

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студенты: ---

Группа: ---

Тема работы: Разработка графического интерфейса для работы со спецификациями изделий

Исходные данные:

Для разработанного в практической работе №2.1 «каркаса» для работы со спецификациями необходимо создать графический интерфейс, который позволит выполнять все соответствующие функции. Необходимо обеспечить: ведение справочника изделий различных типов, используемых в составе изделий; формирование строк спецификаций с указанием норм расхода; поиск всех строк спецификации изделия на всю глубину вложенности; расчёт сводных норм расхода компонентов изделия по ресурсам заданного класса.

Содержание пояснительной записки:

«Содержание», «Введение», «Требования», «Основные диаграммы», «Используемый инструментарий», «Основные SQL-процедуры», «Примеры работы графического интерфейса», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки: не менее 20 страниц.

Студенты _____ ---

Преподаватель _____ ---

АННОТАЦИЯ

В данной курсовой работе был разработан графический интерфейс для работы со спецификациями изделий. В качестве предметной области был взят магазин «Askona», который специализируется на создании ортопедических матрацев и товаров для сна, включая мебель. В частности, рассматриваются диваны и кресла выбранной компании. С помощью графического интерфейса пользователи могут взаимодействовать с информационной системой. Ответственному за справочник доступна работа с классификатором изделий, справочником изделий и справочником параметров изделий. Конструктор/технолог может работать также и с материальной спецификацией изделий, включая её редактирование, расчёт сводных норм расхода материальных ресурсов и управление изменениями.

SUMMARY

In this coursework a graphical interface for working with product specifications was developed. The Askona shop, which specialises in creating orthopaedic mattresses and sleep products, including furniture, has been taken as the subject area. In particular, sofas and armchairs of the selected company are considered. Using a graphical interface, users can interact with the information system. The person responsible for the directory can work with the product classifier, the product directory and the product parameter directory. The designer/technologist can also work with the material specification of the products, including its editing, calculation of material consumption rates and change management.

СОДЕРЖАНИЕ

Введение	5
1. Требования	6
1.1. Функциональные требования	6
1.2. Содержание работы	8
2. Основные диаграммы	9
2.1. Диаграмма вариантов использования	9
2.2. Диаграмма классов	10
2.3. Диаграмма «сущность-связь»	11
3. Используемый инструментарий	13
4. Основные SQL-процедуры	14
5. Примеры работы графического интерфейса	15
5.1. Запуск программы	15
5.2. Основное окно	17
5.3. Роль пользователя: «Ответственный за справочник»	17
5.3.1. Раздел «Классы изделий»	18
5.3.2. Раздел «Изделия»	24
5.3.3. Раздел «Единицы измерения»	27
5.4. Роль пользователя: «Технолог / Конструктор»	30
5.4.1. Раздел «Спецификации»	30
Заключение	36
Список использованных источников	37
Приложение А. Листинг кода программы	38

ВВЕДЕНИЕ

Материальная спецификация изделий является неотъемлемой частью данных об изделии. Они требуются для материального обеспечения изготовления, закупок, хранения. Объем этих данных определяется сложностью изделий и их разнообразием. Материальная спецификация может содержать до нескольких миллионов компонентов. Состав компонентов и нормы их расхода непрерывно изменяются при модификации изделий и технологии их изготовления. В этой связи автоматизация процессов работы с материальными спецификациями изделий, несомненно, актуальна.

Целью нашей работы является создание графического интерфейса для работы со спецификациями изделий.

Для этого нам необходимо:

- Выбрать инструментарий для реализации;
- Написать соответствующие требованиям SQL-процедуры и связать их с графическим интерфейсом;
- Протестировать реализованную программу.

1. ТРЕБОВАНИЯ

1.1. Функциональные требования













В работе необходимо разработать проект каркаса для работы со Спецификациями изделий. Для этого выдвигаются следующие функциональные требования:

- Обеспечить:
 - Ведение справочника изделий различных типов, используемых в составе изделий;
 - Формирование строк спецификаций с указанием норм расхода;
 - Поиск всех строк спецификации изделия на всю глубину вложенности;
 - Расчёт сводных норм расхода компонентов изделия по ресурсам заданного класса.










Для работы было принято решение выбрать в качестве предметной области магазин товаров для сна «*Askona*», а в частности диваны и кресла данной компании. Сайт: <https://www.askona.ru>. Пример состава сборки изделия «Диван-кровать серии Кларк» представлен на рис. 1:

Комплектовочная ведомость

Центральная секция

Упаковка	Деталь	Артикул	Наименование	Количество
Центральная секция		Д37	Центральная секция	1
		Д04	Спинка откидная	1
		Д33	Подушка приспинная	1
			Пакет с фурнитурой	1
Пакет с фурнитурой		ДФ98	Болт М8*30 DIN 933	2
		ДФ82	Шайба 24x8 (пнд)	10
		ДФ42	Шайба 8 ув. DIN 9021	2
		ДФ17	Гайка М8 самоконтр DIN 985	2
		ДФ24	Колпачок болта М8	2
		ДФ114	Гайка М8 ERICSON	2
		ДФ53	Болт М8*40 DIN 933	2
		ДФ113	Ключ шестигранный №5	1

Секция канапе

Упаковка	Деталь	Артикул	Наименование	Количество
Секция канапе		Д25	Секция канапе	1
			Пакет с фурнитурой	1
		Д33	Подушка приспинная	1
		Д04	Спинка откидная	1
Пакет с фурнитурой		ДФ98	Болт М8*30 DIN 933	2
		ДФ82	Шайба 24x8 (пнд)	10
		ДФ42	Шайба 8 ув. DIN 9021	2
		ДФ17	Гайка М8 самоконтр DIN 985	2
		ДФ24	Колпачок болта М8	2
		ДФ114	Гайка М8 ERICSON	2
		ДФ53	Болт М8*40 DIN 933	2
		ДФ113	Ключ шестигранный №5	1

Подлокотник

Упаковка	Деталь	Артикул	Наименование	Количество
Подлокотник		Д06	Подлокотник	2
			Пакет с фурнитурой	1
Пакет с фурнитурой		ДФ101	Болт М8*60 DIN 933	8
		ДФ106	Проставка пластик 20*10	3
		ДФ103	Проставка пластик 20*15	8
		ДФ42	Шайба М8 DIN 9021	8

Комплектовочная ведомость дивана

			Инструкция по сборке Паспорт	1
--	---	--	---------------------------------	---

*варианты подлокотников:

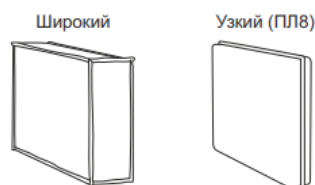


Рисунок 1 – Состав сборки дивана-кровати серии Кларк

1.2. Содержание работы

Работа состоит из следующих этапов:

- 1.** На основании реализованной работы №2.1 разработать и описать необходимый набор SQL-процедур, обеспечивающий поддержку описанных требований.
- 2.** Обеспечить связь разработанных SQL-процедур с графическим интерфейсом.
- 3.** Выполнить тестирование работоспособности реализованного курсового проекта.
- 4.** Привести примеры работы с графическим интерфейсом.
- 5.** Оформить отчёт по курсовой работе и продемонстрировать работу.

2. ОСНОВНЫЕ ДИАГРАММЫ

2.1. Диаграмма вариантов использования

Всего с моделью *Материальной спецификации изделий* могут работать три типа пользователей:

- Ответственный за справочник;
- Конструктор;
- Технолог.

Ответственный за справочник может вести классификатор изделий, справочник изделий и справочник параметров (добавление, удаление, редактирование).

Пользователи справочника (конструктор, технолог) могут вести материальную спецификацию изделий, включающую в себя редактирование спецификаций, управление изменениями, расчёт сводных норм расхода материальных ресурсов, а также ведение справочника параметров и справочника изделий, включающего ведение классификатора.

На рис. 2.1 представлена диаграмма вариантов использования, которая была разработана в работе №2.1 в рамках рассматриваемой предметной области:

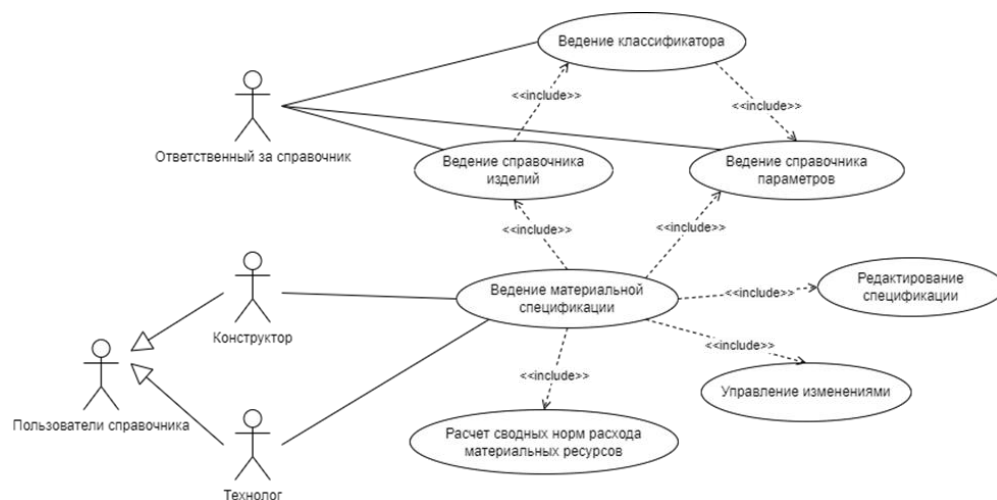


Рисунок 2.1 – Диаграмма вариантов использования «Материальная спецификация изделий»

2.2. Диаграмма классов

На рис. 2.2 представлена разработанная диаграмма классов:

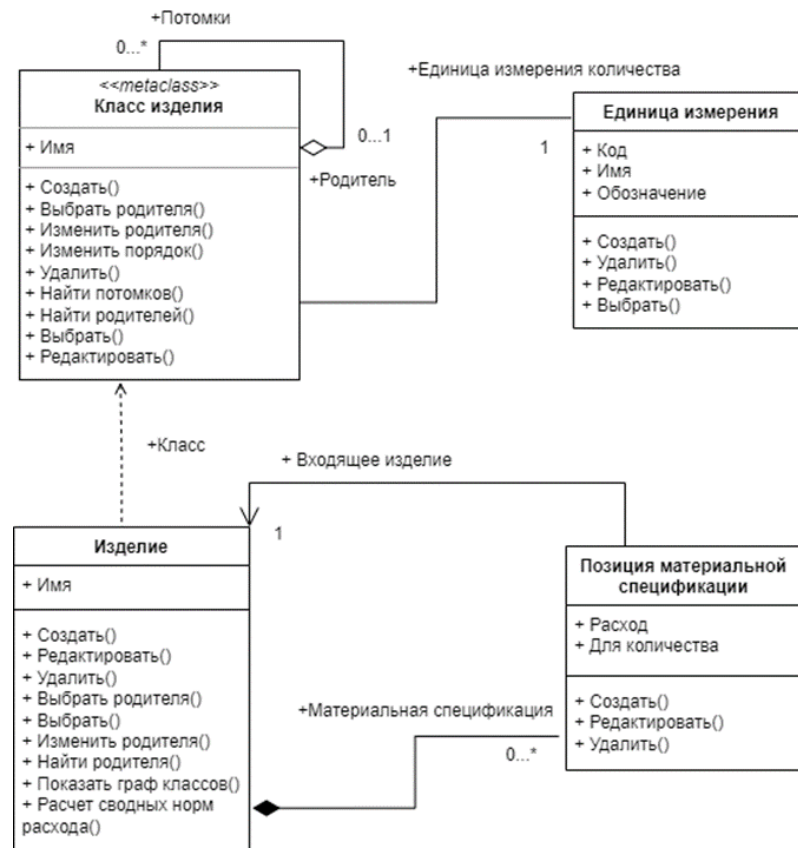


Рисунок 2.2 – Диаграмма классов для прецедента *Управление спецификацией изделий (Bill of materials)*

Изделие может включать в себя некоторую *позицию материальной спецификации*. Она представляет собой класс с атрибутами *расход* и *для количества*. Из методов выделяются создание, редактирование и удаление.

Класс «Изделие» имеет атрибут *имя*. Изделия можно *создавать*, *редактировать*, *удалять*, *выбирать*, *выбирать их родителей*, *изменять их родителей*, *находить их родителей*, *показывать граф классов для них*, *рассчитывать сводные нормы расхода*.

Классификатор изделий («Класс изделия») является *метаклассом* с атрибутом *имя*. Для него доступны такие методы, как *создание*, *выбор родителя*, *изменение родителя*, *изменение порядка*, *удаление*, *поиск потомков*, *поиск*

родителей, выбор, редактирование. У классов могут быть потомки или родители.

Также имеется класс «Единица измерения» количества изделий, включающая атрибуты *код*, *имя* и *обозначение*. Для него доступно *создание*, *удаление*, *редактирование* и *выбор*.

2.3. Диаграмма «сущность-связь»

На рис. 2.3 представлена диаграмма «сущность-связь»:

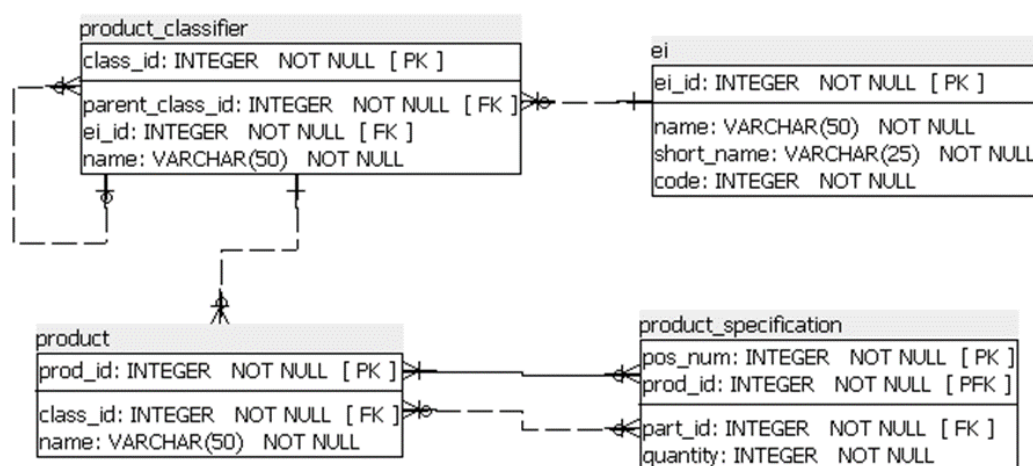


Рисунок 2.3 – Диаграмма «сущность-связь» для проекта *Материальная спецификация изделий*

Сущность **product_classifier** позволяет хранить данные о структуре классификатора. Поле **ei_id** позволяет указать единицу измерения количества изделия. Поле **parent_class_id** позволяет описать дерево классификации изделий.

Сущность **product** позволяет вести каталог изделий. Поле **class_id** позволяет указать терминальный класс изделия. Поле **name** позволяет указать имя изделия.

Сущность **ei** позволяет вести список единиц измерения количества. Поля **short_name**, **name** и **code** позволяют указать обозначение, имя и код ЕИ соответственно.

Сущность ***product_specification*** позволяет вести материальную спецификацию продуктов. Поля ***pos_num***, и ***quantity*** позволяют указать номер позиции спецификации и необходимое количество изделия соответственно. Поле ***prod_id*** связывает спецификацию с продуктом и позволяет указать продукт, спецификация которого описывается. Поле *part_id* позволяет узнать, какое изделие входит в состав того, что указано в ***prod_id***.

3. ИСПОЛЬЗУЕМЫЙ ИНСТРУМЕНТАРИЙ

При разработке был использован следующий инструментарий:

1. Операционная система – **Windows 11**.
2. Среда построения диаграмм – **SQL Power Architect** и онлайн-сервис **draw.io** (<https://www.drawio.com>).
3. Среда для написания кода – **Visual Studio Code** (1.85.1).
4. СУБД – **PostgreSQL 15**.
5. Система для администрирования СУБД – **pgAdmin 4**.
6. Создание backend-части (для запуска и работы сервера) – **Node.js** (v20.10.0) и **npm** (10.2.3).
7. Минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений – **Express** (4.18.2).
8. Язык шаблонов, который позволяет генерировать HTML-разметку с помощью простого JavaScript – **EJS** (3.1.9).
9. Набор модулей node.js для взаимодействия с базой данных PostgreSQL – **node-postgres** (8.11.3).
10. Фреймворк для frontend-разработки веб-приложений – **Bootstrap** (5.3.2) и **bootstrap-icons** (1.11.2)

4. ОСНОВНЫЕ SQL-ПРОЦЕДУРЫ

Все SQL-процедуры из работы №2.1 были перенесены в код программы. В файле *creation.js*, который представлен в *Приложении А* происходит создание начальных таблиц для БД, создание необходимых функций, а также заполнение базы данных начальными значениями (см. листинг А.1).

Все функции применяются к базе данных при помощи запросов на выборку SELECT. Эти запросы используются в функциях, реализованных на языке программирования JavaScript. Полный код представлен в файле *queries.js* (см. листинг А.2).

Взаимодействие с сервером Node.js осуществляется с помощью методов `get` и `post` (получение и отправка данных соответственно). Реализация представлена в файле *index.js* (см. листинг А.3).

5. ПРИМЕРЫ РАБОТЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

5.1. Запуск программы

Для того, чтобы начать работать с программой, необходимо чтобы у пользователя были установлены Node.js и npm, версии которых указаны в разделе 3. Также, необходимо создать и подключить базу данных, которая должна иметь название ***mispris***. Например, можно запустить pgAdmin и с помощью него создать новую пустую базу данных ***mispris*** (см. пример на рис. 3.1).

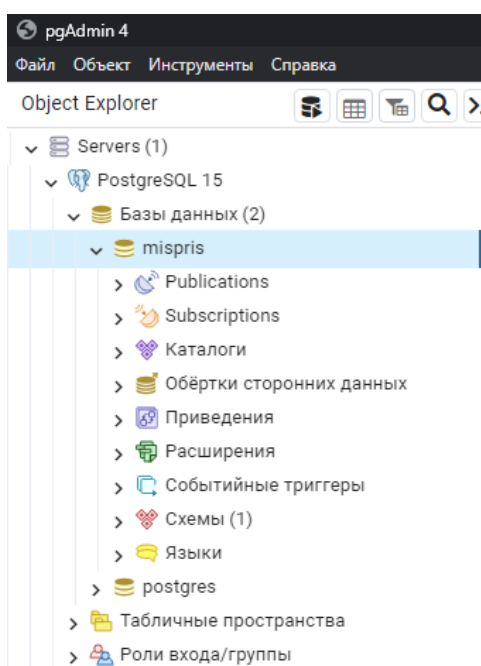


Рисунок 3.1 – Пример создания и подключения базы данных mispris

Управлять параметрами подключения (например, изменить пароль, имя базы данных) можно с помощью файла ***pool.js*** (см. листинг А.4).

Через терминал необходимо перейти к папке CourseWork и последовательно набрать команды `npm install` и `npm start` (см. рис. 3.2). При применении первой команды в CourseWork должна появиться новая папка – `node_modules`, содержащая все необходимые модули для работы программы в браузере.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\tanya\Downloads\CourseWork> npm install

added 97 packages, and audited 98 packages in 12s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\tanya\Downloads\CourseWork> npm start
```

Рисунок 3.2 – Установка модулей и запуск программы

При удачном запуске в терминале появятся следующие сообщения (см. рис. 3.3):

```
PS C:\Users\tanya\Downloads\CourseWork> npm start

> coursework@1.0.0 start
> node index.js

Подключение к Базе Данных прошло успешно!
Server started 8081.
Все 4 таблицы и функции успешно созданы!
Данные успешно вставлены!
```

Рисунок 3.3 – Удачный запуск программы

Как можно заметить, сервер был запущен на локальном хосте в порте 8081. Соответственно, в используемом браузере необходимо перейти по следующей ссылке: <http://localhost:8081>. В итоге, пользователь окажется на странице, представленной на рис. 3.4:

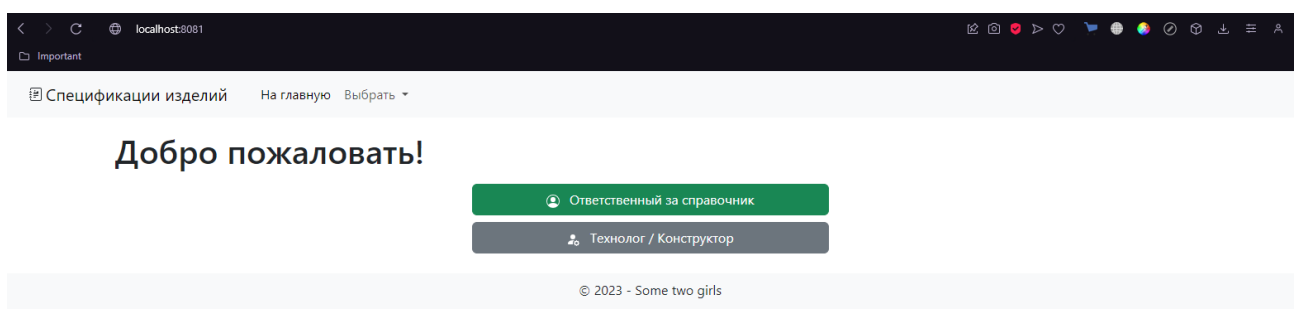


Рисунок 3.4 – Начальный экран программы

5.2. Основное окно

Основное окно программы представлено на рис. 3.4. Здесь пользователю доступно две роли – «*Ответственный за справочник*» и «*Технолог / Конструктор*». При выборе одной из них пользователю выдаётся предупреждение о смене роли.

Навигационная панель содержит две кнопки:

- «*На главную*» - возвращение к основному окну и выбору роли;
- «*Выбор*» - выпадающее меню, которое позволяет переключаться между классами изделий, изделиями, единицами измерений и спецификациями.

5.3. Роль пользователя: «Ответственный за справочник»

На рис. 4.1 представлено то, какое предупреждение выдаётся пользователю при выборе роли «Ответственный за справочник»:

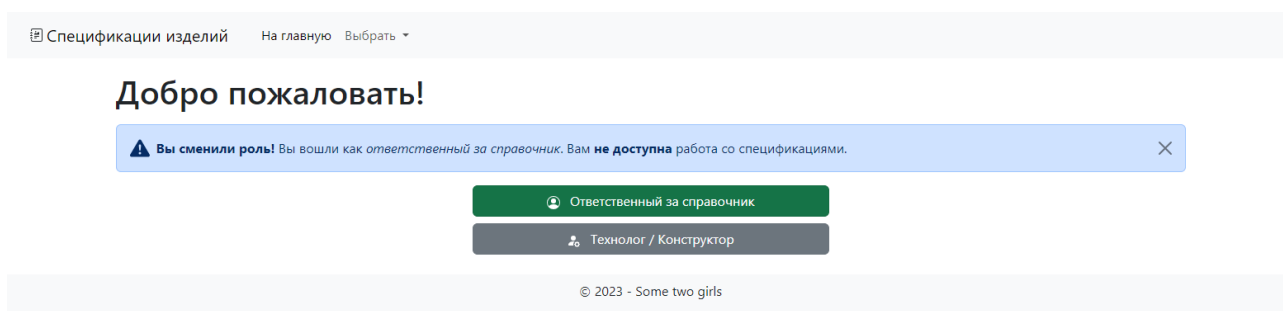


Рисунок 4.1 – Предупреждение о выборе роли «Ответственный за справочник»

При этом ему становится недоступна материальная спецификация изделий (см. рис. 4.2):

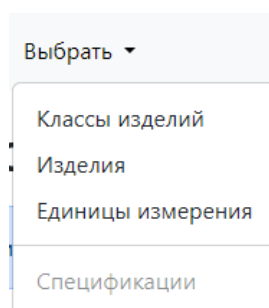
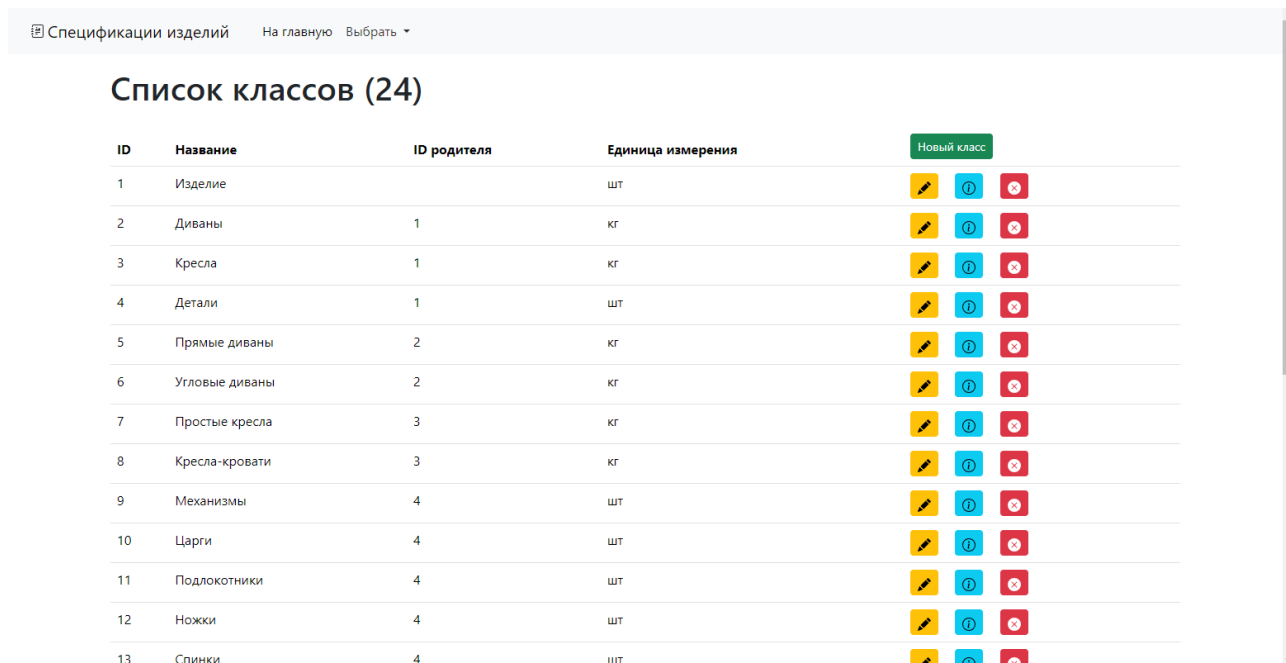


Рисунок 4.2 – Запрет выбора раздела «Спецификации» при роли «Ответственный за справочник»

5.3.1. Раздел «Классы изделий»

Перейдём к работе с классами изделий (см. рис. 4.3). На данной странице можно увидеть название выбранного справочника, а также количество элементов в нём и, непосредственно, сами элементы. Изначально в справочнике есть 24 класса.










































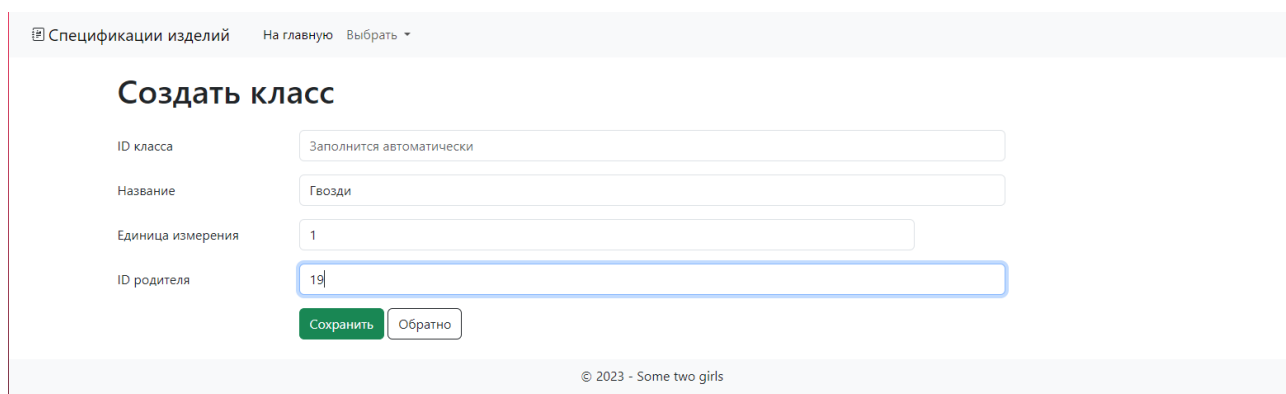
Спецификации изделий				
На главную Выбрать				
Список классов (24)				
ID	Название	ID родителя	Единица измерения	Новый класс
1	Изделие		шт	  
2	Диваны	1	кг	  
3	Кресла	1	кг	  
4	Детали	1	шт	  
5	Прямые диваны	2	кг	  
6	Угловые диваны	2	кг	  
7	Простые кресла	3	кг	  
8	Кресла-кровати	3	кг	  
9	Механизмы	4	шт	  
10	Царги	4	шт	  
11	Подлокотники	4	шт	  
12	Ножки	4	шт	  
13	Спинки	4	шт	  

Рисунок 4.3 – Фрагмент страницы «Классы изделий»

На рис. 4.4 представлен пример добавления нового класса «Гвозди»:



Спецификации изделий

На главную Выбрать

Создать класс

ID класса

Заполнится автоматически

Название

Гвозди

Единица измерения

1

ID родителя

19

Сохранить

Обратно

© 2023 - Some two girls

Рисунок 4.4 – Пример добавления нового класса

Если какое-то из полей заполнено некорректно, то есть в рассматриваемом случае выбран несуществующий ID родителя, то новый класс не сохранится и количество элементов в справочнике классов изделий не изменится.

Все поля как при создании, так и при редактировании в любом из справочников должны быть заполнены. В противном случае будет выведено предупреждение, как показано на рис. 4.5:

The screenshot shows a web application interface for creating a new class. The header includes a logo, the text 'Спецификации изделий', and navigation links 'На главную' and 'Выбрать'. The main heading is 'Создать класс'. Below it are four input fields: 'ID класса' (auto-filled), 'Название' (filled with 'Гвозди'), 'Единица измерения' (filled with 'ID ЕИ, например, 1'), and 'ID родителя' (filled with '19'). A red border highlights the 'ID родителя' field, and a yellow warning icon with the text 'Заполните это поле.' (Fill this field.) is shown next to it. At the bottom are 'Сохранить' (Save) and 'Обратно' (Back) buttons. The footer contains the copyright notice '© 2023 - Some two girls'.






















Рисунок 4.5 – Предупреждение о незаполненном поле при создании или редактировании элемента

Теперь отредактируем созданный класс. Например, изменим название на «Молотковый болт» и ID родителя на 20 (см. рис. 4.6):

The screenshot shows the 'Обновить класс' (Update Class) form. The header is identical to the previous form. The main heading is 'Обновить класс'. Below it are four input fields: 'ID класса' (filled with '25'), 'Название' (filled with 'Молотковый болт'), 'Единица измерения' (filled with '1'), and 'ID родителя' (filled with '20'). The 'ID родителя' field is highlighted with a blue border. At the bottom are 'Обновить' (Update) and 'Обратно' (Back) buttons. The footer contains the copyright notice '© 2023 - Some two girls'.

Рисунок 4.6 – Редактирование названия и ID родителя у класса «Гвозди»

На рис. 4.7 представлен фрагмент таблицы с изменённым классом:

19	Фурнитура	4	шт	  
20	Болты	19	шт	  
21	Шайбы	19	шт	  
22	Винты	19	шт	  
23	Гайки	19	шт	  
24	Ключи	19	шт	  
25	Молотковый болт	20	шт	  


© 2023 - Some two girls

Рисунок 4.7 – Фрагмент таблицы «Классы изделий» с изменённым элементом «Молотковый болт» вместо «Гвозди»

Голубая кнопка информации у каждого класса позволяет перейти к поиску:

- Классов-потомков;
- Классов-потомков и изделий;
- Классов-родителей;
- Изделий этого класса.

Рассмотрим на примере класса «Кресла» (см. рис. 4.8):

 Спецификации изделий
 На главную
 Выбрать ▾

Выбранный класс: Кресла

Найти потомков

Найти всех потомков, включая изделия

Найти родителей

Найти изделия

Обратно

© 2023 - Some two girls

Рисунок 4.8 – Информация о классе «Кресла»

На рис. 4.9 – 4.12 представлены данные, которые выводятся при нажатии соответствующих кнопок в меню информации о выбранном классе:

Спецификации изделий

На главную

Выбрать

ID	Название	ID родителя	Единица измерения
7	Простые кресла	3	кг
8	Кресла-кровати	3	кг

Обратно

© 2023 - Some two girls

Рисунок 4.9 – Потомки класса «Кресла»

Спецификации изделий

На главную

Выбрать

ID класса	Название	ЕИ	ID класса-родителя	ID изделия	Изделие
3	Кресла	кг	1		
7	Простые кресла	кг	3	5	Кресло Муни
7	Простые кресла	кг	3	4	Кресло Хюгге
8	Кресла-кровати	кг	3	8	Кресло-кровать Персей Nova
8	Кресла-кровати	кг	3	1	Кресло-реклайнер Хоган

Обратно

© 2023 - Some two girls

Рисунок 4.10 – Потомки класса «Кресла», включая изделия

Спецификации изделий

На главную

Выбрать

ID	Название	ID родителя	Единица измерения
1	Изделие		шт

Обратно

© 2023 - Some two girls

Рисунок 4.11 – Родители класса «Кресла»

Спецификации изделий

На главную

Выбрать

ID изделия	Изделие	ID класса	Название класса	ID класса-родителя
5	Кресло Муни	7	Простые кресла	3
4	Кресло Хюгге	7	Простые кресла	3
8	Кресло-кровать Персей Nova	8	Кресла-кровати	3
1	Кресло-реклайнер Хоган	8	Кресла-кровати	3

Обратно

© 2023 - Some two girls

Рисунок 4.12 – Изделия класса «Кресла»

Любой класс можно также удалить. При этом можно удалить как один лишь выбранный класс, так и выбранный класс вместе с потомками. Например, попробуем удалить класс «Фурнитура» без потомков (см. рис. 4.13). Он не будет удалён, поскольку имеет несколько классов-потомков. Функция «Удалить»

применима только к классам, у которых нет потомков, чтобы не нарушить целостность данных. Поэтому, попробуем удалить без потомков класс «Молотковый болт» (см. рис. 4.14).

Спецификации изделий

На главную

Выбрать

Удалить?

ID класса

19

Название

Фурнитура

Единица измерения

1

ID родителя

4

Удалить

Удалить с потомками

Обратно

© 2023 - Some two girls

18	Секции/модули	4	шт	<div><div></div><div></div><div></div></div>
19	Фурнитура	4	шт	<div><div></div><div></div><div></div></div>
20	Болты	19	шт	<div><div></div><div></div><div></div></div>
21	Шайбы	19	шт	<div><div></div><div></div><div></div></div>
22	Винты	19	шт	<div><div></div><div></div><div></div></div>
23	Гайки	19	шт	<div><div></div><div></div><div></div></div>
24	Ключи	19	шт	<div><div></div><div></div><div></div></div>

Рисунок 4.13 – Попытка удаление класса «Фурнитура» без потомков и результат попытки в справочнике

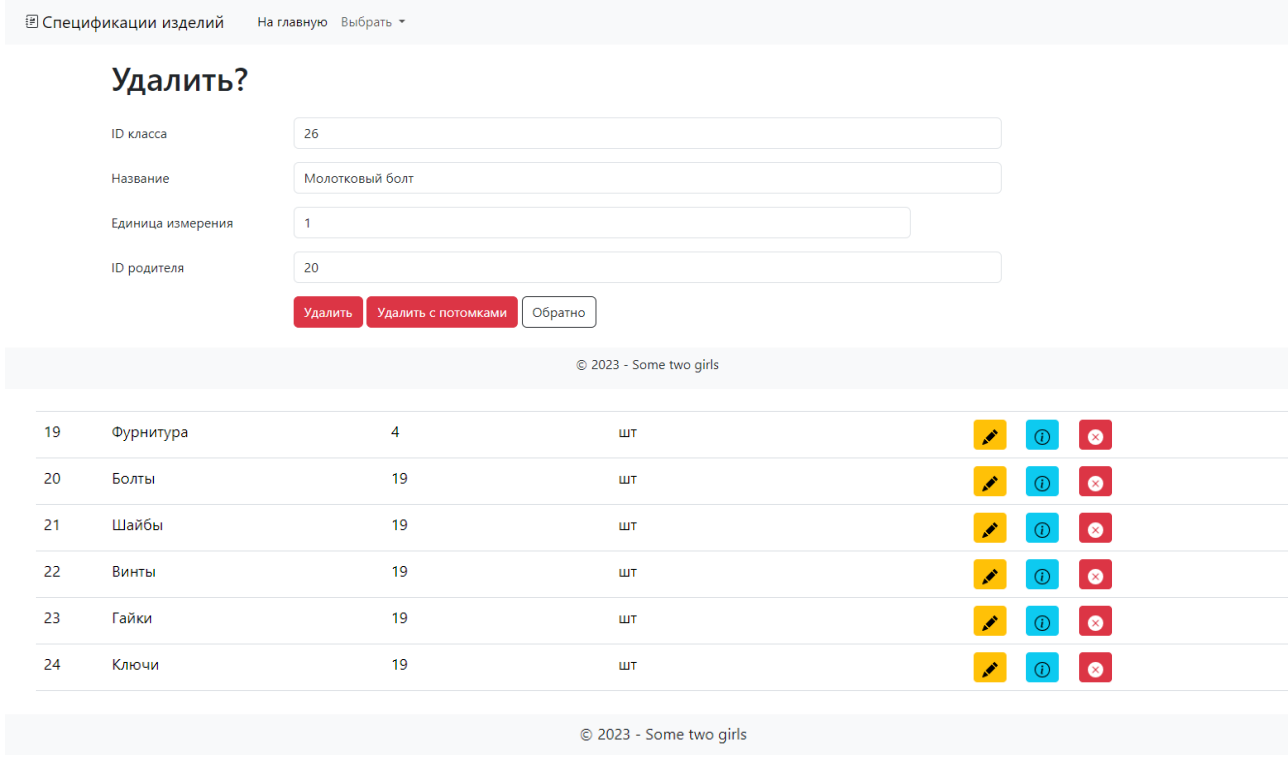






































Рисунок 4.14 – Удаление класса «Молотковый болт» без потомков и результат удаления в справочнике

Теперь нажмём кнопку «Удалить с потомками» у класса «Фурнитура» (см. рис. 4.15) и увидим, что число классов в справочнике сократилось до 18:

7	Простые кресла	3	кг	  
8	Кресла-кровати	3	кг	  
9	Механизмы	4	шт	  
10	Царги	4	шт	  
11	Подлокотники	4	шт	  
12	Ножки	4	шт	  
13	Спинки	4	шт	  
14	Боковины	4	шт	  
15	Подушки	4	шт	  
16	Чехлы	4	шт	  
17	Матрасы	4	шт	  
18	Секции/модули	4	шт	  

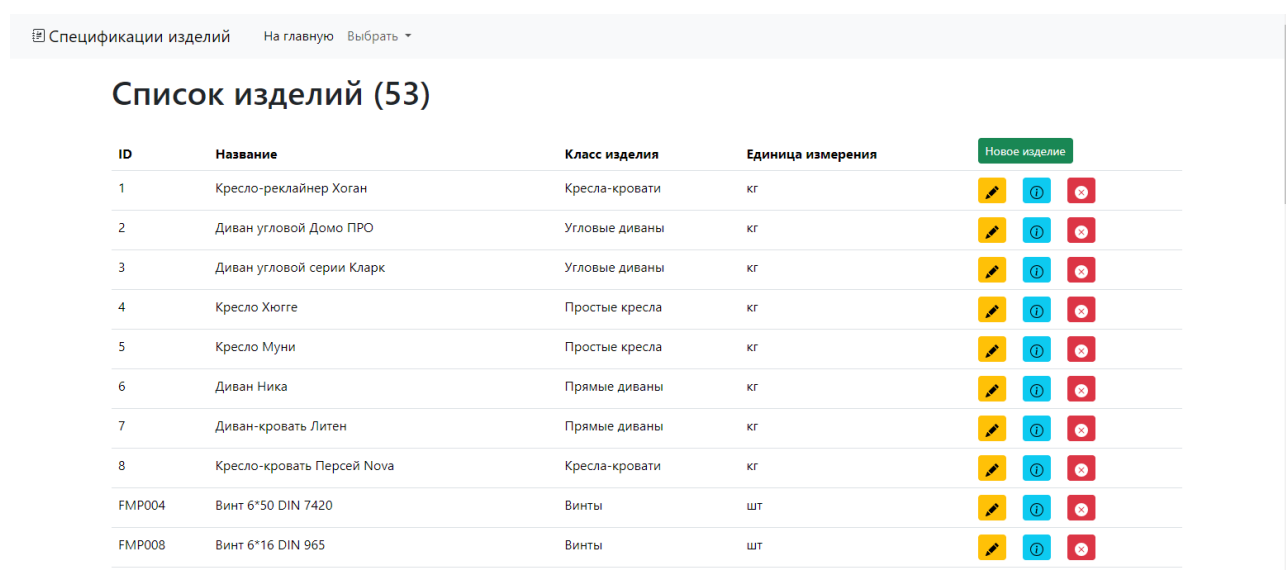
© 2023 - Some two girls

Рисунок 4.15 – Результат удаление класса «Фурнитура» с потомками

5.3.2. Раздел «Изделия»

После манипуляций в пункте 5.3.1 пересоздадим нашу базу данных заново и перезапустим сервер, чтобы вернуть всё в исходное состояние, поскольку с удалением классов удаляются и все изделия этого класса, а также спецификации.

Перейдём в раздел «Изделия» через навигационное меню (см. рис. 4.16). На появившейся странице также можно увидеть название справочника и количество изделий. Всего есть 53 элемента.



Спецификации изделий				На главную	Выбрать ▾
Список изделий (53)					Новое изделие
ID	Название	Класс изделия	Единица измерения		
1	Кресло-реклайнер Хоган	Кресла-кровати	кг		
2	Диван угловой Домо ПРО	Угловые диваны	кг		
3	Диван угловой серии Кларк	Угловые диваны	кг		
4	Кресло Хюгге	Простые кресла	кг		
5	Кресло Муни	Простые кресла	кг		
6	Диван Ника	Прямые диваны	кг		
7	Диван-кровать Литен	Прямые диваны	кг		
8	Кресло-кровать Персей Nova	Кресла-кровати	кг		
FMP004	Винт 6*50 DIN 7420	Винты	шт		
FMP008	Винт 6*16 DIN 965	Винты	шт		

Рисунок 4.16 – Фрагмент раздела «Изделия»

Так же, как и для классов изделий (см. пункт 5.3.1) здесь доступны создание нового изделия, редактирование и удаление существующих изделий. Причём при создании и редактировании, если введено уже существующее ID изделия, то при нажатии на кнопку «Сохранить» произойдёт переход на страницу с изделиями, но новое изделие добавлено не будет. Примеры представлены на рис. 4.17 – 4.19:

Спецификации изделий

На главную

Выбрать

Создать изделие

ID изделия

ДФ57

Название

Шайба 4x6

ID класса

21

Сохранить

Обратно

© 2023 - Some two girls

ДФ11	Гайка M8 самонарезающая	Гайки	шт	<div><div></div><div></div><div></div></div>
ДФ19	Гайка M8 DIN 6923	Гайки	шт	<div><div></div><div></div><div></div></div>
ДФ42	Шайба M8 DIN 9021 увеличенная	Шайбы	шт	<div><div></div><div></div><div></div></div>
ДФ53	Болт M8*40 DIN 933	Секции/модули	шт	<div><div></div><div></div><div></div></div>
ДФ54	Болт M8*55	Болты	шт	<div><div></div><div></div><div></div></div>
ДФ57	Шайба 4x6	Шайбы	шт	<div><div></div><div></div><div></div></div>
ДФ78	Винт M8*50 DIN 7985	Винты	шт	<div><div></div><div></div><div></div></div>
ДФ82	Шайба 24x8 (пнд)	Шайбы	шт	<div><div></div><div></div><div></div></div>
ДФ84	Механизм трансформации	Механизмы	шт	<div><div></div><div></div><div></div></div>

Рисунок 4.17 – Добавление нового изделия «Шайба 4x6» и результат

Спецификации изделий

На главную

Выбрать

Обновить изделие

ID изделия

ДФ113

Название

Ключ шестигранный №5

ID класса

24

Обновить

Обратно

© 2023 - Some two girls

ДФ11	Винт M8*40	Винты	шт	<div><div></div><div></div><div></div></div>
ДФ113	Ключ шестигранный №5	Ключи	шт	<div><div></div><div></div><div></div></div>
ДФ114	Гайка M8 ERICSON	Гайки	шт	<div><div></div><div></div><div></div></div>

Рисунок 4.18 – Начальный ID класса, изменение ID класса у изделия «Ключ шестигранный №5» и результат

Удалить?

ID изделия

7

Название

Диван-кровать Литен

ID класса

5

Удалить

Обратно

© 2023 - Some two girls

Список изделий (53)

ID	Название	Класс изделия	Единица измерения	Новое изделие
1	Кресло-реклайнер Хоган	Кресла-кровати	кг	  
2	Диван угловой Домо ПРО	Угловые диваны	кг	  
3	Диван угловой серии Кларк	Угловые диваны	кг	  
4	Кресло Хюгге	Простые кресла	кг	  
5	Кресло Муни	Простые кресла	кг	  
6	Диван Ника	Прямые диваны	кг	  
8	Кресло-кровать Персей Nova	Кресла-кровати	кг	  
FMP004	Винт 6*50 DIN 7420	Винты	шт	  

Рисунок 4.19 – Удаление изделия «Диван-кровать Литен» и результат

Кроме того, если мы изменим единицы измерения у классов изделий «Диваны» и «Кресла», то они изменятся и у изделий. Результат показан на рис. 4.20:

 Спецификации изделий На главную Выбрать ▾

Список изделий (53)





















ID	Название	Класс изделия	Единица измерения	Новое изделие
1	Кресло-реклайнер Хоган	Кресла-кровати	шт	  
2	Диван угловой Домо ПРО	Угловые диваны	шт	  
3	Диван угловой серии Кларк	Угловые диваны	шт	  
4	Кресло Хюгге	Простые кресла	шт	  
5	Кресло Муни	Простые кресла	шт	  
6	Диван Ника	Прямые диваны	шт	  
8	Кресло-кровать Персей Nova	Кресла-кровати	шт	  

Рисунок 4.20 – Изменение единиц измерения у классов «Диваны» и «Кресла»

Теперь рассмотрим методы, которые доступны при просмотре информации об изделии при нажатии на соответствующую кнопку. В частности:

- Вывод графа классов продукта;

- Поиск родителей.

На рис. 4.21 – 4.23 представлены примеры вывода соответствующей информации об изделии «Гайка 6 DIN 985»:



Рисунок 4.21 – Окно информации об изделии «Гайка 6 DIN 985»

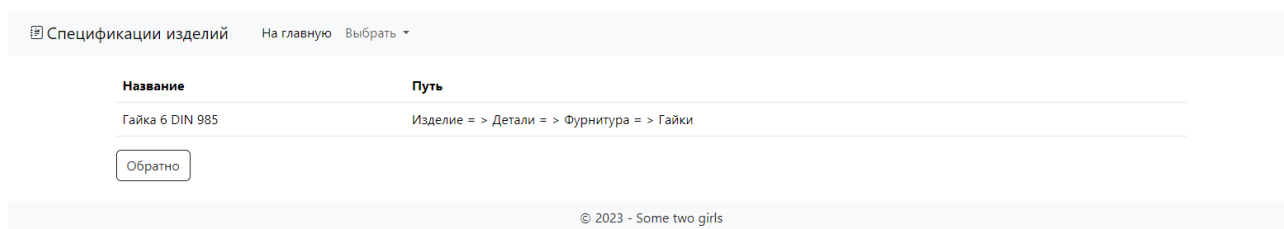


Рисунок 4.22 – Вывод графа классов для изделия «Гайка 6 DIN 985»

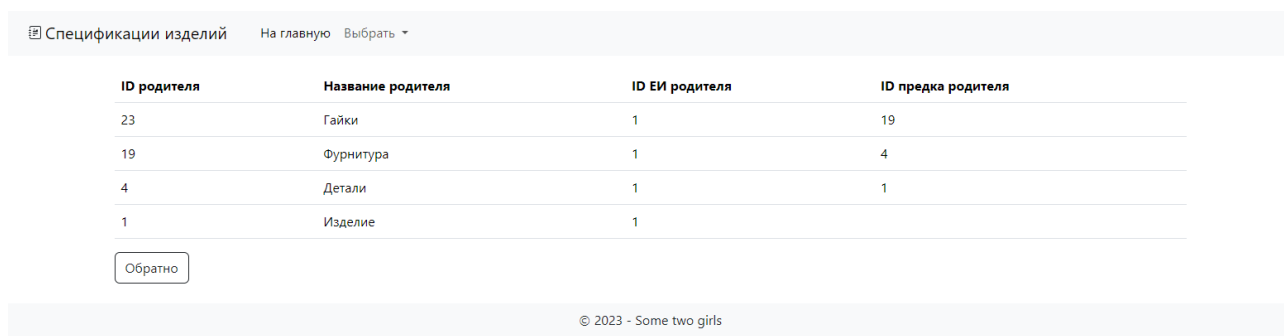


Рисунок 4.23 – Поиск родителей для изделия «Гайка 6 DIN 985»

5.3.3. Раздел «Единицы измерения»

Раздел «Единицы измерения» представлен на рис. 4.24. Всего изначально в таблице представлено 2 единицы измерения – Вес и Штука.

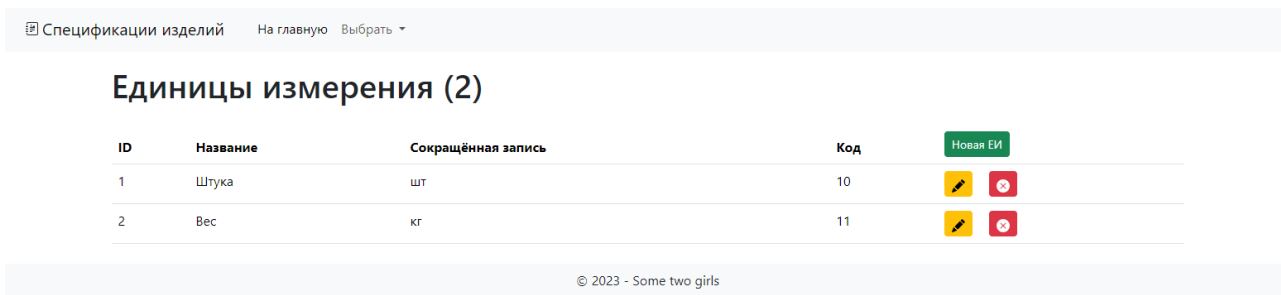
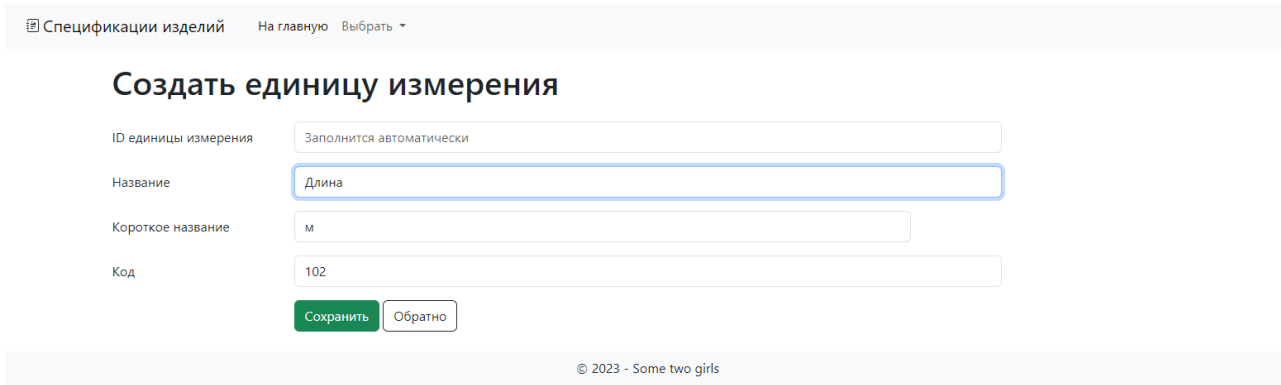


Рисунок 4.24 – Раздел «Единицы измерения»

Здесь также можно создать новую ЕИ, отредактировать и удалить уже существующие ЕИ. Примеры представлены на рис. 4.25 – 4.27:



Единицы измерения (3)







ID	Название	Сокращённая запись	Код	Новая ЕИ
1	Штука	шт	10	 
2	Вес	кг	11	 
3	Длина	м	102	 

Рисунок 4.25 – Создание новой единицы измерения «Длины» и результат

Спецификации изделий

На главную

Выбрать

Обновить ЕИ

ID единицы измерения

3

Название

Длина в сантиметрах

Короткое название

см

Код

112

Обновить

Обратно

© 2023 - Some two girls

Единицы измерения (3)

ID	Название	Сокращённая запись	Код	Новая ЕИ
1	Штука	шт	10	<div><div></div><div></div></div>
2	Вес	кг	11	<div><div></div><div></div></div>
3	Длина в сантиметрах	см	112	<div><div></div><div></div></div>

Рисунок 4.26 – Редактирование единицы измерения «Длины» и результат

Спецификации изделий

На главную

Выбрать

Удалить?

ID единицы измерения

3

Название

Длина в сантиметрах

Короткое название

см

Код

112

Удалить

Обратно

© 2023 - Some two girls

Единицы измерения (2)

ID	Название	Сокращённая запись	Код	Новая ЕИ
1	Штука	шт	10	<div><div></div><div></div></div>
2	Вес	кг	11	<div><div></div><div></div></div>

Рисунок 4.27 – Удаление единицы измерения «Длины» и результат

5.4. Роль пользователя: «Технолог / Конструктор»

После всех действий, которые были совершены в подразделе 5.3, пересоздадим заново базу данных, чтобы вернуть её в исходное состояние, и перезапустим сервер.

На рис. 5.1 представлено то, какое предупреждение выдаётся пользователю при выборе роли «Технолог / Конструктор»:

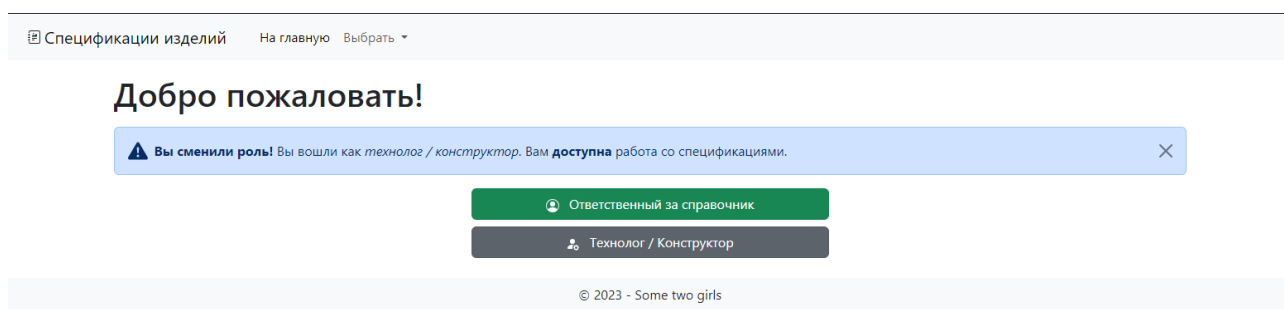


Рисунок 5.1 – Предупреждение о выборе роли «Технолог / Конструктор»

При этом ему становится доступна работа с материальной спецификацией изделий (см. рис. 5.2), а также *со всеми разделами, рассмотренными для роли «Ответственный за справочник»* (см. пункты 5.3.1 – 5.3.3):

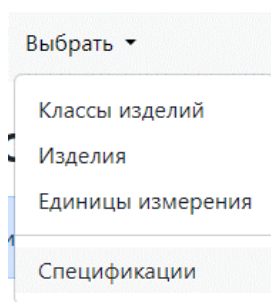


Рисунок 5.2 – Доступ к выбору раздела «Спецификации» для роли «Технолог / Конструктор»

5.4.1. Раздел «Спецификации»

Перейдём к разделу «Спецификации» (см. рис. 5.3). Здесь аналогично представлено название выбранного раздела, а также общее количество записей в просматриваемой таблице. Всего имеется 66 записей.

Все спецификации (66)

№ позиции	ID изделия	Название изделия	ID части изделия	Часть изделия	Количество	Новая спецификация			
1	1	Кресло-реклайнер Хоган	Д32	Модуль Кресло	1				Расход
2	1	Кресло-реклайнер Хоган	Д06	Подлокотник	2				Расход
3	1	Кресло-реклайнер Хоган	ДФ07	Болт М8*60 DIN 933	2				Расход
4	1	Кресло-реклайнер Хоган	ДФ42	Шайба М8 DIN 9021 увеличенная	2				Расход
1	2	Диван угловой Домо ПРО	ДФ84	Механизм трансформации	1				Расход
2	2	Диван угловой Домо ПРО	FMP113	Болт 6*20 DIN 933	3				Расход
3	2	Диван угловой Домо ПРО	FMP068	Болт 6*40 DIN 933	2				Расход
4	2	Диван угловой Домо ПРО	FMP078	Болт 8*45 DIN 933	2				Расход
5	2	Диван угловой Домо ПРО	FMP008	Винт 6*16 DIN 965	4				Расход

Рисунок 5.3 – Фрагмент раздела «Спецификации»

Как и в пунктах 5.3.1 – 5.3.3 здесь доступны:

- Создание позиции спецификации с проверкой на наличие цикла, то есть при попытке добавления цикла в спецификацию произойдёт переход на страницу со всеми спецификациями, но позиция добавлена не будет и количество записей в таблице не изменится (см. примеры на рис. 4.4, 4.17, 4.25);
- Редактирование позиции спецификации (см. примеры на рис. 4.6, 4.18, 4.26);
- Удаление конкретной позиции спецификации (см. примеры на рис. 4.13 - 4.15, 4.19, 4.27).

Удаление всей спецификации конкретного продукта происходит при удалении изделия или класса изделия из разделов «Изделия» или «Классы изделий» соответственно.

При нажатии на кнопку показа информации произойдёт переход на страницу с полной спецификацией конкретного изделия, то есть необходимо смотреть именно на столбцы ***ID изделия*** и ***название изделия*** при поиске какой-либо спецификации. Например, покажем спецификацию изделия «Кресло

Муни» (ID изделия - 5), причём ID изделия отображается в адресной строке сверху браузера (см. рис. 5.4):

Позиция	ID части	Название части	Количество	ЕИ	Спецификация далее
1	Д15	Модуль Кресло	1	Штука	
2	Д100	Подушка сиденья	1	Штука	
3	ДФ42	Шайба M8 DIN 9021 увеличенная	4	Штука	
1	Д04	Спинка	1	Штука	
2	Д06	Подлокотник	2	Штука	
1	Т01	Ткань для подлокотника	2	Штука	
2	О01	Основание подлокотника	2	Штука	

[Обратно](#)

© 2023 - Some two girls

Рисунок 5.4 – Спецификация изделия «Кресло Муни» (ID изделия - 5)

Как можно заметить, здесь есть позиции с повторяющимися номерами. Это означает, что два изделия в составе выбранного кресла также имеют составные части. Для этого предусмотрен столбец «*Спецификация далее*». Если позиция спецификации состоит из других частей, то по соответствующей кнопке можно перейти в спецификацию этой части исходного изделия, иначе перехода не произойдёт и в терминале появится сообщение «structure of query does not match function result type».

К примеру, «Кресло Муни» состоит из «Модуля Кресло», которое включает в себя «Спинку» и два «Подлокотника» (см. рис. 5.5). «Подлокотник» же состоит из «Ткани для подлокотника» и «Основания подлокотника» (см. рис. 5.6).

Позиция	ID части	Название части	Количество	ЕИ	Спецификация далее
1	Д04	Спинка	1	Штука	ⓘ
2	Д06	Подлокотник	2	Штука	ⓘ
1	Т01	Ткань для подлокотника	2	Штука	ⓘ
2	О01	Основание подлокотника	2	Штука	ⓘ

[Назад](#)

© 2023 - Some two girls

Рисунок 5.5 – Спецификация изделия «Модуль Кресло»

Позиция	ID части	Название части	Количество	ЕИ	Спецификация далее
1	Т01	Ткань для подлокотника	1	Штука	ⓘ
2	О01	Основание подлокотника	1	Штука	ⓘ

[Назад](#)

© 2023 - Some two girls

Рисунок 5.6 – Спецификация изделия «Подлокотник»

Также, на странице «Спецификации» у каждой позиции предусмотрена кнопка «Расход». Она отвечает за расчёт сводных норм расхода материальных ресурсов для выбранного изделия.

Продолжим рассмотрение на примере изделия «Кресло Муни». По теоретическим расчётам для 4-х подобных кресел потребуется:

- 4 модуля Кресло;
- 4 подушки сиденья;
- 16 шайб М8 DIN 9021 увеличенных;
- 4 спинки;
- 8 подлокотников;
- 8 тканей для подлокотников;
- 8 оснований для подлокотников.

При нажатии на кнопку «Расход» получаем следующий результат (см. рис. 5.7 и 5.8):

Спецификации изделий На главную Выбрать ▾

ID 5

Количество 4

Рассчитать Обратно

© 2023 - Some two girls

Рисунок 5.7 – Окно при нажатии на кнопку «Расход»

ID	Наименование	Количество	ЕИ
O01	Основание подлокотника	8	Штука
T01	Ткань для подлокотника	8	Штука
ДФ42	Шайба M8 DIN 9021 увеличенная	16	Штука
Д06	Подлокотник	8	Штука
Д100	Подушка сиденья	4	Штука
Д04	Спинка	4	Штука
Д15	Модуль Кресло	4	Штука

Обратно

Рисунок 5.8 – Расчёт сводных норм расхода материальных ресурсов для 4-х «Кресел Муни»

Как можно заметить по результатам на рис. 5.8, программа произвела расчёт верно.

Теперь в качестве эксперимента добавим «Ткань для подлокотника» для «Подушки сиденья» (см. рис. 5.9 и 5.10):

Создать спецификацию

Позиция Заполнится автоматически


ID изделия Д100

ID составной части Т01

Количество частей 1

Сохранить Обратно

Рисунок 5.9 – Создание новой позиции спецификации для «Подушки сиденья»

Позиция	ID части	Название части	Количество	ЕИ	Спецификация далее
1	T01	Ткань для подлокотника	1	Штука	

[Обратно](#)

Рисунок 5.10 – Спецификация изделия «Подушка сиденья»

По теоретическим расчётам для 4-х кресел должен получиться следующий результат:

- 4 модуля Кресло;
- 4 подушки сиденья;
- 16 шайб M8 DIN 9021 увеличенных;
- 4 спинки;
- 8 подлокотников;
- 12 тканей для подлокотников;
- 8 оснований для подлокотников.

При помощи программы будет получен результат, аналогичный расчётам (см. рис. 5.11):

ID	Наименование	Количество	ЕИ
O01	Основание подлокотника	8	Штука
T01	Ткань для подлокотника	12	Штука
ДФ42	Шайба M8 DIN 9021 увеличенная	16	Штука
Д06	Подлокотник	8	Штука
Д100	Подушка сиденья	4	Штука
Д04	Спинка	4	Штука
Д15	Модуль Кресло	4	Штука

[Обратно](#)

Рисунок 5.11 – Расчёт сводных норм расхода материальных ресурсов для 4-х «Кресел Муни» при добавлении новой позиции спецификации для «Подушки сиденья»

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы при помощи Node.js и его веб-фреймворка Express.js, а также с использованием EJS, HTML, JavaScript и Bootstrap был разработан и протестирован графический интерфейс, обеспечивающий работу со спецификациями изделий. На главной странице пользователь может выбрать свою роль – «Ответственный за справочник» или «Технолог / Конструктор». При этом он получит соответствующее предупреждение о смене роли.

Ответственному за справочник (см. подраздел 5.3) доступна работа со всеми справочниками, то есть он может создавать, удалять и редактировать классы изделий, сами изделия и единицы измерений. Кроме того, он может находить потомков, потомков и соответствующие изделия, родителей и все изделия конкретного класса. Для «изделий» ему доступен поиск родительских классов изделия, а также вывод графа классов изделия.

Технологу / конструктору (см. подраздел 5.4) доступны все те же функции и разделы, что и ответственному за справочник. Однако, он дополнительно может управлять материальной спецификацией изделий. В частности, он может добавлять, редактировать и удалять позиции спецификаций, просматривать информацию о спецификации изделия на всю глубину вложенности, а также рассчитывать сводные нормы расхода материальных ресурсов для различного количества изделий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Bootstrap // Bootstrap. The most popular HTML, CSS and JS library in the world. URL: <https://getbootstrap.com> (дата обращения: 24.12.2023)
2. Документация EJS // Express 4.x – API Reference. URL: <https://expressjs.com/en/api.html> (дата обращения: 20.12.2023)
3. Документация Express.js // EJS -- Embedded JavaScript templates. URL: <https://ejs.co> (дата обращения: 20.12.2023)
4. Документация Node.js v20.11.0 // Node.js v20.11.0 documentation. URL: <https://nodejs.org/docs/latest-v20.x/api/all.html#> (дата обращения: 13.12.2023)
5. Документация node-postgres // Welcome – node-postgres. URL: <https://node-postgres.com> (дата обращения: 21.12.2023)
6. В.А. Дубенецкий, А.Г. Кузнецов. Использование объектных моделей при создании информационных систем: учебно-методическое пособие. СПбГЭТУ «ЛЭТИ», 2016. 220 с.
7. В.А. Дубенецкий, А.Г. Кузнецов. Проектирование информационных систем с использованием UML: учебно-методическое пособие. СПбГЭТУ «ЛЭТИ», 2016. 62 с.
8. Информационное обеспечение жизненного цикла изделий. Учебное пособие: СПб, Издательство Политехнического университета, 2012
9. Проектирование корпоративных информационных систем.: СПб, Изд-во СПбГЭТУ «ЛЭТИ», 2013

ПРИЛОЖЕНИЕ А

ЛИСТИНГ КОДА ПРОГРАММЫ

Листинг А.1 – Создание основных таблиц и функций, заполнение базы данных (файл creation.js)

```
const pool = require('./pool').pool;

const sql_create_table_func = `/*Создание таблиц*/
CREATE TABLE IF NOT EXISTS ei (
    ei_id SERIAL NOT NULL,
    name VARCHAR(50) NOT NULL,
    short_name VARCHAR(25) NOT NULL,
    code INTEGER NOT NULL,
    CONSTRAINT ei_id PRIMARY KEY (ei_id)
);

CREATE TABLE IF NOT EXISTS product_classifier (
    class_id SERIAL NOT NULL,
    parent_class_id INTEGER,
    ei_id INTEGER NOT NULL,
    name VARCHAR(50) NOT NULL,
    CONSTRAINT prod_class_id PRIMARY KEY (class_id),
    FOREIGN KEY (ei_id) REFERENCES ei (ei_id)
    ON DELETE CASCADE ON UPDATE CASCADE
    NOT DEFERRABLE,
    FOREIGN KEY (parent_class_id) REFERENCES product_classifier
(class_id)
    ON DELETE CASCADE ON UPDATE CASCADE
    NOT DEFERRABLE
);

CREATE TABLE IF NOT EXISTS product (
    prod_id VARCHAR(50) NOT NULL,
    class_id INTEGER NOT NULL,
    name VARCHAR(50) NOT NULL,
    CONSTRAINT prod_id PRIMARY KEY (prod_id),
    FOREIGN KEY (class_id) REFERENCES product_classifier (class_id)
    ON DELETE CASCADE ON UPDATE CASCADE
    NOT DEFERRABLE
);

CREATE TABLE IF NOT EXISTS product_specification (
    pos_num INTEGER NOT NULL,
    prod_id VARCHAR(50) NOT NULL,
    part_id VARCHAR(50) NOT NULL,
```

```

        quantity INTEGER NOT NULL,
        CONSTRAINT prod_spec_num PRIMARY KEY (pos_num, prod_id),
        FOREIGN KEY (prod_id) REFERENCES product (prod_id)
        ON DELETE CASCADE ON UPDATE CASCADE
        NOT DEFERRABLE,
        FOREIGN KEY (part_id) REFERENCES product (prod_id)
        ON DELETE CASCADE ON UPDATE CASCADE
        NOT DEFERRABLE
    );

/*Методы*/
/*Методы класса product_classifier*/
--Создание класса
CREATE OR REPLACE FUNCTION createClass(new_name VARCHAR(50), new_ei_id INTEGER,
new_parent_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF ((SELECT COUNT(*) FROM product_classifier WHERE class_id = new_parent_id) =
1 OR
        (new_parent_id IS NULL AND (SELECT COUNT(*) FROM product_classifier) =
0)) AND
        (SELECT COUNT(*) FROM ei WHERE ei_id = new_ei_id) = 1 AND
        (SELECT COUNT(*) FROM product_classifier WHERE name = new_name) = 0
    THEN INSERT INTO product_classifier(name, ei_id, parent_class_id) VALUES
(new_name, new_ei_id, new_parent_id);
    RETURN 1;
    ELSE
    RETURN 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Создание корневого класса
CREATE OR REPLACE FUNCTION createRootClass(new_name TEXT, new_ei_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM ei WHERE ei_id = new_ei_id) = 1 AND
        (SELECT COUNT(*) FROM product_classifier WHERE name = new_name) = 0 AND
        ((SELECT COUNT(*) FROM product_classifier WHERE parent_class_id IS NULL) = 1 OR
        (SELECT COUNT(*) FROM product_classifier) = 0) THEN
        INSERT INTO product_classifier (name, ei_id, parent_class_id) VALUES
(new_name, new_ei_id, NULL);
        UPDATE product_classifier SET parent_class_id = (SELECT class_id FROM
product_classifier WHERE name = new_name)
        WHERE class_id = (SELECT class_id FROM product_classifier WHERE
parent_class_id IS NULL AND name != new_name);
    RETURN 1;

```

```

        ELSE
            RETURN 0;
        END IF;
    END;
END;
$$ LANGUAGE plpgsql;

--Выбрать
CREATE OR REPLACE FUNCTION selectClass(target_id INTEGER)
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 1
    THEN
        RETURN QUERY SELECT class_id, name, ei_id, parent_class_id FROM
        product_classifier WHERE class_id = target_id;
    ELSE
        RETURN QUERY SELECT 0, '0', 0, 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Удаление класса
CREATE OR REPLACE FUNCTION deleteClass(target_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN RETURN -1;
    END IF;
    IF (SELECT COUNT(*) FROM product_classifier WHERE parent_class_id = target_id)
    = 0 AND
        (SELECT COUNT(*) FROM product WHERE class_id = target_id) = 0
    THEN DELETE FROM product_classifier WHERE class_id = target_id;
    RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Удалить вместе с потомками
CREATE OR REPLACE FUNCTION deleteClassAndDesc(target_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0

```

```

        THEN RETURN 0;
    END IF;
    IF (SELECT COUNT(*) FROM product_classifier WHERE parent_class_id = target_id)
= 0 AND
        (SELECT COUNT(*) FROM product WHERE class_id = target_id) = 0
    THEN DELETE FROM product_classifier WHERE class_id = target_id;
    RETURN 1;
    ELSE
        DELETE FROM product_classifier WHERE class_id = target_id;
        DELETE FROM product WHERE class_id = target_id;
        DELETE FROM product_classifier WHERE parent_class_id = target_id;
        DELETE FROM product WHERE class_id IN (SELECT class_id FROM
product_classifier WHERE parent_class_id = target_id);
    RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Изменить родителя
CREATE OR REPLACE FUNCTION changePcParent(new_class_id INTEGER, new_parent_id
INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF new_parent_id = new_class_id
        THEN RETURN 0;
    END IF;
    IF ((SELECT COUNT(*) FROM product_classifier WHERE class_id = new_parent_id) =
1 OR
        (new_parent_id IS NULL AND (SELECT COUNT(*) FROM product_classifier) =
1)) AND
        (SELECT COUNT(*) FROM product_classifier WHERE class_id = new_class_id) = 1
    THEN UPDATE product_classifier SET parent_class_id = new_parent_id WHERE
class_id = new_class_id;
    RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Выбрать родителя
CREATE OR REPLACE FUNCTION selectPcParent(target_id INTEGER)
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER)
AS $$
BEGIN

```



```

        IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 1
    THEN
        IF (SELECT parent_class_id IS NOT NULL FROM product_classifier WHERE
class_id = target_id) = TRUE THEN
            RETURN QUERY SELECT class_id, name, ei_id, parent_class_id FROM
product_classifier WHERE class_id = (SELECT parent_class_id FROM product_classifier
WHERE class_id = target_id);
        ELSE
            RETURN QUERY SELECT 0, 'NULL', 0, 0;
        END IF;
    ELSE
        RETURN QUERY SELECT 0, '0', 0, 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Редактировать
CREATE OR REPLACE FUNCTION changePc(target_id INTEGER, new_name VARCHAR(50),
new_ei_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN
        RETURN 0;
    ELSE
        UPDATE product_classifier SET name = new_name, ei_id = new_ei_id WHERE
class_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Найти потомков
CREATE OR REPLACE FUNCTION selectPcChildren(target_id INTEGER)
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN
        RETURN QUERY SELECT 0, '0', 0, 0;
    ELSE
        RETURN QUERY WITH RECURSIVE children AS (
            SELECT class_id, name, ei_id, parent_class_id FROM product_classifier
WHERE class_id = target_id
            UNION ALL

```

```

        SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc, children c WHERE pc.parent_class_id = c.class_id
    ) SELECT * FROM children WHERE class_id != target_id;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Найти всех потомков, включая product
CREATE OR REPLACE FUNCTION selectPcChildrenWithProducts(target_id INTEGER)
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER,
                result_prod_id VARCHAR(50),
                result_prod_class_id INTEGER,
                result_prod_name VARCHAR(50))
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN
        RETURN QUERY SELECT 0, '0', 0, 0, '0', 0, '0';
    ELSE
        RETURN QUERY WITH RECURSIVE children AS (
            SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc WHERE class_id = target_id
            UNION ALL
            SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc, children c WHERE pc.parent_class_id = c.class_id
        ) SELECT *
            FROM children c
            LEFT JOIN product p ON c.class_id = p.class_id
            WHERE (c.class_id != target_id OR c.class_id = target_id);
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Найти родителей
CREATE OR REPLACE FUNCTION selectPcParents(target_id INTEGER)
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN
        RETURN QUERY SELECT 0, '0', 0, 0;
    ELSE
        RETURN QUERY WITH RECURSIVE parents AS (

```

```

        SELECT class_id, name, ei_id, parent_class_id FROM product_classifier
WHERE class_id = target_id
        UNION ALL
        SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc, parents p WHERE pc.class_id = p.parent_class_id
    ) SELECT * FROM parents WHERE class_id != target_id;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Найти изделия этого класса
CREATE OR REPLACE FUNCTION selectClassProd(target_id INTEGER)
RETURNS TABLE (result_prod_id VARCHAR(50),
                result_prod_name VARCHAR(50),
                result_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = target_id) = 0
    THEN
        RETURN QUERY SELECT '0', '0', 0;
    ELSE
        RETURN QUERY WITH RECURSIVE children AS (
            SELECT class_id, name, ei_id, parent_class_id FROM product_classifier
WHERE class_id = target_id
            UNION ALL
            SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc, children c WHERE pc.parent_class_id = c.class_id
        ) SELECT p.prod_id, p.name, c.class_id
        FROM children c
        RIGHT JOIN product p ON c.class_id = p.class_id
        WHERE (c.class_id != target_id OR c.class_id = target_id);
    END IF;
END;
$$ LANGUAGE plpgsql;

/*Методы класса product*/
--Создание изделия
CREATE OR REPLACE FUNCTION createProduct(new_prod_id VARCHAR(50), new_name
VARCHAR(50), new_class_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = new_class_id) = 0
    THEN RETURN 0;
    END IF;
    INSERT INTO product (prod_id, name, class_id) VALUES (new_prod_id, new_name,
new_class_id);
    RETURN 1;
END;

```

```

$$ LANGUAGE plpgsql;

--Удалить изделие
CREATE OR REPLACE FUNCTION deleteProduct(target_id VARCHAR(50))
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0
        THEN RETURN 0;
    ELSE
        DELETE FROM product WHERE prod_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Изменить класс изделия
CREATE OR REPLACE FUNCTION changeProductParent(target_id VARCHAR(50), new_class_id
INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0
        THEN RETURN 0;
    END IF;
    IF (SELECT COUNT(*) FROM product_classifier WHERE class_id = new_class_id) = 0
        THEN RETURN -1;
    ELSE
        UPDATE product SET class_id = new_class_id WHERE prod_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Найти родителей
CREATE OR REPLACE FUNCTION getProductParents(target_id VARCHAR(50))
RETURNS TABLE (parent_class_id INTEGER,
                parent_name VARCHAR(50),
                parent_ei_id INTEGER,
                parent_parent_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0
        THEN RETURN QUERY SELECT 0, '0', 0, 0;
    ELSE
        RETURN QUERY WITH RECURSIVE parents AS (
            SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc WHERE pc.class_id = (SELECT class_id FROM product WHERE
prod_id = target_id)

```

```

        UNION ALL
        SELECT pc.class_id, pc.name, pc.ei_id, pc.parent_class_id FROM
product_classifier pc, parents p WHERE pc.class_id = p.parent_class_id
    ) SELECT * FROM parents;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Выбрать класс продукта
CREATE OR REPLACE FUNCTION selectProductParent(target_id VARCHAR(50))
RETURNS TABLE (result_class_id INTEGER,
                result_name VARCHAR(50),
                result_ei_id INTEGER,
                result_parent_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0 THEN
        RETURN QUERY SELECT 0, '0', 0, 0;
    END IF;
    RETURN QUERY SELECT class_id, name, ei_id, parent_class_id FROM
product_classifier WHERE class_id = (SELECT class_id FROM product WHERE prod_id =
target_id);
END;
$$ LANGUAGE plpgsql;

-- Выбрать
CREATE OR REPLACE FUNCTION selectProduct(target_id VARCHAR(50))
RETURNS TABLE (result_prod_id VARCHAR(50),
                result_name VARCHAR(50),
                result_class_id INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0 THEN
        RETURN QUERY SELECT 0, '0', 0;
    END IF;
    RETURN QUERY SELECT prod_id, name, class_id FROM product WHERE prod_id =
target_id;
END;
$$ LANGUAGE plpgsql;

-- Редактировать
CREATE OR REPLACE FUNCTION changeProduct(target_id VARCHAR(50), new_name
VARCHAR(50))
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = target_id) = 0 THEN
        RETURN 0;
    ELSE

```

```

        UPDATE product SET name = new_name WHERE prod_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

--Просмотр графа классов до продукта
CREATE OR REPLACE FUNCTION showClassTree(target_id VARCHAR(50))
RETURNS TABLE (class_names TEXT,
                product_name VARCHAR(50))
AS $$
DECLARE prod_class_id INTEGER;
DECLARE curr_class_id INTEGER;
BEGIN
prod_class_id = (SELECT class_id FROM product WHERE prod_id = target_id);
curr_class_id = 1;
RETURN QUERY WITH RECURSIVE parents(class_id, parent_class_id, name, path) AS (
    SELECT pc.class_id, pc.parent_class_id, pc.name, CAST (pc.name AS TEXT) as path
    FROM product_classifier pc WHERE pc.class_id=curr_class_id
    UNION ALL
    SELECT pc.class_id, pc.parent_class_id, pc.name, CAST ( p.path || ' > ' ||
pc.name AS TEXT)
    FROM product_classifier pc INNER JOIN parents p ON( p.class_id =
pc.parent_class_id )
SELECT result.path, p.name FROM parents result LEFT JOIN product p on
p.prod_id=target_id
WHERE result.class_id = prod_class_id
ORDER BY path, p.name;
END;
$$ LANGUAGE plpgsql;

/*Методы класса ei*/
--Создать единицу измерения
CREATE OR REPLACE FUNCTION addEi(new_name VARCHAR(50), new_short_name VARCHAR(25),
new_code INTEGER)
RETURNS TABLE (new_ei_id INTEGER,
                success INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM ei WHERE (name = new_name) OR (short_name =
new_short_name) OR (code = new_code)) = 0 THEN
        INSERT INTO ei(name, short_name, code) VALUES (new_name, new_short_name,
new_code);
        RETURN QUERY SELECT ei_id, 1 FROM ei WHERE (name = new_name) OR (short_name
= new_short_name) OR (code = new_code);
    ELSE
        RETURN QUERY SELECT ei_id, 0 FROM ei WHERE (name = new_name) OR (short_name
= new_short_name) OR (code = new_code);
    END IF;
END;

```

```

END;
$$ LANGUAGE plpgsql;

--Удалить единицу измерения
CREATE OR REPLACE FUNCTION deleteEi(target_id INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM ei WHERE ei_id = target_id) = 0 OR
        (SELECT COUNT(*) FROM product_classifier WHERE ei_id = target_id) > 1 THEN
        RETURN 0;
    ELSE
        DELETE FROM ei WHERE ei_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Редактировать
CREATE OR REPLACE FUNCTION changeEi(target_id INTEGER, new_name VARCHAR(50),
new_short_name VARCHAR(25), new_code INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM ei WHERE ei_id = target_id) = 0 THEN
        RETURN 0;
    ELSE
        UPDATE ei SET name = new_name, short_name = new_short_name, code = new_code
WHERE ei_id = target_id;
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Выбрать
CREATE OR REPLACE FUNCTION selectEi(target_id INTEGER)
RETURNS TABLE (result_ei_id INTEGER,
                result_name VARCHAR(50),
                result_short_name VARCHAR(25),
                result_code INTEGER)
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM ei WHERE ei_id = target_id) = 0 THEN
        RETURN QUERY SELECT 0, '0', '0', 0;
    ELSE
        RETURN QUERY SELECT ei_id, name, short_name, code FROM ei WHERE ei_id =
target_id;
    END IF;
END;

```

```

$$ LANGUAGE plpgsql;

/*Методы класса product_specification*/
-- Создать позицию спецификации
CREATE OR REPLACE FUNCTION addProdSpec(new_prod_id VARCHAR(50), new_part_id
VARCHAR(50), new_quantity INTEGER)
RETURNS TABLE (pos_num INTEGER,
                prod_id VARCHAR(50),
                part_id VARCHAR(50),
                quantity INTEGER)
AS $$
DECLARE position INTEGER;
BEGIN
position := (SELECT COUNT(*) FROM product_specification ps WHERE ps.prod_id =
new_prod_id);
position := position +1;
INSERT INTO product_specification(prod_id, pos_num, part_id, quantity)
VALUES (new_prod_id, position, new_part_id, new_quantity);
RETURN QUERY SELECT * FROM product_specification;
END;
$$ LANGUAGE plpgsql;

-- Удалить позицию спецификации
CREATE OR REPLACE FUNCTION deleteProdSpec(target_id VARCHAR(50), target_pos_num
INTEGER)
RETURNS TABLE (pos_num INTEGER,
                prod_id VARCHAR(50),
                part_id VARCHAR(50),
                quantity INTEGER)
AS $$
DECLARE curr_pos INTEGER;
BEGIN
DELETE FROM product_specification WHERE (prod_id = target_id AND pos_num =
target_pos_num);
curr_pos := target_pos_num;
    WHILE (curr_pos < (SELECT COUNT(*) FROM product_specification ps WHERE
ps.prod_id = target_id)+1)
    LOOP
        UPDATE product_specification SET pos_num = curr_pos WHERE (prod_id = target_id
AND pos_num = curr_pos+1);
        curr_pos:= curr_pos +1;
    END LOOP;
RETURN QUERY SELECT * FROM product_specification;
END;
$$ LANGUAGE plpgsql;

-- Удалить всю спецификацию конкретного продукта
CREATE OR REPLACE FUNCTION deleteProdSpecOfProd(target_id VARCHAR(50))
RETURNS product_specification

```



```

AS $$
DELETE FROM product_specification
WHERE prod_id = target_id
RETURNING *;
$$ LANGUAGE sql;

-- Редактировать позицию спецификации
CREATE OR REPLACE FUNCTION changeProdSpec(target_id VARCHAR(50), target_pos_num
INTEGER, new_part_id VARCHAR(50), new_quantity INTEGER)
RETURNS INTEGER
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_specification WHERE (prod_id = target_id AND
pos_num = target_pos_num)) = 0 THEN
        RETURN 0;
    ELSE
        UPDATE product_specification SET part_id = new_part_id, quantity =
new_quantity WHERE (prod_id = target_id AND pos_num = target_pos_num);
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Вывести спецификацию продукта (со всеми уровнями вложенности)
CREATE OR REPLACE FUNCTION selectProdSpec(target_id VARCHAR(50))
RETURNS TABLE (result_pos_num INTEGER,
                result_part_id VARCHAR(50),
                result_part_name VARCHAR(50),
                result_quantity INTEGER,
                result_ei_name VARCHAR(50))
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM product_specification WHERE prod_id = target_id) = 0
THEN
        RETURN QUERY SELECT 0, 0, 0, '0', 0;
    ELSE
        RETURN QUERY WITH RECURSIVE children AS (
            SELECT ps.prod_id, ps.pos_num, ps.part_id, ps.quantity FROM
product_specification ps WHERE prod_id = target_id
            UNION ALL
            SELECT ps.prod_id, ps.pos_num, ps.part_id, ps.quantity*c.quantity FROM
product_specification ps, children c WHERE ps.prod_id = c.part_id
        ) SELECT c.pos_num, c.part_id, p.name, c.quantity, ei.name
        FROM children c LEFT JOIN product p ON c.part_id = p.prod_id
            LEFT JOIN product_classifier pc ON p.class_id = pc.class_id
            LEFT JOIN ei ON pc.ei_id = ei.ei_id;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

```

-- Рассчитать расходы
CREATE OR REPLACE function countNeeds(target_id VARCHAR(50), amount INTEGER)
RETURNS TABLE (prod_id VARCHAR(50),
                name VARCHAR(50),
                quantity INTEGER,
                result_ei_name VARCHAR(50))
AS $$
BEGIN
    RETURN QUERY WITH RECURSIVE children (prod_id, part_id, quantity) AS (
        SELECT ps.prod_id, ps.part_id, ps.quantity*amount as new_amount FROM
product_specification ps WHERE ps.prod_id = target_id
        UNION ALL
        SELECT ps.prod_id, ps.part_id, ps.quantity*c.quantity as final_amount
FROM product_specification ps, children c WHERE ps.prod_id = c.part_id
    ) SELECT c.part_id, p.name, c.quantity, ei.name
    FROM children c LEFT JOIN product p ON c.part_id = p.prod_id
        LEFT JOIN product_classifier pc ON p.class_id = pc.class_id
        LEFT JOIN ei ON pc.ei_id = ei.ei_id;
END;
$$ LANGUAGE plpgsql;

-- Проверка на наличие циклов
CREATE OR REPLACE FUNCTION addProdSpecWithCheckLoops(new_prod_id VARCHAR(50),
new_part_id VARCHAR(50), new_quantity INTEGER)
RETURNS INTEGER
AS $$
DECLARE
    curr_pos INTEGER;
    is_parent INTEGER;
BEGIN
    IF (SELECT COUNT(*) FROM product WHERE prod_id = new_prod_id) = 0
    OR (SELECT COUNT(*) FROM product WHERE prod_id = new_part_id) = 0
    THEN
        RETURN -1;
    END IF;
    is_parent := (WITH RECURSIVE parents(prod_id, part_id) AS (
        SELECT ps.prod_id, ps.part_id FROM product_specification ps WHERE
ps.part_id = new_prod_id
        UNION ALL
        SELECT ps.prod_id, ps.part_id FROM product_specification ps, parents p
WHERE ps.part_id = p.prod_id
    ) SELECT COUNT(*) FROM parents p WHERE p.prod_id = new_part_id);
    IF (SELECT COUNT(*) FROM product_specification WHERE prod_id = new_prod_id AND
part_id = new_part_id) = 0
    AND is_parent = 0
    THEN
        curr_pos := (SELECT COUNT(*) FROM product_specification WHERE prod_id =
new_prod_id) + 1;

```

```

        INSERT INTO product_specification (prod_id, pos_num, part_id, quantity)
        VALUES (new_prod_id, curr_pos, new_part_id, new_quantity);
        RETURN 1;
    ELSE
        RETURN 0;
END IF;
END;
$$ LANGUAGE plpgsql;`

const create = () => {
  pool.query(sql_create_table_func, [], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    console.log("Все 4 таблицы и функции успешно созданы!");

    // Вставим данные в таблицы
    const sql_insert = `/*Заполнение таблиц*/
    SELECT addEi('Штука', 'шт', 10);
    SELECT addEi('Вес', 'кг', 11);

    SELECT createRootClass('Изделие', 1);
    SELECT createClass('Диваны', 2, 1);
    SELECT createClass('Кресла', 2, 1);
    SELECT createClass('Детали', 1, 1);
    SELECT createClass('Прямые диваны', 2, 2);
    SELECT createClass('Угловые диваны', 2, 2);
    SELECT createClass('Простые кресла', 2, 3);
    SELECT createClass('Кресла-кровати', 2, 3);
    SELECT createClass('Механизмы', 1, 4);
    SELECT createClass('Царги', 1, 4);
    SELECT createClass('Подлокотники', 1, 4);
    SELECT createClass('Ножки', 1, 4);
    SELECT createClass('Спинки', 1, 4);
    SELECT createClass('Боковины', 1, 4);
    SELECT createClass('Подушки', 1, 4);
    SELECT createClass('Чехлы', 1, 4);
    SELECT createClass('Матрасы', 1, 4);
    SELECT createClass('Секции/модули', 1, 4);
    SELECT createClass('Фурнитура', 1, 4);
    SELECT createClass('Болты', 1, 19);
    SELECT createClass('Шайбы', 1, 19);
    SELECT createClass('Винты', 1, 19);
    SELECT createClass('Гайки', 1, 19);
    SELECT createClass('Ключи', 1, 19);

    SELECT createProduct('1', 'Кресло-реклайнер Хоган', 8);
    SELECT createProduct('2', 'Диван угловой Домо ПРО', 6);
    SELECT createProduct('3', 'Диван угловой серии Кларк', 6);

```

```

SELECT createProduct('4', 'Кресло Хюгге', 7);
SELECT createProduct('5', 'Кресло Муни', 7);
SELECT createProduct('6', 'Диван Ника', 5);
SELECT createProduct('7', 'Диван-кровать Литен', 5);
SELECT createProduct('8', 'Кресло-кровать Персей Nova', 8);

SELECT createProduct('Д32', 'Модуль Кресло', 18);
SELECT createProduct('Д06', 'Подлокотник', 11);
SELECT createProduct('ДФ07', 'Болт М8*60 DIN 933', 20);
SELECT createProduct('ДФ42', 'Шайба М8 DIN 9021 увеличенная', 21);
SELECT createProduct('ДФ84', 'Механизм трансформации', 9);
SELECT createProduct('FMP113', 'Болт 6*20 DIN 933 ', 20);
SELECT createProduct('FMP068', 'Болт 6*40 DIN 933', 20);
SELECT createProduct('FMP078', 'Болт 8*45 DIN 933', 20);
SELECT createProduct('FMP008', 'Винт 6*16 DIN 965', 22);
SELECT createProduct('FMP086', 'Винт 6*25 DIN 7420', 22);
SELECT createProduct('FMP004', 'Винт 6*50 DIN 7420', 22);
SELECT createProduct('FMP030', 'Винт 6*70 DIN 7420', 22);
SELECT createProduct('FMP045', 'Гайка 6 DIN 985', 23);
SELECT createProduct('FMP096', 'Гайка 8 DIN 985', 23);
SELECT createProduct('FOP053', 'Ключ шестигранный 4 DIN 911', 24);
SELECT createProduct('FMP103', 'Шайба 24*8 пластик', 21);
SELECT createProduct('FMP064', 'Шайба 6 DIN 9021 увеличенная', 21);
SELECT createProduct('Д37', 'Центральная секция', 18);
SELECT createProduct('Д04', 'Спинка', 13);
SELECT createProduct('Д33', 'Подушка приспинная', 15);
SELECT createProduct('ДФ98', 'Болт М8*30 DIN 933 ', 20);
SELECT createProduct('ДФ82', 'Шайба 24x8 (пнд)', 21);
SELECT createProduct('ДФ17', 'Гайка М8 самоконтр DIN 985', 23);
SELECT createProduct('ДФ114', 'Гайка М8 ERICSON', 23);
SELECT createProduct('ДФ53', 'Болт М8*40 DIN 933', 18);
SELECT createProduct('ДФ113', 'Ключ шестигранный №5 ', 18);
SELECT createProduct('Д100', 'Подушка сиденья', 15);
SELECT createProduct('ДФ126', 'Шайба пластмассовая d9/d40*2', 21);
SELECT createProduct('ДФ101', 'Болт 8*60 DIN 933', 20);
SELECT createProduct('Д15', 'Модуль Кресло', 18);
SELECT createProduct('РФ21', 'Рама спинки', 13);
SELECT createProduct('РФ24', 'Подъемный механизм', 9);
SELECT createProduct('РФ25', 'Винт М8x40', 22);
SELECT createProduct('РФ26', 'Винт М8x45', 22);
SELECT createProduct('РФ05', 'Гайка М8', 23);
SELECT createProduct('Д07', 'Чехол дивана', 16);
SELECT createProduct('Д20', 'Матрас', 17);
SELECT createProduct('Д01', 'Механизм трансформации', 9);
SELECT createProduct('ДФ08', 'Болт М8*70 DIN 933', 20);
SELECT createProduct('ДФ78', 'Винт М8*50 DIN 7985', 22);
SELECT createProduct('ДФ19', 'Гайка М8 DIN 6923', 23);
SELECT createProduct('ДФ11', 'Винт-потай М6*30', 22);
SELECT createProduct('ДФ54', 'Болт М8*55', 20);

```

```

SELECT createProduct('Т01', 'Ткань для подлокотника', 16);
SELECT createProduct('001', 'Основание подлокотника', 11);

SELECT addProdSpec('1', 'Д32', 1);
SELECT addProdSpec('1', 'Д06', 2);
SELECT addProdSpec('1', 'ДФ07', 2);
SELECT addProdSpec('1', 'ДФ42', 2);

SELECT addProdSpec('2', 'ДФ84', 1);
SELECT addProdSpec('2', 'FMP113', 3);
SELECT addProdSpec('2', 'FMP068', 2);
SELECT addProdSpec('2', 'FMP078', 2);
SELECT addProdSpec('2', 'FMP008', 4);
SELECT addProdSpec('2', 'FMP086', 4);
SELECT addProdSpec('2', 'FMP004', 4);
SELECT addProdSpec('2', 'FMP030', 2);
SELECT addProdSpec('2', 'FMP045', 7);
SELECT addProdSpec('2', 'FMP096', 2);
SELECT addProdSpec('2', 'FOR053', 1);
SELECT addProdSpec('2', 'FMP103', 2);
SELECT addProdSpec('2', 'FMP064', 8);
SELECT addProdSpec('2', 'Д04', 1);
SELECT addProdSpec('2', 'Д06', 2);

SELECT addProdSpec('3', 'Д37', 1);
SELECT addProdSpec('3', 'Д04', 1);
SELECT addProdSpec('3', 'Д33', 1);
SELECT addProdSpec('3', 'Д06', 2);
SELECT addProdSpec('3', 'ДФ98', 2);
SELECT addProdSpec('3', 'ДФ82', 10);
SELECT addProdSpec('3', 'ДФ42', 10);
SELECT addProdSpec('3', 'ДФ17', 2);
SELECT addProdSpec('3', 'ДФ114', 2);
SELECT addProdSpec('3', 'ДФ53', 2);
SELECT addProdSpec('3', 'ДФ113', 1);

SELECT addProdSpec('4', 'Д100', 1);
SELECT addProdSpec('4', 'Д04', 1);
SELECT addProdSpec('4', 'Д06', 2);
SELECT addProdSpec('4', 'ДФ42', 6);
SELECT addProdSpec('4', 'ДФ126', 4);
SELECT addProdSpec('4', 'ДФ101', 6);

SELECT addProdSpec('5', 'Д15', 1);
SELECT addProdSpec('5', 'Д100', 1);
SELECT addProdSpec('5', 'ДФ42', 4);
SELECT addProdSpec('Д15', 'Д04', 1);
SELECT addProdSpec('Д15', 'Д06', 2);
SELECT addProdSpec('Д06', 'Т01', 1);

```

```

SELECT addProdSpec('Д06', '001', 1);

SELECT addProdSpec('6', 'P021', 1);
SELECT addProdSpec('6', 'P024', 2);
SELECT addProdSpec('6', 'P005', 20);
SELECT addProdSpec('6', 'P025', 12);
SELECT addProdSpec('6', 'P026', 8);
SELECT addProdSpec('6', 'Д07', 1);
SELECT addProdSpec('6', 'Д20', 1);

SELECT addProdSpec('7', 'Д01', 1);
SELECT addProdSpec('7', 'Д07', 1);
SELECT addProdSpec('7', 'Д20', 1);
SELECT addProdSpec('7', 'Д04', 1);
SELECT addProdSpec('7', 'Д08', 4);
SELECT addProdSpec('7', 'Д078', 4);
SELECT addProdSpec('7', 'Д019', 2);
SELECT addProdSpec('7', 'Д042', 4);

SELECT addProdSpec('8', 'Д01', 1);
SELECT addProdSpec('8', 'Д07', 1);
SELECT addProdSpec('8', 'Д20', 1);
SELECT addProdSpec('8', 'Д04', 1);
SELECT addProdSpec('8', 'Д06', 2);
SELECT addProdSpec('8', 'Д042', 4);
SELECT addProdSpec('8', 'Д011', 3);
SELECT addProdSpec('8', 'Д054', 12);`

pool.query(sql_insert, [], (err, result) => {
  if (err) {
    return console.error(err.message);
  }
  console.log("Данные успешно вставлены!");
});
});
}

module.exports = {
  create
}

```

*Листинг A.2 – Основные запросы, используемые при работе с базой данных
(файл queries.js)*

```
const pool = require('./pool').pool;
const create = require('./creation').create;

// Запросы для отображения всех страниц
const getAllClasses = (req, res) => {
  const sql = `SELECT pc.class_id, pc.parent_class_id, pc.name, ei.short_name
FROM product_classifier AS pc
INNER JOIN ei ON pc.ei_id = ei.ei_id
ORDER BY pc.class_id ASC`;

  pool.query(sql, [], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("classes", { model: result.rows });
  });
};

const getAllProducts = (req, res) => {
  const sql = `SELECT p.prod_id, p.name AS prod_name, pc.name AS par_name,
ei.short_name
FROM product AS p
INNER JOIN product_classifier AS pc ON p.class_id = pc.class_id
INNER JOIN ei ON pc.ei_id = ei.ei_id
ORDER BY p.prod_id`;

  pool.query(sql, [], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("products", { model: result.rows });
  });
};

const getAllSpecs = (req, res) => {
  const sql = `SELECT pos_num, ps.prod_id, p.name AS p_name, ps.part_id, pt.name AS
part_name, quantity
FROM product_specification AS ps
INNER JOIN product AS p ON ps.prod_id = p.prod_id
INNER JOIN product AS pt ON ps.part_id = pt.prod_id`;

  pool.query(sql, [], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("specifications", { model: result.rows });
  });
};
```

```

    });
};

const getAllEis = (req, res) => {
    const sql = `SELECT *
    FROM ei
    ORDER BY ei_id ASC`;

    pool.query(sql, [], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("ei", { model: result.rows });
    });
};

// Запросы для работы с классами изделий
const getToCreateClass = (req, res) => {
    res.render("create-class", { model: {} });
};

const createClass = (req, res) => {
    const sql = `SELECT createClass($1, $2, $3)`;
    const newClass = [req.body.name, req.body.ei_id, req.body.parent_class_id];

    pool.query(sql, newClass, (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.redirect("/classes");
    });
};

const getForEditClass = (req, res) => {
    const id = req.params.class_id;
    const sql = `SELECT result_class_id AS class_id, result_name AS name,
    result_ei_id AS ei_id, result_parent_class_id AS parent_class_id
    FROM selectClass($1)`;

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("edit-class", { model: result.rows[0] });
    });
};

const editClass = (req, res) => {
    const id = req.body.class_id;

```



```

    const upd_class = [id, req.body.name, req.body.ei_id, req.body.parent_class_id];
    const sql = `SELECT (change_pc + change_pcparent = 2)::int AS change FROM
changePc($1, $2, $3), changePcParent($1, $4)`;

    pool.query(sql, upd_class, (err, result) => {
      if (err) {
        return console.error(err.message);
      }
      res.redirect("/classes");
    });
  });

const getForDelete = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT *
FROM product_classifier WHERE class_id = $1`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("delete-class", { model: result.rows[0] });
  });
};

const deleteClass = (req, res) => {
  const id = req.params.class_id;
  const sql = "SELECT deleteClass($1)";

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/classes");
  });
};

const deleteClassWithDep = (req, res) => {
  const id = req.params.class_id;
  const sql = "SELECT deleteClassAndDesc($1)";

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/classes");
  });
};

```

```

const getClassID = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT *
FROM product_classifier WHERE class_id = $1`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("selected-class", { model: result.rows[0] });
  });
};

const findClassDesc = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT result_class_id AS class_id, result_name AS name,
ei.short_name AS ei_name, result_parent_class_id AS parent_class_id
FROM selectPcChildren($1) AS spc
INNER JOIN ei ON spc.result_ei_id = ei.ei_id`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("find-class-desc", { model: result.rows });
  });
};

const findClassDescProd = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT *
FROM selectPcChildrenWithProducts($1) AS spcw
INNER JOIN ei ON spcw.result_ei_id = ei.ei_id`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("find-class-desc-prod", { model: result.rows });
  });
};

const findClassParent = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT result_class_id AS class_id, result_name AS name,
ei.short_name AS ei_name, result_parent_class_id AS parent_class_id
FROM selectPcParents($1) AS spp
INNER JOIN ei ON spp.result_ei_id = ei.ei_id`;

```

```

pool.query(sql, [id], (err, result) => {
  if (err) {
    return console.error(err.message);
  }
  res.render("find-parent-class", { model: result.rows });
});
};

const findClassProds = (req, res) => {
  const id = req.params.class_id;
  const sql = `SELECT *
FROM selectClassProd($1) AS scp
INNER JOIN product_classifier AS pc ON scp.result_class_id = pc.class_id`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("find-class-products", { model: result.rows });
  });
};

// Запросы для работы с изделиями
const getToCreateProduct = (req, res) => {
  res.render("create-product", { model: {} });
};

const createProduct = (req, res) => {
  const sql = `SELECT createProduct($1, $2, $3)`;
  const newProd = [req.body.prod_id, req.body.name, req.body.class_id];

  pool.query(sql, newProd, (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/products");
  });
};

const getForEditProduct = (req, res) => {
  const id = req.params.prod_id;
  const sql = `SELECT result_prod_id AS prod_id, result_name AS name,
result_class_id AS class_id
FROM selectProduct($1)`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
  })
};

```

```

    res.render("edit-product", { model: result.rows[0] });
  });
};

const editProduct = (req, res) => {
  const id = req.body.prod_id;
  const upd_class = [id, req.body.name, req.body.class_id];
  const sql = `SELECT (changeProduct + changeProductParent = 2)::int AS change
FROM changeProduct($1, $2), changeProductParent($1, $3)`;

  pool.query(sql, upd_class, (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/products");
  });
};

const getForDeleteProd = (req, res) => {
  const id = req.params.prod_id;
  const sql = `SELECT *
FROM product WHERE prod_id = $1`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("delete-product", { model: result.rows[0] });
  });
};

const deleteProduct = (req, res) => {
  const id = req.params.prod_id;
  const sql = "SELECT deleteProduct($1)";

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/products");
  });
};

const getProdID = (req, res) => {
  const id = req.params.prod_id;
  const sql = `SELECT *
FROM product WHERE prod_id = $1`;

  pool.query(sql, [id], (err, result) => {

```

```

    if (err) {
        return console.error(err.message);
    }
    res.render("selected-product", { model: result.rows[0] });
});
};

const findClasses = (req, res) => {
    const id = req.params.prod_id;
    const sql = `SELECT * FROM showClassTree($1);`;

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("find-classes", { model: result.rows });
    });
};

const findParents = (req, res) => {
    const id = req.params.prod_id;
    const sql = `SELECT * FROM getProductParents($1);`;

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("find-parents", { model: result.rows });
    });
};

// Запросы для работы с единицами измерений
const getToCreateEi = (req, res) => {
    res.render("create-ei", { model: {} });
};

const createEi = (req, res) => {
    const sql = `SELECT addEi($1, $2, $3);`;
    const newEi = [req.body.name, req.body.short_name, req.body.code];

    pool.query(sql, newEi, (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.redirect("/ei");
    });
};

const getForEditEi = (req, res) => {

```

```

    const id = req.params.ei_id;
    const sql = `SELECT result_ei_id AS ei_id, result_name AS name, result_short_name
AS short_name, result_code AS code
FROM selectEi($1);`;

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("edit-ei", { model: result.rows[0] });
    });
};

const editEi = (req, res) => {
    const id = req.body.ei_id;
    const upd_class = [id, req.body.name, req.body.short_name, req.body.code];
    const sql = `SELECT changeEi($1, $2, $3, $4);`;

    pool.query(sql, upd_class, (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.redirect("/ei");
    });
};

const getForDeleteEi = (req, res) => {
    const id = req.params.ei_id;
    const sql = `SELECT *
FROM ei WHERE ei_id = $1`;

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.render("delete-ei", { model: result.rows[0] });
    });
};

const deleteEi = (req, res) => {
    const id = req.params.ei_id;
    const sql = "SELECT deleteEi($1)";

    pool.query(sql, [id], (err, result) => {
        if (err) {
            return console.error(err.message);
        }
        res.redirect("/ei");
    });
};

```

```

};

// Запросы для работы со спецификациями
const getToCreateSpec = (req, res) => {
  res.render("create-spec", { model: {} });
};

const createSpec = (req, res) => {
  const sql = `SELECT addProdSpecWithCheckLoops($1, $2, $3);`;
  const newSpec = [req.body.prod_id, req.body.part_id, req.body.quantity];

  pool.query(sql, newSpec, (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.redirect("/specifications");
  });
};

const findSpec = (req, res) => {
  const id = req.params.prod_id;
  const sql = `SELECT * FROM selectProdSpec($1);`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }

    res.render("selected-spec-prod", { model: result.rows });
  });
};

const findCostForm = (req, res) => {
  const id = req.params.prod_id;
  const sql = `SELECT *
FROM product WHERE prod_id = $1`;

  pool.query(sql, [id], (err, result) => {
    if (err) {
      return console.error(err.message);
    }
    res.render("find-cost-form", { model: result.rows[0] });
  });
};

const findCost = (req, res) => {
  const id = req.body.prod_id;
  const sql = `SELECT prod_id, name, SUM(quantity) AS quantity, result_ei_name
FROM countNeeds($1, $2)`

```

```

GROUP BY prod_id, name, result_ei_name`;

pool.query(sql, [id, req.body.amount], (err, result) => {
  if (err) {
    return console.error(err.message);
  }
  res.render("find-cost", { model: result.rows });
});
};

module.exports = {
  pool,
  create,
  getAllClasses,
  getAllProducts,
  getAllEis,
  getAllSpecs,

  getClassID,
  findClassDesc,
  findClassDescProd,
  findClassParent,
  findClassProds,
  getToCreateClass,
  createClass,
  getForEditClass,
  editClass,
  getForDelete,
  deleteClass,
  deleteClassWithDep,

  getToCreateProduct,
  createProduct,
  getForDeleteProd,
  deleteProduct,
  getForEditProduct,
  editProduct,
  getProdID,
  findClasses,
  findParents,

  getToCreateEi,
  createEi,
  getForEditEi,
  editEi,
  getForDeleteEi,
  deleteEi,

  getToCreateSpec,

```



```
createSpec,  
findSpec,  
findCostForm,  
findCost  
}
```

Листинг А.3 – Основной код для работы с сервером (файл index.js)

```
const express = require("express");
const db = require('./scripts/queries');
const path = require("path");

const app = express();

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));
app.use(express.static(path.join(__dirname, "public")));
app.use(express.urlencoded({ extended: false }));

db.create();

const PORT = process.env.PORT || 8081;
app.listen(PORT, () => {
  console.log(`Server started ${PORT}.`);
});

app.get("/", (req, res) => {
  res.render("index");
});

// Отображения основных страниц
app.get("/classes", db.getAllClasses);
app.get("/products", db.getAllProducts);
app.get("/ei", db.getAllEis);
app.get("/specifications", db.getAllSpecs);

// Работа с классами изделий
app.get("/selected-class/:class_id", db.getClassID);
app.get("/selected-class/find-class-desc/:class_id", db.findClassDesc);
app.get("/selected-class/find-class-desc-prod/:class_id", db.findClassDescProd);
app.get("/selected-class/find-parent-class/:class_id", db.findClassParent);
app.get("/selected-class/find-class-products/:class_id", db.findClassProds);

app.get("/create-class", db.getToCreateClass);
app.post("/create-class", db.createClass);

app.get("/edit-class/:class_id", db.getForEditClass);
app.post("/edit-class/:class_id", db.editClass);

app.get("/delete-class/:class_id", db.getForDelete);
app.post("/delete-class/:class_id", (req, res) => {
  if (req.body.action === "Удалить") {
    db.deleteClass(req, res);
  } else if (req.body.action === "Удалить с потомками") {
    db.deleteClassWithDep(req, res);
  } else {

```

```

        console.error("Action not found!");
    }
});

// Работа со списком изделий
app.get("/create-product", db.getToCreateProduct);
app.post("/create-product", db.createProduct);

app.get("/delete-product/:prod_id", db.getForDeleteProd);
app.post("/delete-product/:prod_id", db.deleteProduct);

app.get("/edit-product/:prod_id", db.getForEditProduct);
app.post("/edit-product/:prod_id", db.editProduct);

app.get("/selected-product/:prod_id", db.getProdID);
app.get("/selected-product/find-classes/:prod_id", db.findClasses);
app.get("/selected-product/find-parents/:prod_id", db.findParents);

// Работа с единицами измерений
app.get("/create-ei", db.getToCreateEi);
app.post("/create-ei", db.createEi);

app.get("/edit-ei/:ei_id", db.getForEditEi);
app.post("/edit-ei/:ei_id", db.editEi);

app.get("/delete-ei/:ei_id", db.getForDeleteEi);
app.post("/delete-ei/:ei_id", db.deleteEi);

// Работа со спецификациями
app.get("/create-spec", db.getToCreateSpec);
app.post("/create-spec", db.createSpec);

app.get("/selected-spec-prod/:prod_id", db.findSpec);

app.get("/find-cost-form/:prod_id", db.findCostForm);
app.post("/find-cost/:prod_id", db.findCost);

```

Листинг А.4 – Подключение к базе данных (файл pool.js)

```
const { Pool } = require("pg");

const pool = new Pool({
  user: "postgres",
  host: "localhost",
  database: "mispris",
  password: "1324",
  port: 5432
});
console.log("Подключение к Базе Данных прошло успешно!");

module.exports = {
  pool
}
```