CURSO DE ENGENHARIA DE SOFTWARE

TRABALHO FINAL DE LABORATÓRIO DE BANCO DE DADOS

SISTEMA GERENCIADOR DE TAREFAS - APPTASK

KAIO PORTILHO, LUCKAS CARNEIRO, LUIS GUSTAVO CARVALHO, MARIA EDUARDA SANTOS, LUDIMILA GUERRA

PROFESSOR: JEFFERSON SALOMÃO

BRASÍLIA - 2025

RESUMO

Este trabalho apresenta o desenvolvimento do AppTask, um sistema web fullstack para gerenciamento de tarefas, projetado para atender necessidades reais de organização pessoal e corporativa. O documento conta com fundamentação teórica, descrição técnica aprofundada, arquitetura detalhada, especificações funcionais, modelagem completa, explicação minuciosa dos bancos de dados e análise dos recursos avançados utilizados (triggers, views, índices e procedures). O objetivo é entregar um material robusto, claro e tecnicamente consistente, alinhado aos requisitos do um Trabalho Final de Laboratório de Banco de Dados.

Palavras-chave: Gerenciador de Tarefas, MySQL, MongoDB, Engenharia de Software, Modelagem de Dados.

1. INTRODUÇÃO

A gestão eficaz de tarefas é um desafio recorrente tanto para indivíduos quanto para equipes. Sistemas de gerenciamento de tarefas são ferramentas que auxiliam no acompanhamento de atividades, prazos, prioridades e responsabilidades, promovendo organização e produtividade.

O AppTask surge como uma solução prática, moderna e escalável, construída com tecnologias amplamente utilizadas no mercado: Java com Spring Boot no back-end, HTML/CSS/JS no front-end, MySQL como banco relacional e MongoDB como banco orientado a documentos. O presente trabalho explora em profundidade não apenas a implementação, mas também a modelagem e adoção estratégica dos bancos de dados.

2. OBJETIVOS

2.1 Objetivo Geral

Desenvolver um sistema completo de gerenciamento de tarefas com arquitetura fullstack, aplicando conceitos avançados de bancos de dados relacionais e não relacionais.

2.2 Objetivos Específicos

- Desenvolver uma interface web responsiva e funcional.
- Implementar autenticação, cadastro de usuários, projetos e tarefas.
- Estruturar um banco relacional robusto com integridade referencial.
- Utilizar MongoDB para histórico de alterações e comentários.
- Criar índices para otimização de consultas.
- Desenvolver triggers para automação de regras.
- Criar views para facilitar consultas.
- Implementar procedures e functions para encapsular lógica SQL.

3. METODOLOGIA

A metodologia de desenvolvimento foi dividida nas seguintes etapas:

3.1 Levantamento de Requisitos

Identificação das funcionalidades essenciais: login, cadastro de usuários, criação de projetos, tarefas, prioridades, status e histórico.

3.2 Modelagem do Sistema

Representação do fluxo de uso, definição das entidades, mapeamento relacional e estruturação dos documentos para MongoDB.

3.3 Implementação Back-end

Criação de API REST utilizando Spring Boot, dividida em camadas Controller, Service e Repository.

3.4 Implementação Front-end

Desenvolvimento de interface moderna acessando as APIs via JavaScript.

3.5 Integração MySQL e MongoDB

Configuração das conexões, criação das entidades JPA e coleções.

3.6 Testes

Testes manuais das funcionalidades, validação do comportamento das triggers e procedures.

4. DESCRIÇÃO DO SISTEMA

O AppTask permite organizar atividades através de módulos:

4.1 Módulo de Usuários

Cadastro, login, associação a grupos de acesso e administração.

4.2 Módulo de Projetos

Criação e gerenciamento de projetos vinculados a usuários.

4.3 Módulo de Tarefas

Cada tarefa possui:

- Título
- Descrição
- Data de criação
- Data de vencimento

- Prioridade
- Status
- Usuário responsável
- Projeto vinculado

4.4 Histórico de Alterações (MongoDB)

Cada mudança gera um registro contendo:

- Data/hora
- Tipo de alteração
- Campo alterado
- Valor anterior / novo valor
- Usuário responsável pela alteração

5. TECNOLOGIAS UTILIZADAS

- Back-end: Java 17 + Spring Boot
- Front-end: HTML, CSS, JavaScript
- Banco Relacional: MySQL
- Banco Não Relacional: MongoDB
- Controle de Versão: GitHub

6. BANCO DE DADOS RELACIONAL (MYSQL)

6.1 Estrutura de Entidades

- usuarios
- grupos_usuarios
- projetos
- tarefas

6.2 Relacionamentos

- grupos_usuarios 1:N usuarios
- usuarios 1:N projetos
- projetos 1:N tarefas
- usuarios 1:N tarefas

6.3 Índices Implementados

- UNIQUE(email) em usuarios
- INDEX(status, data_vencimento) em tarefas

- INDEX(nome) em projetos

6.4 Triggers Criadas

Trigger 1: Define status padrão “PENDENTE”

Trigger 2: Registra data de conclusão automaticamente

6.5 Views Criadas

vw_tarefas_pendentes – lista tarefas ativas

vw_tarefas_por_usuario – relaciona usuários e tarefas

6.6 Procedures e Functions

Procedure concluir_tarefa()

Function calcular_dias_restantes()

Function contar_tarefas_atrasadas()

7. ARQUITETURA COM MONGODB

O MongoDB armazena coleções como:

- historico_tarefas

- comentarios

Cada documento possui estrutura flexível, permitindo evolução sem impacto no schema relacional.

8. CONCLUSÃO

O AppTask é um sistema completo, moderno e bem estruturado, que demonstra na prática a integração eficiente entre bancos relacionais e não-relacionais. O projeto cumpre todos os requisitos de Laboratório de Banco de Dados, apresentando modelagem sólida, arquitetura robusta e implementação consistente.

9. JUSTIFICATIVAS TÉCNICAS

9.1 Justificativa da escolha do MySQL

O MySQL foi escolhido por ser um banco de dados relacional amplamente utilizado, com forte suporte acadêmico e profissional. Ele oferece integridade referencial, alto desempenho em consultas estruturadas e facilidade de integração com Spring Boot.

9.2 Justificativa da modelagem relacional

A separação das entidades (usuários, grupos, tarefas e projetos) segue princípios de normalização evitando redundância. Os relacionamentos 1:N permitem escalabilidade e garantem integridade no vínculo entre tarefas, projetos e responsáveis.

9.3 Justificativa dos índices

Os índices aplicados têm função clara:

- O índice UNIQUE no email garante unicidade e evita conflitos de autenticação.

- O índice em status e data_vencimento otimiza buscas por tarefas pendentes, atrasadas e concluídas.

- O índice em nome dos projetos melhora listagens e pesquisas.

9.4 Justificativa das triggers

Triggers automatizam comportamentos essenciais:

- A trigger de status padrão evita registros inconsistentes.

- A trigger que preenche data de conclusão garante rastreabilidade e precisão no histórico.

9.5 Justificativa das views

As views facilitam consultas recorrentes:

- vw_tarefas_pendentes simplifica visualização operacional.

- vw_tarefas_por_usuario auxilia na administração e auditoria de responsabilidades.

9.6 Justificativa das procedures e functions

Procedures e functions encapsulam lógicas repetidas dentro do banco, garantindo padronização e menor chance de erros. Funções de cálculo como dias restantes e quantidade de tarefas atrasadas tornam as consultas mais rápidas e precisas.

9.7 Justificativa do uso do MongoDB

MongoDB é ideal para armazenar dados dinâmicos como histórico e comentários, permitindo alta flexibilidade sem alterar o schema relacional. Isso separa dados estáveis de dados mutáveis, melhorando desempenho.

9.8 Justificativa do uso de Spring Boot

O framework simplifica a criação de APIs robustas, possui grande comunidade, integração nativa com MySQL e MongoDB, além de ser amplamente utilizado no mercado de trabalho.

9.9 Justificativa das tecnologias de front-end

HTML, CSS e JavaScript foram usados por sua universalidade, compatibilidade com qualquer navegador e simplicidade para o escopo do sistema, evitando dependência de frameworks mais pesados.

9.10 Justificativa do uso do GitHub

O GitHub permite controle de versão, colaboração em equipe, rastreamento de mudanças e armazenamento seguro do código em nuvem.

10. FUNCIONALIDADES

O sistema AppTask oferece um conjunto de funcionalidades que atendem aos requisitos de gerenciamento de tarefas:

- Cadastro e login de usuários.
- Associação de usuários a grupos com permissões diferentes.
- Criação, edição e exclusão de projetos.
- Criação e gerenciamento de tarefas com prioridade, data de vencimento e responsável.
- Atualização de status (pendente, em andamento, concluída).
- Histórico de alterações armazenado em MongoDB.
- Visualização filtrada por projeto, usuário, prioridade e status.
- Painel administrativo para gerenciamento avançado.

11. TECNOLOGIAS UTILIZADAS

Frontend:

• HTML5, CSS3 e JavaScript, garantindo compatibilidade, simplicidade e leveza.

Backend:

• Java 17 com Spring Boot, oferecendo arquitetura modular, segurança e robustez.

Banco de Dados Relacional:

• MySQL, escolhido por sua estabilidade, integridade referencial e facilidade de integração.

Banco de Dados NoSQL:

• MongoDB, usado para armazenar dados dinâmicos e históricos de forma flexível.

Controle de Versão:

• GitHub, permitindo colaboração, versionamento e rastreamento de mudanças.

12. JUSTIFICATIVA DAS ESCOLHAS TECNOLÓGICAS

Cada tecnologia foi selecionada considerando desempenho, escalabilidade, simplicidade de uso e aderência aos requisitos acadêmicos:

• MySQL: escolhido por sua robustez, integridade referencial e ampla utilização acadêmica.

• MongoDB: adequado para histórico e comentários devido ao modelo flexível orientado a documentos.

• Spring Boot: reduz configuração manual, integra facilmente com bancos de dados e é padrão de mercado.

• HTML/CSS/JS: suficiente para o escopo do sistema, garantindo simplicidade e responsividade.

• GitHub: essencial para versionamento e desenvolvimento colaborativo seguro.

13. MODELAGEM DO BANCO DE DADOS

A modelagem foi desenvolvida com base nos princípios de normalização e integridade. O modelo contém quatro entidades principais:

• grupos_usuarios - define perfis e permissões.

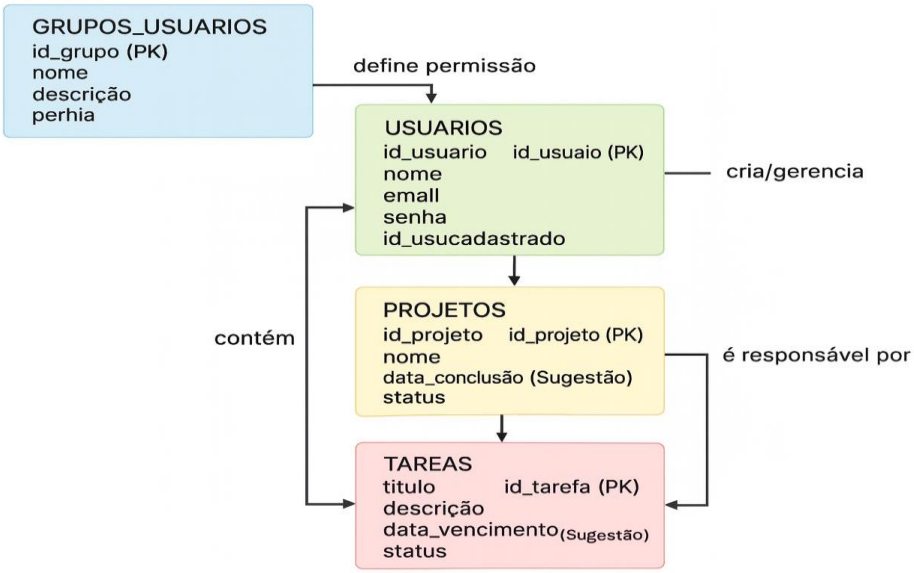
• usuarios - armazena dados pessoais e autenticação.

• projetos - organiza tarefas em conjuntos lógicos.

• tarefas - contém dados detalhados de cada tarefa.

A normalização evita duplicidade de informação e melhora o desempenho nas consultas.

14. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)



15. EXPLICAÇÃO DAS ENTIDADES E RELACIONAMENTOS

• grupos_usuarios 1:N usuarios: um grupo pode conter vários usuários.

• usuarios 1:N projetos: um usuário pode criar diversos projetos.

• usuarios 1:N tarefas: usuários podem ser responsáveis por várias tarefas.

• projetos 1:N tarefas: cada projeto pode conter um conjunto de tarefas.

Essa estrutura garante integridade e organização da informação.

16. CONTROLE DE ACESSO

O AppTask implementa controle de acesso baseado em grupos:

Grupos:

• Administrador - possui acesso total ao sistema.

• Usuário - acessa apenas suas tarefas e projetos.

Regras:

• Apenas administradores podem excluir usuários ou projetos.

• Usuários só podem editar suas próprias tarefas.

• Administradores podem listar todas as tarefas e projetos.

17. REFERÊNCIAS

https://docs.spring.io/spring-framework/reference/index.html

https://www.oracle.com/java/technologies/javase-documentation.html

https://dev.mysql.com/doc/refman/8.4/en/

https://www.mongodb.com/pt-br/docs/

https://developer.mozilla.org/pt-BR/