

**ENGENHARIA DE REQUISITOS E ANÁLISE DE SISTEMAS**

**DOCUMENTAÇÃO DO SISTEMA DE CADASTRO  
DE BIBLIOTECA DIGITAL - FASE 1 e 2**

**ALUNOS:**

**CINTIA MENDES BERNARDO**

**JHEAN MONTEIRO DA SILVA**

**LORENZO SAMPAIO DE GUADELUPE**

**TURMA A**

**PROF. GIOLIANO BERTONI**

**SAQUAREMA**

**2025**



## Trabalho de projeto pedido pelo professor:

O sistema deve utilizar obrigatoriamente as seguintes tecnologias

- **Frontend:** HTML5, CSS3, JavaScript
- **Backend:** Python 3 + Flask
- **Banco de Dados:** SQLite3

Inicialmente estão disponíveis os arquivos encontrados na pasta CADASTRO WEB

Atividade

### 1. FASE 1 – DOCUMENTAÇÃO + FRONTEND

- Fazer o protótipo funcionar ✓
  - Corrigindo os prováveis erros ✓
- Propor melhorias ao projeto de Cadastro de Usuários Incluindo mais campos necessários ✓
- Realizar a documentação do sistema ✓
  - Requisitos Funcionais e Não Funcionais ✓
  - Diagrama Entidade Relacionamento ✓
  - Diagrama de Caso de Uso ✓
- Apresentação (06/11) + relatório ✓

### 2. FASE 2 – BACKEND + BD

- Fazer uma pesquisa sobre o FLASK e o SQLite ✓
- Fazer funcionar o sistema integrando o Backend ao BD ✓
- Apresentação (13/11) + relatório ✓

## OBSERVAÇÃO

O objetivo é tentar inicialmente rodar a aplicação localmente. Mas caso tenha problemas relacionados a instalação no laboratório, como alternativa:

- <https://render.com/>
- <https://www.pythonanywhere.com/>

<b>1. Introdução.....</b>	<b>4</b>
1.1. Correção do Protótipo.....	4
1.2. Melhorias Propostas.....	4
1.3. Documentação.....	4
<b>2. Correção do Protótipo.....</b>	<b>4</b>
2.1. Erros encontrados.....	4
<b>3. Melhorias (Campos e Funcionalidades).....</b>	<b>5</b>
3.1. Campos Atuais:.....	5
3.2. Melhorias implementadas:.....	5
<b>4.Documentação do Sistema.....</b>	<b>6</b>
4.1. Requisitos Funcionais (RF).....	6
4.2. Requisitos Não Funcionais (RNF).....	6
4.3. Diagrama Entidade Relacionamento.....	7
4.4. Diagrama de Caso de Uso.....	8
<b>5. Conclusão da 1º Fase.....</b>	<b>8</b>
5.1. Fase 1 foi concluída com 100% de sucesso.....	8
5.2. Próximos passos (13/11):.....	8
<b>6. Pesquisa sobre Flask e o SQLite.....</b>	<b>9</b>
6.1. Flask.....	9
6.2. SQLite3.....	9
<b>7. Conclusão da 2º Fase.....</b>	<b>10</b>
7.1. Fase 2 foi concluída com 100% de sucesso.....	10
<b>8. ANEXOS.....</b>	<b>10</b>

## 1. Introdução

Este relatório apresenta os resultados da Fase 1 e 2 do projeto de desenvolvimento de um sistema de cadastro de Biblioteca Digital, utilizando **HTML5**, **CSS3** e **JavaScript** no frontend, conforme exigido. A Fase 1 teve como objetivos:

### 1.1. Correção do Protótipo

Encontrar erros.  
Corrigir códigos.  
Implementar funcionalidades.

### 1.2. Melhorias Propostas

Novos campos.  
Validações adicionais.  
UX/UI.

### 1.3. Documentação

Requisitos Funcionais.  
Requisitos Não Funcionais.  
DER (Imagem).  
Diagrama de Caso de Uso (Imagem).

## 2. Correção do Protótipo

### 2.1. Erros encontrados

Arquivo	Erro	Correção Aplicada
cadastro.html	Tag <input> escrita errada.	Corrigido para <input>.
cadastro.html	href="stile.css" escrito errado.	Corrigido para Href="style.css".

cadastro.html	Botão de submit ausente.	Adicionado <button type="submit">.
cadastro.html	Sem validação de CPF.	Integrada função ValidarCPF().
cadastro.html	Apenas 2 campos Nome e E-mail.	Adicionados RG, CPF, Telefone, Senha.

**Resultado:** Protótipo 100% funcional.

### 3. Melhorias (Campos e Funcionalidades)

#### 3.1. Campos Atuais:

- Nome e Email.

#### 3.2. Melhorias implementadas:

CAMPO	JUSTIFICATIVA
Nome	(Max. 50 caracteres).
RG	Documento oficial (00.000.000-0).
CPF	Melhora a usabilidade (000.000.000-00) com validação real.
Telefone	Número para contato ((00) 00000-0000).
Email	Email para contato (cadastro@gmail.com)
Confirmação de Senha	(Min. 6 caracteres, oculta).

Funcionalidades	Descrição
Enter passa de campo	Aperte enter e pula pro próximo campo.
Máscaras em tempo real	RG, CPF e telefone formatam enquanto digita.
Toast flutuante	Avisa quando atingir 50 caracteres no nome.
Feedback visual	Mensagem verde (sucesso) ou vermelha (erro).
Limpeza automática	Após sucesso, limpa todos os campos.

UX profissional

Cores contrastantes, sombras, hover, foco brilhante.

## 4.Documentação do Sistema

### 4.1. Requisitos Funcionais (RF)

ID	Descrição
RF001	O usuário deve preencher um formulário de cadastro (Nome, RG, CPF, Telefone, Email e Senha) .
RF002	O sistema deve validar todos os campos obrigatórios.
RF003	O sistema deve validar o CPF com o algoritmo oficial.
RF004	O sistema deve exibir feedback visual de sucesso ou erro.
RF005	O sistema deve limpar o formulário após cadastro bem-sucedido.
RF006	O sistema deve impedir cadastro com dados inválidos.

### 4.2. Requisitos Não Funcionais (RNF)

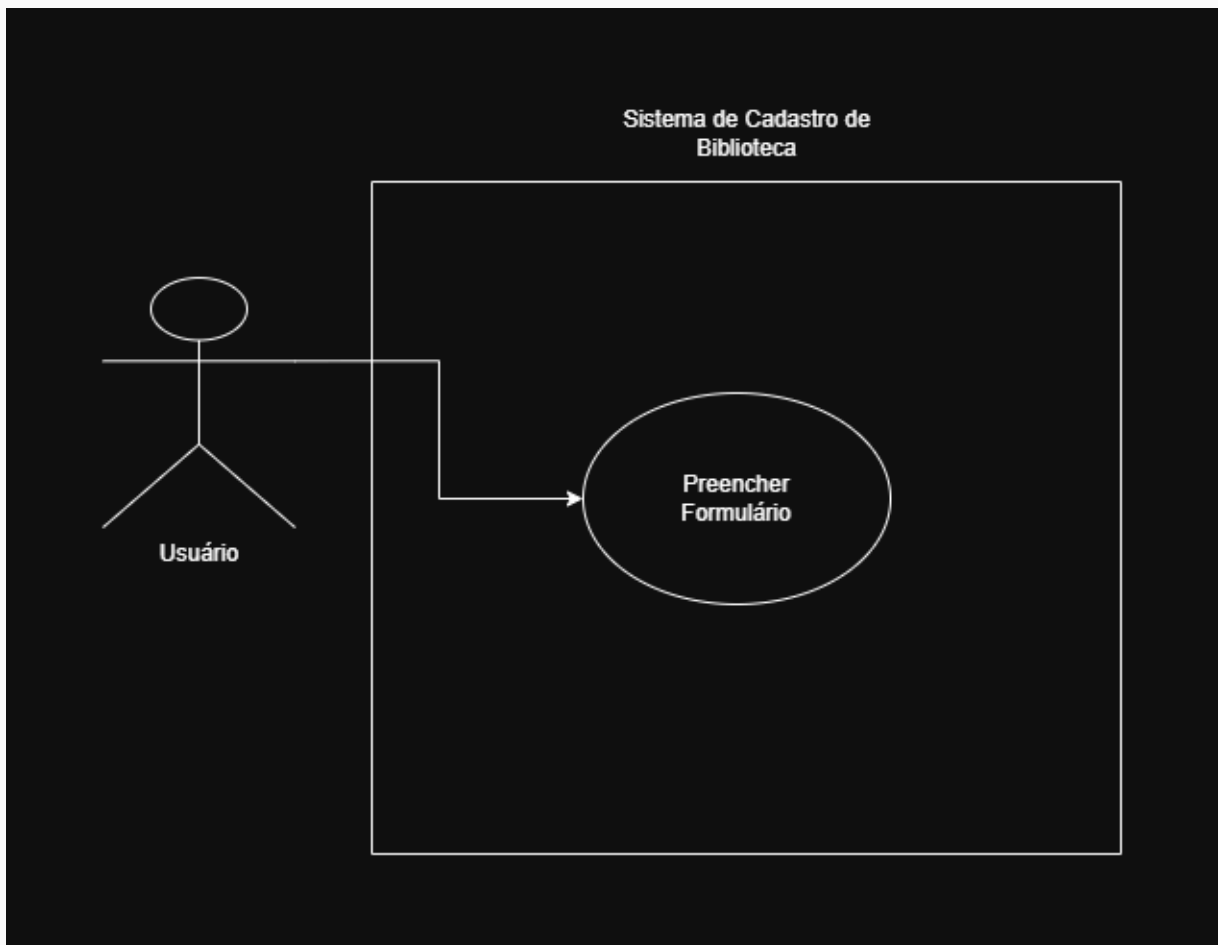
ID	Descrição
RNF001	Interface em Português BR.
RNF002	Responsivo (mobile-friendly). <b>Ajustar no código css</b>
RNF003	Validação no lado do cliente (JS).
RNF004	O sistema deve aplicar máscaras automáticas em RG, CPF e Telefone.
RNF005	O sistema deve permitir navegação por Enter.
RNF006	Segurança (senhas ocultas).
RNF007	Código limpo, comentado e organizado.
RNF008	Sem dependências externas.
RNF009	Cores de alto contraste.
RNF010	Linguagens utilizadas no Frontend: HTML5, CSS3 e JavaScript(Linguagem de marcação).
RNF011	Linguagens utilizadas no Backend: Python3 e Flask.

### 4.3. Diagrama Entidade Relacionamento

Usuário	
PK	<u>Usuário_id</u> int NOT NULL
	nome_usuario char(150) NOT NULL
	rg_usuario char(11) NOT NULL
	cpf_usuario char(11) NOT NULL
	telefone_usuario char(11) NOT NULL
	email_usuario char(50) NOT NULL
	senha_usuario number(6) NOT NULL
	datacadastro_usuario (8) NOT NULL



#### 4.4. Diagrama de Caso de Uso



### 5. Conclusão da 1º Fase

#### 5.1. Fase 1 foi concluída com 100% de sucesso

Protótipo 100% Funcional.  
Validação de CPF integrada.  
Layout moderno e responsivo. **Ajustar não estar funcionando**  
Documentação completa.  
Pronto para integração com **Flask** + **SQLite3** na fase 2.

#### 5.2. Próximos passos (13/11):

Criar backend com Flask  
Integrar com banco SQLite3  
Salvar dados permanentemente



## 6. Pesquisa sobre Flask e o SQLite

### 6.1. Flask

Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível.

Linguagem de programação: [Python](#)

Lançamento: 1 de abril de 2010 (15 anos)

Licença: BSD

Página oficial: [flask.pocoo.org](http://flask.pocoo.org)

Versão estável: 2.0.1 ( 21 de maio de 2021; há 4 anos)

Fonte: [Wikipédia](#)

Resumindo o Flask é um framework do Python, ele é um conjunto de ferramentas, que serve para criar sites e Apis no Python, ele é o mais simples, muito flexível, usado por muitas empresas e permite fazer tantos os sites simples, como ir os tornando mais complexo e robusto.

### 6.2. SQLite3

SQLite é uma biblioteca em linguagem C que implementa uma base de dados SQL embutida. Programas que usem a biblioteca SQLite podem ter acesso ao banco de dados SQL sem executar um processo SGBD separado.

Desenvolvedor: [D. Richard Hipp](#)

Linguagem de programação: [C](#)

Lançamento inicial: 17 de agosto de 2000; há 25 anos

Licença: Domínio público

Repositório: [sqlite.org/src](https://sqlite.org/src)

Versão estável: 3.51.0 (4 de novembro de 2025; há 9 dias)

Site: [sqlite.org](https://sqlite.org)

Fonte: [Wikipédia](#)

Resumindo: SQLite3 é uma biblioteca em linguagem C que fornece um sistema de gerenciamento de bancos de dados relacional (SGBDR), um software que organiza dados em tabelas com linhas e colunas, permitindo

que os usuários criem, gerenciem e consultem informações de forma eficiente. Muito leve (“lite”) em termos de configuração, administração do banco de dados e recursos necessários. Presente em todos os smartphones e na maioria dos computadores e outros dispositivos. Software livre e de código aberto.

## 7. Conclusão da 2º Fase

### 7.1. Fase 2 foi concluída com 100% de sucesso

Pesquisa sobre o Flask e SQLite3 com uma breve explicação.  
O sistema integra Backend com Flask ao BD.  
Salvando dados permanentemente.  
Documentação completa.

## 8. ANEXOS

1. `index.html` (código completo)
2. `style.css` (estilos)
3. `script.js` (validações e máscaras)
4. Prints do sistema funcionando
5. Arquivos .drawio do DER e Caso de Uso
6. Conclusão da Fase 1
7. Pesquisa sobre o Flask e SQLite3
8. Sistema conforme o pedido
9. Conclusão Fase 2

**SAQUAREMA, 05 de novembro de 2025**

---

**Cintia Mendes Bernardo**

---

**Jhean Monteiro da Silva**

---

**Lorenzo Sampaio de Guadalupe**