TypeORM: World Cup

Apresentadores: Felipe Brenner, Felipe Ricardi e Lucas Oliveira da Silva

Disciplina: Sistemas de Gerência de Banco de Dados

Universidade do Vale do Rio dos Sinos - UNISINOS

Projeto

Base de dados da copa do mundo!

Seleções, estádios, grupos, partidas e mais.

TypeORM

- Object-Relational Mapping (ORM);
- NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript etc;
- Padrões: Active Record e Data Mapper;
- TypeScript ou JavaScript.

Data Source

- Conexão com o banco de dados (configurações e initial connection pool)

```
import "reflect-metadata";
import { DataSource } from "typeorm";
import * as entities from "./entity";

export const AppDataSource = new DataSource({
  type: "sqlite",
  database: "database.sqlite",
  synchronize: true,
  logging: false,
  entities,
  migrations: [],
  subscribers: [],
});
```

Entity

- É uma classe que faz o mapping com o banco de dados

```ts
import { Column, Entity, ManyToOne, PrimaryGeneratedColumn } from "typeorm";
import { Match } from "./Match";

@Entity()
export class Goal {
  @PrimaryGeneratedColumn("uuid")
  id: number;

  @Column()
  name: string;

  @Column()
  minute: number;

  @ManyToOne(() => Match)
  math: Match;
}
```

Relations

- Relações entre entidades (tabelas)
- one-to-one using @OneToOne
- many-to-one using @ManyToOne
- one-to-many using @OneToMany
- many-to-many using @ManyToMany

```
@Entity()
export class WorldCup {
  @PrimaryGeneratedColumn("uuid")
  id: number;

  @Column()
  name: string;

  @OneToMany(() => Country, (country) => country.worldCup)
  countries: Country[];

  @OneToMany(() => Group, (group) => group.worldCup)
  groups: Group[];

  @OneToMany(() => Stadium, (stadium) => stadium.worldCup)
  stadiums: Stadium[];

  @OneToMany(() => Round, (round) => round.worldCup)
  rounds: Round[];
}
```

QueryBuilder

- Permite a criação de SQL queries

```
// Example of QueryBuilder
const firstUser = await dataSource
    .getRepository(User)
    .createQueryBuilder("user")
    .where("user.id = :id", { id: 1 })
    .getOne()
```

```
/* Build the SQL Query: */
SELECT
    user.id as userId,
    user.firstName as userFirstName,
    user.lastName as userLastName
FROM users user
WHERE user.id = 1
```

```
// Returns an instance of User
User {
    id: 1,
    firstName: "Timber",
    lastName: "Saw"
}
```

Implementação

- populators para adicionar dados
- endpoints para recuperação, deleção, atualização e inserção de dados

Mapeamento

```
  @OneToOne(() => Federation)
  @JoinColumn()
  federation: Federation
}
```

```
@Entity()
export class WorldCup {
  @PrimaryGeneratedColumn("uuid")
  id: number;

  @Column()
  name: string;

  @OneToMany(() => Country, (country) => country.worldCup)
  countries: Country[];
```

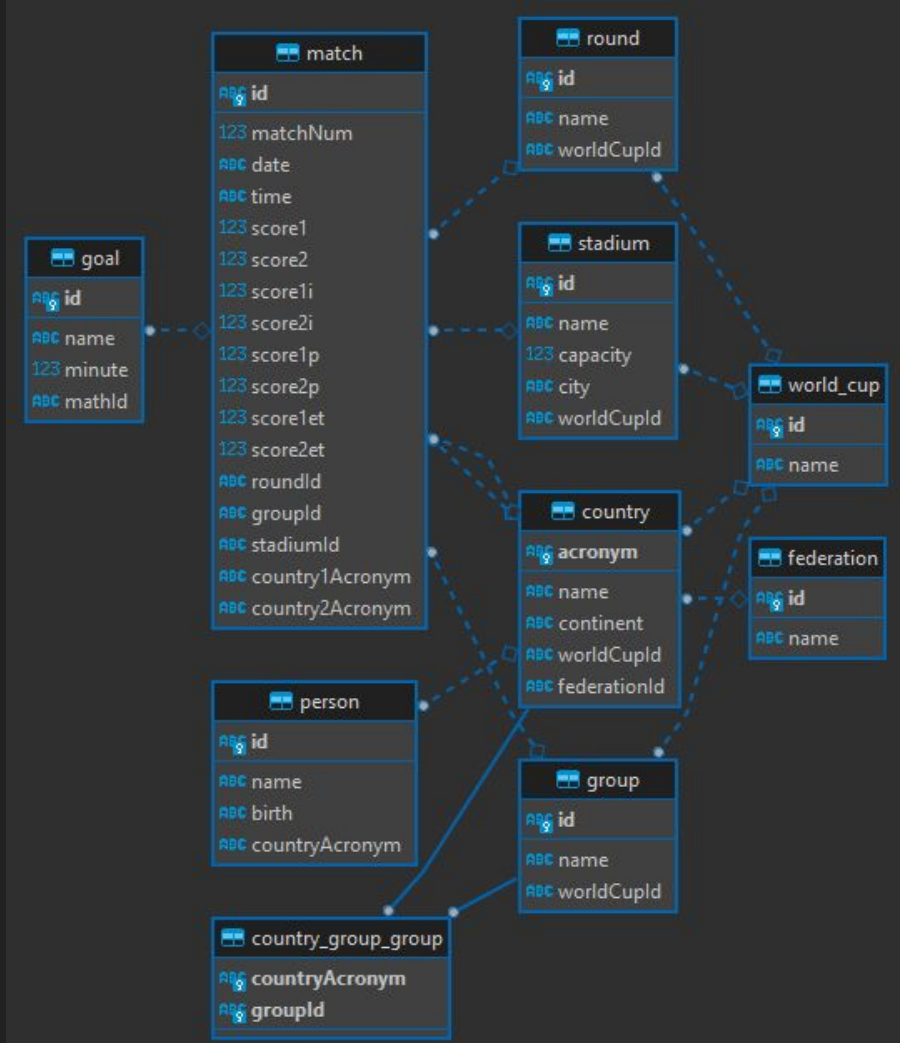
1:N

```
@Entity()
export class Group {
  @PrimaryGeneratedColumn("uuid")
  id: number;

  @Column()
  name: string;

  @ManyToOne(() => WorldCup, (worldCup) => worldCup.groups)
  worldCup: WorldCup;

  @ManyToMany(() => Country, (country) => country.group)
  countries: Country[];
}
```

N:N

Tabelas

- Copas do Mundo
- Países
- Estádios
- Grupos
- Partidas
- Rodadas
- Pessoas
- Gols
- Federações



Inserção de Registros

```
export default async function populateCountry() {

    const countryRepository = AppDataSource.getRepository(Country);

    for (var i = 0; i < teams.length; i++) {
        const existingCountry = await countryRepository.findOneBy({acronym: teams[i].code})
        if (!existingCountry) {
            console.log(`Inserting ${teams[i].code}...`)
            await countryRepository.save({
                acronym: teams[i].code,
                name: teams[i].name,
                continent: teams[i].continent
            })
        } else {
            console.log(`Country ${teams[i].code} already inserted`)
        }
    }

}
```

Consultas

```
  async cupParticipations(request: Request) {
    let country = await this.countryRepository.findOneBy({
      acronym: request.params.acronym,
    });
    if (!country) {
      throw new Error();
    }

    return this.worldCupRepository.find({
      where: {
        groups: {
          countries: country
        }
      },
      order: {
        name: "ASC"
      }
    })
  }
}
```

Demonstração

Conclusão

- Tecnologias:
  - TypeORM
  - VS Code
  - DBeaver
  - Insomnia
- Boa documentação
- Aplicação consistente
- Ótimos Resultados

Referências

- https://coday.netlify.app/conceitos-typeorm/
- https://typeorm.io/