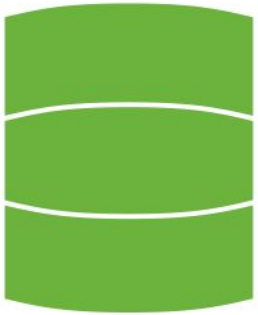
Spring Data - Modelado

UNRN Universidad Nacional de Río Negro

LIA LABORATORIO DE INFORMÁTICA APLICADA

Spring Data



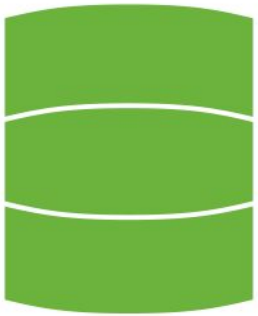
Spring Data es un proyecto de Spring Framework.

“La misión de Spring Data es proporcionar un modelo de programación familiar y consistente basado en Spring para el acceso a datos, al tiempo que conserva las características especiales de la base de datos subyacente.”

https://spring.io/projects/

https://spring.io/projects/spring-data

Spring Data



Su objetivo es unificar y facilitar el acceso a tecnologías de acceso a datos y de esta manera facilitar el desarrollo de aplicaciones que interactúan con bases de datos

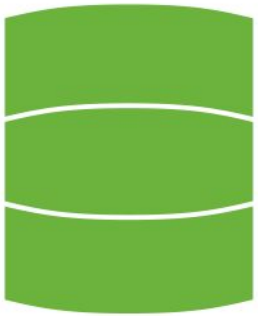
Tanto bases de datos relacionales, bases de datos NoSQL, servicios basados en la nube, etc.

Nos aporta la mayor parte del código que tendríamos que implementar para trabajar con esas tecnologías.

UNRN

Spring Data

LIA LABORATORIO DE INFORMÁTICA APLICADA



Combina patrones de diseño comunes utilizados en el acceso a datos, como el patrón Repository, con características específicas de Spring Framework, como la inyección de dependencias y la administración transaccional.

Reduce la cantidad del código repetitivo necesario para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos.

Consultas personalizadas, paginación, ordenación y soporte para consultas nativas de la base de datos.

Facilita la integración con Spring, Spring Boot, Spring MVC, etc.

UNRN

Spring Data - Características

LIA LABORATORIO DE INFORMÁTICA APLICADA

Es un proyecto general que contiene muchos subproyectos que son específicos de una base de datos dada.

https://programmerclick.com/article/8067144330/

UNRN

JPA

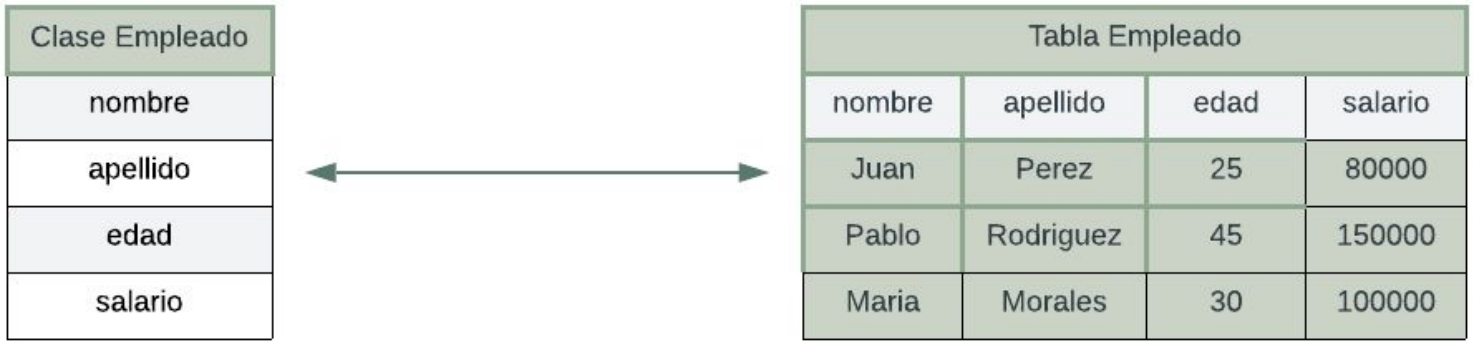
LIA LABORATORIO DE INFORMÁTICA APLICADA

https://www.programaenlinea.net/la-api-de-persistencia-de-java-que-es-jpa/

JPA

- JPA (Java Persistence API) es una especificación de Java que facilita el desarrollo de aplicaciones que interactúan con bases de datos relacionales.
- Proporciona una interfaz de alto nivel para el mapeo de objetos Java a tablas de bases de datos..
- Es una API de persistencia de POJOs (Plain Old Java Object). Es decir, objetos simples que no heredan ni implementan otras clases . No deben extender de ninguna clase e implementar ninguna interface

JPA


Clase Empleado
nombre
apellido
edad
salario
Tabla Empleado
nombre
apellido
edad
salario
Juan
Perez
25
80000
Pablo
Rodriguez
45
150000
Maria
Morales
30
100000

UNRN

Entidades

LIA LABORATORIO DE INFORMÁTICA APLICADA

- En JPA, una entidad es una clase Java que representa un objeto persistente, es decir, un objeto que se almacena y recupera de la base de datos.

- Las entidades se mapean a tablas de las bases de datos y pueden contener atributos simples, relaciones con otras entidades y anotaciones que indican cómo se realiza el mapeo.

UNRN

Anotaciones

LIA LABORATORIO DE INFORMÁTICA APLICADA

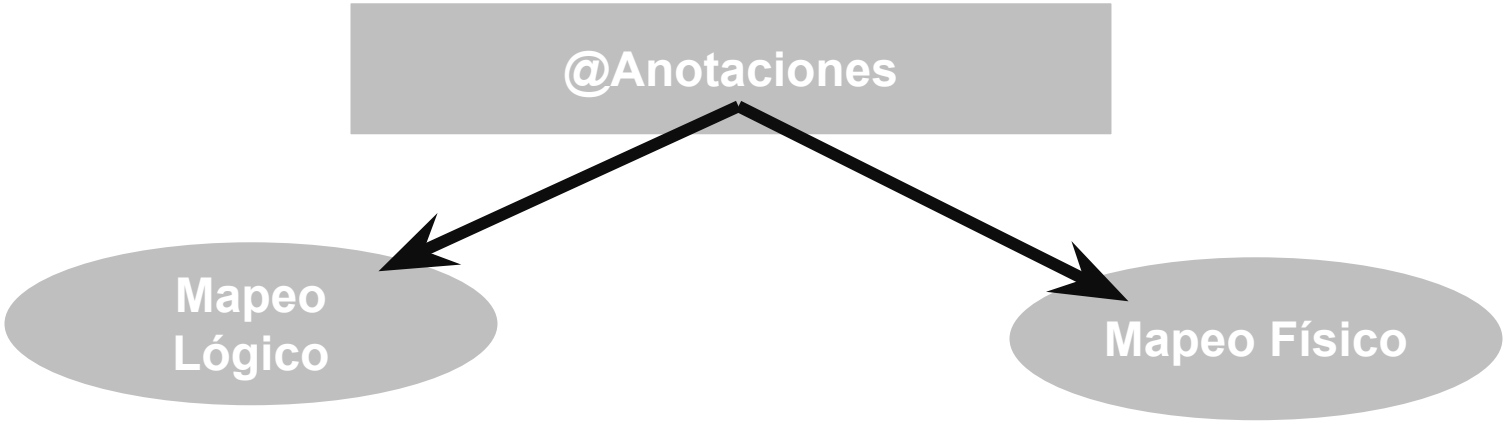
- El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren solamente ficheros descriptores XML.

UNRN

Anotaciones

LIA LABORATORIO DE INFORMÁTICA APLICADA

- Las anotaciones de JPA se clasifican en dos categorías:



Permiten describir modelo de objeto, asociaciones de clase, etc

@OneToOne,@OneToMany,etc

Describen esquemas físicos de base de datos, tablas, columnas, índices, etc.

@Column,@JoinColumn,etc

UNRN

Anotaciones

LIA LABORATORIO DE INFORMÁTICA APLICADA

- De clase

@Entity	Indica que la clase anotada es una entidad	
@Table	Especifica el nombre de la tabla relacionada con la entidad.	
	name	Nombre de la tabla

UNRN

Anotaciones

LIA LABORATORIO DE INFORMÁTICA APLICADA

- De propiedad

@Column	Indica el nombre de la columna en base de datos.	
	name	Nombre de la columna
@Enumerated	Indica cómo se debe mapear un atributo de enumeración (enum) a una columna en la base de datos.	
	value	Determina el tipo de mapeo a utilizar. Puede usarse el enumerador EnumType
@Id	Indica que la propiedad es la clave primaria de una entidad.	
@GeneratedValue	Especifica la estrategia de generación de la clave primaria.	
	strategy	Indica la estrategia a seguir para la obtención de un nuevo identificador. Permite utilizar el enum GenerationType

Anotaciones

- De propiedad

@Trasient	Se utiliza para marcar un atributo en una entidad como no persistente, lo que significa que no se debe almacenar en la base de datos.	
@JoinColumn	Especifica la columna en la base de datos que se utilizará para establecer una relación entre dos entidades.	
	name	Nombre de la columna

UNRN

Anotaciones

LIA LABORATORIO DE INFORMÁTICA APLICADA

- De Relación

@OneToOne	Especifica un valor único que contiene una relación uno-a-uno con otro objeto.	
	cascade	Indica el tipo de operación de cascada a realizar. Permite utilizar el enumerador CascadeType
	fetch	Indica la forma en que se consultarán las entidades asociadas.
@OneToMany	Especifica una relación uno-a-muchos.	
	cascade	Especifica el tipo de cascada. Permite utilizar el enumerador CascadeType
	mappedBy	Especifica la propiedad de la entidad hija que sirve para enlazar con la entidad principal.

Anotaciones

- De Relación

@ManyToOne	Especifica una relación muchos-a-uno	
	Se complementa con la anotación @JoinColumn para especificar el nombre de la columna que se utiliza como clave foránea de la relación	
@ManyToMany	Especifica una relación muchos-a-muchos	
	mappedBy	Si la relación es bidireccional, específica el extremo propietario (el que posee la clave principal)
	Se complementa con la anotación @JoinTable para especificar el nombre de la tabla intermedia y las columnas que se utilizarán como claves extranjeras para las entidades.	

Anotaciones

- De Relación

@JoinTable	Especifica datos relativos a la tabla de unión en la relación. Se añade en el extremo propietario	
	name	Indica el nombre de la tabla intermedia
	joinColumns	Especifica las columnas de la tabla intermedia que se utilizarán como claves foráneas para la entidad propietaria de la relación
	inverseJoinColumns	Especifica las columnas de la tabla intermedia que se utilizarán como claves foráneas para la entidad relacionada
	Se complementa con la anotación @JoinColumn para especificar los nombres de las columnas.	

UNRN

Enumeradores

LIA LABORATORIO DE INFORMÁTICA APLICADA

- CascadeType. Tipos de operación de cascada:
  - REMOVE
  - REFRESH
  - PERSIST
  - MERGE
  - DETACH
  - ALL
- GenerationType. Tipos de estrategia para la generación de identificadores:
  - AUTO: el proveedor de persistencia escoge el método más adecuado según el modelo de base de datos
  - IDENTITY: el proveedor de persistencia escoge un identificador basándose en una columna *identity* de la tabla
  - SEQUENCE: utiliza una secuencia de la base de datos
  - TABLE: utiliza una tabla auxiliar para asegurarse de que la clave es verdaderamente única

UNRN

Enumeradores

LIA LABORATORIO DE INFORMÁTICA APLICADA

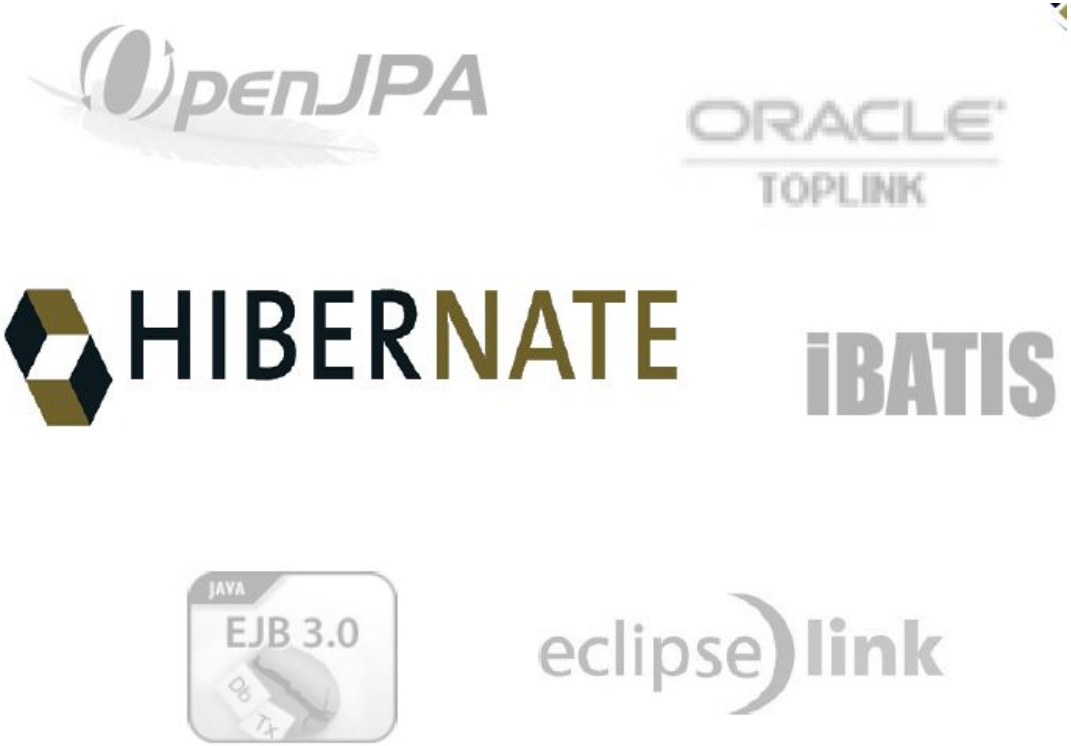
- FetchType. Tipos de obtención de entidades relacionadas:
  - LAZY
  - EAGER
- EnumType. Tipo de enumerador. Indica cómo va a ser el tipo de persistencia en la base de datos:
  - ORDINAL: formato predeterminado. En base de datos, se almacenará el ordinal correspondiente al valor del enum.
  - STRING: en base de datos, se almacenará el valor String correspondiente al enum. Usando este tipo de persistencia, se evitan errores en caso de que los valores del enum cambien o se intercalen valores nuevos, que puedan alterar el ordinal asignado.

Frameworks–JPA providers

UNRN

Frameworks–JPA providers

LIA LABORATORIO DE INFORMÁTICA APLICADA



UNRN

Hibernate

LIA LABORATORIO DE INFORMÁTICA APLICADA

Hibernate es un framework de mapeo objeto-relacional (ORM) ampliamente utilizado en el desarrollo de aplicaciones Java.

Proporciona una capa de abstracción sobre JDBC (Java Database Connectivity) y permite a los desarrolladores trabajar con objetos Java en lugar de escribir directamente consultas SQL.

Con Hibernate, los desarrolladores pueden trabajar con objetos Java y dejar que el framework se encargue de traducir y ejecutar las operaciones de persistencia en la base de datos.

Hibernate

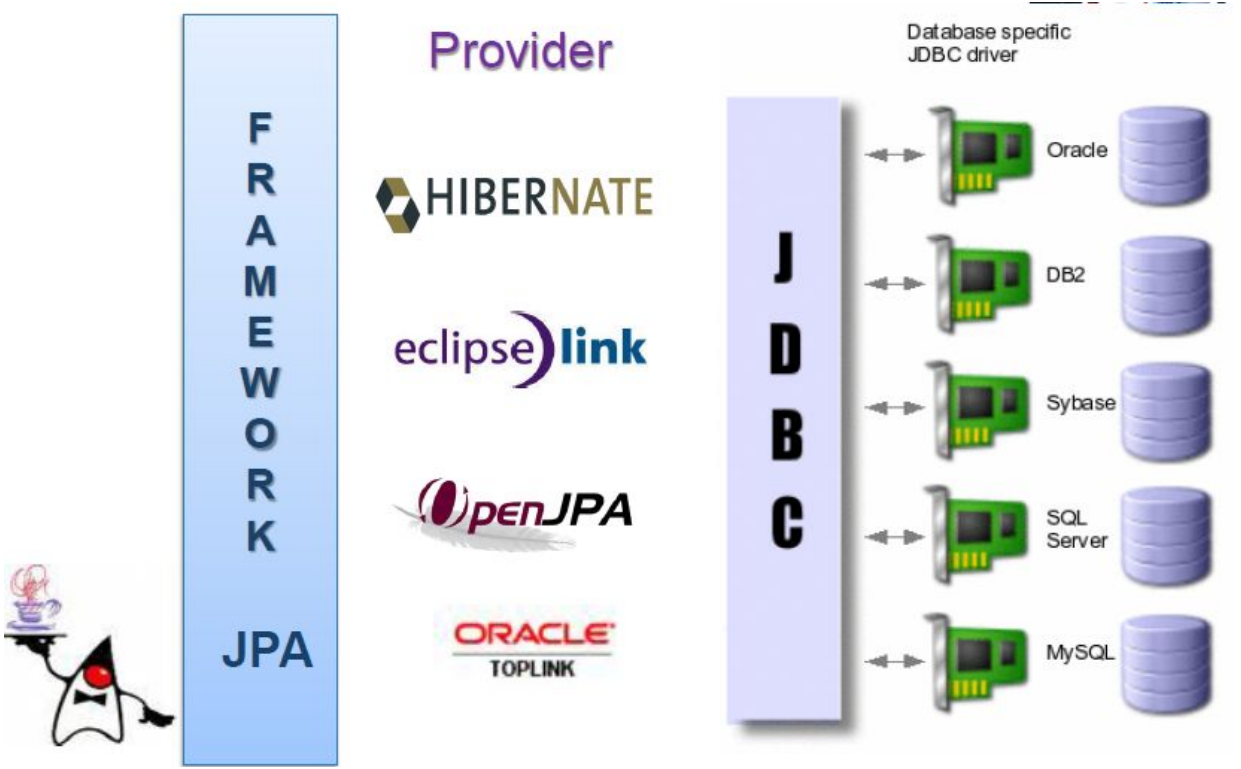
Algunas de las características y ventajas clave de Hibernate son:

- Mapeo objeto-relacional.
- Transparencia de persistencia.
- Consultas y recuperación de datos.
  - HQL
  - Consultas SQL nativas.
- Caché de primer y segundo nivel.

UNRN

Frameworks–JPA providers

LIA LABORATORIO DE INFORMÁTICA APLICADA



UNRN

Spring Data JPA

LIA LABORATORIO DE INFORMÁTICA APLICADA

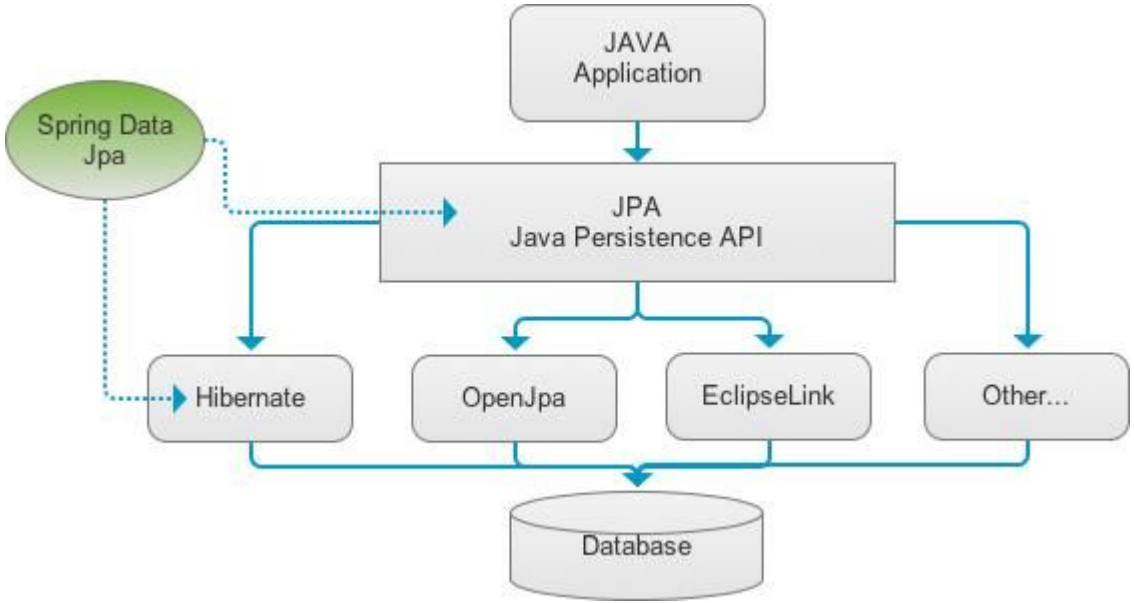
Spring Data JPA, es el módulo de Spring Data que se enfoca en la integración con JPA. Proporciona una capa adicional de abstracción sobre JPA y Hibernate, permitiendo un manejo más sencillo y eficiente de las operaciones de persistencia.

Proporciona una serie de características y funcionalidades adicionales, como consultas automáticas, paginación, ordenamiento y manejo de transacciones. También facilita la creación de consultas personalizadas y ofrece soporte para diferentes bases de datos, lo que te permite cambiar fácilmente el proveedor de JPA sin afectar tu código.

UNRN

Spring Data JPA

LIA LABORATORIO DE INFORMÁTICA APLICADA

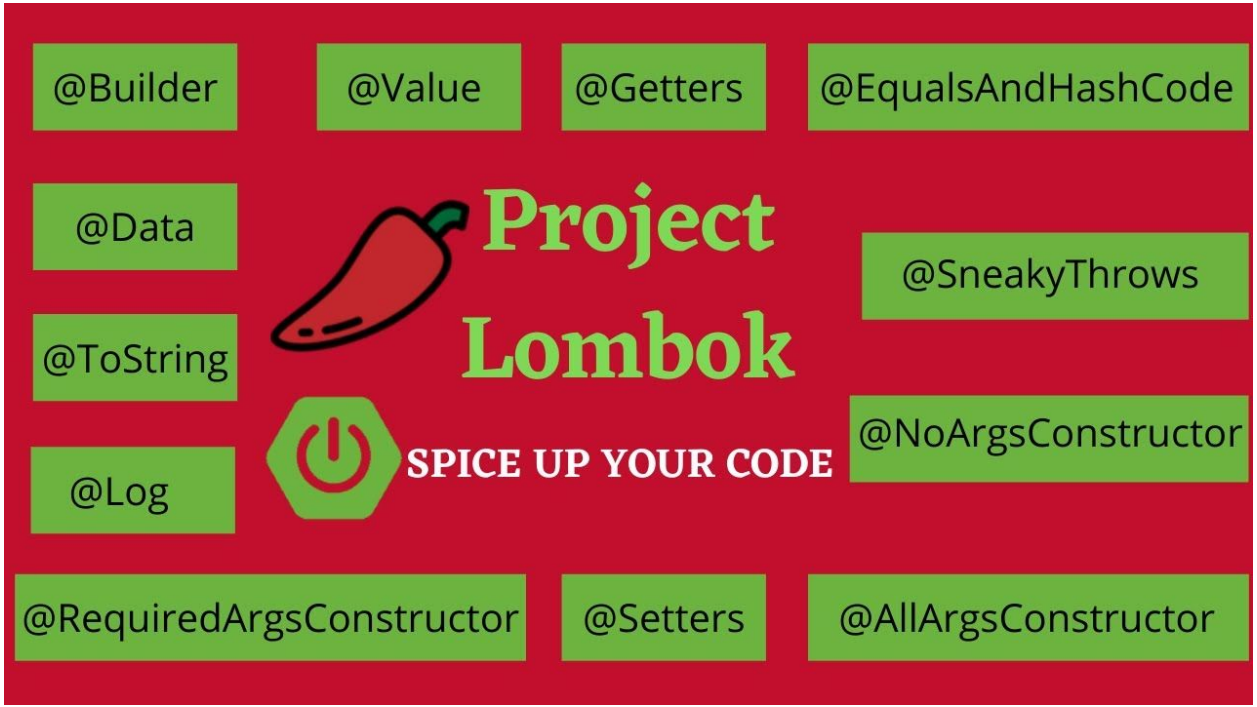


https://programmerclick.com/article/8067144330/

UNRN

Lombok

LIA LABORATORIO DE INFORMÁTICA APLICADA



https://medium.com/javarevisited/what-is-project-lombok-and-how-to-use-it-with-spring-boot-d6034f865541

UNRN

Lombok

LIA LABORATORIO DE INFORMÁTICA APLICADA

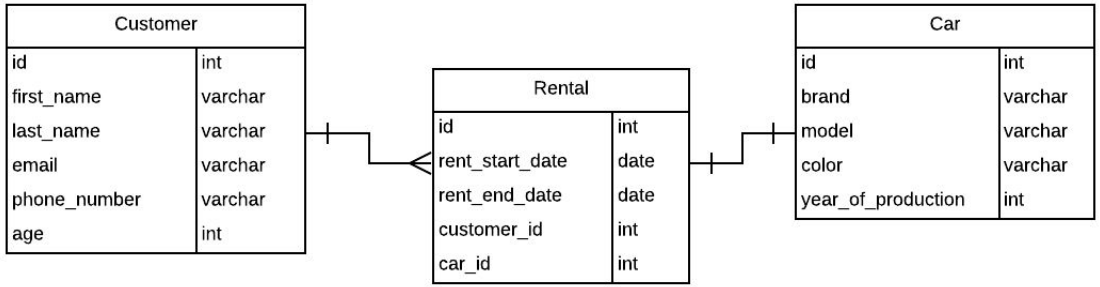
- Lombok es una biblioteca de Java que se utiliza para reducir la cantidad de código repetitivo en las clases, como la generación automática de getters, setters, constructores, entre otros.

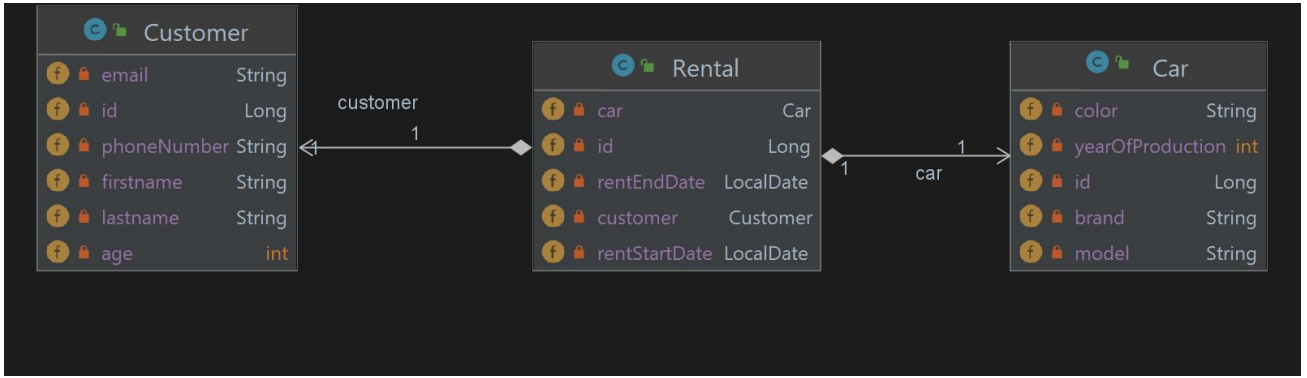
- En lugar de escribir manualmente estas estructuras repetitivas, Lombok se encarga de generarlas en tiempo de compilación.

- Logra esto mediante el uso de anotaciones en el código fuente de Java y luego generando el código repetitivo después de compilar el código.

- Podemos ver algunas de estas anotaciones en https://projectlombok.org/features/

Modelo DER - Modelo de Clases





UNRN

Entidad Customer

LIA LABORATORIO DE INFORMÁTICA APLICADA

Customer	
id	int
first_name	varchar
last_name	varchar
email	varchar
phone_number	varchar
age	int

```java
package com.example.demo.domain;

import jakarta.persistence.*;

@Entity
@Table(name = "customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstname;

    @Column(name = "last_name")
    private String lastname;

    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    private int age;
}
```

UNRN

Entidad Rental

LIA LABORATORIO DE INFORMÁTICA APLICADA

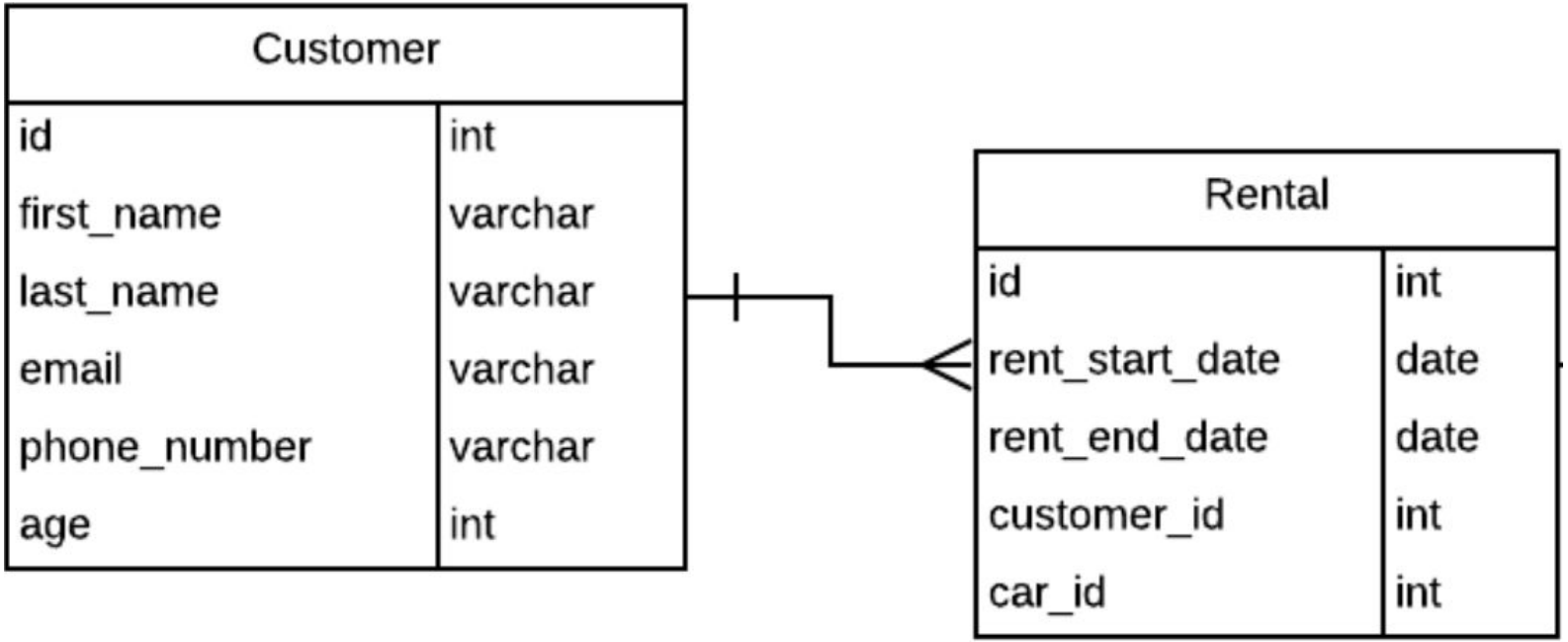
Rental	
id	int
rent_start_date	date
rent_end_date	date
customer_id	int
car_id	int

```java
package com.example.demo.domain;

import jakarta.persistence.*;

import java.time.LocalDate;

@Entity
@Table(name = "rental")
public class Rental {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "rent_start_date")
    private LocalDate rentStartDate;

    @Column(name = "rent_end_date")
    private LocalDate rentEndDate;
}
```

UNRN

Relación DER Customer - Rental

LIA LABORATORIO DE INFORMÁTICA APLICADA



UNRN

Relación Customer - Rental

LIA LABORATORIO DE INFORMÁTICA APLICADA

Rental.java

```java
@Entity
@Table(name = "rental")
public class Rental {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "rent_start_date")
    private LocalDate rentStartDate;

    @Column(name = "rent_end_date")
    private LocalDate rentEndDate;

    @ManyToOne
    @JoinColumn(name = "customer_id")
    private Customer customer;
}
```

Customer.java

```java
@Entity
@Table(name = "customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstname;

    @Column(name = "last_name")
    private String lastname;

    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    private int age;
}
```

UNRN

Repository

LIA LABORATORIO DE INFORMÁTICA APLICADA

```java
package com.example.demo.repository;

import com.example.demo.domain.Customer;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer, Long> {
}
```

UNRN

CommandLineRunner

LIA LABORATORIO DE INFORMÁTICA APLICADA

DemoApplication.java

```java
package com.example.demo;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    no usages
    @Bean
    public CommandLineRunner commandLineRunner(ApplicationContext ctx) {
        return args -> {
            System.out.println("La aplicación se ha iniciado");
        };
    }

}
```

UNRN

Conectarse a la base de datos

LIA LABORATORIO DE INFORMÁTICA APLICADA

```
# Configuración de la base de datos
spring.datasource.url=jdbc:postgresql://<hostname>:<port>/<database-name>
spring.datasource.username=<username>
spring.datasource.password=<password>

# Otras configuraciones
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.driver-class-name=org.postgresql.Driver
```

UNRN

Conectarse a la base de datos

LIA LABORATORIO DE INFORMÁTICA APLICADA

application.properties

```
# Configuración de la base de datos
spring.datasource.url=jdbc:postgresql://silly.db.elephantsql.com:5432/mvamkius
spring.datasource.username=mvamkius
spring.datasource.password=EX8UCBGKx_bmUtmWPOGfTl-h6NYreSoh

# Otras configuraciones
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.driver-class-name=org.postgresql.Driver
```

UNRN Universidad Nacional de **Río Negro**

LIA LABORATORIO DE INFORMÁTICA APLICADA

unrn.edu.ar @unrionegro