Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

ПИиКТ, кафедра ИПМ

Курсовая работа по дисциплине

«Программирование интернет-приложений»

Этап 2

Выполнили:

Каменский Валерий

Чудаков Матвей

P3219

Преподаватели:

Цопа Евгений Алексеевич

Харитонова Анастасия Евгеньевна

Санкт-Петербург 2017

**Исходный код**

/models/index.js

**const** fs = *require*(**'fs'**);  
**const** path = *require*(**'path'**);  
**const** Sequelize = *require*(**'sequelize'**);  
**const** basename = path.*basename*(module.**filename**);  
**const** env = ***process***.env.NODE\_ENV || **'development'**;  
**const** config = *require*(**`**${\_\_dirname}**/../config/config.json`**)[env];  
**const** db = {};  
  
**let** sequelize;  
**if** (config.use\_env\_variable) {  
 sequelize = **new** Sequelize(***process***.env[config.use\_env\_variable]);  
} **else** {  
 sequelize = **new** Sequelize(  
 config.**database**, config.**username**, config.**password**, config  
 );  
}  
  
fs  
 .*readdirSync*(\_\_dirname)  
 .filter(file =>  
 (file.indexOf(**'.'**) !== 0) &&  
 (file !== basename) &&  
 (file.slice(-3) === **'.js'**))  
 .forEach(file => {  
 **const** model = sequelize.import(path.*join*(\_\_dirname, file));  
 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;

/models/players

module.**exports** = (sequelize, DataTypes) => {  
 **const** players = sequelize.define(**'players'**, {  
 **name**: {  
 **type**: DataTypes.**STRING**,  
 **allowNull**: **false**,  
 },  
 **surname**: {  
 **type**: DataTypes.**STRING**,  
 **allowNull**: **false**,  
 }  
 });  
 players.associate = (models) => {  
 players.hasMany(models.contracts, {  
 **foreignKey**: **'playerID'**,  
 **as**: **'contracts'**,  
 });  
 players.hasMany(models.playerStatistics, {  
 **foreignKey**: **'playerID'**,  
 **as**: **'playerStatistics'**,  
 });  
 };  
 **return** players;  
};

/models/contracrs

module.**exports** = (sequelize, DataTypes) => {  
 **const** contracts = sequelize.define(**'contracts'**, {  
 **description**: {  
 **type**: DataTypes.**STRING**,  
 **allowNull**: **true**,  
 }  
 });  
 contracts.associate = (models) => {  
 contracts.**belongsTo**(models.players, {  
 **foreignKey**: **'playerID'**,  
 **onDelete**: **'CASCADE'**,  
 });  
 contracts.**belongsTo**(models.teams, {  
 **foreignKey**: **'teamID'**,  
 **onDelete**: **'CASCADE'**,  
 });  
 };  
 **return** contracts;  
};

/controller/playerController

**const** Players = *require*(**'../models'**).players;  
**const** Contracts = *require*(**"../models"**).contracts;  
**const** Teams = *require*(**"../models"**).teams;  
  
module.**exports** = {  
 create(req, res){  
 **return** Players  
 .create({  
 **name**: req.**body**.**name**,  
 **surname**: req.**body**.**surname**,  
 })  
 .then((player) => res.status(201).send(player))  
 .catch((error) => res.status(400).send(error));  
 },  
 list(req, res) {  
 **return** Players  
 .all()  
 .then(players => res.status(200).send(players))  
 .catch(error => res.status(400).send(error));  
 },  
 listLimit(req, res){  
 **return** Players  
 .*findAll*({ **limit**: req.**params**.**limit** })  
 .then(players => res.status(200).send(players))  
 .catch(error => res.status(400).send(error));  
 },  
 listTeam(req, res){  
 **return** Contracts  
 .*findAll*({  
 **include**:[{  
 **model**: Players, **as**: **'player'** },{  
 **model**: Teams, **as**: **'team'**,  
 **where**: { **name**: req.**params**.**team** }  
 }],  
 })  
 .then(contracts => res.status(200).send(contracts))  
 .catch(error => res.status(400).send(error));  
 },  
 update(req, res) {  
 **return** Players  
 .*findById*(req.**params**.**id**)  
 .then(player => {  
 **if** (!player) {  
 **return** res.status(404).send({  
 **message**: **'Player Not Found'**,  
 });  
 }  
 **return** player  
 .update({  
 **name**: req.**body**.**name** || player.**name**,  
 })  
 .then(() => res.status(200).send(player))  
 .catch((error) => res.status(400).send(error));  
 })  
 .catch((error) => res.status(400).send(error));  
 },  
  
 **delete**(req, res){  
 **return** Players  
 .*findById*(req.**params**.**id**)  
 .then(player => {  
 **if** (!player) {  
 **return** res.status(400).send({  
 **message**: **'Player Not Found'**,  
 });  
 }  
 **return** player  
 .destroy()  
 .then(() => res.status(204).send())  
 .catch(error => res.status(400).send(error));  
 })  
 .catch(error => res.status(400).send(error));  
 }  
};

**DDL**

