



INSTITUTO FEDERAL

Sertão Pernambucano

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO
PERNAMBUCANO**

COORDENAÇÃO DO CURSO DE SISTEMAS PARA INTERNET

CAMPUS SALGUEIRO

ARTHUR FERREIRA DA SILVA DUARTE

**PROPOSTA DE DESENVOLVIMENTO DO SISTEMA DE AGENDAMENTO DE
VISITAS DO MUSEU DE CIÊNCIA PROFESSOR ANTÔNIO CARNEIRO**

SALGUEIRO-PE

2023

ARTHUR FERREIRA DA SILVA DUARTE

**PROPOSTA DE DESENVOLVIMENTO DO SISTEMA DE AGENDAMENTO DE
VISITAS DO MUSEU DE CIÊNCIA PROFESSOR ANTÔNIO CARNEIRO**

Trabalho de Conclusão de Curso apresentado a Coordenação do curso de Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, campus Salgueiro, como requisito parcial à obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador(a): Prof. Esp. Francisco Junio da Silva Fernandes.

SALGUEIRO-PE

2023

Dados Internacionais de Catalogação na Publicação (CIP)

D812 Duarte, Arthur Ferreira da Silva.

Proposta de desenvolvimento do sistema de agendamento de visitas do museu de ciência professor Antônio Carneiro / Arthur Ferreira da Silva Duarte. - Salgueiro, 2023.

55 f. : il.

Trabalho de Conclusão de Curso (Sistemas para Internet) -Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, Campus Salgueiro, 2023.
Orientação: Prof. Esp. Francisco Junio da Silva Fernandes.

1. Desenvolvimento de software. 2. Produto de Software. 3. Sistema Web. 4. Arquitetura REST. I. Título.

CDD 005.2



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO
Salgueiro - Código INEP: 26548747
Rod Br 232, Km 508, S/N, CEP 56000000, Salgueiro (PE)
CNPJ: 10.830.301/0005-20 - Telefone: 87 3421-0050

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Na presente data realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulada **Proposta de Desenvolvimento do Sistema de Agendamento de Visitas do Museu de Ciência Professor Antônio Carneiro**, sob orientação de Francisco Junio da Silva Fernandes, apresentada pelo aluno **Arthur Ferreira da Silva Duarte (201814010003)** do Curso **Tecnologia em Sistemas para Internet (Salgueiro)**. Os trabalhos foram iniciados às 19h pelo Professor presidente da banca examinadora, constituída pelos seguintes membros:

- **Francisco Junio da Silva Fernandes** (Presidente)
- **Heraldo Gonçalves Lima Junior** (Examinador Interno)
- **Orlando Silva de Oliveira** (Examinador Interno)

A banca examinadora, tendo terminado a apresentação do conteúdo do Trabalho de Conclusão de Curso, passou à arguição do candidato. Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo aluno, tendo sido atribuído o seguinte resultado:

Aprovado

Reprovado

Nota (quando exigido): 75

Observação / Apreciações:

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu **Francisco Junio da Silva Fernandes** lavrei a presente ata que assino juntamente com os demais membros da banca examinadora.

Documento assinado digitalmente
ORLANDO SILVA DE OLIVEIRA
Data: 30/08/2023 20:22:25-0300
Verifique em <https://validar.it.gov.br>

Orlando Silva de Oliveira

Heraldo Gonçalves Lima Junior

Documento assinado digitalmente
HERALDO GONCALVES LIMA JUNIOR
Data: 29/08/2023 18:43:19-0300
Verifique em <https://validar.it.gov.br>

Salgueiro / PE, 10/07/2023

Assinado digitalmente por Francisco Junio da Silva Fernandes00372780377
"00372780377" - Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano - IFCSP/PE - CEP: 56000000 - Salgueiro (PE)
Assinatura: Francisco Junio da Silva Fernandes
CPF: 00372780377

Francisco Junio da Silva Fernandes

Aos meus pais, Regilane e Moacy, por me amarem a cada dia.

AGRADECIMENTOS

Ao meu orientador Francisco Junio, cuja inteligência me conduziu até aqui.

Aos professores participantes da banca examinadora, Heraldo e Orlando, pelas valiosas colaborações e sugestões.

Aos amigos de turma, Leidiane, Veronica, Matheus, Danilo e Fabrício, que em meio a risadas, refeições improvisadas, trabalhos inacabados, e todos os altos e baixos, caminharam comigo nesta jornada.

E um agradecimento especial a minha família, que por vezes precisei me ausentar de sua companhia e permaneceram sendo pacientes. Obrigado a todos!

RESUMO

Após o início da pandemia de SARS-CoV-2 em 2020, ocorreu uma ampliação significativa das aplicações disponíveis na internet. Essa evolução, impulsionada pela necessidade de garantir a segurança das pessoas e evitar a propagação do vírus, possibilitou a transição de atividades que anteriormente eram realizadas de forma presencial. Empresas, públicas e privadas, tiveram que se adaptar a essa nova realidade. Considerando esse cenário, esse trabalho propõe a construção de um produto de software Web para o Museu de Ciências Professor Antônio Carneiro (MCPAC), localizado no Instituto Federal do Sertão Pernambucano (IFSertaoPE) - Campus Salgueiro, que possibilita o agendamento online de visitas ao museu. A aplicação será composta por uma API REST e um banco de dados.

Palavras-chave: Produto de Software, Sistema Web, Arquitetura REST.

ABSTRACT

Following the start of the SARS-CoV-2 pandemic in 2020, there was a significant expansion of the applications available on the internet. This evolution, driven by the need to ensure people's safety and prevent the spread of the virus, made it possible to transition from activities that were previously carried out in person. Companies, public and private, had to adapt to this new reality. Considering this scenario, this work proposes the construction of a Web software product for the Museu de Ciências Professor Antônio Carneiro (MCPAC), located at Instituto Federal do Sertão Pernambucano (IFSertaoPE) - Campus Salgueiro, which allows online scheduling of visits to the museum. The application will consist of a REST API and a database.

Keywords: Software Product, Web System, REST Architecture.

LISTA DE FIGURAS

Figura 1 - Visão geral do Spring Framework.....	19
Figura 2 - Arquitetura do Spring Boot.....	20
Figura 3 - Fluxo básico da requisição em aplicativos Spring Boot.....	21
Figura 4 - Persistência de dados com o Hibernate.....	25
Figura 5 - Fluxo básico da requisição da aplicação proposta.....	26
Figura 6 - Fluxo básico da requisição em aplicativos Spring Boot.....	28
Figura 7 - Chave-valor de um objeto JSON.....	28
Figura 8 - Quadro de tarefas Kanban Trello.....	32
Figura 9 - Quadro de tarefas do projeto.....	33
Figura 10 - Diagrama de casos de uso.....	35
Figura 11 - Arquivo MuseuCienciasApplication.java.....	36
Figura 12 - Arquivo AdmMuseuController.java.....	37
Figura 13 - Métodos da classe AdmMuseuController.....	37
Figura 14 - Classe VisitanteController.....	38
Figura 15 - Atributos da classe AgendamentoController.....	39
Figura 16 - Métodos da Classe AgendamentoController.....	39
Figura 17 - Arquivo ResourceNotFoundException.java.....	40
Figura 18 - Atributos da classe AdmMuseu do package domain.....	41
Figura 19 - Métodos Getters e Setters da class AdmMuseu do package domain.....	42
Figura 20 - Atributos da classe Visitante do package domain.....	43
Figura 21 - Métodos Getters e Setters da classe Visitante do package domain.....	43
Figura 22 - Atributos da classe Agendamento do package domain.....	44
Figura 23 - Métodos Getters e Setters da classe Agendamento do package domain.....	45
Figura 24 - Arquivo Post.java do package model.....	46

LISTA DE QUADROS

Quadro 1- Anotações do Spring Framework.....	22
Quadro 2 - Fundamentos e práticas do Kanban.....	32
Quadro 3 - Requisitos funcionais.....	33
Quadro 4 - Requisitos não funcionais.....	34
Quadro 5 - História de usuário.....	35

LISTA DE ABREVIATURAS E SIGLAS

API –	Application Programming Interface
CRUD –	Create, Read, Updated, Delete
DI –	Dependency Injection
HTTP –	Hypertext Transfer Protocol
IDE –	Integrated Development Environment
IFSertãoPE –	Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano
IoC –	Inversion of Control
Java EE –	Java Enterprise Edition
JSON –	JavaScript Object Notation
MCPAC –	Museu de Ciências Professor Antônio Carneiro
MVC –	Model, View, Controller
ORM –	Object Relational Mapping
REST –	Representational state transfer
RF –	Requisito Funcional
RNF –	Requisito não Funcional
STS –	Spring Tool Suite
SQL –	Structured Query Language
URL –	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO.....	14
2 FUNDAMENTOS TÉCNICOS.....	16
2.1 MUSEU DE CIÊNCIAS PROFESSOR ANTÔNIO CARNEIRO.....	16
2.2 ENGENHARIA DE SOFTWARE.....	17
2.3 O UNIVERSO JAVA.....	18
2.4 SPRING FRAMEWORK.....	18
2.4.1 Spring Boot.....	19
2.4.2 Spring Web.....	21
2.4.3 Spring Data JPA.....	21
2.4.4 Uso de Anotações.....	22
2.5 BANCO DE DADOS MYSQL.....	24
2.6 HIBERNATE.....	24
2.7 INVERSÃO DE CONTROLE E INJEÇÃO DE DEPENDÊNCIA.....	25
2.8 API.....	26
2.8.1 REST E RESTFUL.....	27
2.9 JSON (JAVASCRIPT OBJECT NOTATION).....	27
3 TRABALHOS RELACIONADOS.....	29
4. METODOLOGIA.....	30
5. DESENVOLVIMENTO.....	31
5.1 MÉTODO KANBAN.....	31
5.2 LEVANTAMENTO DE REQUISITOS.....	33
5.3 DIAGRAMA DE CASOS DE USO DA APLICAÇÃO.....	35
5.4 PROTÓTIPO.....	36
6. RESULTADOS E DISCUSSÕES.....	47
7. CONSIDERAÇÕES FINAIS.....	49
REFERÊNCIAS.....	50

1 INTRODUÇÃO

Em meados de 2020, com o advento da pandemia causada pela COVID-19, o Museu de Ciências Professor Antônio Carneiro (MCPAC), assim como outras tantas instituições, precisou adaptar suas atividades e rotinas para continuar a atender o público. O que antes era ofertado de forma presencial, passou a ser online, como lives, vídeos e conteúdos produzidos pelo museu. Ainda assim, outros segmentos não possuíam estrutura para tal adaptação.

Desde o ano de 2020, o MCPAC buscou adaptar e inovar seus projetos, com o objetivo de torná-los mais acessíveis a todos os públicos. Entretanto, os meios de agendamento das visitas permaneceram os mesmos, através de um contato direto com os responsáveis pelo museu, seja por meio de ligação telefônica, ou envio de ofício. Vale ressaltar também que o MCPAC está localizado na região do sertão Central de Pernambuco e que muitos dos seus potenciais visitantes residem na zona rural e apresentam dificuldades de acesso ao museu devido à falta de transporte adequado. Diante desse panorama, surgiu a ideia de criar uma aplicação que ofereça um serviço online de agendamento de visitas para o Museu de Ciência Professor Antônio Carneiro.

A aplicação proposta é uma API (*Application Programming Interface* ou Interface de Programação de Aplicação) que utiliza o estilo arquitetural REST (*Representational State Transfer* ou Transferência Representacional de Estado), desenvolvida com ferramentas open source, como a IDE (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) Spring Tool Suite (STS), banco de dados MySQL para persistência dos dados, o framework Spring e seus módulos, para a criação do backend.

Ao explorar as fases de desenvolvimento, será possível compreender a prática dos conceitos teóricos utilizados, bem como analisar os desafios enfrentados e as soluções adotadas ao longo do processo. Além disso, a pesquisa visa levantar os requisitos do sistema com base nas demandas reais do museu, garantindo que a ferramenta desenvolvida atenda às necessidades dos usuários.

Para embasar o estudo, serão utilizadas fontes como trabalhos acadêmicos, livros, tutoriais e a documentação oficial do Spring. Esses recursos fornecem uma base teórica sólida, ao mesmo tempo em que permitem explorar as melhores

práticas para o desenvolvimento da ferramenta. Também será utilizada a ferramenta Postman¹, que possibilitará avaliar o comportamento da API em situações reais de uso, através de requisições HTTP.

Ao final deste estudo de caso, espera-se obter uma compreensão aprofundada da arquitetura do sistema e das etapas relacionadas ao desenvolvimento da aplicação, bem como identificar as contribuições relevantes proporcionadas por essa tecnologia no contexto do agendamento de visitas em um museu.

¹ O Postman é uma ferramenta open source poderosa, para construir, utilizar e testar API.

2 FUNDAMENTOS TÉCNICOS

Nesta seção, serão apresentados conceitos e definições essenciais para orientar a compreensão do leitor, bem como explorar as tecnologias e ferramentas utilizadas na construção desse trabalho.

2.1 MUSEU DE CIÊNCIAS PROFESSOR ANTÔNIO CARNEIRO

O Museu de Ciências Professor Antônio Carneiro (MCPAC) é oriundo de uma parceria entre o Espaço Ciência e a prefeitura municipal da cidade de Salgueiro-PE. De acordo com documentos internos do MCPAC (2019), até meados de 2013, o museu não estava preparado para atender o público, e a utilização de sua estrutura era pontual e precária. Poucas atividades foram desenvolvidas neste espaço durante esse período, onde a carência de recursos humanos se configura como um dos principais entraves para o efetivo funcionamento do museu. Com a solenidade de inauguração do museu nas dependências do IFSertãoPE - Campus Salgueiro, como instituição gestora, o MCPAC passou a receber visitantes das redes públicas e privadas de ensino de todo o Sertão-Central, com o objetivo de estimular a busca pelo conhecimento e o gosto pela ciência.

Por meio de editais da FACEPE², direcionados para o apoio de atividades de monitoria em centros e museus de ciências, as atividades do museu continuam a ocorrer semanalmente entre visitas ao museu, formação, palestras e mostras itinerantes fora das dependências do espaço, e através da realização de exposições em Salgueiro e cidades vizinhas. Apesar das bolsas concedidas pela FACEPE (Edital 26/2021), ainda existe uma carência relacionada a recursos humanos e transporte, que garantam o funcionamento do Museu em tempo integral. Além disso, a administração do museu é alterada de tempos em tempos, e os meios de contato para agendamento de atividades também são alterados. Isso justifica a necessidade de criar uma aplicação que possibilite automatizar o gerenciamento da agenda de atividades do museu.

² FACEPE - Fundação de Amparo à Ciência e Tecnologia de Pernambuco.

2.2 ENGENHARIA DE SOFTWARE

O desenvolvimento de um sistema, envolve o uso e o conhecimento de diversas tecnologias, ferramentas e técnicas para projetar, desenvolver, testar e manter um software de qualidade. O ciclo de vida de um produto de software está presente nas diversas áreas de conhecimento previstas na engenharia de software. O termo "Engenharia de Software" foi usado em 1969 durante uma conferência da OTAN³. Nessa ocasião, foi elaborado um relatório abrangente com mais de 130 páginas, no qual se destacava a necessidade de desenvolver software com base em princípios práticos e teóricos (VALENTE, 2020).

De acordo com BOEHM (2006, Traduzido) uma das contribuições mais significativas para a engenharia de software foi a proposta do modelo em cascata. Esse modelo estabeleceu um conjunto de etapas e atividades para guiar o processo de criação de software, levando em consideração a Análise de requisitos; Design de software; Programação; Teste de software; Gerenciamento de projetos e a Manutenção. O modelo cascata foi uma tentativa de trazer ordem e disciplina ao desenvolvimento de software, facilitando o planejamento e a previsibilidade do processo. Ainda segundo BOEHM (2006, Traduzido) outro avanço proporcionado pela engenharia de software foi o surgimento das linguagens de programação nos anos 60, como o COBOL, FORTRAN, C e Pascal que possibilitaram a criação de sistemas mais complexos e produtivos. O artigo aborda que a engenharia de software também contribuiu com o surgimento de metodologias ágeis que facilitam a colaboração, a adaptação e a entrega contínua de sistemas funcionais. Essas metodologias trouxeram mais flexibilidade e capacidade de resposta às necessidades de mudanças dos sistemas, tornando-os mais adaptáveis às demandas da atualidade.

Já para SOMMERVILLE (2011), a engenharia de software é um campo de estudo que se concentra na aplicação e discussão de problemas relacionados com todas as fases do ciclo de desenvolvimento de software.

Para o desenvolvimento da aplicação Web sugerida, foram utilizadas as áreas da engenharia de software: engenharia de requisitos, design, construção de software e testes de software.

³ OTAN - Organização do Tratado do Atlântico Norte.

2.3 O UNIVERSO JAVA

De acordo com a documentação oficial da ORACLE (2023), Java é uma das linguagens de programação de alto nível orientada a objetos e portátil mais populares do mundo, devido à sua ampla utilização. Além de ser uma linguagem de programação, Java é uma plataforma de desenvolvimento que possui uma grande versatilidade para o desenvolvimento de aplicações em ambientes diversos. Ainda segundo a documentação da ORACLE (2023), o Java EE (*Java Enterprise Edition*) é uma plataforma que oferece um conjunto de APIs e serviços para o desenvolvimento de aplicações empresariais. Essa tecnologia é amplamente utilizada para criar aplicações executadas em servidores de aplicativos Java EE, como o Apache Tomcat, JBoss, GlassFish, IBM WebSphere e Oracle WebLogic. Esses servidores implementam as diretrizes e padrões estabelecidos pelo Java EE para garantir alta escalabilidade e segurança nas aplicações corporativas.

2.4 SPRING FRAMEWORK

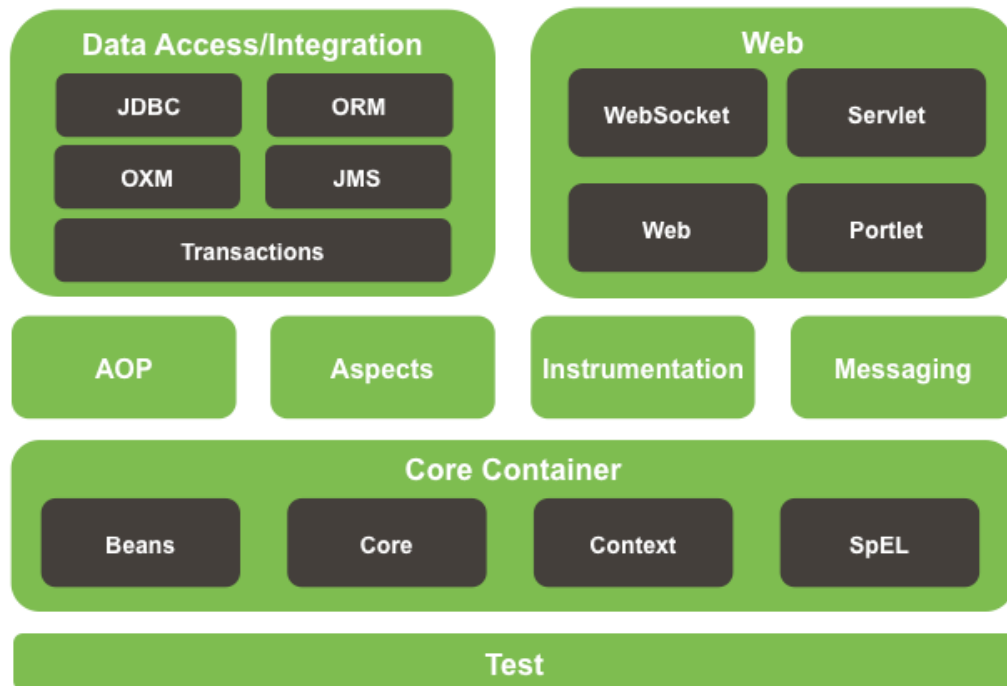
O Spring é um framework open source, criado em 2002 por Rod Johnson e apresentado pela primeira vez no livro intitulado “*Expert One-To-One J2EE Design and Development*”. Esse acontecimento é um marco importante para a história do desenvolvimento de aplicações corporativas baseadas na plataforma Java EE.

O framework traz benefícios significativos para o desenvolvimento de aplicações Java, incluindo a implementação da *Inversion of Control* (IoC) ou Inversão de Controle e *Dependency Injection* (DI) ou Injeção de Dependência, que permitem uma maior flexibilidade, menor acoplamento e auxiliam a criação de um código limpo e modular. Essas funcionalidades têm impulsionado a adoção do Spring como uma escolha popular entre os desenvolvedores Java para construção de aplicações empresariais de alto nível.

Atualmente, o framework encontra-se na versão 5.3.10, sendo uma tecnologia que oferece um modelo abrangente de programação e recursos de configuração, por meio de anotações, para aplicativos empresariais modernos baseados em Java, (SPRING FRAMEWORK, 2023).

O framework Spring é dividido em módulos, como demonstrado na figura 1.

Figura 1 - Visão geral do Spring Framework



Fonte: SPRING FRAMEWORK, 2023

Para o desenvolvimento deste trabalho, foram usados os módulos Spring Boot, Spring Web e Spring Data, esses módulos serão detalhados nas seções a seguir.

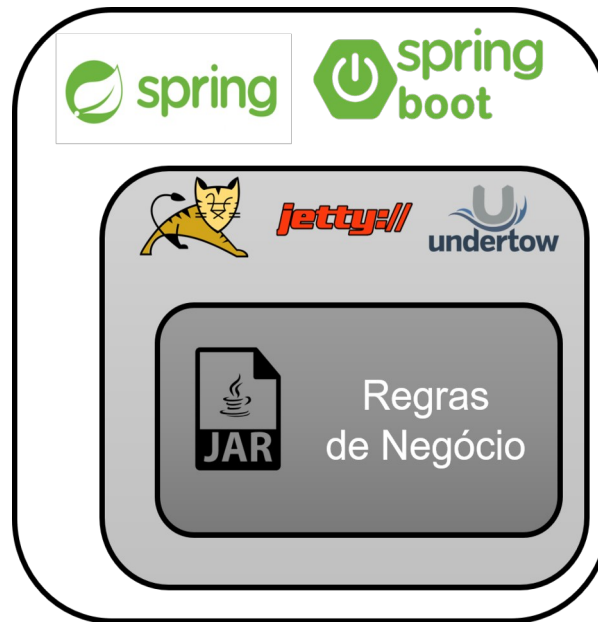
2.4.1 Spring Boot

O Spring Boot é uma estrutura para desenvolvimento de aplicações que se concentra em tornar a criação de aplicativos Spring mais fácil e rápida. Ele usa uma abordagem de "convenção sobre configuração", o que significa que muitas decisões de configuração são pré-definidas e podem ser substituídas se necessário (SPRING IO, 2023).

O Spring Boot auxilia na configuração e desenvolvimento para microsserviços, importando e configurando automaticamente todas as bibliotecas e dependências necessárias para o projeto. Ele também utiliza o Spring Actuator, que fornece *endpoints* para monitorar e gerenciar a aplicação, permitindo que os desenvolvedores coletem informações sobre o status da aplicação e suas métricas.

No novo conceito do contêiner do framework, o Spring Boot está no controle total das regras de negócio empacotadas e providenciando o servidor Web, como é apresentado na figura 2.

Figura 2 - Arquitetura do Spring Boot

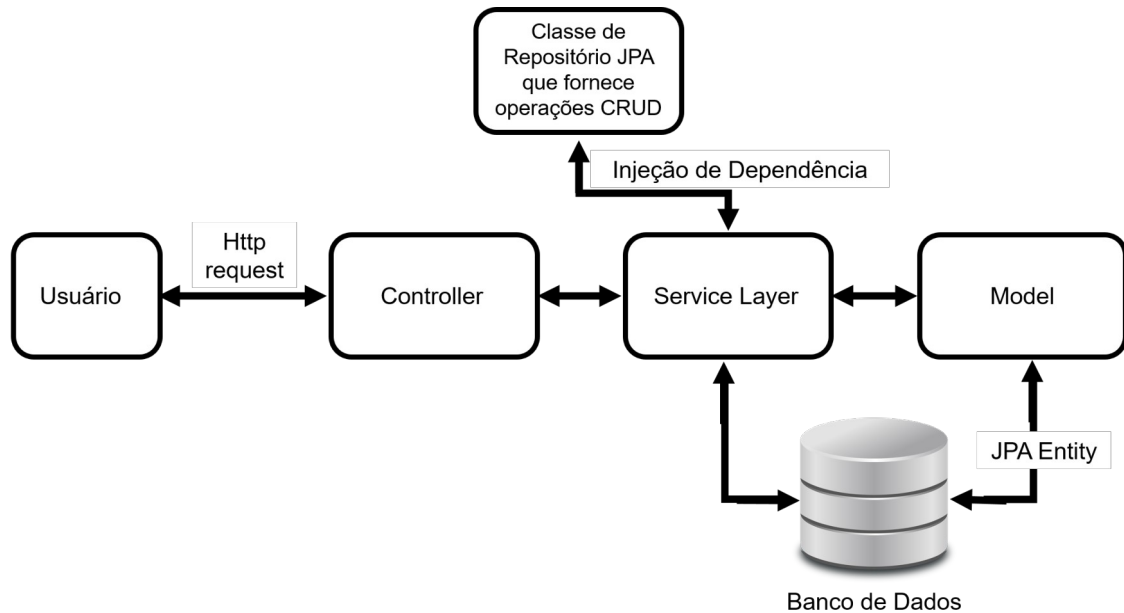


Fonte: BOAGLIO, 2017

Como observado na imagem, a arquitetura do Spring Boot segue o padrão de arquitetura em camadas, com uma separação clara de responsabilidades. O Spring Boot é altamente modular e possui muitas extensões, incluindo suporte para bancos de dados, serviços de mensagens, cache e outros. Ele também é altamente configurável, permitindo que os desenvolvedores personalizem sua aplicação de acordo com suas necessidades. O ecossistema do Spring Boot está dividido hierarquicamente em quatro camadas: apresentação, negócios, persistência e banco de dados (JACOBSEM, 2021).

A partir de uma requisição do cliente em uma aplicação Spring Boot, seu fluxo de processamento inicia no controlador. O controlador interage com a camada de serviço que, utilizando injeção de dependência, realiza o procedimento da requisição. Em seguida, atualiza o modelo e o banco de dados. A camada de controle é responsável por gerenciar as solicitações HTTP e lidar com a lógica de negócios da aplicação. A camada de serviço é responsável por executar a lógica de negócios da aplicação e a camada de persistência é responsável por interagir com o banco de dados. Esse fluxo é retratado na figura 3.

Figura 3 - Fluxo básico da requisição em aplicativos Spring Boot



Fonte: RAO & SWAMY, 2020

2.4.2 Spring Web

Segundo a documentação oficial de 2023, o Spring Web é um dos módulos do Spring Framework que fornece suporte para o desenvolvimento de aplicações Web. O Spring Web utiliza o padrão MVC (Model, View, Controller) para promover uma separação clara das responsabilidades entre a regra de negócio, apresentação e fluxo de controle. Ainda segundo a documentação, o Spring Web possui componentes centrais como o DispatcherServlet que atua como controlador principal, gerenciando as requisições e o fluxo do aplicativo, e outros controladores responsáveis por processar as requisições recebidas.

Outro aspecto em destaque na documentação é a integração do Spring Web com outros módulos do Framework, como o Spring Data e o Spring Security e outras integrações que facilitam o acesso a bancos de dados e implementar recursos de autenticação e autorização em aplicações Web.

2.4.3 Spring Data JPA

O Spring Data JPA é um módulo do framework Spring que oferece suporte ao desenvolvimento de aplicações Java com persistência de dados usando a tecnologia JPA (Java Persistence API). O JPA é uma especificação padrão da plataforma

Java para Mapeamento Objeto Relacional (ORM), e permite mapear objetos Java para tabelas em um banco de dados relacional.

O Spring Data JPA simplifica o processo de interação com o banco de dados ao fornecer uma abstração de alto nível sobre a camada de acesso a dados. Ele utiliza o conceito de repositórios, e através da definição de interfaces é possível criar métodos que representam operações comuns de persistência, como buscar, salvar, atualizar e excluir, (SPRING DATA, 2023).

2.4.4 Uso de Anotações

As anotações do framework Spring são marcadores que definem comportamentos e responsabilidades que as classes, métodos ou componentes terão na execução da aplicação. Esses marcadores simplificam a configuração e personalização de um aplicativo Spring, reduzindo o trabalho manual e oferecendo recursos, como injeção de dependência, mapeamento de URLs (*Uniform Resource Locators*), configuração de transações e tratamento de eventos, entre outros (SOUSA, 2016). O quadro 1, apresenta e descreve algumas das anotações usadas na construção da aplicação Web.

Quadro 1- Anotações do Spring Framework

Anotação	Descrição
@SpringBootApplication	Descreve a classe principal de uma aplicação Spring Boot. Ela é usada em conjunto com outras anotações para iniciar a aplicação de forma simples e eficiente.
@Repository	Usado para marcar uma classe como responsável pela persistência e recuperação de dados em um banco de dados. Ela permite ao Spring o gerenciamento da criação de instâncias da classe e fornece recursos adicionais, como exceção de tradução e gerenciamento de transações. Comumente utilizada em aplicações que usam o padrão MVC para separar a lógica de negócios da camada de acesso a dados.
@Autowired	Usado para identificar e injetar automaticamente as dependências do tipo apropriado para ser utilizado em uma classe.

@Service	Usada para marcar uma classe como pertencente à camada de Serviço da aplicação.
@Data	Fornecida pela biblioteca do Lombok, é usada para simplificar a criação de classes JavaBeans, fornecendo automaticamente getters, setters, métodos equals, hashCode e toString evitando a necessidade de escrevê-los manualmente.
@NoArgsConstructor	Fornecida pela biblioteca do Lombok, é usada para gerar automaticamente um construtor sem argumentos para uma classe permitindo que objetos dessa classe sejam instanciados sem a necessidade de fornecer argumentos para o construtor.
@AllArgsConstructor	Construtor com todos os argumentos.
@Entity	Usado para dizer que a classe é uma entidade.
@CrossOrigin	Usado em um controlador ou método específico, para configurar quais origens (hosts) são permitidos acessar a API.
@Id	Usando para identificar uma variável com chave primária.
@Column	Usada para mapear uma propriedade de uma entidade (classe) com uma coluna específica em uma tabela do banco de dados.
@Configuration	Marcar uma classe de configuração, indicando que a classe contém definições de beans e configurações para a aplicação. A anotação permite substituir o uso de arquivos XML de configuração, fornecendo uma abordagem baseada em código para configurar o ambiente de execução da aplicação.
@Api	Usada para descrever APIs RESTful.
@RestController	Descreve uma classe como um controlador de API REST, indicando que os métodos dessa classe tratam as solicitações HTTP e retornam os dados JSON ou XML, simplificando a criação de serviços RESTful.
@RequestMapping	Usada para definir endereços de URL específica para os métodos. Quando acessados por pelo usuário, especifica qual método de controle deve ser invocado para lidar com a solicitação, através de solicitação HTTP.

@PostMapping	Método usado para cadastrar novos dados.
@DeleteMapping	Método usado para deletar os dados da tabela.
@PutMapping	Método usado para editar os dados da tabela.
@GetMapping	Método usado para consultar os dados da tabela.

Fonte: Elaborado pelo autor

2.5 BANCO DE DADOS MYSQL

O MySQL é um sistema de gerenciamento de banco de dados relacional da Oracle que oferece uma combinação de desempenho, confiabilidade e facilidade de uso, tornando-se uma escolha popular entre as aplicações. O MySQL executa consultas rápidas e eficientes, o que o torna adequado para lidar com grandes volumes de dados e cargas de trabalho intensas. Outro aspecto importante sobre a ferramenta é sua interface intuitiva e suporte a linguagem de consulta SQL, que é conhecida e usada pela comunidade de desenvolvedores (ORACLE, 2023).

Outra característica do MySQL é sua capacidade de suporte a diferentes plataformas, como Windows, Linux e macOS, e flexibilidade em ser integrado a aplicativos desenvolvidos em linguagens como Java, PHP, Python, entre outras.

Em termos de segurança, o MySQL oferece recursos robustos para proteger os dados armazenados no banco de dados. De acordo com a documentação oficial ORACLE (2023), a tecnologia suporta autenticação de usuário, criptografia de dados em trânsito e em repouso, além de permitir a configuração de permissões de acesso granulares para garantir que apenas usuários autorizados possam manipular os dados. No geral, o MySQL oferece amplo suporte a desenvolvedores devido a sua documentação detalhada e comunidade ativa.

2.6 HIBERNATE

O Hibernate é um framework de persistência de dados para projetos Java, podendo ser usado em aplicações desktop e Web. Além disso, é de fácil configuração normalmente feita por XML. Para isso, o Hibernate traz uma solução em ORM, com o intuito de poupar o programador de tarefas repetitivas, como escrever o código de acesso a banco de dados e de SQL, acelerando a velocidade

do seu desenvolvimento de uma forma dinâmica (DevMedia, 2020). A figura 4 mostra como funciona a persistência de dados das páginas Java através do Hibernate.

Figura 4 - Persistência de dados com o Hibernate



Fonte: DevMedia, 2020

Segundo Linhares (2005), o Hibernate não é recomendado em sistemas que fazem uso extensivo de *stored procedures*, *triggers* ou que implementam a maior parte da lógica da aplicação no banco de dados. Ele é mais indicado para sistemas que contam com um modelo robusto, onde a maior parte da lógica de negócios fica na própria aplicação Java, dependendo pouco de funções específicas do banco de dados.

2.7 INVERSÃO DE CONTROLE E INJEÇÃO DE DEPENDÊNCIA

Inversão de controle e injeção de dependência são conceitos de projetos relacionados à organização de código e gerenciamento de dependências em um sistema de software. Segundo a documentação oficial do SPRING (2023), a inversão de controle é um padrão de projeto que visa inverter a relação de controle entre os objetos em um sistema. Em vez de um objeto criar e gerenciar as instâncias de outros objetos que ele depende, ele delega essa responsabilidade a um framework ou container que gerencia a instância dos objetos e as injeta nos objetos que dependem delas. Dessa forma, os objetos ficam desacoplados e o controle de instância é transferido para um componente externo. Para Walls (2022), no framework Spring a inversão de controle é implementada através do mecanismo de injeção de dependência, permitindo que os desenvolvedores se concentrem na implementação dos negócios enquanto o Spring cuida da resolução e fornecimento

das dependências. Isso simplifica o desenvolvimento, promove a reutilização de código e oferece diversas opções de configuração, tornando a inversão de controle uma prática comum e eficaz no desenvolvimento de aplicações com o Spring.

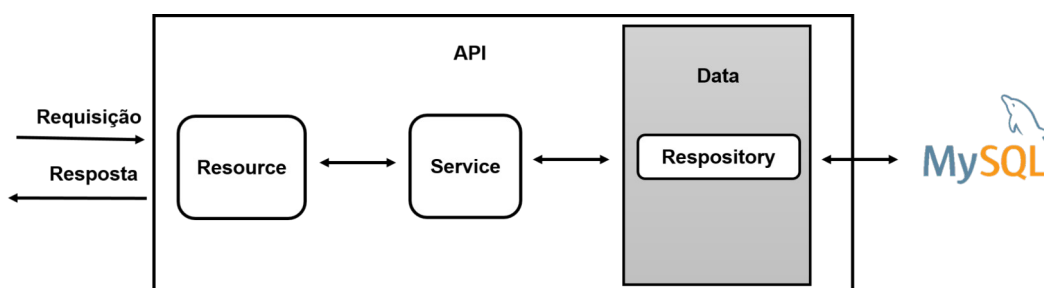
Ainda, segundo a documento do SPRING (2023), a injeção de dependência é uma técnica utilizada para os objetos definirem suas dependências por meio de argumentos do construtor (ou seja, implementar a inversão de controle). Ela consiste em injetar as dependências de um objeto através de um mecanismo automático em vez de criá-las manualmente dentro do objeto. Isso permite que as dependências sejam gerenciadas de forma centralizada e com maior flexibilidade. Através deste princípio o código se torna mais limpo, e o desacoplamento é mais eficaz, tornando suas classes mais fáceis de testar, especialmente quando as dependências estão em interfaces ou classes abstratas. Em resumo, a inversão de controle é um conceito mais amplo que se refere à inversão de controle entre objetos, enquanto a injeção de dependência é uma técnica específica para implementar a inversão de controle.

2.8 API

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web (NASCIMENTO, 2014).

Uma API pode ser considerada como uma camada de abstração que permite que desenvolvedores utilizem determinados recursos ou serviços de um sistema sem precisar conhecer todos os detalhes de sua implementação interna. Ela define os métodos, parâmetros e formatos de dados aceitos e retornados, permitindo que os desenvolvedores criem integrações ou construam aplicativos usando esses recursos disponibilizados. A figura 5 apresenta a arquitetura de alto nível da aplicação proposta.

Figura 5 - Fluxo básico da requisição da aplicação proposta



Fonte: Elaborado pelo autor

2.8.1 REST E RESTFUL

REST é um estilo arquitetural amplamente utilizado para projetar APIs para serviços Web. Para Fielding (2000), o principal conceito por trás do REST é o uso dos métodos HTTP como (GET, POST, PUT e DELETE) para operar em recursos identificados por URLs.

Uma aplicação RESTful utiliza esses métodos e URLs para permitir a comunicação entre clientes e servidores, onde os clientes podem solicitar a representação ou manipulação de recursos. Os princípios do REST incluem, (FIELDING, 2000):

- **Uso de recursos:** Uma API REST expõe recursos, que podem ser objetos, dados ou funcionalidades específicas. Cada recurso é identificado por uma URI (*Uniform Resource Identifier*), que permite que os clientes acessem e interajam com esses recursos.
- **Representações de recursos:** Os recursos em uma API REST podem ser representados em diferentes formatos, como JSON (*JavaScript Object Notation*) ou XML (*Extensible Markup Language*). As representações são enviadas de volta para o cliente em resposta às solicitações.
- **Estado do aplicativo:** Uma API REST é "*stateless*", o que significa que cada requisição do cliente deve conter todas as informações necessárias para ser processada pelo servidor. O servidor não mantém informações de estado sobre as sessões dos clientes.

No geral, o RESTful é uma abordagem versátil adotada para o desenvolvimento de APIs Web. Ele fornece uma arquitetura simples e baseada em padrões da Web, que promove a interoperabilidade e a escalabilidade. Através do

uso adequado dos métodos HTTP, URLs e recursos, as APIs RESTful podem oferecer uma maneira eficiente e consistente de criar serviços Web.

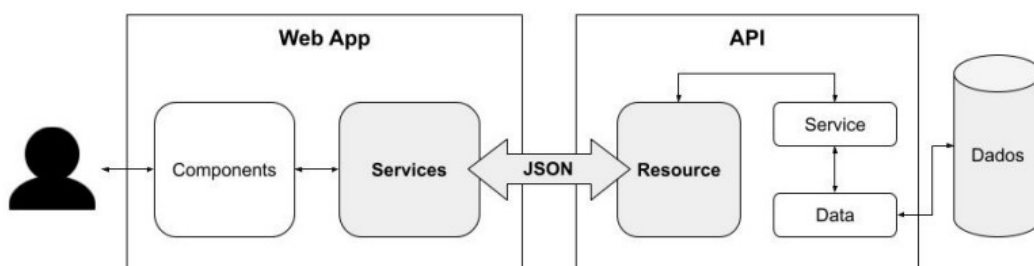
2.9 JSON (JAVASCRIPT OBJECT NOTATION)

O JSON (JavaScript Object Notation) é um padrão de comunicação de arquivos, com um formato leve e fácil para troca de dados entre sistemas. O JSON é muito parecido com a estrutura de um objeto JavaScript, sendo composto por pares de chave-valor. As chaves são *strings* que atuam como identificadores para os valores correspondentes, que podem ser *strings*, números, booleanos, *arrays*, objetos ou *null*, porém não é necessário implementar o JavaScript para utilizar o JSON (JSON, 2023).

Algumas de suas vantagens incluem leitura mais simples, analisador(parsing) mais fácil, suporte a objetos, velocidade maior na execução e transporte de dados, arquivo com tamanho reduzido. De acordo com a documentação da classe Java *ObjectMapper* do Jackson-databind , o JSON é comumente usado em um ambiente Web para transmitir dados serializados entre um servidor e um aplicativo da web, como parte de uma API, através do protocolo HTTP (FASTERXML, 2023).

A API, por sua vez, realiza o processamento do JSON em objeto Java. A figura 6 mostra como funciona a interação do JSON com o sistema.

Figura 6 - Fluxo básico da requisição em aplicativos Spring Boot



Fonte: (RAO; SWAMY, 2020)

Já na figura 7 é retratado um exemplo simples da exibição de um objeto JSON, com sua estrutura.

Figura 7 - Chave-valor de um objeto JSON

```
json
{
  "nome": "João",
  "idade": 30,
  "solteiro": true,
  "hobbies": ["séries", "academia"]
}
```

Fonte: Elaborado pelo autor

O objeto acima tem quatro pares de chave-valor. A chave "nome" tem o valor "João", a chave "idade" tem o valor 30, a chave "solteiro" tem o valor true e a chave "hobbies" tem um array como valor, contendo as Strings "séries" e "academia". O JSON é amplamente utilizado porque é fácil de ser interpretado e manipulado por muitas linguagens de programação. Além disso, é uma forma eficiente de transmitir dados pela rede, pois possui uma estrutura compacta e pode ser facilmente convertido em objetos em memória.

3 TRABALHOS RELACIONADOS

A revisão da literatura sobre o tema do Spring e Java EE como tecnologias para desenvolvimento Web, apresentou as seguintes pesquisas técnicas para compor a seção de trabalhos relacionados: "Sistema web para reserva de salão de festas em condomínios", o artigo de Gibikoski (2019) apresenta o desenvolvimento de um sistema Web para gerenciamento de reservas de salão de festas em condomínios utilizando tecnologias como Java Server Faces, Spring Framework e Hibernate. O sistema permite que os usuários verifiquem a disponibilidade do salão, realizem reservas e pagamentos online. O trabalho demonstra a importância de soluções tecnológicas para otimizar processos de gerenciamento em condomínios.

A monografia de Batista e Bastos (2017), apresenta a utilização do framework Spring para o desenvolvimento de aplicações Web baseadas em

microserviços. O trabalho discute os conceitos de microserviços e a arquitetura de software em que eles se baseiam, além de fornecer uma introdução ao Spring Framework. São apresentados exemplos de implementação de microserviços utilizando o Spring Boot e Spring Cloud, destacando suas vantagens em relação a abordagens monolíticas.

O trabalho de Teixeira (2021), descreve a construção de uma API REST que permite a contratação de profissionais autônomos por meio da plataforma digital Severino. O artigo apresenta uma visão geral do desenvolvimento da API utilizando o framework Spring Boot. O trabalho também descreve o processo de testes e validação da API, bem como os resultados obtidos.

Sousa e Pereira (2007), apresentam um estudo de caso sobre a criação de aplicações Web utilizando o framework Spring em conjunto com a plataforma J2EE, desde a configuração inicial do ambiente até a criação de CRUDs e a implementação de segurança da aplicação. O objetivo é mostrar como o Spring pode ser uma ferramenta útil para a implementação de projetos Web robustos.

O artigo de Júnior e Sousa (2018) apresentam uma proposta de desenvolvimento de um aplicativo mobile que visa auxiliar na realização de treinos em casa. O uso do Spring Boot é destacado como uma escolha estratégica, permitindo a criação de um ambiente de desenvolvimento mais eficiente e simplificando a integração com outras ferramentas. Esses trabalhos abordam diferentes aspectos do uso de Spring e Java EE para desenvolvimento de sistemas, desde a comparação entre as tecnologias até a implementação prática de projetos utilizando esses frameworks.

4. METODOLOGIA

Inicialmente foi realizada uma pesquisa documental sobre como é realizado o processo de agendamento de visitas e exposições realizadas pelo museu. Segundo Lakatos & Marconi (2017, p. 174) a pesquisa documental é uma abordagem sistemática que envolve a coleta e análise de diversos tipos de documentos como o uso de fonte primária de informação para responder a questões de pesquisa. A coleta das informações foi feita através do livro de atas, ofícios e documentos do museu. O agendamento de visitas é realizado por contato direto com os responsáveis pelo museu. O desenvolvimento da aplicação envolveu pesquisa

bibliográfica sobre engenharia de software, estudos acadêmicos, vídeos tutoriais e a documentação oficial do Spring Boot. Ainda de acordo com Lakatos & Marconi (2017, p.158), a pesquisa bibliográfica é fundamental para embasar teoricamente um estudo e compreender o estado da arte sobre determinado tema, contribuindo para a fundamentação teórica e a construção do referencial teórico de um trabalho acadêmico.

Simulações de requisições foram realizadas com a ferramenta Postman para avaliar o desempenho da API, verificando funcionalidades e identificando melhorias. Em conclusão, a combinação de pesquisa bibliográfica, vídeos tutoriais e a documentação oficial do Spring Boot proporcionou uma base de conhecimento teórico e prático, permitindo uma análise aprofundada do desempenho da tecnologia em relação aos requisitos propostos. Essa metodologia foi fundamental para o sucesso e a validade do estudo de caso.

5. DESENVOLVIMENTO

Neste capítulo, são descritos os procedimentos adotados no processo de desenvolvimento do projeto. Os métodos abrangem as etapas realizadas para a criação do sistema, englobando desde a coleta dos requisitos até a etapa de codificação.

5.1 MÉTODO KANBAN

O Kanban é um método de gestão de trabalho, originado do Sistema Toyota de Produção (TPS). A abordagem representa um sistema baseado na demanda do cliente, em vez da prática padrão de produzir certas quantidades de mercadorias e empurrá-las ao mercado. Seu sistema único de produção criou a base da produção *Lean*. Seu propósito central é minimizar as atividades, sem sacrificar a produtividade. O objetivo principal é criar mais valor para o cliente, sem gerar mais custos (KANBANIZE, 2023).

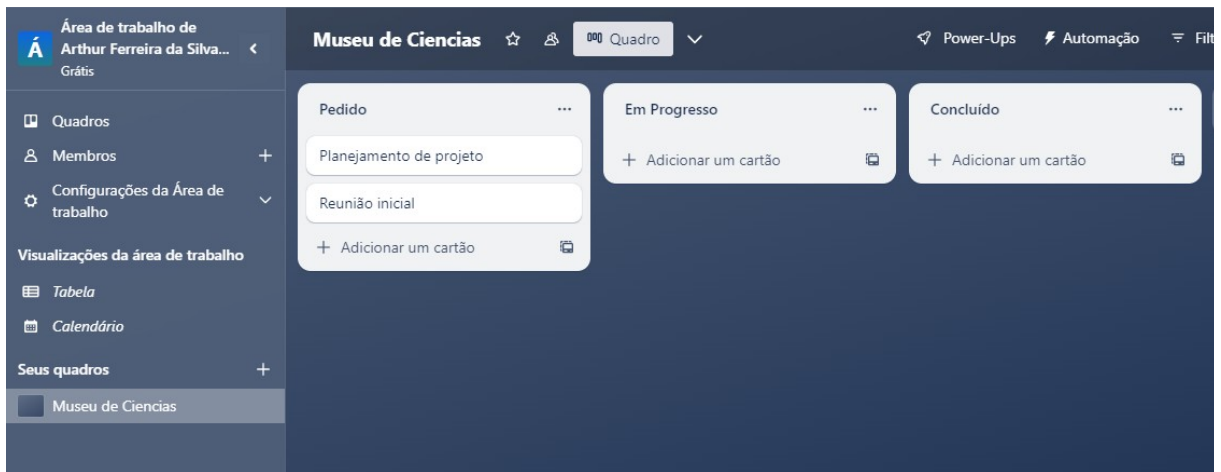
No início do século 21, os principais participantes da indústria de software perceberam rapidamente como o Kanban poderia ser usado para mudar positivamente o formato de entrega de produtos e serviços. Com um foco maior na eficiência e aproveitando, Kanban deixou os confins da indústria automotiva e foi cada vez mais aplicado a outros setores industriais e comerciais. O que de fato é

hoje reconhecido como o Método Kanban surgiu no início de 2007, e é resultado de anos de testes, experiência e esforços conjuntos de figuras de destaque na comunidade Lean e Agile (BOEG, 2012).

Algumas das características evidenciadas no Kanban incluem visualização do fluxo de trabalho, limites do trabalho em progresso, mais conhecido como *Work In Progress* (WIP), gerenciamento de fluxo, transparência das políticas do processo, foco na melhoria contínua (feedbacks) e gestão colaborativa (BOEG, 2012).

Ainda segundo Boeg (2012) o tamanho do quadro Kanban vai depender da visualização de limites do projeto. O quadro Kanban mais simples possui três colunas – “Pedido”, “Em Progresso” e “Concluído”. A Figura 8 exibe um quadro Kanban simples, feito na ferramenta Trello.

Figura 8 - Quadro de tarefas Kanban Trello



Fonte: Elaborado pelo autor

Os fundamentos do Kanban podem ser divididos em dois tipos de princípios e seis práticas, demonstrados no quadro 2.

Quadro 2 - Fundamentos e práticas do Kanban

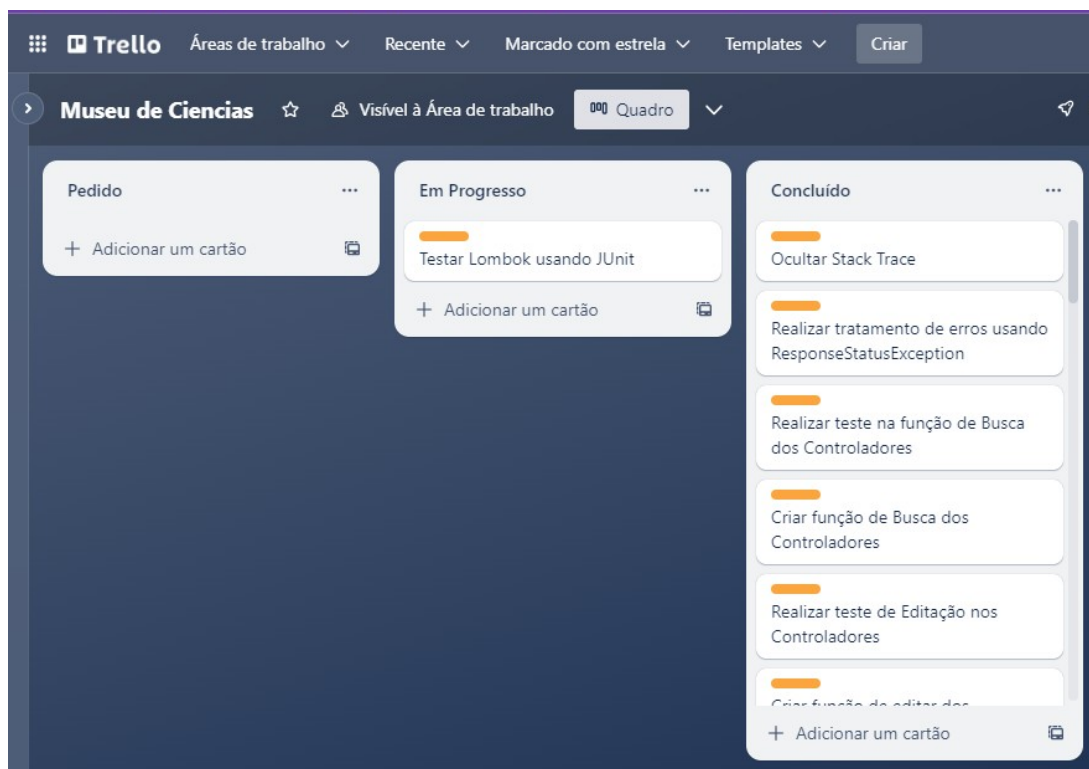
Princípios do Gerenciamento de Mudanças	Princípios da Entrega de Serviços
Comece com o que você faz agora.	Foco nas necessidades e expectativas do cliente.
Concorde em Buscar Mudanças Incrementais e Evolutivas.	Gerencie o trabalho, não os trabalhadores.

Incentivar atos de liderança em todos os níveis.	Revise regularmente a rede de serviços.
--	---

Fonte: KANBANIZE, 2023.

A figura 9 apresenta o fluxo de trabalho do projeto, utilizando a plataforma Trello para controle das atividades a serem realizadas. Seguindo a metodologia Kanban, as atividades foram classificadas em três tipos de status: “Pedido”, “Em progresso” e “Concluído”.

Figura 9 - Quadro de tarefas do projeto



Fonte: Elaborado pelo autor

5.2 LEVANTAMENTO DE REQUISITOS

Primeira etapa no ciclo de desenvolvimento de um software, é considerada de extrema importância, pois falhas nessa etapa pode prejudicar todas as outras. O levantamento de requisitos, de acordo com Sommerville (2011), é uma etapa crucial no processo de desenvolvimento de software. Ele envolve a identificação, análise e documentação das necessidades e expectativas dos usuários e demais partes interessadas em relação ao sistema a ser desenvolvido.

Os quadros 3 e 4 demonstram os Requisitos Funcionais (RF) e Não Funcionais (RFN) levantados após análise de ofícios e relatórios de visitas conseguidos com a coordenação do museu.

Quadro 3 - Requisitos funcionais

Referência	Descrição
RF001 - Efetuar Login	Permite que os usuários façam autenticação no sistema, fornecendo suas credenciais de acesso.
RF002 - Efetuar Logout	Permite que os usuários encerrem sua sessão no sistema, encerrando o acesso às funcionalidades restritas.
RF003 - Cadastrar usuário	Permite que novos usuários se cadastrem no sistema, fornecendo as informações necessárias para criar uma conta.
RF004 - Atualizar dados do usuário	Permite que os usuários alterem suas informações cadastrais, como nome, endereço ou informações de contato.
RF005 - Remover usuário	Permite que os usuários sejam excluídos do sistema, removendo permanentemente suas informações e acesso.
RF006 - Buscar Usuário	Permite a busca e recuperação de informações de usuários cadastrados no sistema, com base em critérios específicos.
RF007 - Solicitar agendamento	Permite que os usuários solicitem agendamentos de serviços ou recursos disponíveis no sistema.
RF008 - Cancelar agendamento	Permite que os usuários cancelem agendamentos previamente solicitados.
RF009 - Alterar dados do agendamento	Permite que os usuários modifiquem informações de um agendamento existente, como data, horário ou detalhes adicionais.
RF010 - Listar agendamentos	Permite que os usuários visualizem uma lista de agendamentos realizados, exibindo informações como data, horário e detalhes relevantes.

Fonte: Elaborado pelo autor

Quadro 4 - Requisitos não funcionais

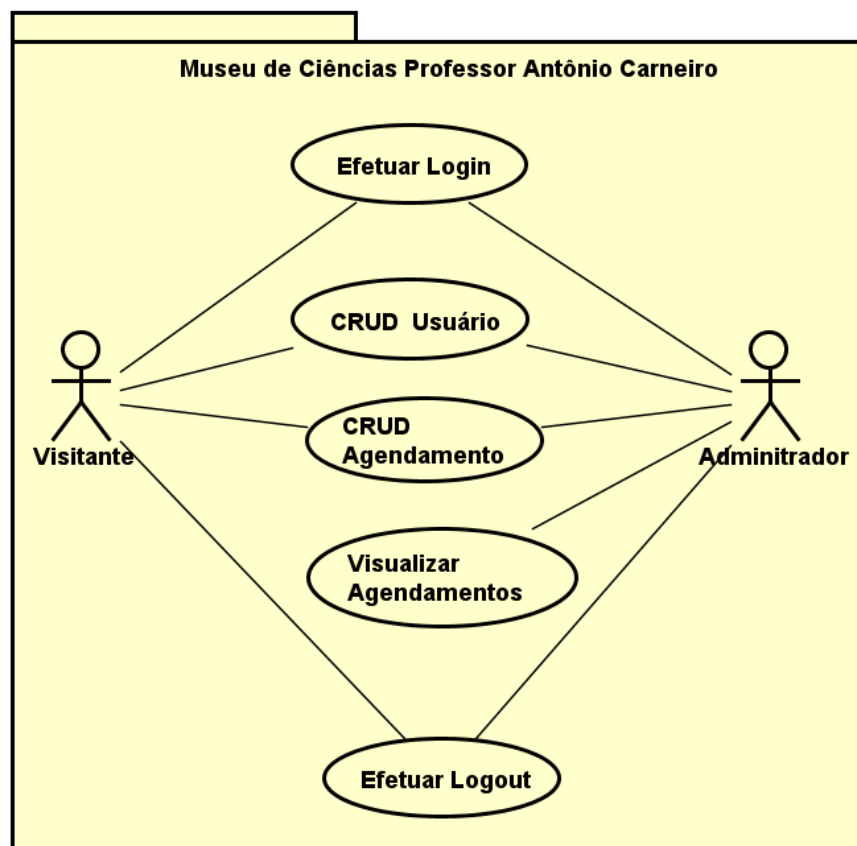
Referência	Descrição
RNF001 - Segurança	O sistema deverá garantir que cada usuário só utilizará as funcionalidades definidas para seu perfil.
RNF002 - Usabilidade	Os usuários com conhecimento básico, após treinamento de 30 minutos não devem cometer mais de 3 (três) erros após 30 minutos de uso.

Fonte: Elaborado pelo autor

5.3 DIAGRAMA DE CASOS DE USO DA APLICAÇÃO

A Figura 10 exibe o diagrama de casos de uso, que ilustra as interações dos atores e as funcionalidades do sistema proposto.

Figura 10 - Diagrama de casos de uso



Fonte: Elaborado pelo autor

No sistema, todos os atores terão acesso aos requisitos do sistema, porém

cada um com um nível de autorização diferente. A exemplo disso o ator visitante poderá consultar seus próprios agendamentos, enquanto o ator administrador poderá consultar todos os agendamentos cadastrados na aplicação. A seguir, será apresentada no quadro 5 a história de usuário da funcionalidade “Solicitar agendamento”.

Quadro 5 - História de usuário

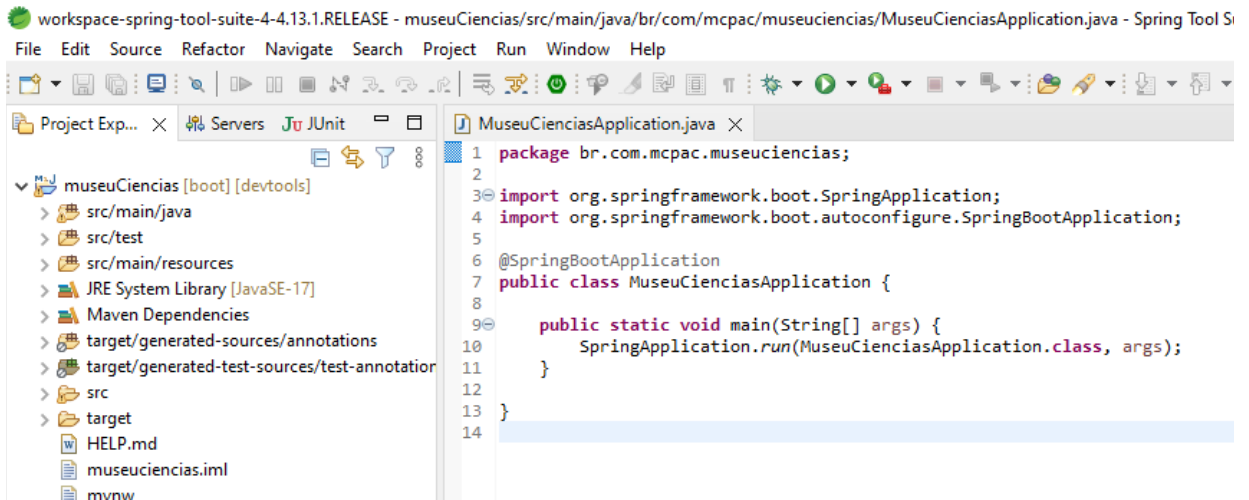
História de usuário 005: Solicitar agendamento	
Descrição	Como visitante, gostaria de realizar o agendamento no sistema, para ter acesso às dependências do museu.
Prioridade	Alta.

Fonte: Elaborado pelo autor

5.4 PROTÓTIPO

Após o levantamento de requisitos, foi feito a construção do protótipo (*back-end*) do sistema do museu, que consiste em uma API para agendamento online. A figura 11 apresenta o código em Java que é o ponto de entrada para a aplicação Spring Boot denominada "MuseuCienciasApplication.java". Esse código inicia a aplicação Spring Boot e a configura para ser executada a partir da classe MuseuCienciasApplication. O método *main* é o ponto de entrada do programa e é onde a aplicação Spring Boot é iniciada. O código está localizado no pacote `br.com.mcpac.museuciencias`.

Figura 11 - Arquivo MuseuCienciasApplication.java



```

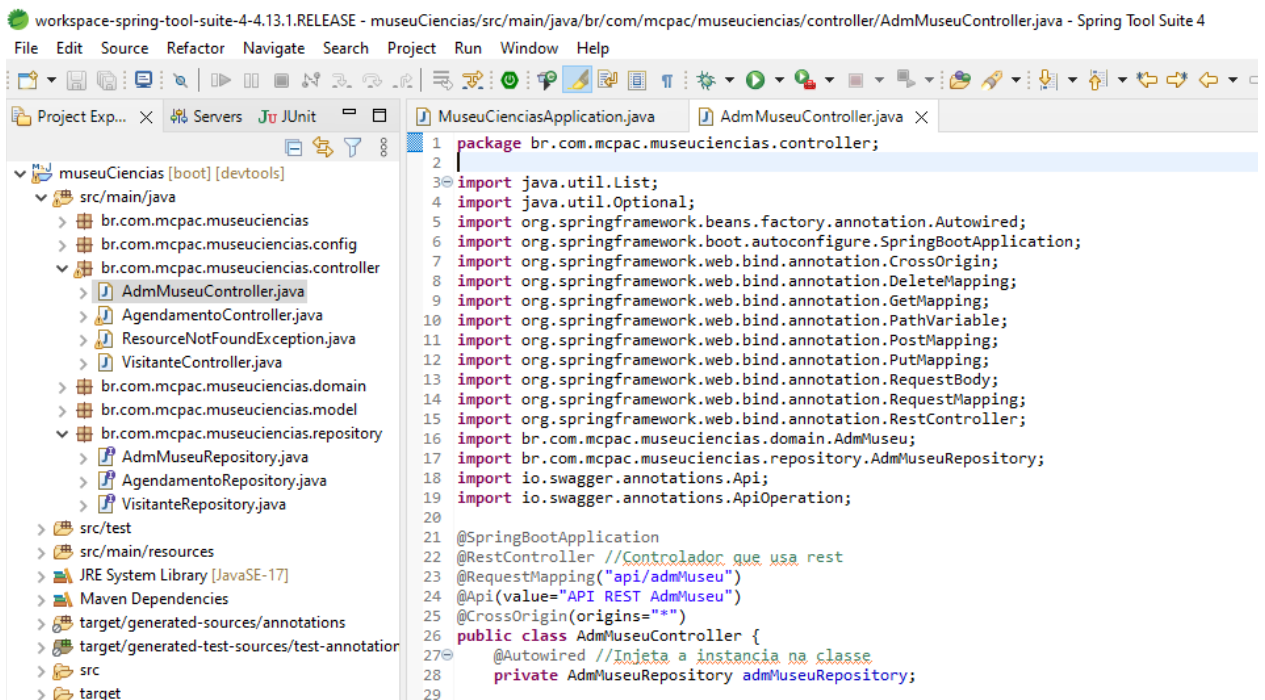
workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/MuseuCienciasApplication.java - Spring Tool S
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotation
  src
  target
  HELP.md
  museuciencias.iml
  mvnw
MuseuCienciasApplication.java
1 package br.com.mcpac.museuciencias;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class MuseuCienciasApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(MuseuCienciasApplication.class, args);
11     }
12
13 }
14

```

Fonte: Elaborado pelo autor

As figuras 12 e 13, apresentam o código da classe AdmMuseuController localizado na pasta br.com.mcpac.museuciencias.controller. Essa classe define um controlador REST que lida com solicitações HTTP em uma API REST. Nesse caso, os métodos são responsáveis por listar todos os Administradores; Buscar um único administrador com base no telefone fornecido; Salvar o novo Administrador no banco de dados usando o método; Deletar um Administrador do banco de dados e por fim Atualizar dados do Administrador.

Figura 12 - Arquivo AdmMuseuController.java



```

workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/controller/AdmMuseuController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
    br.com.mcpac.museuciencias.domain
    br.com.mcpac.museuciencias.model
    br.com.mcpac.museuciencias.repository
      AdmMuseuRepository.java
      AgendamentoRepository.java
      VisitanteRepository.java
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotation
  src
  target
MuseuCienciasApplication.java
AdmMuseuController.java
1 package br.com.mcpac.museuciencias.controller;
2
3 import java.util.List;
4 import java.util.Optional;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7 import org.springframework.web.bind.annotation.CrossOrigin;
8 import org.springframework.web.bind.annotation.DeleteMapping;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import org.springframework.web.bind.annotation.PutMapping;
13 import org.springframework.web.bind.annotation.RequestBody;
14 import org.springframework.web.bind.annotation.RequestMapping;
15 import org.springframework.web.bind.annotation.RestController;
16 import br.com.mcpac.museuciencias.domain.AdmMuseu;
17 import br.com.mcpac.museuciencias.repository.AdmMuseuRepository;
18 import io.swagger.annotations.Api;
19 import io.swagger.annotations.ApiOperation;
20
21 @SpringBootApplication
22 @RestController //Controlador que usa rest
23 @RequestMapping("api/admMuseu")
24 @Api(value="API REST AdmMuseu")
25 @CrossOrigin(origins="*")
26 public class AdmMuseuController {
27     @Autowired //Injeta a instancia na classe
28     private AdmMuseuRepository admMuseuRepository;
29

```

Fonte: Elaborado pelo autor

Figura 13 - Métodos da classe AdmMuseuController

```

26 public class AdmMuseuController {
27     @Autowired //Injeta a instancia na classe
28     private AdmMuseuRepository admMuseuRepository;
29
30     @GetMapping
31     @ApiOperation(value="Retorna uma lista de todos os Administradores")
32     public List<AdmMuseu> listar(){
33         List<AdmMuseu> resultado = admMuseuRepository.findAll();
34         return resultado;
35     }
36
37     @GetMapping("/{telefone}")
38     @ApiOperation(value="Retorna um unico Administrador")
39     public Optional<AdmMuseu> buscar(@PathVariable Integer telefone) {
40         return admMuseuRepository.findById(telefone);
41     }
42
43     @PostMapping
44     @ApiOperation(value="Cadastra um novo Administrador")
45     public AdmMuseu inserir(@RequestBody AdmMuseu admMuseu) {
46         return admMuseuRepository.save(admMuseu);
47     }
48
49     @DeleteMapping("/{telefone}")
50     @ApiOperation(value="Deleta um Administradore")
51     public Optional<AdmMuseu> excluir(@PathVariable Integer telefone) {
52         Optional<AdmMuseu> adm = admMuseuRepository.findById(telefone);
53         admMuseuRepository.delete(adm.get());
54         return adm;
55     }
56
57     @PutMapping
58     @ApiOperation(value="Atualiza um Administrador")
59     public AdmMuseu editar(@RequestBody AdmMuseu admMuseu) {
60         return admMuseuRepository.save(admMuseu);
61     }
62 }

```

Fonte: Elaborado pelo autor

A figura 14, apresenta o código da classe VisitanteController também localizado na pasta `br.com.mcpac.museuciencias.controller`. Assim como a classe anterior, essa classe, também define um controlador REST que lida com solicitações HTTP em uma API REST que irão listar todos os visitantes; Buscar um único visitante com base no telefone fornecido; Salvar o novo visitante no banco de dados; Deletar um visitante do banco de dados e por fim Atualizar dados do visitante.

Figura 14 - Classe VisitanteController

```

workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/controller/VisitanteController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
    br.com.mcpac.museuciencias.domain
    br.com.mcpac.museuciencias.model
    br.com.mcpac.museuciencias.repository
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotator
  src
  target
  HELP.md
  museuciencias.iml
  mvnw
  mvnw.cmd
  pom.xml

21 @SpringBootApplication
22 @RestController
23 @RequestMapping("/api/visitante")
24 public class VisitanteController {
25     @Autowired
26     private VisitanteRepository visitanteRepository;
27
28     @GetMapping
29     @ApiOperation(value="Retorna uma lista de todos os Administradores")
30
31     public List<Visitante> listar(){
32         List<Visitante> visitantes = visitanteRepository.findAll();
33         return visitantes;
34     }
35
36     @GetMapping("/{telefone}")
37     public Optional<Visitante> buscar(@PathVariable Integer telefone) {
38         return visitanteRepository.findById(telefone);
39     }
40
41     @PostMapping
42     public Visitante inserir(@RequestBody Visitante visita) {
43         return visitanteRepository.save(visita);
44     }
45
46     @DeleteMapping("/{telefone}")
47     public Optional<Visitante> excluir(@PathVariable Integer telefone) {
48         Optional<Visitante> visitantes = visitanteRepository.findById(telefone);
49         visitanteRepository.delete(visitantes.get());
50         return visitantes;
51     }
52
53     @PutMapping
54     public Visitante editar(@RequestBody Visitante visita) {
55         return visitanteRepository.save(visita);
56     }
57

```

Fonte: Elaborado pelo autor

As figuras 15 e 16, apresentam o código da classe AgendamentoController também localizado no package `br.com.mcpac.museuciencias.controller`. Assim como a classe anterior, essa classe também define um controlador REST que lida com solicitações HTTP em uma API REST. Os métodos dessa classe são responsáveis por listar todos os Agendamentos; Inserir o novo agendamento no banco de dados, buscando o visitante relacionado ao agendamento pelo Id fornecido, usando o método `findById()` do `visitanteRepository`. Caso não seja encontrado, lança uma exceção `ResourceNotFoundException`; Deletar um agendamento do banco de dados e por fim Atualizar dados do agendamento.

Figura 15 - Atributos da classe AgendamentoController

```

workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/controller/AgendamentoController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
    br.com.mcpac.museuciencias.domain
      AdmMuseu.java
      Agendamento.java
      AgendamentoRequest.java
      Visitante.java
    br.com.mcpac.museuciencias.model
    br.com.mcpac.museuciencias.repository
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotation
  target
  HELP.md
  museuciencias.iml
  mvnw
  mvnw.cmd
  pom.xml

29 @SpringBootApplication
30 @RestController //Controlador, que usa rest
31 @RequestMapping("/api/agendamento")
32
33 public class AgendamentoController {
34
35     @Autowired
36     private AgendamentoRepository agendamentoRepository;
37     @Autowired
38     private VisitanteRepository visitanteRepository;
39
40     @GetMapping
41     @ApiOperation(value="Retorna uma lista de todos os Administradores")
42
43     public List<Agendamento> listar(){
44         List<Agendamento> agenda = agendamentoRepository.findAll();
45         return agenda;
46     }
47
48     @GetMapping("/{idVisita}")
49     public Optional<Agendamento> buscar(@PathVariable Integer idVisita){
50         Optional<Agendamento> resultado = agendamentoRepository.findById(idVisita);
51         return resultado;
52     }
53
54     @PostMapping
55     public ResponseEntity<Agendamento> inserir(@RequestBody AgendamentoRequest agendamento) {
56         Visitante visitante = visitanteRepository.findById(agendamento.getVisitanteId())
57             .orElseThrow(() -> new ResourceNotFoundException("Visitante", "telefone", agendamento.getVisitanteId()));
58
59         Agendamento novoAgendamento = new Agendamento();
60
61         novoAgendamento.setHorario(agendamento.getHorario());
62         novoAgendamento.setData(agendamento.getData());
63         novoAgendamento.setVisitante(visitante);
64         novoAgendamento = agendamentoRepository.save(novoAgendamento);
65
66         return new ResponseEntity<>(novoAgendamento, HttpStatus.CREATED);
67     }

```

Fonte: Elaborado pelo autor

Figura 16 - Métodos da Classe AgendamentoController

```

workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/controller/AgendamentoController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
    br.com.mcpac.museuciencias.domain
      AdmMuseu.java
      Agendamento.java
      AgendamentoRequest.java
      Visitante.java
    br.com.mcpac.museuciencias.model
    br.com.mcpac.museuciencias.repository
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotation
  target
  HELP.md
  museuciencias.iml
  mvnw
  mvnw.cmd
  pom.xml

46 }
47
48 @GetMapping("/{idVisita}")
49 public Optional<Agendamento> buscar(@PathVariable Integer idVisita){
50     Optional<Agendamento> resultado = agendamentoRepository.findById(idVisita);
51     return resultado;
52 }
53
54 @PostMapping
55 public ResponseEntity<Agendamento> inserir(@RequestBody AgendamentoRequest agendamento) {
56     Visitante visitante = visitanteRepository.findById(agendamento.getVisitanteId())
57         .orElseThrow(() -> new ResourceNotFoundException("Visitante", "telefone", agendamento.getVisitanteId()));
58
59     Agendamento novoAgendamento = new Agendamento();
60
61     novoAgendamento.setHorario(agendamento.getHorario());
62     novoAgendamento.setData(agendamento.getData());
63     novoAgendamento.setVisitante(visitante);
64     novoAgendamento = agendamentoRepository.save(novoAgendamento);
65
66     return new ResponseEntity<>(novoAgendamento, HttpStatus.CREATED);
67 }
68
69 @DeleteMapping("/{idVisita}")
70 public Optional<Agendamento> excluir(@PathVariable Integer idVisita) {
71     Optional<Agendamento> agenda = agendamentoRepository.findById(idVisita);
72     agendamentoRepository.delete(agenda.get());
73     return agenda;
74 }
75
76 @PutMapping
77 public Agendamento editar(@RequestBody Agendamento agenda) {
78     Agendamento agendaEditada = agendamentoRepository.save(agenda);
79     return agendaEditada;
80 }
81

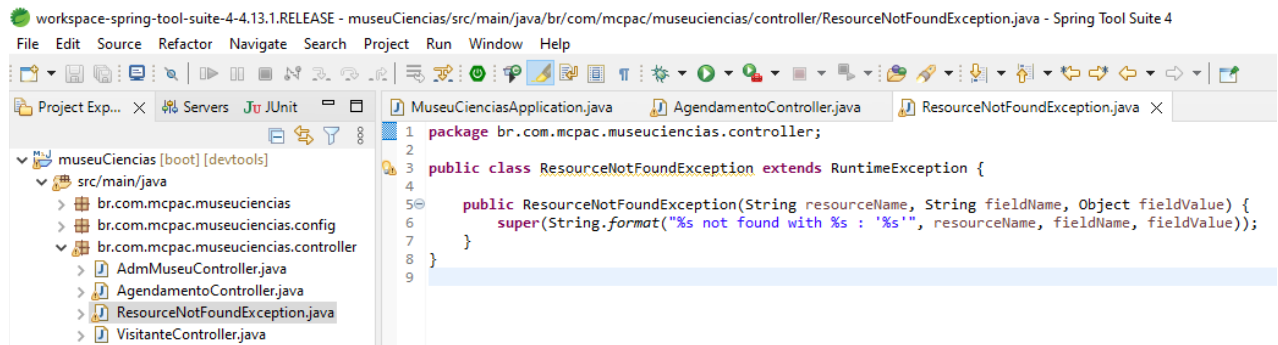
```

Fonte: Elaborado pelo autor

A figura 17 apresenta um código Java que define uma classe

`ResourceNotFoundException` que estende a classe `RuntimeException`. Essa classe é usada para lançar uma exceção personalizada quando um recurso específico não é encontrado.

Figura 17 - Arquivo `ResourceNotFoundException.java`



```
workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/controller/ResourceNotFoundException.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit MuseuCienciasApplication.java AgendamentoController.java ResourceNotFoundException.java
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
1 package br.com.mcpac.museuciencias.controller;
2
3 public class ResourceNotFoundException extends RuntimeException {
4
5     public ResourceNotFoundException(String resourceName, String fieldName, Object fieldValue) {
6         super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));
7     }
8 }
9
```

Fonte: Elaborado pelo autor

A figura 18 apresenta a classe `AdmMuseu` que representa uma entidade no banco de dados com os campos `telefone`, `nome`, `endereco`, `email` e `senha`, onde o `telefone` é definido como chave primária através da anotação `@Id`. Essa classe é usada para mapear e persistir em um banco de dados, permitindo a manipulação dos dados relacionados a administradores de um museu.

Figura 18 - Atributos da classe AdmMuseu do package domain

```

1 package br.com.mcpac.museu.ciencias.domain;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6
7 //import java.time.LocalDate;[]
8
9 import lombok.AllArgsConstructor;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Entity
17
18 public class AdmMuseu {
19     @Id
20     @Column(length = 11, nullable = false)
21     private int telefone;
22
23     @Column(length = 50, nullable = false, unique = true)
24     private String nome;
25
26     @Column(length = 50, nullable = false, unique = true)
27     private String endereco;
28
29     @Column(length = 50, nullable = false, unique = true)
30     private String email;
31
32     @Column(length = 20, nullable = false, unique = true)
33     private String senha;
34
35     //Metodos Getters e Setters
36
37     public int getTelefoneAdm() {
38         return telefone;
39     }

```

Fonte: Elaborado pelo autor

A figura 19 apresenta os métodos Getters e Setters da classe AdmMuseu. Esses métodos permitem acessar e modificar os valores dos atributos da classe de forma controlada, seguindo o princípio de encapsulamento.

Figura 19 - Métodos Getters e Setters da class AdmMuseu do package domain

```

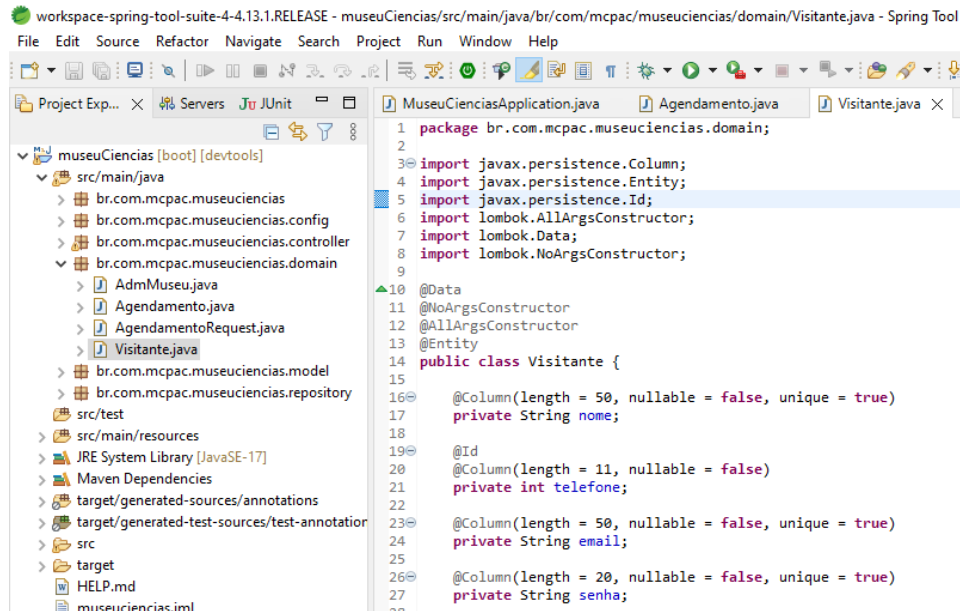
workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/domain/AdmMuseu
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit JUnit
museuCienciasApplication.java AdmMuseu.java
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
      AdmMuseuController.java
      AgendamentoController.java
      ResourceNotFoundException.java
      VisitanteController.java
    br.com.mcpac.museuciencias.domain
      AdmMuseu.java
      Agendamento.java
      AgendamentoRequest.java
      Visitante.java
    br.com.mcpac.museuciencias.model
    br.com.mcpac.museuciencias.repository
  src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotator
  src
  target
  HELP.md
  museuciencias.iml
  mvnw
  mvnw.cmd
  pom.xml
37 public int getTelefoneAdm() {
38     return telefone;
39 }
40 }
41 public void setTelefoneAdm(int telefone) {
42     this.telefone = telefone;
43 }
44 }
45 public String getNomeAdm() {
46     return nome;
47 }
48 }
49 public void setNomeAdm(String nome) {
50     this.nome = nome;
51 }
52 public void setEndereco(String endereco) {
53     this.endereco = endereco;
54 }
55 }
56 public String getEndereco() {
57     return endereco;
58 }
59 }
60 public void setEmailAdm(String email) {
61     this.email = email;
62 }
63 }
64 public String getEmailAdm() {
65     return email;
66 }
67 }
68 public void setSenhaAdm(String senha) {
69     this.senha = senha;
70 }
71 }
72 public String getSenhaAdm() {
73     return senha;
74 }
75 }

```

Fonte: Elaborado pelo autor

A figura 20 apresenta a classe Visitante que representa uma entidade no banco de dados com os campos nome, telefone, email e senha, onde o telefone é definido como chave primária através da anotação @Id. Essa classe é usada para mapear e persistir dados relacionados aos visitantes do museu.

Figura 20 - Atributos da classe Visitante do package domain



```

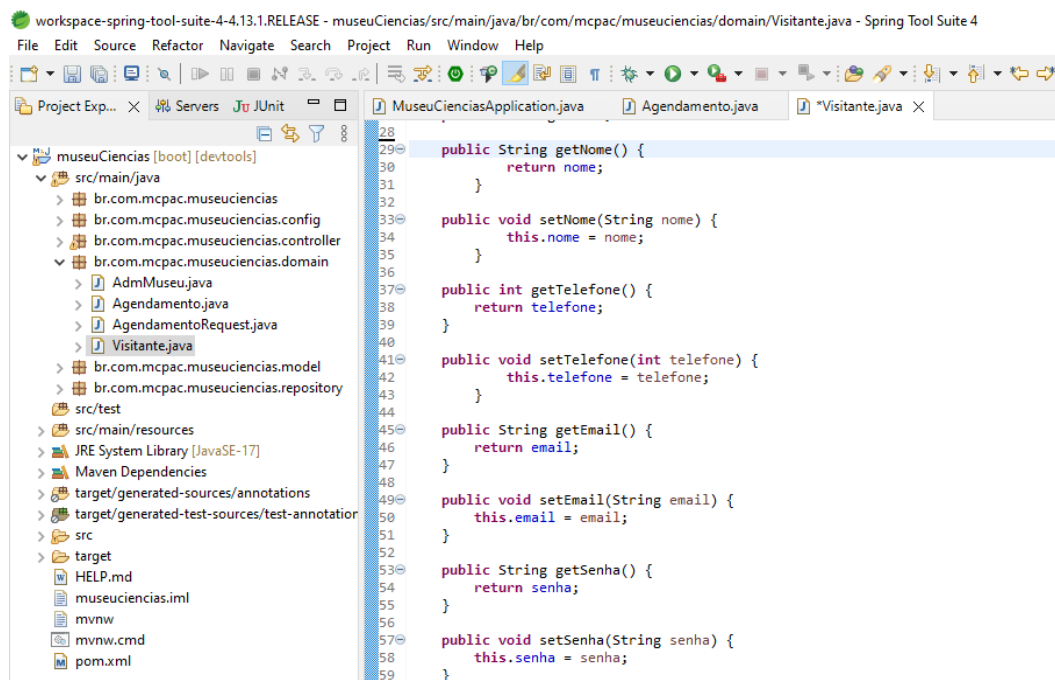
1 package br.com.mcpac.museociencias.domain;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6 import lombok.AllArgsConstructor;
7 import lombok.Data;
8 import lombok.NoArgsConstructor;
9
10 @Data
11 @NoArgsConstructor
12 @AllArgsConstructor
13 @Entity
14 public class Visitante {
15
16     @Column(length = 50, nullable = false, unique = true)
17     private String nome;
18
19     @Id
20     @Column(length = 11, nullable = false)
21     private int telefone;
22
23     @Column(length = 50, nullable = false, unique = true)
24     private String email;
25
26     @Column(length = 20, nullable = false, unique = true)
27     private String senha;
28
29 }

```

Fonte: Elaborado pelo autor

A figura 21 apresenta os métodos Getters e Setters da classe Visitante, usados para acessar e modificar os valores dos atributos da classe de forma controlada, seguindo o princípio de encapsulamento.

Figura 21 - Métodos Getters e Setters da classe Visitante do package domain



```

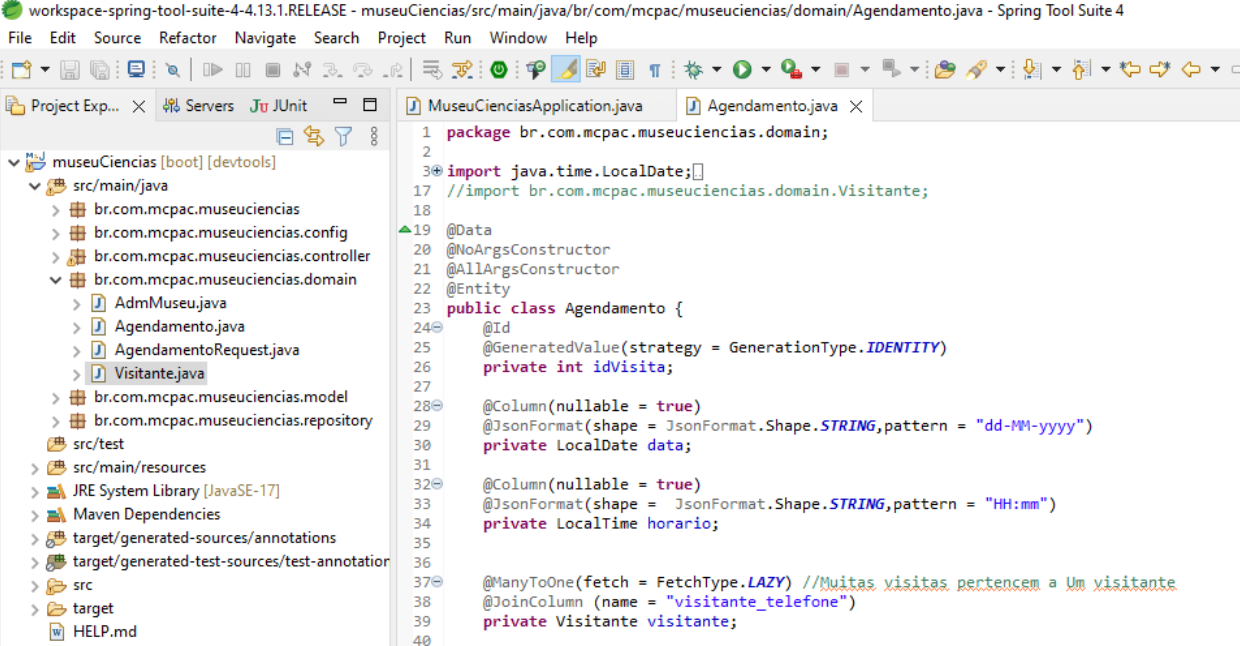
28
29 public String getNome() {
30     return nome;
31 }
32
33 public void setNome(String nome) {
34     this.nome = nome;
35 }
36
37 public int getTelefone() {
38     return telefone;
39 }
40
41 public void setTelefone(int telefone) {
42     this.telefone = telefone;
43 }
44
45 public String getEmail() {
46     return email;
47 }
48
49 public void setEmail(String email) {
50     this.email = email;
51 }
52
53 public String getSenha() {
54     return senha;
55 }
56
57 public void setSenha(String senha) {
58     this.senha = senha;
59 }

```

Fonte: Elaborado pelo autor

A figura 22 apresenta os atributos da classe Agendamento. Esses atributos representam o ID do agendamento; a data do agendamento; o horário do agendamento; e o visitante associado ao agendamento, respectivamente. a anotação `@ManyToOne (fetch = FetchType.LAZY)`, define um relacionamento muitos-para-um entre Agendamento e Visitante, onde muitos agendamentos podem pertencer a um único visitante.

Figura 22 - Atributos da classe Agendamento do package domain



```

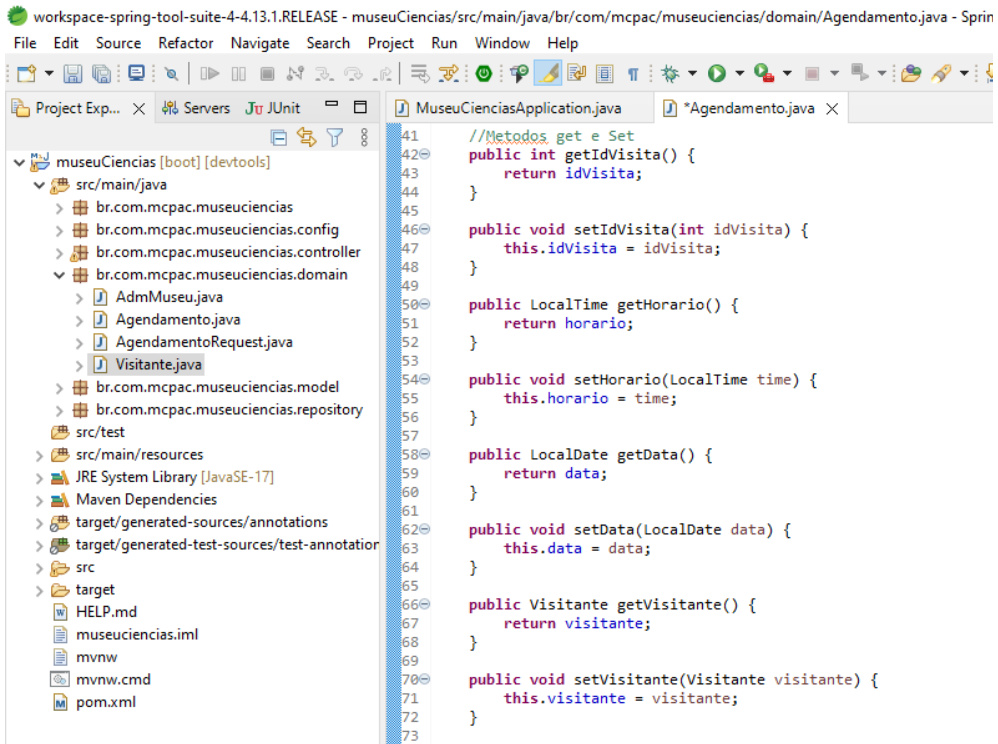
1 package br.com.mcpac.museuciencias.domain;
2
3 import java.time.LocalDate;
17 //import br.com.mcpac.museuciencias.domain.Visitante;
18
19 @Data
20 @NoArgsConstructor
21 @AllArgsConstructor
22 @Entity
23 public class Agendamento {
24     @Id
25     @GeneratedValue(strategy = GenerationType.IDENTITY)
26     private int idVisita;
27
28     @Column(nullable = true)
29     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
30     private LocalDate data;
31
32     @Column(nullable = true)
33     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "HH:mm")
34     private LocalTime horario;
35
36
37     @ManyToOne(fetch = FetchType.LAZY) //Muitas visitas pertencem a Um visitante
38     @JoinColumn(name = "visitante_telefone")
39     private Visitante visitante;
40

```

Fonte: Elaborado pelo autor

A figura 23 apresenta os métodos Getters e Setters para os atributos da classe Agendamento. Os métodos `getIdVisita()`, `getHorario()`, `getData()`, `getVisitante()` retornam os valores dos atributos. Enquanto os métodos `setIdVisita()`, `setHorario()`, `setData()`, `setVisitante()` definem o valor dos atributos com base no argumento passado.

Figura 23 - Métodos Getters e Setters da classe Agendamento do package domain



```

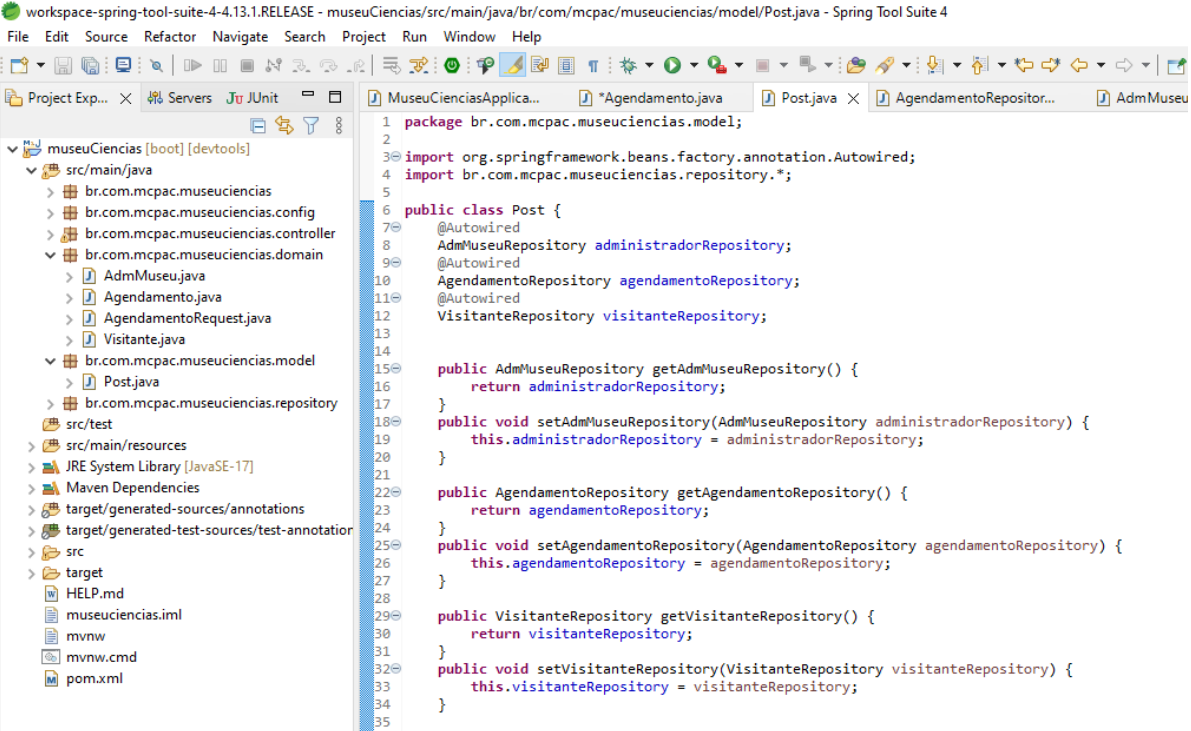
41 //Métodos get e Set
42 public int getIdVisita() {
43     return idVisita;
44 }
45
46 public void setIdVisita(int idVisita) {
47     this.idVisita = idVisita;
48 }
49
50 public LocalTime getHorario() {
51     return horario;
52 }
53
54 public void setHorario(LocalTime time) {
55     this.horario = time;
56 }
57
58 public LocalDate getData() {
59     return data;
60 }
61
62 public void setData(LocalDate data) {
63     this.data = data;
64 }
65
66 public Visitante getVisitante() {
67     return visitante;
68 }
69
70 public void setVisitante(Visitante visitante) {
71     this.visitante = visitante;
72 }
73

```

Fonte: Elaborado pelo autor

A figura 24 apresenta o código da classe Post.java localizada no package br.com.mcpac.museuciencias.model. Essa classe recebe uma instância dos repositórios AdmMuseuRepository, AgendamentoRepository e VisitanteRepository. Os métodos getters e setters permitem acessar e modificar essas referências, permitindo o acesso aos repositórios nos lugares onde o objeto Post é utilizado.

Figura 24 - Arquivo Post.java do package model



```
workspace-spring-tool-suite-4-4.13.1.RELEASE - museuCiencias/src/main/java/br/com/mcpac/museuciencias/model/Post.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Exp... Servers JUnit
museuCiencias [boot] [devtools]
  src/main/java
    br.com.mcpac.museuciencias
    br.com.mcpac.museuciencias.config
    br.com.mcpac.museuciencias.controller
    br.com.mcpac.museuciencias.domain
      AdmMuseu.java
      Agendamento.java
      AgendamentoRequest.java
      Visitante.java
    br.com.mcpac.museuciencias.model
      Post.java
    br.com.mcpac.museuciencias.repository
    src/test
  src/main/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotator
  src
  target
    HELP.md
    museuciencias.iml
    mvnw
    mvnw.cmd
    pom.xml

1 package br.com.mcpac.museuciencias.model;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import br.com.mcpac.museuciencias.repository.*;
5
6 public class Post {
7     @Autowired
8     AdmMuseuRepository administradorRepository;
9     @Autowired
10    AgendamentoRepository agendamentoRepository;
11    @Autowired
12    VisitanteRepository visitanteRepository;
13
14
15    public AdmMuseuRepository getAdmMuseuRepository() {
16        return administradorRepository;
17    }
18    public void setAdmMuseuRepository(AdmMuseuRepository administradorRepository) {
19        this.administradorRepository = administradorRepository;
20    }
21
22    public AgendamentoRepository getAgendamentoRepository() {
23        return agendamentoRepository;
24    }
25    public void setAgendamentoRepository(AgendamentoRepository agendamentoRepository) {
26        this.agendamentoRepository = agendamentoRepository;
27    }
28
29    public VisitanteRepository getVisitanteRepository() {
30        return visitanteRepository;
31    }
32    public void setVisitanteRepository(VisitanteRepository visitanteRepository) {
33        this.visitanteRepository = visitanteRepository;
34    }
35 }
```

Fonte: Elaborado pelo autor

6. RESULTADOS E DISCUSSÕES

Este estudo abordou a implementação de uma API REST para agendamento online de visitas no museu MCPAC. O modelo desenvolvido pelo estudo foi construído seguindo os conceitos de engenharia de software, programação orientada a objetos, banco de dados relacional, e desenvolvimento Web, fornecendo para o futuro, as informações necessárias para homologação do projeto, assim como a realização dos testes de software. Durante o desenvolvimento, uma das dificuldades encontradas foi a obtenção da chave estrangeira de uma entidade por meio das classes Java. Apesar de seguir as melhores diretrizes recomendadas pelo Spring Boot, foi constatado que não era possível passar apenas o valor da chave estrangeira, como seria intuitivo. Em vez disso, a regra de negócio fornece o objeto completo da classe relacionada. Essa dificuldade levantou questões sobre a forma correta de obter e manipular as chaves estrangeiras no contexto do Spring Boot, especialmente devido à falta de documentação clara sobre esse aspecto. Para contornar esse problema, foi criada uma classe adicional chamada *'AgendamentoRequest'* para fins de teste. Essa classe simplificada continha apenas os atributos necessários para o agendamento, excluindo a classe relacionada *'Visitante'*. Dessa forma, pudemos utilizar o campo *'visitanteld'* como a representação simplificada da chave estrangeira como uma *string*. No entanto, essa solução alternativa trouxe novos desafios. Foram enfrentados problemas ao lidar com os dados do tipo *LocalDate* e *LocalTime* no contexto de conversões de formato. Embora utilizado a anotação *@JsonFormat* para definir os formatos esperados para os campos *'data'* e *'horario'*, foi necessário garantir que os dados fornecidos estivessem no formato correto antes da persistência no banco de dados. Além disso, para persistir corretamente os dados, tivemos que realizar conversões dos campos *'data'* e *'horario'* da classe de teste *'AgendamentoRequest'* para os tipos *LocalDate* e *LocalTime*, respectivamente, antes de atribuí-los ao objeto *'Agendamento'*. Isso envolveu o uso de métodos como *'LocalDate.parse()'* e *'LocalTime.parse()'* para garantir a conversão adequada. Também destacamos a importância de lidar com exceções apropriadas, como *'DateTimeParseException'*, para tratar casos em que os formatos não eram válidos. Embora tenham sido encontradas soluções para contornar essas dificuldades específicas, é importante ressaltar que a falta de clareza na documentação e nos

recursos relacionados ao Spring Boot referentes à obtenção de chaves estrangeiras pode ser um obstáculo significativo para outros desenvolvedores que enfrentam problemas semelhantes.

Esse estudo destaca a necessidade de melhorar a documentação e os recursos disponíveis, fornecendo orientações mais claras e exemplos práticos para lidar com casos específicos, como a obtenção de chaves estrangeiras no contexto do Spring Boot. É recomendado que a equipe de desenvolvimento do Spring Boot e a comunidade trabalhem em conjunto para abordar as lacunas existentes. Isso pode ser alcançado por meio da melhoria da documentação fornecendo exemplos mais precisos que abordem os casos comuns enfrentados pelos desenvolvedores. Ao fazer isso, será possível reduzir a curva de aprendizado e facilitar o trabalho dos desenvolvedores, permitindo uma implementação mais eficiente e eficaz de aplicações que utilizam o Spring Boot, inclusive no que diz respeito à obtenção e manipulação de chaves estrangeiras. Espera-se que essas experiências e contribuições possam auxiliar na evolução contínua do Spring Boot, proporcionando um ambiente mais amigável para os desenvolvedores e incentivando a adoção dessa tecnologia para o desenvolvimento de aplicações mais eficientes.

7. CONSIDERAÇÕES FINAIS

O sistema mostra-se uma ferramenta em potencial, considerando que à medida que o projeto avança novos requisitos são considerados para implementações futuras. O sistema tem o potencial de otimizar o tempo dos responsáveis em organizar as visitas e exposições, e também garante um controle maior para que não ocorram conflitos entre datas. Em trabalhos futuros, o projeto pretende avançar para implementação do *front-end* utilizando a tecnologia do ReactJS, observando questões de acessibilidade. Uma possível abordagem seria a integração da biblioteca VLibras, que oferece recursos de tradução e interpretação em Libras (Língua Brasileira de Sinais) para pessoas com deficiência auditiva. Dessa forma, seria possível garantir que as páginas do sistema sejam inclusivas para um público mais amplo. Além disso, serão realizados testes abrangentes para verificar a conformidade com os requisitos estabelecidos e assegurar a qualidade e a confiabilidade do sistema antes de sua virtualização em ambiente de Web.

REFERÊNCIAS

ARAÚJO, Claudiomar Pereira de. **Projeto e Implementação de um Serviço Web RESTful com Técnicas de Segurança**. 2018. 76 f. Trabalho de conclusão de curso (Bacharelado em Ciência da Computação) - Universidade Federal da Paraíba, João Pessoa, 2018. Disponível em: <<https://repositorio.ufpb.br/jspui/bitstream/123456789/15636/1/CPA20112018.pdf>>. Acesso em: 17 mai. 2023.

BATISTA, Mateus Alfredo; BASTOS, William Carvalho. **Desenvolvendo aplicações web baseadas em microserviços utilizando o framework Spring**. 2017. 124 f. Trabalho de conclusão de curso (Bacharel em Sistemas da Informação) - Universidade do Sul de Santa Catarina, Palhoça, 2017. Disponível em: <<https://repositorio.animaeducacao.com.br/handle/ANIMA/11003>>. Acesso em: 11 jun. 2022.

BOAGLIO, Fernando. **Spring Boot - Acelere o desenvolvimento de microsserviços**. Editora Casa do Código, 2017. Disponível em: <https://www.academia.edu/41263289/spring_boot_acelere_o_desenvolvimento_de_microsservicos>. Acesso em: 17 fev. 2023.

BOEHM, B., A View of 20th and 21st Century Software Engineering, ICSE 2006.

BOEG, Jesper. **Kanban em 10 passos**: Otimizando o fluxo de trabalho em sistemas de entrega de software. Traduzido. [S.l.]: [s.n.], 2012. Disponível em: <<https://www.infoq.com/br/minibooks/priming-kanban-jesper-boeg/>>. Acesso em: 07 jul. 2023.

CARVALHO, Vinícius. **MySQL: Comece com o principal banco de dados open source do mercado**. São Paulo: Casa do Código, 2015. Disponível em: <https://www.academia.edu/23820095/MySQL_Comece_com_o_principal_banco_de_dados_open_source_do_mercado>. Acesso em: 13 mar. 2023.

COSLOVSKY, Vinícius. **Vire o jogo com Spring Framework**. São Paulo: Casa do Código, 2017. Disponível em: <<https://books.google.com.br/books?hl=pt-BR&lr>>. Acesso em 28 de abr. 2023.

DELFINO, Sérgio Roberto. **[Desenvolvimento Back-end em Java com Spring]**. YouTube, 27 jan. 2020. Disponível em: <<https://www.youtube.com/watch?v=Dk0uZ1SrY2M&list>>. Acesso em: 15 fev. 2022.

DEV MEDIA. **Aprendendo Java com JDBC**. DevMedia, 2013a. Disponível em: <<https://www.devmedia.com.br/aprendendo-java-com-jdbc/29116>>. Acesso em: 20 maio 2023.

DEV MEDIA. **Guia Completo de SQL**. DevMedia, 2020. Disponível em: <<https://www.devmedia.com.br/guia/guia-completo-de-sql/38314>>. Acesso em: 02 de mar. 2023.

DEVMEDIA. **Introdução ao MySQL**. DevMedia, 2013b. Disponível em: <<https://www.devmedia.com.br/introducao-ao-mysql/27799>>. Acesso em: 02 jul. 2023.

DEVMEDIA. **Introdução Prática ao Spring Framework com Uso de Anotações**. DevMedia, 2013c. Disponível em: <<https://www.devmedia.com.br/introducao-pratica-ao-spring-framework-com-uso-de-anotacoes/27859#:~:text=As%20anota%C3%A7%C3%B5es%20s%C3%A3o%20funcionalidades%20introduzidas,toda%20configura%C3%A7%C3%A3o%20era%20via%20XML>>. Acesso em: 04 jul. 2023.

FASYNXML. **Class ObjectMapper**. Fasyncxml, 2018. Disponível em: <<https://fasterxml.github.io/jackson-databind/javadoc/2.9/com/fasterxml/jackson/databind/ObjectMapper.html>>. Acesso em: 06 abr. 2023.

FIELDING, Roy. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. 200p. Dissertação de Doutorado. University of California, Irvine. Disponível em: <<https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>. Acesso em: 18 de mai. 2023.

GIBIKOSKI, Denis. **Sistema web para reserva de salão de festas em condomínios**. 2019. 43 f. Monografia (Especialização em Engenharia de Software) - Universidade Tecnológica Federal do Paraná, Pato Branco. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/25927/1/PB_CEETJ_V_2020_08.pdf>. Acesso em: 14 mai. 2023.

Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE). **IEEE Std. 830 – 1998: IEEE Recommended Practice for Software Requirements Specifications**. [S.l.]: IEEE, 2003. Disponível em: <<http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf>>. Acesso em: 18 jun. 2023.

INSTITUTO FEDERAL DO SERTÃO PERNAMBUCANO. Museu de Ciência Professor Antônio Carneiro. **Inovação, divulgação científica e popularização da ciência no Sertão Central do estado de Pernambuco**. Salgueiro, 20 p. Não publicado.

JACOBSEM, Vinícius Louzada. **Desenvolvimento de Sistema Web para Busca de Publicações na Base de Documentos da Polícia Militar do Espírito Santo**. 2021. 67 f. Trabalho de conclusão de curso (Bacharel em Sistemas de Informação) - Instituto Federal do Espírito Santo, Serra, 2021. Disponível em: <<https://repositorio.ifes.edu.br/handle/123456789/1601?show=full>>. Acesso em: 27 mar. 2023.

JAVA, Desenvolvimento. **O que é 'Web Service' com Spring Boot - Parte 1**. YouTube, 14 jun. 2021. Disponível em: <<https://www.youtube.com/watch?v=xXESCLAx8MU>>. Acesso em 09 nov. 2022.

JSON. **Apresentando o JSON: Documentação oficial do JSON**. Json.org, 2023.

Disponível em : <<https://www.json.org/json-en.html>>. Acesso em 06 abr. 2023.

JÚNIOR, José Ronaldo Nogueiro Fonseca. **Estudo de Caso Usando Spring Framework para Criação de Aplicações Web com J2EE**. 2007. 40 f. Monografia (Especialização em Sistemas de Computação para Web) – Universidade Federal de Santa Maria, Santa Maria, 2007.

JÚNIOR, Manoel Nazaro da Silva; SOUSA, Hercílio de Medeiros. **Proposta de desenvolvimento de aplicativo mobile voltado para treinos em casa**. Revista Eletrônica Científica Inovação e Tecnologia, v. 1, n. 1, p. 11-26, 2018. Disponível em: <<https://www.iesp.edu.br/sistema/uploads/arquivos/publicacoes/proposta-de-desenvolvimento-de-aplicativo-mobile-voltado-para-treinos-em-casa-autor-silva-junior-manoel-nazaro-da-.pdf>>. Acesso em: 04 de mai. 2023.

MALAQUIAS, Matheus. **Projeto Lombok: escrevendo menos código em Java**. Medium, 2018. Disponível em: <<https://medium.com/collabcode/projeto-lombok-escrevendo-menos>>. Acesso em: 04 jul, 2023.

MARTINS, Thamyris Furquim; JUNIOR, Gilmar T. **HiPet – Management System of Pets**. Disponível em: <<https://www.anais.ueg.br/index.php/jaueg/article/view/7321>>. Acesso em 14 de fev. 2022.

NASCIMENTO, Anderson. **O que é API ?**. Canaltech, 2014. Disponível em: <<https://canaltech.com.br/software/o-que-e-api/>>. Acesso em: 02 jul. 2023.

ORACLE. **Java SE Documentation. 8.0**. Redwood City: Oracle, 2023a. Disponível em: <<https://docs.oracle.com/javase/8/docs/>>. Acesso em: 08 abr. 2023.

ORACLE. **Java Platform, Enterprise Edition (Java EE): Documentation, 2023**. Oracle, 2023b. Disponível em: <<https://docs.oracle.com/javaee/7/index.html>>. Acesso em: 08 abr. 2023.

ORACLE. **MySQL 8.0 Reference Manual**. Oracle, 2023c. Disponível em: <<https://dev.mysql.com/doc/refman/8.0/en/mysql-enterprise-security.html>>. Acesso em: 20 de jul. 2023.

ORACLE. **O que é um MySQL Database?** Oracle, 2023d. Disponível em: <<https://www.oracle.com/br/database/what-is-database/#link7>>. Acesso em: 02 jul. 2023.

RAO, Rakshith; SWAMY, Sr. **Review on spring boot and spring webflux for reactive web development**. v. 7, 05 2020. Disponível em: <https://www.researchgate.net/publication/341151097_Review_on_Spring_Boot_and_Spring_Webflux_for_Reactive_Web_Development>. Acesso em: 20 fev. 2023.

KANBANIZE. **O que é Kanban Explicado para Iniciantes**. Kanbanize, 2023. Disponível em: <<https://kanbanize.com/pt/recursos-kanban/primeiros-passos/o-que-e-kanban>>. Acesso em: 07 jul. 2023.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003. Disponível em: <https://docente.ifrn.edu.br/olivianeta/disciplinas/copy_of_historia-i/historia-ii/china-e-india/view>. Acesso em: 16 jun. 2023.

LINHARES, Maurício. **Introdução ao hibernate 3**. Grupo de usuários Java. Disponível em: <https://www.cin.ufpe.br/~rdma/documentos/intruducaao_hibernate3_guj.pdf>. Acesso em: 13 dez. 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução Kalinka Oliveira; Ivan Bosnic. 9. ed. São Paulo: Pearson Education do Brasil, 2011. 530 p. Tradução de: Software Engineering. Disponível em: <<https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf>>. Acesso em: 25 abr. 2023.

SOARES, Sílvia. **Engenharia de Software - Aula 5: Modelagem de Sistemas**. Universidade Federal do Paraná, 2005. Disponível em: <<https://www.inf.ufpr.br/silvia/ES/projeto/aulas/aula5.pdf>> . Acesso em: 02 de jul. 2023.

SOUSA, Alberto. **Guia das annotations do Spring**. Domine o spring, 2016. Disponível em: <<https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/#:~:text=Anota%C3%A7%C3%A3o%20utilizada%20para%20marcar%20o,o%20seu%20construtor%20com%20argumentos>>. Acesso em: 04 jul. 2023.

SOUSA, João Paulo; PEREIRA, Rodrigo da Silveira. **Estudo de caso usando Spring Framework para a criação de aplicações web com J2EE**. 2007. 40 f. Monografia (Especialização em Desenvolvimento de Sistemas para Web) - Universidade Federal de Santa Maria, Santa Maria, 2007. Disponível em: <<https://repositorio.ufsm.br/handle/1/1725>>. Acesso em: 29 abr. 2023.

SPRING FRAMEWORK. **Spring Documentation**. Disponível em: <<https://docs.spring.io/spring-framework/docs/current/reference/html/>>. Acesso em: 15 abr. 2023.

SPRING FRAMEWORK. **Spring Web Documentation**. Disponível em: <<https://docs.spring.io/spring-framework/docs/current/reference/html/web.html>>. Acesso em: 01 jul. 2023.

SPRING IO. **Spring Boot Reference Documentation**. Disponível em: <<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>>. Acesso em: 07 abr. 2023.

SPRING. **Spring Data 2021.2.5**. Disponível em: <<https://spring.io/projects/spring-data>>. Acesso em 07 abr. 2023.

TEIXEIRA, Pedro Henrique Faria. **Uma API REST para a contratação de profissionais na aplicação Severino**. 2021. 61 f. Trabalho de Conclusão de Curso

(Graduação em Ciência da Computação) – Universidade Federal de Uberlândia, Uberlândia, 2021. Disponível em: <<https://repositorio.ufu.br/handle/123456789/32435>>. Acesso em: 28 abr. 2023.

VALENTE, M. T. **Engenharia de software moderna: princípios e práticas para desenvolvimento de software com produtividade.** In: ENGENHARIA de Software Moderna. [S. l.], 2020. Disponível em: <https://engsoftmoderna.info>. Acesso em: 07 abr. 2023.

WALLS, Craing. **Spring in Action, Sixth Edition.** Manning Publications, 2022. Disponível em: <https://books.google.com.br/books?id=2zVbEAAAQBAJ&dq=%22Spring+in+Action%22%2B%22Craig+Walls%22&lr=&hl=pt-BR&source=gbs_navlinks_s>. Acesso em 28 jun. 2023.

XAVECODING. **Introdução ao Spring MVC.** YouTube, 8 de jul. de 2021. Disponível em: <<https://www.youtube.com/playlist?list=PL3ZsII15yo2ppY0GsRFDjRdHZAUuPnQ6M>>. Acesso em: 20 jul. 2022.