Trackin.API - Documento Complementar Sprint 1

Visão Geral do Projeto

O Trackin.API é parte de um sistema abrangente que visa automatizar o mapeamento e monitoramento de motocicletas nos pátios da Mottu, utilizando uma combinação de tecnologia RFID e visão computacional para localização em tempo real. Baseado em uma arquitetura robusta em .NET, o sistema oferece APIs RESTful para gestão completa das motos nos pátios da empresa.

Arquitetura Geral do Sistema

O sistema completo, conforme planejado na arquitetura, é composto por várias camadas e componentes integrados:

Modelo de Domínio Completo

1. **Moto**: Informações sobre as motocicletas, incluindo placa, modelo, status, etc.
2. **Pátio**: Informações sobre os locais onde as motos são armazenadas.
3. **Camera**: Dispositivos para captura de imagens para visão computacional.
4. **LocalizacaoMoto**: Registro da posição atual das motos no pátio.
5. **Filial**: Unidades da empresa que gerenciam os pátios.
6. **DeteccaoVisual**: Registros de detecção das motos por câmeras.
7. **Usuario**: Usuários do sistema.
8. **EventoMoto**: Registro de eventos relacionados às motos.
9. **ZonaPatio**: Áreas específicas dentro dos pátios.
10. **SensorRFID**: Dispositivos para leitura de tags RFID nas motos.

Arquitetura em Camadas

1. **Camada de Apresentação**: Controllers da API REST.
2. **Camada de Aplicação**: Serviços que implementam a lógica de negócio.
3. **Camada de Domínio**: Entidades de negócio e regras de domínio.
4. **Camada de Infraestrutura**: Persistência, serviços externos e integrações.

Tecnologias Integradas

- **RFID**: Para identificação e aproximação da localização das motos.
- **Visão Computacional**: Usando ML.NET para localização precisa e identificação visual.
- **Fusão de Dados**: Combinação de informações de RFID e visão computacional.

Implementação da Sprint 1

Na Sprint 1, focamos na criação da estrutura base do sistema e na implementação dos componentes essenciais. A implementação priorizou o CRUD das entidades principais e a integração com o banco de dados Oracle, seguindo as boas práticas de desenvolvimento e a arquitetura planejada.

Entidades Implementadas

1. **Moto**
   - Atributos: Id, Placa, Modelo, Ano, Status, RFIDTag, DataAquisicao, UltimaManutencao
   - Relacionamentos: Localização no pátio, eventos associados
2. **Patio**
   - Atributos: Id, Nome, Endereço, Cidade, Estado, País, Dimensões, PlantaBaixa
   - Relacionamentos: Contém zonas, câmeras e motos
3. **SensorRFID**
   - Atributos: Id, PatioId, Posição, Status
   - Relacionamentos: Associado a um pátio
4. **ZonaPatio**
   - Atributos: Id, PatioId, Nome, Coordenadas, Cor
   - Relacionamentos: Pertence a um pátio

Rotas Implementadas

MotoController

- `GET /api/moto` - Lista todas as motos
- `GET /api/moto/{id}` - Obtém moto específica
- `GET /api/moto/patio/{patioId}` - Lista motos por pátio
- `GET /api/moto/status/{status}` - Lista motos por status
- `POST /api/moto` - Cadastra nova moto
- `PUT /api/moto/{id}` - Atualiza moto
- `DELETE /api/moto/{id}` - Exclui moto
- `POST /api/moto/{id}/imagem` - Adiciona imagem de referência

PatioController

- `GET /api/patio` - Lista todos os pátios
- `GET /api/patio/{id}` - Obtém pátio específico
- `POST /api/patio` - Cadastra novo pátio
- `DELETE /api/patio/{id}` - Remove pátio

RFIDController

- `POST /api/rfid` - Processa leitura RFID e atualiza localização

SensorRFIDController

- `GET /api/sensorRFID` - Lista todos os sensores
- `GET /api/sensorRFID/{id}` - Obtém sensor específico
- `POST /api/sensorRFID` - Cadastra novo sensor
- `PUT /api/sensorRFID/{id}` - Atualiza sensor
- `DELETE /api/sensorRFID/{id}` - Remove sensor

ZonaPatioController

- `GET /api/zonaPatio` - Lista todas as zonas
- `GET /api/zonaPatio/{id}` - Obtém zona específica
- `POST /api/zonaPatio` - Cadastra nova zona
- `PUT /api/zonaPatio/{id}` - Atualiza zona
- `DELETE /api/zonaPatio/{id}` - Remove zona

Aspectos Técnicos Implementados

Camadas da Aplicação

1. **Domain**
   - Entidades ricas com comportamento e validações de negócio
   - Interfaces de repositórios
   - Enums e tipos de valor
2. **Application**
   - DTOs para transferência de dados
   - Serviços de aplicação com lógica de negócio
   - MappingConfig com AutoMapper para conversão entre entidades e DTOs
   - Validações de entrada
3. **Infrastructure**
   - Implementação de repositórios com EF Core
   - Contexto de banco de dados e configurações
   - Migrations para versionamento do banco
   - Configurações de entidades (Fluent API)

4. **API (Presentation)**
    - Controllers RESTful
    - Filtros e middlewares
    - Configuração do Swagger/OpenAPI
    - Tratamento de exceções globais

Banco de Dados

- Integração com Oracle via Entity Framework Core
- Migrations para criação e versionamento das tabelas
- Configuração de relacionamentos entre entidades
- Uso de variáveis de ambiente para strings de conexão

Documentação

- Swagger/OpenAPI configurado e funcional
- Documentação clara dos endpoints, parâmetros e respostas
- README detalhado com instruções de instalação e execução

Diferenças entre a Arquitetura Planejada e a Implementação da Sprint 1

Alterações nas Rotas

- As rotas implementadas utilizam nomes no singular (ex.: `/api/moto` em vez de `/api/motos`)
- Ainda não foram implementadas todas as rotas previstas na arquitetura completa

Entidades Não Implementadas na Sprint 1

- Camera
- LocalizacaoMoto (parcialmente implementada via RFIDController)
- DeteccaoVisual
- Usuario
- EventoMoto
- Filial

Funcionalidades Adiadas para Próximas Sprints

- Integração completa com visão computacional
- Fusão de dados entre RFID e visão computacional
- Atualizações em tempo real via SignalR
- Sistema de notificações e alertas

- Calibração de câmeras e mapeamento de coordenadas

Próximos Passos

Sprint 2 (Planejada)

- Implementação das demais entidades: Camera, LocalizacaoMoto, Usuario, EventoMoto
- Desenvolvimento da integração básica com câmeras IP
- Implementação do SignalR para atualizações em tempo real
- Autenticação e autorização via JWT

Sprint 3 (Planejada)

- Implementação da visão computacional com ML.NET
- Fusão de dados entre RFID e visão computacional
- Calibração de câmeras e mapeamento de coordenadas
- Melhoria do sistema de notificações e alertas

Conclusão

A Sprint 1 estabeleceu as fundações do sistema Trackin.API, implementando as funcionalidades essenciais seguindo a arquitetura planejada. Apesar de algumas diferenças nas rotas e entidades implementadas em relação à arquitetura completa, a estrutura básica foi estabelecida de acordo com as boas práticas de desenvolvimento e os requisitos iniciais do projeto.

As próximas sprints continuarão a construir sobre esta base, adicionando as funcionalidades mais avançadas de visão computacional e fusão de dados conforme planejado na arquitetura geral do sistema.