Vacinação

- **Aluno: Daniel Nogueira da Costa - 202105024**

Framework de persistência Spring-boot data:

- Mapeamento ORM por meio de Annotations Java
- Utiliza a interface JPA e a implementação Hibernate

Exemplo de mapeamento:

- Código:

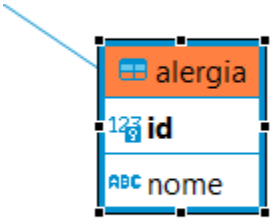
```java
package com.github.kalheeso.provax.domain;

import jakarta.persistence.*;
import lombok.Data;
import lombok.Setter;

@Data
@Entity
public class Alergia {
    @Setter(onMethod_ = @Deprecated)
    @Id
    @GeneratedValue(strategy = GenerationType.IDEN
    private long id;
    private String nome;

    protected Alergia() {}
}
```

  - A anotação **@Entity** eu indico que essa é uma classe para ser mapeada para o banco de dados.
  - A anotação **@Id** indica a chave primária dessa tabela.
- Tabela:



Exemplo de relacionamento entre as tabelas:

- Código:

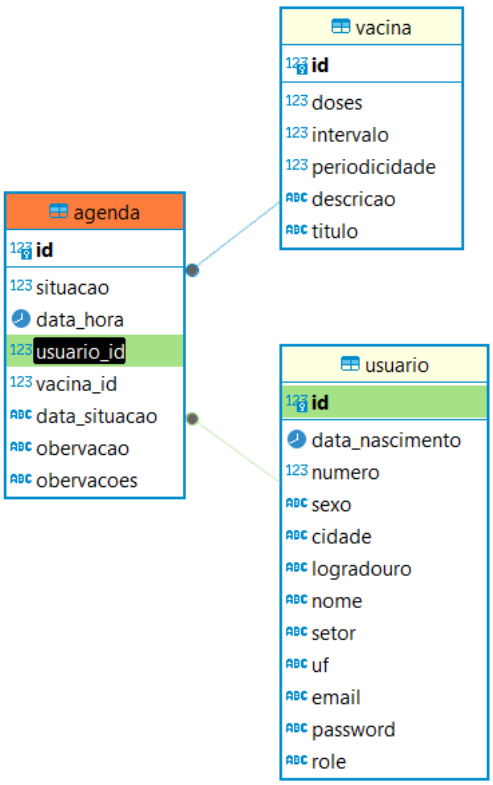
```java
@Data
@Entity
public class Agenda implements Serializable {
    @Setter(onMethod_ = @Deprecated)
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    // Data e Hora do agendamento
    private LocalDateTime dataHora;
    private Situacao situacao;
    private String dataSituacao;
    private String obervacoes;

    @ManyToOne
    @JoinColumn(nullable = false, name = "vacina_id")
    private Vacina vacina;

    @ManyToOne
    @JoinColumn(nullable = false, name = "usuario_id")
    private Usuario usuario;
}
```

Esse é um relacionamento 1-N

- Tabela:



Armazena o usuario_id e o vacina_id das entidades que essa tabela se relaciona.

Relacionamentos N-N:

- Código:

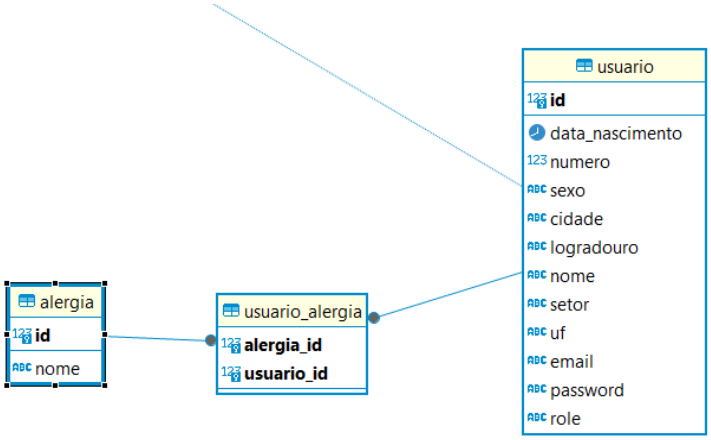
  Relacionamento de Usuário com Alergia

```java
@Getter
@Entity
public class Usuario {
    @Setter(onMethod_ = @Deprecated)
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(unique = true, nullable = false)
    private String email;
    @JsonIgnore
    @Column(nullable = false)
    private String password;
    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private UsuarioRole role;
    private String nome;
    private LocalDate dataNascimento;
    private char sexo;
    private String logradouro;
    private int numero;
    private String setor;
    private String cidade;
    private String uf;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
            name = "usuario_alergia",
            joinColumns = @JoinColumn(name = "usuario_id"),
            inverseJoinColumns = @JoinColumn(name = "alergia_id")
    )
    @OnDelete(action = OnDeleteAction.CASCADE)
    private Set<Alergia> alergias = new HashSet<>();
```

- Tabela:



Por ser um relacionamento N-N foi criada uma nova tabela para armazenar as chaves primárias e relacionar as duas tabelas mencionadas.

Conexão com o Banco:

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=${URL_DATABASE}
spring.datasource.username=${USERNAME_DATABASE}
spring.datasource.password=${PASSWORD_DATABASE}
spring.datasource.driver-class-name=org.postgresql.Driver

api.security.jwt.secret=${JWT_SECRET:persistencia-ufg}

server.port=8080
```

No arquivo **application.properties** usando Spring-boot é configurada o valor da url do banco, o nome do usuário que tem acesso ao banco e a senha deste usuário. Além desses dados pertinentes ao acesso, também nesse arquivo é informado o drive do banco de dados.

Operações de acesso ao banco de dados:

```
import com.github.kalheeso.provax.domain.Usuario;
import org.springframework.data.repository.CrudRepository;

import java.util.Optional;

public interface UsuarioRepository extends CrudRepository<Usuario, Long> {
    Optional<Usuario> findByEmail(String email);

    boolean existsByEmail(String email);
}
```

Usando Spring por meio dessas interfaces Repository é feito o acesso ao banco de dados, algumas operações já são definidas, como o save, outras é necessário fazer a declaração através do padrão estabelecido, como mostrado na imagem **findByEmail.** É feita somente a declaração da operação usando o padrão e a implementação fica a cargo do Spring. Para operações mais complexas e personalizadas tem que fazer a declaração e implementação.