INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO – CAMPUS SALGUEIRO

VISÃO DO PRODUTO

Versão 0.1

28/07/2025

Caren Beatriz Silva Oliveira

Daiane Maria dos Santos Ribeiro

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO – CAMPUS SALGUEIRO

HISTÓRICO

Data	Versão	Descrição	Autor
28/07/2025	0.1	Documento de visão inicial	Caren Daiane

SUMÁRIO

1. INTRODUÇÃO

O presente documento tem como objetivo principal apresentar a visão do sistema que será desenvolvido na disciplina de Programação Web II, do curso superior de Tecnologia em Sistemas para Internet do IFSertãoPE campus Salgueiro. O sistema em questão consiste em uma *Application Programming Interface* (API) para reservas de salas, implementada com Java EE, utilizando os frameworks *Spring Boot, Spring Data JPA* e *Hibernate*, além de integrar o banco de dados *MySQL*. A referida API será responsável por fornecer as funcionalidades necessárias para o agendamento, consulta e gerenciamento de reservas de salas, garantindo organização, disponibilidade e facilidade de uso para os usuários.

Sendo assim, esse documento de visão apresenta detalhes sobre a modelagem do sistema, uso da arquitetura REST, padrões do projeto, e as tecnologias utilizadas.

1.1 Escopo

O sistema permitirá que os usuários cadastrem salas, reservem horários disponíveis, cancelem reservas e consultem a disponibilidade dos espaços. O público-alvo são servidores e administradores da instituição de ensino que necessitam reservar espaços físicos para reuniões, aulas, palestras, eventos entre outros.

As funcionalidades principais do sistema são:

- Cadastro de usuários e autenticação.
- Cadastro e listagem de salas.
- Realização, alteração e cancelamento de reservas.
- Consulta da disponibilidade de horários.
- Controle de permissões de acesso (administrador e usuário comum).

1.2 Definições, Acrônimos e Abreviações

API : Application Programming Interface
CRUD : Create, Read, Update, Delete
DTO: Data Transfer Object
Java EE : Java Platform, Enterprise Edition
JPA: Java Persistence API
JSON:JavaScript Object Notation
JWT: JSON Web Token
ORM: Object-Relational Mapping
REST: Representational State Transfer
RF : Requisito Funcional
RNF : Requisito Não Funcional

SQL: Structured Query Language
UC : User Case (Caso de Uso)
URLs: Uniform Resource Locator
EX: exceções

2. MODELAGEM DE SISTEMAS

A modelagem do sistema API REST para reserva de salas foi realizada com base na identificação das principais funcionalidades que devem estar presentes no referido sistema, utilizando os conceitos de orientação a objetos e arquitetura REST. Abaixo, são apresentados os principais casos de uso e a estrutura do modelo de objetos, que servem de base para o desenvolvimento da API.

2.1.1 Requisitos Específicos

Nesse subtópico serão apresentadas duas tabelas que descrevem os requisitos funcionais (tabela 1) e os requisitos não funcionais (tabela 2), nelas estão os campos sigla, nome, descrição, prioridade e a dependência de outro requisito. No campo "prioridade" será definido qual das três categorias o requisito correspondente se adequa no sistema, sendo eles:

- Desejável
    - Requisito que sua ausência não compromete as funcionalidades principais do sistema;
    - Sua utilização resultaria em mais utilidades no sistema, é implementada caso haja tempo ou recurso.
- Importante
    - Os requisitos desta categoria proporcionarão bons diferenciais para o sistema, assim ele se tornará mais agradável aos usuários.
- Essencial
    - Requisitos obrigatórios, sem eles o sistema não conseguirá cumprir seus principais objetivos.

SIGLA	NOME	DESCRIÇÃO	PRIORIDADE	DEPENDE DE
RF-001	Cadastrar usuário	Possibilitar que o próprio usuário faça seu cadastro no sistema.	Essencial	—
RF-002	Autenticar Usuário	Permitir que usuários façam login com suas credenciais.	Essencial	RF001
RF-003	Cadastrar salas de reunião	Os administradores cadastrem as salas de reuniões disponíveis do prédio.	Essencial	RF002
RF-004	Consultar de Salas Disponíveis	Permiti que os usuários verifiquem qual sala está disponível em determinado horário.	Essencial	RF003
RF-005	Realizar Reserva	Permitir que usuários reservem uma sala de acordo com horário e data.	Essencial	RF004

RF-006	Cancelar Reserva	Permitir que usuários cancelem suas reservas futuras.	Importante	RF005
RF-007	Enviar notificação de Reserva	Enviar confirmação por e-mail ou sistema ao realizar uma reserva.	Importante	RF005
RF-008	Mostrar histórico de Reservas	Permitir que usuários visualizem o histórico de suas reservas realizadas.	Desejável	RF005
RF-009	Gerenciar usuários	Permitir que administradores gerenciem usuários (editar, bloquear, etc.).	Importante	RF001, RF002
RF-010	Gerar relatórios de Utilização	Gerar relatórios sobre a utilização das salas (frequência, horários, etc.)	Desejável	RF005, RF009

Tabela 1 – Requisitos funcionais.

2.1.2 Requisitos Não Funcionais

SIGLA	NOME	DESCRIÇÃO	PRIORIDADE
RNF-001	Tempo de resposta	O sistema deve fornecer uma resposta às solicitações do usuário em até 2 segundos.	Essencial
RNF-002	Disponibilidade	O sistema não pode ocultar suas funções essenciais em horário comercial.	Essencial
RNF-003	Segurança de acesso	O sistema deve usar o protocolo https.	Essencial
RNF-004	Compatibilidade com navegadores	O sistema deve funcionar nos principais navegadores utilizados atualmente.	Essencial
RNF-005	Suporte a dispositivos móveis	Deve fornecer recursividade para os usuários que acessarem pelos seus dispositivos móveis	Importante
RNF-006	Backup diário	realizar backups para não perder informações importantes que são atualizadas diariamente.	Importante
RNF-007	Fácil usabilidade	A interface deve ser intuitiva e fácil de usar.	Importante
RNF-008	Escalabilidade	Suportar a possibilidade de aumento de usuários sem perder desempenho.	Desejável
RNF-009	Registro de Logs	O sistema deve registrar em logs as ações críticas, como o usuário que realizou login, fez reservas, cancelou e quaisquer falhas que possam ocorrer.	Desejável
RNF-010	Tempo de recuperação	Em falhas graves, como queda no servidor ou banco de dados, o sistema deve ser restaurado em até 1 hora.	Importante

Tabela 2 – Requisitos não funcionais.

2.2 Casos de Uso

Os casos de uso (UC) são usados para descrever como ocorre a interação entre determinado um ator (usuário ou outro programa) e o sistema desenvolvido, como base usando os requisitos funcionais. Com base nisso, foram elaborados exemplos de UC que serão tratados na API para reservas de salas.

Lista de Casos de Uso

Sigla	Nome
UC01	Cadastrar Usuário
UC02	Autenticar Usuário
UC03	Cadastrar Sala
UC04	Realizar Reserva de Sala
UC05	Alterar Reserva
UC06	Cancelar Reserva
UC07	Consultar Disponibilidade

UC01: Cadastrar Usuário

Descrição: Permite que um novo usuário se registre no sistema com nome, email e senha.

Cenário Principal:

1. O usuário acessa a tela de cadastro.
2. Preenche os dados obrigatórios (nome, email e senha). **EX01**.
3. O sistema valida os dados e os salva no banco de dados.

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- EX01: Dados inválidos ou email já registrado.

UC02: Autenticar Usuário

Descrição: Permite que o usuário realize login com email e senha cadastrados.

Cenário Principal:

1. O usuário informa email e senha. **EX01**.
2. O sistema valida as credenciais.
3. Se válidas, é gerado um token de autenticação.

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- EX01: Senha ou email incorretos.

UC03: Cadastrar Sala (Administrador)

Descrição: Permite ao administrador cadastrar uma nova sala no sistema.

Cenário Principal:

1. O administrador acessa a tela de cadastro de sala.
2. Informa nome, local e capacidade. **EX01.**
3. O sistema salva a sala no banco de dados. **EX02.**

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- EX01: Dados incompletos ou inválidos.
- EX02: Sala já cadastrada.

UC04: Realizar Reserva de Sala

Descrição: Permite que o usuário selecione uma sala e reserve um horário.

Cenário Principal:

1. O usuário autentica-se no sistema.
2. O sistema apresenta as salas disponíveis.
3. O usuário escolhe uma sala e um horário livre.**EX01.**
4. O sistema confirma a reserva.

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- **EX01:** Caso o horário esteja indisponível, o sistema informa o conflito e solicita nova seleção.

UC05: Alterar Reserva

Descrição: Permite que o usuário edite uma reserva já realizada.

Cenário Principal:

1. O usuário acessa suas reservas.
2. Seleciona a reserva a ser modificada. **EX01.**

3. Escolhe novo horário disponível. **EX02.**
4. O sistema salva a alteração.

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- **EX01**: Tentativa de editar reserva de outro usuário.
- **EX02**: Novo horário indisponível.

UC06: Cancelar Reserva

Descrição: Permite que o usuário cancele uma reserva.

Cenário Principal:

1. O usuário acessa suas reservas.
2. Seleciona a reserva desejada.
3. Solicita o cancelamento. **EX01.**

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- **EX01**: Cancelamento de reserva já expirada.

UC07: Consultar Disponibilidade

Descrição: Permite que o usuário veja os horários disponíveis para uma sala.

Cenário Principal:

1. O usuário acessa a tela de busca de disponibilidade.
2. Seleciona uma sala e uma data. **EX01.**
3. O sistema exibe os horários disponíveis.

Fluxo Alternativo:

- Não há fluxo alternativo.

Exceções:

- **EX01**: Nenhuma sala disponível no período.

2.3 Modelo de Objetos

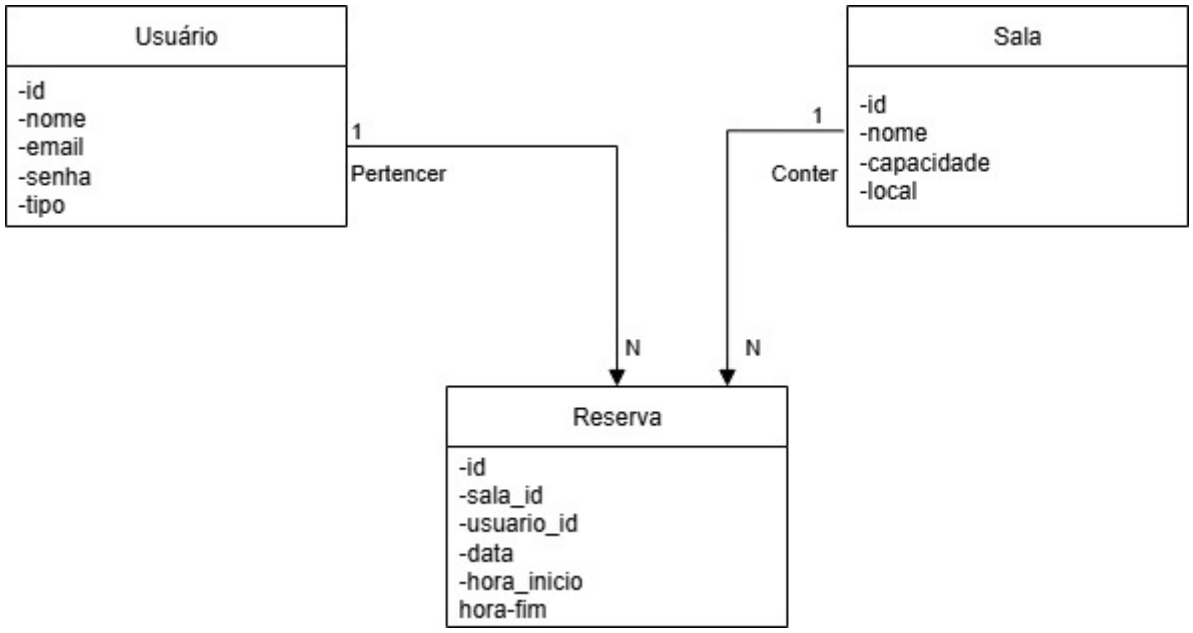
O sistema possui as seguintes entidades principais:

- **Usuário:** Atributos: id, nome, email, senha, tipo (admin/comum)
- **Sala:** Atributos: id, nome, capacidade, local
- **Reserva:** Atributos: id, sala_id, usuario_id, data, hora_inicio, hora_fim

O relacionamento entre as entidades é o seguinte:

- Um usuário pode fazer muitas reservas.
- Uma sala pode conter muitas reservas.
- Cada reserva pertence a um único usuário e a uma única sala.

A seguir apresentaremos o Diagrama de Classes UML contendo os atributos e relacionamentos, seguindo o padrão da orientação a objetos.



3. MODELO DE ARQUITETURA REST

A arquitetura REST é uma forma simples de organizar os caminhos (URLs) da nossa API. Cada funcionalidade do sistema terá um endereço específico e uma ação (como listar, cadastrar, alterar ou excluir - CRUD). Por exemplo:

- POST /usuarios — Cadastrar novo usuário
- POST /auth/login — Fazer login
- GET /salas — Listar salas
- POST /salas — Cadastrar sala (somente administrador)
- GET /reservas — Ver reservas do usuário
- POST /reservas — Criar uma reserva
- PUT /reservas/{id} — Editar uma reserva
- DELETE /reservas/{id} — Cancelar uma reserva
- GET /disponibilidade?sala_id=1&data=2025-01-01 — Verificar horários disponíveis

Os dados serão enviados e recebidos no formato JSON. Para proteger a API, será usado login com token (JWT), que permite o acesso às rotas somente após o usuário se autenticar.

4. PADRÕES DE PROJETO

Nesta API usaremos alguns padrões simples que ajudam a deixar o código mais organizado e fácil de manter, como: I) DTO (Data Transfer Object) - Serve para organizar os dados que entram e saem da API. Por exemplo, ao enviar uma reserva, usamos um objeto com apenas os dados necessários, como data e horário, sem enviar a sala inteira. II) Repository - Esse padrão é usado para separar a parte do código que fala com o banco de dados. Por exemplo, temos um ReservaRepository que cuida apenas de salvar, editar e buscar reservas no banco.

Esses padrões ajudam a deixar o projeto mais limpo e separado por responsabilidades.

5. FRAMEWORKS

5.1 Spring Framework

O **Spring Boot** ajuda a criar APIs de forma rápida e prática. Ele já vem com várias ferramentas prontas que usaremos no projeto:

- **Spring Web:** para criar os caminhos da API (rotas REST)
- **Spring Security:** para proteger as rotas com login
- **Spring Data JPA:** para facilitar a conexão com o banco de dados

Tudo isso permite focar mais nas regras do sistema e menos em configurações complicadas.

5.2 Hibernate

O **Hibernate** é uma ferramenta que facilita salvar e buscar dados no banco de dados usando objetos Java. Em vez de escrever comandos SQL, usamos classes e ele faz o trabalho por trás.

Por exemplo, ao criar uma classe Reserva, o Hibernate cuida de gravar e buscar os dados dela nas tabelas do banco.

6. REFERÊNCIAS

SPRING.IO. *Spring Boot Documentation*. Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/html/.

SPRING.IO. *Spring Security Reference*. Disponível em: https://docs.spring.io/spring-security/reference/.

HIBERNATE. *Hibernate ORM Documentation*. Disponível em: https://hibernate.org/orm/documentation/.

ORACLE. *Java Platform, Enterprise Edition (Java EE)*. Disponível em: https://docs.oracle.com/javaee/.