# MOV – Frontend e Backend

**Mickeias Charles de Oliveira Paiva, Tallys Araújo dos Santos, Renan Silvestre de Oliveira Vale e Wellington Henrique Santos Marques.**

Ciência da Computação

**Residência Tecnológica Porto Digital – Prof. Ranyelson**

**Residência Tecnológica Porto Digital – Mentor Michel**

Universidade Católica de Brasília (UCB)

Brasília – DF – Brasil

## 1. Introdução e Propósito

## 2. Este documento descreve a arquitetura e os componentes técnicos do sistema MOV (Monitoramento Ostensivo de Violação). O objetivo principal do MOV é fornecer uma interface para rastrear ativos (bueiros) em tempo real e gerenciar eventos de segurança e manutenções agendadas.

Público-Alvo: Desenvolvedores, Engenheiros de Software e Equipes de Manutenção.

### 2.1. Contribuição Individual

**Mickeias Charles de Oliveira Paiva**,

No projeto 'MOV', atou como desenvolvedor backend, devops, tester, designer e arquitetura do sistema.

**Tallys Araújo dos Santos,**

Atuou como desenvolvedor frontend e banco de dados.

**Renan Silvestre de Oliveira Vale,**

Atuou como desenvolvedor frontend.

**Wellington Henrique Santos Marques,**

Atuou como desenvolvedor frontend.

## 3. Arquitetura geral do Sistema

O MOV utiliza uma arquitetura Cliente-Servidor (Frontend/Backend) desacoplada, comunicando-se exclusivamente via API RESTful.

| Componente | Localização | Função |
|---|---|---|
| Frontend | mov-frontend/ | Interface do Usuário (UI), gerenciamento de estado e chamadas de API. |
| Backend | mov-backend/ | API RESTful, lógica de negócio, autenticação e acesso ao banco de dados. |
| Banco de dados | Aiven Cloud MySQL | Armazenamento persistente de usuários, ativos e eventos. |

# 4. Detalhamento do Backend (API e Lógica de Negócio)

## 4.1. Stack Tecnológico

| Tecnologia | Finalidade |
|---|---|
| Framework | Node.js (ESM) + Express.js |
| Banco de Dados | MySQL (Driver mysql2/promise) |
| Segurança | bcryptjs (Hashing de Senha) |
| Middlewar | multer (Upload de arquivos) |

## 4.2. Modelo de Rota e Estrutura de Pastas

A API segue o padrão MVC (Model-View-Controller, sendo o "Model" representado pelas Queries nos Controllers).

| Rota Base (app.js) | Controller | Descrição |
|---|---|---|
| /api/usuarios | authController.js | Login, Registro, Listagem (CRUD básico de usuários). |
| /api/perfil | perfilController.js | Busca, atualização de dados pessoais e upload de foto de perfil. |
| /api/bueiros | bueirosController.js | CRUD de ativos (bueiros). Lógica de exclusão em cascata manual (deleta eventos relacionados). |
| /api/manutencoes | manutencoesController.js | Agendamento de manutenções. |
| /api/dashboard | dashboardController.js | Retorna dados agregados (gráficos, alertas e listagem completa de eventos). **Contém a lógica de Prioridade e Status.** |

## 4.3. Lógica Crítica (dashboardController.js)

As funções de Dashboard calculam as métricas de incidentes:

Prioridade: Calculada usando `TIMESTAMPDIFF(HOUR, e.timestamp, NOW())`.

`Alta`: <= 1 hora.

`Média`: > 1 hora e <= 24 horas.

Status de Atendimento: Baseado na idade do evento (Aberto, Em Atendimento, Fechado).

Seeding: Contém lógica `seedEventosSeVazio` para popular a tabela de eventos em ambiente de desenvolvimento.

## 5. Detalhamento do Frontend

### 5.1. Arquitetura React

6. **Roteamento: Utiliza React Router v6 com Rotas Aninhadas (**<DashboardLayout /> **contém** <Outlet /> **para carregar as páginas filhas).**

7. **Autenticação (Contexto): O** AuthContext.jsx **gerencia o estado global do** currentUser **e usa o** localStorage **para persistência de login (**mov-user**).**

8. **Segurança (Rotas): O componente** ProtectedRoute.jsx **implementa o RBAC (Role-Based Access Control), verificando** allowedRoles **antes de renderizar páginas sensíveis (**/gestao-usuarios**,** /gestao-manutencoes**).**

9. **Comunicação API: Centralizada em** services/api.js **(Axios), padronizando respostas para** {ok: true, data: ...}**.**

### 9.1. Componentes Chave

| Componente | Propósito | Observação |
|---|---|---|
| Modal.jsx | Modal genérico e reutilizável. | Utiliza CSS :has() para ajustar o tamanho ao conteúdo (Mapas vs. Formulários). |
| MapaDashboard.jsx | Widget de mapa na Home. | Interatividade desativada (dragging=false) para evitar conflito com a rolagem da página. |
| MapaModal.jsx | Versão ampliada do mapa. | Interatividade ativada (scrollWheelZoom=true). |
| Sidebar.jsx | Navegação principal. | Implementa a lógica de exibição de links baseada na role do currentUser. |

### 9.2. Banco de Dados

O banco de dados principal é o mov_db.

| Tabela | Descrição | Chaves Estrangeiras (FKs) |
|---|---|---|
| usuarios | Dados dos usuários e nível de permissão (role). | Nenhuma |
| bueiros | Cadastro de ativos (dispositivos IoT), coordenadas e status. | Nenhuma |
| manutencoes | Agendamentos de manutenção. | bueiro_id -> bueiros(id) |
| eventos | Dados brutos de alerta (abertura), inseridos via hardware/IoT. | bueiro_id -> bueiros(id) |