Relatório de Desenvolvimento e Guia da API R+Cidades (Backend)

Este documento resume a arquitetura técnica, o processo de desenvolvimento e o guia de uso da API (back-end) do projeto R+Cidades, construída em 06 de Novembro de 2025.

1. Relatório do que Foi Feito

O objetivo foi implementar o back-end da "Proposta A" do R+Cidades, baseando-se nos documentos de pesquisa, no "Cérebro do Aplicativo (Schema v3.0)" e nas "Jornadas de Usuário".

1.1. Arquitetura e Tecnologias

- **Framework:** Laravel 12
- **Ambiente:** Laravel Sail (Docker) rodando sobre WSL 2 (Ubuntu no Windows)
- **Banco de Dados:** MySQL
- **Arquitetura:** API *Headless* (Desacoplada). O Laravel serve apenas dados (JSON) e não telas (HTML). O front-end (React) será um projeto separado que consome esta API.

1.2. Fases do Desenvolvimento

Nosso trabalho foi dividido em 4 fases principais:

- **Fase 1: Entidades de Suporte (A Fundação)**
  - Criamos as tabelas base que não dependiam de outras: users, categoria_materials e bancos_de_materiais.
  - Refatoramos a tabela anuncios para adicionar as chaves estrangeiras (categoria_material_id) e colunas de métrica (peso_estimado_kg).
- **Fase 2: População do Banco (Seeding)**
  - Criamos *Factories* e *Seeders* para popular o banco de dados.
  - O banco agora é populado automaticamente com **10 usuários**, **6 categorias** e **50 anúncios** de teste, essenciais para o desenvolvimento do dashboard.
- **Fase 3: Fluxo de Transação (O "Core")**
  - Implementamos o fluxo principal da jornada do usuário.
  - Criamos as tabelas solicitacaos (para o "match") e agendamento_logisticas (para a entrega).
- **Fase 4: Autenticação e Analytics (A "Inteligência")**
  - Implementamos o **Laravel Sanctum** para autenticação de API baseada em Token (Bearer Token).
  - Criamos rotas de register, login e logout.

- Protegemos os *endpoints* de "ação" (como criar anúncios/solicitações) com o *middleware* auth:sanctum.
- Criamos o *endpoint* final GET /api/dashboard para calcular e entregar os KPIs de impacto.

2. Guia da API R+Cidades (localhost:8001)

Este é um guia prático para usar os *endpoints* da API (através de ferramentas como o Thunder Client).

2.1. Autenticação

A API usa "Bearer Token". Você deve primeiro se registrar ou fazer login para obter um access_token. Esse token deve ser enviado no *Header* de autorização para rotas protegidas.

Header (Para rotas protegidas):
Authorization: Bearer <SEU_TOKEN_AQUI>

POST /api/register

Registra um novo usuário (Doador ou Beneficiário).

- **Body (JSON):**
```
{
  "name": "Beneficiaria Ana",
  "email": "ana@teste.com",
  "password": "password123"
}
```

- **Resposta (Sucesso 201 Created):**
```
{
  "message": "Usuário registrado com sucesso!",
  "user": { "id": 11, "name": "Beneficiaria Ana", ... }
}
```

POST /api/login

Autentica um usuário e retorna um token de acesso.

- **Body (JSON):**
```
{
  "email": "ana@teste.com",
  "password": "password123"
}
```

- **Resposta (Sucesso 200 OK):**
  {
    "message": "Login bem-sucedido!",
    "access_token": "4|aBcDeFgHiJkL123456...",
    "token_type": "Bearer",
    "user": { "id": 11, ... }
  }

2.2. Endpoints Públicos (Leitura)

GET /api/anuncios

Retorna uma lista de todos os anúncios publicados (o Catálogo).

- **Resposta (Sucesso 200 OK):**
  [
   { "id": 1, "titulo": "Sobra de Tijolos", ... },
   { "id": 2, "titulo": "Lote de Madeiras", ... },
   ... 50+ anúncios ...
  ]

GET /api/anuncios/{id}

Retorna os detalhes de um único anúncio.

- **Exemplo de URL:** http://localhost:8001/api/anuncios/2
- **Resposta (Sucesso 200 OK):**
  { "id": 2, "titulo": "Lote de Madeiras", ... }

GET /api/dashboard

Retorna os KPIs (Métricas) calculados para o Dashboard de Impacto.

- **Resposta (Sucesso 200 OK):**
  {
   "total_peso_doado_kg": 10609,
   "total_solicitacoes_feitas": 1,
   "total_usuarios_registrados": 11,
   "total_anuncios_publicados": 51
  }

2.3. Endpoints Protegidos (auth:sanctum)

(Obrigatório enviar o Bearer Token no Header "Authorization")

POST /api/anuncios

Cria um novo anúncio. O usuario_id é pego automaticamente do usuário logado.

- **Body (JSON):**
  ```
  {
    "titulo": "Anuncio da Ana (Logada)",
    "descricao": "Meu primeiro anuncio logada.",
    "quantidade": "1 peca",
    "condicao": "Nova"
  }
  ```

- **Resposta (Sucesso 201 Created):**
  { "id": 52, "usuario_id": 11, "titulo": "Anuncio da Ana (Logada)", ... }

POST /api/solicitacoes

Cria uma nova solicitação para um anúncio. O beneficiario_id é pego automaticamente do usuário logado.

- **Body (JSON):**
  ```
  {
    "anuncio_id": 1,
    "justificativa_beneficiario": "Preciso deste material para o muro."
  }
  ```

- **Resposta (Sucesso 201 Created):**
  { "id": 2, "anuncio_id": 1, "beneficiario_id": 11, ... }

POST /api/agendamentos

Cria um novo agendamento de logística para uma solicitação.

- **Body (JSON):**
  ```
  {
    "solicitacao_id": 1,
    "data_agendada": "2025-11-07 14:00:00",
    "transportador_id": 3
  }
  ```

- **Resposta (Sucesso 201 Created):**
  { "id": 1, "solicitacao_id": 1, "transportador_id": 3, ... }

GET /api/solicitacoes e GET /api/agendamentos

Rotas (protegidas) que listam todas as solicitações e agendamentos, provavelmente para um painel de Admin.

3. Dificuldades Encontradas (Debug)

Durante o desenvolvimento, enfrentamos vários desafios técnicos que foram cruciais para o aprendizado e para a estabilidade final da API:

1. **Conflito de Portas (Ambiente):** O localhost (porta 80) do novo projeto R+Cidades entrou em conflito com o projeto ServLink.
    - **Solução:** Editamos o .env para usar portas únicas (APP_PORT=8001 e FORWARD_DB_PORT=3307).
2. **Erro Connection refused (Banco de Dados):** O comando sail php artisan migrate era executado rápido demais, antes que o contêiner do MySQL estivesse pronto.
    - **Solução:** Adicionamos uma espera de 10-15 segundos após o sail up -d.
3. **Erro Table 'categoria_materials' doesn't exist (Laravel):** Tivemos um erro de convenção onde a Migração criou a tabela categorias_material (singular) mas o Seeder/Model procurou por categoria_materials (plural).
    - **Solução:** Corrigimos as migrações para usar o nome plural padrão do Laravel, garantindo consistência.
4. **Erro Route [login] not defined (API):** O erro mais complexo. A API, ao bloquear um usuário não logado (auth:sanctum), tentava redirecioná-lo para uma rota "web" chamada login (que não existia), causando um erro 500.
    - **Solução:** Editamos o bootstrap/app.php para capturar a AuthenticationException e forçar o Laravel a retornar um erro 401 em JSON (o comportamento correto de uma API).
5. **Erro Call to undefined method User::createToken() (API):** O login falhou porque o Model User não estava configurado para o Sanctum.
    - **Solução:** Adicionamos o *Trait* use HasApiTokens; ao arquivo app/Models/User.php.

4. Resultado Final

O back-end da API R+Cidades está 100% operacional, seguro e populado com dados de teste. Todos os *endpoints* principais definidos na proposta inicial foram implementados e testados. A API agora está pronta para ser consumida por um front-end (React, Vue, etc.) ou por ferramentas de análise (Power BI).