**UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN**
Pró-Reitoria de Ensino e Graduação - PROEG
Unidade Universitária: Campus de Natal – CaN
Ciência da Computação

# 📝 Documento de Planejamento – AT2

## 1. Nome e descrição do projeto

**Nome:** Gerenciador de Mangás
**Descrição:**
Este projeto é uma aplicação web que simula o gerenciamento de mangás em uma livraria. A aplicação foi desenvolvida utilizando o framework Django para o back-end e HTML + JavaScript puro no front-end.

Nesta segunda etapa (AT2), a aplicação foi evoluída para implementar um **CRUD completo com persistência em um arquivo JSON local**. Diferente da AT1, os dados agora são **salvos em disco**, permitindo que sejam mantidos mesmo após o encerramento do servidor. Foi adicionada também a funcionalidade de atualização (`PUT`), completando o ciclo CRUD.

---

## 2. Tema e entidade escolhida

**Tema:** Catálogo de produtos de uma livraria
**Entidade:** `Manga` — representa um mangá com seus atributos principais: título, autor, quantidade de volumes e preço. O campo `status`, presente na AT1, deixou de ser utilizado na interface para simplificar a modelagem de dados.

---

## 3. Estrutura dos dados (JSON)

A estrutura de cada objeto da entidade `Manga` é representada em formato JSON da seguinte maneira (exemplo):

```
{
  "id": 1,
  "titulo": "One Piece",
  "autor": "Eiichiro Oda",
  "volumes": 105,
  "preco": 39.90
```

```
}
```

**Explicação dos campos:**

- `id`: identificador único gerado automaticamente
- `titulo`: nome do mangá
- `autor`: criador da obra
- `volumes`: número total de volumes disponíveis
- `preco`: valor monetário do mangá

---

## 4. Lista de rotas REST utilizadas

A aplicação implementa as seguintes rotas REST:

| Método HTTP | Rota | Função |
|---|---|---|
| GET | `/mangas/` | Retorna todos os mangás cadastrados |
| GET | `/mangas/<id>/` | Retorna os dados de um mangá específico |
| POST | `/mangas/` | Cria um novo mangá |
| PUT | `/mangas/<id>/` | Atualiza os dados de um mangá existente |
| DELETE | `/mangas/<id>/` | Remove um mangá da lista |

🛠️ **Como funciona cada método:**

- **GET /mangas/**
  Lê o conteúdo do arquivo `mangas.json`. Se a lista estiver vazia, exibe mensagem:

```
  if not mangas:
return JsonResponse({"mensagem": "Nenhum mangá cadastrado."})
```

- **GET /mangas/<id>/**
  Busca o mangá pelo ID e o retorna. Caso não exista, retorna erro 404.

- **POST /mangas/**
  Recebe os dados via JSON, gera um novo ID, adiciona o mangá à lista e **salva no arquivo** usando:

```
with open('mangas.json', 'w') as f:
json.dump(mangas, f)
```

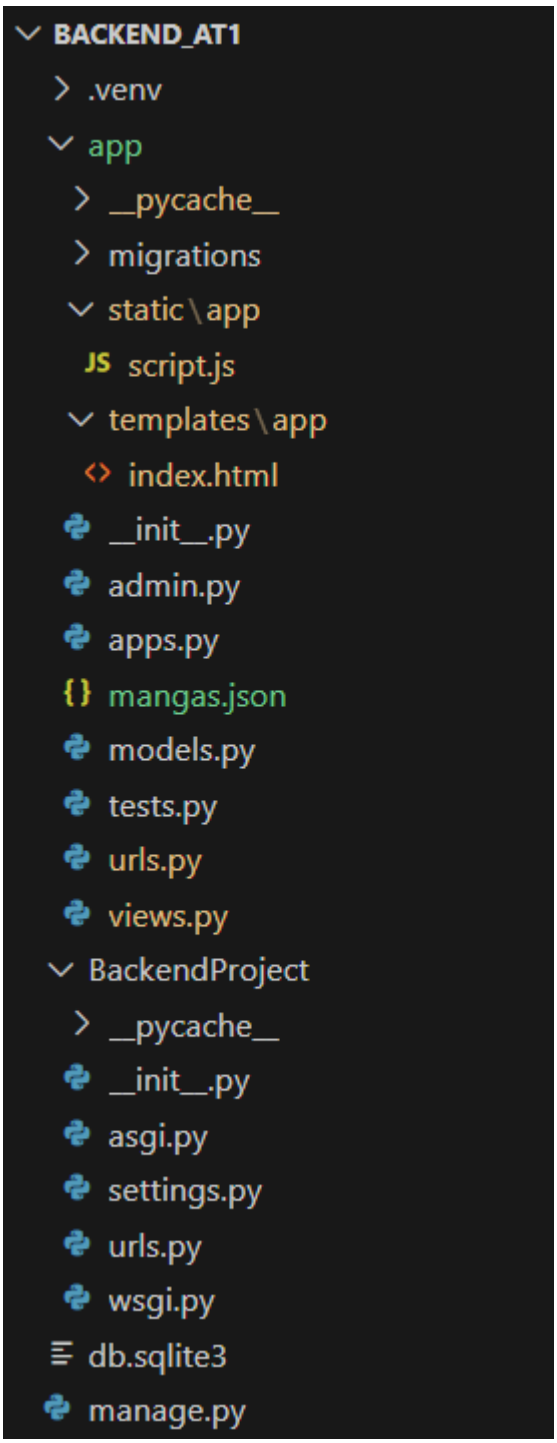
- **PUT /mangas/<id>/**
  Permite editar os dados de um mangá existente. Apenas os campos enviados são atualizados. O restante é mantido.

- **DELETE /mangas/<id>/**
  Remove o item com o ID fornecido e salva a nova lista no arquivo.

---

## 5. Estrutura geral da aplicação

A aplicação continua estruturada em:




## Roteamento e exibição de páginas

- **Página inicial `/`**
  Usa `HttpResponse` para exibir uma saudação e um link para a interface gráfica.

- **Interface `/mangas/interface/`**
  Exibe um HTML com botões para:

- Listar mangás (GET)
- Adicionar novo (POST)
- Buscar por ID (GET /id)
- Deletar (DELETE)
- Atualizar (PUT) – **novo na AT2**

---

## Funcionalidades da interface (HTML + JS)

| Função | Implementado? | Observação |
|---|---|---|
| **Listar** | ✅ | **Com alternância (mostrar/ocultar)** |
| **Adicionar** | ✅ | **Com formulário e fetch POST** |
| **Buscar por ID** | ✅ | **Mostra dados detalhados** |
| **Remover** | ✅ | **Botão de exclusão ao lado de cada item** |
| **Atualizar (PUT)** | ✅ **Novo** | **Formulário para editar dados por ID** |

---

## 6. Persistência em arquivo local (novo)

Os dados agora são armazenados e carregados a partir do arquivo `mangas.json`.
Para isso, foram criadas duas funções auxiliares:

```
def ler_dados():
    with open('mangas.json', 'r') as f:
        return json.load(f)
```

```
def salvar_dados(lista):
    with open('mangas.json', 'w') as f:
        json.dump(lista, f)
```

Assim, **os dados persistem mesmo com o servidor sendo reiniciado**, sem usar banco de dados.

---

## ✅ Conclusão

A Atividade 2 foi concluída com sucesso, evoluindo a estrutura da AT1. Agora a aplicação conta com um CRUD completo e persistência de dados em arquivo local, com todas as rotas REST implementadas e integradas à interface front-end. O projeto está organizado, funcional e pronto para futuras expansões com banco de dados e autenticação (AT3).