Capítulo 35

Desenvolvimento do Backend da Plataforma Mapa Social do Maranhão com Ferramentas de Código Aberto

Backend Development of the Maranhão Social Map Platform Using Open-Source Tools

Júlio César Costa da Silva
Curso de Engenharia da Computação. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Ana Elisa Oliveira Silva
Curso de Engenharia da Computação. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Bruno Raphael Andrade Santos
Curso de Engenharia da Computação. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Carlos Vinícius Rodrigues Amorim
Curso de Engenharia da Computação. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Keyll Carlos Ribeiro Martins
Programa de Pós-Graduação em Engenharia Mecânica. Instituto Federal do Maranhão (IFMA). São Luís, MA, Brasil.

Waldemir dos Passos Martins
Programa de Pós-Graduação em Engenharia Mecânica. Instituto Federal do Maranhão (IFMA). São Luís, MA, Brasil.

Denner Guilhon
Departamento de Engenharia Mecânica. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Paulo Fernandes da Silva Junior
Departamento de Engenharia da Computação. Universidade Estadual do Maranhão (UEMA). São Luís, MA, Brasil.

Resumo: Este capítulo apresenta o desenvolvimento do back-end da plataforma Mapa Social do Maranhão, voltado à organização e à disponibilização de indicadores sociais dos 217 municípios do estado. O objetivo é descrever a arquitetura dos serviços, a persistência dos dados e os mecanismos de integração que sustentam as consultas da aplicação. A metodologia compreende a estruturação de um banco de dados PostgreSQL, a implementação de uma API REST em Java com Spring Boot e a comunicação com um serviço interno em Python com FastAPI para processamento analítico. O processo de coleta e tratamento de dados em Python fornece as informações utilizadas pelos serviços. No back-end principal, Spring Data JPA realiza o mapeamento das entidades, enquanto a validação de parâmetros, o tratamento global de exceções, a limitação de requisições e a documentação com Springdoc OpenAPI apoiam a organização da API. Como resultado, a versão descrita reúne consultas temáticas, séries históricas, comparação territorial e resumos estatísticos, além de integrar funcionalidades de projeção e agrupamento. A documentação no Swagger UI explicita operações, formatos de resposta e condições de acesso ao ambiente interno. Conclui-se que a separação de responsabilidades entre persistência, regras de negócio e processamento analítico constitui uma base para a evolução da plataforma, permanecendo necessária a avaliação formal de desempenho e segurança em produção.

Palavras-chave: back-end; API REST; indicadores sociais; Spring Boot; PostgreSQL.

DOI: 10.47573/aya.5379.3.33.35

Abstract: This chapter presents the development of the back end of the Mapa Social do Maranhão platform, designed to organize and provide social indicators for the state's 217 municipalities. The objective is to describe the service architecture, data persistence, and integration mechanisms supporting application queries. The methodology comprises a PostgreSQL database, a Java REST API built with Spring Boot, and communication with an internal Python service using FastAPI for analytical processing. A Python data collection and processing pipeline supplies the information consumed by these services. In the main back end, Spring Data JPA maps entities, while parameter validation, global exception handling, request limiting, and Springdoc OpenAPI documentation support API organization. The described version provides thematic queries, historical series, territorial comparisons, and statistical summaries, alongside integration with forecasting and clustering functions. Swagger UI documentation describes operations, response formats, and access conditions for the internal environment. The results show how database access and business rules can remain in the main service while analytical processing is delegated to a separate component. This separation provides a foundation for further development of the platform. Formal assessments of performance and security in production remain necessary to characterize its operational behavior and limitations.

Keywords: back-end; REST API; social indicators; Spring Boot; PostgreSQL.

INTRODUÇÃO

A integração de indicadores sociais exige serviços capazes de consultar dados de diferentes áreas, preservar suas referências e disponibilizar respostas para os componentes de uma aplicação. No projeto Mapa Social do Maranhão, desenvolvido em parceria entre o Ministério Público do Estado do Maranhão (MPMA) e a Universidade Estadual do Maranhão (UEMA), essa necessidade se materializa no atendimento a consultas sobre os *217* municípios maranhenses, em áreas como Saúde, Educação e Assistência Social.

O back-end constitui a camada responsável pela comunicação com a base relacional, pela aplicação das regras de negócio e pela disponibilização dos dados por meio de uma API REST. O Spring Boot oferece recursos para a construção de aplicações Java e constitui o framework adotado no serviço principal da plataforma (Spring, s.d.). Sua organização influencia a forma como municípios, indicadores, referências temporais e resultados analíticos são recuperados e encaminhados à interface. Além das operações de consulta, a plataforma demanda integração com serviços de projeção e de agrupamento de municípios.

A persistência dos indicadores requer uma estrutura que preserve os relacionamentos entre municípios, períodos e valores observados. Os fundamentos do modelo relacional e da integridade dos dados apresentados por Date (*2003*) e Celko (*2014*) orientam a discussão dessa organização. Na plataforma, o PostgreSQL fornece o sistema gerenciador utilizado para armazenar e consultar os registros relacionais (PostgreSQL Global Development Group, s.d.).

A camada de serviços também se articula com a preparação e o processamento analítico dos dados. Os recursos de tratamento com Python e Pandas apresentados por McKinney (*2018*) apoiam a compreensão da etapa que

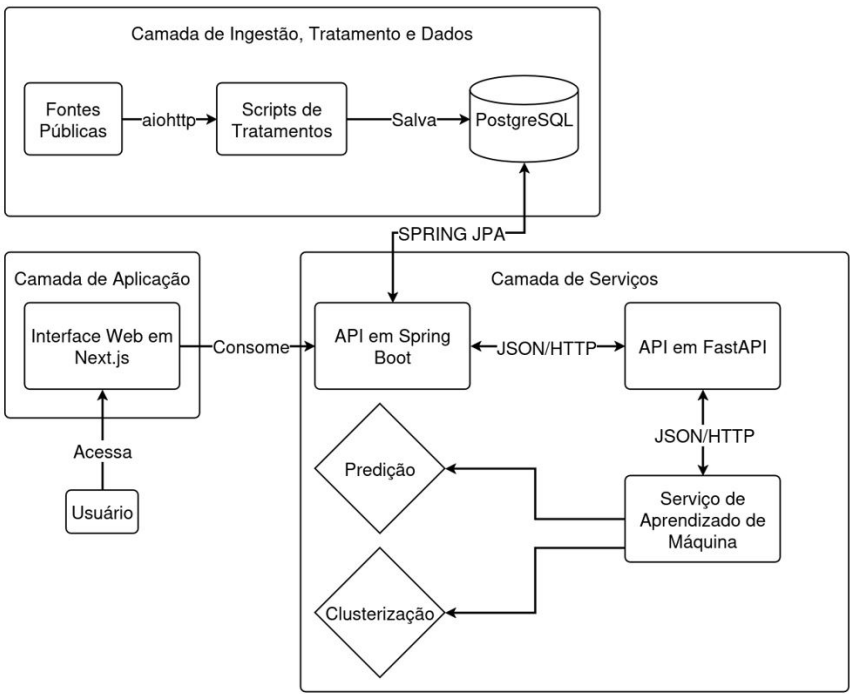
abastece a base consultada pelo back-end. A biblioteca Scikit-learn, descrita por Pedregosa *et al.* (*2011*), integra o serviço analítico utilizado nas funcionalidades de projeção e agrupamento. Esse conjunto situa o back-end entre a persistência dos indicadores e a disponibilização de resultados para a aplicação.

O objetivo deste capítulo é descrever o desenvolvimento do back-end da plataforma, com ênfase na arquitetura, no armazenamento relacional, na implementação da API, na documentação das operações e na integração com o serviço analítico. O recorte concentra-se em PostgreSQL, Spring Boot e FastAPI; a coleta de dados e a interface web são apresentadas somente quando necessárias para compreender os fluxos que chegam aos serviços e as respostas que eles disponibilizam.

ARQUITETURA E ORGANIZAÇÃO DOS SERVIÇOS

A arquitetura separa a ingestão e o tratamento de dados, a persistência, as regras de negócio, o processamento analítico e a apresentação das informações. O pipeline em Python prepara os dados das fontes governamentais; o PostgreSQL armazena os indicadores; o Spring Boot organiza as consultas e a integração dos serviços; e o FastAPI executa as rotinas analíticas. A Figura *1* situa o back-end na arquitetura geral da plataforma.

Figura 1 – Arquitetura geral da plataforma Mapa Social do Maranhão.



Fonte: elaboração dos autores (2026).

O Spring Boot concentra o acesso ao banco de dados e as regras de negócio. Quando uma operação exige processamento analítico, esse componente encaminha ao serviço FastAPI os dados necessários, recebe o resultado e o devolve ao componente solicitante. Essa divisão mantém o processamento dos modelos separado das consultas e das validações realizadas no serviço principal.

A interface web participa desse fluxo como consumidora das operações disponibilizadas. Na configuração documentada, o back-end permanece na rede interna e as operações publicadas são encaminhadas por um proxy. Por isso, a documentação do serviço interno pode incluir operações que não estão disponíveis diretamente à interface pública. A distinção entre esses dois conjuntos de operações deve orientar a leitura do catálogo da API.

ARMAZENAMENTO E PREPARAÇÃO DOS DADOS

Modelo Relacional e Referência Municipal

O armazenamento utiliza o PostgreSQL como sistema gerenciador de banco de dados relacional, pela sua natureza de código aberto e por seus recursos de integridade relacional (PostgreSQL Global Development Group, s.d.). A tabela de referência municipal utiliza o código IBGE como identificador e contempla os *217* municípios do Maranhão. As tabelas de indicadores são organizadas em esquemas temáticos e relacionam-se à referência municipal por meio de chaves primárias e estrangeiras.

Essa estrutura busca reduzir redundâncias e preservar a integridade dos relacionamentos, em consonância com os fundamentos de bancos de dados relacionais apresentados por Celko (*2014*) e Date (*2003*). A identificação municipal compartilhada permite recuperar indicadores de diferentes áreas a partir de uma mesma referência territorial. As consultas também precisam considerar a referência temporal, a fonte e o valor observado, evitando tratar registros de períodos distintos como equivalentes.

A organização temática não elimina as diferenças entre as fontes. Unidades, escalas e referências temporais podem variar entre indicadores, e a documentação apresentada informa que a API não normaliza automaticamente unidades ou moedas. Desse modo, a recuperação de um valor deve ser acompanhada da interpretação de seus metadados; a comparação de respostas requer verificar o significado de cada indicador e o período a que ele se refere.

Integração com o Processo de Coleta

Os dados utilizados pelo back-end são preparados por scripts em Python (Python Software Foundation, s.d.). A coleta utiliza *asyncio* e *aiohttp* para realizar requisições assíncronas às fontes SIDRA, SAGICAD, IMESC e IPEADATA. O tratamento utiliza Pandas para selecionar colunas, tratar valores ausentes, converter tipos e padronizar referências temporais e códigos municipais (McKinney, *2018*).

Após o tratamento, os dados são armazenados no PostgreSQL por meio da biblioteca SQLAlchemy, com atualizações periódicas controladas por um agendador interno. A versão descrita do pipeline reúne *48* séries de indicadores integrados. Essa etapa constitui a entrada de dados para os serviços de consulta: os controladores da API recuperam os registros persistidos, enquanto a coleta e a padronização permanecem sob responsabilidade do pipeline.

DESENVOLVIMENTO DA API REST

Persistência e Regras de Negócio

O back-end principal é desenvolvido em Java com o *framework Spring Boot* (Spring, s.d.). A API utiliza Spring Data JPA para o mapeamento das entidades e para a comunicação com a base relacional. Os serviços organizam o acesso às informações e a aplicação das regras de negócio, disponibilizando consultas por meio dos controladores REST.

Entre as funcionalidades documentadas estão a comparação territorial, a recuperação da evolução de um indicador, a obtenção de resumos estatísticos e a listagem de tabelas de esquemas analíticos. Também estão disponíveis grupos temáticos de consultas municipais, como as operações relacionadas a benefícios, Cadastro Único e infraestrutura domiciliar. A separação por finalidade e por área temática facilita localizar as operações durante o desenvolvimento e a integração.

Validação e Tratamento de Respostas

A implementação utiliza *Spring Validator* para validar parâmetros e dispõe de tratamento global de exceções. A organização desses mecanismos busca oferecer um ponto comum para tratar falhas e entradas inválidas, em vez de repetir a mesma lógica em cada operação. A configuração de CORS e o controle de requisições complementam os recursos utilizados no serviço principal.

Os formatos de resposta precisam ser interpretados conforme cada operação. Segundo as orientações da documentação, indicadores pontuais utilizam um envelope com propriedades em português, enquanto operações de predição, agrupamento e transparência podem adotar estruturas distintas. Em algumas respostas, o conteúdo associado ao indicador é uma string contendo JSON, o que exige atenção do consumidor para a desserialização.

A documentação também distingue a ausência de um registro da ausência de uma série: determinadas consultas retornam HTTP *200* com conteúdo nulo, enquanto listas sem registros podem ser representadas por um vetor vazio. Esses comportamentos devem ser considerados na integração para que a interface represente adequadamente a falta de dados, sem atribuir a ela um valor numérico ou uma falha de comunicação.

Acesso e Controle de Requisições

O back-end documentado opera em ambiente interno. O proxy encaminha apenas as operações publicadas e inclui a credencial privada exigida pelo serviço. A documentação informa ainda que o acesso ao Swagger depende de sua habilitação em ambiente administrado. A exposição da documentação e a publicação de operações para consumo pela interface constituem, portanto, configurações distintas.

O controle de requisições utiliza Resilience*4*j nas rotas anotadas, e a documentação registra limites e respostas associados ao excesso de chamadas. Esses mecanismos fazem parte da implementação descrita, mas sua presença, isoladamente, não demonstra a capacidade do sistema sob carga nem substitui uma avaliação de segurança. A interpretação de erros também deve considerar que o proxy pode gerar respostas próprias antes que a solicitação alcance o back-end.

DOCUMENTAÇÃO E INTEGRAÇÃO ANALÍTICA

Documentação com OpenAPI e Swagger UI

A API utiliza Springdoc OpenAPI para documentar os endpoints e disponibiliza sua consulta no Swagger UI. A página inicial, apresentada na Figura *2*, reúne orientações sobre acesso, formatos de resposta, ausência de dados, interpretação dos indicadores e limites de requisições. Essas informações complementam o catálogo de operações e ajudam a compreender as condições de uso do serviço interno.

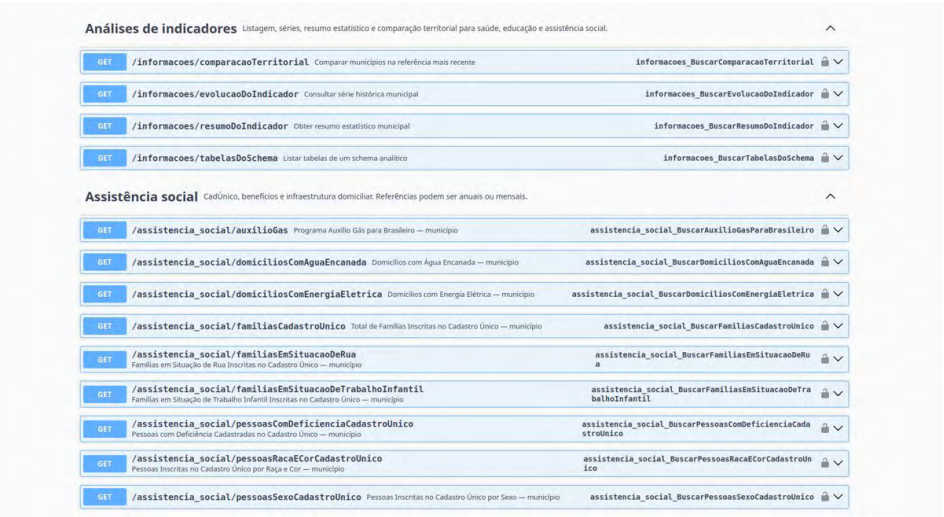
Figura 2 – Página inicial da documentação da API no Swagger UI.



Fonte: elaboração dos autores (2026).

A Figura *3* apresenta a organização das operações nos grupos de análises de indicadores e de assistência social. A identificação do método HTTP e a descrição de cada operação permitem localizar as consultas e compreender sua finalidade. Na documentação exibida, as operações de consulta utilizam GET; chamadas POST destinadas ao serviço Python pertencem ao fluxo interno de processamento.

Figura 3 – Operações de análises de indicadores e de assistência social no Swagger UI.



Fonte: elaboração dos autores (2026).

Comunicação com o Serviço FastAPI

Um serviço interno em Python, desenvolvido com FastAPI e Scikit-learn, executa os modelos de regressão e de agrupamento (Pedregosa *et al., 2011*). O Spring Boot encaminha os dados necessários ao processamento e recebe os resultados para devolvê-los à interface. O acesso ao banco e as regras de negócio permanecem concentrados no serviço principal.

Na funcionalidade de projeção, o serviço analítico avalia modelos de Regressão Linear e Random Forest a partir das séries disponíveis. O processo descrito no projeto utiliza o Erro Absoluto Médio e o coeficiente de determinação para avaliar os resultados, selecionando o modelo de menor erro absoluto médio. Para indicadores percentuais, a projeção é limitada ao intervalo entre zero e cem.

No agrupamento de municípios, os valores são normalizados e são avaliadas configurações de agrupamento, com seleção baseada no índice silhouette. Para o back-end, essas funcionalidades representam operações de integração: os dados de entrada são obtidos e encaminhados pelo Spring Boot, enquanto o treinamento e a execução dos modelos pertencem ao serviço analítico. As projeções e os grupos resultantes não substituem a interpretação especializada dos indicadores.

RESULTADOS E DISCUSSÃO

O desenvolvimento resultou em uma API documentada que conecta a base relacional aos componentes consumidores e integra o processamento analítico. O uso de uma referência municipal compartilhada permite consultar informações dos *217* municípios sem duplicar sua identificação em cada fluxo de consulta. Os esquemas temáticos e a organização das operações favorecem a localização dos dados e a manutenção dos serviços.

A documentação evidencia que a integração exige mais do que conhecer o endereço de uma operação. Formatos de resposta distintos, referências temporais heterogêneas, valores ausentes e diferenças de unidade precisam ser considerados pelos consumidores da API. A explicitação desses aspectos reduz ambiguidades durante o desenvolvimento, embora não elimine a necessidade de validar o tratamento adotado em cada funcionalidade.

Durante o desenvolvimento do projeto, foram realizados testes automatizados para o serviço de projeção e agrupamento, incluindo casos de sucesso, validações de entrada, limites percentuais e funcionamento dos endpoints. O back-end principal também foi validado por compilação e execução. Esses procedimentos indicam o funcionamento das funcionalidades verificadas, mas não constituem uma avaliação formal de desempenho ou de segurança em produção.

A separação entre serviço principal e serviço analítico permite distinguir a recuperação dos dados das rotinas de aprendizado de máquina. Entretanto, a operação integrada continua dependente da disponibilidade dos componentes envolvidos e da coerência entre os formatos enviados e recebidos. Como desdobramento, a avaliação em produção deve abranger os fluxos completos, incluindo proxy, API, banco e serviço analítico, e não apenas a execução isolada de um endpoint.

CONSIDERAÇÕES FINAIS

Este capítulo descreveu o desenvolvimento do back-end da plataforma Mapa Social do Maranhão, com ênfase em persistência relacional, consultas REST, documentação e integração analítica. O PostgreSQL fornece a base de armazenamento, o Spring Boot concentra o acesso aos dados e as regras de negócio, e o FastAPI executa as rotinas de projeção e agrupamento. A arquitetura estabelece responsabilidades distintas para componentes que precisam atuar de forma integrada.

A implementação contempla validação de parâmetros, tratamento de exceções, controle de requisições e documentação das operações. A experiência também evidencia a necessidade de preservar referências temporais, fontes e formatos de resposta para interpretar corretamente os indicadores. Como trabalhos futuros, destacam-se a ampliação dos testes de integração, a avaliação formal de desempenho e segurança e o aperfeiçoamento da consistência dos contratos de resposta, considerando os diferentes grupos de operações.

REFERÊNCIAS

CELKO, J. **Joe Celko's SQL for Smarties: Advanced SQL Programming.** 5. ed. Morgan Kaufmann, 2014.

DATE, C. J. **An Introduction to Database Systems**. 8. ed. Pearson, 2003.

MCKINNEY, W. **Python for Data Analysis: Data Wrangling with Pandas**, NumPy and IPython. 2. ed. O'Reilly Media, 2018.

PEDREGOSA, F. *et al.* Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research,** v. 12, p. 2825–2830, 2011.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL 14 Documentation**. Disponível em: https://www.postgresql.org/docs/14/. Acesso em: 23 jun. 2026.

PYTHON SOFTWARE FOUNDATION. **Python Documentation**. Disponível em: https://docs.python.org/. Acesso em: 19 jun. 2026.

SPRING. **Spring Boot Reference Documentation**. Disponível em: https://docs.spring.io/spring-boot/. Acesso em: 23 jun. 2026.

AGRADECIMENTOS

Os autores agradecem à Universidade Estadual do Maranhão (UEMA), à Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão (FAPEMA), pela concessão da bolsa PIBITI, e ao Ministério Público do Estado do Maranhão (MPMA), pela parceria institucional que viabilizou o projeto Mapa Social.