



Aluno(a): Melissa Francielle dos Santos – 3º ano

Professor(a): André Luis Andrade Menolli – Disciplina: Engenharia de Software II

### PROJETO FINAL – DESCRIÇÃO DO TRABALHO

- Este trabalho não foi realizado em dupla.
  
- a. Arquitetura Proposta para o projeto, que deve conter:

A arquitetura proposta para este projeto foi a Arquitetura em Camadas.

O padrão arquitetural mais comum desse tipo de arquitetura é o MVC (*Model-View-Controller*) que foi o escolhido para esse projeto e principal do sistema desenvolvido.

- A visão conceitual da arquitetura  
Resposta:  
A Arquitetura em camadas, possui a ideia de dividir o sistema em módulos distintos e garantir a separação das responsabilidades, facilitando manutenção, escalabilidade e testes. Sendo essa separação da seguinte forma:
  - Camada de UI: que será a interface de interação com o usuário.
  - Camada de lógica de negócios: que irá conter as regras de negócios e gerencia os processos centrais.
  - Camada de persistência: manipulação dos dados no banco de dados.

Essas camadas são utilizadas de maneira hierárquica, interagindo cada camada de uma forma direta com sua camada adjacente.

Visão do padrão arquitetural utilizado:

O padrão MVC busca separar as responsabilidades dentro do sistema entre Modelo, Visão e Controlador. Tendo também uma variação de projeto o MVP que tendo uma visualização do projeto podemos dizer que há uma certa influência do MVP (*Model-View-Presenter*).

- Modelo (MODEL): responsável por conter a lógica do negócio.
  
- Visão (VIEW): sendo responsável pela questão gráfica, interface gráfica, do usuário. Irá apresentar as informações ao usuário.

- Controlador (CONTROLLER): seria o intermediário do código, estaria presente entre o view e o model para poder receber os comandos do view e processar utilizando o model.

Esse padrão é muito utilizado e facilita nas questões de organização do código e ajuda na manutenibilidade futura do código.

- A descrição dos elementos da arquitetura e as dependências

Resposta:

Camada de apresentação: exibe as informações a partir de uma interface gráfica utilizando uma forma de visualização para a interação com o usuário.

Capturando essas interações e enviando para uma camada de lógica de negócio. Dependendo da camada de lógica para prosseguir seu funcionamento corretamente.

Camada de lógica de negócios: irá implementar as lógicas de negócios e logica central do sistema atuando como intermediaria entre a apresentação e a persistência.

Dependendo apenas da camada de persistência para acessar a manipulação dos dados

Camada de persistência: essa camada é responsável pelo fornecimento de operações do CRUD e encapsuladas em repositórios, ela gerencia a interação com o banco de dados e outros recursos de armazenamentos.

Não é dependente de outras camadas como as camadas anteriores.

### **Descrição dos elementos voltados ao padrão:**

- a. Modelo – Gerencia o acesso e a manipulação dos dados, representando a lógica do negócio sendo responsável por atualizar as informações que serão passadas para o Controller e assim para o View. Suas responsabilidades são voltadas a operações do CRUD, manter o estado da aplicação.

**Dependências:** Não necessariamente depende do Controller ou do View, porém irá se comunicar e pode se comunicar com o Controller para receber as informações de entrada com base no que for manipulado pelo usuário.

- b. Visão – Irá realizar a apresentação dos dados para o usuário. Exibindo a interface gráfica visualizando o que o usuário vê e interage, apenas lidando com a interface. Exibe os dados do model que são passados pelo Controller e apresenta a UI, recebendo as interações com o usuário.

**Dependência:** Dependente do model apenas para a exibição dos dados e para poder atualizar a interface sempre que o estado mudar.

- c. Controlador – atuando como intermedia de VIEW e do MODEL, irá processar as entradas dos usuários e converter em ações dadas pelo model, para atualização da interface. Processando as entradas esse componente é responsável pela interpretação desses dados recebidos da view e do model.

**Dependências:** O controlador é dependente da view e do model, por sua vez a view solicita o controlador para atualização dos dados com base nas ações do usuário.

Dependências em geral:

- O view envia a interações para o Controller.
- O Controller manipula os eventos e solicita o Model os dados e informações.
- Model atualiza os dados e notifica a view.

O padrão contém uma separação das responsabilidades, isso sendo uma de suas principais vantagens e além disso permite que a interface do usuário seja alterada sem afetar a lógica dos negócios e vice-versa.

Dependências gerais do projeto

- JAVAFX – utilizando o JavaFX para construção da interface gráfica.
- JDBC – usando o JDBC para postgresSQL para comunicação com o banco de dados e armazenamento.
- JUnit – dependência para testes unitários.

Além do uso do Maven para gerenciar essas dependências e ciclo de vida do projeto. Possui um pom.xml definindo bibliotecas e os plugins que são necessários para o projeto.

- Descrever qual ou quais padrões arquiteturais foram base para escolha da arquitetura e justificar porque o escolheram.

Resposta:

A arquitetura utilizada teve como escolha principal para seu uso, pois é de fácil separação de responsabilidade, além de se tornar mais fácil a manutenção futura do código e modificação das partes do sistema sem que haja problemas em outras partes do software. Permite também a ideia de testes unitário.

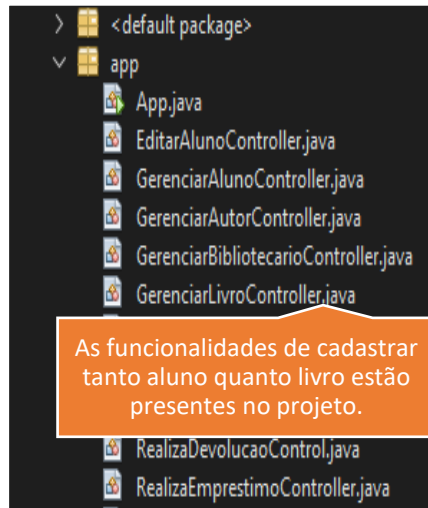
### **O padrão arquitetural utilizando o MVC**

Foi adotado para poder organizar as camadas do software, separando view, Controller e model, permitindo as alterações na interface e regras de forma independentemente. Otimizando a interação entre as camadas e intermediário claro da lógica com a apresentação do projeto.

A combinação tanto dessa forma de arquitetura quanto do padrão usado foi para a necessidade de um sistema com suas responsabilidades e organização, tornando escalável e de fácil visualização e manutenção. Focando principalmente nas questões voltadas as separações de responsabilidades que torna mais fácil essa visão de métodos e questões que estão voltadas a determinadas partes do projeto e quais não são.

- b. Adicionar funcionalidades de cadastrar livros e alunos (inclusive as interfaces gráficas)

Resposta:



A interface gráfica será apresentada no vídeo para melhor visualização do funcionamento do código e da interface gráfica.

- c. Finalizar a implementação do Caso de Uso Emprestar Livro, e essa deve contemplar:

Resposta:

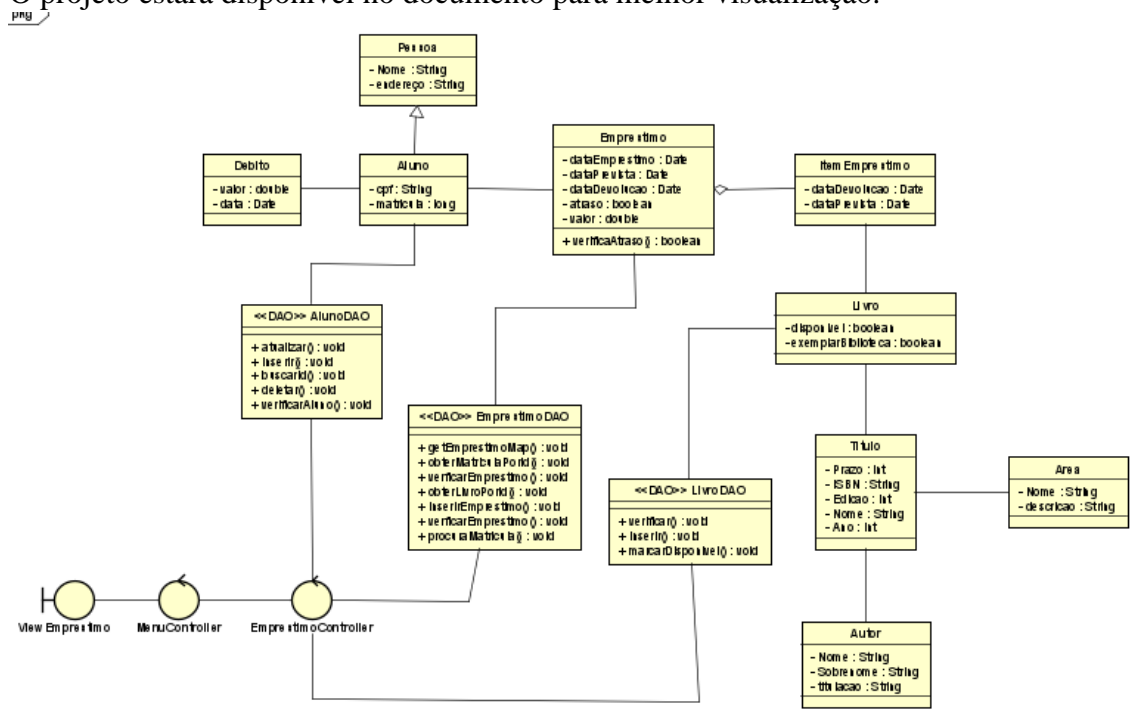
Nome caso de uso: Emprestar Livro	
Ator: bibliotecário	
<b>Fluxo principal</b> <ol style="list-style-type: none"> <li>1. O aluno apresenta os livros ao funcionário e sua identificação</li> <li>2. O funcionário insere a identificação e os livros no sistema</li> <li>3. O sistema verifica se o Aluno está cadastrado.</li> <li>4. O sistema verifica se o aluno possui pendencias.</li> <li>5. O sistema cria um empréstimo</li> <li>6. Para cada livro</li> </ol>	<b>Fluxo Alternativo</b> <ol style="list-style-type: none"> <li>3.a. Aluno não cadastro <ol style="list-style-type: none"> <li>3.a.1. O sistema informa que o aluno não está cadastrado.</li> <li>3.b.1. O sistema finaliza o caso de uso.</li> </ol> </li> <li>4.a Aluno possui débitos <ol style="list-style-type: none"> <li>4.a.1 O sistema informa que o aluno está em débito</li> <li>4.a.2 O sistema finaliza o caso de uso</li> </ol> </li> </ol>

<p>6.1.O sistema verifica se o livro pode ser emprestado.</p> <p>6.1.1. Se disponível prosseguir</p> <p>6.1.2. Se não disponível, seguir o fluxo 6.1.a ou b.</p> <p>6.2.O sistema cria um item de empréstimo.</p> <p>6.3.O sistema associa o livro a item.</p> <p>7. O sistema calcula a data de devolução</p> <p>8. O sistema grava os dados do empréstimo</p> <p>9. O sistema imprime os dados de empréstimo</p>	<p>6.1.a Livro reservado</p> <p>6.a.1 O sistema informa que o livro está reservado e não pode se emprestado</p> <p>6.a.2 O sistema informa a data de devolução do livro</p> <p>6.1.a.3 Retorna o passo 6</p> <p>6.1.b. Livro de não pode ser emprestado</p> <p>6.1.b.1 o sistema informa que o livro é exemplar que não pode ser emprestado</p> <p>6.1.b.3 retorna ao passo 6.</p>
--	--

- Atualização do diagrama de classe, inserido a camada de persistência aplicando o padrão DAO

Resposta:

O projeto estará disponível no documento para melhor visualização.



- Finalização do Diagrama de Sequência apresentado na Figura 17 (apresentado no material da aula), para que tenha todas as funcionalidades para implementar o Caso de Uso.

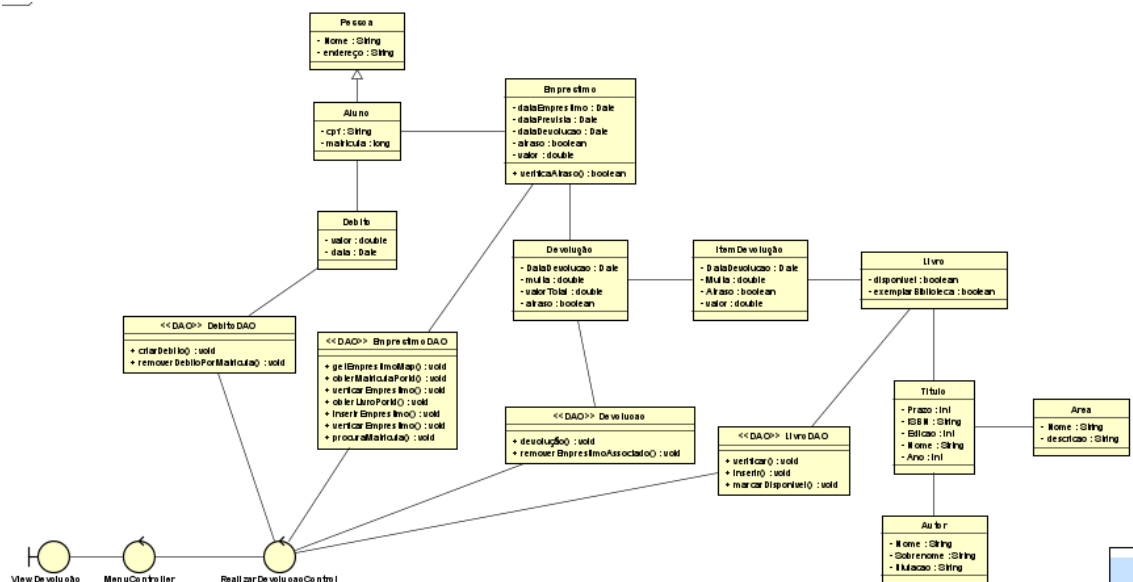


<p>2. O sistema valida a existência do membro do livro no cadastro.</p> <p>3. o sistema verifica se o livro está sendo desenvolvido dentro do prazo:</p> <p>3.1 Se está dentro do prazo, executar o passo 4.</p> <p>3.1 Se há atraso, o sistema calcula a multa correspondente e registra a transação.</p> <p>4. A devolução é realizada.</p> <p>5. O sistema reduz o número de livros em pose do aluno no cadastro</p>	<p>2.a. O código do livro é inválido.</p> <p>2.a.1. Uma mensagem de erro é exibida.</p> <p>2.a.2. O processo é cancelado.</p> <p>2.a.3. retorna ao passo 1.</p> <p>2.b. Livro não foi emprestado ao membro informado.</p> <p>2.b.1. exibe uma mensagem de erro informando o ocorrido.</p> <p>2.b.2. retorna ao passo 1.</p> <p>4.a. Falha na atualização do sistema</p> <p>4.a.1 O sistema exibe uma mensagem do problema.</p> <p>4.a.2. tenta novamente o passo 4.</p>
---	---

- Atualização do diagrama de classe, inserido a camada de persistência aplicando o padrão DAO.

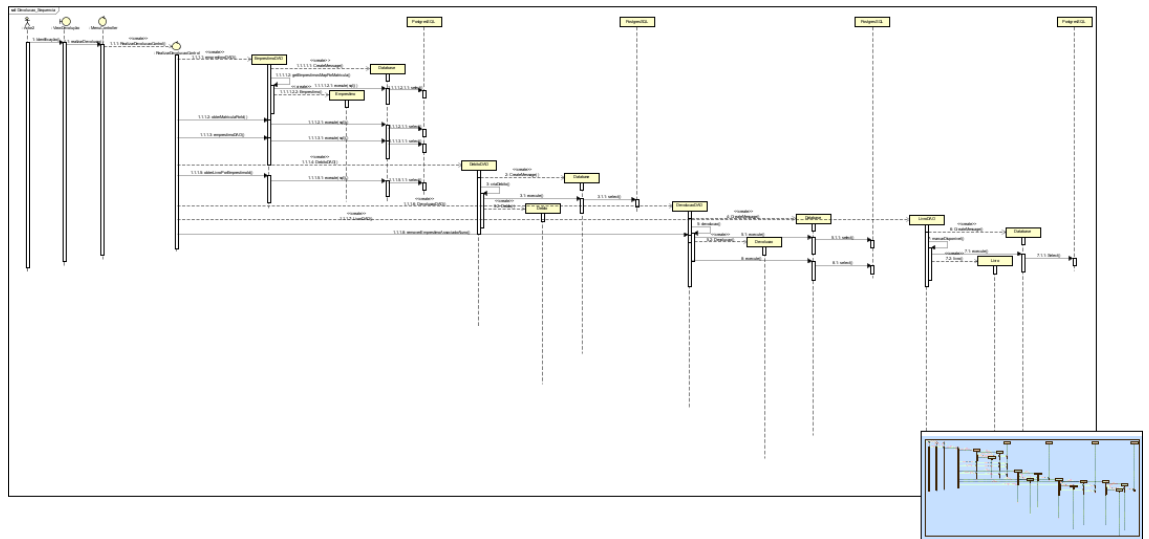
Resposta:

O projeto realizado no Astah estará disponível no documento para melhor visualização.



- Fazer o diagrama de sequência da funcionalidade

Resposta:



- Implementar o caso de uso de acordo com as regras definidas no caso de uso, e com os novos diagramas de o diagrama de classes e sequência (para isso crie um a base de dados em um SGBD da sua escolha para ser o método de armazenamento persistente).

Resposta:

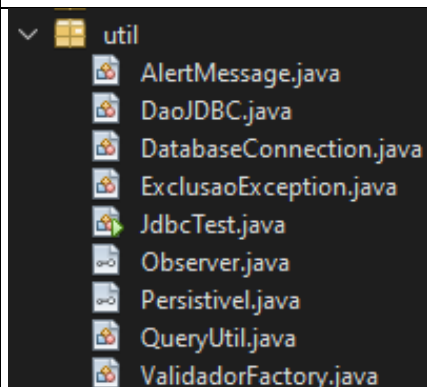
A base de dados usada para a implementação do projeto foi o PostgreSQL com tecnologia JDBC e o DAO para a comunicação com o banco de dados a partir dos comandos.

- e. O sistema deve contemplar ao menos três padrões de projeto, em qualquer camada.

Resposta:

Os padrões utilizados no projeto:

ValidadorFactory e JDBCTest – Factory Method

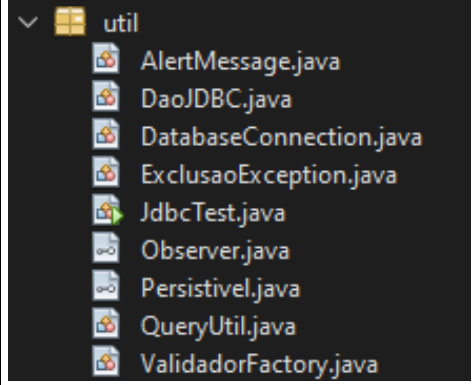


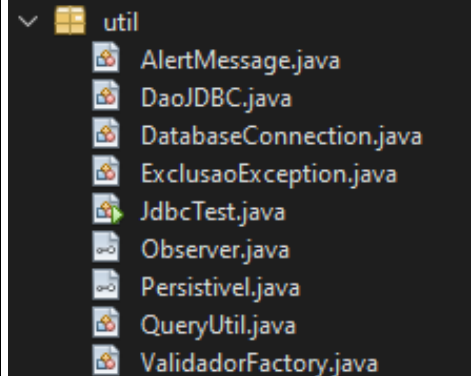
A classe ValidadorFactory segue o padrão Factory Method, centralizando a lógica de criação dos objetos do tipo. Criando os objetos sem que a lógica seja exposta diretamente no código.

O JDBCTest usa indiretamente o padrão Factory Method, tendo a criação de uma conexão com o banco de dados que pode ser vista coo uma simplificação do padrão.

DatabaseConnection e AlertMessage – Singleton



	<p>A classe DatabaseConnection carrega o padrão de projeto Singleton utilizado para que haja apenas uma instancia da classe e reutilizada em toda a aplicação.</p> <p>E a classe AlertMessage utiliza do Singleton, onde o construtor é privado para impedir a criação de novas instâncias, também sendo reutilizado durante a aplicação.</p>
---	---

DaoJDBC – DAO	
	<p>A classe DAOJDBC implementa uma versão do DAO mais genérica, enquanto não somente essa classe, como também diversas outras classes DAO implementam o padrão DAO separando a lógica de acesso a dados da lógica.</p>

ShowAlert (Classe EmprestimoController) – Strategy	
<pre data-bbox="336 1290 1302 1491">private void showAlert(String title, String content, Alert.AlertType alertType) {     Alert alert = new Alert(alertType);     alert.setTitle(title);     alert.setContentText(content);     alert.showAndWait(); }</pre>	<p>Na classe empréstimo controladora há um método que usa o padrão Strategy, utilizando em diferentes tipos de alerta para ajudar o usuário no entendimento das informações e interações que estão sendo feitas.</p>

O projeto em si – MVC	
-----------------------	--

	<p>O projeto em si, possui o padrão de projeto principal o MVC como mencionado anteriormente. Sendo:</p> <ul style="list-style-type: none"> <li>- App na pasta principal do projeto, a parte de controller.</li> <li>- App na pasta the other sources, a parte de view.</li> <li>- Modelo sendo a pasta responsável pela parte de model</li> </ul>
--	--

- f. Na camada de persistência deve ser utilizado o padrão DAO. (pode utilizar qualquer tecnologia para realizar a persistência).

Resposta:

As classes DAO foram utilizadas para a persistência do projeto, no caso elas foram implementadas DAO para cada modelo existente no projeto para adicionar as responsabilidades.

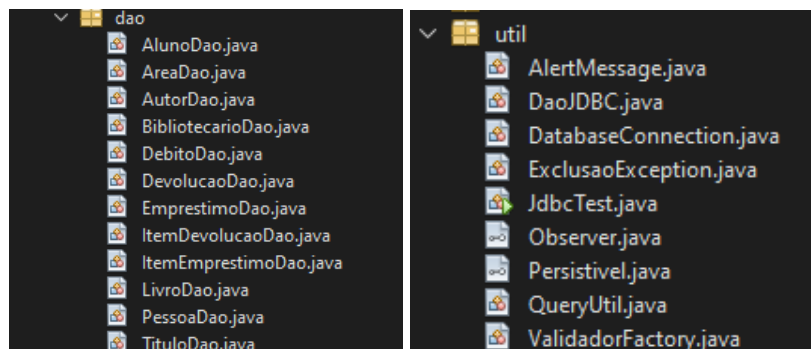


Figura 1. Amostra do uso de classes DAO no projeto

A tecnologia para realizar essa persistência foi o PostgreSQL usando o JDBC para isso.

- g. Discuta qual seria o custo para o projeto suporte um outro tipo de SGBD como mecanismo de armazenamento persistente.

Resposta:

O projeto quando implementado usando outro tipo SGBD como parte de tecnologia de armazenamento persistente, tem um custo que pode ser analisado de várias formas desde o tempo que leva o desenvolvimento até o nível de manutenção que será realizado no futuro. Primeiramente quando foi realizado o sistema de biblioteca era um projeto feito usando JPA com Hibernate, que infelizmente não garantiu bons resultados e diversos bugs que não eram facilmente solucionados e que foi muito mais fácil de solucionar usando o

JDBC.

Mas analisando bem alguns pontos voltado para uma implementação geral sem o uso somente do JDBC.

Há a necessidade de conhecimento de outros SGBDs e o funcionamento deles presentes no sistema, desde o Oracle até o PHP Server entre outras formas de armazenamentos, necessitando conhecimento das consultas e da integração com o código necessitando de bibliotecas ou frameworks de conexão para garantir a melhor compatibilidade com o SGBD. Isso já garante um custo de tempo de desenvolvimento maior para aqueles que não tem entendimento.

A complexidade de uso de drivers de conexão, isso se dá por conta de cada SGBD possui um tipo diferente de driver para conexão com o banco de dados. Cada tipo de configuração vai variar para cada tipo de banco de dados, isso é, se caso use MYSQL, PGSQL, SQL Server poderia usar os drivers do JDBC, pois a implementação do JDBC permite o uso com esses SGBD como foi usado no sistema de biblioteca acima.

Os níveis de manutenção isso porque vai depender de como for feita a abstração da camada de persistência. Como foi realizado o mapeamento do projeto para o modelo objeto-relacional. Além disso pode haver as questões de custos da infraestrutura, pois pode ser necessário hospedar o banco de dados, podendo ter custos de licenciamento e hardware.

O custo disso pode variar dependendo do SGBD que está sendo utilizado, dependendo muito a complexidade do projeto no caso do sistema de biblioteca que ainda não é em nível extremamente alto ainda assim já o custo de tempo de desenvolvimento, tendo diferentes formas de estruturação da camada de persistência. Para garantir que o uso de outros SGBD deem bons resultados uma boa ideia seria boas práticas e ferramentas que possibilitem isso, como ORM.