REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

SAUDEPET: APLICATIVO DE CUIDADOS COM ANIMAIS DE ESTIMAÇÃO

SAUDEPET: PET CARE APPLICATION

SAUDEPET: APLICACIÓN DE CUIDADOS PARA MASCOTAS

Arthur Prescinotti Severino[1], Felipe Diniz Dallilo[2], André Luiz da Silva[3]

e6127057
https://doi.org/10.47820/recima21.v6i12.7057

PUBLICADO: 12/2025

RESUMO

O artigo teve como objetivo o desenvolvimento do aplicativo SaudePET, voltado ao auxílio de tutores no acompanhamento da saúde e rotina de seus animais de estimação. O projeto foi desenvolvido com base em métodos de análise, modelagem e implementação de *software*, utilizando as tecnologias *React Native* para a interface móvel, *Java* com o *framework Spring Boot* para o *backend* e *MySQL* como banco de dados. A aplicação permite o gerenciamento de informações relacionadas a vacinas, medicamentos e alimentação dos *pets*, promovendo um controle mais eficiente e organizado. Os resultados obtidos demonstraram que a integração entre as tecnologias adotadas possibilitou o desenvolvimento de um sistema funcional, seguro e de fácil manutenção. Conclui-se que o SaudePET tem potencial de contribuir para o bem-estar animal, oferecendo uma ferramenta prática e acessível aos tutores.

PALAVRAS-CHAVE: Aplicativo móvel. Banco de dados. *React Native*. Saúde animal. *Spring Boot*. Tutores de *pets*.

ABSTRACT

The article aimed to develop the SaudePET application, designed to assist pet owners in monitoring the health and daily routines of their animals. The project was developed using software analysis, modeling, and implementation methods, employing React Native for the mobile interface, Java with the Spring Boot framework for the backend, and MySQL as the database. The application enables the management of information related to vaccines, medications, and pet feeding, promoting more efficient and organized control. The results obtained showed that the integration of the chosen technologies allowed for the development of a functional, secure, and easily maintainable system. It is concluded that SaudePET has the potential to contribute to animal well-being by offering a practical and accessible tool for pet owners.

KEYWORDS: *Mobile application. Database. React Native. Animal health. Spring Boot. Pet owners.*

RESUMEN

El artículo tuvo como objetivo desarrollar la aplicación SaudePET, destinada a ayudar a los tutores en el seguimiento de la salud y la rutina diaria de sus animales. El proyecto fue desarrollado utilizando métodos de análisis, modelado e implementación de software, empleando React Native para la interfaz móvil, Java con el framework Spring Boot para el backend y MySQL como base de

[1] Estudante de graduação do curso Sistemas de Informação, da UNIARA - Universidade de Araraquara. Araraquara-SP.
[2] Graduação em Processamento de Dados pela FATEC, mestrado em Engenharia de Software pela USP, doutorado em Engenharia de Software pela USP, MBA em Gestão de Pessoas, MBA em Gestão de Projetos e MBA em Data Science e Analytics.
[3] Graduação em Engenharia de Computação pela UNIARA, especialização em Licenciatura em Informática pela FATEC, MBA em Gerenciamento de Projetos pela UNIARA, Mestrado e Doutorado em Educação Escolar pela Unesp, e especialista de TI.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

datos. La aplicación permite gestionar información relacionada con vacunas, medicamentos y alimentación de las mascotas, promoviendo un control más eficiente y organizado. Los resultados obtenidos demostraron que la integración de las tecnologías adoptadas permitió el desarrollo de un sistema funcional, seguro y de fácil mantenimiento. Se concluye que SaudePET tiene el potencial de contribuir al bienestar animal, ofreciendo una herramienta práctica y accesible para los tutores.

***PALABRAS CLAVE**: Aplicación móvil. Base de datos. React Native. Salud animal. Spring Boot. Dueños de mascotas.*

INTRODUÇÃO

Segundo uma pesquisa realizada pela ABINPET (2023), o Brasil ocupa a terceira posição mundial no *ranking* de países com maior população *pet*, ficando atrás apenas dos Estados Unidos e da China respectivamente. Estima-se que o país possua mais de 160 milhões de animais de estimação, sendo 62 milhões de cães, 42 milhões de aves, 30 milhões de gatos, cerca de 22 milhões de peixes ornamentais e 2,8 milhões de pequenos répteis e mamíferos.

A FGVCIA (2024) revela que existem em média 1,2 *smartphone* por habitante, totalizando 258 milhões de celulares inteligentes em uso no Brasil. Somando *notebooks* e *tablets*, o número de dispositivos portáteis chega a 384 milhões, ou seja, 1,8 dispositivo por habitante. Em relação aos computadores (incluindo *desktops*, *notebooks* e *tablets*), o Brasil possui 222 milhões de dispositivos, o que corresponde a um computador por habitante.

De acordo com um levantamento realizado pelo IBGE (2024), o uso de dispositivos para acessar a internet no Brasil, em 2023, aponta que a grande maioria da população, 98,8%, utiliza telefone móvel celular como o principal meio de acesso à internet. Além disso, 49,8% acessam a internet por meio da televisão, 34,2% utilizam microcomputadores e 7,6% fazem uso de *tablets*.

Estudos apontam que muitos tutores apresentam dificuldades em manter cuidados adequados com seus animais de estimação, especialmente no que diz respeito à vacinação, alimentação e acompanhamento preventivo, evidenciando lacunas no manejo de *pets* que podem comprometer sua saúde (Dias *et al.,* 2011; Silva; Vital, 2019; Souza; Godoy, 2020). Dessa forma, torna-se evidente a necessidade de ferramentas tecnológicas que auxiliem na organização desses cuidados, contribuindo para o bem-estar animal.

Este trabalho tem como objetivo desenvolver um aplicativo móvel para auxiliar tutores no gerenciamento da rotina de cuidados de seus animais de estimação, incluindo registro e acompanhamento de vacinação, alimentação, administração de medicamentos, alertas, lembretes e carteirinha digital, utilizando *React Native*, *Spring Boot* e *MySQL*.

Foi realizada pesquisa bibliográfica e, para o desenvolvimento do aplicativo, o *React Native* foi escolhido por permitir criar aplicações para *Android* e *iOS* com o mesmo código base, utilizando *JavaScript*. Em vez de programar separadamente para ambas as plataformas com

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Kotlin ou *Java* (para *Android*) e *Swift* ou *Objective-C* (para *iOS*), essa abordagem otimiza o desenvolvimento (Cunha, 2023).

O *Java*, com o *framework Spring Boot*, foi escolhido pois oferece uma série de vantagens, como criar aplicativos *Spring* autônomos, a configuração automática de bibliotecas *Spring* e de terceiros, e recursos prontos para produção, facilitando a escalabilidade e manutenção do sistema descrito no *Spring Boot*. Segundo Erickson (2024), o *MySQL*, utilizado como banco de dados, apresenta alta escalabilidade e baixo risco de perda de informações.

Um dos desafios no desenvolvimento deste aplicativo está relacionado às operações de persistência em banco de dados utilizando o padrão *CRUD* (Criar, Ler, Atualizar e Deletar), conforme apresentado por Martin (1983), garantindo organização e consistência nas operações essenciais do sistema. O uso desse padrão garante organização e consistência nas operações essenciais do sistema. No entanto, o *Spring Boot* oferece ferramentas que simplificam esse processo.

Outro desafio é a construção de uma interface intuitiva, princípio central de *UX* (*User Experience*), associado à clareza, consistência e previsibilidade, fatores que reduzem a carga cognitiva do usuário e facilitam a navegação (Norman, 2013). No contexto deste aplicativo, esses elementos são essenciais para permitir que tutores registrem informações, consultem vacinas, acessem a carteirinha digital e recebam lembretes de forma rápida e sem dificuldades.

REVISÃO BIBLIOGRÁFICA

Esta seção apresenta as principais tecnologias utilizadas no desenvolvimento do aplicativo, abordando seus fundamentos teóricos, características e justificando sua aplicação no contexto da solução proposta.

JavaScript

O *JavaScript* é uma linguagem de programação interpretada, leve e orientada a eventos, amplamente adotada no desenvolvimento *web*. De acordo com Mozilla (2025), trata-se de uma linguagem de alto nível, dinâmica e suportada pela maioria dos navegadores modernos, possibilitando a construção de interfaces mais interativas, responsivas e acessíveis. Sua sintaxe flexível, aliada ao suporte a múltiplos paradigmas, incluindo programação funcional e orientada a objetos, torna o *JavaScript* adequado para diversos tipos de aplicações.

Flanagan (2013) descreve o *JavaScript* como essencial para aplicações que exigem manipulação dinâmica de elementos na página, integração com *API*s e tratamento de eventos em tempo real. No contexto deste projeto, o *JavaScript* constitui a base para o uso do *React Native*, possibilitando a criação da interface do aplicativo com maior produtividade e compatibilidade entre plataformas.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Java

O *Java* é uma linguagem de programação compilada para *bytecode,* ou seja, código objeto intermediário gerado a partir do código-fonte *Java*, interpretado pela *Java Virtual Machine (JVM).* Segundo (Deitel, P.; Deitel, H., 2010), esse mecanismo possibilita a portabilidade entre diferentes sistemas operacionais, uma vez que o mesmo código pode ser executado em qualquer ambiente que possua uma *JVM* compatível. Essa característica torna o *Java* uma das linguagens mais utilizadas para sistemas corporativos e aplicações que demandam robustez e padronização.

Além disso, seu modelo orientado a objetos proporciona modularidade, organização e manutenção facilitada. No contexto deste projeto, o *Java* foi empregado para a implementação do *backend* em conjunto com o *Spring Boot*, garantindo estabilidade, segurança e integração eficiente com bibliotecas voltadas à persistência de dados.

React Native

O *React Native* é um *framework* proposto inicialmente pelo Facebook e atualmente mantido pela Meta. Ele permite a criação de aplicativos móveis utilizando *JavaScript* e componentes que são convertidos em elementos nativos durante a execução. Segundo Cunha (2023), essa abordagem viabiliza o desenvolvimento multiplataforma, reduzindo o retrabalho e garantindo consistência visual entre dispositivos com sistemas operacionais distintos.

Conforme a documentação oficial da Meta (2025), o *React Native* possibilita o uso de *API*s nativas do dispositivo, como câmera, armazenamento local e sensores, por meio de módulos próprios ou bibliotecas externas. No desenvolvimento deste aplicativo, o *React Native* foi escolhido devido à sua eficiência na renderização de componentes e pela facilidade de integração com serviços externos, atendendo às necessidades de interface móvel moderna e responsiva.

Visual Studio Code

O *Visual Studio Code* é um editor de código multiplataforma focado em produtividade, extensibilidade e suporte a diversas linguagens. Conforme Edson (2016), o editor oferece recursos como depuração integrada, controle de versão por *Git*, realce de sintaxe e suporte aprimorado para tecnologias *web*. Segundo a documentação oficial da Microsoft (2025), sua arquitetura extensível permite instalar plugins compatíveis com *React Native*, facilitando o desenvolvimento da camada de *interface* do aplicativo.

Sua utilização neste projeto se justifica pela leveza, suporte a múltiplos ambientes e ferramentas voltadas para *JavaScript*, proporcionando um fluxo de desenvolvimento mais eficiente.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Spring Boot

O *Spring Boot* é um *framework* pertencente ao ecossistema *Spring*, projetado para simplificar o desenvolvimento de aplicações *Java*. A documentação oficial (Spring.io, 2025) destaca que o *Spring Boot* reduz configurações manuais por meio de convenções, *starters* e integração automatizada com bibliotecas essenciais, como *Spring Web* e *Spring Data JPA*.

Sua arquitetura facilita o desenvolvimento modular e o gerenciamento de dependências, permitindo um ambiente de execução otimizado. Neste projeto, o *Spring Boot* foi empregado para estruturar o *backend*, oferecendo suporte para criação de *API*s *REST*, controle de rotas e persistência de dados com segurança e eficiência.

IntelliJ Idea

O *IntelliJ Idea* é uma IDE consolidada para desenvolvimento em *Java* e *Kotlin*. De acordo com JetBrains (2025), a ferramenta fornece análise inteligente de código, refatoração automática, inspeções avançadas, gerenciamento de dependências e integração com ferramentas de *build* como *Maven* e *Gradle*. Essas funcionalidades auxiliam diretamente no desenvolvimento de sistemas construídos com *Spring Boot*, tornando o fluxo de codificação mais seguro, organizado e produtivo.

MySQL

O *MySQL* é um Sistema de Gerenciamento de Banco de Dados Relacional amplamente utilizado em sistemas *web* e corporativos. Segundo Date (2004), bancos de dados relacionais estruturam dados em tabelas interconectadas e manipuladas por meio da linguagem *SQL*, garantindo organização, integridade e consistência das informações. O *MySQL* segue esse modelo, oferecendo suporte a transações, controle de concorrência e segurança.

Conforme Erickson (2024), o *MySQL* destaca-se por sua estabilidade, desempenho e confiabilidade, atendendo tanto aplicações de pequeno porte quanto sistemas complexos. No projeto, o banco de dados é responsável por armazenar informações de usuários, agendamentos e registros dos animais de forma segura.

MÉTODOS

A metodologia adotada para o desenvolvimento do SaudePET segue um processo estruturado de engenharia de *software,* contemplando definição de requisitos, arquitetura, modelagem, implementação e testes. O objetivo desta seção é apresentar de forma clara as etapas, métodos, tecnologias e padrões utilizados na construção do aplicativo.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Método de Desenvolvimento

O desenvolvimento do *software* foi conduzido utilizando o modelo incremental, permitindo que as funcionalidades fossem implementadas e validadas em pequenos ciclos. Cada ciclo contemplou:

- Levantamento e refinamento dos requisitos.
- Modelagem das funcionalidades previstas no incremento.
- Implementação do *frontend* e *backend*.
- Integração entre as camadas.
- Testes unitários e de integração.
- Correções e ajustes antes do próximo ciclo.

O modelo incremental foi escolhido por proporcionar flexibilidade, rápida evolução do sistema e melhor controle sobre cada funcionalidade adicionada.

Arquitetura do *Software*

Para o SaudePET foi adotada a arquitetura em camadas, amplamente recomendada em sistemas corporativos e de médio porte por promover modularidade, organização e facilidade de manutenção. As camadas utilizadas foram:

- Camada de Apresentação (*frontend*): Desenvolvida em *React Native*, responsável pela interface com o usuário e pela interação direta com as funcionalidades disponíveis.
- Camada de Controle (*controllers*): Responsável por receber requisições *HTTP* provenientes do *frontend* e direcioná-las para as regras de negócio apropriadas.
- Camada de Serviço (*services*): Centraliza a lógica de negócio, aplicando regras, validações e tratando exceções antes das operações de persistência.
- Camada de Persistência (*repository*): Implementada por meio do *Spring Data JPA*, seguindo o *Repository Pattern* (Fowler, 2002), garantindo organização e desacoplamento entre lógica de negócio e operações de acesso ao banco.
- Camada de Banco de Dados: Utiliza o *MySQL* para armazenamento seguro e estruturado das informações referentes a tutores, animais, vacinas, medicamentos e rações.

A comunicação entre *frontend* e *backend* segue o estilo arquitetural *REST*, garantindo padronização nas requisições e respostas da *API*.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Padrões e Tecnologias Adotados

Para assegurar qualidade, segurança e organização, foram incorporados os seguintes padrões:

- *CRUD*, conforme Martin (1983), para operações de criação, leitura, atualização e exclusão.
- JPA (*Java* Persistence *API*) como especificação de persistência.
- *Hibernate* como implementação padrão da JPA.
- *BCrypt* para criptografia de senhas, garantindo proteção dos dados de autenticação.
- Principais práticas de *UX* para garantir intuitividade na interface (Norman, 2013).

Ferramentas Utilizadas

O desenvolvimento contou com ferramentas de apoio essenciais:

- *Visual Studio Code* para implementação do *frontend*.
- *IntelliJ Idea* para desenvolvimento do *backend*.
- *MySQL Workbench* para modelagem e manipulação do banco de dados.

Processo de Modelagem

Foram construídos dois principais diagramas:

- Diagrama de Classes: Representa a estrutura do sistema, definindo classes, atributos privados, métodos e relações. Serviu como base para implementar entidades e estruturas de persistência.
- Diagrama de Caso de Uso: Representa as interações entre o usuário (tutor) e as funcionalidades do sistema, permitindo visualizar fluxos principais do aplicativo.

Os diagramas garantiram alinhamento entre requisitos, regras de negócio e implementação.

Fluxo Geral de Execução

O fluxo padrão do sistema segue o modelo cliente-servidor:

Quando o tutor realiza uma ação no *frontend*, o aplicativo envia uma requisição *HTTP* ao *controller*. Este encaminha os dados para a camada de serviço, que aplica validações e aciona a camada de persistência (*repository*). Os dados são registrados ou consultados no banco *MySQL*, e uma resposta estruturada retorna ao *frontend*, atualizando a interface do usuário.

Esse fluxo se repete para todas as entidades: usuários, animais, vacinas, medicamentos e rações.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

DESENVOLVIMENTO

O objetivo desta seção é descrever, as etapas essenciais para o desenvolvimento do aplicativo proposto, abrangendo desde a concepção inicial até a entrega do produto final.

Requisitos do Sistema

Os requisitos definem as funcionalidades essenciais para o funcionamento da aplicação, bem como critérios de qualidade esperados no *software*.

Requisitos Funcionais

- O sistema deve permitir o cadastro de tutores.
- O tutor deve poder cadastrar um ou mais animais.
- O sistema deve registrar vacinas, medicamentos e rações.
- O sistema deve permitir editar e excluir registros.
- O sistema deve exibir a carteirinha digital do animal.
- O sistema deve permitir *login* com autenticação segura.
- O *frontend* deve se comunicar com o *backend* via requisições *HTTP*.
- O *backend* deve disponibilizar *APIs REST* para todas as operações *CRUD*.

Requisitos Não Funcionais

- O sistema deve utilizar criptografia de senha (*BCrypt*).
- O aplicativo deve apresentar interface intuitiva (Norman, 2013).
- O banco de dados deve garantir integridade e consistência dos dados.
- A arquitetura deve ser modular, permitindo manutenção facilitada.
- As respostas da *API* devem atender a padrões *REST*.

Modelagem

A modelagem do sistema teve início com a elaboração do diagrama de classes, ferramenta fundamental para representar de forma estruturada os principais componentes da aplicação ainda nas primeiras fases do desenvolvimento. O diagrama permite identificar as classes do sistema, seus atributos, métodos e os relacionamentos entre elas, proporcionando uma compreensão clara da organização interna da aplicação.

Além de evidenciar os elementos que compõem o sistema, o diagrama facilita o entendimento da lógica geral de funcionamento, permitindo visualizar como as entidades se conectam entre si. Segundo Melo (2010), essa representação gráfica contribui para a precisão da

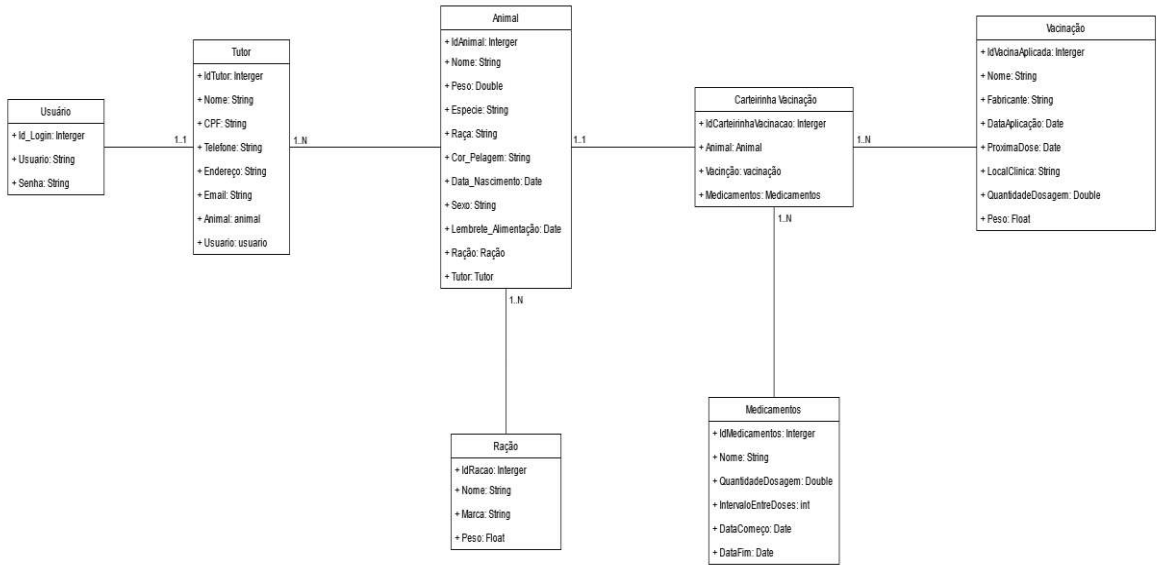
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

modelagem ao demonstrar, de maneira objetiva, como objetos do mundo real são transformados em entidades do *software.*

No contexto deste trabalho, o diagrama de classes (Figura 1) foi fundamental para definir a estrutura dos dados que seriam persistidos no banco de dados. Ele serviu como base para a implementação das entidades no *backend*, garantindo alinhamento entre projeto e desenvolvimento.

Figura 1. Diagrama de Classes do Sistema SaudePET



Fonte: Elaborado pelo autor (2025)

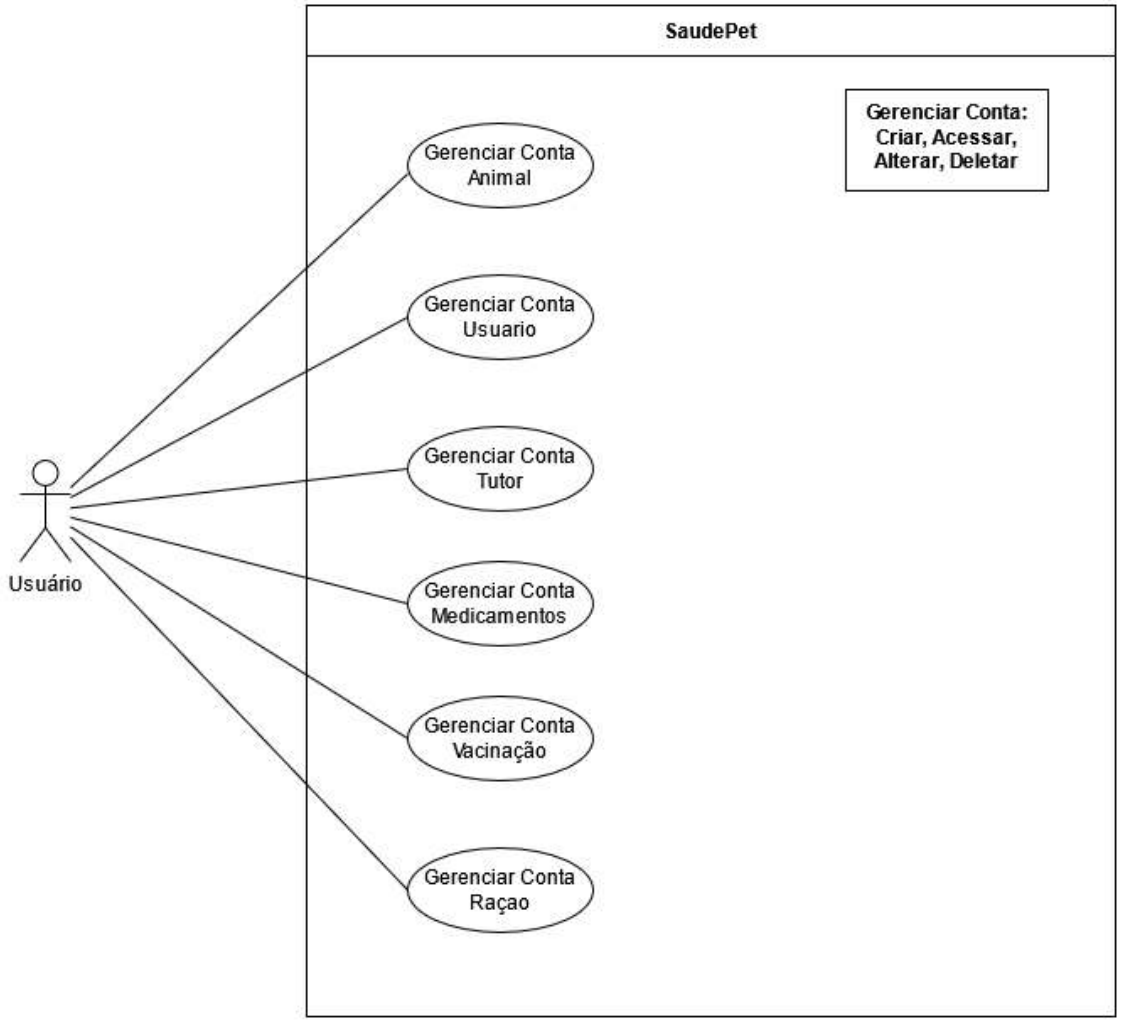
Também foi desenvolvido o diagrama de caso de uso (Figura 2), responsável por representar graficamente as interações entre os atores e as funcionalidades disponíveis no sistema. Conforme descrito por Melo (2010), esse tipo de diagrama se concentra no comportamento externo do sistema, apresentando o fluxo das ações realizadas pelo usuário.

No caso do SaudePET, o ator principal é o tutor, que interage com funcionalidades como cadastro, consulta, alteração e exclusão de informações sobre animais, vacinas, medicamentos e rações. O diagrama demonstra de forma clara como o usuário realiza o gerenciamento completo dos dados, reforçando a proposta do sistema de centralizar informações relacionadas à saúde e bem-estar dos *pets*.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Figura 2. Diagrama de caso de uso do Sistema SaudePET



Fonte: Elaborado pelo autor (2025)

Desenvolvimento do *Frontend*

O *frontend* é responsável por apresentar as informações e permitir a interação direta do usuário com o sistema. Para essa camada, foi utilizado o *framework React Native*, que possibilita o desenvolvimento de aplicativos móveis nativos para *Android* e *iOS* a partir de um único código-base escrito em *JavaScript*. A escolha da tecnologia se deu pela sua alta performance, compatibilidade entre plataformas e grande ecossistema de bibliotecas.

O desenvolvimento foi realizado no *Visual Studio Code*, escolhido devido à sua leveza, extensibilidade e integração eficiente com as ferramentas utilizadas no projeto.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

As telas da aplicação foram projetadas buscando simplicidade, clareza e responsividade, a fim de proporcionar uma navegação intuitiva ao usuário.

A comunicação entre o *frontend* e o *backend* ocorre por meio de requisições *HTTP* enviadas aos *endpoints* implementados no servidor. Essas requisições permitem realizar operações como criação, leitura, atualização e exclusão de dados, garantindo que as informações exibidas na interface estejam sempre sincronizadas com o banco de dados.

A seguir são apresentadas as principais telas desenvolvidas:

Tela de *Login*

A tela de *login* (Figura 3) permite que o tutor insira suas credenciais para acessar o sistema. Após o envio dos dados, o *backend* valida as informações e, em caso de sucesso, gera um *token* de autenticação que habilita o acesso seguro às demais funcionalidades.

Figura 3. Tela de *login* do usuário na aplicação *mobile*



Fonte: Elaborado pelo autor (2025)

Tela de Listagem de Animais

Após a autenticação, o tutor é direcionado para a tela exibida na Figura 4, onde são apresentados todos os animais previamente cadastrados. Cada card apresenta dados resumidos como nome, espécie e raça. Ao selecionar um animal, o usuário pode visualizar detalhes completos, editar informações e registrar novas vacinas, medicamentos e rações.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 4. Tela de dados do tutor



Fonte: Elaborado pelo autor (2025)

Carteirinha Digital

A carteirinha de vacinação (Figura 5) apresenta o histórico de vacinas e medicamentos aplicados ao animal. Além disso, permite acessar dados completos do *pet*, como espécie, raça, idade e peso, possibilitando um acompanhamento detalhado do perfil do animal.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Figura 5. Tela da carteirinha de vacinação do animal



Fonte: Elaborado pelo autor (2025)

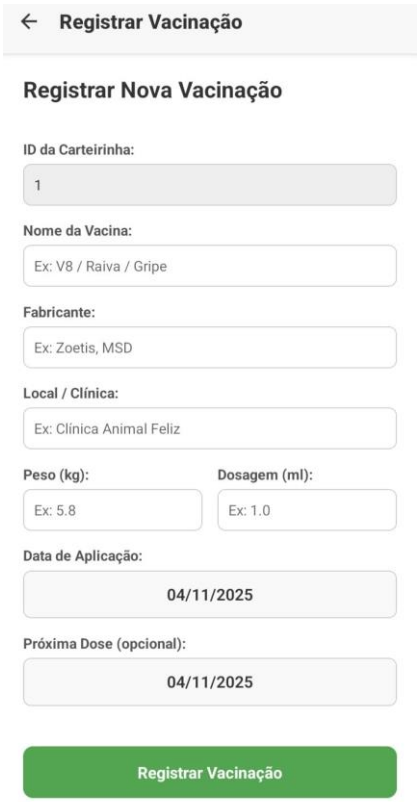
Cadastro de Vacinas

A tela de cadastro de vacinas (Figura 6) permite inserir informações como nome da vacina, data de aplicação e lote. Os dados enviados são processados pelo *backend* e imediatamente exibidos na carteirinha do animal.

.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 6. Tela de cadastro de vacinação



Fonte: Elaborado pelo autor (2025)

Telas estruturadas de maneira semelhante foram desenvolvidas para o cadastro de medicamentos e ração, mantendo consistência visual e funcionalidade.

Desenvolvimento do *Backend*

A implementação do *backend* foi realizada com a linguagem *Java* utilizando o *framework Spring Boot*, adotado pela sua robustez, modularidade e suporte nativo ao desenvolvimento de *APIs REST*. O *backend* foi construído seguindo a arquitetura em camadas, dividindo o sistema em entidades, repositórios, serviços e controladores.

Modelagem das Entidades

A criação das entidades teve como base o diagrama de classes apresentado anteriormente. Cada classe foi mapeada para o banco de dados utilizando anotações do JPA, garantindo persistência e integridade dos dados.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Camada *Repository*

A camada *repository* (Figura 7) foi desenvolvida com suporte do *Spring Data JPA*, que simplifica operações de persistência, como salvar, atualizar, consultar e deletar registros. Ela abstrai a necessidade de escrever consultas SQL manualmente, tornando o código mais limpo e eficiente.

Figura 7. Estrutura da camada *Repository*

```
@Repository  2 usages
public interface UsuarioRepository extends JpaRepository<Usuario, Integer> {
    //Método para buscar usuário por nome de usuário e senha (login)
    @Query(value = "SELECT * FROM usuarios WHERE BINARY usuario = ?1", nativeQuery = true)  1 usage
    Optional<Usuario> findByUsuario(String usuario);
}
```

Fonte: Elaborado pelo autor (2025)

Camada *Service*

A camada *service* (Figura 8) centraliza a lógica de negócio da aplicação. Nela são aplicadas regras como validação de dados e criptografia de senha utilizando *BCrypt* antes do salvamento no banco. Essa abordagem aumenta significativamente a segurança do sistema.

Figura 8. Estrutura da camada *Service*

```
@Service  2 usages
public class UsuarioService {

    @Autowired  5 usages
    private UsuarioRepository usuarioRepository;

    @Autowired  2 usages
    private PasswordEncoder passwordEncoder;

    // Cria usuário com senha criptografada
    public Usuario save(Usuario usuario) {
        // Evita recriptografar senha se for uma atualização
        if (usuario.getSenha() != null && !usuario.getSenha().startsWith("$2a$")) {
            String senhaCriptografada = passwordEncoder.encode(usuario.getSenha());
            usuario.setSenha(senhaCriptografada);
        }
        return usuarioRepository.save(usuario);
    }

    // Login: busca pelo nome de usuário e compara a senha
    public Optional<Usuario> login(String usuario, String senha) {  1 usage
        Optional<Usuario> usuarioOpt = usuarioRepository.findByUsuario(usuario);

        if (usuarioOpt.isPresent()) {
            Usuario encontrado = usuarioOpt.get();

            if (encontrado.getUsuario().equals(usuario) && passwordEncoder.matches(senha, encontrado.getSenha())) {
                return Optional.of(encontrado);
            }
        }

        return Optional.empty();
    }

    public List<Usuario> findAll() { return usuarioRepository.findAll(); }

    public Optional<Usuario> findById(Integer id) { return usuarioRepository.findById(id); }

    public void deleteById(Integer id) { usuarioRepository.deleteById(id); }
}
```

Fonte: Elaborado pelo autor (2025)

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Camada *Controller*

A camada *controller* (Figura 9) é responsável por definir as rotas da *API* e intermediar a comunicação entre *frontend* e *backend*. Nessa camada, as requisições recebidas são processadas e repassadas aos serviços, que executam a lógica de negócio e retornam uma resposta estruturada.

Figura 9. Estrutura da camada *Controller*

```
@RestController  no usages
public class UsuarioController {

    @Autowired  5 usages
    private JwtTokenProvider jwtTokenValidator;

    @Autowired  7 usages
    private UsuarioService usuarioService;

    @PostMapping(Constant.API_LOGIN)  no usages
    public ResponseEntity<?> login(@RequestBody Usuario usuario) {
        Optional<Usuario> usuarioOpt = usuarioService.login(usuario.getUsuario(), usuario.getSenha());

        if (usuarioOpt.isEmpty()) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                    .body(Map.of( k1: "message",  v1: "Usuário ou senha inválidos"));
        }

        Usuario usuarioEncontrado = usuarioOpt.get();

        String token = jwtTokenValidator.generateToken(usuarioEncontrado.getUsuario());

        Integer idTutor = (usuarioEncontrado.getTutor() != null)
                ? usuarioEncontrado.getTutor().getIdTutor()
                : null;

        Map<String, Object> response = new HashMap<>();
        response.put("usuario", usuarioEncontrado.getUsuario());
        response.put("token", token);
        response.put("idUsuario", usuarioEncontrado.getIdUsuario());
        response.put("idTutor", idTutor);

        return ResponseEntity.ok(response);
    }
```

Fonte: Elaborado pelo autor (2025)

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

Configuração do Banco de Dados

No arquivo application.properties foram definidas as configurações de conexão com o banco de dados *MySQL*, incluindo URL, credenciais, *driver* e propriedades do JPA necessárias para o correto funcionamento da persistência.

RESULTADOS

O desenvolvimento do SaudePET permitiu materializar uma solução tecnológica voltada ao apoio de tutores no acompanhamento da saúde e das rotinas de seus animais de estimação. A execução do projeto possibilitou a aplicação prática dos conhecimentos adquiridos na área de Sistemas de Informação, abrangendo desde a fase de análise e modelagem até a implementação e integração das camadas que compõem o sistema, resultando em um aplicativo funcional e operacional.

As tecnologias adotadas demonstraram-se adequadas para o escopo do projeto. O *React Native* possibilitou o desenvolvimento de uma interface *mobile* responsiva, moderna e compatível com diferentes dispositivos. No *backend*, a utilização de *Java* com o *framework Spring Boot* garantiu robustez, segurança, padronização e modularidade, especialmente na construção das *API*s responsáveis pela comunicação com o *frontend*. A integração com o banco de dados *MySQL* assegurou persistência confiável das informações, permitindo armazenar dados de tutores, animais, vacinas, medicamentos e rações de forma estruturada.

A interação entre as tecnologias empregadas evidenciou uma comunicação eficiente entre as camadas da aplicação. As requisições enviadas pela interface foram corretamente processadas pelos controladores, encaminhadas para a camada de serviços, tratadas conforme as regras de negócio e persistidas no banco de dados. Esse fluxo validou a arquitetura projetada, garantindo consistência e integridade das operações realizadas pelo usuário.

As Figuras 4 e 5 ilustram duas telas centrais do sistema: a tela de dados do tutor e a tela de detalhes do animal. Na tela de dados do tutor (Figura 4), o usuário pode visualizar suas informações pessoais, acessar a lista de animais cadastrados e adicionar novos *pets* ao perfil. Já a tela de detalhes do animal (Figura 5) apresenta informações essenciais, como nome, espécie, raça, idade e peso, além do acesso às carteirinhas digitais de vacinas e medicamentos. Essas telas demonstram a clareza e a organização da interface do SaudePET, refletindo diretamente os princípios de usabilidade considerados no desenvolvimento.

Com base nos testes realizados e na análise das funcionalidades implementadas, os resultados demonstram que O SaudePET apresenta potencial para centralizar informações relevantes sobre o cuidado animal e oferece ao tutor uma plataforma intuitiva para gerenciamento da saúde de seus *pets*. Apesar de ainda não ter sido validado por pesquisa de campo, o *software* desenvolvido demonstra capacidade de contribuir para o bem-estar animal.

ISSN: 2675-6218 - RECIMA21
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

CONSIDERAÇÕES

O desenvolvimento do SaudePET permitiu alcançar plenamente o objetivo proposto, apresentando uma solução tecnológica funcional e eficiente para o gerenciamento de vacinas, medicamentos e hábitos alimentares de animais de estimação. A aplicação mostrou-se segura, prática e acessível, proporcionando aos tutores maior organização e controle sobre os cuidados essenciais destinados aos seus *pets*.

Ao centralizar informações importantes e disponibilizar recursos que facilitam o acompanhamento da saúde animal, o SaudePET demonstrou potencial para contribuir com o bem-estar dos animais, reduzindo falhas na administração de vacinas, medicamentos e demais rotinas. Além disso, a arquitetura adotada, aliada às tecnologias empregadas, garantiu um sistema robusto, escalável e de fácil manutenção, alinhado às boas práticas de desenvolvimento de *software*.

Como perspectiva de evolução do projeto, propõe-se a implementação de um sistema de notificações automáticas, responsável por alertar o tutor sobre datas de vacinação, administração de medicamentos e outras atividades essenciais. Esse aprimoramento ampliará o alcance e a eficácia do SaudePET, tornando-o uma ferramenta ainda mais completa para o apoio à saúde e à qualidade de vida dos animais de estimação.

REFERÊNCIAS

ABINPET. **O Setor Pet e Seus Números**. [S. l.]: Abinpet, 2023. Disponível em: https://abinpet.org.br/informacoes-gerais-do-setor/. Acesso em: 01 abr. 2025.

BCRYPT. ***BCrypt* Password Hashing Function.** Disponível em: https://BCrypt.sourceforge.net/. Acesso em: 01 abr. 2025.

CUNHA, A. O. **React Native:** o que é e tudo sobre o *Framework*. [*S. l.: s. n.*], 2023. Disponível em: https://www.alura.com.br/artigos/*React*-native. Acesso em: 01 abr. 2025.

DATE, C. J. **Introdução a sistemas de banco de dados.** 7. reimpr. Rio de Janeiro: Editora Campus, 2004.

DEITEL, Paul; DEITEL, Harvey. **Java:** como programar. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

DIAS, R. A. *et al.* Fatores associados à falta de vacinação de cães e gatos contra a raiva no Brasil. **Revista de Saúde Pública**, São Paulo, v. 45, n. 6, p. 1112-1118, 2011.

EDSON. **Introdução ao Visual Studio Code**. [*S. l.: s. n.*], 2016. Disponível em: https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418. Acesso em: 01 abr. 2025

ERICKSON, J. **MySQL:** Entendendo o que é e como é usado. [*S. l.: s. n.*], 2024. Disponível em: https://www.oracle.com/be/*MySQL*/what-is-*MySQL*/. Acesso em: 01 abr. 2025.

ISSN: 2675-6218 - RECIMA21

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

FGV. Pesquisa revela que Brasil tem 480 milhões de dispositivos digitais em uso, sendo 2,2 por habitante. **FGV Notícias**, 2024. Disponível em: https://portal.fgv.br/noticias/pesquisa-revela-brasil-tem-480-milhoes-dispositivos-digitais-uso-sendo-22-habitante. Acesso em: 01 abr.2025.

FLANAGAN, David. **JavaScript:** o guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.

FOWLER, Martin. **Patterns of Enterprise Application Architecture**. Boston: Addison-Wesley, 2002.

HIBERNATE. **Hibernate ORM Documentation**. [S. l.]: Hibernate, s. d. Disponível em: https://*Hibernate*.org/orm/. Acesso em: 10 jun. 2025.

JAVA.COM. **O que é o Java?** [S. l.]: Java.com, s. d. Disponível em: https://www.*Java*.com/pt-BR/download/help/whatis_*Java*.html. Acesso em: 10 jun. 2025.

JETBRAINS. **IntelliJ IDEA – IDE para desenvolvimento profissional em *Java* e *Kotlin***. [S. l.]: JETBRAINS, 2025. Disponível em: https://www.jetbrains.com/pt-br/idea/. Acesso em: 10 jun. 2025.

MARTIN, James. **Managing the Data-base Environment.** Englewood Cliffs: Prentice-Hall, 1983.

MELO, Ana Cristina, **Desenvolvendo aplicações com UML 2.2:** do conceitual à implementação. 3. ed. Rio de Janeiro: Brasport, 2010.

META. **React Native Documentation**. [S. l.], Meta, 2025. Disponível em: https://*React*native.dev/. Acesso em: 10 jun. 2025.

MICROSOFT. **Visual Studio Code Documentation**. [S. l.]: Microsof, 2025. Disponível em: https://code.visualstudio.com/docs. Acesso em: 10 jun. 2025.

MOZILLA. **Sobre *JavaScript***. [S. l.]: MOZILLA, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/conflicting/Web/*JavaScript*. Acesso em: 01 abr. 2025.

NERY, C. Em 2023, 88,0% das pessoas com 10 anos ou mais utilizaram internet. **Agencia de Noticias IBGE**, 16 ago. 2024. Disponível em: https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/41026-em-2023-87-2-das-pessoas-com-10-anos-ou-mais-utilizaram-internet. Acesso em: 01 abr. 2025.

NORMAN, Donald A. **The Design of Everyday Things - Revised and Expanded Edition**. Cambridge: MIT Press, 2013.

SILVA, J.; VITAL, A. Perfil dos tutores e práticas de cuidados de cães e gatos. **Revista Ciência Animal**, Fortaleza, v. 29, n. 2, p. 45-54, 2019.

SOUZA, A. C.; GODOY, L. Avaliação dos cuidados de tutores com cães domiciliados. **Pesquisa Veterinária Brasileira**, Seropédica, v. 40, n. 5, p. 345-352, 2020.

SPRING.IO. **Why Spring**. [S. l.]: SPRING.IO, 2025. Disponível em: https://spring.io/why-spring. Acesso em: 10 jun. 2025.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.