Relatório Explicativo – Sprint 3

Projeto: Folha Fácil

Grupo: FolhaFácil Team

Introdução

O sistema Folha Fácil tem como objetivo automatizar o processo de cálculo de folha de pagamento, integrando informações de funcionários, benefícios, horas extras e deduções legais. Nesta terceira sprint, o foco foi o uso de conceitos avançados de Java, com ênfase em Polimorfismo Paramétrico (Generics), Streams, persistência de dados e implementação de eventos.

Além das implementações técnicas, também foram realizados testes unitários e a preparação da arquitetura para integração com o frontend Angular.

Objetivos da Sprint

- Implementar polimorfismo paramétrico (Generics) em partes do sistema para reduzir duplicação de código.
- Utilizar coleções Java (List, Set, Map) na manipulação de dados.
- Empregar Streams para filtragem, transformação e agregação de dados.
- Garantir persistência de dados via banco de dados relacional.
- Implementar eventos para ações automatizadas, como logs e notificações.
- Preparar o backend para futura integração com o frontend Angular.
- Criar testes unitários para as novas funcionalidades.

Tecnologias Utilizadas

Categoria	Tecnologia	Finalidade
Linguagem	**Java 17**	Desenvolvimento backend
Framework	**Spring Boot 3**	Injeção de dependência, persistência e APIs
Banco de dados	**PostgreSQL / MySQL**	Armazenamento relacional dos dados
ORM	**JPA / Hibernate**	Mapeamento objeto-relacional
Frontend	**Angular 17**	Interface web (em desenvolvimento)

Containerização	**Docker**	Padronização do ambiente e execução dos serviços
Segurança	**Keycloak**	Autenticação e autorização via JWT
Testes	**JUnit 5 e Mockito**	Testes unitários e de integração
Gerenciamento	**Maven**	Controle de dependências e build

Arquitetura da Aplicação

A aplicação segue uma arquitetura em camadas, promovendo baixo acoplamento e alta coesão:

- **Controller:** recebe requisições HTTP e retorna respostas REST.
- **Service:** contém as regras de negócio e lógica de aplicação.
- **Repository:** realiza a comunicação com o banco de dados.
- **Entity e DTO:** representam os modelos de domínio e objetos de transferência de dados.

Implementação de Polimorfismo Paramétrico (Generics)

Foi criado um repositório genérico (RepositorioGenerico<T>) e um serviço genérico (ServiceGenerico<T, ID>) que centralizam operações comuns de CRUD. Classes como FuncionarioServiceImpl e FolhaPagamentoServiceImpl agora herdam desse serviço genérico, reduzindo duplicação e padronizando o CRUD.

Uso de Coleções e Streams

O sistema faz uso extensivo de coleções Java, como List e Set, especialmente na manipulação de listas de benefícios, logs e funcionários. Também foram implementados Streams para processar dados de forma funcional e eficiente.

Persistência de Dados

A aplicação utiliza Spring Data JPA para realizar a persistência de dados em um banco relacional PostgreSQL, executado dentro de um container Docker.
As entidades como Funcionário, Beneficio, FolhaPagamento e HoraExtra são mapeadas por meio de anotações JPA e gerenciadas automaticamente pelo Hibernate.

O ambiente é orquestrado via Docker Compose, garantindo que todos os serviços necessários (banco de dados, backend e autenticação Keycloak) sejam executados de forma padronizada e reprodutível em qualquer máquina.

Implementação de Eventos

Foram criados **eventos automáticos** para ações-chave do sistema:

- **Criação de funcionário:** gera automaticamente um log no LogFuncionarioServiceImpl.
- **Geração de folha de pagamento:** dispara log em LogFolhaPagamentoServiceImpl.
- **Alterações de status:** (habilitar/desabilitar funcionário) registram eventos de auditoria.

Esses eventos seguem o padrão Observer/Listener, garantindo rastreabilidade das ações.

Testes Unitários

Foram desenvolvidos testes unitários e de integração utilizando JUnit 5 e Mockito. Os testes garantem o correto funcionamento dos métodos novos que foram gerados.

Dockerização e Execução

O projeto possui um arquivo docker-compose.yml com os serviços necessários:

- **Backend (Spring Boot)**
- **Banco de dados (PostgreSQL)**
- **Keycloak (Autenticação)**
- **Frontend (Angular)**

Para rodar o sistema:

1. Execute o comando: docker compose up --build
2. Após a inicialização, rode a aplicação principal (FolhaFacilApplication.java).
3. Acesse:
   - Backend: http://localhost:8080

Conclusão

A Sprint 3 consolidou a base técnica do sistema FolhaFácil, implementando o uso de Generics, Streams e eventos, além de garantir a persistência dos dado. Com isso, o sistema está preparado para a próxima etapa, que incluirá novas funcionalidades e integração completa com o módulo web.