Autoestudo 2.0

Stack

- **Front-end:** Vue 3, TypeScript, Vuetify 3, Pinia, Vue Router, Axios
- **Back-end:** Spring Boot 3, Spring Web, Spring Security, JWT, JPA/Hibernate, Validation, Lombok
- **Banco:** PostgreSQL

---

1) Objetivo

Sistema com autenticação segura (BCrypt + JWT com **expiração de 1 hora**), gestão de **Funcionários**, **Setores** e **Supervisores**, mantendo relacionamentos (Funcionário pertence a um Setor; Setor pode ter um Supervisor responsável).

---

2) Entidades & Relacionamentos

2.1. Entidades

- **Usuario** (técnica para autenticação): `id`, `email`, `senhaHash`, `papel` (enum: `SUPERVISOR`, `FUNCIONARIO`), `ativo`, `criadoEm`, `atualizadoEm`.
- **Supervisor**: `id`, `usuarioId(FK)`, `nome`, `telefone`, `observacoes`.
- **Setor**: `id`, `nome` (único), `descricao`, `supervisorId(FK opcional)`.
- **Funcionario**: `id`, `usuarioId(FK opcional)`, `nome`, `matricula` (única), `emailCorporativo`, `setorId(FK)`, `telefone`, `ativo`.

> Observação: Manter **Usuario** separado permite logins tanto de Supervisores quanto de Funcionários (papéis distintos), sem duplicar lógica de autenticação.

2.2. Relacionamentos

- **Funcionario (N) — (1) Setor**

- **Setor (N) — (1) Supervisor** (opcional; um supervisor pode gerir vários setores)
- **Supervisor (1) — (1) Usuario** (perfil)
- **Funcionario (0/1) — (1) Usuario** (perfil, opcional se alguns funcionários não acessarem o sistema)

---

3) Regras de Segurança

- **Senha**: armazenada com **BCrypt** (Spring `BCryptPasswordEncoder`).
- **Sessão/JWT**: Access Token **expira em 1 hora** (3600s). Opcional: Refresh Token via cookie HttpOnly (exp. maior, ex.: 7 dias). Caso não use refresh, o usuário reloga ao expirar.
- **Autorização** por papéis: rotas administrativas restritas a `SUPERVISOR`.
- **CORS** habilitado para o domínio do front-end.
- **Stateless** (sem sessão de servidor).

---

4) Convenções da API

- Base path: `/api/v1`
- **Paginação:** `?page=0&size=20&sort=nome,asc`
- **Busca:** `?q=term`
- **Padrões de resposta:** envelope `{ data, meta, errors }`
- **Ids:** UUID recomendados

---

5) Endpoints Sugeridos

5.1 Autenticação

Método	Caminho	Papel	Descrição
POST	`/auth/login`	Público	Login com `email` e `senha`. Retorna `accessToken` (1h) e, se adotado, `refreshToken` (cookie HttpOnly).
POST	`/auth/refresh`	Público	Gera novo `accessToken` usando `refreshToken` (se implementado).
POST	`/auth/logout`	Autenticado	Invalida o refresh atual (se existir).
GET	`/auth/me`	Autenticado	Retorna usuário logado e papel (para montar o menu/guardas).

Request `POST /auth/login`

```
{
  "email": "ana@empresa.com",
  "senha": "Secr3t!"
}
```

Response 200

```
{
  "data": {
    "accessToken": "<jwt>",
    "expiresIn": 3600,
    "user": { "id": "uuid", "email": "ana@empresa.com", "papel": "SUPERVISOR" }
  }
}
```

5.2 Supervisores

Método	Caminho	Papel	Descrição
GET	`/supervisores`	SUPERVISOR	Lista com paginação e `q`.
POST	`/supervisores`	SUPERVISOR	Cria supervisor (+ opcional criar `Usuario` vinculado).
GET	`/supervisores/{id}`	SUPERVISOR	Detalhe.

PUT	`/supervisores/{id}`	SUPERVISOR	Atualiza todos os campos.
PATCH	`/supervisores/{id}`	SUPERVISOR	Atualização parcial (ex.: `telefone`).
DELETE	`/supervisores/{id}`	SUPERVISOR	*Soft delete* (marca `ativo=false`).
GET	`/supervisores/{id}/setores`	SUPERVISOR	Lista setores sob responsabilidade.

5.3 Setores

Método	Caminho	Papel	Descrição
GET	`/setores`	Autenticado	Lista setores (`q`, paginação).
POST	`/setores`	SUPERVISOR	Cria setor (nome único).
GET	`/setores/{id}`	Autenticado	Detalhe.
PUT	`/setores/{id}`	SUPERVISOR	Atualiza setor.
DELETE	`/setores/{id}`	SUPERVISOR	*Soft delete*.
PATCH	`/setores/{id}/supervisor`	SUPERVISOR	Atribui/Remove supervisor do setor.
GET	`/setores/{id}/funcionarios`	Autenticado	Funcionários do setor.

Exemplo `PATCH /setores/{id}/supervisor`

```
{ "supervisorId": "uuid" }
```

5.4 Funcionários

Método	Caminho	Papel	Descrição
GET	`/funcionarios`	Autenticado	Lista (filtros: `setorId`, `ativo`, `q`).

POST	`/funcionarios`	SUPERVISOR	Cria funcionário (+ opcional criar `Usuario`).
GET	`/funcionarios/{id}`	Autenticado	Detalhe.
PUT	`/funcionarios/{id}`	SUPERVISOR	Atualiza funcionário.
PATCH	`/funcionarios/{id}/setor`	SUPERVISOR	Move funcionário de setor.
PATCH	`/funcionarios/{id}/status`	SUPERVISOR	Ativa/Inativa.
DELETE	`/funcionarios/{id}`	SUPERVISOR	*Soft delete*.

Exemplo `PATCH /funcionarios/{id}/setor`

```
{ "setorId": "uuid" }
```

---

6) Schemas (DTOs)

6.1. SupervisorDTO

```
{
  "id": "uuid",
  "nome": "Ana Souza",
  "telefone": "+55 85 99999-0000",
  "usuarioId": "uuid"
}
```

6.2. SetorDTO

```
{
  "id": "uuid",
  "nome": "Financeiro",
  "descricao": "Pagamentos e contabilidade",
  "supervisorId": "uuid|null"
}
```

6.3. FuncionarioDTO

```
{
  "id": "uuid",
```

```
  "nome": "Carlos Lima",
  "matricula": "F12345",
  "emailCorporativo": "carlos@empresa.com",
  "telefone": "+55 85 90000-0000",
  "setorId": "uuid",
  "ativo": true
}
```

---

7) Validações (exemplos)

- `Usuario.email`: formato válido e único
- `Usuario.senha`: mínimo 8 caracteres, maiúscula, minúscula, dígito e especial
- `Setor.nome`: obrigatório, único (case-insensitive)
- `Funcionario.matricula`: obrigatória, única
- Mudança de `setorId`: setor precisa existir e estar ativo

---

8) Back-end (Spring Boot) — Diretrizes

- **Security**: `BCryptPasswordEncoder`, filtro JWT, `@PreAuthorize("hasRole('SUPERVISOR')")` onde couber.
- **Exceptions** mapeadas para `{ errors: [{ code, message, field? }] }`.
- **JPA**: `@Column(unique = true)` em `Setor.nome` e `Funcionario.matricula`.
- **Auditoria**: `@CreatedDate`, `@LastModifiedDate`.
- **Soft delete**: campo `ativo` + filtros de consulta (Specification/Repository).

---

9) Front-end (Vue 3 + TS + Vuetify)

9.1 Estrutura Sugerida

```
src/
  api/          # axios instance, services por recurso
  stores/       # Pinia (auth, funcionarios, setores, supervisores)
  router/       # rotas + guards
  types/        # TS interfaces (DTOs)
  views/        # páginas
  components/   # componentes reutilizáveis
```

9.3 Pinia (auth store)

- Guarda `accessToken`, `expiresAt` e `user` em memória (ou `sessionStorage`).
- **Ao expirar (1h)**: interceptador 401 do Axios → limpa store e **redireciona para `/login`**.
- *Opcional*: tentar `refresh()` se cookie HttpOnly presente.

9.4 Guardas de Rota

- `beforeEach`: se rota exige auth e `token` ausente/expirado → `/login`.
- Se rota exige papel `SUPERVISOR`, verificar `user.papel`.
