**Лабораторная работа №4,5**

**SEQUALIZE NODE JS**

**Цель**: изучить генерацию моделей, создание связей 1:1, 1:N, N:M. CRUD - операции.

**Генерация модели:**

**В терминале:**

Установим пакет npm и драйвер к mysql.

npm install --save sequelize

$ npm install --save mysql2

Создали пустой проект, используя команду:

npx sequelize-cli init

Это создаст следующие папки:

* config;
* models;
* migrations;
* seeders.

В файле congig.json выставим соответствующие настройки БД (рис. 1)

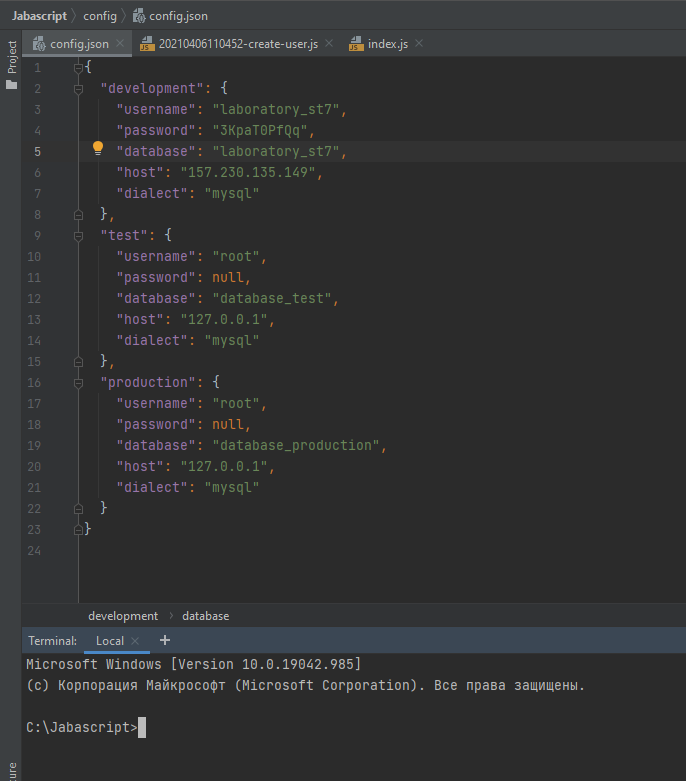


Рисунок – 1 Параметры БД

Создаем миграции и модели

Для таблицы Users

npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string

Для таблицы Comment

npx sequelize-cli model:generate --name Comment --attributes id\_user:string comment:string

Для таблицы Login

npx sequelize-cli model:generate --name Login –attributes login:string

Для таблицы Courses

npx sequelize-cli model:generate --name Course --attributes name:string

Для таблицы Enties

npx sequelize-cli model:generate --name Enty --attributes row1:string row2:string row3:string

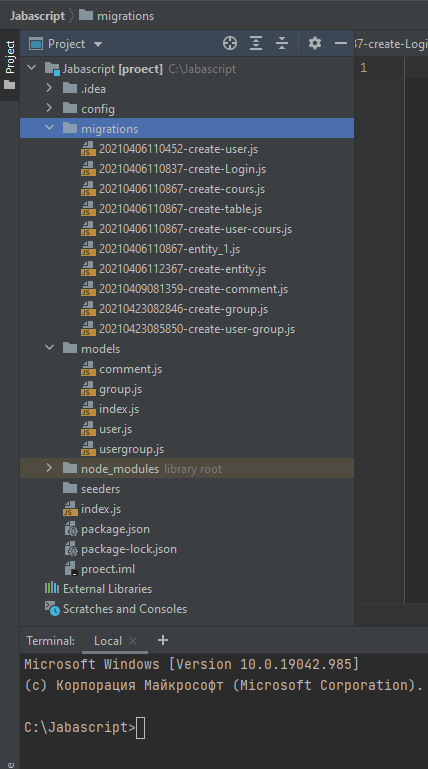


Рисунок – 2 Созданные миграции и модели

Теперь необходимо добавить созданные модели в базу данных. Для добавления таблицы в БД выполнили команду:

npx sequelize-cli db:migrate

**Один-к-Одному**

Листинг 1. Работа со связью 1:1

  //ONE - TO - ONE

    // добавляем пользователя

    User.create({

        firstName: "Artem",

        lastName: "Reshetilov",

        email: "reshetilov.a.n@srspu.ru"

    }).then(user=>{

    // Добавляем логин

        Login.create({name:"Murrxie"}).then(login=>{

            // устнавливаем логин для пользователя

            user.setLogin(login).catch(err=>console.log(err));

        });

    }).catch(err=>console.log(err));

    //

    const jane = await User.create({

        firstName: "Jane",

        lastName: "Smit",

        email: "smit.j@srspu.ru"

    });

    const user = await User.findByPk(32);

    const user\_login = await Login.create({id\_user:user.id, login:"cute\_user"});

//Удаление записи

    User.destroy({

        where: {

            id: 29

        }

    });

Добавили пользователей в таблицу Users

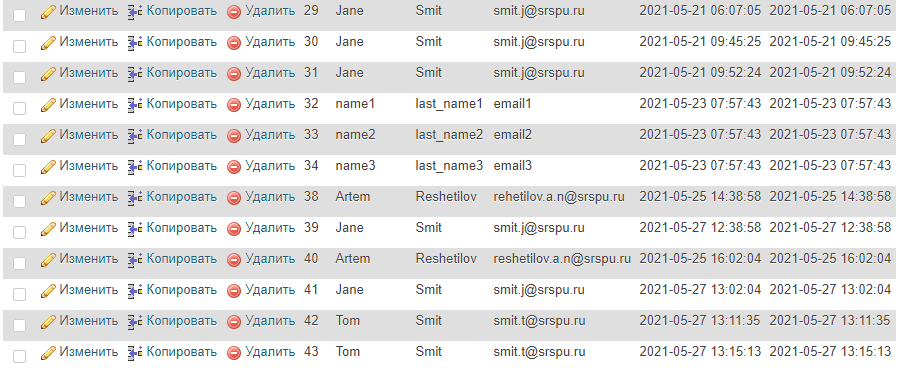


Рисунок – 3

Добавили логины для этих пользователей в таблицу Logins

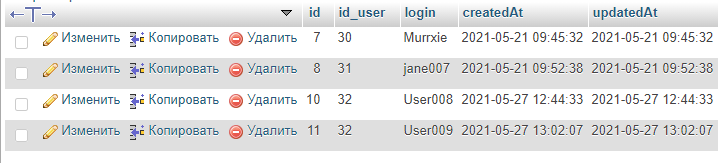


Рисунок – 4

Выполнили удаление записи пользователя с id = 29.



Рисунок – 5 Наличие записи



Рисунок – 6 Удаление записи

**Один-ко-Многим**

Листинг 1. Работа со связью 1:N

    //ONE - TO - MANY

    const artem = await User.findByPk(23);

    console.log("Artem`s auto-generated ID:", artem.id);

    //Добавить комментарий

    const comment = await Comment.create({user\_id:artem.id, comment: "Комментарий Artema" });

    const comment\_1 = await Comment.create({user\_id:artem.id, comment: "Привет });

    const comment\_2 = await Comment.create({user\_id:artem.id, comment: "artem`s comment" });

    const comment\_delete = await Comment.create({user\_id:24, comment:"удалить"});

    //Удалить комментарий

    await Comment.destroy({

        where:{

            id:9,

            user\_id:jane.id

            //comment: "удалить"

        }

    });

    //Найти комментарии

        User.findAll({

        include: [{

          model: Comment,

          as: 'Comments'

        }],

        attributes:{ exclude: ['createdAt','updatedAt'] }

      }).then((users)=>{

        console.log(users)

        });;



Рисунок – 7 Создание комментариев

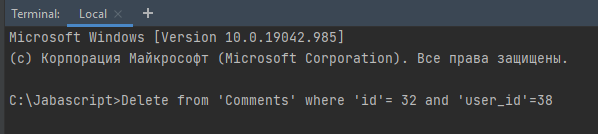


Рисунок – 8 Удаление комментария

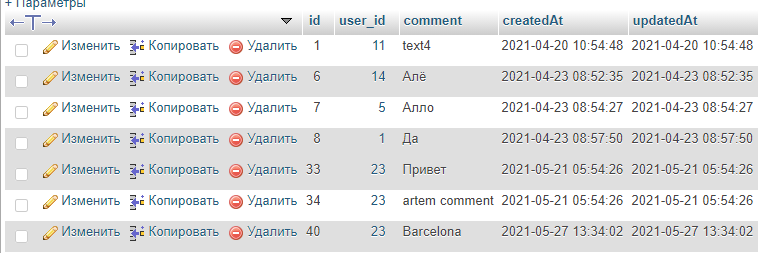


Рисунок – 9 Удаление комментария

**Многие-ко-Многим**

Устанавливаем внешние ключи в таблицах User и Courses.

В user.js:

User.belongsToMany(models.Course,{through:'enty'});

В course.js:

Course.belongsToMany(models.User,{through:'enty'});

Создаем записи в этих таблицах и связываем таблицы и добавляем запись типа связи (Листинг)

Листинг index.js

User.create({

        first\_name:"Имя1",

        last\_name:"Фамилия1",

        email:"email2"

    });

    User.create({

        first\_name:"Имя2",

        last\_name:"Фамилия2",

        email:"email2"

    });

    User.create({

        firstName:"Имя3",

        lastName:"Фамилия3",

        email:"email3"

    });

    Course.create({

        name:"name1"

    });

    Course.create({

        name:"name2"

    });

    Course.create({

        name:"name3"

    });

     //нашли пользователя

    const user = await User.findByPk(23);

//нашли курс

    const course = await Course.findOne({

        where:{name:"name2"}

    });

//добавили запись в таблицу связку

    await user.addCourse(course, { through: { row\_1: "false", row\_2:"true", row\_3:"Лабораторная" } });

        //вывели результат работы

User.findOne({where: {id:23}})

        .then(user=>{

            if(!user) return;

            user.getCourses().then(courses=>{

                for(course of courses){

                    console.log(user.lastName + " " + course.name);

                }

            });

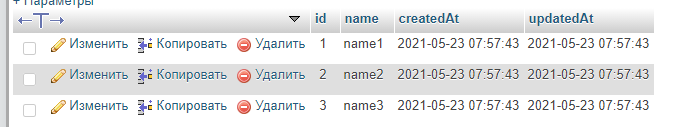


Рисунок – 10

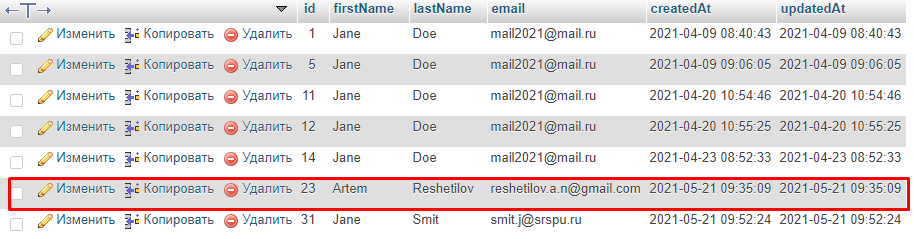


Рисунок – 11

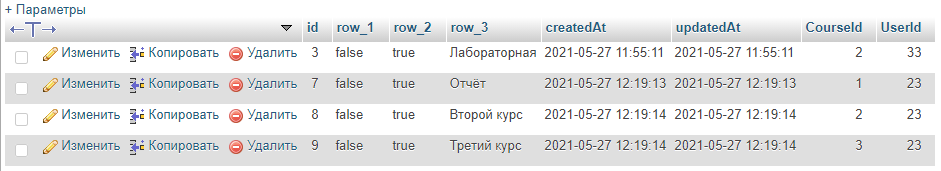


Рисунок – 12

Вывод: изучил основные принципы и возможности работы с SEQUALIZE NODE JS.

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| Решетилов А.Н. | Хорошко М.Б. |