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 _____ Смирнов Д.Ю.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Цель работы.

Развернуть Sequalize, написать запросы для создания и заполнения таблиц, написать запросы к БД используя ORM, отвечающие на вопросы задания.

Задание.

Вариант 19

Пусть требуется создать программную систему, предназначенную для работников технического архива предприятия. Технический архив содержит стеллажи, полки и ячейки, в которых хранится документация. Ячейка архива может быть пустой или хранить все экземпляры одного документа. Каждый экземпляр документации имеет инвентарный номер и название. В базе данных должна храниться следующая информация о каждом документе архива: номер стеллажа, номер полки, номер ячейки, где хранится документ, название документа и название темы, к которой он относится, его инвентарный номер, количество экземпляров документа, содержащихся в ячейке, дата поступления документа в архив. Документ может быть востребован абонентом архива. Абонент характеризуется фамилией, именем, отчеством, номером и телефоном отдела, где он работает. Работники архива, выдавая документ, должны зафиксировать, когда и кому он был выдан. Архив может пополняться документами, как новыми, так и копиями уже имеющихся в архиве. Экземпляр документа может быть утрачен. Возможна закупка новых стеллажей и списание старых. Документ может поменять место хранения и инвентарный номер. Возможно и изменение сведений об абонентах. Абонент может поменять фамилию, перейти в другой отдел, уволится с предприятия. Возможно изменение номеров телефонов отделов. Работнику архива могут потребоваться следующие сведения:

- Название наиболее востребованного документа?
- Общее количество документов на заданную тему?
- Тема документа по заданному названию?

- Название документа, который имеется в архиве в максимальном количестве экземпляров?
- Фамилия, имя и отчество абонента, который брал указанный документ последним?
- Есть ли в архиве пустые стеллажи, полки, ячейки, и в каком количестве?
- Список документов, не востребованных в течение более, чем 5 лет?

Выполнение работы.

Создана база данных “Archive”, состоящая из 9 моделей “Shelf”, “Rack”, “Cell”, “Document”, “Instance”, “Issues”, “IssuesArchive”, “Abonent”, “Department”, рисунок 1-9.

Полученные модели заполнены значениями, в каждой от 5 строк. Результат представлен на рисунках рисунки 10-12.

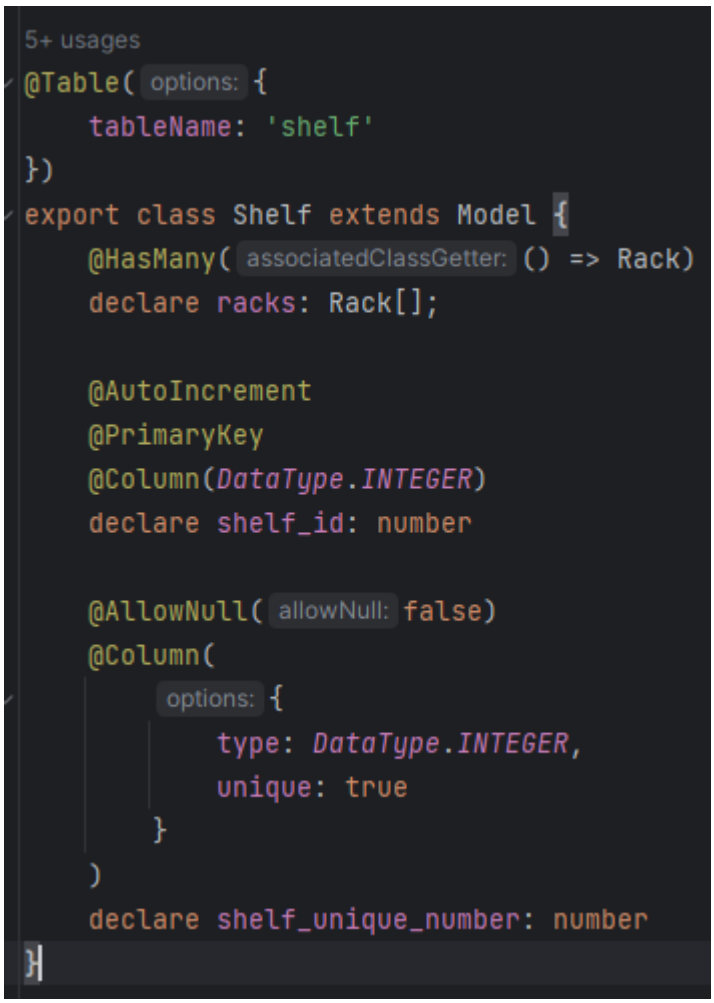
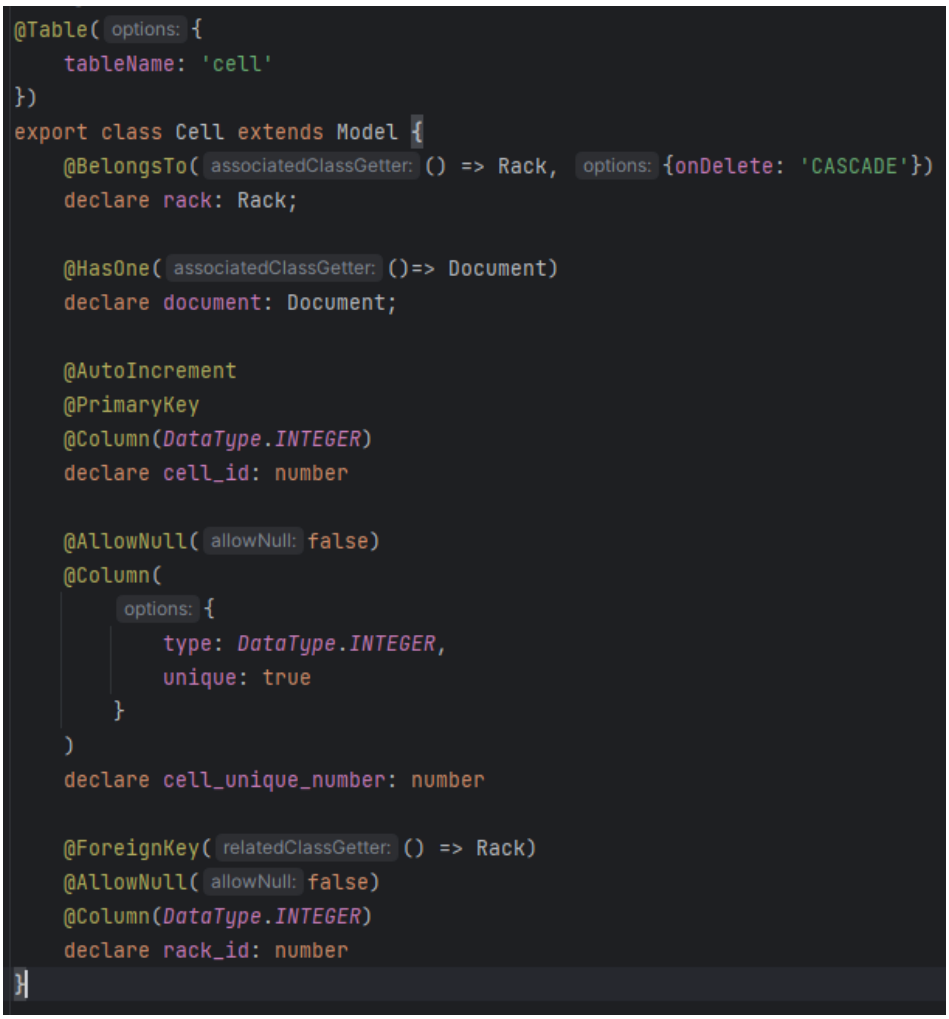


Рисунок 1 - модель "Shelf"

```
5+ usages
@Table( options: {
    tableName: 'rack'
})
export class Rack extends Model {
    @BelongsTo( associatedClassGetter: ()=> Shelf,  options: {onDelete: 'CASCADE'})
    declare shelf: Shelf;
    @HasMany( associatedClassGetter: ()=> Cell)
    declare cells: Cell[];

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare rack_id: number;

    @AllowNull( allowNull: false)
    @Column(
        options: {
            type: DataType.INTEGER,
            unique: true
        }
    )
    declare rack_unique_number: number;

    @ForeignKey( relatedClassGetter: () => Shelf)
    @AllowNull( allowNull: false)
    @Column(DataType.INTEGER)
    declare shelf_id: number;
}
```

Рисунок 2 - Модель "Rack"



```
@Table( options: {
    tableName: 'cell'
})
export class Cell extends Model {
    @BelongsTo( associatedClassGetter: () => Rack,  options: {onDelete: 'CASCADE'})
    declare rack: Rack;

    @HasOne( associatedClassGetter: ()=> Document)
    declare document: Document;

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare cell_id: number

    @AllowNull( allowNull: false)
    @Column(
        options: {
            type: DataType.INTEGER,
            unique: true
        }
    )
    declare cell_unique_number: number

    @ForeignKey( relatedClassGetter: () => Rack)
    @AllowNull( allowNull: false)
    @Column(DataType.INTEGER)
    declare rack_id: number
}
```

Рисунок 3 - Модель "Cell"

```
@Table( options: {
    tableName: 'document',
    paranoid: true
})
export class Document extends Model{
    @BelongsTo( associatedClassGetter: () => Cell,  options: {onDelete: "RESTRICT"})
    declare cell: Cell

    @HasMany( associatedClassGetter: () => Instance)
    declare instances: Instance[]

    @HasMany( associatedClassGetter: () => IssuesArchive)
    declare previous_issues: IssuesArchive[]

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare document_id: number

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare theme_name: string

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare document_title: string

    @AllowNull( allowNull: false)
    @Column( options: {
            type: DataType.STRING,
            unique: true
        })
    declare inventory_number: string

    @ForeignKey( relatedClassGetter: () => Cell)
    @AllowNull( allowNull: false)
    @Column(DataType.INTEGER)
    declare cell_id: number
}
```

Рисунок 4 - Модель "Document"

```
@Table( options: {
    tableName: 'instance'
})
export class Instance extends Model {
    @BelongsTo( associatedClassGetter: () => Document,  options: {onDelete: "CASCADE"})
    declare document: Document
    @HasOne( associatedClassGetter: () => Issues)
    declare issue: Issues

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare instance_id: number

    @AllowNull( allowNull: false)
    @Column( options: {
        type: DataType.STRING,
        unique: true
    })
    declare inventory_number: string

    @AllowNull( allowNull: false)
    @Column( options: {
        type: DataType.ENUM,
        values: ['есть', 'утерян'],
        defaultValue: 'есть'
    })
    declare status: string

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare title: string

    @ForeignKey( relatedClassGetter: () => Document)
    @AllowNull( allowNull: false)
    @Column(DataType.INTEGER)
    declare document_id: number
}
```

Рисунок 5 - Модель "Instance"

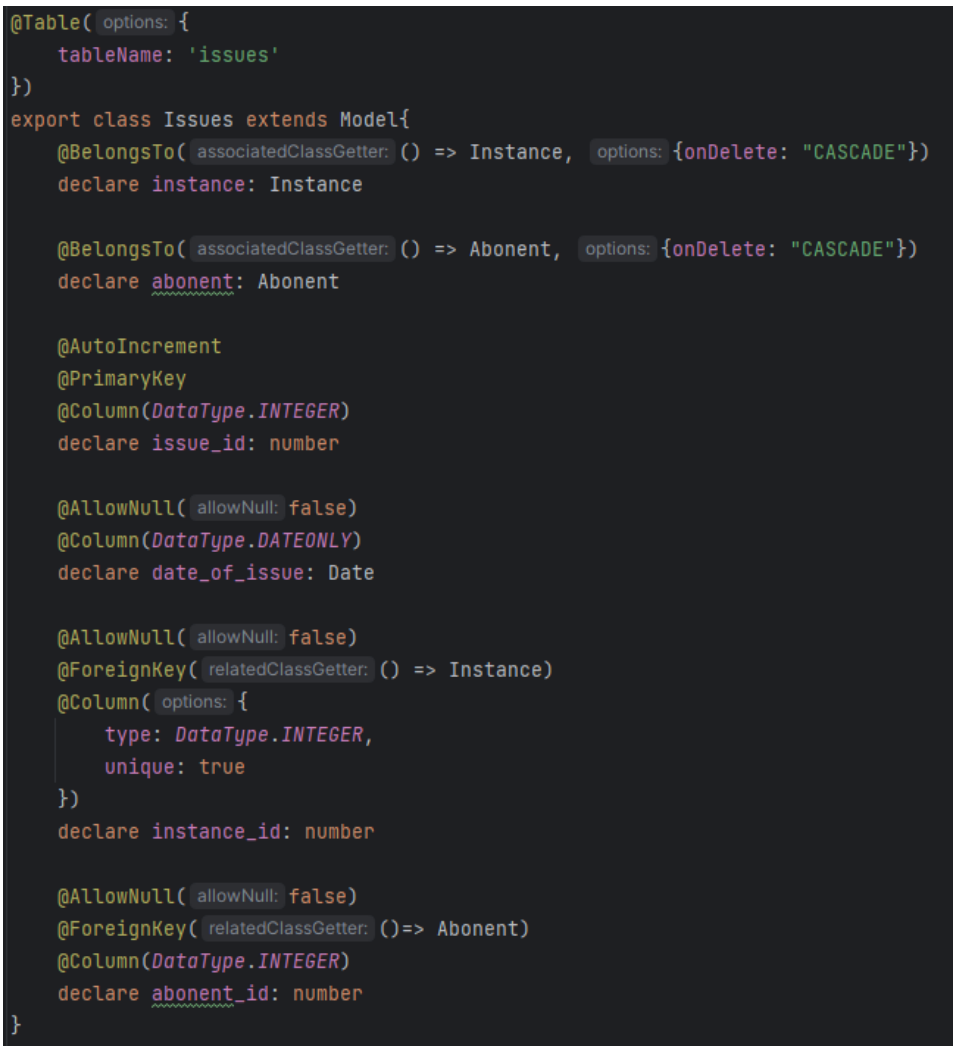


Рисунок 6 - модель "Issues"

```
@Table( options: {
    tableName: 'issues_archive',
    paranoid: true
})
export class IssuesArchive extends Model{

    @BelongsTo( associatedClassGetter: () => Document,  options: {onDelete: "CASCADE"})
    declare document: Document

    @BelongsTo( associatedClassGetter: ()=> Abonent,  options: {onDelete: "CASCADE"})
    declare abonent: Abonent

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare issues_archive_id

    @AllowNull( allowNull: false)
    @ForeignKey( relatedClassGetter: () => Document)
    @Column(DataType.INTEGER)
    declare document_id: number

    @AllowNull( allowNull: false)
    @Column(DataType.DATEONLY)
    declare date_of_issue: Date

    @AllowNull( allowNull: false)
    @ForeignKey( relatedClassGetter: () => Abonent)
    @Column(DataType.INTEGER)
    declare abonent_id: number
}
```

Рисунок 7 - моделей "IssuesArchive"

```
@Table( options: {
    tableName: 'abonent',
    paranoid: true
})
export class Abonent extends Model{
    @BelongsTo( associatedClassGetter: () => Department,  options: {onDelete: "CASCADE"})
    declare department: Department

    @HasMany( associatedClassGetter: () => Issues)
    declare issues: Issues[]

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare abonent_id: number

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare phone_number: string

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare name: string

    @Column(DataType.STRING)
    declare surname: string | null

    @Column(DataType.STRING)
    declare middle_name: string | null

    @AllowNull( allowNull: false)
    @ForeignKey( relatedClassGetter: () => Department)
    @Column(DataType.INTEGER)
    declare department_id: number
```

Рисунок 8 - модель "Abonent"

```
@Table( options: {
    tableName: 'department'
})
export class Department extends Model{
    @HasMany( associatedClassGetter: () => Abonent)
    declare abonents: Abonent[]

    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    declare department_id: number

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare department_name: string

    @AllowNull( allowNull: false)
    @Column(DataType.STRING)
    declare department_phone: string
}
```

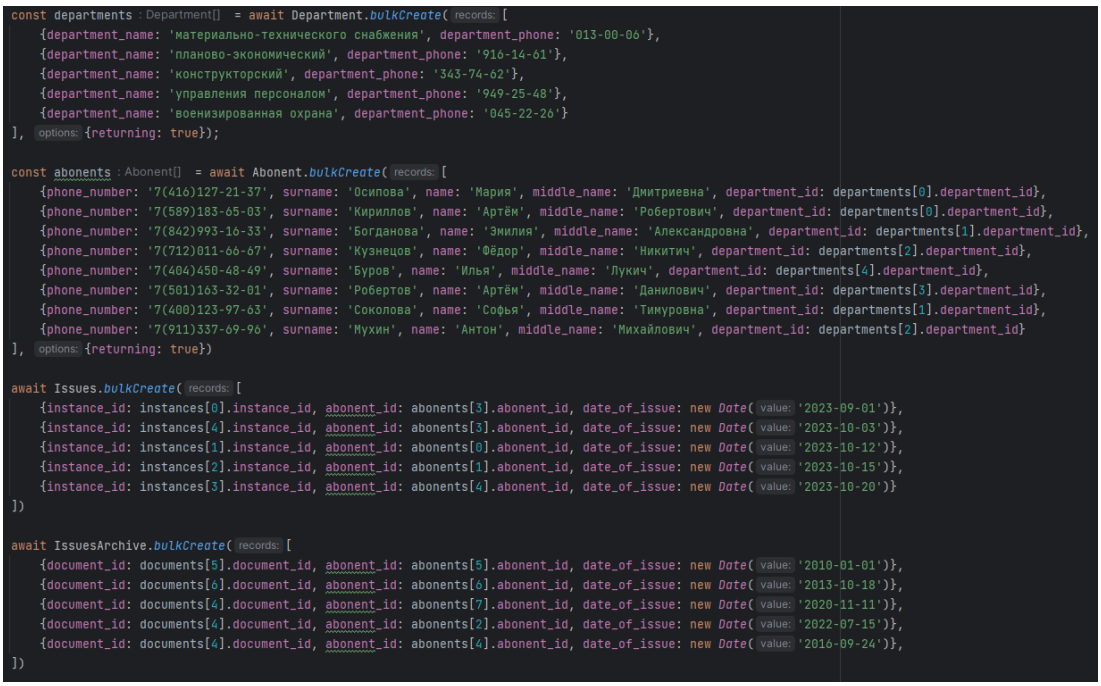
Рисунок 9 - модель "Department"

```
const shelfs : Shelf[]  = await Shelf.bulkCreate( records: [
    {shelf_unique_number: 1789131},
    {shelf_unique_number: 2124789},
    {shelf_unique_number: 3247893},
    {shelf_unique_number: 4489534},
    {shelf_unique_number: 5526783},
    {shelf_unique_number: 6890146}
], options: {returning: true});

const racks : Rack[]  = await Rack.bulkCreate( records: [
    {shelf_id: shelfs[0].shelf_id, rack_unique_number: 1247890},
    {shelf_id: shelfs[0].shelf_id, rack_unique_number: 1589161},
    {shelf_id: shelfs[1].shelf_id, rack_unique_number: 2781313},
    {shelf_id: shelfs[1].shelf_id, rack_unique_number: 2894131},
    {shelf_id: shelfs[2].shelf_id, rack_unique_number: 3471123},
    {shelf_id: shelfs[2].shelf_id, rack_unique_number: 3857841},
    {shelf_id: shelfs[3].shelf_id, rack_unique_number: 4938732},
    {shelf_id: shelfs[3].shelf_id, rack_unique_number: 4538713},
    {shelf_id: shelfs[4].shelf_id, rack_unique_number: 5982412},
    {shelf_id: shelfs[4].shelf_id, rack_unique_number: 5001378},
    {shelf_id: shelfs[5].shelf_id, rack_unique_number: 6982412},
    {shelf_id: shelfs[5].shelf_id, rack_unique_number: 6001378}
], options: {returning: true});
```

Рисунок 10 - заполнение моделей, часть 1

```
const cells : Cell[]  = await Cell.bulkCreate( records: [
    {rack_id: racks[0].rack_id, cell_unique_number: 1189313},
    {rack_id: racks[0].rack_id, cell_unique_number: 1289431},
    {rack_id: racks[1].rack_id, cell_unique_number: 2178941},
    {rack_id: racks[1].rack_id, cell_unique_number: 2278914},
    {rack_id: racks[2].rack_id, cell_unique_number: 3178913},
    {rack_id: racks[2].rack_id, cell_unique_number: 3289234},
    {rack_id: racks[3].rack_id, cell_unique_number: 4138732},
    {rack_id: racks[4].rack_id, cell_unique_number: 5182412},
    {rack_id: racks[5].rack_id, cell_unique_number: 6182412},
    {rack_id: racks[6].rack_id, cell_unique_number: 7178123},
    {rack_id: racks[7].rack_id, cell_unique_number: 8189723},
    {rack_id: racks[8].rack_id, cell_unique_number: 9136781},
    {rack_id: racks[9].rack_id, cell_unique_number: 8901331},
    {rack_id: racks[10].rack_id, cell_unique_number: 1678921},
    {rack_id: racks[11].rack_id, cell_unique_number: 8913124}
], options: {returning: true});

const documents : Document[]  = await Document.bulkCreate( records: [
    {theme_name: 'учет инвентаря', document_title: 'Учтенный инвентарь за 2023', inventory_number: '01000000012023', cell_id: cells[0].cell_id},
    {theme_name: 'документация', document_title: 'Правила инвентаризации', inventory_number: '01000100012023', cell_id: cells[3].cell_id},
    {theme_name: 'отчеты экономистов', document_title: 'План экономического развития на 2024', inventory_number: '02000020242023', cell_id: cells[4].cell_id},
    {theme_name: 'налоговая', document_title: 'Результаты налоговой инспекции за 2022', inventory_number: '02000011112022', cell_id: cells[6].cell_id},
    {theme_name: 'документация', document_title: 'Комплектная документация на станки от Bosh', inventory_number: '03000019982019', cell_id: cells[7].cell_id},
    {theme_name: 'история предприятия', document_title: 'История предприятия с момента октябрьской революции', inventory_number: '07108104522010', cell_id: cells[8].cell_id},
    {theme_name: 'охрана труда', document_title: 'Политика организации по охране труда', inventory_number: '08000035642010', cell_id: cells[9].cell_id}
], options: {returning: true});

const instances : Instance[]  = await Instance.bulkCreate( records: [
    {title: 'Учтенные станки', inventory_number: '000000010001', document_id: documents[0].document_id},
    {title: 'Правила инвентаризации экз.1', inventory_number: '000100010001', document_id:  documents[1].document_id},
    {title: 'Правила инвентаризации экз.2', inventory_number: '000100010002', document_id:  documents[1].document_id},
    {title: 'Новый экономический план на предприятии', inventory_number: '000020240001', document_id:  documents[2].document_id},
    {title: 'Документация на станок BOSH W313ADC31', inventory_number: '000019980001', document_id:  documents[4].document_id}
], options: {returning: true});
```

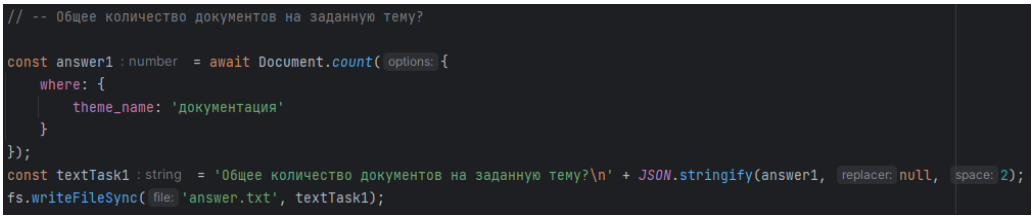
Рисунок 11 - заполнение моделей, часть 2



```
const departments : Department[] = await Department.bulkCreate( records: [
    {department_name: 'материально-технического снабжения', department_phone: '013-00-06'},
    {department_name: 'планово-экономический', department_phone: '916-14-61'},
    {department_name: 'конструкторский', department_phone: '343-74-62'},
    {department_name: 'управления персоналом', department_phone: '949-25-48'},
    {department_name: 'военизированная охрана', department_phone: '045-22-26'}
], options: {returning: true});

const abonents : Abonent[] = await Abonent.bulkCreate( records: [
    {phone_number: '7(416)127-21-37', surname: 'Осипова', name: 'Мария', middle_name: 'Дмитриевна', department_id: departments[0].department_id},
    {phone_number: '7(589)183-65-03', surname: 'Кириллов', name: 'Артём', middle_name: 'Робертович', department_id: departments[0].department_id},
    {phone_number: '7(842)993-16-33', surname: 'Богданова', name: 'Эмилия', middle_name: 'Александровна', department_id: departments[1].department_id},
    {phone_number: '7(712)011-66-67', surname: 'Кузнецов', name: 'Фёдор', middle_name: 'Никитич', department_id: departments[2].department_id},
    {phone_number: '7(404)450-48-49', surname: 'Буров', name: 'Илья', middle_name: 'Лукич', department_id: departments[4].department_id},
    {phone_number: '7(501)163-32-01', surname: 'Робертов', name: 'Артём', middle_name: 'Данилович', department_id: departments[3].department_id},
    {phone_number: '7(400)123-97-63', surname: 'Соколова', name: 'Софья', middle_name: 'Тимуровна', department_id: departments[1].department_id},
    {phone_number: '7(911)337-69-96', surname: 'Мухин', name: 'Антон', middle_name: 'Михайлович', department_id: departments[2].department_id}
], options: {returning: true})

await Issues.bulkCreate( records: [
    {instance_id: instances[0].instance_id, abonent_id: abonents[3].abonent_id, date_of_issue: new Date( value: '2023-09-01')},
    {instance_id: instances[4].instance_id, abonent_id: abonents[3].abonent_id, date_of_issue: new Date( value: '2023-10-03')},
    {instance_id: instances[1].instance_id, abonent_id: abonents[0].abonent_id, date_of_issue: new Date( value: '2023-10-12')},
    {instance_id: instances[2].instance_id, abonent_id: abonents[1].abonent_id, date_of_issue: new Date( value: '2023-10-15')},
    {instance_id: instances[3].instance_id, abonent_id: abonents[4].abonent_id, date_of_issue: new Date( value: '2023-10-20')}
])

await IssuesArchive.bulkCreate( records: [
    {document_id: documents[5].document_id, abonent_id: abonents[5].abonent_id, date_of_issue: new Date( value: '2010-01-01')},
    {document_id: documents[6].document_id, abonent_id: abonents[6].abonent_id, date_of_issue: new Date( value: '2013-10-18')},
    {document_id: documents[4].document_id, abonent_id: abonents[7].abonent_id, date_of_issue: new Date( value: '2020-11-11')},
    {document_id: documents[4].document_id, abonent_id: abonents[2].abonent_id, date_of_issue: new Date( value: '2022-07-15')},
    {document_id: documents[4].document_id, abonent_id: abonents[4].abonent_id, date_of_issue: new Date( value: '2016-09-24')},
])
```

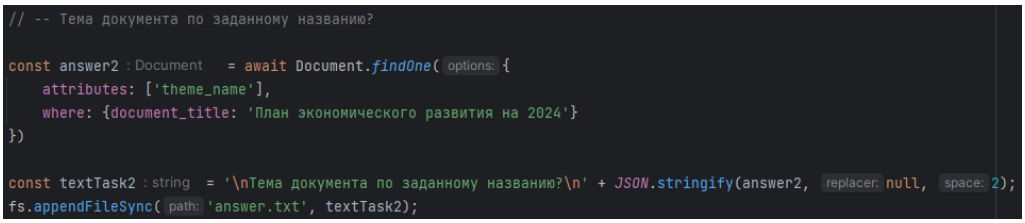
Рисунок 12 - заполнение моделей, часть 3

Написаны запросы для ответа на вопросы в задании. Результаты приведены на рисунках 13-19.



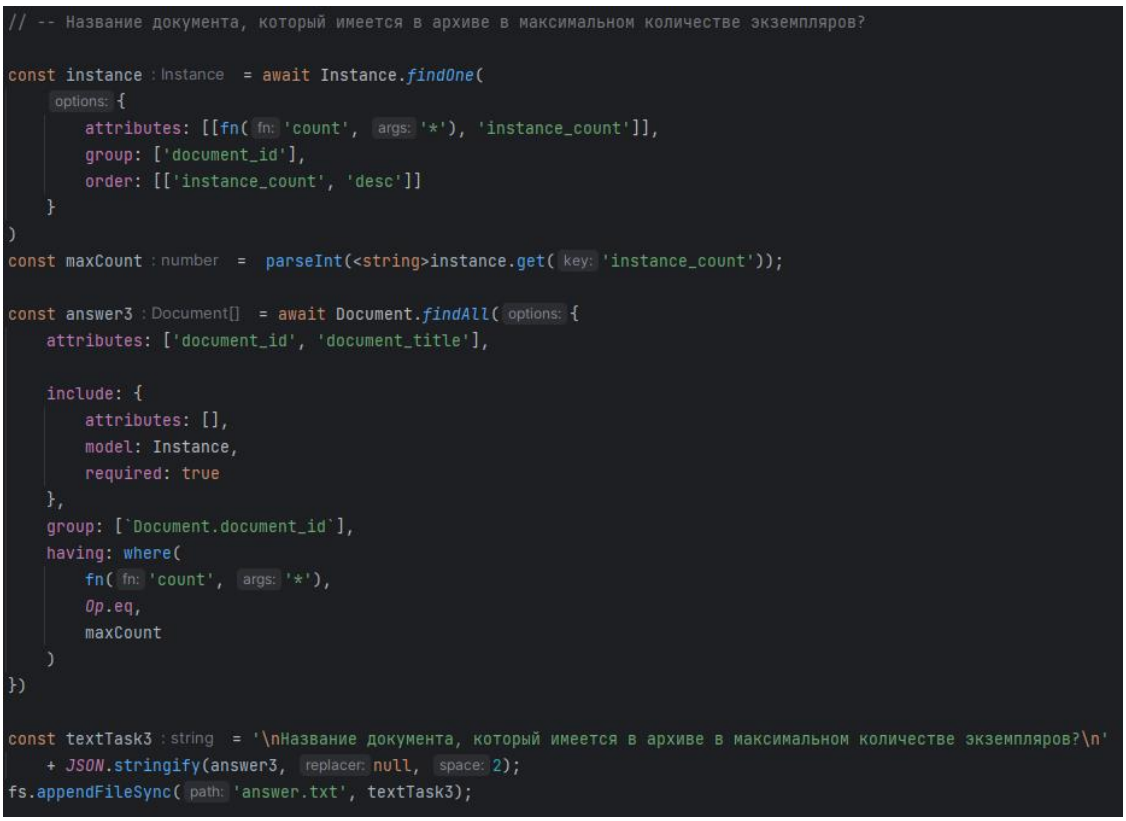
```
// -- Общее количество документов на заданную тему?

const answer1 : number = await Document.count( options: {
    where: {
        theme_name: 'документация'
    }
});
const textTask1 : string = 'Общее количество документов на заданную тему?\n' + JSON.stringify(answer1, replacer: null, space: 2);
fs.writeFileSync( file: 'answer.txt', textTask1);
```

Рисунок 13 - ответы на вопросы задания, часть 1



```
// -- Тема документа по заданному названию?

const answer2 : Document = await Document.findOne( options: {
    attributes: ['theme_name'],
    where: {document_title: 'План экономического развития на 2024'}
})

const textTask2 : string = '\nТема документа по заданному названию?\n' + JSON.stringify(answer2, replacer: null, space: 2);
fs.appendFileSync( path: 'answer.txt', textTask2);
```

Рисунок 14 - ответы на вопросы задания, часть 2



```
// -- Название документа, который имеется в архиве в максимальном количестве экземпляров?

const instance : Instance  = await Instance.findOne(
    options: {
        attributes: [[fn( fn: 'count',  args: '*'), 'instance_count']],
        group: ['document_id'],
        order: [['instance_count', 'desc']]
    }
)
const maxCount : number  =  parseInt(<string>instance.get( key: 'instance_count'));

const answer3 : Document[]  = await Document.findAll( options: {
    attributes: ['document_id', 'document_title'],

    include: {
        attributes: [],
        model: Instance,
        required: true
    },
    group: [`Document.document_id`],
    having: where(
        fn( fn: 'count',  args: '*'),
        Op.eq,
        maxCount
    )
})

const textTask3 : string  = '\nНазвание документа, который имеется в архиве в максимальном количестве экземпляров?\n'
    + JSON.stringify(answer3,  replacer: null,  space: 2);
fs.appendFileSync( path: 'answer.txt', textTask3);
```

Рисунок 15 - ответы на вопросы задания, часть 3

```
// -- Фамилия, имя и отчество абонента, который брал указанный документ последним?
const documentName : "Политика организации по охран... = 'Политика организации по охране труда'
const resFromArchive : IssuesArchive  = await IssuesArchive.findOne( options: {
    attributes: ['date_of_issue'],
    include: [
        {
            attributes: ['document_title'],
            model: Document,
            where: {
                document_title: documentName
            },
            required: true
        },
        {
            attributes: ['name', 'surname', 'middle_name'],
            model: Abonent,
            required: true
        }
    ],
    order: [['date_of_issue', 'DESC']]
})
const resFromIssues : Issues  = await Issues.findOne( options: {
    attributes: ['date_of_issue'],
    include: [
        {
            model: Instance,
            required: true,
            include: [{
                attributes: ['document_title'],
                model: Document,
                where: {
                    document_title: documentName
                }
            }]
        },
        {
            attributes: ['name', 'surname', 'middle_name'],
            model: Abonent,
            required: true
```

Рисунок 16 - ответы на вопросы задания, часть 4.1

```
            required: true
        }
    ],
    order: [['date_of_issue', 'DESC']]
})
const textTask4 : "Фамилия, имя и отчество абон...  = '\nФамилия, имя и отчество абонента, который брал указанный документ последним?\n';
let answer4: string;
if (!resFromArchive && !resFromIssues)
    answer4 = 'Документ не брали';
else if (resFromArchive && !resFromIssues)
    answer4 = resFromArchive.abonent.fullName
else if (!resFromArchive && resFromIssues)
    answer4 = resFromIssues.abonent.fullName
else if (resFromIssues.date_of_issue > resFromArchive.date_of_issue)
    answer4 = resFromIssues.abonent.fullName
else
    answer4 = resFromArchive.abonent.fullName

fs.appendFileSync( path: 'answer.txt',  data: textTask4 + answer4);
```

Рисунок 17 - ответы на вопросы задания, часть 4.2

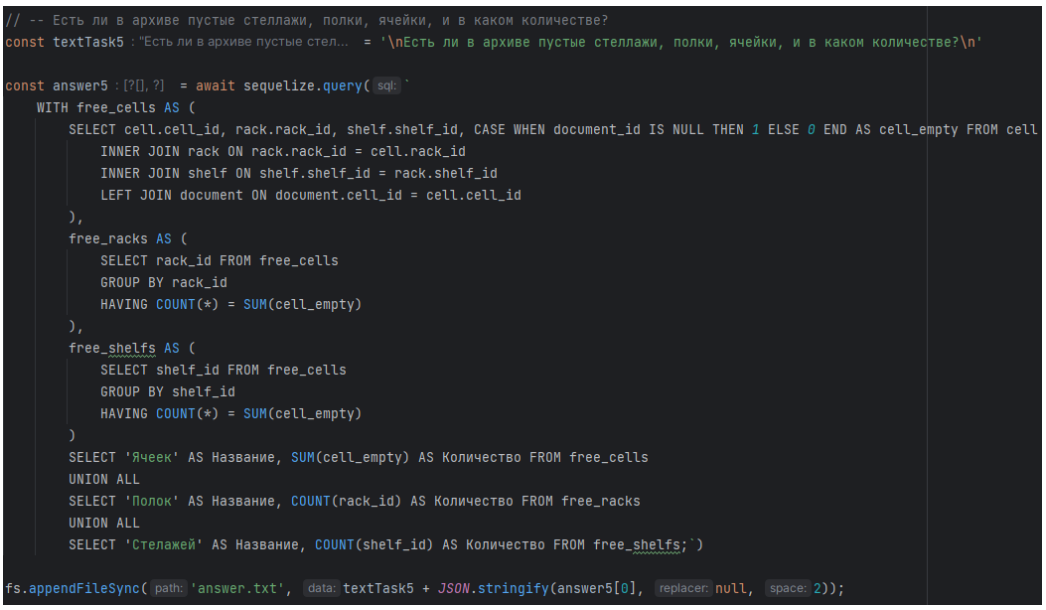


Рисунок 18 - ответы на вопросы задания, часть 5

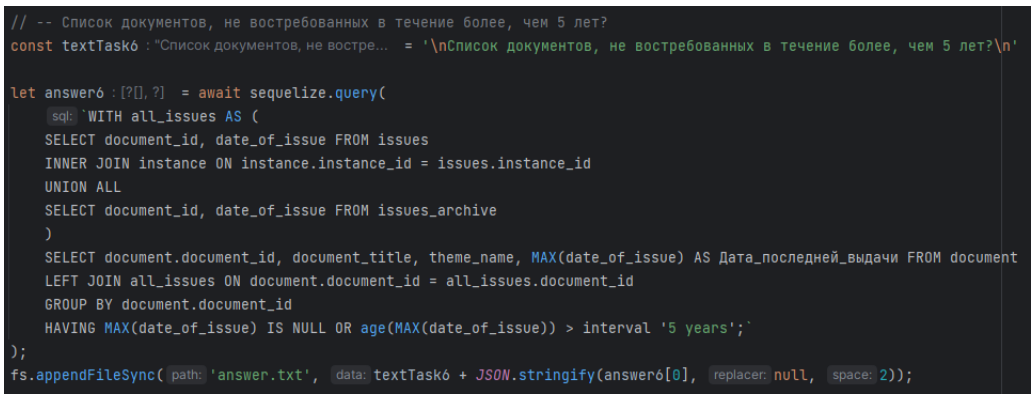


Рисунок 19 - ответы на вопросы задания, часть 6

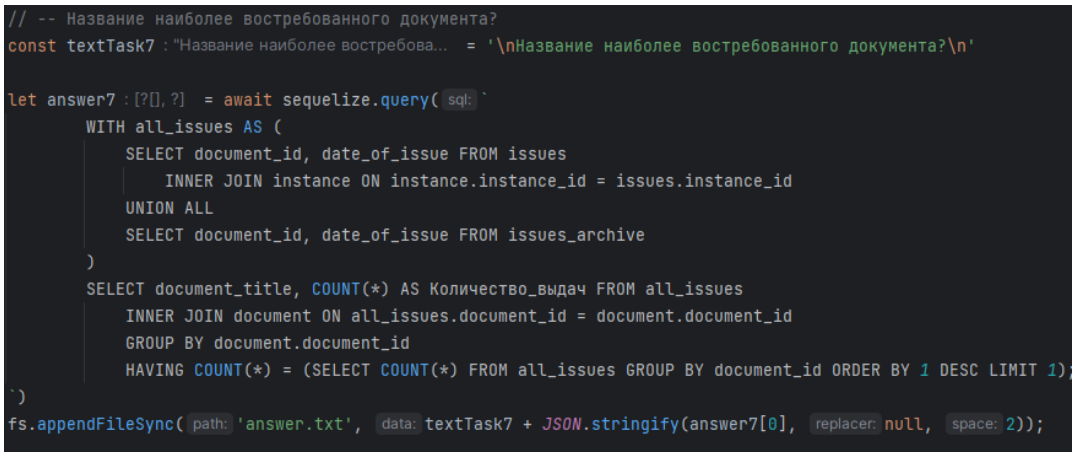


Рисунок 20 - ответы на вопросы задания, часть 7

Для запуска необходимо:

- Создать базу данных в PostgreSQL “Archive”.

- Установить необходимые пакеты командой *npm i*

- Запустить скрипт *start* из *package.json* командой *npm start*

Вывод.

Развернут *Sequalize*, написаны запросы для создания и заполнения таблиц в соответствии со структурой БД, написаны запросы к БД с использованием ORM, отвечающие на вопросы в задания.

ПРИЛОЖЕНИЕ А

Ссылка на PR: https://github.com/moevm/sql-2023-1303/pull/48