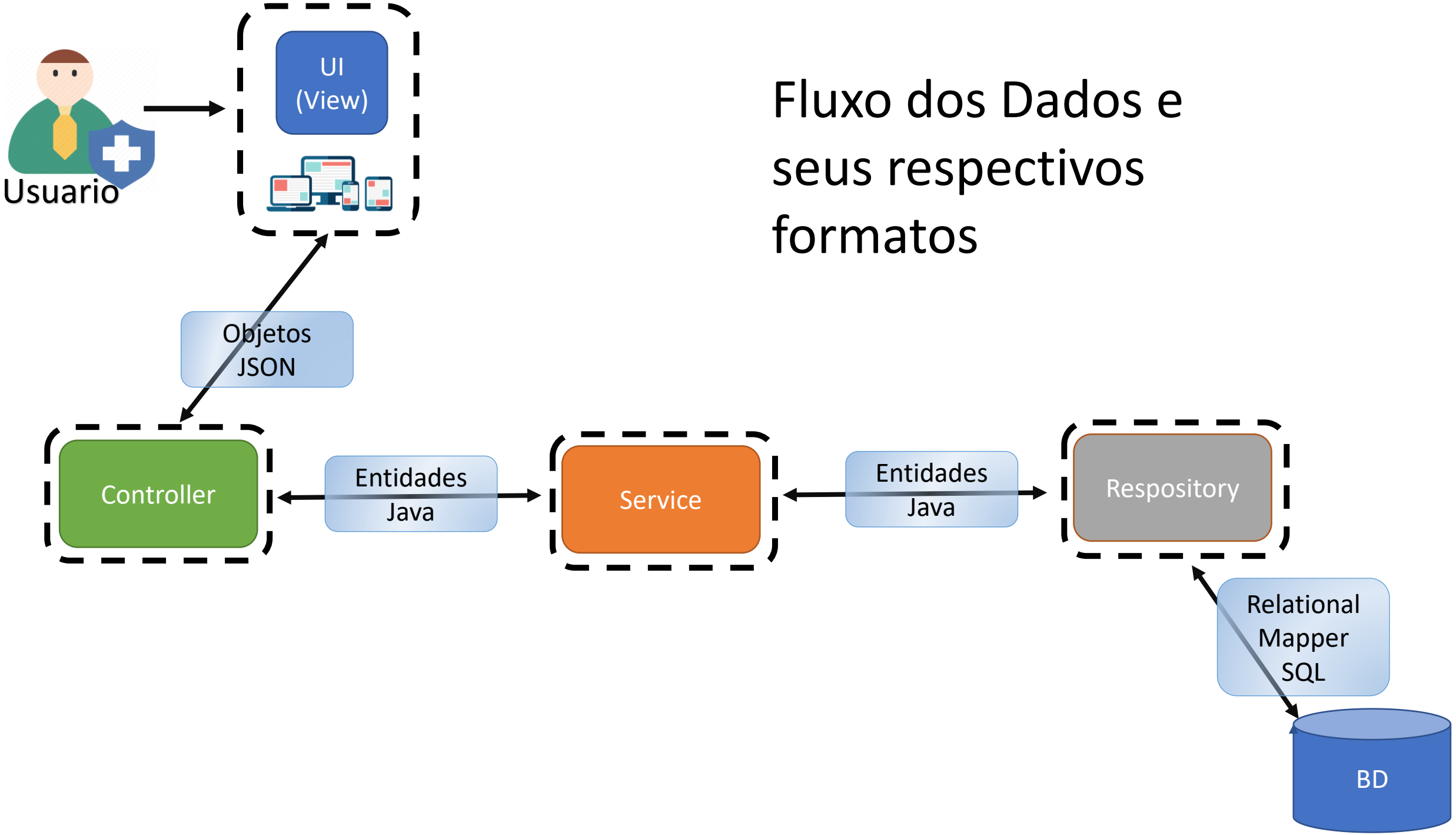
Ciência da Computação

Programação Orientada a Objetos Avançado

Prof. Luciano Rodrigo Ferretto

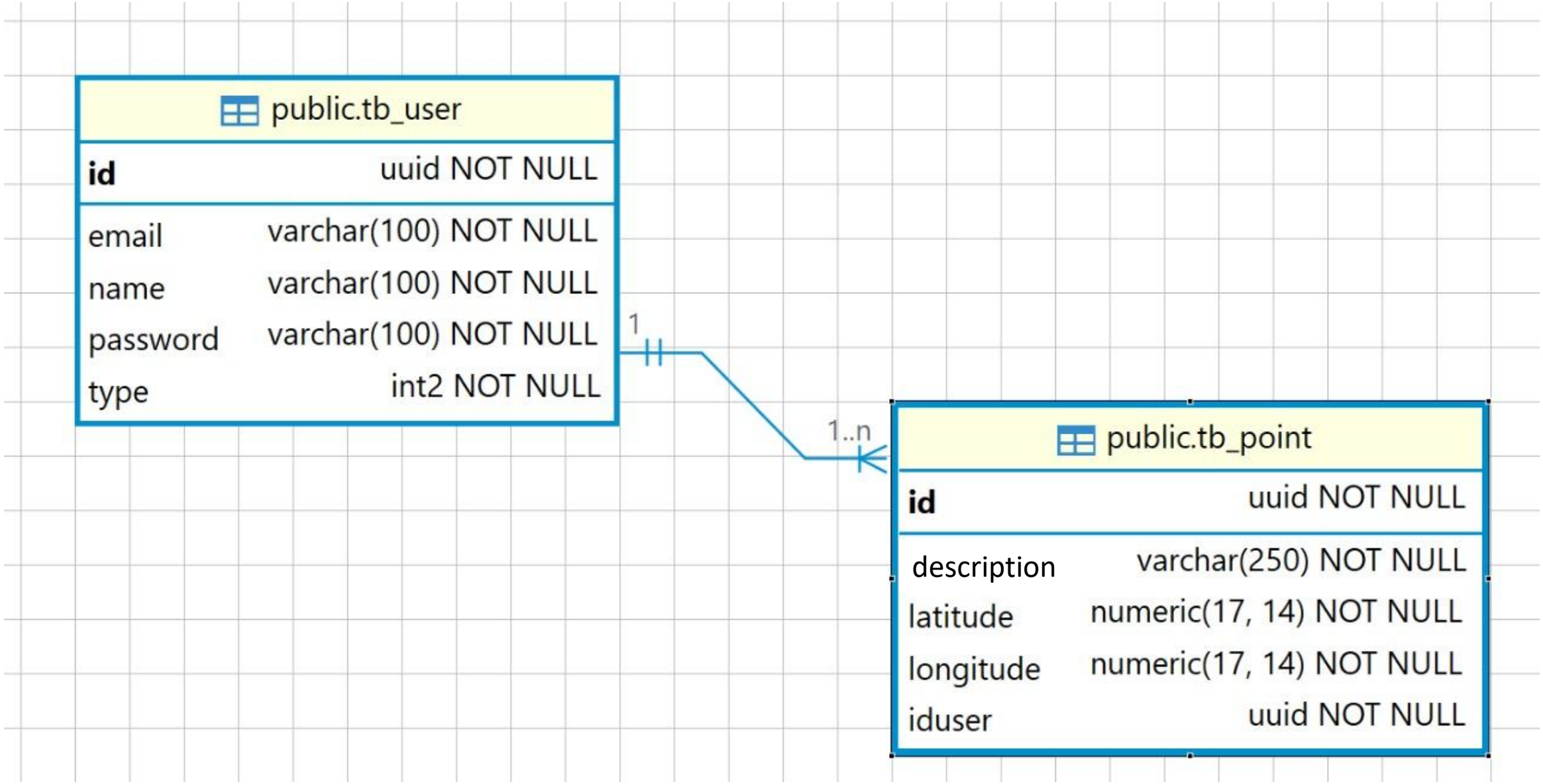
Fluxo dos Dados e seus respectivos formatos



Banco de Dados Relacionais

Relational Database (BD Relacional)

- Tabelas interconectadas.
  - Cada tabela é como uma planilha gigante, cheia de linhas e colunas.
  - Cada linha é uma entrada única (registro, entidade)
  - Cada coluna é um tipo diferente de informação sobre essa entrada.
- As tabelas podem ser relacionadas (vinculadas) por meio de chaves.
  - Isso economiza espaço e ajuda na consistência dos dados.
  - Se alguém mudar uma informação em um lugar, todos os lugares relacionados à essa informação automaticamente se atualizam


public.tb_user
id uuid NOT NULL
email varchar(100) NOT NULL
name varchar(100) NOT NULL
password varchar(100) NOT NULL
type int2 NOT NULL
1
1..n
public.tb_point
id uuid NOT NULL
description varchar(250) NOT NULL
latitude numeric(17, 14) NOT NULL
longitude numeric(17, 14) NOT NULL
iduser uuid NOT NULL

SQL
Structured Query Language

SQL

- SQL, ou Structured Query Language, é uma linguagem de programação projetada para **gerenciar**, **manipular** e **consultar** bancos de dados relacionais.

- Essa linguagem oferece uma abordagem estruturada para interações com sistemas de gerenciamento de banco de dados (SGBD ou DataBase Manager System - DBMS), possibilitando a realização de diversas operações sobre conjuntos de dados armazenados.

SQL – Principais instruções

- **SELECT (SELECIONAR)**: Permite a extração de dados específicos de uma tabela.
  - SELECT uuid, nome, email FROM user;

- **INSERT (INSERIR):** Facilita a adição de novos registros a uma tabela.
  - INSERT INTO artist (uuid, nome, image) VALUES (‘xxx-xx-xx', ‘Raul Seixas’, ‘http://images.com/raul.jpg);

- **UPDATE (ATUALIZAR)**: Possibilita a modificação de registros existentes.
  - UPDATE music SET duration = ‘3:25' WHERE uuid = ‘yyyy-yyy-yy’;

- **DELETE (EXCLUIR):** Permite a remoção de dados.
  - DELETE FROM playlist_music WHERE music_uuid = ‘xxxx-xxx-xx’ AND playlist_uuid=‘yyyy-yy-tt’;

- **JOIN (JUNTAR):** Facilita a combinação de dados de duas ou mais tabelas com base em relações específicas.
  - SELECT playlist.name, user.name, user.username FROM playlist JOIN user ON playlist.user_uuid= user.uuid;.

Registro Relacional x Objeto

Vamos imaginar alguns dados no PostgreSQL

tb_point

id	latitude	longitude	description	id_user
3c344532-3245-4f82-b310-8c5822058685	-28.26537262	-52.39748352	Atitus: Campus Santa Teresinha	5f7b82c2-5634-4696-aa0e-4064c67a39e3

Relacionamento

tb_user

id	name	email	password	type
5f7b82c2-5634-4696-aa0e-4064c67a39e3	Luciano Rodrigo Ferretto	luciano.ferretto@atitus.edu.br	4f904a77b123a8614c489e72494b1a85	0

Como fica a nível de **Objetos**

register				
id	latitude	longitude	description	id_user
3c344532-3245-4f82-b310-8c5822058685	-28.26537262	-52.39748352	Atitus: Campus Santa Teresinha	5f7b82c2-5634-4696-aa0e-4064c67a39e3

Relacionamento

user				
id	name	email	password	type
5f7b82c2-5634-4696-aa0e-4064c67a39e3	Luciano Rodrigo Ferretto	luciano.ferretto@atitus.edu.br	4f904a77b123a8614c489e72494b1a85	0


```
{
    "id": "3c344532-3245-4f82-b310-8c5822058685",
    "latitude": -28.26537262,
    "longitude": -52.39748352,
    "description": "Atitus: Campus Santa Teresinha",
    "user": {
        "id": "5f7b82c2-5634-4696-aa0e-4064c67a39e3",
        "name": "Luciano Rodrigo Ferretto"
        "email": "luciano.ferretto@atitus.edu.br"
        "password": "4f904a77b123a8614c489e72494b1a85"
        "type": 0
    }
}
```

“No braço”

Primeiramente precisaríamos criar a consulta SQL para buscar os dados no BD.

```
select register.*, user.*
    from register
    join user on register.id_user= user.id
    where register.id = ?;
```

Após, teríamos que criar as instâncias dos objetos e atribuir os valores aos respectivos atributos

```
User user = new User();
user.setName(rs.name);
...

Register register = new Register();
register.setUser(user);
register.setDescription(rs.description);
...
```

ORM – Object Relational Mapper

ORM – Object Relational Mapper

- ORM (Mapeamento Objeto-Relacional, em português). É uma técnica de programação que visa facilitar a interação entre sistemas orientados a objetos e bancos de dados relacionais.

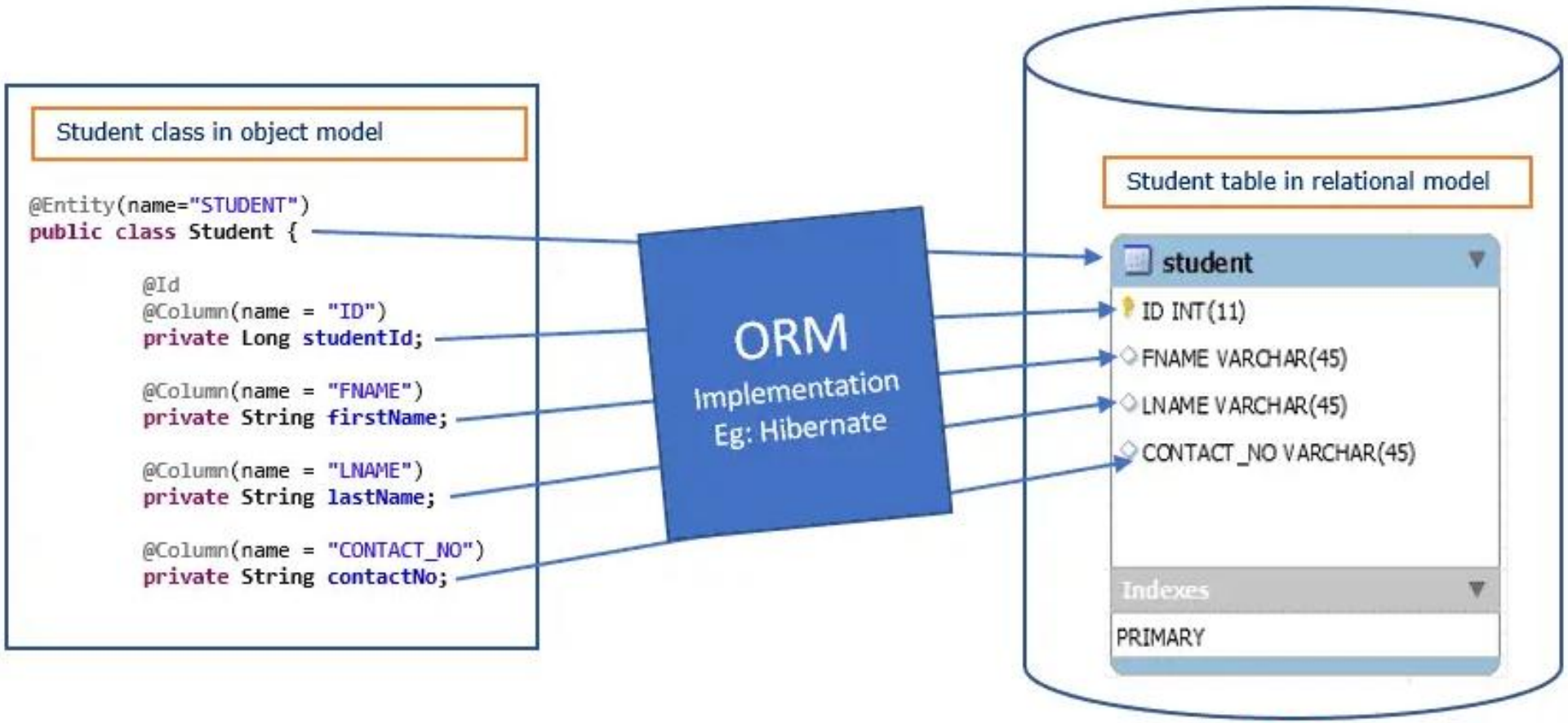
- Em vez de escrever consultas SQL diretamente, você pode usar classes e métodos em uma linguagem de programação orientada a objetos para realizar operações no banco de dados.

- O ORM **mapeia objetos** da linguagem de programação para tabelas no banco de dados e vice-versa.

ORM – Object Relational Mapper

- Torna o código mais legível, facilita a manutenção e reduz a quantidade de código necessário.
- Permite que os desenvolvedores usem conceitos orientados a objetos ao lidar com dados do banco de dados.
- Frameworks populares que implementam ORM incluem:
  - Django para Python
  - Hibernate para Java
  - Entity Framework para .NET
    - Nhibernate (inspirado no Hibernate)
  - Eloquent para PHP.

ORM – Object Relational Mapper



ORM implements responsibility of mapping the Object to Relational Model.

JavaByDeveloper

JPA – Java Persistence API

JPA – Java Persistence API

- Especificação de API padrão do Java para mapeamento objeto-relacional e gerenciamento de persistência de dados.

- Define uma interface comum para frameworks de persistência em Java, permitindo que os desenvolvedores escrevam código que seja independente do fornecedor do banco de dados subjacente.

- **Annotations**
- **EntityManager**
  - Fornece métodos para realizar operações de CRUD (Create, Read, Update, Delete) e gerenciar transações, tornando a persistência de dados em Java mais eficiente e abstrata em relação ao banco de dados subjacente
- **JPQL**
  - Java Persistence Query Language: JPA tem sua própria linguagem de consulta chamada JPQL. Ela é uma linguagem orientada a objetos semelhante ao SQL, mas opera em termos de entidades gerenciadas pela JPA.
- **Transações**
- **Portabilidade**
- **Frameworks JPA**
  - JPA é uma especificação, então você precisa de uma implementação concreta para usá-la.
    - Hibernate
    - EclipseLink
    - Apache OpenJPA.

Hibernate

Hibernate

- Hibernate é uma implementação específica do Java Persistence API (JPA).
  - Em outras palavras, a JPA define como a persistência deve funcionar em Java, e o Hibernate é uma biblioteca que segue essas diretrizes.
  - Esses recursos extras podem incluir extensões da JPA, funcionalidades específicas do Hibernate, e otimizações para melhorar o desempenho.

Hibernate

- Independência de Banco de Dados
  - MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, H2 Database, DB2, SQLite, Sybase, Informix, etc.
  - Pode ser configurado para trabalhar com bancos de dados personalizados, desde que exista um dialeto adequado para a comunicação com o banco de dados em questão.
- Gerenciamento de Transações
- Consultas HQL
  - Hibernate Query Language (HQL) é uma linguagem de consulta orientada a objetos semelhante ao SQL, mas que opera em termos de classes e objetos persistentes.
- Cache
- Suporte a Herança e Associações

Ciência da
Computação

Spring Data JPA

Spring Data JPA

Spring Data JPA

- Spring Data JPA é uma extensão do framework Spring que simplifica ainda mais o desenvolvimento de aplicações que utilizam a JPA
- Fornece um conjunto de abstrações e facilita a implementação de repositórios de dados, reduzindo a quantidade de código **boilerplate** necessário para realizar operações de persistência.

  - Boilerplate: Em programação de computadores, código boilerplate ou boilerplate se refere a seções de código que devem ser incluídas em muitos lugares com pouca ou nenhuma alteração. (https://pt.wikipedia.org/wiki/Boilerplate_code)

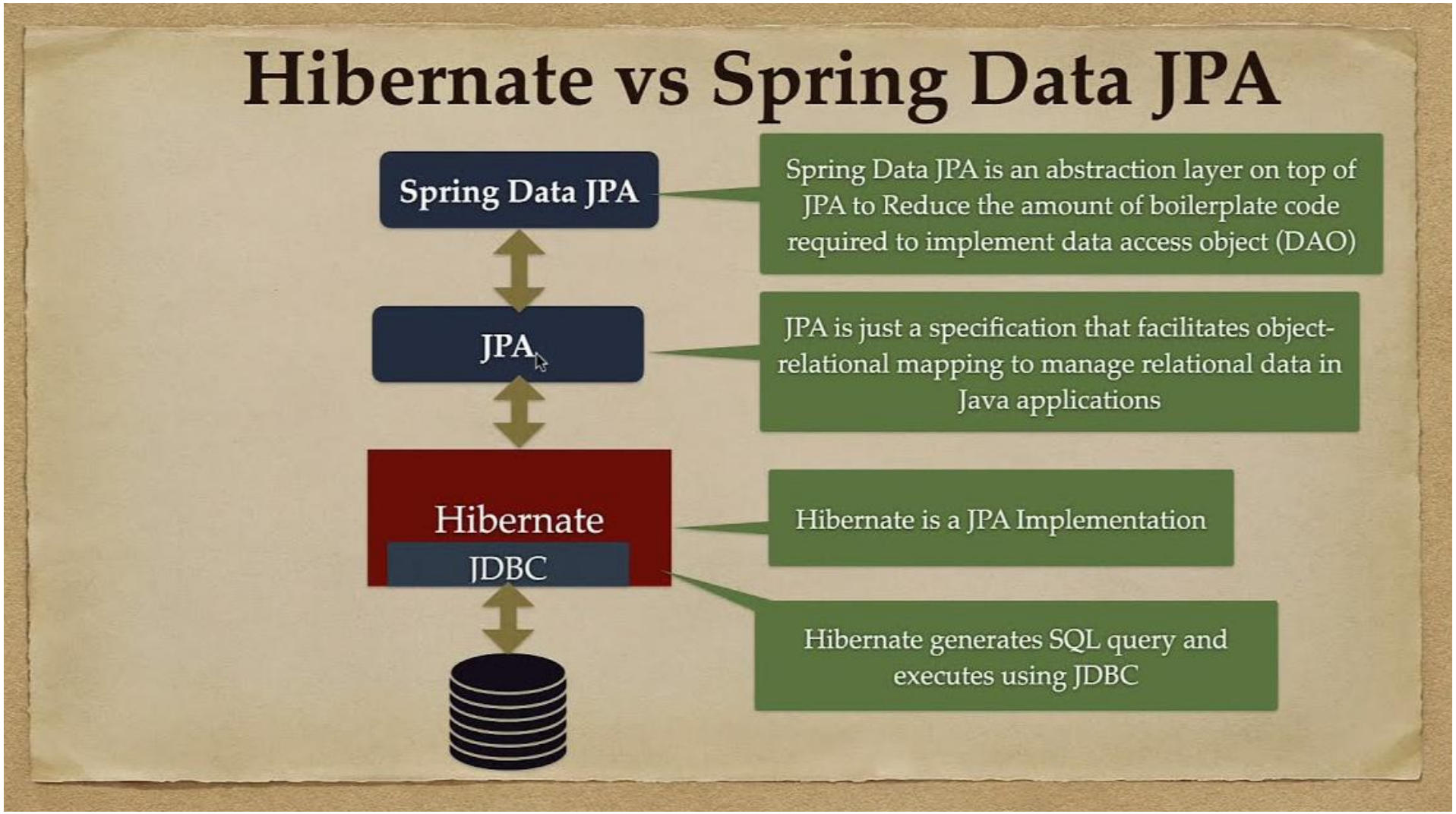
Spring Data JPA

- **Repositórios**
  - Interfaces que estendem uma interface base fornecida pelo Spring Data JPA.
  - Essas interfaces oferecem métodos comuns para realizar operações de leitura e gravação no banco de dados, eliminando a necessidade de escrever consultas SQL manualmente.
- **Query Methods**
  - Ao nomear métodos nas interfaces de repositório de acordo com convenções específicas, o Spring Data JPA gera automaticamente consultas SQL correspondentes. Isso torna as consultas mais legíveis e elimina a necessidade de escrever consultas manualmente.
- **Suporte a Transações**
- **Integração com o Spring Framework**
- **Suporte a Outros Bancos de Dados**
  - Embora o foco principal seja na JPA, o Spring Data oferece suporte a vários bancos de dados, permitindo que você troque o provedor JPA subjacente sem alterar significativamente o código.

Spring Data JPA

Portanto, o Spring Data JPA complementa o Hibernate e a JPA, simplificando ainda mais o desenvolvimento de aplicações baseadas em Java que precisam interagir com bancos de dados relacionais.

JPA – Hibernate – Spring Data JPA - JDBC



Lets’go Developers!!!!!!!