L15

Alunos: Guilherme Alves, Heitor Freitas, João Pedro Oliveira

Raiz do projeto

index.js

Arquivo de entrada do projeto.

```
const db = require("./src/models");
const readline = require("readline-sync");

const hierarquia = async (idArbitro, idx = 1) => {
  try {
    // busca todos subordinados de um determinado arbitro, incluindo os dados da
pessoa
    const subordinados = await db.arbitro.findAll({
      include: [
        {
          model: db.pessoa,
        },
      ],
      where: {
        idcoo: idArbitro,
      },
      raw: true,
      nest: true,
    });

    for (const arbitro of subordinados) {
      // para cada subordinado, busca também os subordinados desse arbitro
      if (arbitro.idpes != idArbitro) {
        arbitro.subordinados = await hierarquia(arbitro.idpes, idx + 1);
        arbitro.nivel = idx;
      }
    }

    return subordinados;
  } catch (error) {
    console.error(error);
  }
};

const logHierarquia = async (arbitroCoordenador, idx = 0) => {
  try {
    // imprime no console os subordinados de cada arbitro
    let prefix = "";
    for (let i = 0; i < idx; i++) prefix += "\t";
```

```
    console.log(
      `| ${prefix} (${arbitroCoordenador.nivel},  ${arbitroCoordenador.idpes},
${arbitroCoordenador.pessoa.nomepes}, ${arbitroCoordenador.idcoo})`
    );

    for (const arbitro of arbitroCoordenador.subordinados) {
      // para cada subordinado, busca também os subordinados desse arbitro
      if (arbitro.idpes != arbitroCoordenador.idpes) {
        logHierarquia(arbitro, idx + 1);
      }
    }
  } catch (error) {
    console.error(error);
  }
};

const see = async () => {
  try {
    const idArbitro = Number(readline.question("Digite o id do arbitro: "));
    const arbitroRaiz = await db.arbitro.findByPk(idArbitro, {
      include: [
        {
          model: db.pessoa,
        },
      ],
      raw: true,
      nest: true,
    });
    arbitroRaiz.nivel = 0;
    arbitroRaiz.subordinados = await hierarquia(arbitroRaiz.idpes, 1);

    logHierarquia(arbitroRaiz);

    return;
  } catch (error) {
    console.error(error.message);
    return;
  }
};

const main = async () => {
  await see();

  process.exit(1);
};

main();
```

.sequelizerc

> Arquivo de configuração do ORM Sequelize.

```
const path = require('path')

module.exports = {
  config: path.resolve('src', 'config', 'database.js'),
  'models-path': path.resolve('src', 'models'),
  'seeders-path': path.resolve('src', 'database', 'seeders'),
  'migrations-path': path.resolve('src', 'database', 'migrations'),
}
```

./src/config

database.js

> Arquivo de conexão do banco de dados.

```
// configurações de acesso ao banco de dados
module.exports = {
  host: "200.131.206.13",
  username: "heitor_ff",
  password: "SENHA",//aqui vai a senha do banco
  database: "heitor_ff",
  dialect: "postgres",
  storage: "./__tests__/database.sqlite",
  dialectOptions: null,
  logging: false,
  pool: {
    max: 10,
    min: 1,
    acquire: 25000,
    idle: 50000,
  },
  define: {
    timestamps: false,
    underscored: false,
    underscoredAll: false,
    freezeTableName: true,
  },
};
```

./src/models

arbitro.js

> Arquivo de modelo do banco de dados.

```
const Sequelize = require("sequelize");
module.exports = function (sequelize, DataTypes) {
  const arbitro = sequelize.define(
```

```
    "arbitro",
    {
      idpes: {
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true,
        references: {
          model: "funcionario",
          key: "idpes",
        },
      },
      idcoo: {
        type: DataTypes.INTEGER,
        allowNull: false,
        references: {
          model: "arbitro",
          key: "idpes",
        },
      },
    },
    {
      sequelize,
      tableName: "arbitro",
      schema: "see",
      timestamps: false,
      indexes: [
        {
          name: "arbitro_pk",
          unique: true,
          fields: [{ name: "idpes" }],
        },
      ],
    }
  );

  arbitro.associate = function (models) {
    arbitro.belongsTo(models.funcionario, {
      foreignKey: "idpes",
    });

    arbitro.belongsTo(models.pessoa, {
      foreignKey: "idpes",
    });
  };

  return arbitro;
};
```

funcionario.js

> Arquivo de modelo do banco de dados.

```
const Sequelize = require("sequelize");

module.exports = function (sequelize, DataTypes) {
  const funcionario = sequelize.define(
    "funcionario",
    {
      idpes: {
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true,
        references: {
          model: "pessoa",
          key: "idpes",
        },
      },
    },
    {
      sequelize,
      tableName: "funcionario",
      schema: "see",
      timestamps: false,
      indexes: [
        {
          name: "funcionario_pk",
          unique: true,
          fields: [{ name: "idpes" }],
        },
      ],
    }
  );

  funcionario.associate = function (models) {
    funcionario.belongsTo(models.pessoa, {
      foreignKey: "idpes",
    });
  };

  return funcionario;
};
```

pessoa.js

> Arquivo de modelo do banco de dados.

```
const Sequelize = require("sequelize");
module.exports = function (sequelize, DataTypes) {
  const pessoa = sequelize.define(
    "pessoa",
    {
      idpes: {
        autoIncrement: true,
```

```
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true,
      },
      cpf: {
        type: DataTypes.CHAR(11),
        allowNull: false,
        unique: "cpf_pes_sk",
      },
      nomepes: {
        type: DataTypes.STRING(40),
        allowNull: true,
      },
      /*  titulopes: {
        type: DataTypes.CHAR(1),
        allowNull: true,
      }, */
    },
    {
      sequelize,
      tableName: "pessoa",
      schema: "see",
      timestamps: false,
      indexes: [
        {
          name: "cpf_pes_sk",
          unique: true,
          fields: [{ name: "cpf" }],
        },
        {
          name: "pessoa_pk",
          unique: true,
          fields: [{ name: "idpes" }],
        },
      ],
    }
  );

  pessoa.associate = function (models) {
    pessoa.hasOne(models.funcionario, {
      foreignKey: "idpes",
    });
  };

  return pessoa;
};
```

index.js

> Arquivo que une todos os models para que sejam utilizados em conjunto, e não cada model ser um novo TAD.

```
"use strict";

const fs = require("fs");
const path = require("path");
const Sequelize = require("sequelize");
const basename = path.basename(__filename);
const config = require("../config/database");
const db = {};

const sequelize = new Sequelize(
  config.database,
  config.username,
  config.password,
  config
);

fs.readdirSync(__dirname)
  .filter((file) => {
    return (
      file.indexOf(".") !== 0 && file !== basename && file.slice(-3) === ".js"
    );
  })
  .forEach((file) => {
    const model = require(path.join(__dirname, file))(
      sequelize,
      Sequelize.DataTypes
    );
    db[model.name] = model;
  });

Object.keys(db).forEach((modelName) => {
  if (db[modelName].associate) {
    db[modelName].associate(db);
  }
});

db.Sequelize = Sequelize;
db.sequelize = sequelize;

module.exports = db;
```