

**UNIVERSIDADE ESTADUAL DO CEARÁ – UECE**

**CENTRO DE CIÊNCIAS E TECNOLOGIA – CCT**

**CURSO DE LICENCIATURA EM COMPUTAÇÃO**

**Membros da Equipe**

Antonio Victor Almada Carvalho

Eugenio Alves Paulino

Sarah Lays Saraiva Grangeiro

**DOCUMENTO DE ESPECIFICAÇÃO**

**(Sistema de Gestão de Games)**

**Fortaleza – CE**

**Setembro de 2023**

**SUMÁRIO**

**1. INTRODUÇÃO**................................................................................................ 3

**2. REQUISITOS** ................................................................................................. 3

**3. CASOS DE USO** ........................................................................................... 4

**4. ARQUITETURA** ............................................................................................. 6

**5. ANEXO** .......................................................................................................... 6

**Código** .................................................................................................... 6

**Vídeo no Youtube** .................................................................................. 6

**Estrutura da Tabela** ............................................................................... 6

**1. INTRODUÇÃO**

Documento de especificação de um sistema de gestão de games com Requisitos Funcionais, Não Funcionais, Casos de Uso e Anexos para link do Github com código fonte e script de criação das tabelas.

**2. REQUISITOS**

**REQUISITOS FUNCIONAIS**

**RF01: Adicionar Games**

* Os usuários devem poder adicionar novos games à lista.
* Cada game deve ter um ID único, título do jogo, plataforma, desenvolvedor, classificação etária, disponibilidade e preço.

**RF02: Listar Games**

* Os usuários devem poder visualizar a lista de games existentes.
* A lista de games deve mostrar o título e a descrição de cada game.

**RF03: Atualizar Games**

* Os usuários devem poder atualizar o título e a descrição de um game existente.
* A atualização de um game deve ser identificada pelo seu ID exclusivo.

**RF04: Remover Games**

* Os usuários devem poder remover um game da lista.
* A remoção de um game deve ser identificada pelo seu ID exclusivo.

**REQUISITOS NÃO FUNCIONAIS**

**RNF01 – Usabilidade:**

* A interface do usuário deve ser intuitiva, com fácil navegação e compreensão das funcionalidades oferecidas.
* As mensagens de erro devem ser claras e informativas para auxiliar o usuário a entender e corrigir problemas.

**RNF02 – Manutenibilidade:**

* O código do sistema deve seguir boas práticas de programação, facilitando na manutenção futura e incorporação de novos recursos.
* Deve haver uma documentação abrangente que explique a arquitetura, o código e os procedimentos de manutenção.

**RNF03 – Confiabilidade:**

* O sistema deve ser confiável, minimizando falhas e erros inesperados.
* Deve ser implementado um mecanismo de backup e recuperação para garantir a integridade dos dados em caso de falha do sistema.

**RNF04 – Desempenho:**

* O tempo de carregamento das páginas no frontend não deve exceder um limite aceitável, mesmo em condições de carga máxima.
* O sistema deve oferecer boa responsividade ao usuário, garantindo tempos de resposta aceitáveis para todas as operações.

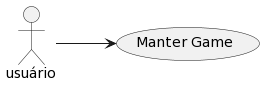
**RNF05 – Arquitetura de Três Camadas:**

* O sistema deve ser implementado seguindo uma arquitetura de três camadas, garantindo a separação clara das camadas de apresentação, lógica de negócios e acesso a dados. Isso proporcionará uma estrutura organizada e facilitará a manutenção e expansão do sistema.

**RNF06 – Tecnologias e Ferramentas:**

* O sistema deve ser desenvolvido para Web utilizando React e tecnologias associadas no Frontend e Java como plataforma e linguagem de programação no Backend.
* O sistema deve ser desenvolvido usando as versões especificas das tecnologias recomendadas, incluindo o SQLite 3.43.0.0, Spring Boot 2.17.15 e React. A equipe de desenvolvimento deve manter essas versões durante todo o ciclo do projeto.
* As tecnologias selecionadas, como o SQLite e o Spring Boot, devem ser capazes de oferecer o desempenho necessário para lidar com a quantidade prevista de dados e transações no sitema.
* As tecnologias selecionadas devem ser compatíveis com as IDEs e ferramentas de desenvolvimento recomendadas, como Eclipse STS, DBeaver, Postman e VSCode para garantir uma integração suave do fluxo de trabalho de desenvolvimento.
* A integração entre o Backend (Spring Boot) e o Frontend (React) deve ser facilitadas pelas tecnologias recomendadas, como REST e JSON, para permitir a transferência eficiente de dados entre as camadas.

**3. ESPECIFICAÇÃO – CASOS DE USO**



**Caso de Uso: Manter Game**

**Ator Principal**: Usuário

**Resumo:** Este caso de uso descreve como o usuário pode criar, listar, atualizar, remover games.

**Pré-condições:**

* O usuário deve estar autenticado no sistema (se a autenticação for um requisito não funcional).
* Existem games cadastrados (para os casos de atualização, remoção e marcação de conclusão).

**Fluxo Básico:**

1. O sistema exibe a lista de games disponíveis, incluindo opção para criar, listar, atualizar, remover e marcar tarefas como concluídas.

**2. Criar Game:**

* O usuário escolhe a opção “Cadastrar Game”.
* O sistema solicita o título e a descrição do novo game.
* O usuário fornece o título e a descrição.
* O sistema cria o novo game e a adiciona à lista de games.

**3. Listar Games:**

* O usuário escolhe a opção “Listar Games”.
* O sistema exibe a lista de games, mostrando o título e a descrição de cada game.

**4. Atualizar Game:**

* O usuário escolhe a opção “Atualizar Game”.
* O sistema solicita o ID do game que deseja atualizar.
* O usuário fornece o ID do game.
* O sistema exibe o game existente e solicita os novos valores para o título e descrição.
* O usuário fornece os novos valores.
* O sistema atualiza o game com os novos valores.

**5. Remover Game:**

* O usuário escolhe a opção “Remover Game”.
* O sistema solicita o ID do game que deseja remover.
* O usuário fornece o ID do game.
* O sistema verifica se o game existe.
* Se o game existe, o sistema a remove da lista de games.

**Pós-condições:**

* As operações de criação, atualização, remoção e marcação de conclusão de tarefas são refletidas na lista de games.
* O sistema exibe mensagens de sucesso ou erro após cada operação.

**Exceções:**

* Se o usuário fornecer informações inválidas durante a criação ou atualização de games, o sistema exibirá uma mensagem de erro.
* Se o usuário tentar atualizar ou remover um game que não existe, o sistema exibirá uma mensagem de erro.
* Se ocorrerem falhas no sistema, o usuário será notificado com uma mensagem de erro genérica.

**Requisitos Associados:**

* RF1: Criar Game
* RF2: Listar Games
* RF3: Atualizar Game
* RF4: Remover Game
* RF5: Concluir Game

**4. ARQUITETURA**

A Arquitetura de Três Camadas é um padrão arquitetural que divide uma aplicação em três camadas distintas: Apresentação, Negócio e Persistência. Cada camada tem uma reponsabilidade especifica na aplicação.

Modelo Geral das Camadas Na Arquitetura de 3 Camadas:

* **Camada de Apresentação:** Responsável pela interação com o usuário, exibição de informações e coleta de entrada do usuário.
* **Camada de Negócio:** Contém a lógica de negócio da aplicação, incluindo regra de negócios e manipulação de dados.
* **Camada de Persistência:** Responsável pelo acesso aos dados e interação com sistemas de armazenamento de dados, como banco de dados.

**5. ANEXO**

* **Código:**

<https://github.com/Almada77/npcpccalgoritmoeprogramacao.git>

* **Vídeo no Youtube:**

<https://www.youtube.com/watch?v=QbhRLQirQDA&ab_channel=VictorAlmada>

* **Estrutura da Tabela**

**Conteúdo do script.sql:**

