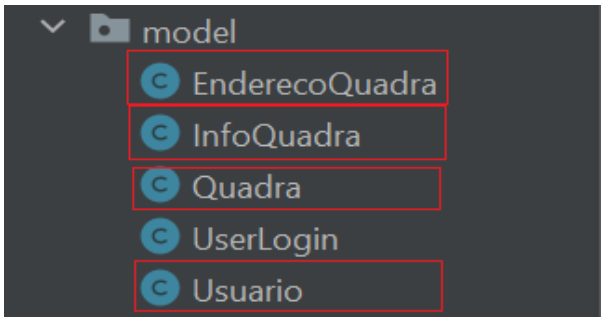
Documentação backend Share Fields versão 1.0

Entidades e relacionamento entre elas

Você pode checar a tabela de relacionamento do banco de dados no pdf que está nessa mesma pasta. Mas basicamente temos 4 entidades que se relacionam em diferentes níveis:



O primeiro relacionamento se trata de **um para um(OneToOne)** entre a entidade de **EnderecoQuadra** e **Quadra.**

```java
@Entity
public class EnderecoQuadra {


 @OneToOne(mappedBy = "endereco")
 @JsonIgnoreProperties({"endereco","proprietarioQuadra","infoQuadra"})
 private Quadra quadra;
```

```java
@Entity
@Table(name = "tb_quadra")
public class Quadra {

@OneToOne(cascade = CascadeType.ALL)
@JsonIgnoreProperties({"quadra"})
private EnderecoQuadra endereco;
```

A anotação que cria essa relação é feita acima do atributo que representa a entidade, no caso o atributo de **Quadra** na entidade de **EnderecoQuadra** e o atributo de **EnderecoQuadra** na entidade de **Quadra**. Quando fazemos a relação precisamos definir o lado dominante da relação, nesse caso o lado dominante é a entidade de **Quadra**, então o parâmetro **mappedBy** fica na entidade não dominante e referenciando o atributo da entidade dominante que faz a relação.

A entidade **Quadra** possui também o relacionamento de **muitos para um(ManyToOne)** com a entidade de **Usuario**, na qual podem exsitir muitas quadras cadastradas por um **Usuario.** A anotação de **JsonIgnoreProperties** serve para evitarmos recursividade quando chamamos um recurso, então ignoramos propriedades que podem se chamar infinitamente (e alguns atributos que não interessam em determinado get).

```
@Entity
@Table(name = "tb_quadra")
public class Quadra {


@ManyToOne
@JsonIgnoreProperties({"senha","email","quadrasDoUsuario","usaQuadras"})
private Usuario proprietarioQuadra;
```

Na outra ponta, a entidade de **Usuario** vai possuir um relacionamento de **um para muitos(OneToMany)** com a entidade **Quadra**, o parâmetro **mapped by** em relacionamentos assim sempre fica no lado de **um para muitos** indicando também que o lado dominante é sempre o de **muitos para um.** O parâmetro **CascadeType.ALL** significa que toda mudança no lado dominante vai ser refletido no lado não dominante.

```
@Entity
@Table(name = "tb_usuario")
public class Usuario {

@OneToMany(mappedBy = "proprietarioQuadra", cascade = CascadeType.ALL)
@JsonIgnoreProperties({"proprietarioQuadra","quadrasDoUsuario"})
private List<Quadra> quadrasDoUsuario;
```

A entidade **Quadra** possui o relacionamento de **muitos para um** com a entidade I**nfoQuadra.**

```java
@Entity
@Table(name = "tb_quadra")
public class Quadra {


  @OneToMany(mappedBy = "quadra", cascade = CascadeType.ALL)
  @JsonIgnoreProperties("quadra")
  private List<InfoQuadra> infoQuadra;
```

```java
@Entity
@Table(name = "tb_infoQuadra")
public class InfoQuadra {

@ManyToOne
@JsonIgnoreProperties({"infoQuadra","endereco"})
private Quadra quadra;
```

E por fim o relacionamento de **muitos para muitos(ManyToMany)** entre a entidade **Usuario** e a entidade **InfoQuadra**, na qual é gerado uma tabela com as chaves primarias de ambas as entidades. Esse relacionamento serve para dizer que um **Usuario** pode se cadastrar em vario horários de várias quadras e varias **InfoQuadra** podem ter vários usuários cadastrados.

```java
@Entity
@Table(name = "tb_infoQuadra")
public class InfoQuadra {

   @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
   @JoinTable(
           joinColumns = @JoinColumn(name = "jogadores_id"),
           inverseJoinColumns = @JoinColumn(name = "quadras_id") )
   @JsonIgnoreProperties({"quadra", "usaQuadras","quadrasDoUsuario","email","senha","nome"})
   private Set<Usuario> jogadores = new HashSet<>();
```

```java
@Entity
@Table(name = "tb_usuario")
public class Usuario {

   @ManyToMany(mappedBy = "jogadores")
   @JsonIgnoreProperties({"jogadores","proprietarioQuadra"})
   private Set<InfoQuadra> usaQuadras = new HashSet<>();
```

Models:

public class EnderecoQuadra {

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private long id;

@OneToOne(mappedBy = "endereco")
@JsonIgnoreProperties({"endereco","proprietarioQuadra","infoQuadra"})
private Quadra quadra;
```

```
    private int cep;

    private String rua;

    private int numero;

    private String complemento;

    private String bairro;

    private String cidade;

    private String uf;

    private String referencia;
}


public class InfoQuadra {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @ManyToOne
    @JsonIgnoreProperties({"infoQuadra","endereco"})
    private Quadra quadra;

    private boolean disponivel;

    private String dataDisponivel;

    private String horaInicio;

    private String horaFim;

    @ManyToMany(cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
    @JoinTable(
            joinColumns = @JoinColumn(name =
"jogadores_id"),
```

```
            inverseJoinColumns = @JoinColumn(name =
"quadras_id") )
    @JsonIgnoreProperties({"quadra",
"usaQuadras","quadrasDoUsuario","email","senha","nome"})
    private Set<Usuario> jogadores = new HashSet<>();
}


public class Quadra {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Size(min = 5, max = 50)
    private String nome;

    private String imagem;

    @NotNull
    private String modalidade;

    @NotNull
    private int qtdJogadoresMax;

    @NotNull
    @Size(min = 5, max = 2000)
    private String descricao;

    @OneToOne(cascade = CascadeType.ALL)
    @JsonIgnoreProperties({"quadra"})
    private EnderecoQuadra endereco;

    @ManyToOne

@JsonIgnoreProperties({"senha","email","quadrasDoUsuario",
"usaQuadras"})
    private Usuario proprietarioQuadra;
```

```
    @OneToMany(mappedBy = "quadra", cascade =
CascadeType.ALL)
    @JsonIgnoreProperties("quadra")
    private List<InfoQuadra> infoQuadra;
}


public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @NotNull
    private boolean disponibilizadorDeQuadra;

    @NotNull
    @Size(min = 3, max = 100)
    private String nome;

    private String avatar;

    @NotNull
    @Size(min = 3, max = 100)
    private String apelido;

    @NotNull
    @Size(min = 5, max = 100)
    private String email;

    @NotNull
    @Size(min = 8, max = 100)
    private String senha;

    @OneToMany(mappedBy = "proprietarioQuadra", cascade =
CascadeType.ALL)

@JsonIgnoreProperties({"proprietarioQuadra","quadrasDoUsua
rio"})
    private List<Quadra> quadrasDoUsuario;
```

```
    @ManyToMany(mappedBy = "jogadores")

@JsonIgnoreProperties({"jogadores","proprietarioQuadra"})
    private Set<InfoQuadra> usaQuadras = new HashSet<>();

}
```

Endpoints

Todos os controladores possuem as opções de **CRUD** completo.

EnderecoQuadraController

(/api/v1/enderecoQuadra) no verbo **get** retorna todos os endereços de quadra, no verbo **post** cria um endereço de quadra via **body,** no verbo **put** atualiza um endereço de quadra via **body.**

(/api/v1/enderecoQuadra/id) passando o id no verbo **get** retorna um endereço de quadra específico de acordo com o id, e no verbo **delete** exclui o recurso com o id especificado.

QuadraController

(api/v1/quadra) no verbo **get** retorna todas as quadras, no verbo **post** cria uma quadra via **body,** no verbo **put** atualiza uma quadra via **body.**

(api/v1/quadra/id) passando o id no verbo **get** retorna uma quadra específica de acordo com o id, e no verbo **delete** exclui o recurso com o id especificado.

InfoQuadraController

(api/v1/infoQuadra) no verbo **get** retorna todas as informações de quadra, no verbo **post** cria uma informação via **body,** no verbo **put** atualiza uma informação de quadra via **body.**

(api/v1/infoQuadra/id) passando o id no verbo **get** retorna uma informação de quadra específica de acordo com o id, e no verbo **delete** exclui o recurso com o id especificado.

(api/v1/infoQuadra/inserir/infoquadra/id/usuario/id) Esse endpoint é para inserir o usuário na informação de quadra e deve ser passado no verbo **put**, passando no primeiro **id** o id da **InfoQuadra** e no segundo **id** o id do **Usuario.**

(api/v1/infoQuadra/remover/infoquadra/id/usuario/id) Esse endpoint é para remover o usuário na informação de quadra e deve ser passado no verbo **put**, passando no primeiro **id** o id da **InfoQuadra** e no segundo **id** o id do **Usuario.**