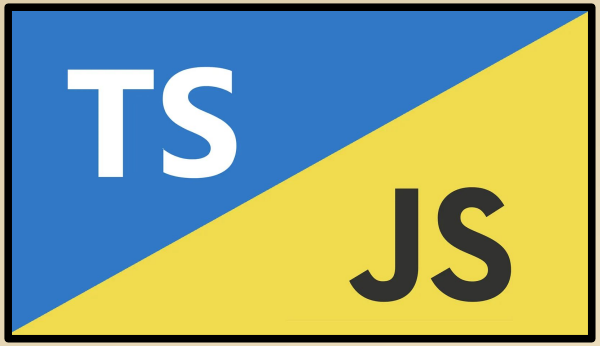
- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

Ambiente


```
//gera package.json
npm init -y
//dependências ambiente produção
npm install @prisma/client
//dependências ambiente local
npm install prisma typescript tsx @types/node --save-dev
//npx gera tsconfig.json
npx tsc --init
//ambiente local
//compila e executa
npx tsx main.ts
```

```
{ //tsconfig.json
 "compilerOptions": {
 "target": "es2020",
```

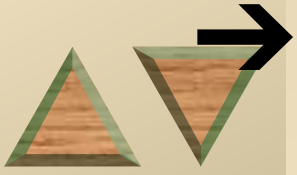
```
{ //Adicionar no package.json
 "prisma": {
  "seed": "npx tsx prisma/seed.ts"
 },
```

TS JS

Introdução

docker-compose.yaml

```yaml
services:
    db:
     image: postgres:15.3
     volumes:
       - ./volumes/postgres/data:/var/lib/postgresql/data
     environment:
       POSTGRES_PASSWORD: thor
       POSTGRES_USER: thor
       POSTGRES_DB: thor
     ports:
       - "5432:5432"
```



docker compose up -d

TS JS

Ambiente

Introdução

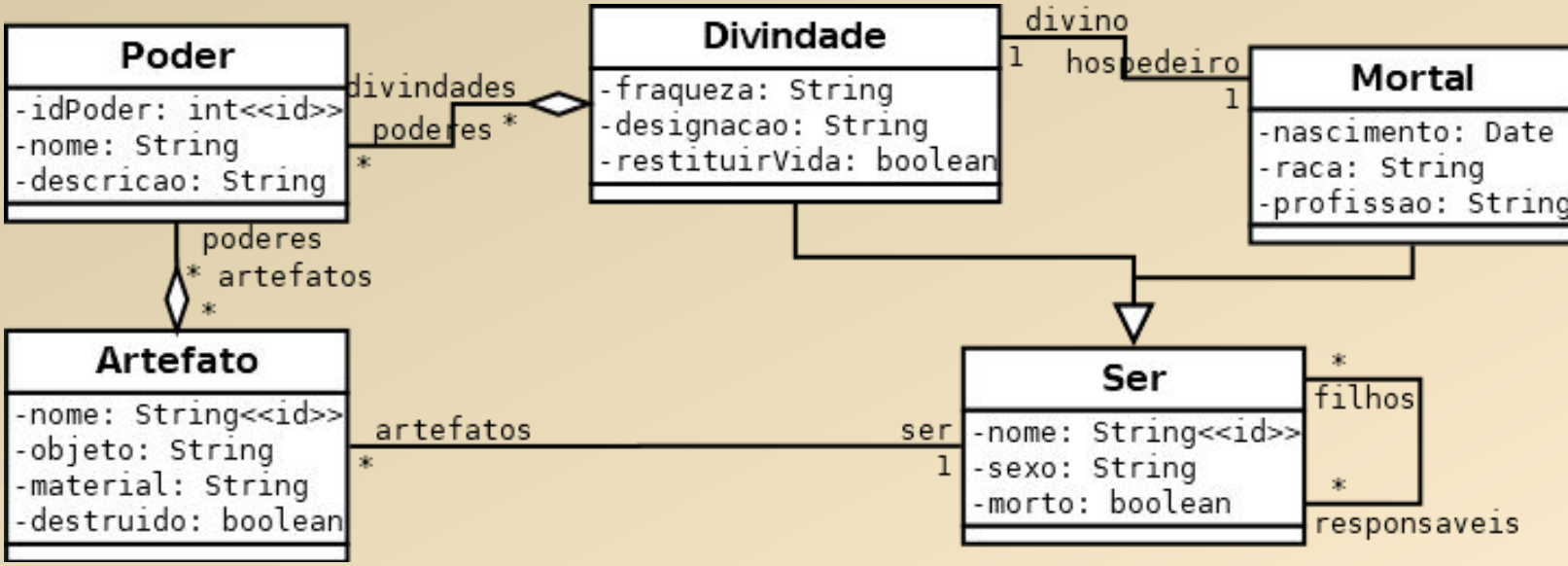
```
//prefixar o prisma com o executor de pacotes:
npx prisma
//./prisma/schema.prisma e .env
npx prisma init --datasource-provider postgresql
```

```
//.env
DATABASE_URL="postgresql://thor:thor@localhost
:5432/thor?schema=public"
```

TS JS

Diagrama Exemplo

Introdução



```
model Poder {
 idPoder Int @id @default(autoincrement())
 nome String @unique
 descricao String
 poderes ArtefatoPoder[]
 divindades PoderDivindade []
}
model PoderDivindade {
 serNome String
 divindade Ser @relation(fields: [serNome], references: [nome])
 poderId Int
 poder Poder @relation(fields: [poderId], references: [idPoder])
 @@id([serNome , poderId])
}
model ArtefatoPoder {
  artefatoNome String
  artefato Artefato @relation(fields: [artefatoNome], references: [nome])
  poderId Int
  poder Poder @relation(fields: [poderId], references: [idPoder])
  @@id([artefatoNome, poderId])
  @@map("ArtefatoPoder")
}
```

```
model Artefato{
  nome String @id
  objeto String
  material String
  destruido Boolean
  nomeSer String
  ser Ser @relation(fields: [nomeSer], references: [nome])
  artefatos ArtefatoPoder[]
}
enum TipoSer {
  SER
  MORTAL
  DIVINDADE
}
model SerSer{
 responsavel Ser @relation("serResponsavel", fields: [nomeResponsavel],
references: [nome])
 nomeResponsavel String
 filho Ser @relation("serFilho", fields: [nomeFilho], references: [nome])
 nomeFilho String
 @@id([nomeResponsavel, nomeFilho])
}
```

```
model Ser {
 nome String @id
 sexo String
 morto Boolean
 tipo TipoSer
 fraqueza String? //Divindade
 designacao String? //Divindade
 restituirVita Boolean? //Divindade
 poderes PoderDivindade[] //Divindade
 nascimento DateTime? //Mortal
 raca String? //Mortal
 profissao String? //Mortal
 artefatos Artefato[]
 responsaveis SerSer[] @relation("serResponsavel")
 filhos SerSer[] @relation("serFilho")
 nomeMortal String? @unique //Divindade
 hospedeiro Ser? @relation("serMortal", fields: [nomeMortal], references:
[nome]) //Divindade
 divino Ser? @relation("serMortal") //Mortal
}
```

TS JS

Ambiente

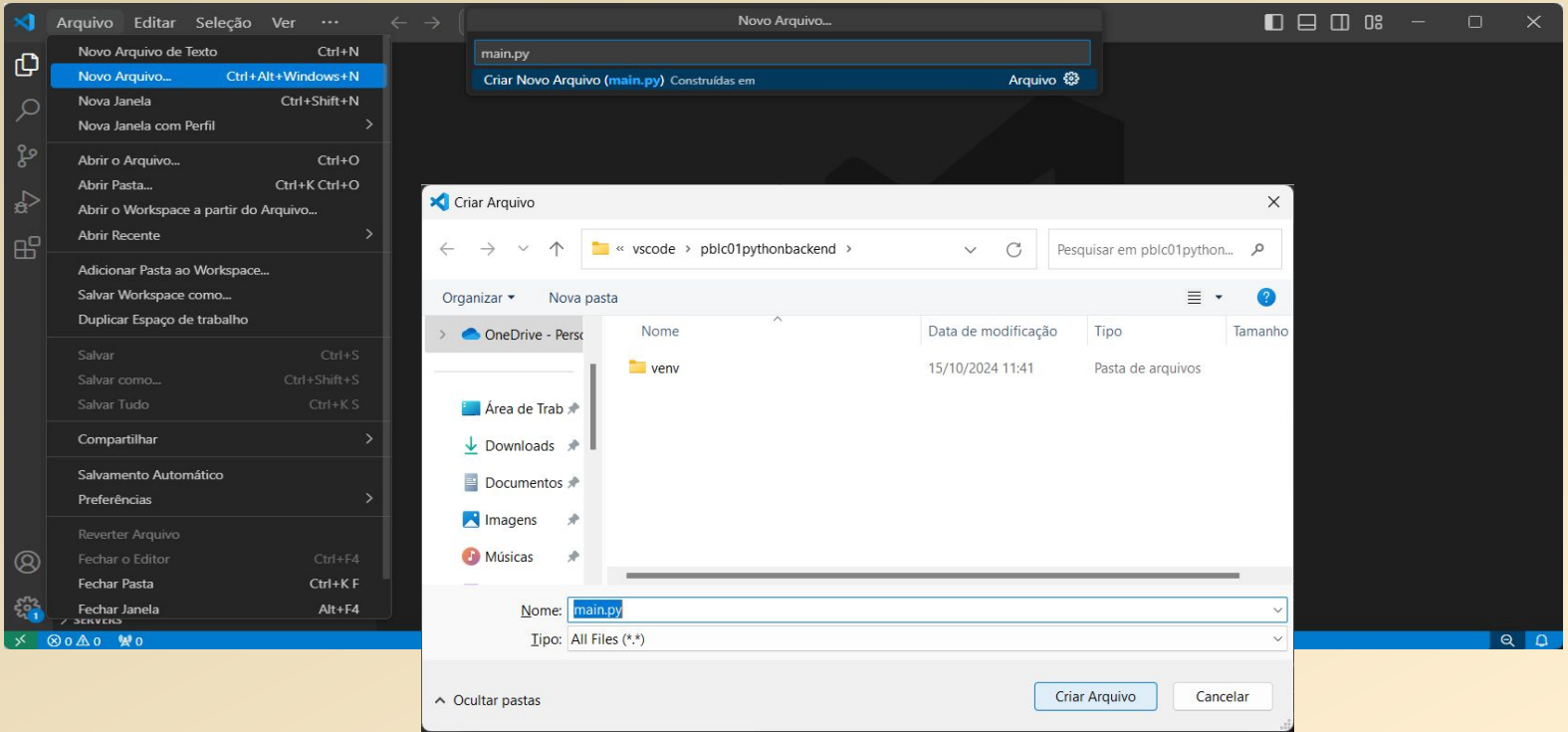
Introdução

```
//gera Prisma Client
npx prisma generate
//gera arquivo de migração e executa SQL no banco
npx prisma migrate dev --name init
//Visualizar dados
npx prisma studio
```

seed.ts

- Arquivo | Novo Arquivo
- /prisma/seed.ts

Ambiente



```
import { PrismaClient} from '../generated/prisma'
const prisma = new PrismaClient()
async function main() {
 console.log('Iniciando o seed...')
 const tecido = await prisma.poder.create({data:
   {nome: 'tecido denso',descricao: 'torna mais forte'}})
 const longe = await prisma.poder.create({data:
   {nome: 'longevidade',descricao: 'evelhecimento muito lento'}})
 const cura = await prisma.poder.create({data:
   {nome: 'fator de cura',descricao: 'recuperação dos ferimentos'}})
 const eletrica = await prisma.poder.create({data:
   {nome: 'eletricidade instantânea',descricao: 'concentra raios elétricos'}})
 const allspeak = await prisma.poder.create({data:
   {nome: 'allspeak',descricao: 'comunicar-se em todos os idiomas'}})
 const disfarce = await prisma.poder.create({data:
   {nome: 'disfarce',descricao: 'muda aparência'}})
 const invoca = await prisma.poder.create({data:
   {nome: 'invocação',descricao: 'retorno do objeto à mão'}})
 const voo = await prisma.poder.create({data:
   {nome: 'vôo',descricao: 'deslocar sobre o ar'}})
 const bifrost = await prisma.poder.create({data:
   {nome: 'energia de Bifrost',descricao: 'conjura um portal'}})
 const escuridao = await prisma.poder.create({data:
   {nome: 'escuridão viva',descricao: 'permite controlar a escuridão'}})
```

```
const blacke = await prisma.ser.create({data:
   {nome: 'Don Blake', sexo: 'Masculino', morto: false, tipo: 'MORTAL', raca:'humano',
profissao:'médico'}})
 const jane = await prisma.ser.create({data:
   {nome: 'Jane Foster', sexo: 'Feminino', morto: true, tipo: 'MORTAL', raca:'humano',
profissao:'astrofísica'}})
 const bill = await prisma.ser.create({data:
   {nome: 'Bill Raio Beta', sexo: 'Masculino', morto: false, tipo: 'MORTAL',
raca:'Korbinita', profissao:'caçador de recompensas'}})
 const gorr = await prisma.ser.create({data:
   {nome: 'Gorr', sexo: 'Masculino', morto: true, tipo: 'MORTAL', raca:'Klyntar',
profissao:'camponês'}})
 const odin = await prisma.ser.create({data:
   {nome: 'Odin', sexo: 'Masculino', morto: true, tipo: 'DIVINDADE', restituirVita: false,
fraqueza: 'tempo sem dormir', designacao: 'protetor dos reinos'}, })
 const thor = await prisma.ser.create({data:
   {nome: 'Thor', sexo: 'Masculino', morto: false, tipo: 'DIVINDADE', restituirVita:
false, fraqueza: 'tornar não digno', designacao: 'deus do trovão', nomeMortal:
blacke.nome}})
 const freya = await prisma.ser.create({data:
   {nome: 'Freya', sexo: 'feminino', morto: false, tipo: 'DIVINDADE', restituirVita:
false, fraqueza: 'flor Lobelia', designacao: 'deusa do amor'}})
 const loki = await prisma.ser.create({data:
   {nome: 'Loki', sexo: 'Masculino', morto: false, tipo: 'DIVINDADE', restituirVita:
false, fraqueza: 'imerso na água', designacao: 'deus da trapaça'}})
```

```
const filhos = await prisma.serSer.createMany({data: [
  {nomeResponsavel:odin.nome, nomeFilho: thor.nome},
  {nomeResponsavel:freya.nome, nomeFilho: thor.nome},
  {nomeResponsavel:odin.nome, nomeFilho: loki.nome},
  {nomeResponsavel:freya.nome, nomeFilho: loki.nome}]})
const poderDiv = await prisma.poderDivindade.createMany({data: [
  {serNome:odin.nome, poderId: tecido.idPoder},
  {serNome:odin.nome, poderId: longe.idPoder},
  {serNome:odin.nome, poderId: cura.idPoder},
  {serNome:thor.nome, poderId: tecido.idPoder},
  {serNome:thor.nome, poderId: longe.idPoder},
  {serNome:thor.nome, poderId: cura.idPoder},
  {serNome:thor.nome, poderId: eletrica.idPoder},
  {serNome:freya.nome, poderId: tecido.idPoder},
  {serNome:freya.nome, poderId: longe.idPoder},
  {serNome:freya.nome, poderId: cura.idPoder},
  {serNome:freya.nome, poderId: allspeak.idPoder},
  {serNome:loki.nome, poderId: tecido.idPoder},
  {serNome:loki.nome, poderId: longe.idPoder},
  {serNome:loki.nome, poderId: cura.idPoder},
  {serNome:loki.nome, poderId: disfarce.idPoder}]})
```

```
 const mjolnir = await prisma.artefato.create({data:
   {nome: 'mjolnir', objeto: 'martelo', material:'uru', destruido: true, nomeSer:
thor.nome}})
 const rompe = await prisma.artefato.create({data:
   {nome: 'Rompe-Tormentas', objeto: 'martelo', material:'uru', destruido: false, nomeSer:
bill.nome}})
 const espada = await prisma.artefato.create({data:
   {nome: 'Necroespada', objeto: 'espada', material:'simbiose All-Black', destruido:
false, nomeSer: gorr.nome}})
 const poderArte = await prisma.artefatoPoder.createMany({data: [
   {artefatoNome:mjolnir.nome, poderId: eletrica.idPoder},
   {artefatoNome:mjolnir.nome, poderId: invoca.idPoder},
   {artefatoNome:mjolnir.nome, poderId: voo.idPoder},
   {artefatoNome:rompe.nome, poderId: eletrica.idPoder},
   {artefatoNome:rompe.nome, poderId: invoca.idPoder},
   {artefatoNome:rompe.nome, poderId: bifrost.idPoder},
   {artefatoNome:espada.nome, poderId: tecido.idPoder},
   {artefatoNome:espada.nome, poderId: longe.idPoder},
   {artefatoNome:espada.nome, poderId: cura.idPoder},
   {artefatoNome:espada.nome, poderId: escuridao.idPoder}]})
}
```

```
main()
.catch((e) => {
  console.error('Erro no seed:', e)
   process.exit(1)
})
.finally(() => {
  prisma.$disconnect();
  console.log('Seed concluído!');
});
```

TS JS

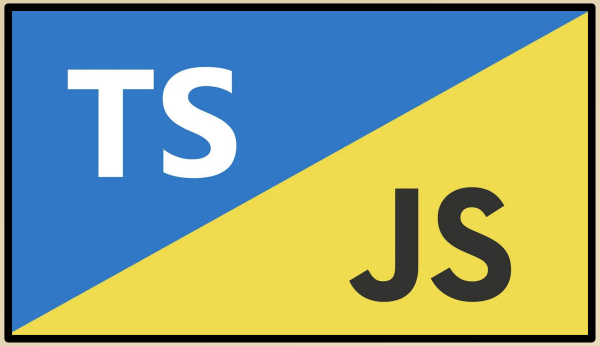
Ambiente

Introdução

```
//Executa as inserções no banco
npx prisma db seed

//Reinicia os dados com os inserts do seed
npx prisma migrate reset
```

- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

TS JS

→ FindMany → retorna lista de resultados

findMany/findFirst

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const seres = await prisma.ser.findMany()
  seres.forEach(s => {
    console.log(s.nome + " ", s.tipo)})
}
main()
```

```
Don Blake   MORTAL
Jane Foster   MORTAL
Bill Raio Beta   MORTAL
Gorr   MORTAL
```

```
Odin   DIVINDADE
Thor   DIVINDADE
Freya   DIVINDADE
Loki   DIVINDADE
```

TS JS

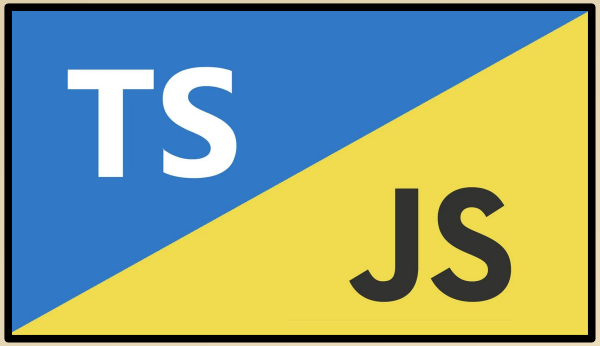
findMany/findFirst

→ FindFirst → retorna primeiro registro que satisfaz o critério.

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const s = await prisma.ser.findFirst()
 if (s != null ){
   console.log(s.nome + " ", s.tipo)
 }
}
main()
```

```
Don Blake  MORTAL
```

- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

TS JS

select:{}/omit:{}

→ select:{} → Especifica quais propriedades serão incluídas no retorno.

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const seres = await prisma.ser.findMany({
  select:{nome: true}})
 seres.forEach(s => {
  console.log(s.nome + " " + s.tipo)})
}
main()
```

```
Don Blake  undefined
Jane Foster  undefined
Bill Raio Beta  undefined
Gorr  undefined
```

```
Odin  undefined
Thor  undefined
Freya  undefined
Loki  undefined
```

select:{} → pode incluir relacionamentos

include:{}

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const seres = await prisma.ser.findMany({
  select:{
   nome: true,
   artefatos :{
    select:{nome: true}}}})
 seres.forEach(s => {
  console.log("ser:"+s.nome)
  s.artefatos.forEach(a =>{
   console.log("artefato:"+
     a.nome)})})
}
main()
```

```
ser:Don Blake
ser:Jane Foster
ser:Bill Raio Beta
artefato:Rompe-Tormentas
ser:Gorr
artefato:Necroespada
ser:Odin
ser:Thor
artefato:mjolnir
ser:Freya
ser:Loki
```

TS JS

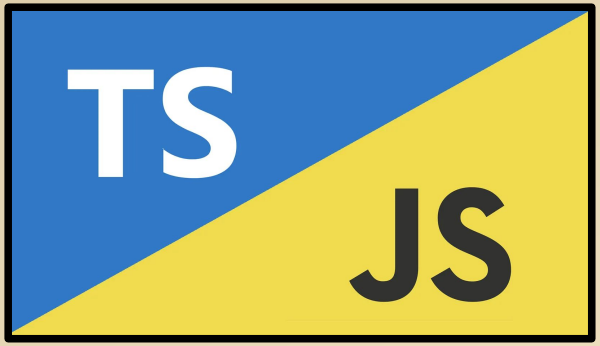
select:{}/omit:{}

→ omit:{} → Especifica quais propriedades serão excluídas no retorno.

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const seres = await prisma.ser.findMany({
  omit:{nome: true}})
 seres.forEach(s => {
  console.log(s.nome + " " + s.tipo)})
}
main()
```

undefined MORTAL	undefined DIVINDADE
undefined MORTAL	undefined DIVINDADE
undefined MORTAL	undefined DIVINDADE
undefined MORTAL	undefined DIVINDADE

- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

TS JS

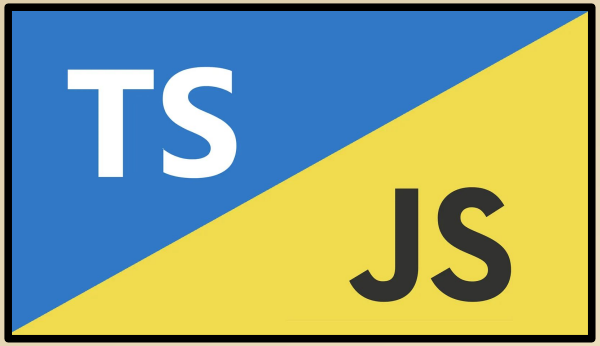
include:{} →quais relações são carregadas no retorno

include:{}

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const seres = await prisma.ser.findMany({
  include:{artefatos: true}})
 seres.forEach(s => {
  console.log(s.nome)
  s.artefatos.forEach(a =>{
   console.log("artefato:"+
a.nome+" "+a.destruido)})})
}
main()
```

```
Don Blake
Jane Foster
Bill Raio Beta
artefato:Rompe-Tormentas
false
Gorr
artefato:Necroespada false
Odin
Thor
artefato:mjolnir true
Freya
Loki
```

- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

Where

Where


```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const ativos = await prisma.artefato.findMany({
    where: {
      destruido: false,
    },
  })
  console.table(ativos)
}
main()
```

(index)	nome	objeto	material	destruido	nomeSer
0	'Rompe-Tormentas'	'martelo'	'uru'	false	'Bill Raio Beta'
1	'Necroespada'	'espada'	'simbiose All-Black'	false	'Gorr'

Where - AND implícito

Where

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const artefatos= await prisma.artefato.findMany({
    where: {
      destruido: false,
      objeto: 'martelo',
    },
  })
  console.table(artefatos)
}
main()
```

(index)	nome	objeto	material	destruido	nomeSer
0	'Rompe-Tormentas'	'martelo'	'uru'	false	'Bill Raio Beta'

Where - AND explícito

Where

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const artefatos = await prisma.artefato.findMany({
    where: {
      AND: [
        { destruido: false },
        { objeto: 'martelo' },
      ]
    }
  })
  console.table(artefatos)
}
main()
```

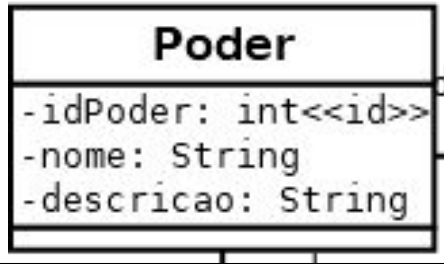
(index)	nome	objeto	material	destruido	nomeSer
0	'Rompe-Tormentas'	'martelo'	'uru'	false	'Bill Raio Beta'

Where OR

Where

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const poderes = await prisma.poder.findMany({
    select: {
      nome: true,
      descricao: true,
    },
    where: {
      OR: [
        { nome: 'invocação' },
        { nome: 'vôo' },
      ]
   }})
  console.table(poderes)
}
main()
```

(index)	nome	descricao
0	'invocação'	'retorno do objeto à mão'
1	'vôo'	'deslocar sobre o ar'



TS JS

Where Operadores

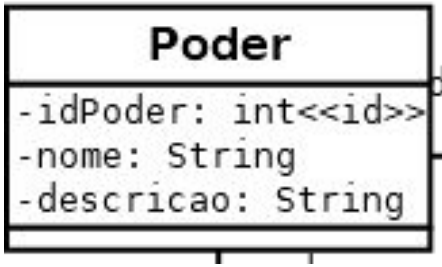
Operadores

- equals - igual - {id:{equals: 10}}
- not - diferente de - {nome:{not: 'Thor'}}
- in - dentro de lista - {id:{in:[1, 2, 3]}}
- notIn - fora da lista - {nome:{notIn: ['Loki']}}
- lt - menor que - {idade:{lt: 18}}
- lte - menor ou igual - {idade:{lte: 18}}
- gt - maior que - {idade:{gt: 18}}
- gte - maior ou igual - {idade:{gte: 18}}
- contains - contém substring - {nome:{contains:'raio'}}
- startsWith - começa com - {nome:{startsWith: 'Thor'}}
- endsWith - termina com - {nome:{endsWith: 'son'}}

Where

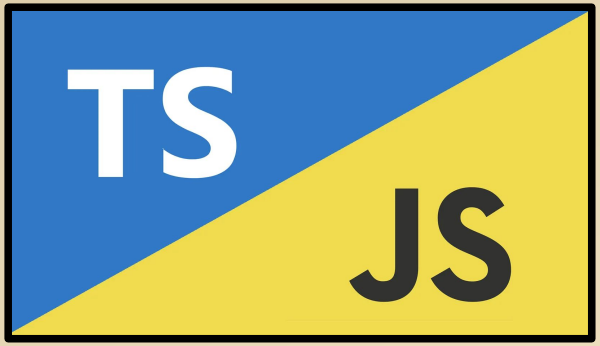
Where

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const poderes = await prisma.poder.findMany({
    where: {
      nome: {
        contains: 'ele',
        mode: 'insensitive',
      }
    }
  })
  console.table(poderes)
}
main()
```



(index)	idPoder	nome	descricao
0	4	'eletricidade instantânea'	'concentra raios elétricos'

- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}



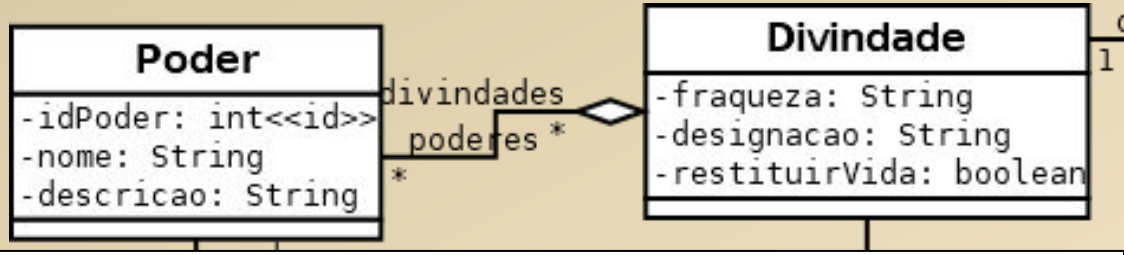
Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

TS JS

Group by



Group by

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
 const podDiv = await prisma.poderDivindade.groupBy({
  by: ['serNome'],
    _count: {
     poderId: true,
  }})
  console.table(podDiv)
}
main()
```

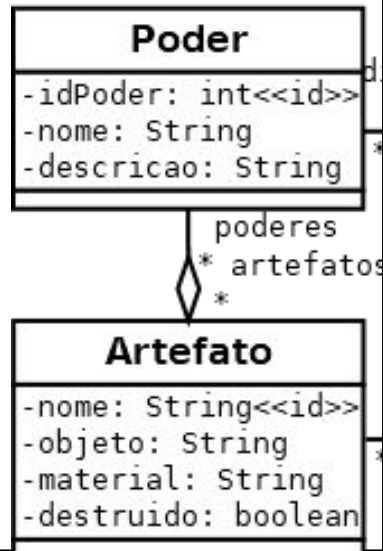
(index)	_count	serNome
0	{ poderId: 3 }	'Odin'
1	{ poderId: 4 }	'Thor'
2	{ poderId: 4 }	'Freya'
3	{ poderId: 4 }	'Loki'

Order by

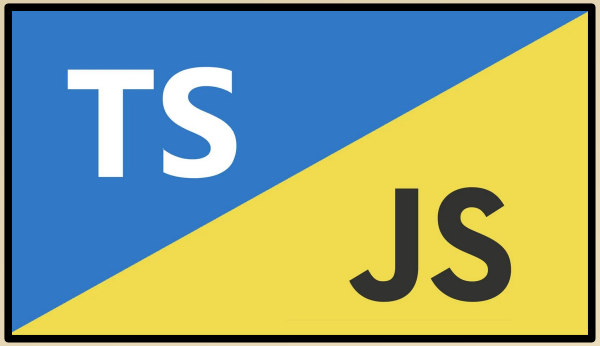
(index)	_count	artefatoNome
0	{ poderId: 4 }	'Necroespada'
1	{ poderId: 3 }	'mjolnir'
2	{ poderId: 3 }	'Rompe-Tormentas'

Order by

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const podDiv = await prisma.artefatoPoder.groupBy({
    by: ['artefatoNome'],
    _count: {
      poderId: true,
    },
    orderBy: {
      _count: {
        poderId: 'desc',
    }}})
  console.table(podDiv )
}
main()
```



- Introdução
- findMany/findFirst
- select:{}/omit:{}
- include: {}
- where:{}
- by: {}
- having: {}

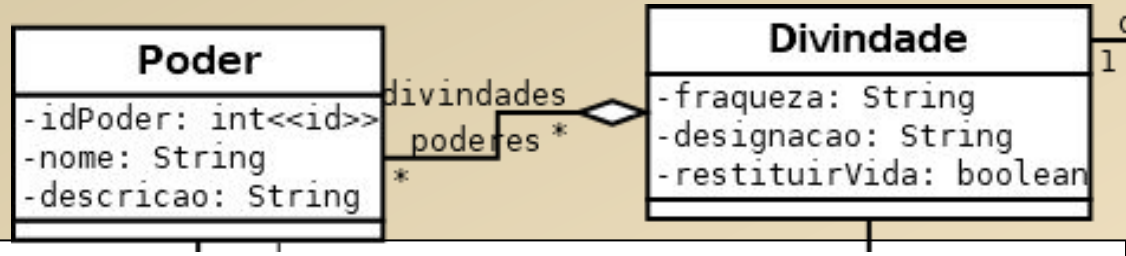


Prisma TS - Parte 2
Consulta

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano

Having



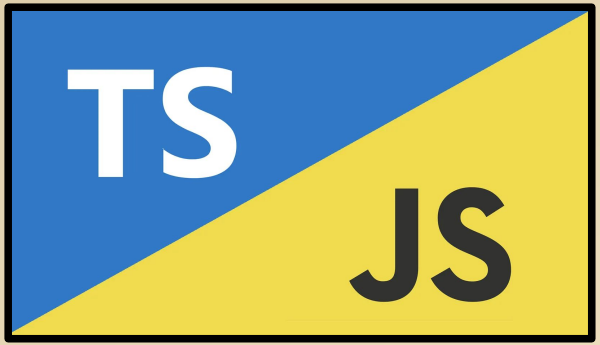

```
import { PrismaClient} from './generated/prisma'
const prisma = new PrismaClient()
async function main() {
  const podDiv = await prisma.poderDivindade.groupBy({
    by: ['serNome'],
    _count: {
      poderId: true,
    },
    having: {
      serNome: {
        _count: {
          gt: 3,
  }}}})
 console.table(podDiv)}
main()
```

Having

(index)	_count	serNome
0	{ poderId: 4 }	'Thor'
1	{ poderId: 4 }	'Freya'
2	{ poderId: 4 }	'Loki'

Prof. Enzo Seraphim

Profa. Bárbara Pimenta Caetano



Prisma TS - Parte 2
Consulta