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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Базы данных»

Тема: Реализация базы данных с использованием ORM

Студент гр. 1303 _____ Гирман А.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Цель работы.

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

Задание.

Вариант 5

- 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 человеко-читабельный вывод)
- Запушить в репозиторий исходный код проекта, соблюсти .gitignore, убрать исходную базу из проекта (или иные на генерированные данные бд если они есть).
- Описать процесс запуска: команды, зависимости
- В отчете 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 работы бы установлены следующие пакеты:

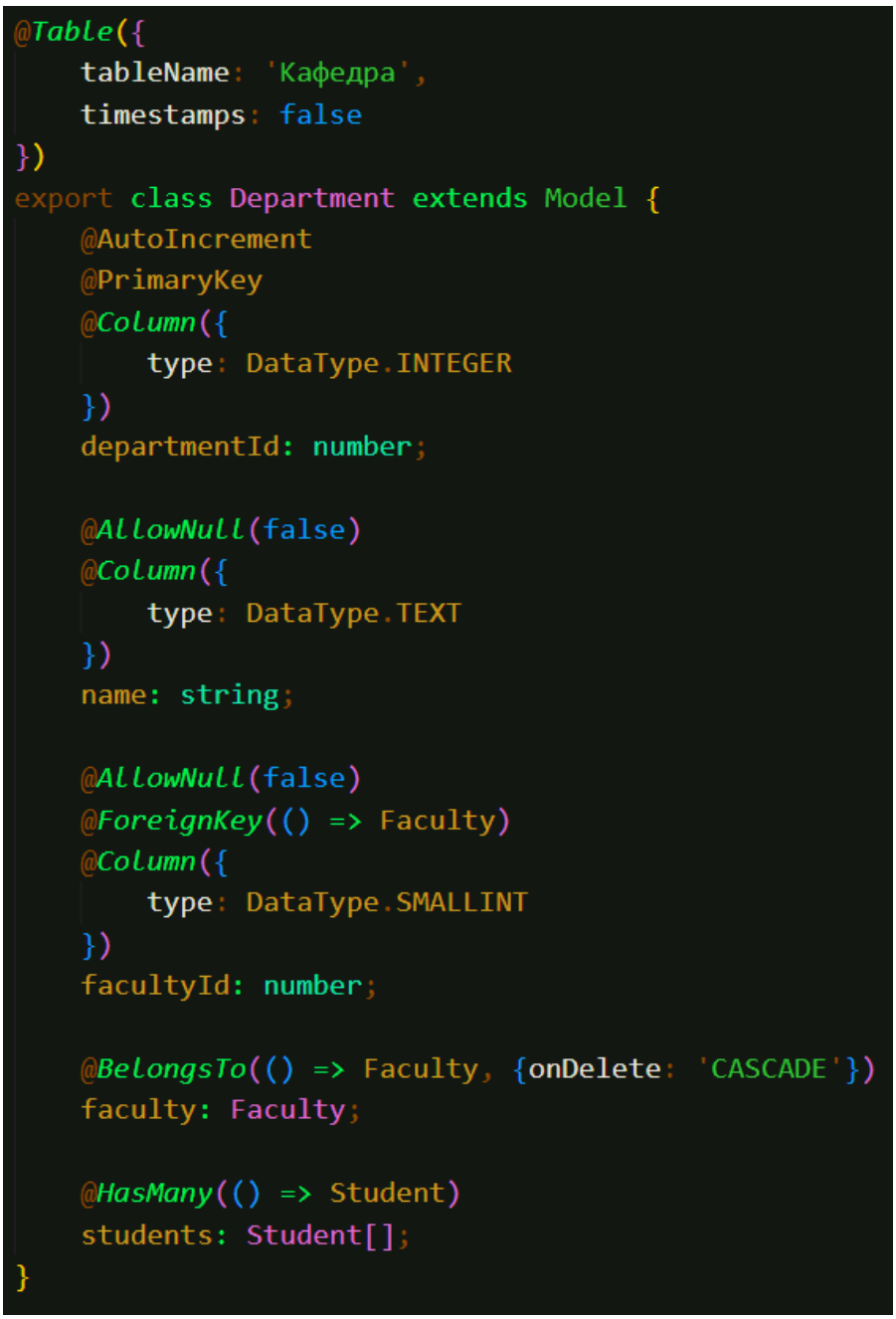
```
"dependencies": {
  "pg": "^8.11.3",
  "sequelize": "^6.34.0",
  "sequelize-typescript": "^2.1.5"
},
```

Рисунок 1. Файл package.json

Для запуска необходимо написать команду npx tsx app.ts.

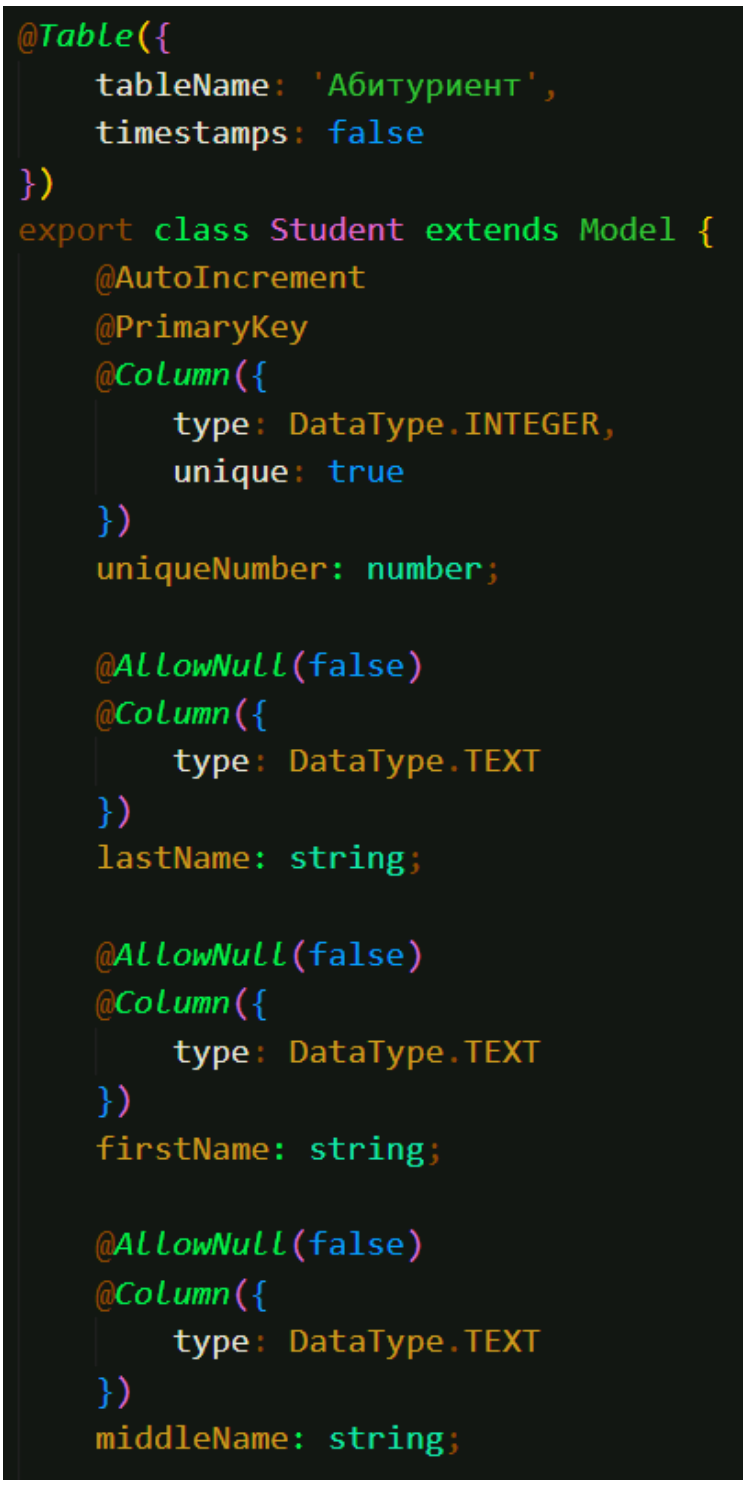
```
@Table({
    tableName: 'Факультет',
    timestamps: false
})
export class Faculty extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.SMALLINT
    })
    facultyId: number;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT,
        unique: true
    })
    name: string;

    @AllowNull(false)
    @Column({
        type: DataType.SMALLINT
    })
    places: number;

    @HasMany(() => Department)
    departments: Department[];
}
```

Рисунок 2. Модель таблицы «Факультет».



```
@Table({
    tableName: 'Кафедра',
    timestamps: false
})
export class Department extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.INTEGER
    })
    departmentId: number;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT
    })
    name: string;

    @AllowNull(false)
    @ForeignKey(() => Faculty)
    @Column({
        type: DataType.SMALLINT
    })
    facultyId: number;

    @BelongsTo(() => Faculty, {onDelete: 'CASCADE'})
    faculty: Faculty;

    @HasMany(() => Student)
    students: Student[];
}
```

Рисунок 3. Модель таблицы «Кафедра»



```
@Table({
    tableName: 'Абитуриент',
    timestamps: false
})
export class Student extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.INTEGER,
        unique: true
    })
    uniqueNumber: number;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT
    })
    lastName: string;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT
    })
    firstName: string;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT
    })
    middleName: string;
```

Рисунок 4.1. Модель таблицы «Абитуриент»

```
@AllowNull(false)
@Column({
    type: DataType.SMALLINT
})
passportSeries: number;

@AllowNull(false)
@Column({
    type: DataType.INTEGER
})
passportNumber: number;

@Column({
    type: DataType.TEXT
})
educationalInstitution: string | null;

@Column({
    type: DataType.TEXT
})
city: string | null;

@Column({
    type: DataType.DATEONLY
})
graduationDateOfTheEI: Date | null;

@Column({
    type: DataType.TEXT
})
medal: string | null;
```

Рисунок 4.2. Модель таблицы «Абитуриент»

```
@Column({
    type: DataType.TEXT
})
statusOfDocuments: string | null;

@AllowNull(false)
@Column({
    type: DataType.SMALLINT
})
numberOfExams: number;

@ForeignKey(() => Department)
@Column({
    type: DataType.SMALLINT
})
departmentId: number | null;

@BelongsTo(() => Department, {onDelete: 'SET NULL'})
department: Department;

@ForeignKey(() => Group)
@Column({
    type: DataType.INTEGER
})
groupId: number | null;

@BelongsTo(() => Group, {onDelete: 'SET NULL'})
group: Group;

@HasMany(() => ExaminationSheet)
examinationSheets: ExaminationSheet[];
```

Рисунок 4.3. Модель таблицы «Абитуриент»

```
@Table({
    tableName: 'Группа',
    timestamps: false
})
export class Group extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.INTEGER
    })
    groupId: number;

    @AllowNull(false)
    @Column({
        type: DataType.INTEGER
    })
    groupNumber: number;

    @ForeignKey(() => Stream)
    @Column({
        type: DataType.SMALLINT
    })
    streamId: number | null;

    @BelongsTo(() => Stream, {onDelete: 'SET NULL'})
    stream: Stream;

    @HasMany(() => Student)
    students: Student[];
}
```

Рисунок 5. Модель таблицы «Группа»

```
@Table({
    tableName: 'Поток',
    timestamps: false
})
export class Stream extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.SMALLINT
    })
    streamId: number;

    @AllowNull(false)
    @Column({
        type: DataType.SMALLINT
    })
    streamNumber: number;

    @HasMany(() => Group)
    groups: Group[];

    @HasMany(() => Exam)
    exams: Exam[];
}
```

Рисунок 6. Модель таблицы «Поток»

```
@Table({
    tableName: 'Экзамен',
    timestamps: false
})
export class Exam extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.SMALLINT
    })
    examId: number;

    @ForeignKey(() => Stream)
    @Column({
        type: DataType.SMALLINT
    })
    streamId: number | null;

    @BelongsTo(() => Stream, {onDelete: 'SET NULL'})
    stream: Stream;

    @AllowNull(false)
    @ForeignKey(() => Subject)
    @Column({
        type: DataType.SMALLINT
    })
    subjectId: number;

    @BelongsTo(() => Subject, {onDelete: 'CASCADE'})
    subject: Subject;
```

Рисунок 7.1. Модель таблицы «Экзамен»

```
    @AllowNull(false)
    @Column({
        type: DataType.DATEONLY
    })
    date: Date;

    @AllowNull(false)
    @Column({
        type: DataType.TIME
    })
    time: string;

    @Column({
        type: DataType.INTEGER
    })
    auditorium: number | null;

    @HasMany(() => ExaminationSheet)
    examinationSheets: ExaminationSheet[];

    @HasMany(() => Consultation)
    consultations: Consultation[];
}
```

Рисунок 7.2. Модель таблицы «Экзамен»

```
@Table({
    tableName: 'Предмет',
    timestamps: false
})
export class Subject extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.SMALLINT
    })
    subjectId: number;

    @AllowNull(false)
    @Column({
        type: DataType.TEXT
    })
    name: string;

    @HasMany(() => Exam)
    exams: Exam[];
}
```

Рисунок 8. Модель таблицы «Предмет»

```
@Table({
    tableName: 'Консультация',
    timestamps: false
})
export class Consultation extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column({
        type: DataType.SMALLINT
    })
    consultationId: number;

    @ForeignKey(() => Exam)
    @Column({
        type: DataType.SMALLINT
    })
    examId: number | null;

    @BelongsTo(() => Exam, {onDelete: 'SET NULL'})
    exam: Exam;

    @AllowNull(false)
    @Column({
        type: DataType.DATEONLY
    })
    date: Date;

    @AllowNull(false)
    @Column({
        type: DataType.TIME
    })
    time: string;
```

Рисунок 9.1. Модель таблицы «Консультация»

```
    @Column({
        type: DataType.INTEGER
    })
    auditorium: number | null;
}
```

Рисунок 9.2. Модель таблицы «Консультация»

```
@Table({
    tableName: 'Ведомость',
    timestamps: false
})
export class ExaminationSheet extends Model {
    @PrimaryKey
    @ForeignKey(() => Exam)
    @Column({
        type: DataType.SMALLINT
    })
    examId: number | null;

    @BelongsTo(() => Exam, {onDelete: 'SET NULL'})
    exam: Exam;

    @PrimaryKey
    @ForeignKey(() => Student)
    @Column({
        type: DataType.INTEGER
    })
    studentId: number | null;

    @BelongsTo(() => Student, {onDelete: 'SET NULL'})
    student: Student;

    @AllowNull(false)
    @Column({
        type: DataType.SMALLINT
    })
    grade: number;
```

Рисунок 10.1. Модель таблицы «Ведомость»

```
    @AllowNull(false)
    @Column({
        type: DataType.BOOLEAN
    })
    Appeal: boolean;
}
```

Рисунок 10.2. Модель таблицы «Ведомость»

Рассмотрим запросы на заполнение таблиц:

```
await Faculty.bulkCreate([
    {name: 'ФРТ', places: 155},
    {name: 'ФЭЛ', places: 198},
    {name: 'ФКТИ', places: 308},
    {name: 'ФЭА', places: 243},
    {name: 'ФИБС', places: 215},
    {name: 'ИНПРОТЕХ', places: 52},
    {name: 'ГФ', places: 20}
]);
```

Рисунок 11. Заполнение таблицы «Факультет».

```
await Department.bulkCreate([
    { name: 'РЭС', facultyId: 1 },
    { name: 'ФЭТ', facultyId: 2 },
    { name: 'МО ЭВМ', facultyId: 3 },
    { name: 'КСУ', facultyId: 4 },
    { name: 'БТС', facultyId: 5 },
    { name: 'МСК', facultyId: 6 },
    { name: 'ИНЯЗ', facultyId: 7 }
]);
```

Рисунок 12. Заполнение таблицы «Кафедра».

```
await Stream.bulkCreate([
    {streamNumber: 1},
    {streamNumber: 2}
]);
```

Рисунок 13. Заполнение таблицы «Поток»

```
await Group.bulkCreate([
    { groupNumber: 1301, streamId: 1 },
    { groupNumber: 1302, streamId: 1 },
    { groupNumber: 1303, streamId: 1 },
    { groupNumber: 1304, streamId: 1 },
    { groupNumber: 1305, streamId: 2 },
    { groupNumber: 1306, streamId: 2 },
    { groupNumber: 1307, streamId: 2 }
]);
```

Рисунок 14. Заполнение таблицы «Группа»

```
await Student.bulkCreate([
    { uniqueNumber: 15834161, lastName: 'Коновалов', firstName: 'Гавриил', middleName: 'Анатольевич',
      passportSeries: 5719, passportNumber: 529588, educationalInstitution: 'МАОУ СОШ №3', city: 'Пермь',
      graduationDateOfTheEI: new Date('2023-05-20'), medal: 'Золотая', statusOfDocuments: 'Поданы',
      numberOfExams: 1, departmentId: 7, groupId: 5 },
    { uniqueNumber: 15089468, lastName: 'Шестаков', firstName: 'Вальтер', middleName: 'Всеволодович',
      passportSeries: 2419, passportNumber: 666949, educationalInstitution: 'Лицей №67', city: 'Иваново',
      graduationDateOfTheEI: new Date('2023-05-25'), medal: 'Золотая', statusOfDocuments: 'Поданы',
      numberOfExams: 1, departmentId: 3, groupId: 1 },
    { uniqueNumber: 13669205, lastName: 'Авдеев', firstName: 'Леонтий', middleName: 'Эдуардович',
      passportSeries: 6119, passportNumber: 975207, educationalInstitution: 'Многопрофильная школа № 17',
      city: 'Рязань', graduationDateOfTheEI: new Date('2023-05-31'), medal: null, statusOfDocuments: null,
      numberOfExams: 4, departmentId: 7, groupId: 5 },
    { uniqueNumber: 19828121, lastName: 'Меркушев', firstName: 'Альфред', middleName: 'Протасьевич',
      passportSeries: 8919, passportNumber: 447416, educationalInstitution: 'Школа №27', city: 'Мордовия',
      graduationDateOfTheEI: new Date('2023-05-31'), medal: null, statusOfDocuments: 'Поданы',
      numberOfExams: 4, departmentId: 3, groupId: 1 },
    { uniqueNumber: 21952948, lastName: 'Носова', firstName: 'Рамина', middleName: 'Михаиловна',
      passportSeries: 9620, passportNumber: 602158, educationalInstitution: 'Школа №48', city: 'Грозный',
      graduationDateOfTheEI: new Date('2023-05-15'), medal: 'Золотая', statusOfDocuments: null,
      numberOfExams: 1, departmentId: 3, groupId: 3 },
    { uniqueNumber: 66862382, lastName: 'Лукина', firstName: 'Анэля', middleName: 'Аристарховна',
      passportSeries: 7919, passportNumber: 269571, educationalInstitution: 'Школа №15', city: 'Майкоп',
      graduationDateOfTheEI: new Date('2023-05-31'), medal: 'Серебряная', statusOfDocuments: 'Поданы',
      numberOfExams: 1, departmentId: 3, groupId: 2 },
    { uniqueNumber: 98664711, lastName: 'Кузнецова', firstName: 'Юнона', middleName: 'Ильяовна',
      passportSeries: 3620, passportNumber: 328391, educationalInstitution: 'МБОУ Школа №32', city: 'Самара',
      graduationDateOfTheEI: new Date('2023-05-31'), medal: null, statusOfDocuments: 'Поданы',
      numberOfExams: 4, departmentId: 3, groupId: 3 },
    { uniqueNumber: 12636043, lastName: 'Мартынова', firstName: 'Карина', middleName: 'Мартыновна',
      passportSeries: 4519, passportNumber: 473248, educationalInstitution: 'Школа №91', city: 'Москва',
      graduationDateOfTheEI: new Date('2023-05-31'), medal: null, statusOfDocuments: 'Перевод',
      numberOfExams: 4, departmentId: 7, groupId: 6 }
]);
```

Рисунок 15. Заполнение таблицы «Абитуриент»

```
await Subject.bulkCreate([
    {name: 'Математика'},
    {name: 'Физика'},
    {name: 'Информатика'},
    {name: 'Физкультура'},
    {name: 'Русский Язык'},
    {name: 'Английский Язык'},
    {name: 'Литература'}
]);
```

Рисунок 16. Заполнение таблицы «Предмет»

```
await Exam.bulkCreate([
    { streamId: 1, subjectId: 1, date: new Date('2023-08-10'), time: '10:00', auditorium: 3322 },
    { streamId: 1, subjectId: 2, date: new Date('2023-08-15'), time: '9:00', auditorium: 3102 },
    { streamId: 1, subjectId: 3, date: new Date('2023-08-20'), time: '15:00', auditorium: 5413 },
    { streamId: 2, subjectId: 7, date: new Date('2023-08-16'), time: '11:00', auditorium: 3413 },
    { streamId: 2, subjectId: 6, date: new Date('2023-08-19'), time: '9:00', auditorium: 5413 },
    { streamId: 2, subjectId: 5, date: new Date('2023-08-23'), time: '14:00', auditorium: 3102 }
])
```

Рисунок 17. Заполнение таблицы «Экзамен»

```
await Consultation.bulkCreate([
    { examId: 1, date: new Date('2024-08-05'), time: '18:00:00', auditorium: null },
    { examId: 2, date: new Date('2024-08-14'), time: '13:00:00', auditorium: 3102 },
    { examId: 3, date: new Date('2024-08-19'), time: '15:00:00', auditorium: null },
    { examId: 4, date: new Date('2024-08-08'), time: '15:30:00', auditorium: null },
    { examId: 5, date: new Date('2024-08-14'), time: '17:00:00', auditorium: null },
    { examId: 6, date: new Date('2024-08-17'), time: '18:30:00', auditorium: null }
])
```

Рисунок 18. Заполнение таблицы «Консультация»

```
await ExaminationSheet.bulkCreate([
    { examId: 1, studentId: 15089468, grade: 5, Appeal: false },
    { examId: 1, studentId: 19828121, grade: 4, Appeal: false },
    { examId: 1, studentId: 21952948, grade: 3, Appeal: false },
    { examId: 1, studentId: 66862382, grade: 2, Appeal: true },
    { examId: 1, studentId: 98664711, grade: 3, Appeal: true },
    { examId: 2, studentId: 19828121, grade: 5, Appeal: false },
    { examId: 2, studentId: 98664711, grade: 4, Appeal: false },
    { examId: 3, studentId: 19828121, grade: 3, Appeal: true },
    { examId: 3, studentId: 98664711, grade: 4, Appeal: false },
    { examId: 4, studentId: 15834161, grade: 5, Appeal: false },
    { examId: 4, studentId: 13669205, grade: 4, Appeal: false },
    { examId: 4, studentId: 12636043, grade: 4, Appeal: true },
    { examId: 5, studentId: 13669205, grade: 5, Appeal: false },
    { examId: 5, studentId: 12636043, grade: 5, Appeal: false },
    { examId: 6, studentId: 13669205, grade: 4, Appeal: false },
    { examId: 6, studentId: 12636043, grade: 3, Appeal: false }
]);
```

Рисунок 19. Заполнение таблицы «Ведомость»

Рассмотрим запросы на вывод информации из задания:

```
task = 'Список абитуриентов на заданный факультет?\n';
const res1 = await Student.findAll({
    attributes: [
        'uniqueNumber',
        'lastName',
        'firstName',
        'middleName',
        [Sequelize.col('department.faculty.name'), 'facultyName']
    ],
    include: [{
        model: Department,
        required: true,
        attributes: [],
        include: [{
            model: Faculty,
            required: true,
            attributes: [],
            where: { name: 'ФКТИ' }
        }]
    }],
});
fs.writeFileSync('taskResult.txt', task + JSON.stringify(res1, null, 2));
```

Рисунок 20. Запрос «Список абитуриентов на заданный факультет?»

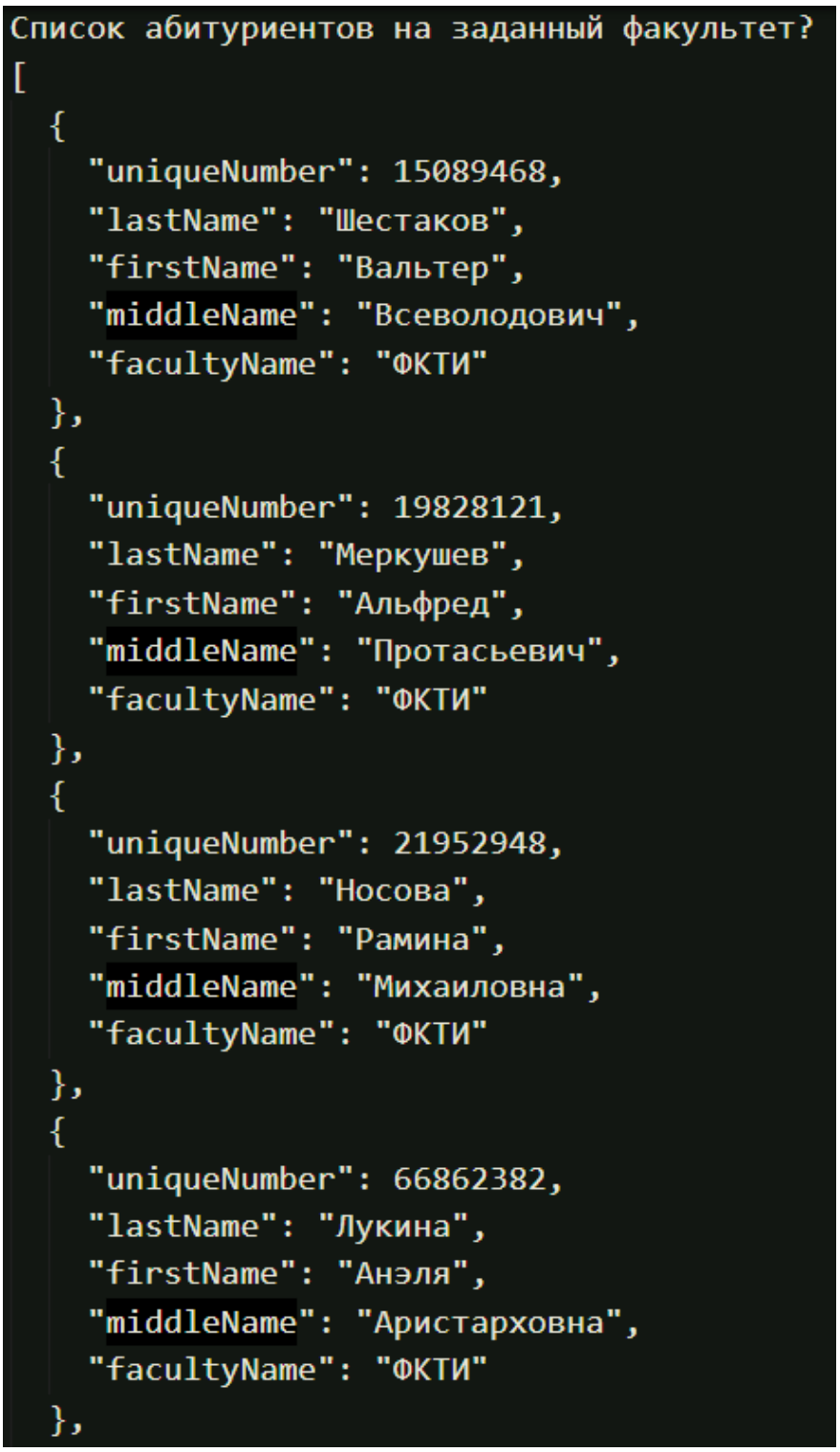


Рисунок 21.1. Результат запроса.

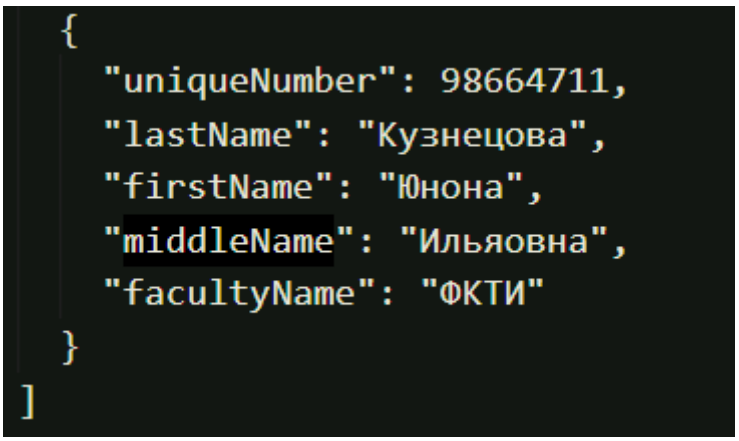


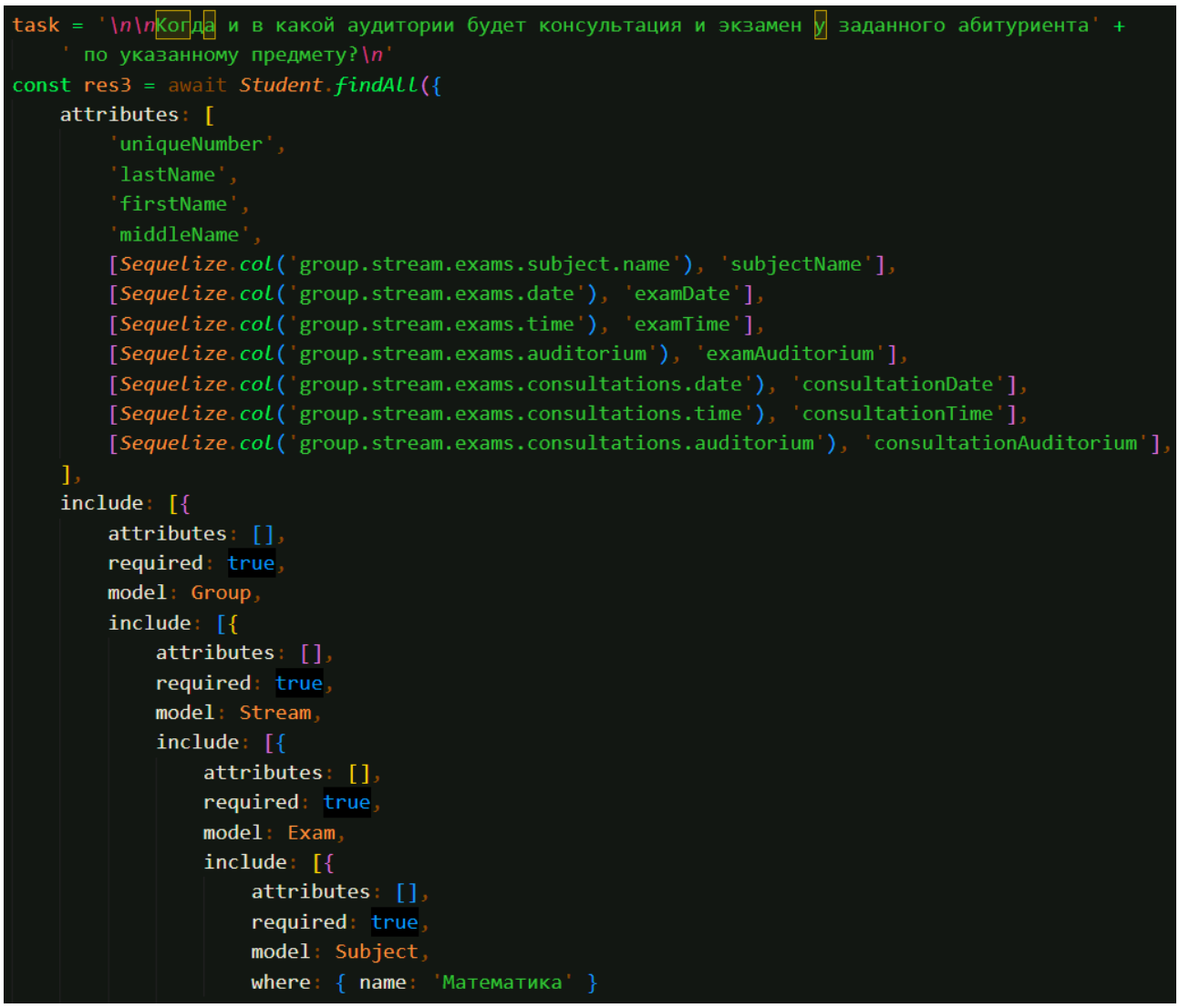
Рисунок 21.2. Результат запроса.

```
task = '\n\nОценки, полученные указанным абитуриентом?\n'
const res2 = await ExaminationSheet.findAll({
    attributes: [
        [Sequelize.col('student.lastName'), 'lastName'],
        [Sequelize.col('student.firstName'), 'firstName'],
        [Sequelize.col('student.middleName'), 'middleName'],
        [Sequelize.col('exam.subject.name'), 'subjectName'],
        'grade'
    ],
    raw: true,
    include: [{
        model: Student,
        required: true,
        attributes: [],
        where: {uniqueNumber: 19828121},
    }, {
        model: Exam,
        required: true,
        attributes: [],
        include: [{
            model: Subject,
            required: true,
            attributes: []
        }]
    }]
});
fs.appendFileSync('taskResult.txt', task + JSON.stringify(res2, null, 2));
```

Рисунок 22. Запрос «Оценки, полученные указанным абитуриентом?»

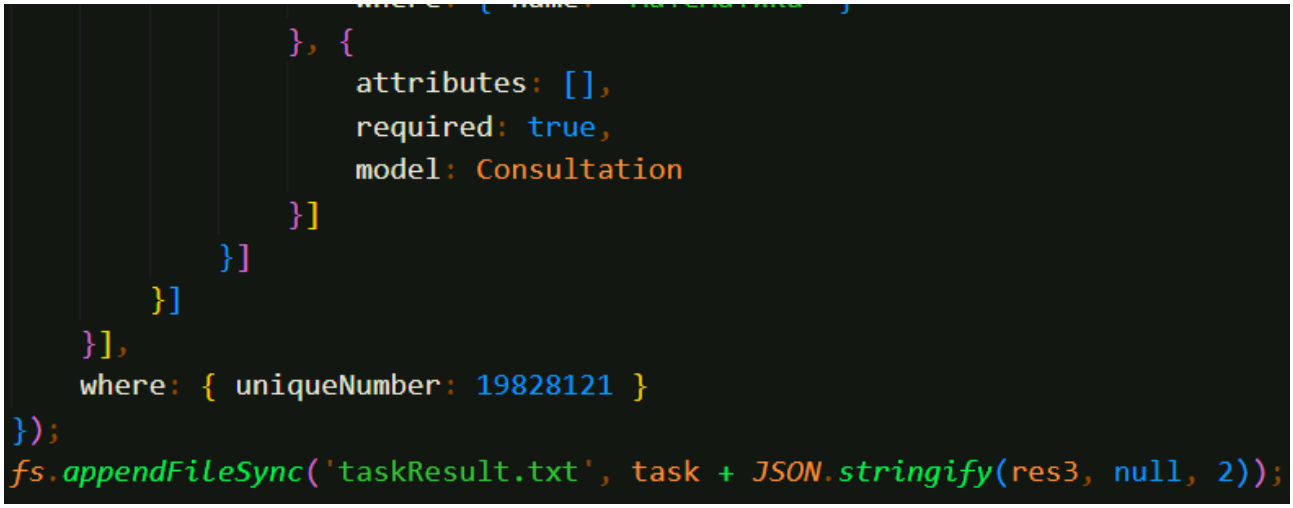
```
Оценки, полученные указанным абитуриентом?
[
  {
    "lastName": "Меркушев",
    "firstName": "Альфред",
    "middleName": "Протасьевич",
    "subjectName": "Математика",
    "grade": 4
  },
  {
    "lastName": "Меркушев",
    "firstName": "Альфред",
    "middleName": "Протасьевич",
    "subjectName": "Физика",
    "grade": 5
  },
  {
    "lastName": "Меркушев",
    "firstName": "Альфред",
    "middleName": "Протасьевич",
    "subjectName": "Информатика",
    "grade": 3
  }
]
```


Рисунок 23. Результат запроса



```
task = '\n\nКогда и в какой аудитории будет консультация и экзамен у заданного абитуриента' +
    ' по указанному предмету?\n'
const res3 = await Student.findAll({
    attributes: [
        'uniqueNumber',
        'lastName',
        'firstName',
        'middleName',
        [Sequelize.col('group.stream.exams.subject.name'), 'subjectName'],
        [Sequelize.col('group.stream.exams.date'), 'examDate'],
        [Sequelize.col('group.stream.exams.time'), 'examTime'],
        [Sequelize.col('group.stream.exams.auditorium'), 'examAuditorium'],
        [Sequelize.col('group.stream.exams.consultations.date'), 'consultationDate'],
        [Sequelize.col('group.stream.exams.consultations.time'), 'consultationTime'],
        [Sequelize.col('group.stream.exams.consultations.auditorium'), 'consultationAuditorium'],
    ],
    include: [{
        attributes: [],
        required: true,
        model: Group,
        include: [{
            attributes: [],
            required: true,
            model: Stream,
            include: [{
                attributes: [],
                required: true,
                model: Exam,
                include: [{
                    attributes: [],
                    required: true,
                    model: Subject,
                    where: { name: 'Математика' }
```

Рисунок 24.1. Запрос «Когда и в какой аудитории будет консультация и экзамен у заданного абитуриента по указанному предмету?»



```
                }, {
                    attributes: [],
                    required: true,
                    model: Consultation
                }]
            }]
        }]
    }],
    where: { uniqueNumber: 19828121 }
});
fs.appendFileSync('taskResult.txt', task + JSON.stringify(res3, null, 2));
```

Рисунок 24.2 Запрос «Когда и в какой аудитории будет консультация и экзамен у заданного абитуриента по указанному предмету?»

```
Когда и в какой аудитории будет консультация и экзамен у заданного абитуриента по указанному предмету?
[
  {
    "uniqueNumber": 19828121,
    "lastName": "Меркушев",
    "firstName": "Альфред",
    "middleName": "Протасьевич",
    "subjectName": "Математика",
    "examDate": "2023-08-10",
    "examTime": "10:00:00",
    "examAuditorium": 3322,
    "consultationDate": "2024-08-05",
    "consultationTime": "18:00:00",
    "consultationAuditorium": null
  }
]
```

Рисунок 25. Результат запроса

```
task = '\n\nГде, когда и по каким предметам будут проходить экзамены у заданной группы?\n'
const res4 = await Group.findAll({
    attributes: [
        'groupNumber',
        [Sequelize.col('stream.exams.subject.name'), 'subjectName'],
        [Sequelize.col('stream.exams.date'), 'date'],
        [Sequelize.col('stream.exams.time'), 'time'],
        [Sequelize.col('stream.exams.auditorium'), 'auditorium']
    ],
    raw: true,
    where: { groupNumber: 1303 },
    include: [{
        attributes: [],
        model: Stream,
        include: [{
            attributes: [],
            model: Exam,
            include: [{
                attributes: [],
                model: Subject
            }]
        }]
    }]
});
fs.appendFileSync('taskResult.txt', task + JSON.stringify(res4, null, 2));
```

Рисунок 26. Запрос «Где, когда и по каким предметам будут проходить экзамены у заданной группы?»

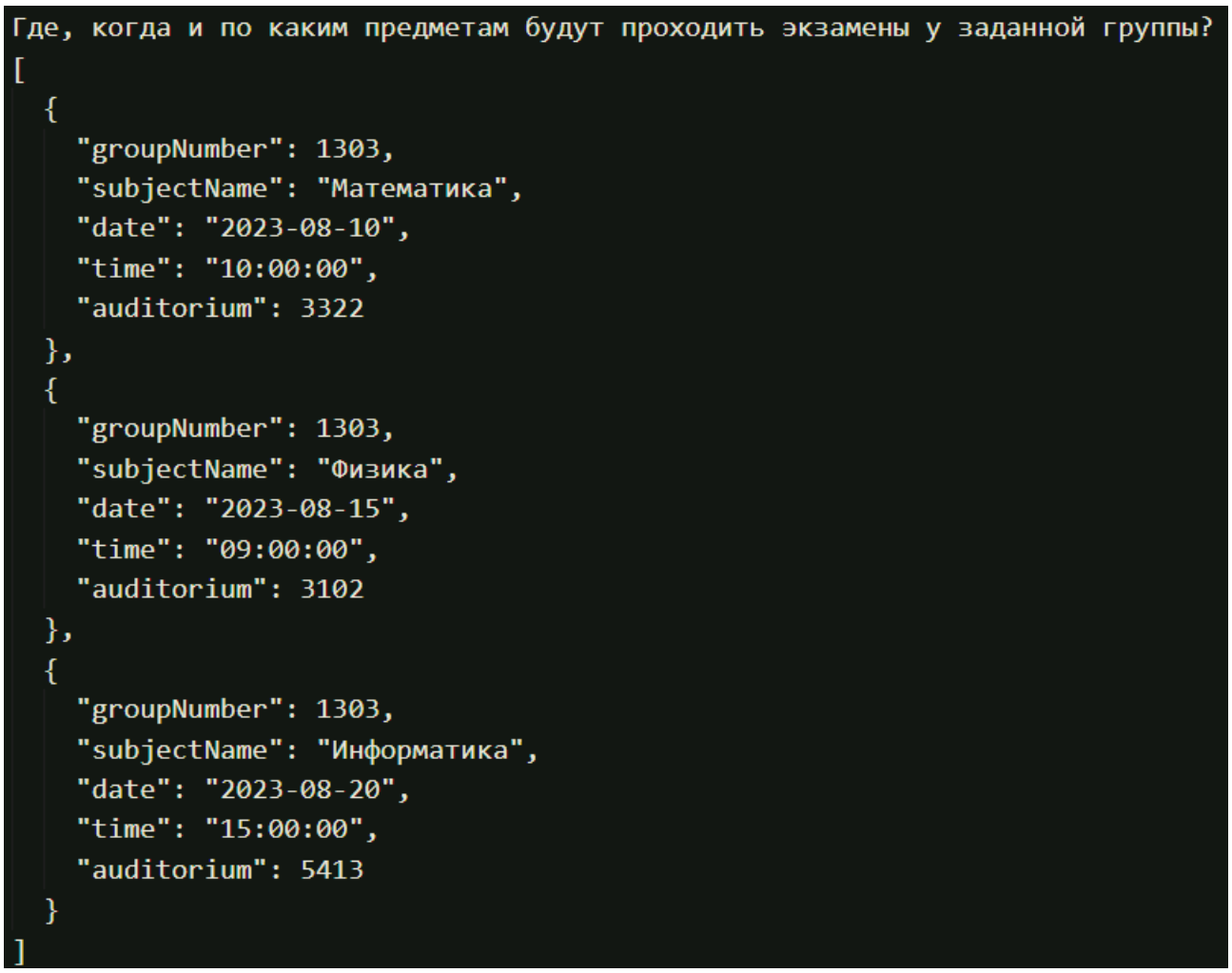


Рисунок 27. Результат запроса

```
task = '\n\nКонкурс на каждый факультет?\n'
const res5 = await Student.findAll({
    attributes: [
        [Sequelize.col('department.faculty.name'), 'facultyName'],
        [Sequelize.fn('COUNT', Sequelize.col('uniqueNumber')), 'competition']
    ],
    include: [{
        attributes: [],
        model: Department,
        include: [{
            attributes: [],
            model: Faculty,
        }]
    }],
    group: [Sequelize.col('department.faculty.name')],
    raw: true
});
fs.appendFileSync('taskResult.txt', task + JSON.stringify(res5, null, 2));
```

Рисунок 28. Запрос «Конкурс на каждый факультет?»

```
Конкурс на каждый факультет?
[
  {
    "facultyName": "ГФ",
    "competition": "3"
  },
  {
    "facultyName": "ФКТИ",
    "competition": "5"
  }
]
```

Рисунок 29. Результат запрос

```
task = '\n\nСредний балл по каждому предмету на каждом факультете?\n'
const res6 = await Student.findAll({
    attributes: [
        [Sequelize.col('department.faculty.name'), 'facultyName'],
        [Sequelize.col('examinationSheets.exam.subject.name'), 'subjectName'],
        [Sequelize.fn('ROUND', Sequelize.fn('AVG', Sequelize.col('examinationSheets.grade')), 2), 'averageScore']
    ],
    raw: true,
    include: [{
        attributes: [],
        model: Department,
        include: [{
            attributes: [],
            model: Faculty
        }]
    }, {
        attributes: [],
        model: ExaminationSheet,
        include: [{
            attributes: [],
            model: Exam,
            include: [{
                attributes: [],
                model: Subject
            }]
        }]
    }],
    group: ['facultyName', 'subjectName'],
    order: Sequelize.literal('1'),
});
fs.appendFileSync('taskResult.txt', task + JSON.stringify(res6, null, 2))
```

Рисунок 30. Запрос «Средний балл по каждому предмету на каждом факультете?»

```
Средний балл по каждому предмету на каждом факультете?
[
  {
    "facultyName": "ГФ",
    "subjectName": "Английский Язык",
    "averageScore": "5.00"
  },
  {
    "facultyName": "ГФ",
    "subjectName": "Литература",
    "averageScore": "4.33"
  },
  {
    "facultyName": "ГФ",
    "subjectName": "Русский Язык",
    "averageScore": "3.50"
  },
  {
    "facultyName": "ФКТИ",
    "subjectName": "Информатика",
    "averageScore": "3.50"
  },
  {
    "facultyName": "ФКТИ",
    "subjectName": "Математика",
    "averageScore": "3.40"
  },
  {
    "facultyName": "ФКТИ",
    "subjectName": "Физика",
    "averageScore": "4.50"
  }
]
```


Рисунок 31. Результат запроса

Вывод.

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

Приложение А

Ссылки

Pull Request: https://github.com/moevm/sql-2023-1303/pull/46