

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

Студент гр. 1303

Бутыло Е.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2023

Цель работы.

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

Текст задания

Вариант 3

- 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-читаемый вывод)
- Запустить в репозиторий исходный код проекта, соблюсти .gitignore, убрать исходную базу из проекта (или иные сгенерированные 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 использовались следующие команды:

```
npm install sequelize
npm install pg pg-hstore
```

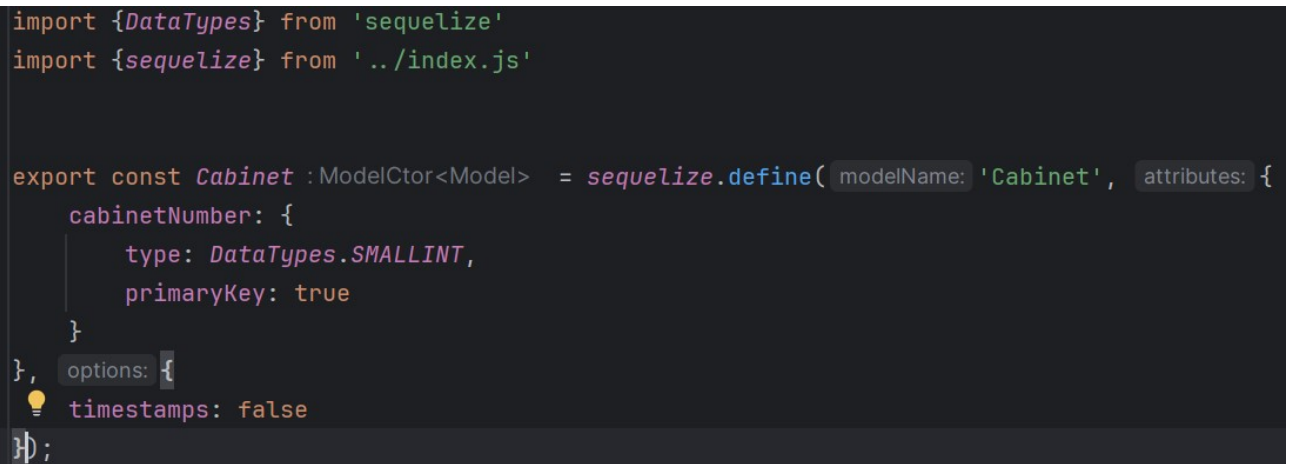
Подключение к базе данных:

```
const dbName = 'db-lab3'
export const sequelize = new Sequelize(dbName, 'postgres', '1', {
  host: 'localhost',
  dialect: 'postgres'
});

try {
  await sequelize.authenticate();
  console.log('Connection has been established successfully.');
```

```
} catch (error) {
  console.error('Unable to connect to the database:', error);
}
```

Реализованные модели:

A screenshot of a code editor showing the definition of a Sequelize model named 'Cabinet'. The code is written in JavaScript and uses TypeScript-style syntax. It imports 'DataTypes' from 'sequelize' and 'sequelize' from '../index.js'. The 'Cabinet' model is defined with a 'cabinetNumber' attribute of type 'SMALLINT' and 'primaryKey: true'. The 'options' object is set to 'timestamps: false'.

```
import {DataTypes} from 'sequelize'
import {sequelize} from '../index.js'

export const Cabinet : ModelCtor<Model> = sequelize.define( modelName: 'Cabinet', attributes: {
  cabinetNumber: {
    type: DataTypes.SMALLINT,
    primaryKey: true
  }
}, options: {
  timestamps: false
});
```

Рисунок 1. – Структура созданной БД.

```

import {DataTypes} from 'sequelize'
import {sequelize} from '../index.js'

export const Class : ModelCtor<Model> = sequelize.define( modelName: 'Class', attributes: {
  className: {
    type: DataTypes.TEXT,
    primaryKey: true
  }
}, options: {
  timestamps: false
});

```

Рисунок 2. – Структура созданной БД.

```

export const Grade : ModelCtor<Model> = sequelize.define( modelName: 'Grade', attributes: {
  studentId: {
    type: DataTypes.SMALLINT,
    primaryKey: true,
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE',
    references: {
      model: Student,
      key: 'id'
    }
  },
  subjectName: {
    type: DataTypes.TEXT,
    primaryKey: true,
    onDelete: 'CASCADE',
    references: {
      model: Subject,
      key: 'subjectName'
    }
  },
  grade: {
    type: DataTypes.SMALLINT,
    allowNull: true
  }
}, options: {
  timestamps: false
});

```

Рисунок 3. – Структура созданной БД.

```

export const Schedule : ModelCtor<Model> = sequelize.define( modelName: 'Schedule', attributes: {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  orderNumber: {
    type: DataTypes.SMALLINT,
    allowNull: false
  },
  day: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  className: {type: DataTypes.TEXT...},
  subjectName: {
    type: DataTypes.TEXT,
    onDelete: 'CASCADE',
    references: {model: Subject...}
  },
  teacherId: {type: DataTypes.SMALLINT...},
  cabinetNumber: {type: DataTypes.SMALLINT...}
}, options: {
  timestamps: false
});

```

Рисунок 4. – Структура созданной БД.

```

export const Student : ModelCtor<Model> = sequelize.define( modelName: 'Student', attributes: {
  id: {
    type: DataTypes.SMALLINT,
    autoIncrement: true,
    primaryKey: true
  },
  firstName: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  lastName: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  patronymic: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  className: {type: DataTypes.TEXT...}
}, options: {
  timestamps: false
});

```

Рисунок 5. – Структура созданной БД.

```
export const Subject : ModelCtor<Model> = sequelize.define( modelName: 'Subject', attributes: {
  subjectName: {
    type: DataTypes.TEXT,
    primaryKey: true
  }
}, options: {
  timestamps: false
});
```

Рисунок 6. – Структура созданной БД.

```
export const Teacher : ModelCtor<Model> = sequelize.define( modelName: 'Teacher', attributes: {
  id: {
    type: DataTypes.SMALLINT,
    autoIncrement: true,
    primaryKey: true
  },
  firstName: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  lastName: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  patronymic: {
    type: DataTypes.TEXT,
    allowNull: false
  }
}, options: {
  timestamps: false
});
```

Рисунок 7. – Структура созданной БД.

```

export const TeacherCabinet : ModelCtor<Model> = sequelize.define( modelName: 'Teacher-Cabinet', attributes: {
  teacherId: {
    type: DataTypes.SMALLINT,
    primaryKey: true,
    onDelete: 'CASCADE',
    references: {
      model: Teacher,
      key: 'id'
    }
  },
  cabinetNumber: {
    type: DataTypes.SMALLINT,
    onDelete: 'CASCADE',
    references: {
      model: Cabinet,
      key: 'cabinetNumber'
    }
  }
}, options: {
  timestamps: false
});

```

Рисунок 8. – Структура созданной БД.

Выполним запросы к БД предложенные вариантом:

```

console.log('\n')
await sequelize.query(
  sql: 'SELECT "subjectName" AS "subject" FROM "Schedules" ' +
    'WHERE "className" = \'4B\' AND "day" = \'Среда\' AND "orderNumber" = 2;'
).then( ([res : unknown[] , _]) : void => {...});

console.log('\n')
await sequelize.query(
  sql: 'SELECT DISTINCT concat_ws(\' \', "lastName", "firstName", "patronymic") AS "teacher" ' +
    'FROM "Teachers" JOIN "Schedules" ' +
    'ON "Schedules"."teacherId" = "Teachers"."id" ' +
    'WHERE "className" = \'4B\';'
).then( ([res : unknown[] , _]) : void => {...});

console.log('\n')
await sequelize.query(
  sql: 'SELECT "cabinetNumber" AS "cabinet" FROM "Schedules" ' +
    'WHERE "className" = \'4B\' AND "orderNumber" = 5 AND "day" = \'Среда\';'
).then( ([res : unknown[] , _]) : void => {...});

```

Рисунок 9. – Запросы 1-3.

```

console.log('\n')
await sequelize.query(
  sql: 'SELECT DISTINCT "className" AS "class" FROM "Schedules" JOIN "Teachers" ' +
  'ON "Schedules"."teacherId" = "Teachers"."id" ' +
  'WHERE "subjectName" = \'Математика\' AND ' +
  '("lastName" = \'Иванов\' AND "firstName" = \'Иван\' AND "patronymic" = \'Иванович\');'
> ).then( ([res : unknown[], _]) : void => {...});

console.log('\n')
await sequelize.query(
  sql: 'SELECT "orderNumber", "subjectName" AS "subject", ' +
  'concat_ws(\' \', "lastName", "firstName", "patronymic") AS "teacher", "cabinetNumber" AS "cabinet" ' +
  'FROM "Schedules" JOIN "Teachers" ' +
  'ON "Schedules"."teacherId" = "Teachers"."id" ' +
  'WHERE "className" = \'4B\' AND "day" = \'Среда\' ' +
  'ORDER BY "orderNumber";'
> ).then( ([res : unknown[], _]) : void => {...});

console.log('\n')
await sequelize.query(
  sql: 'SELECT COUNT(*) AS "studentsCount" FROM "Students" ' +
  'WHERE "className" = \'4B\';'
> ).then( ([res : unknown[], _]) : void => {...});

```

Рисунок 10. – Запросы 4-6.

Ответы запросов:

```

Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?
[
  {
    "subject": "Физика"
  }
]

Кто из учителей преподает в заданном классе?
[
  {
    "teacher": "Забудь Светлана Ивановна"
  },
  {
    "teacher": "Иванов Иван Иванович"
  },
  {
    "teacher": "Макаревич Андрей Игоревич"
  },
  {
    "teacher": "Ожог Николай Витальевич"
  },
  {
    "teacher": "Попова Ульяна Владиславовна"
  },
  {
    "teacher": "Семашко Зинаида Климовна"
  }
]

В каком кабинете будет 5-й урок в среду у некоторого класса?
[
  {
    "cabinet": 12
  }
]

```



```
В каких классах преподает заданный предмет заданный учитель?  
[  
  {  
    "class": "4B"  
  },  
  {  
    "class": "9Б"  
  }  
]
```

```
Расписание на заданный день недели для указанного класса?  
[  
  {  
    "orderNumber": 1,  
    "subject": "Математика",  
    "teacher": "Иванов Иван Иванович",  
    "cabinet": 17  
  },  
  {  
    "orderNumber": 2,  
    "subject": "Физика",  
    "teacher": "Иванов Иван Иванович",  
    "cabinet": 12  
  },  
  {  
    "orderNumber": 3,  
    "subject": "Физкультура",  
    "teacher": "Попова Ульяна Владиславовна",  
    "cabinet": 16  
  },  
  {  
    "orderNumber": 4,  
    "subject": "География",  
    "teacher": "Макаревич Андрей Игоревич",  
    "cabinet": 14  
  },  
  {  
    "orderNumber": 5,  
    "subject": "Химия",  
    "teacher": "Семашко Зинаида Климовна",  
    "cabinet": 12  
  }  
]
```

```
Сколько учеников в указанном классе?  
[  
  {  
    "studentsCount": "2"  
  }  
]
```

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/39>