FECAP

**PROJETO.....**

**Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas**

São Paulo

2024

**INTEGRANTES DO PROJETO e RA’S**

ANDERSON YAVI FERNANDEZ - 24025678

GABRIEL GONÇALVES PIRES - 24026418

GUSTAVO CUNHA FERNANDES - 24025890

KAIQUE NERES DE OLIVEIRA - 24026134

LUIZ FELIPE GALDINO DE CARVALHO - 24026568

**1 INTRODUÇÃO**

Tendo.......

**2. DOCUMENTO DE ABERTURA DO PROJETOS**

**Prefácio**

Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.

**Introdução**

Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.

**Glossário**

Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.

**Definição de requisitos de usuário**

Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados.

**Arquitetura do sistema**

Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.

**Especificação de requisitos do sistema**

Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.

**Modelos do sistema**

Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.

**Evolução do sistema**

Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.

**Apêndices**

Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados.

**3. REQUISITOS DE SISTEMA**

A requisitos funcionais

1 Cadastro de usuário e instituições.

2 Perfil de usuário gerenciado pela instituição.

3 Ambiente de programação em blocos para usuários.

4 Avaliações;

5 Comunicação e suporte;

6 Área do docente;

7 Relatórios;

8 Mostrar Ranking;

9 Criar Níveis de avanço;

10 Acessibilidade;

11 Responsividade;

------------------------------------------\\--------------------------------------------

B requisitos não-funcionais

1 Segurança de dados - LGPD;

2 Desempenho do Site;

3 Disponibilidade do Serviço;

4 Capacidade de sustentar diversos usuários;

5 usabilidades

6 tolerâncias a falhas;

**3.1 REQUISITOS FUNCIONAIS DE SOFTWARE**

Necessários 6 requisitos

|  |  |
| --- | --- |
| **RFS01** | |
| **Função** |  |
| **Descrição** |  |
| **Entradas** |  |
| **Fonte** |  |
| **Saídas** |  |
| **Ação** |  |
|  |  |
| **RFS02** | |
| **Função** |  |
| **Descrição** |  |
| **Entradas** |  |
| **Fonte** |  |
| **Saídas** |  |
| **Ação** |  |

**3.2 REQUISITOS NÃO FUNCIONAIS DE SOFTWARE**

Necessários 6 requisitos

|  |  |
| --- | --- |
| **RFS01** | |
| **Função** |  |
| **Descrição** |  |
| **Entradas** |  |
| **Fonte** |  |
| **Saídas** |  |
| **Ação** |  |
|  |  |
| **RFS02** | |
| **Função** |  |
| **Descrição** |  |
| **Entradas** |  |
| **Fonte** |  |
| **Saídas** |  |
| **Ação** |  |

**4. CASOS DE USO**

1. Criação de Jogos Interativos

Descrição: As crianças podem usar blocos de programação para criar seus próprios jogos simples. Funcionalidade: O site oferece uma interface intuitiva onde os usuários arrastam e soltam blocos para definir ações, como mover personagens, coletar itens ou definir regras de vitória. Benefício: Isso promove a criatividade e a resolução de problemas, além de ajudar as crianças a entenderem a lógica por trás da programação de forma divertida.

2. Desafios de Programação

Descrição: O site apresenta desafios diários ou semanais que incentivam as crianças a resolver problemas de lógica usando blocos de programação. Funcionalidade: Cada desafio tem um nível de dificuldade crescente, com dicas e tutoriais disponíveis. As crianças podem ganhar pontos ou medalhas conforme completam os desafios. Benefício: Isso mantém as crianças engajadas, desenvolvendo habilidades de raciocínio lógico e pensamento crítico ao mesmo tempo.

3. Aprendizado Colaborativo

Descrição: As crianças podem trabalhar em projetos em grupos, promovendo o aprendizado colaborativo. Funcionalidade: O site permite que os usuários formem equipes e compartilhem seus projetos. Eles podem comentar, dar feedback e ajudar uns aos outros na resolução de problemas. Benefício: Isso incentiva o trabalho em equipe, a comunicação e a socialização, enquanto ensina conceitos de programação de uma maneira mais interativa e coletiva.

**5. ARQUITETURA DO SISTEMA**

1. Camada de Apresentação (Front-end)

Tecnologias: HTML, CSS, JavaScript, framework React.JS

Componentes:

Interface do Usuário: Páginas para login, criação de projetos, visualização de desafios e comentários.

Editor de Blocos: Ferramenta visual para arrastar e soltar blocos de programação.

Dashboard: Área onde os usuários podem ver seu progresso, jogos criados e desafios completados.

2. Camada de Lógica de Negócio (Back-end)

Tecnologias: Node.js.  
Componentes:

API RESTful: Para gerenciar requisições do front-end, como criar projetos, acessar desafios e gerenciar usuários.

Controle de Autenticação: Gerenciamento de contas de usuários, autenticação e autorização.

Gerenciamento de Desafios e Projetos: Lógica para criar, editar e excluir desafios e projetos, além de armazenar progresso.

3. Camada de Banco de Dados

MySQL

Estrutura:

Usuários: Armazena informações dos usuários, como nome, e-mail, e progresso em projetos.

Projetos: Armazena dados dos projetos criados, incluindo blocos de código e feedback.

Desafios: Armazena informações sobre os desafios, incluindo descrição e nível de dificuldade.

Feedback: Armazena comentários e avaliações dos projetos.

4. Camada de Integração

APIs Externas: Para recursos adicionais, como tutoriais em vídeo, materiais de leitura ou integração com plataformas educacionais.

Serviços de Notificação: Para enviar atualizações e lembretes aos usuários sobre novos desafios ou feedback em seus projetos.

5. Infraestrutura

**6. REFERÊNCIAS BIBLIOGRÁFICAS**

SOMMERVILLE, I. **Engenharia de Software.** 11ª Edição. São Paulo: Pearson Addison-Wesley, 2017.