-- Tabela de alternativas
CREATE TABLE IF NOT EXISTS alternativas (
  id_alternativa INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  letra_alternativa CHAR(1) NOT NULL, -- (A, B, C, D, E)
  texto_alternativa TEXT NOT NULL,
  id_pergunta INTEGER NOT NULL REFERENCES perguntas(id_pergunta)
);
-- Tabela de obras
CREATE TABLE IF NOT EXISTS obras (
  id_obra INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  nome_autor VARCHAR(50) NOT NULL,
  nome_obra VARCHAR(50) NOT NULL,
  texto TEXT NOT NULL
);
-- Tabela de avaliação
CREATE TABLE IF NOT EXISTS avaliacao (
  id_avaliacao INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  pontos_experiencia INTEGER NOT NULL,
  comentarios TEXT,
  id_usuario INTEGER NOT NULL REFERENCES usuarios(id_usuario)
```

```
);


-- Relacionamento entre perguntas e usuários
CREATE TABLE IF NOT EXISTS pergunta_usuario (
  id_pergunta INTEGER NOT NULL REFERENCES perguntas(id_pergunta),
  id_usuario INTEGER NOT NULL REFERENCES usuarios(id_usuario),
  resposta_usuario VARCHAR(1),  -- (A, B, C, D, E) para comparar com a resposta correta
  PRIMARY KEY (id_pergunta, id_usuario)
);
-- Relacionamento entre perguntas e obras
CREATE TABLE IF NOT EXISTS pergunta_obra (
  id_pergunta INTEGER NOT NULL REFERENCES perguntas(id_pergunta),
  id_obra INTEGER NOT NULL REFERENCES obras(id_obra),
  PRIMARY KEY (id_pergunta, id_obra)
);
-- Relacionamento entre usuários e obras
CREATE TABLE IF NOT EXISTS usuario_obra (
  id_obra INTEGER NOT NULL REFERENCES obras(id_obra),
  id_usuario INTEGER NOT NULL REFERENCES usuarios(id_usuario),
  PRIMARY KEY (id_obra, id_usuario)
);
-- Observação: abaixo tem um check constraint para garantir que o banco só aceite alunos ou professores
```

```sql
ALTER TABLE usuarios

ADD CONSTRAINT chk_tipo_usuario

CHECK (tipo_usuario IN ('aluno', 'professor'));

ALTER TABLE perguntas

ADD CONSTRAINT chk_letra_correta

CHECK (letra_correta IN ('A', 'B', 'C', 'D'));

ALTER TABLE alternativas

ADD CONSTRAINT chk_letra_alternativa

CHECK (letra_alternativa IN ('A', 'B', 'C', 'D'));
```