**UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO – UFERSA**

**CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS – CMPF**

**ENGENHARIA DE SOFTWARE**

**DISCENTE: VERA CÉLIA GADELHA**

**DOCENTE: ALYSSON FILGUEIRA MILANEZ**

**DISCIPLINA: GERÊNCIA DE CONFIGURAÇÃO E MUDANÇAS**

**LISTA INICIAL DOS REQUISITOS (O DESAFIO DO SABER)**

Essa Lista Inicial dos Requisitos formaliza o que o sistema deve e não deve fazer. Estes requisitos são essenciais, pois serão rastreados e verificados com seus Commits Semânticos e Testes JUnit. Então, o sistema será implementado como um aplicativo comInterface Gráfica em Java.

**Requisitos Funcionais (RFs)**

Estes definem as funções que o usuário irá executar no sistema.

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito Funcional** | **Descrição** |
| **RF1** | Carregamento de Dados | O sistema deve ler e carregar as questões, opções e respostas de um arquivo de configuração externo (JSON ou CSV), garantindo a separação do conteúdo e do código. |
| **RF2** | Seleção de Categoria | O sistema deve listar as categorias disponíveis (Matemática, Português, Cidadania/Sustentabilidade) e permitir que o usuário escolha uma via interação na interface gráfica (ex: *dropdown* ou botões). |
| **RF3** | Exibição da Questão | O sistema deve apresentar a pergunta e suas opções de resposta numeradas (ou com letras) na interface gráfica para a interação do usuário. |
| **RF4** | Processamento de Resposta | O sistema deve receber a resposta do usuário (via painel da interface gráfica) e verificar se ela corresponde à resposta correta da questão. |
| **RF5** | Cálculo e Feedback | O sistema deve fornecer *feedback* imediato ("Correto" ou "Incorreto") após cada resposta e calcular/atualizar a pontuação do jogador. |
| **RF6** | Relatório de Desempenho | Ao final de um ciclo predefinido de questões, o sistema deve exibir na interface gráfica um resumo com a pontuação total e a taxa de acerto do jogador. |

**B. Requisitos Não-Funcionais (RNFs)**

Estes definem as restrições, qualidades e características técnicas do projeto, focados nas exigências da disciplina de Gerência de Configuração e Mudanças (GCM).

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requisito Não-Funcional** | **Foco da Qualidade** | **Mensuração (Como será avaliado)** |
| **RNF1** | Interface Simples | O sistema deve ser implementado com uma interface gráfica em Java (ex.: Swing ou JavaFX), priorizando a facilidade de uso e clareza das informações. | Teste de Usabilidade: O usuário deve concluir um ciclo completo de questões (RF6) sem a necessidade de consultar instruções externas ou manuais de uso. |
| **RNF2** | Orientação a Objetos (OO) | O código deve seguir o paradigma OO, com classes bem definidas (ex: Questao, Quiz, Jogador) para garantir modularidade e manutenibilidade. | Estrutura de Classes: O projeto deve ter no mínimo 3 classes distintas que representem os elementos principais da lógica de negócio (*Questão*, *Quiz* e *Jogador*). |
| **RNF3** | Versionamento Semântico | O código-fonte deve ser versionado no GitHub, com todas as alterações documentadas por meio de *Commits* Semânticos (e.g., feat, fix, refactor). | Conformidade: Pelo menos 70% dos *commits* (do *branch* principal) devem seguir o padrão *Conventional Commits* (usando *prefixos* como feat, fix, refactor). |
| **RNF4** | Cobertura de Testes | A lógica de negócio principal (validação de resposta, pontuação) deve ser coberta por Testes de Unidade JUnit, buscando uma cobertura ≥80% para garantir a robustez. | Teste Essencial: A lógica de validação de resposta (RF4) e o cálculo de pontuação (RF5) devem ter cobertura de Teste de Unidade JUnit, com o objetivo de atingir um mínimo de 50% de cobertura de código da lógica principal. |