Serviço Público Federal
Ministério da Educação
Fundação Universidade Federal de Mato Grosso do Sul

Sistema bibliotecário

Curso: Engenharia de Software
Disciplina: Laboratório de Banco de Dados
Professora: Vanessa Borges
Participantes:

- Pedro Nicoletti Sotoma RGA: 202219060729
- Roberto Fernandes Fortes Neto RGA: 202219060605
- Daniel Jordão Schutz RGA: 202219060060

Data de Entrega: 09/11/2023

Tabela de índices

1. Especificação do problema (pág.3)
2. Esquema relacional (pág. 4)
3. Triggers (pág. 5)
4. Tecnologias utilizadas (págs.6 e 7)
5. Tutorial de instalação (pág.8)

Problema

Nosso projeto tem como objetivo permitir o gerenciamento de uma aplicação web bibliotecária, onde por meio de interações rápidas e simples o usuário pode gerenciar os principais elementos necessários para o seu negócio. Optamos por utilizar esse tema pela ideia de fácil abstração de modelagem, assim como bom entendimento para o usuário.

Optamos por utilizar uma arquitetura baseada em serviços. Para melhor entendimento do problema, desenvolvemos um modelo entidade relacionamento a fim de abstrair todas as entidades necessárias para o banco de dados e seus relacionamentos.

Funcionalidades do sistema:
Gestão de Leitores:

- Registre os leitores, que são os clientes da biblioteca, permitindo sua manutenção e o controle das operações relacionadas a eles.

Gestão de Livros:

- Registre os livros disponíveis na biblioteca, incluindo informações relevantes como título, autor e disponibilidade.

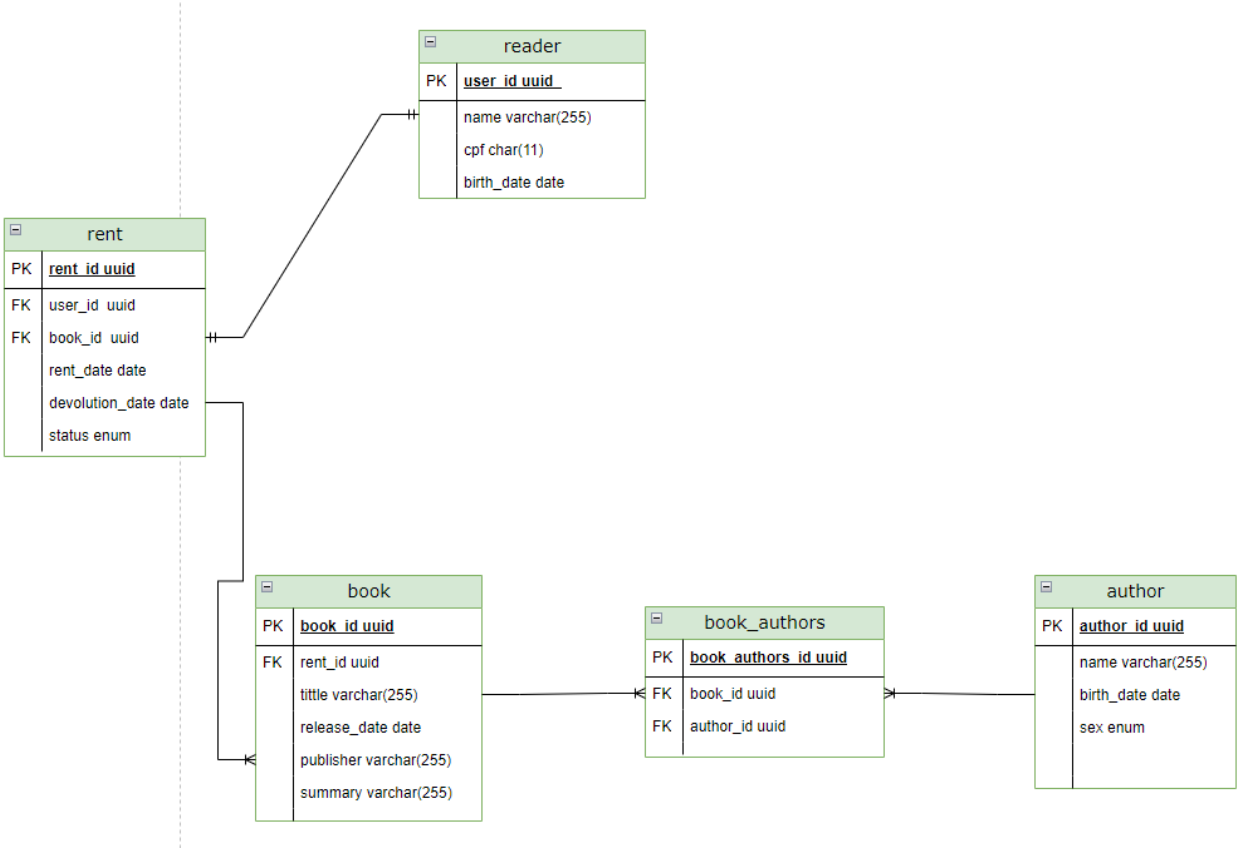
Gestão de Autores:

- Registre os autores dos livros no sistema, permitindo a associação de autores aos livros disponíveis na biblioteca.

Gestão de Aluguel:

- Registre os aluguéis de livros, o que é essencial para o gerenciamento das operações comerciais da biblioteca. Isso inclui informações sobre os livros alugados, os leitores envolvidos e as datas de aluguel.

Modelo Relacional



Desenvolvemos o modelo relacional através de um diagrama entidade-relacionamento (DER) com o objetivo de fornecer uma representação prática e de fácil compreensão para os usuários. É possível constatar que todos os relacionamentos requeridos foram devidamente implementados, incluindo as associações 1:n (entre as entidades "rent" e "book"), 1:1 (entre "rent" e "reader"), e n:n (entre "book" e "author").

Triggers

Em nosso sistema, foram introduzidos triggers para monitorar e registrar operações em quatro entidades-chave: author, reader, book e rent, como também em uma entidade(bookAuthor) originada da tabela de junção do relacionamento M:N entre livro e autor. Esses triggers podem ser localizados no diretório de código em src/main/resources/db/migration. Porém, também nos utilizamos das notations do ORM para estabelecer funções semelhantes aos triggers na aplicação. Logo, desenvolvemos duas camadas de “triggers”, uma mediada pela aplicação e outra diretamente na database.

A implementação dos triggers tem como objetivo proporcionar um controle detalhado sobre o histórico de alterações das entidades armazenadas no banco de dados. Para alcançar essa funcionalidade, utilizamos funções que são acionadas automaticamente sempre que operações de criação, atualização ou exclusão são executadas. Essas funções rastreiam as datas de criação, exclusão e a última modificação de cada entidade. Dessa forma, mantemos um registro completo e preciso de todas as atividades relevantes no banco de dados.

Em resumo, os triggers implementados são mecanismos poderosos para automatizar a manutenção de informações de data e hora relacionadas aos registros na tabela reader. Eles ajudam a garantir a integridade e a rastreabilidade dos dados, simplificando as operações de criação, atualização e exclusão.

Tecnologias

Spring Boot:
O framework Spring Boot foi escolhido como a base da aplicação, proporcionando uma estrutura sólida e altamente configurável para o desenvolvimento Java. Ele simplifica o processo de configuração e oferece suporte a muitos recursos essenciais para o desenvolvimento de aplicativos web.

JPA (Java Persistence API) com QueryDSL:
O JPA é a camada de persistência que permite a interação eficiente com o banco de dados. A integração com QueryDSL permite a criação de consultas complexas de forma fácil e legível.

Flyway:
O Flyway é uma ferramenta essencial para o controle de versionamento do banco de dados. Ele permite a migração contínua e controlada das estruturas do banco de dados à medida que a aplicação evolui.

Spring Web:
O Spring Web, com o uso do Spring MVC, é responsável por criar a lógica de negócios e controlar as requisições HTTP, permitindo a construção de um aplicativo web robusto.

Lombok:
A biblioteca Lombok é usada para reduzir o código boilerplate, tornando o código mais limpo e fácil de manter. Ela oferece anotações para gerar automaticamente getters, setters e outros métodos comuns.

Swagger:
A ferramenta Swagger é usada para documentar as APIs da aplicação. A documentação é acessível através de uma interface web amigável, facilitando a compreensão e a utilização das APIs.

Banco de Dados PostgreSQL:
O PostgreSQL foi escolhido como o sistema de gerenciamento de banco de dados devido à sua ampla adoção na indústria de TI, à abundância de recursos educacionais disponíveis e ao conhecimento prévio da equipe, que já possui experiência com este sistema. Foi utilizada a versão 7.4 do PgAdmin 4.

Docker:
O uso do Docker simplifica o empacotamento da aplicação e suas dependências em um container, garantindo a portabilidade entre diferentes ambientes. O Docker Compose é utilizado para a rápida execução da aplicação, com a definição de serviços e configurações no arquivo docker-compose.yml.

Frontend com React.js:
Para a interface do usuário, foi escolhido o React.js, uma biblioteca JavaScript amplamente adotada para o desenvolvimento de interfaces web interativas e responsivas.

Instalação do sistema

Para instalação do sistema é necessário seguir os seguintes passos:

- Pelo terminal, abrir o diretório do projeto e entrar na pasta backend
- Abrir o docker desktop e subir um container utilizando o comando docker-compose up no terminal
- Entrar no PgAdmin 4 e se conectar ao servidor utilizando como ip: localhost, usuário: dev_user e senha: dev_password
- Num segundo terminal, executar o comando mvn install para instalar as dependências do backend
- Executar o backend nessse terminal utilizando o comando: SPRING_PROFILES_ACTIVE=dev ./mvnw spring-boot:run
- Usando um terceiro terminal, abrir a pasta do frontend e instalar as dependências do React js utilizando o comando npm install
- Rodar a aplicação utilizando o comando npm start

Obs: para acessar a documentação gerada pelo swagger, o usuário pode, ao rodar a aplicação backend na porta 3001, entrar no link: http://localhost:3001/swagger-ui/index.html#/