МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Базы данных»

Тема: Реализация базы данных с использованием ORM

Студентка гр. 1303 _____ Андреева Е.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2023

Цель работы.

Создание базы данных с использованием Object-Relational Mapping (ORM).

Текст задания

Вариант 1

- Описать в виде моделей Sequelize таблицы из 1-й лабораторной работы
- Написать скрипт заполнения тестовыми данными: 5-10 строк на каждую таблицу, обязательно наличие связи между ними, данные приближены к реальности.
- Написать запросы к БД, отвечающие на вопросы из 1-й лабораторной работы с использованием ORM. Вывести результаты в консоль (или иной человек-читабельный вывод)
- Запушить в репозиторий исходный код проекта, соблюсти .gitignore, убрать исходную базу из проекта (или иные нагенерированные данные бд если они есть).
- Описать процесс запуска: команды, зависимости
- В отчёте описать цель, текст задания в соответствии с вариантом, выбранную ORM, инструкцию по запуску, скриншоты (код) моделей ORM, скриншоты на каждый запрос (или группу запросов) на изменение/таблицы с выводом результатов (ответ), ссылку на PR в приложении, вывод

Выполнение работы

Для установки необходимых библиотек использовались следующие команды:

```
npm install sequelize
npm install pg
```

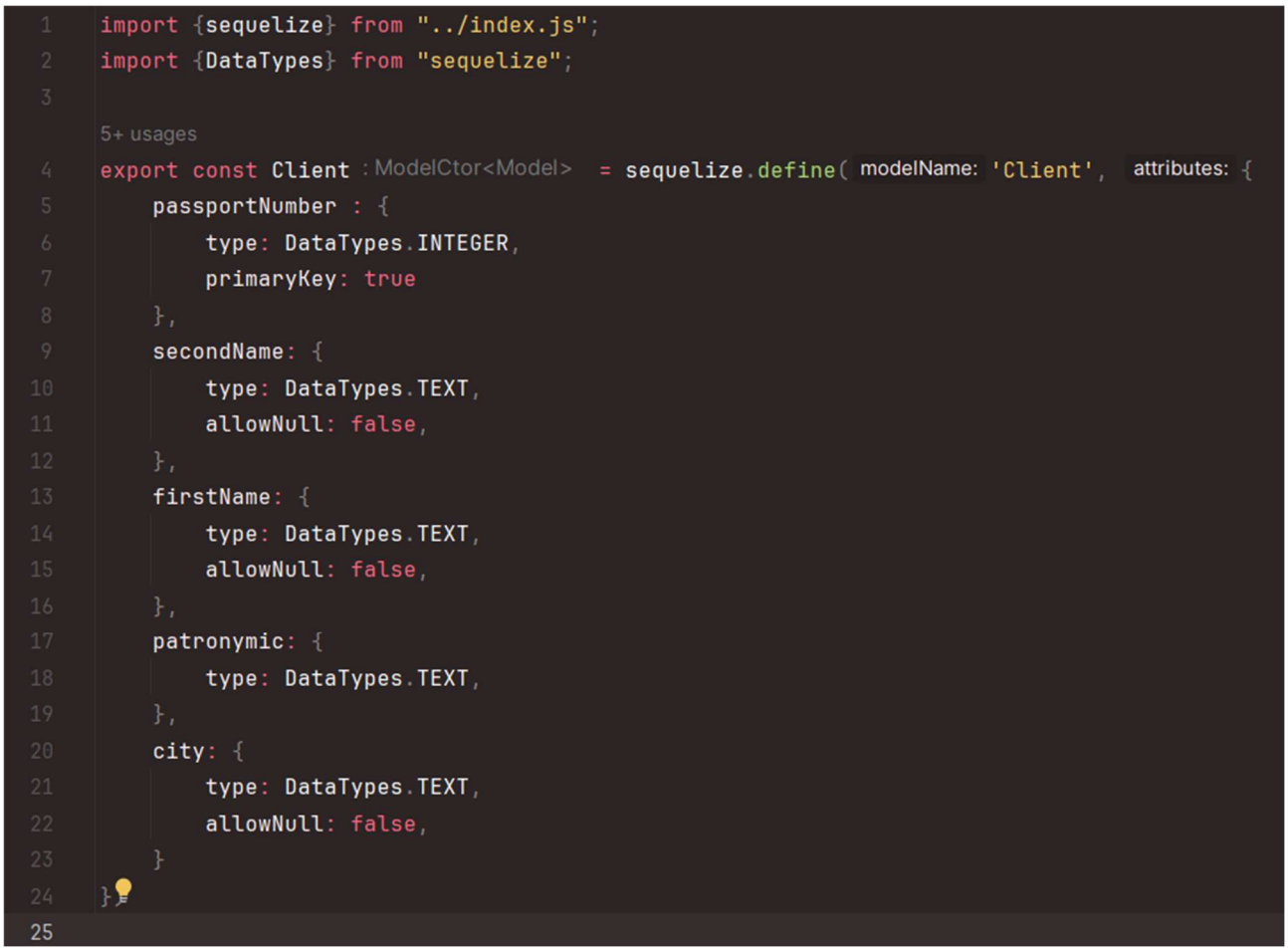
В файле index.js происходит настройка соединения с базой данных:

```
export const sequelize = new Sequelize('hotel', 'postgres', '1', {
    host: 'localhost',
    dialect: 'postgres',
});
```

Здесь создается объект sequelize, который представляет собой соединение с базой данных. В конструкторе Sequelize передаются следующие параметры:

- hotel: Название базы данных.
- postgres: Имя пользователя базы данных.
- 1: Пароль пользователя базы данных.
- { host: 'localhost', dialect: 'postgres' }: Дополнительные опции, указывающие хост (localhost) и используемый диалект (PostgreSQL).

Реализованные модели:

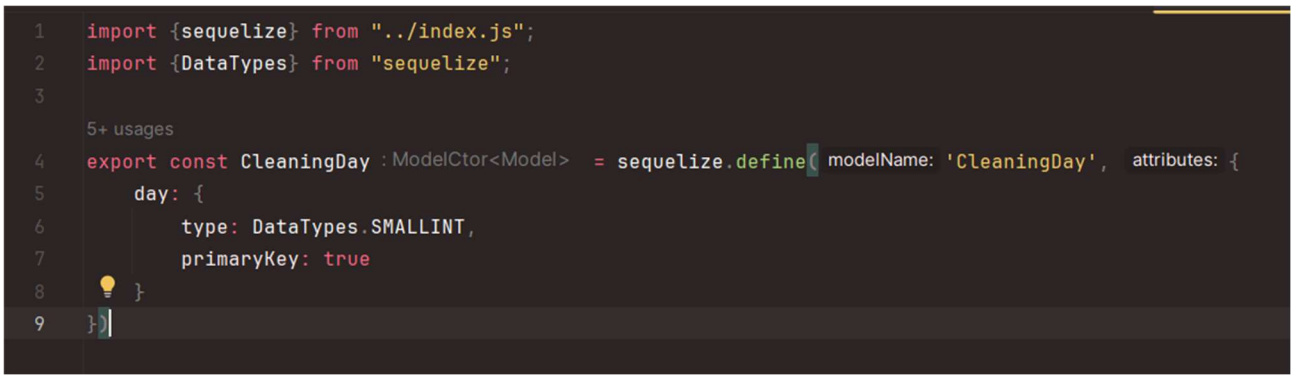


```
import {sequelize} from "../index.js";
import {DataTypes} from "sequelize";

5+ usages
export const Client : ModelCtor<Model>  = sequelize.define( modelName: 'Client',  attributes: {
    passportNumber : {
        type: DataTypes.INTEGER,
        primaryKey: true
    },
    secondName: {
        type: DataTypes.TEXT,
        allowNull: false,
    },
    firstName: {
        type: DataTypes.TEXT,
        allowNull: false,
    },
    patronymic: {
        type: DataTypes.TEXT,
    },
    city: {
        type: DataTypes.TEXT,
        allowNull: false,
    }
});

```

Рисунок 1. – Структура созданной БД.



```
import {sequelize} from "../index.js";
import {DataTypes} from "sequelize";

5+ usages
export const CleaningDay : ModelCtor<Model>  = sequelize.define( modelName: 'CleaningDay',  attributes: {
    day: {
        type: DataTypes.SMALLINT,
        primaryKey: true
    }
});
```

Рисунок 2. – Структура созданной БД.

```
5+ usages
export const Room : ModelCtor<Model>  = sequelize.define( modelName: 'Room',  attributes: {
    roomNumber: {
        type: DataTypes.INTEGER,
        primaryKey: true
    },
    type: {
        type: DataTypes.TEXT,
        allowNull: false
    },
    price: {
        type: DataTypes.INTEGER,
        allowNull: false
    },
    phoneNumber: {
        type: DataTypes.INTEGER
    },
    floorNumber: {
        type: DataTypes.SMALLINT,
        allowNull: false,
        references: {
            model: Floor,
            key: 'floorNumber'
        },
        onDelete: 'CASCADE'
    },
})

```

Рисунок 3. – Структура созданной БД.

```
import {sequelize} from "../index.js";
import {DataTypes} from "sequelize";

5+ usages
export const Floor : ModelCtor<Model>  = sequelize.define( modelName: 'Floor',  attributes: {
    floorNumber: {
        type: DataTypes.SMALLINT,
        primaryKey: true
    }
})

```

Рисунок 4. – Структура созданной БД.

```
import {sequelize} from "../index.js";
import {DataTypes} from "sequelize";

5+ usages
export const Employee : ModelCtor<Model>  = sequelize.define( modelName: 'Employee',  attributes: {
    employeeId: {
        type: DataTypes.INTEGER,
        autoIncrement: true,
        primaryKey: true
    },
    secondName: {
        type: DataTypes.TEXT,
        allowNull: false
    },
    firstName: {
        type: DataTypes.TEXT,
        allowNull: false
    },
    patronymic: {
        type: DataTypes.TEXT
    }
})

```

Рисунок 5. – Структура созданной БД.

```
export const ClientRoom : ModelCtor<Model>  = sequelize.define( modelName: 'ClientRoom',  attributes: {
    passportNumber: {
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true,
        references: {
            model: Client,
            key: 'passportNumber'
        }
    },
    roomNumber: {
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true,
        references: {
            model: Room,
            key: 'roomNumber'
        }
    },
    checkInDate: {
        type: DataTypes.DATE,
        allowNull: false
    },
    checkOutDate: {
        type: DataTypes.DATE,
        allowNull: false
    },
})

```

Рисунок 6. – Структура созданной БД.

```
export const EmployeeCleaning : ModelCtor<Model> = sequelize.define( modelName: 'EmployeeCleaning', attributes: {
    employeeCleaningId: {
        type: DataTypes.INTEGER,
        autoIncrement: true,
        primaryKey: true
    },
    floorNumber: {
        type: DataTypes.INTEGER,
        allowNull: false,
        references: {
            model: Floor,
            key: 'floorNumber'
        },
        onDelete: 'CASCADE',
    },
    day: {
        type: DataTypes.INTEGER,
        references: {
            model: CleaningDay,
            key: 'day'
        },
        onDelete: 'SET NULL',
    },
    employeeId: {
        type: DataTypes.INTEGER,
        references: {
            model: Employee,
            key: 'employeeId'
        },
        onDelete: 'SET NULL',
    },
```

Рисунок 7. – Структура созданной БД.

Выполним запросы к БД предложенные вариантом:

```
async function doTasks() : Promise<void> {
    await Client.findAll( options: {
        attributes: ['secondName', 'firstName', 'patronymic'],
        include: {
            model: ClientRoom,
            required: true,
            where: {
                roomNumber: 555
            },
            attributes: []
        }
    }).then((res : (Model<...>)[] ) : void => {
        console.log("\n\nКлиент, проживающий в заданном номере:\n", JSON.stringify(res, replacer: null, space: 2), "\n"
    })
```

Рисунок 9. – Запрос 1.

```
Клиент, проживающий в заданном номере (555):
 [
  {
    "secondName": "Афанасьева",
    "firstName": "Анна",
    "patronymic": "Васильевна"
  }
]
```

Рисунок 10. – Результат запроса 1.

```
await Client.findAll( options: {
    attributes: ['secondName', 'firstName', 'patronymic'],
    where: {
        city: 'Брест',
    }
}).then((res : (Model<...>)[] ) : void  => {
    console.log("\n\nКлиент, прибывший из заданного города:\n", JSON.stringify(res,  replacer: null,  space: 2), "\n"
})
```

Рисунок 11 – Запрос 2.

```
Клиент, прибывший из заданного города:
 [
  {
    "secondName": "Андреева",
    "firstName": "Елизавета",
    "patronymic": "Алексеевна"
  },
  {
    "secondName": "Бутыло",
    "firstName": "Егор",
    "patronymic": "Алексеевич"
  }
]
```

Рисунок 12 – Результат запроса 2.

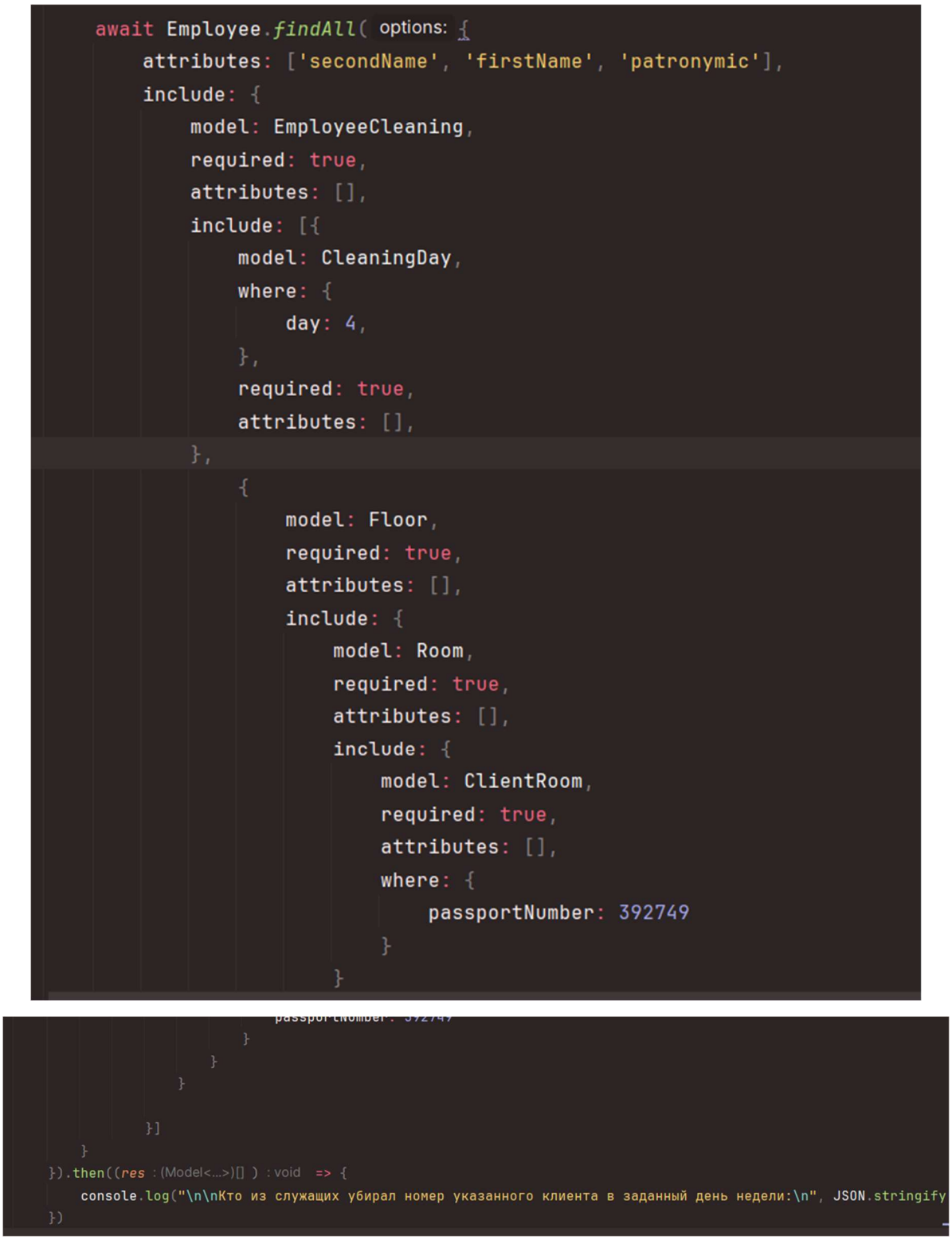


Рисунок 13 – Запрос 3.

```
Кто из служащих убирал номер указанного клиента в заданный день недели:
 [
  {
    "secondName": "Иванов",
    "firstName": "Олег",
    "patronymic": "Геннадьевич"
  }
]
```

Рисунок 14 – Результат запроса 3.

```
    const today :Date  = new Date();

    await Room.count({
        group: '',
        include: {
            model: ClientRoom,
            attributes: [],
            required: true,
            where: {
                [Op.or]: [{
                    checkOutDate: {
                        [Op.lte]: today
                    }
                },
                    {
                        checkInDate: {
                            [Op.gte]: today
                        }
                    }]
            }
        },
    }).then((res :GroupedCountResultItem[] ) :void  => {
        console.log("\n\nЕсть ли в гостинице свободные места и свободные номера и, если есть, то сколько:\n", JSON.s
    })
}
```

Рисунок 15 – Запрос 4.

```
Есть ли в гостинице свободные места и свободные номера и, если есть, то сколько:
 4
```

Рисунок 16 – Результат запроса 4.

Выводы.

В данной лабораторной работе освоена работа с ORM для Node.js - Sequelize.

ПРИЛОЖЕНИЕ А

Pull request: https://github.com/moevm/sql-2023-1303/pull/41