**Documento de Requisitos**

**SQUAD 3 - POKÉDEX**

**Versão 1.0**



Arthur Bernardino

Gabriel Ribeiro

Guilherme Régis

Gustavo César

José Ricardo

Marcelo Cláudio

Maria Gabriella

Mateus Rodrigues

**SUMÁRIO**

[**1.Introdução**](#_oh118gklz32) **3**

[1.1 Visão geral deste documento](#_xba6c0i7lx77) 3

[1.2 Convenções, termos e abreviações](#_5t9zzcw257sg) 3

[1.2.1 Identificação dos Requisitos](#_4a57cpr93zmu) 3

[1.2.2 Prioridades dos Requisitos](#_nfaarhwun8wq) 3

[**2 Descrição geral do sistema**](#_k9ydxnqokj0e) **4**

[2.1 Abrangência e sistemas relacionados](#_t3obbsq2o2z) 4

[2.2 Descrição dos usuários](#_fdo76sao0wpf) 4

[**3.Estudo de Viabilidade do Sistema Proposto**](#_iebkrznzpmnl) **4**

[3.1 Viabilidade Operacional](#_99b4pqmfv9uy) 5

[3.2 Viabilidade Técnica](#_54vxrhe7qxkj) 5

[3.3 Viabilidade de Tempo](#_r16fvn2wmy4y) 6

[3.4 Viabilidade Econômica](#_ug1y6l2733pa) 7

[**4. Requisitos Funcionais**](#_f9ow53t3f416)8

[4.1 Implementar Lazy Loading](#_al61hhhnz3lg) 9

[4.2 Implementar Pesquisa](#_yjxthjowfx0e) 9

[4.3 Implementar paginação](#_5zb66jp0m341) 11

[4.4 Implementar cadastro de novos Pokémons](#_l8ij9cyjufu7) 11

[**5. Requisitos não funcionais**](#_li5b9x96scwv) **13**

[5.1 Usabilidade](#_ji6e4kd1rgd) 13

[5.2 Desempenho](#_kytr3cd7vo7c) 13

[5.3 Desenvolvimento](#_40b2x9mnj18d) 14

[**6. Conclusão**](#_fki0ip8uft4a)15

[**7.Referências**](#_hgyjcd9j6gxh)16

# 1.Introdução

O presente documento descreve o desenvolvimento de uma pokédex que permite realizar consultas e cadastros de Pokémons, exibindo suas características, para a empresa SIDI, a qual é um dos maiores institutos de ciência e tecnologia do Brasil, presente em Manaus, Recife e Campinas. Há quase 20 anos no mercado, eles desenvolvem soluções de software utilizando tecnologias inovadoras para transformar o dia a dia de pessoas e organizações.

## 1.1 Visão geral deste documento

Esta introdução fornece as informações necessárias para fazer uma aplicação visual de boa usabilidade que permite realizar consultas e cadastros de Pokémons, com a exibição de suas imagens e características.

## 1.2 Convenções, termos e abreviações

### 1.2.1 Identificação dos Requisitos

- [Login RF001];

- [Cadastro de Usuário.RF002];

- [Gerar Casos de Uso.RF006];

- [Usabilidade.NF001];

- [Usabilidade.NF002].

### 1.2.2 Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações **“essencial”**, **“importante**” e **“desejável”**.

**Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.

**Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.

**Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis sãorequisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

# 2 Descrição geral do sistema

O projeto objetiva o desenvolvimento de uma aplicação visual de uma pokédex que será responsável por cadastrar, e realizar buscas por meio de nome, número ou tipo/elemento de um Pokémon, que será exibido por páginas com no máximo 10 registros por vez, utilizando o padrão MVVM(Model-View-Viewmodel), fazendo uso de banco de dados SQLite e implementando o lazy loading.

## 2.1 Abrangência e sistemas relacionados

O sistema irá fazer o reconhecimento do Pokémon de acordo com a pesquisa que o usuário deverá fazer, será descrito o seu tipo, as suas vantagens e desvantagens, o número de acordo com a Pokédex. Esse sistema não atenderá as necessidades de quem está em busca de criar ou interagir com esses Pokémons, eles não serão pets digitais. Utilizaremos como principal as linguagens C# e .NET, além dos repositórios Git e GitHub.

## 2.2 Descrição dos usuários

Os usuários serão jovens Gamers e Fãs do universo Pokémon que necessitam de uma ferramenta para visualizar e cadastrar informações de pokémons quando estão jogando os jogos ou querendo consultar algum dado dos mesmos.

# 

# 3.Estudo de Viabilidade do Sistema Proposto

Descrição de todas as opções consideradas para análise do sistema que está sendo proposto.

## 3.1 Viabilidade Operacional

O estudo da Viabilidade Operacional baseou-se na estrutura PIECES, que promove a análise de Performance (desempenho), Informação, Economia, Controle, Eficiência e Serviços, da seguinte forma:

|  |  |
| --- | --- |
| **Performance** | **A aplicação contará com um bom tempo de resposta e taxa de transferência das informações** |
| **Informação** | **A aplicação deverá fornecer uma boa fluidez nos inputs e outputs, deverá armazenar os dados corretamente e também fornecer os dados corretamente e de forma organizada** |
| **Economia** | **O desenvolvimento da aplicação não acarretará custos, pois as ferramentas utilizadas serão gratuitas.** |
| **Controle** | **A aplicação deverá fornecer segurança das informações, e um bom controle para os casos de desvio de padrão dessas informações.** |
| **Eficiência** | **O esforço e os materiais necessários para a realização das tarefas é intermediário.** |
| **Serviços** | **O sistema produz resultados precisos, consistentes e confiáveis. O sistema é fácil de aprender e usar. A Aplicação será compatível com todos os navegadores web.** |

## 3.2 Viabilidade Técnica

Será utilizada a linguagem de programação C# e .NET. Com ela podemos escrever e criar aplicações desktop, web, back end e mobile. Com C# e a plataforma .NET temos a disposição ferramentas que podem ser usadas por uma grande quantidade de atividades e objetivos. O ambiente de desenvolvimento será o visual studio community. Como base vamos usar uma API pública chamada PokéApi auxiliando no desenvolvimento. Banco de dados vamos usar o DB Browser for SQLite um banco de dados simples e de código aberto com recursos de um banco de dados relacional, com sintaxe SQL. Iremos utilizar também o git e github para gerenciamento do código fonte da aplicação. Usando Postman facilitando no desenvolvimento para criar, compartilhar, documentar e testar APIs.

## 3.3 Viabilidade de Tempo

O fator tempo é o mais crítico no cenário do projeto em pauta. Influenciam este fator as atividades paralelas dos membros do projeto, bem como a dificuldade de se estimar o tempo necessário do projeto e o investido por cada membro deste. Nessa seção é detalhado o cronograma previsto para a realização do projeto. Foram levadas em consideração as etapas do processo de desenvolvimento de software para montar esse cronograma.

|  |  |  |  |
| --- | --- | --- | --- |
| Etapas | Atividades | Quantidade em  dias | Quantidade em semanas |
| Especificação | Estudo de Viabilidade  Elicitação e Análise de requisitos  Especificação de Requisitos  Validação de Requisitos | 2  7  2  4 | 0  1  0  0 |
| Projeto  e Implementação | Projeto de Interface | 8 | 1 |
| Entrega  e Evolução |  |  |  |

## 3.4 Viabilidade Econômica

Estudar a viabilidade econômica do projeto tem como objetivo principal avaliar se o desenvolvimento do projeto acarretará em algum benefício econômico para o cliente. Se sim, identificar quais.

**Custos envolvidos:**

1. Salário dos desenvolvedores do sistema;

2. Aquisição de equipamentos;

3. Hospedagem do sistema;

4. Utilização de equipamentos durante o desenvolvimento do projeto;

5. Custos com energia, água e telefone;

**Gastos previstos:**

**Custos com Infraestrutura e mão de obra especializada**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantidade** | **Valor Unitário** | **Valor Total** |
| Desenvolvedor BackEnd | 2 | 0,00 | 0,00 |
| Analista de Sistemas | 1 | 0,00 | 0,00 |
| Testador | 1 | 0,00 | 0,00 |
| Gerente de Projeto | 2 | 0,00 | 0,00 |
| Desenvolvedor FrontEnd | 2 | 0,00 | 0,00 |
| **TOTAL** | **8** | **0,00** | **0,00** |

**Custos com Software e Programas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantidade** | **Valor Unitário** | **Valor Total** |
| Visual Studio Community | 8 | 0,00 | 0,00 |
| DB Browser for SQLite | 8 | 0,00 | 0,00 |
| Postman | 8 | 0,00 | 0,00 |
| Windows Community Toolkit Sample App | 8 | 0,00 | 0,00 |

**Custos de Configuração e Instalação**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantidade** | **Valor Unitário** | **Valor Total** |
| Hospedagem | 01 | 0,00 | 0,00 |

**Custo Total da Aplicação**

|  |  |
| --- | --- |
| **Item** | **Valor** |
| Custos com Infraestrutura e mão de obra especializada. | 0,00 |
| Custos com Software e Programas | 0,00 |
| Custos de Configuração e Instalação | 0,00 |
| **TOTAL** | **0,00** |

# 

# 

# 4. Requisitos Funcionais

De acordo com Sommerville (2007, pág. 80) requisitos funcionais são as declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações. Em alguns casos os requisitos funcionais podem também estabelecer explicitamente o que o sistema não deve fazer. A seguir são especificados os requisitos não funcionais identificados.

## 4.1 Implementar Lazy Loading

**[RF 0001] <Lazy Loading>**

Requisito primordial para busca no banco de dados.

**Ator:** Desenvolvedor

**Prioridade:** Importante

**Interface(s) associada(s)**: Otimização.

**Entradas e pré-condições:** Busca no banco de dados.

**Saídas e pós-condições:** Exibir a informação para o usuário.

**Descrições dos casos de uso:** As buscas são feitas no banco de dados, se o dado não existir, deve ser feita uma requisição ao PokéAPI para coletar os dados e então persistir os dados retornados no banco antes de exibir a informação para o usuário.

**Fluxo principal:**

1 - O usuário faz a busca por pokémons.

2 - O sistema verifica se existe no banco de dados;

3 - O sistema retorna mostrando na tela as informações.

**Fluxo secundário:**

Se o dado não existir, deve ser feita uma requisição ao PokéAPI para coletar os dados.

**Entradas e Pré-condições:** O usuário deverá realizar uma busca. O usuário deverá ter feito uma pesquisa.

**Saídas e pós-condições:** Após sua ativação o sistema vai otimizar o desempenho da página.

**Descrições dos casos de uso:** O usuário deverá realizar uma busca, logo em seguida o sistema irá ativar a funcionalidade e otimizar o desempenho da página.

## 4.2 Implementar Pesquisa

**[RF0002] <Busca Por Nome>**

Este requisito é importante, pois é através dele que o usuário pode acessar o registro do item específico a partir da informação do nome.

**Ator:** Desenvolvedor

**Prioridade:** Essencial

**Fluxo Principal:**

1 - O usuário acessa a aplicação

2 - O usuário clica no botão Pesquisar

3 - O usuário preenche os dados necessários para a pesquisa

4 - O usuário clica no botão de pesquisar para executar o comando.

**Fluxo Secundário:** Se o valor inserido for inválido ou não corresponder a um dado já existente, o sistema exibirá um alerta informando o acontecido e como executar a busca corretamente.

**Interface(s) associada(s):** Exibição dos dados requisitados

**Entradas e Pré-condições:** O usuário deverá inserir um nome válido. O usuário deverá preencher todos os dados obrigatórios solicitados pelo sistema.

**Saídas e pós-condições:** Após realizada a pesquisa o sistema exibe os dados requisitados de forma ordenada.

**Descrições dos casos de uso:** O usuário deverá clicar na barra de pesquisa e digitar o termo válido desejado, o sistema então irá exibir as informações requisitadas.

**[RF0003] <Busca por número do Pokémon>**

Este requisito é importante, pois é através dele que o usuário pode acessar o registro do pokémon que pertence a um determinado número, por exemplo: 1, 15, 100 e etc.

**Ator:** Desenvolvedor

**Prioridade:** Essencial

**Fluxo Principal:**

1 - O usuário acessa a aplicação

2 - O usuário clica no botão Pesquisar

3 - O usuário preenche os dados necessários para a pesquisa

4 - O usuário clica no botão de pesquisar para executar o comando..

**Fluxo Secundário:** Se o valor inserido for inválido ou não corresponder a um dado já existente, o sistema exibirá um alerta informando o acontecido e como executar a busca corretamente.

**Interface(s) associada(s):** Exibição dos dados requisitados

**Entradas e Pré-condições:**Sem pré-condições. O usuário deverá preencher todos os dados obrigatórios solicitados pelo sistema.

**Saídas e pós-condições:** Após realizada a pesquisa o sistema exibe os dados requisitados de forma ordenada.

**Descrições dos casos de uso:** O usuário deverá clicar na barra de pesquisa e digitar o termo válido desejado, o sistema então irá exibir as informações requisitadas.

**[RF0004] <Busca Por Tipo>**

Este requisito é importante, pois é através dele que o usuário pode acessar à todos os pokémons que pertencem a um determinado tipo ou elemento, por exemplo: fogo, normal, solo, etc.

**Ator:** Desenvolvedor

**Prioridade:** Essencial

**Fluxo Principal:**

1 - O usuário acessa a aplicação

2 - O usuário clica no botão Pesquisar

3 - O usuário preenche os dados necessários para a pesquisa

4 - O usuário clica no botão de pesquisar para executar o comando.

**Fluxo Secundário:** Se o valor inserido for inválido ou não corresponder a um dado já existente, o sistema exibirá um alerta informando o acontecido e como executar a busca corretamente.

**Interface(s) associada(s):** Exibição dos dados requisitados

**Entradas e Pré-condições:**Sem pré-condições. O usuário deverá preencher todos os dados obrigatórios solicitados pelo sistema.

**Saídas e pós-condições:** Após realizada a pesquisa o sistema exibe os dados requisitados de forma ordenada.

**Descrições dos casos de uso:** O usuário deverá clicar na barra de pesquisa e digitar o termo válido desejado, o sistema então irá exibir as informações requisitadas.

## 4.3 Implementar paginação

**[RF0005] <Implementar paginação>**

Requisito responsável por numerar a quantidade de páginas e conteúdos para assim facilitar a visualização do sistema.

**Ator:** Desenvolvedor

**Prioridade:** Importante

**Fluxo Principal:**

1 - Usuário acessa o sistema.

2 - Usuário visualiza a lista de Pokémons.

3 - O usuário clica no botão para visualizar a próxima página.

**Fluxo Secundário:** No Fluxo secundário, se o botão de passar a página não funcionar, o sistema exibirá uma caixa de alerta e atualizará a página.

**Interface(s) associada(s):** usabilidade, facilidade de visualização.

**Entradas e Pré-condições:** Sem pré-condições.

**Saídas e pós-condições:** Após realizar a pesquisa o sistema agrupa os pokémons por páginas.

**Descrições dos casos de uso:** O sistema deverá exibir as informações requisitadas pela pesquisa em páginas, contendo no máximo 10 itens por página.

## 4.4 Implementar cadastro de novos Pokémons

**[RF0006]** **<Cadastro de pokémons>**

Requisito responsável por cadastrar novos pokémons e suas respectivas informações.

**Ator:** Desenvolvedor

**Prioridade:** Importante

**Interface(s) associada(s):** Segurança, Disponibilidade e Escalabilidade.

**Entradas e Pré-condições:** Sem pré-condições.

**Saídas e pós-condições:** Após realizar o cadastro o sistema exibirá o cadastro realizado com sucesso e o novo pokémon será adicionado na biblioteca da pokédex.

**Descrições dos casos de uso:** O sistema deverá exibir as informações requisitadas pela pesquisa em páginas, contendo no máximo 10 itens por página.

**Fluxo Principal:**

1 - Usuário acessa o sistema.

2 - Usuário visualiza a lista de Pokémons.

3 - O usuário clica no botão para visualizar a próxima página.

**Fluxo Secundário:** No Fluxo secundário, se o botão de passar a página não funcionar, o sistema exibirá uma caixa de alerta e atualizará a página.

# 5. Requisitos não funcionais

Segundo Sommerville (2011, pág.80) requisitos não funcionais são restrições sobre os serviços ou as funções oferecidas pelo sistema. Eles incluem restrições de timing, restrições sobre o processo de desenvolvimento e padrões. Os requisitos não funcionais aplicam-se frequentemente ao sistema como um todo. Em geral, eles não se aplicam às características ou serviços individuais de sistema. A seguir são especificados os requisitos não funcionais identificados:

## 5.1 Usabilidade

**[RNF001] <Uso de Design responsivo nas interfaces gráficas>**

O sistema de Atendimento a Clientes será construído para rodar em ambiente web. Deverá possuir um design responsivo. A interface do sistema deverá se comportar adequadamente independente do front-end que será utilizado para acesso – Browser, Smartphone ou Tablet.

**Prioridade:** Desejável.

## 5.2 Desempenho

**[RNF002] <Fazer uso de persistência de dados>**

Requisito responsável por garantir que os dados pesquisados recentemente fiquem armazenados para posteriores consultas, através do banco de dados SQLite, diminuindo assim o tempo de espera de carregamento do dado.

**Prioridade:** Importante.

**[RNF003]< Utilizar o padrão MVVM (Model-View-ViewModel)>**

Esse padrão reduz as dependências rígidas entre diferentes tipos de código, facilitando a alteração de unidades de código individuais sem causar efeitos colaterais indesejados em outras unidades. Simplificando os testes de unidades e aumentando a facilidade de manutenção.

**Prioridade:** Essencial.

## 5.3 Desenvolvimento

**[RNF004] <Utilizar a linguagem C# e .NET>**

Para o desenvolvimento da aplicação será utilizado C# é uma linguagem de programação orientada a objetos criada pela Microsoft, faz parte da sua plataforma e .NET, que é uma plataforma de desenvolvimento de software livre gratuita para a criação de vários tipos de aplicativos.

**Prioridade:** Essencial.

**[RNF005] <Utilizar o Visual Studio community>**

Para o desenvolvimento da aplicação será necessário o uso da IDE Visual Studio community, que é gratuita, completa e extensível para a criação de aplicativos modernos para Android, iOS e Windows, bem como aplicativos Web e serviços de nuvem. Ela também permite a colaboração de forma totalmente integrada.

**Prioridade:** Desejável.

# 6. Conclusão

Neste presente documento foi possível realizar a análise de requisitos, definindo assim o que o sistema deve fazer e sob quais restrições ele irá funcionar para atender os clientes, stakeholders e usuários, a partir da definição dos requisitos funcionais, que são as funcionalidades e requisitos não funcionais, chamados de restrições, relacionadas a portabilidade, usabilidade, desempenho, padrões, dentre outros. Foi possível também analisar os processos envolvidos para o desenvolvimento de um sistema de uma pokédex, como suas viabilidades econômica, operacional, técnica e de tempo. É importante lembrar que todos os requisitos, após sua especificação, devem ser validados e verificados, para que estejam corretos, precisos, consistentes e verificáveis, e por fim sejam priorizados.

# 7.Referências

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de Requisitos. Software Orientado ao Negócio.** 1ª edição. Rio de Janeiro: Brasport, 2016.

MACHADO, Felipe Nery Rodrigues. **Análise e Gestão de Requisitos de Software.** 3ª edição. São Paulo: Saraiva, 2016

**DevMedia:**<<https://www.devmedia.com.br/introducao-a-engenharia-de-requisitos/29454>>. Acesso em 09 de abril. 2022.

**Monitoratec:**<<https://www.monitoratec.com.br/blog/servico-de-engenharia-de-requisitos/>>. Acesso em 09 de abril. 2022.

**BlogBetrybe:**<<https://blog.betrybe.com/tecnologia/engenharia-de-requisitos-tudo-sobre/>>. Acesso em 11 de abril. 2022.

**Alura:**<<https://www.alura.com.br/conteudo/engenharia-requisitos>>. Acesso em 11 de abril. 2022.

**Unesp:**<<https://www.dcce.ibilce.unesp.br/~ines/cursos/eng_soft/aula05.pdf>>.

Acesso em 11 de abril. 2022.

**IBQTS:**<<http://www.ibqts.com.br/>>. Acesso em 11 de abril. 2022.

**Abramti:**<<https://www.abramti.org.br/b9/sobre-cpre-fl>>. Acesso em 11 de abril. 2022.