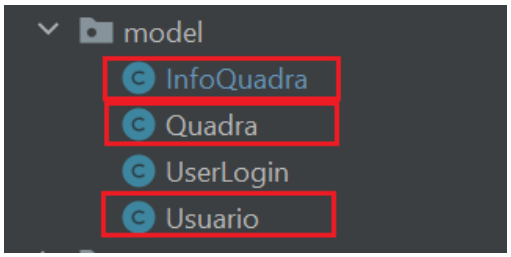
Documentação back-end Share Fields versão 1.0

Entidades e relacionamento entre elas

Você pode checar a tabela de relacionamento do banco de dados no pdf que está nessa mesma pasta com o nome de Sharefields DER. Mas basicamente temos 3 entidades que se relacionam em diferentes níveis:



A entidade **Quadra** possui 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.

```java
@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 e suas constraints:

Classe InfoQuadra:

Id = **Long;**

dataDisponivel**(Não nulo) = String**;

horaInicio**(Não Nulo) = String**;

horaFim**(Não Nulo) = String**;

jogadores = **HashSet<Usuario>**;

Classe Quadra:

Id = **Long;**
nome **(Não Nulo)(Tamanho min=3, max=50) = String**;
imagem **(Não Nulo) = String**;
modalidade **(Não Nulo) = String**;
qtdJogadoresMax **(Não Nulo) = int**;
descricao **(Não Nulo)(Tamanho min=5, max=2000) = String**;
proprietarioQuadra = **Usuario**;
infoQuadra = **List<InfoQuadra>**;
cep **(Não Nulo) = int**;
rua **(Não Nulo) = String**;
numero **(Não Nulo) = int**;
complemento **= String**;
bairro **(Não Nulo) = String**;
cidade **(Não Nulo) = String**;
uf **(Não Nulo) (Tamanho min=2, max=2) = String**;
referencia **= String**;

Classe Usuario:

id **= Long**;
disponibilizadorDeQuadra **(Não Nulo) = boolean**;
nome **(Não Nulo)(Tamanho min=3, max=100) = String**;
avatar **= String**;
apelido **(Não Nulo)(Tamanho min=3, max=100) = String**;
email **(Não Nulo) = String**;
senha **(Não Nulo) = String**;
quadrasDoUsuario **= List<Quadra>**;
usaQuadras **= HashSet<InfoQuadra>**;

Endpoints:

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

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.**