GRUPO 07

INTEGRANTES

BRUNA LUIZA DA SILVA GONÇALVES
DANIEL ANDRADE
JULIANA RODRIGUES MATOS
KEVEN DOS SANTOS COSTA
LUCAS DE OLIVEIRA MAFRA
NAYANE RODRIGUES MATOS SANTOS OI

PROJETO INTEGRADOR
ESCOPO DO PROJETO

TEMA: SISTEMAS DE CADASTRO RH

São Paulo
Dezembro de 2025

GRUPO 07

INTEGRANTES

BRUNA LUIZA DA SILVA GONÇALVES
DANIEL ANDRADE
JULIANA RODRIGUES MATOS
KEVEN DOS SANTOS COSTA
LUCAS DE OLIVEIRA MAFRA
NAYANE RODRIGUES MATOS SANTOS OI

PROJETO INTEGRADOR
ESCOPO DO PROJETO

TEMA: SISTEMAS DE CADASTRO RH

Relatório solicitado pela Generation Brasil para compor o projeto final. O relatório refere-se ao escopo do projeto integrador.

São Paulo
Dezembro de 2025

1 Título do Projeto e Modelo de Negócio escolhido

O Modelo de Negócio é um Sistema corporativo de Gestão de Recursos Humanos – **People Flow**, desenvolvido para atender organizações que necessitam automatizar processos de RH, garantindo padronização, segurança e confiabilidade das informações.

2 Descrição Geral do Projeto

Este projeto consiste no desenvolvimento do backend de um Sistema de Recursos Humanos (RH), criado para organizar, automatizar e centralizar processos essenciais da área de RH de uma empresa.

O sistema permite o cadastro e gerenciamento de unidades organizacionais, cargos e colaboradores, além de realizar o cálculo automático de salários, considerando horas trabalhadas, horas extras, bônus e descontos. Dessa forma, o sistema reduz erros manuais e aumenta a confiabilidade das informações.

A aplicação foi desenvolvida utilizando o framework NestJS e segue a arquitetura MVC, que organiza o código em camadas bem definidas:

- Model: representa os dados do sistema (entidades e tabelas do banco)
- Controller: recebe as requisições e define os endpoints
- Service: concentra as regras de negócio e validações

O sistema expõe seus dados por meio de uma API RESTful, ou seja, os recursos são acessados por endpoints HTTP padronizados, utilizando métodos como GET, POST, PUT e DELETE, com respostas no formato JSON.

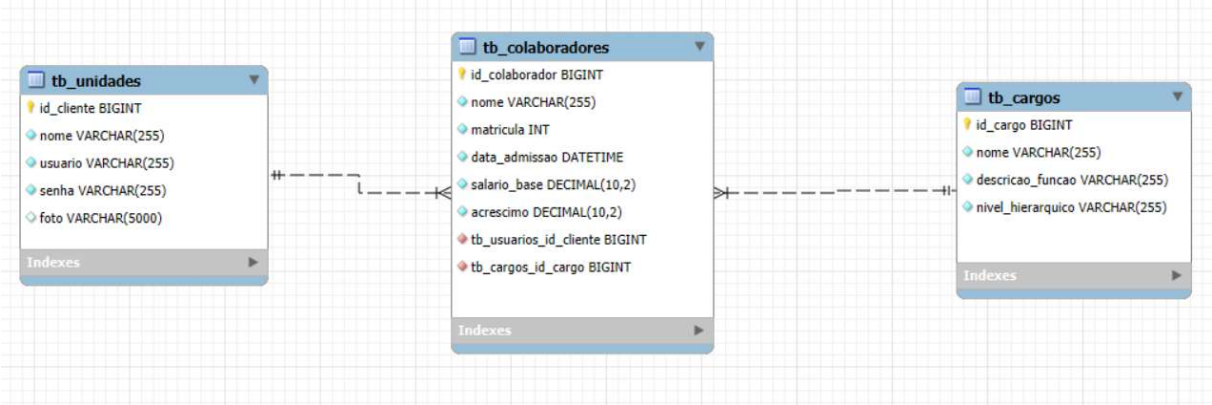
Para garantir a segurança das informações, o sistema utiliza autenticação e autorização com JWT (JSON Web Token), assegurando que apenas usuários autenticados tenham acesso às funcionalidades protegidas.

Além disso, a API conta com documentação automática por meio do Swagger, permitindo a visualização e o teste dos endpoints disponíveis, facilitando o entendimento da API por desenvolvedores, bem como a manutenção, validação e integração com outros sistemas.

Trata-se de um sistema backend organizado, seguro e escalável, desenvolvido para apoiar a gestão de pessoas de forma simples, eficiente e confiável.

O projeto contou com uma equipe de 6 pessoas, sendo um PO (Product Owner), um Scrum Master, três Desenvolvedores e um Tester.

3 Descrição da Entidade/Model foi criada e seus atributos



Unidade

Entidade responsável por representar as unidades organizacionais da empresa (filiais, departamentos ou setores administrativos).

Tabela: tb_unidades

Atributos Técnicos:

id: number (PK, auto incremental)

nome: varchar(255), obrigatório

usuario: varchar(255), obrigatório (utilizado como login/e-mail)

senha: varchar(255), obrigatório (armazenamento seguro previsto com hash)

foto: varchar(5000), opcional

Responsabilidade Técnica:

Base para autenticação de usuários

Relacionamento hierárquico com cargos

Cargo

Entidade responsável por representar cargos, funções e níveis hierárquicos dentro da organização.

Tabela: tb_cargos

Atributos Técnicos:

id_cargos: number (PK)

nome: varchar(255), obrigatório

descricao: varchar(255), obrigatório

hierarquia: varchar(255), obrigatório

Relacionamentos:

ManyToOne → Unidade

OneToMany → Colaborador

Responsabilidade Técnica:

Organização hierárquica

Associação funcional dos colaboradores

Colaborador

Entidade responsável por representar os funcionários da organização.

Tabela: tb_colaboradores

Atributos Técnicos:

id: bigint (PK)

nome: varchar, obrigatório

matricula: int, obrigatório

data_admissao: datetime, obrigatório

salario_base: decimal(10,2), obrigatório

acrescimo: decimal(10,2), opcional

Responsabilidade Técnica:

Armazenar dados contratuais

Base para cálculo de folha de pagamento

4 Funcionalidades Principais (CRUD) implementadas

Esta seção descreve de forma detalhada todas as funcionalidades CRUD (Create, Read, Update, Delete) implementadas no sistema. As funcionalidades foram implementadas em cada um dos módulos: Unidades, Cargos, Colaboradores.

Funcionalidades CRUD – Unidades

1. Listar todas as unidades – findAll()

O método findAll() é responsável por retornar todos os registros da tabela tb_unidades. Ele utiliza o método this.unidadeRepository.find() do TypeORM para realizar uma consulta completa no banco de dados, retornando uma lista com todas as unidades cadastradas.

Essa funcionalidade é utilizada principalmente para exibição geral das unidades no sistema.

2. Buscar unidade pelo ID – findById()

O método findById(id) busca uma unidade específica utilizando seu identificador único. Internamente, ele utiliza o método findOne() com a cláusula where para localizar o registro.

Caso a unidade não seja encontrada, o sistema lança uma exceção HttpException com status 404 – NOT FOUND, garantindo o tratamento adequado de erros e evitando operações inválidas.

3. Criar unidade – create()

O método create(unidade) é responsável por cadastrar uma nova unidade no sistema. Antes de salvar os dados, o método verifica se já existe uma unidade cadastrada com o mesmo usuário (e-mail), garantindo a unidade da credencial de acesso.

Se não houver duplicidade, o método utiliza this.unidadeRepository.save() para persistir os dados no banco. Caso contrário, é lançada uma exceção com status 400 – BAD REQUEST.

4. Atualizar unidade – update()

O método update(unidade) realiza a atualização dos dados de uma unidade existente. Inicialmente, o sistema valida se a unidade informada realmente existe por meio do ID.

Em seguida, verifica se o e-mail informado já está em uso por outra unidade. Caso todas as validações sejam atendidas, os dados são atualizados utilizando o método save() do repositório.

Funcionalidades CRUD – Cargos

1. Criar cargo – criar()

O método criar(cargo) realiza o cadastro de um novo cargo. Antes da persistência, o sistema valida se a unidade informada existe no banco de dados.

Essa validação é feita através do método findOne() no repositório de Unidades. Caso a unidade não seja encontrada, é lançada uma exceção 404 – NOT FOUND. Após a validação, o cargo é salvo utilizando this.cargosRepository.save().

2. Listar todos os cargos – listarTodos()

O método listarTodos() retorna todos os cargos cadastrados no sistema, incluindo seus relacionamentos com unidades e colaboradores.

Para isso, é utilizado o método find() com a propriedade relations, permitindo a carga completa dos dados associados.

3. Buscar cargo por ID – buscarPorId()

O método buscarPorId(id) localiza um cargo específico utilizando seu identificador. Caso o cargo não exista, o sistema lança uma exceção 404 – NOT FOUND, garantindo segurança na operação.

4. Atualizar cargo – alterar()

O método alterar(id, cargo) atualiza os dados de um cargo existente. Inicialmente, o sistema valida a existência do cargo e, se necessário, valida a nova unidade informada.

Após as validações, os dados são mesclados e persistidos utilizando save().

5. Deletar cargo – deletar()

O método deletar(id) remove um cargo do sistema. Antes da exclusão, o sistema verifica se o cargo existe, garantindo integridade referencial.

Funcionalidades CRUD – Colaboradores

1. Listar todos os colaboradores – findAll()

O método findAll() retorna todos os colaboradores cadastrados na tabela tb_colaboradores, ordenados alfabeticamente pelo nome. Para isso, é utilizado o método find() com critério de ordenação.

São Paulo
Dezembro de 2025

2. Buscar colaborador por ID – findById()

O método findById(id) busca um colaborador específico utilizando seu identificador único. Caso o colaborador não seja encontrado, o sistema lança uma exceção 404 – NOT FOUND, garantindo tratamento adequado do erro.

3. Buscar colaborador por nome – findByNome()

O método findByNome(nome) realiza uma busca parcial utilizando o operador LIKE, permitindo localizar colaboradores cujo nome contenha o texto informado.

Caso nenhum registro seja encontrado, o sistema retorna uma exceção informativa.

4. Criar colaborador – create()

O método create(colaborador) cadastra um novo colaborador no sistema. Antes da persistência, o sistema verifica se já existe um colaborador com o mesmo nome, evitando duplicidades.

Se a validação for atendida, o colaborador é salvo no banco utilizando save().

5. Atualizar colaborador – update()

O método update(id, colaborador) atualiza os dados de um colaborador existente. O sistema valida a existência do colaborador antes de aplicar as alterações e garante a consistência do ID informado.

6. Deletar colaborador – delete()

O método delete(id) remove um colaborador do sistema. Antes da exclusão, o sistema valida se o colaborador existe, garantindo segurança na operação.

5 Tecnologias Utilizadas (banco de dados e backend)

O backend foi desenvolvido utilizando as seguintes tecnologias e bibliotecas:

NestJS

TypeScript

TypeORM

Class-validator

JWT

Swagger (documentação de API)

Banco de Dados

Banco de dados relacional MySQL

São Paulo
Dezembro de 2025