

Министерство просвещения Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**“ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ  
ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ”**

**ФАКУЛЬТЕТ ИНФОРМАТИКИ И ЭКОНОМИКИ**

Кафедра прикладной информатики, информационных систем и технологий

Выпускная квалификационная работа

**РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ  
«СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ  
РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ»**

Работу выполнил:  
студент группы z1253  
направления 09.03.03  
«Прикладная информатика»  
профиль «Прикладная  
информатика»  
Черепанов Александр Михайлович

---

(подпись)

«Допущен к защите»  
и.о. зав. кафедрой  
Казаринова Наталья Леонидовна

---

(подпись)

«\_\_\_» \_\_\_\_\_ 2024г.

Научный руководитель:  
Главный тренер по региональному  
федерации по спортивному  
программированию  
Кудреватых Виталий Анатольевич

---

(подпись)

ПЕРМЬ  
2024

## Оглавление

<b>ВВЕДЕНИЕ.....</b>	<b>5</b>
<b>ГЛАВА 1. АНАЛИЗ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ПОИСКЕ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ.....</b>	<b>10</b>
1.1 Особенности и методы поиска датчиков в базах данных магазинов.....	10
1.2 Анализ существующих баз данных и онлайн-ресурсов по электронным компонентам и датчикам.....	11
1.3 Информационная и инфологическая модель.....	12
1.3.1 Информационная модель.....	12
1.3.2 Инфологическая модель.....	12
1.4 Формирование функциональных и нефункциональных требований к проектированию приложения по поиску датчиков в базах данных магазинов электронных компонентов.....	15
Вывод по главе 1.....	21
<b>ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ».</b>	<b>22</b>
2.1 Техническое задание на разработку программного обеспечения «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ».....	22
2.2 Проектирование программного обеспечения «Схемотехническое и программное обеспечение реализации робототехнических решений».....	23
2.2.1 Объектное проектирование.....	23
2.3 Разработка программного обеспечения «Схемотехническое и программное обеспечение реализации робототехнических решений».....	33
2.3.1 Разработка программы-парсера и реализация базы данных.....	33
2.3.2 Разработка клиентского приложения.....	40
2.4 Подготовка, ввод и систематизация базы данных датчиков по выбранным магазинам..	62
Вывод по главе 2.....	63
<b>ГЛАВА 3. ФИНАНСОВОЕ И ДОКУМЕНТАЦИОННОЕ ОБЕСПЕЧЕНИЕ РАЗРАБОТКИ ПО «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ». СОСТАВЛЕНИЕ БЮДЖЕТА ПРОЕКТА. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ ПО. НАПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОЙ ДОКУМЕНТАЦИИ.....</b>	<b>64</b>
3.1 Документационное обеспечение.....	64
<b>ЗАКЛЮЧЕНИЕ.....</b>	<b>67</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....</b>	<b>69</b>
<b>ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....</b>	<b>71</b>
<b>1. ВВЕДЕНИЕ.....</b>	<b>73</b>
Описание.....	73
Уровень подготовки пользователя.....	73

Перечень эксплуатационной документации.....	73
<b>2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ.....</b>	<b>74</b>
Назначение.....	74
Условия применения.....	74
<b>3. ПОДГОТОВКА К РАБОТЕ.....</b>	<b>75</b>
Первоначальная настройка.....	75
<b>4. ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ.....</b>	<b>76</b>
<b>5. АВАРИЙНЫЕ СИТУАЦИИ.....</b>	<b>77</b>
Недоступность системы.....	77
Аппаратные сбои.....	77
<b>6. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ.....</b>	<b>78</b>
<b>ПРИЛОЖЕНИЕ 2 РУКОВОДСТВО АДМИНИСТРАТОРА.....</b>	<b>81</b>
<b>1. НАЗНАЧЕНИЕ СИСТЕМЫ.....</b>	<b>84</b>
Назначение ПО.....	84
Функции системы.....	84
<b>2. УСЛОВИЯ РАБОТЫ СИСТЕМЫ.....</b>	<b>85</b>
Требования к аппаратуре и программным средствам.....	85
Требования к компетенциям администратора.....	85
<b>3. УСТАНОВКА И НАСТРОЙКА СИСТЕМЫ.....</b>	<b>86</b>
<b>4. СОЗДАНИЕ РЕЗЕРВНОЙ КОПИИ И ВОССТАНОВЛЕНИЕ СИСТЕМЫ.....</b>	<b>87</b>
Создание резервной копии.....	87
Восстановление работоспособности.....	87
<b>5. ПОРЯДОК ЗАПУСКА И ОСТАНОВКИ.....</b>	<b>88</b>
Порядок запуска.....	88
Порядок остановки.....	88
<b>6. АВАРИЙНЫЕ СИТУАЦИИ.....</b>	<b>89</b>
Ошибка установки соединения.....	89
Ошибка установки сторонних пакетов пакетным менеджером.....	89
<b>7. СООБЩЕНИЯ АДМИНИСТРАТОРУ.....</b>	<b>90</b>
<b>ПРИЛОЖЕНИЕ 3. МЕТОДИКА ИСПЫТАНИЙ.....</b>	<b>93</b>
<b>1. ОБЩИЕ ПОЛОЖЕНИЯ.....</b>	<b>96</b>
1.1. Основные понятия и определения.....	96
<b>2. ОБЪЕКТ ИСПЫТАНИЙ.....</b>	<b>97</b>
2.1. Наименование и условные обозначения.....	97
2.2. Цели, назначение системы.....	97
2.3. Комплектность испытательной системы.....	97
<b>3. Цель испытаний.....</b>	<b>98</b>
<b>4. Общие положения.....</b>	<b>99</b>
4.1. Перечень руководящих документов, на основании которых проводятся испытания..	99
4.2. Место проведения и продолжительность испытаний.....	99
4.3. Ведомства и организации, участвующие в испытаниях.....	99

4.4. Перечень предъявляемых на испытания документов.....	99
<b>5. ОБЪЕМ ИСПЫТАНИЙ.....</b>	<b>101</b>
5.1. Перечень испытаний.....	101
5.1.1. Перечень проводимых проверок по документации.....	101
5.1.2. Количественные и качественные характеристики, подлежащие оценке.....	101
5.2. Последовательность проведения испытаний.....	101
5.3. Перечень работ, проводимых после завершения испытаний.....	102
<b>6. УСЛОВИЯ И ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЙ.....</b>	<b>103</b>
6.1. Условия проведения испытаний.....	103
6.2. Условия начала и завершения отдельных этапов испытаний.....	103
6.3. Ограничения в условиях проведения испытаний.....	103
6.4. Требования к техническому обслуживанию.....	103
6.5. Меры, обеспечивающие безопасность и безаварийность проведения испытаний.....	103
6.6. Порядок взаимодействия организаций, участвующих в испытаниях.....	103
6.7. Требования к персоналу, проводящему испытания.....	104
<b>7. ТРЕБОВАНИЯ ПО.....</b>	<b>105</b>
<b>8. Материально-техническое обеспечение испытаний.....</b>	<b>106</b>
8.1. Технические средства, используемые во время испытаний.....	106
8.2. Программные средства, используемые во время испытаний.....	106
<b>9. Метрологическое обеспечение испытаний.....</b>	<b>107</b>

## **ВВЕДЕНИЕ**

В условиях стремительного развития цифровых технологий и национальной цифровой трансформации, мой проект призван внести вклад в сферу информационных технологий. Я осуществляю разработку приложения для упрощения процесса поиска датчиков в базах данных магазинов радиоэлементов с информацией о наличии и местонахождении.

В современных условиях, когда точность и оперативность получения данных становятся все более важными, необходимость автоматизированного и удобного инструмента для поиска компонентов становится очевидной. Моё приложение предназначено для предоставления пользователям возможности быстро находить нужные датчики, оптимизируя процесс поиска и обеспечивая доступ к актуальной информации о наличии товаров в различных магазинах.

Основной целью моего проекта является создание инновационного и полезного решения, которое сможет удовлетворить потребности как технических специалистов, так и обычных пользователей, нуждающихся в эффективных инструментах для работы с электронными компонентами. В процессе разработки мы используем современные методологии и лучшие практики, чтобы обеспечить высокое качество и удобство использования нашего приложения.

Приложение станет важным инструментом в развитии цифровой инфраструктуры и автоматизации процессов, способствуя достижению целей национальной программы цифровой трансформации. Благодаря этому решению, пользователи смогут значительно сэкономить время и усилия при поиске и приобретении необходимых датчиков, что, в свою очередь, положительно скажется на эффективности их работы и развитии технологической сферы в целом.

Актуальность выбранной темы подтверждает необходимость автоматизации процессов поиска и подбора датчиков в электронных магазинах. Разработка приложения для поиска датчиков в базах данных магазинов предоставляет удобный инструмент для пользователей, оптимизируя процессы поиска и обеспечивая доступ к актуальной информации о наличии товаров.

Проектирование данной информационной системы требует изучения специализированной литературы по автоматизации процессов поиска электронных компонентов, анализа существующих решений и методов интеграции с базами данных магазинов. Разработка такого приложения улучшит взаимодействие между покупателями и продавцами, повысит эффективность поиска и заказа необходимых датчиков, что особенно важно для технических специалистов и инженеров.

Целью данной работы является создание информационной системы, которая облегчит поиск и подбор датчиков, предоставляя пользователям доступ к информации о наличии и местонахождении товаров в различных магазинах Перми. Это позволит сократить время и усилия, затрачиваемые на поиск нужных компонентов, и повысит общую эффективность работы.

Объект исследования: процессы поиска и подбора электронных компонентов в базах данных магазинов.

Предмет исследования: использование информационных технологий для разработки и внедрения приложения, обеспечивающего эффективный поиск датчиков по заданным критериям.

В соответствии с объектом, предметом и целью данной работы были поставлены следующие задачи:

- Провести анализ существующих информационных систем-аналогов по поиску и подбору электронных компонентов;

- Разработать структуру базы данных для хранения информации о датчиках, включая наличие и местонахождение в различных магазинах;
- Выполнить проектирование дизайна приложения с учетом удобства использования и специфики представления данных о датчиках;
- Сформировать требования к информационной системе;
- Предложить модель информационной системы для поиска датчиков в базах данных магазинов;
- Провести сбор информации для ввода данных в базу, включая детали о датчиках и их наличии в магазинах;
- Разработать техническое задание, согласовать и утвердить его с заинтересованными сторонами;
- Провести расчеты финансового обеспечения разработки информационной системы;
- Реализовать прототип приложения;
- Провести тестирование приложения для выявления и устранения ошибок;
- Разработать программу и методику испытаний для сдачи системы в эксплуатацию;
- Разработать пользовательскую документацию: руководство пользователя и руководство администратора;
- Определить направления дальнейшего развития информационной системы: реализовать модуль для анализа спроса на датчики и автоматического обновления данных о наличии товаров в магазинах.

Актуальность разработки – приложение для поиска датчиков в базах данных магазинов, которое будет полезно широкому кругу пользователей, включая инженеров, технических специалистов, преподавателей, студентов и всех заинтересованных в электронике. Приложение позволит быстро находить необходимые датчики, предоставляя информацию о наличии и местонахождении в различных магазинах.

Информационная система актуальна для использования в образовательных учреждениях и на специализированных платформах, а также для личного пользования при самостоятельной работе с электронными компонентами.

В данной разработке будет использоваться итерационная модель жизненного цикла ИС, так как данная модель предоставляет возможности дополнять, модернизировать и уточнять отдельные модули, а также всю программу в целом.

Этапы разработки включают:

- Формирование требований;
- Проектирование;
- Реализация;
- Тестирование;
- Внедрение;
- Эксплуатация и сопровождение.

Эти аспекты подтверждают эффективность и доступность использования приложения. Приложение для поиска датчиков в базах данных магазинов позволит пользователям быстро находить необходимые компоненты, что повысит их эффективность в разработке электронных устройств. Данная идея является оригинальной и актуальной в настоящее время.

Работа состоит из введения, трех глав, заключения, библиографического списка и приложения.

- В первой главе описывается анализ существующих информационных систем для поиска электронных компонентов и сформированы функциональные и нефункциональные требования.



- Вторая глава посвящена проектированию и реализации информационной системы для поиска датчиков в базах данных магазинов Перми.

- В третьей главе рассчитано финансовое и документационное обеспечение разработки информационной системы для поиска датчиков.

# ГЛАВА 1. АНАЛИЗ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ПОИСКЕ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ

## 1.1 Особенности и методы поиска датчиков в базах данных магазинов

Разработка информационной системы для поиска датчиков включает анализ существующих технологий и методов, используемых для поиска и систематизации данных в базах данных магазинов электронных компонентов. Основное внимание уделяется критериям эффективности, доступности и точности поиска.

Особенности поиска датчиков:

1. **Каталогизация товаров:** Для успешного поиска необходимо, чтобы все датчики были правильно каталогизированный по типу.
2. **Обновляемость данных:** Важно, чтобы информация о наличии и местонахождении датчиков регулярно обновлялась, обеспечивая актуальность данных для пользователей.
3. **Интерфейс пользователя:** Удобный и интуитивно понятный интерфейс для поиска и навигации по базе данных, обеспечивающий простоту использования даже для пользователей с минимальными техническими навыками.
4. **Интеграция с магазинами:** Обеспечение интеграции с базами данных магазинов для автоматического обновления информации о наличии и местоположении датчиков.

5. **Поддержка различных форматов данных:** Способность системы работать с различными форматами данных, используемыми в базах данных магазинов, для обеспечения универсальности и гибкости.

Методы поиска:

1. **Поиск по категориям:** Возможность поиска по категориям товаров, что упрощает навигацию и поиск для пользователей.
2. **Поиск по местоположению:** Определение ближайших магазинов, имеющих в наличии требуемые датчики, с указанием точного местоположения на карте.

Данные аспекты подтверждают важность и актуальность разработки информационной системы для поиска датчиков в базах данных магазинов. Правильная реализация этих функций и методов обеспечит удобство и эффективность использования системы для широкого круга пользователей.

## **1.2 Анализ существующих баз данных и онлайн-ресурсов по электронным компонентам и датчикам.**

После тщательной проверки выяснилось, что не существует единой базы данных, содержащей информацию о датчиках, которые могли бы быть использованы в нашем проекте. Вместо этого были обнаружены лишь отдельные базы данных, хранящие информацию о товарах, включая датчики, в различных магазинах. Таким образом, для достижения наших целей необходимо будет объединить и систематизировать эту информацию из различных источников.

## 1.3 Информационная и инфологическая модель.

### 1.3.1 Информационная модель

Рассмотрим предложенную информационную модель данного приложения на рисунке 1

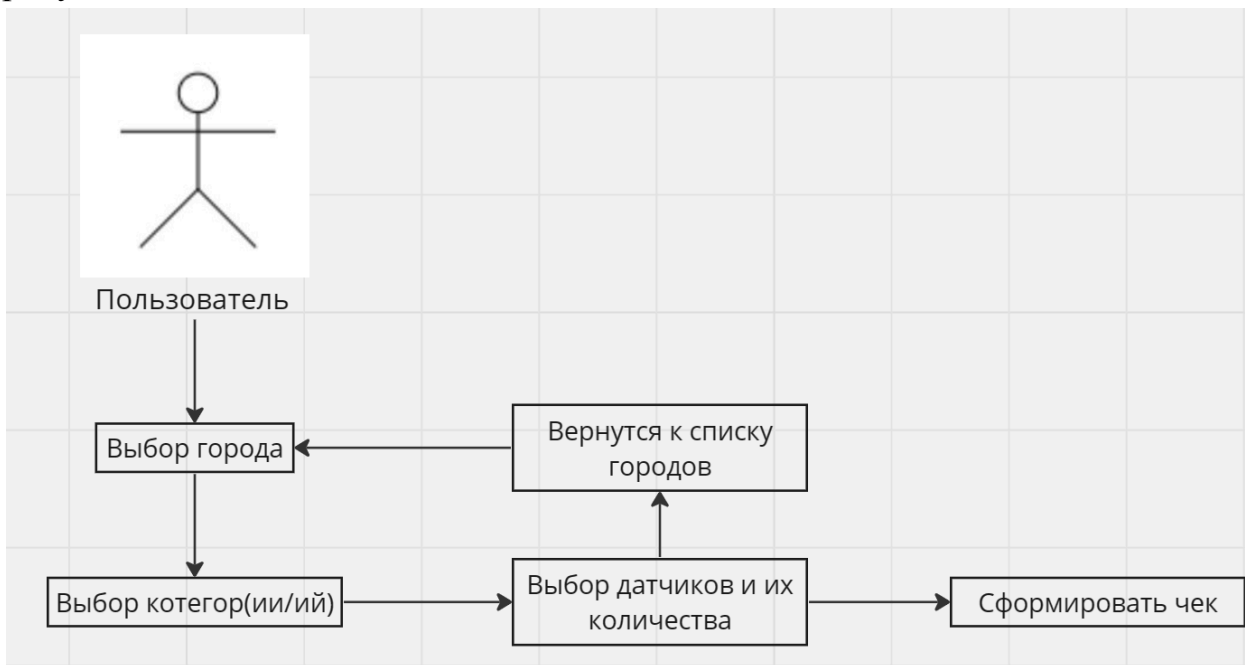


Рисунок 1. Информационная модель

### 1.3.2 Инфологическая модель

Инфологическая модель для приложения по поиску датчиков в базах данных магазинов электронных компонентов представляет собой схему, отражающую структуру и взаимосвязи между объектами предметной области. Основными объектами модели являются датчики и их местоположение.

Каждый датчик имеет уникальное название и может обладать различными характеристиками, такими как тип, диапазон цены и т.д.. Кроме того, каждый датчик связан с конкретным магазином, в котором он доступен для приобретения.

Модель может быть представлена в виде таблицы, где каждый датчик представлены отдельными строками.

Каждый датчик должен быть уникально идентифицирован, чтобы обеспечить точность данных. Для этого можно использовать специальные идентификаторы, которые будут привязаны к каждому датчику.

Таким образом, инфологическая модель для приложения по поиску датчиков позволит пользователям быстро находить нужные датчики.

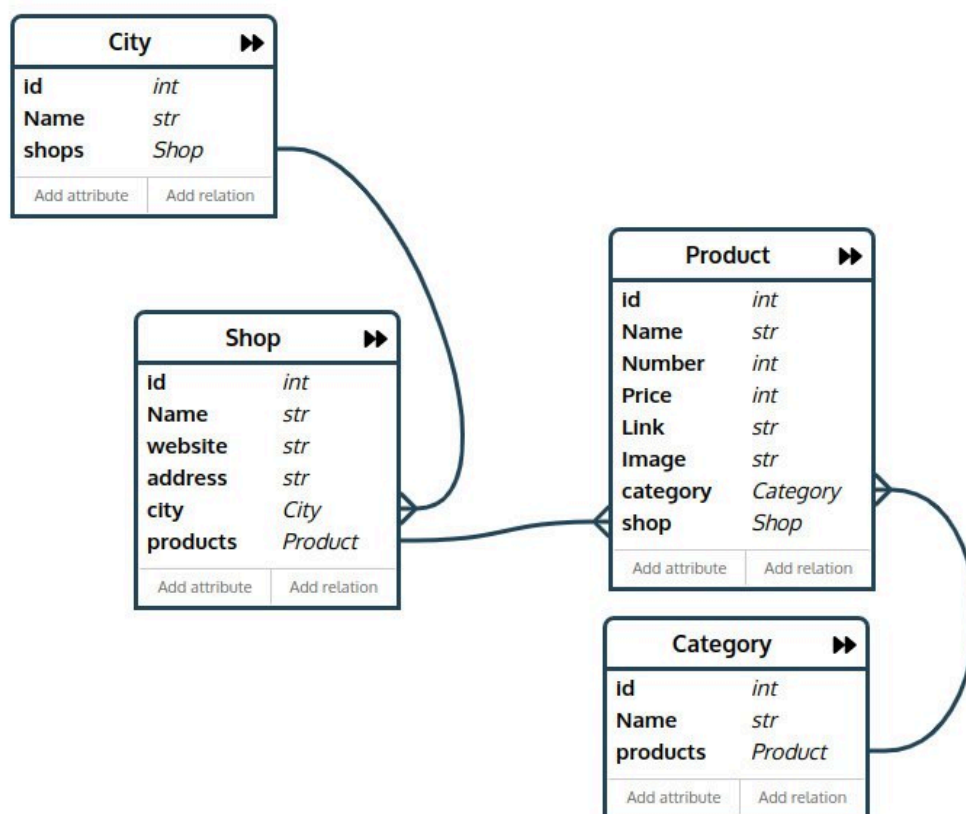


Рисунок 2. Инфологическая модель

Инфологическая модель будет использована во второй главе для реализации информационной системы.

### Инструментальные средства

Перейдем к выбору инструментальных средств для реализации ИС, а также к выбору СУБД в таблице 1.

Таблица 1. Выбор инструментальных средств и СУБД

Стек технологий	Описание
Django 5	Django — это фреймворк для веб-разработки на языке Python. Он предоставляет набор инструментов и библиотек для создания веб-приложений, основанных на принципах модели-шаблон-контроллер (MVC) и обеспечивает быстрое и эффективное создание веб-приложений.
СУБД PonyORM	PonyORM - это объектно-реляционный маппер (ORM) для Python, который обеспечивает удобный способ взаимодействия с базами данных, используя объектно-ориентированный подход. Он позволяет работать с базами данных, такими как SQLite, MySQL, PostgreSQL и другими, используя язык Python и не требуя напрямую писать SQL-запросы.

Данного стека технологий хватит для реализации системы

## **1.4 Формирование функциональных и нефункциональных требований к проектированию приложения по поиску датчиков в базах данных магазинов электронных компонентов.**

Приложение должно обладать следующими функциональностями:

1. Предоставление информации о доступных датчиках.
2. Классификация датчиков по типу.
3. Вывод (Сравнительный список датчиков для выбора и предложения решения)

Кроме того, приложение должно соответствовать следующим нефункциональным требованиям:

1. Быстрая и удобная навигация по приложению.
2. Оптимизированная производительность для обеспечения быстрой загрузки информации.
3. Поддержка различных операционных систем для максимального охвата аудитории.

Таким образом, приложение должно обеспечить удобство и функциональность для пользователей.

### **Функциональные требования.**

Функциональные требования определяют, каким должно быть поведение ИС в тех или иных условиях его эксплуатации.

Описание ролей пользователей приложения:

1. Пользователь - обеспечивает любому пользователю возможность зайти в данное приложение. Отдельные ресурсы для данного пользователя будут недоступны.

2. Модератор - системный администратор - имеет доступ к сформированной базе данных, с возможностью вносить в нее изменения. Управляет состоянием системы.

### **Функционал пользователя**

#### **1. Пользователь**

- Выбор города для сравнительного анализа.
- Формирование выборки датчиков из представленных категорий
- Формирование отчетного документа (чека) для выбранных единиц товаров.

#### **2. Модератор – системный администратор**

- Добавление в базу товаров из новых магазинов-партнеров.
- Редактирование имеющейся базы товаров.
- Управление состоянием приложения.

### **Нефункциональные требования**

Разработка информационной системы должна проводиться в соответствии с межгосударственными стандартами:

- ГОСТ Р 7.0.97-2016. Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Организационно-распорядительная документация. Требования к оформлению документов
- РД 50-34.698-90 Автоматизированные системы, требования к содержанию документов;
- ГОСТ 34.603-92 Виды испытаний автоматизированных систем;



- ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.
- Приказ Минпросвещения России, Рособрнадзора № 232/551 от 04.04.2023г. «Об утверждении Порядка проведения государственной итоговой аттестации по образовательным программам основного общего образования»
- Методические документы, рекомендуемые при организации и проведении государственной итоговой аттестации по образовательным программам основного общего и среднего общего образования в 2024 году (направлены письмом Рособрнадзора № 04–4 от 16.01.2024 г.)
- Методические документы, рекомендуемые при организации и проведении государственной итоговой аттестации по образовательным программам основного общего и среднего общего образования в 2024 году (направлены письмом Рособрнадзора № 04–28 от 06.02.2024 г.)

### **Пользовательский интерфейс**

Основные требования к пользовательским интерфейсам:

- Понятность и логичность;
- Функциональность (Соответствие требованиям пользователя);
- Обеспечение высокой скорости работы для пользователей;

### **Эскиз пользовательского интерфейса**

На рисунке (Рисунок 3) представлен эскиз пользовательского интерфейса для авторизованного пользователя.

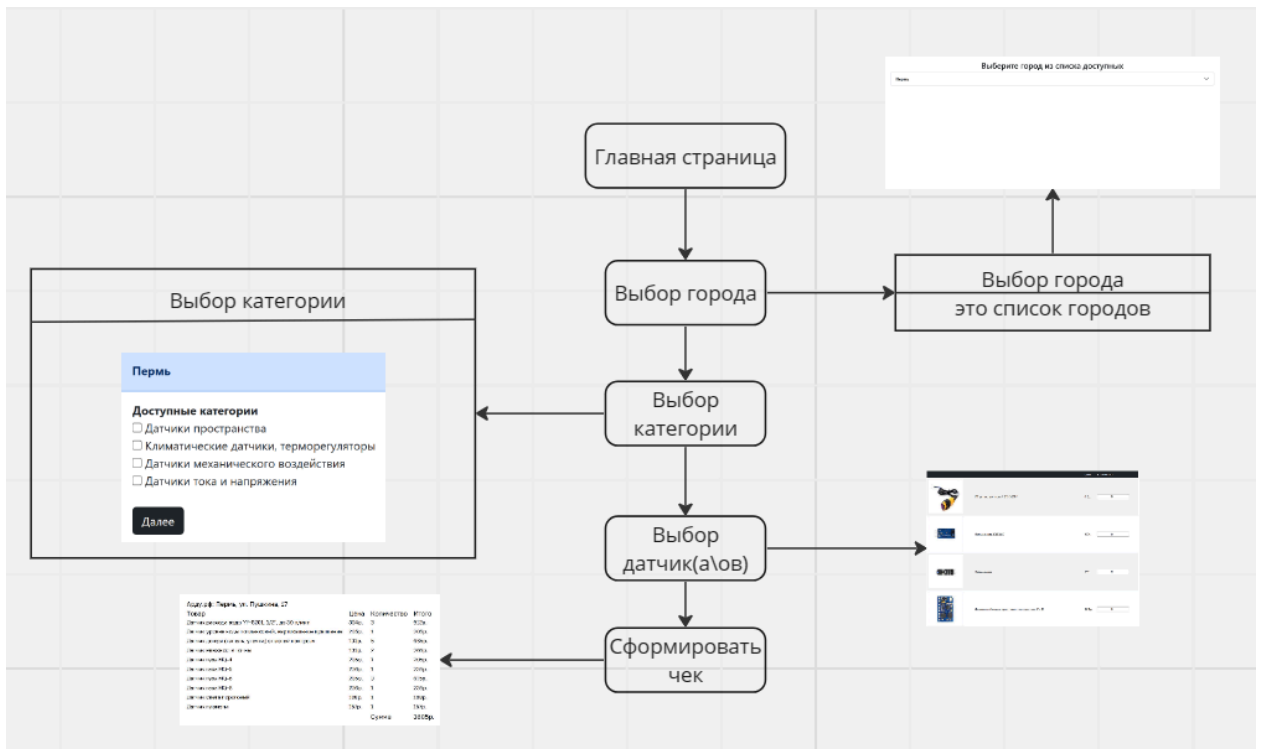


Рисунок 3. Эскиз интерфейса авторизованного пользователя

Как описывалось в пункте функциональных требований, на рисунке представлены возможности пользователя

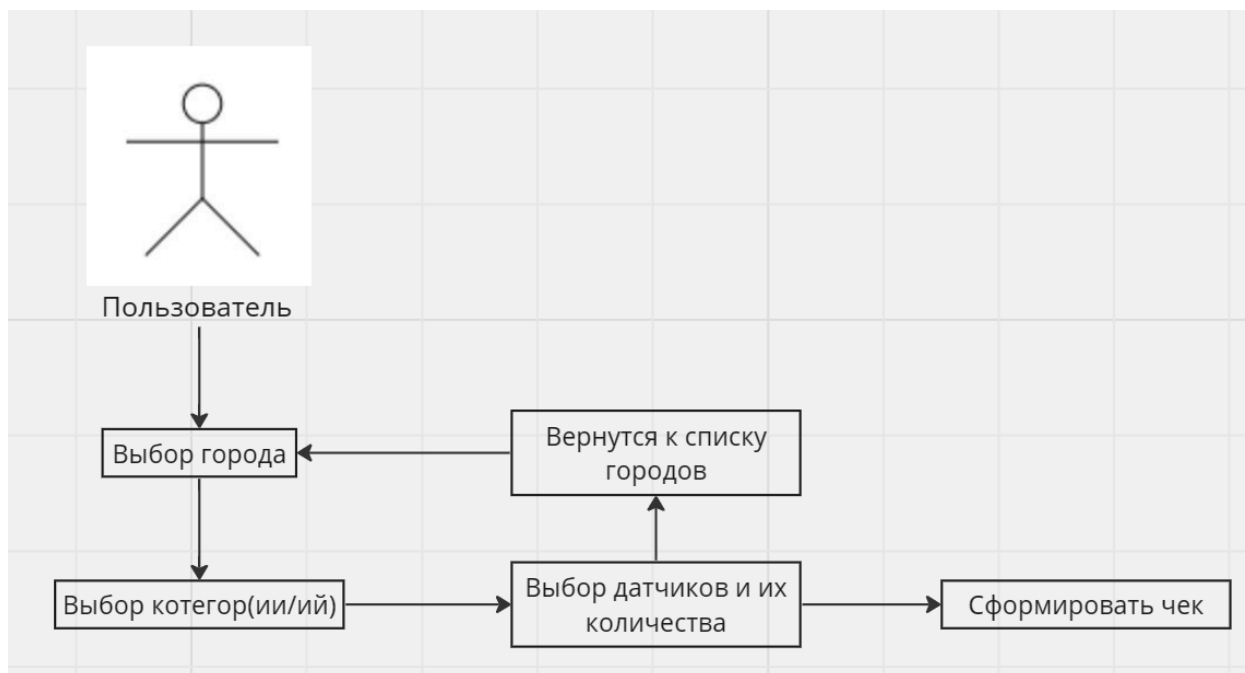


Рисунок 4. Схема сайта

Схема сайта детально расписана в таблицах 2, на рисунке представлена обобщённая система, которая включает в себя функционал пользователя.

Рассмотрим подробнее описание пунктов (Таблицы 3) пользовательского интерфейса проектируемой информационной системы.

Таблица 2. описание

Пункт	Описание
Выбор города	Пользователь выбирает город для поиска датчиков
Выбор категории	Пользователь выбирает категорию датчиков
Выбор датчиков	При нажатии происходит увеличение товара в корзине
Сформировать чек	При нажатии происходит формирование документа(рисунок 7)

Вернуться к списку городов

При нажатии пользователь  
возвращается к выбору города

### Сравнительный список датчиков для выбора и предложения решения

Арду.рф: Пермь, ул. Пушкина, 17

Товар	Цена	Количество	Итого
Датчик расхода воды YF-S201, 1/2", до 30 л/мин	304р.	3	912р.
Датчик уровня воды поплавковый, вертикальное крепление	205р.	1	205р.
Датчик дождя (капель, утечки) с платой контроля	133р.	5	665р.
Датчик влажности почвы	133р.	2	266р.
Датчик газа MQ-4	205р.	1	205р.
Датчик газа MQ-5	236р.	1	236р.
Датчик газа MQ-6	205р.	3	615р.
Датчик газа MQ-8	236р.	1	236р.
Датчик света пороговый	108р.	1	108р.
Датчик пламени	157р.	1	157р.
		Сумма	3605р.

### Рисунок 5. Чек

В таблицах 1 были подробно разобраны основной функционал интерфейса для авторизованного пользователя (Рисунок 2). Что позволит разработчику реализовать все описанные требования и функции.

## **Вывод по главе 1**

В данной главе проведен анализ существующих баз данных и систем для поиска датчиков. Рассмотрены базы данных таких магазинов, как Арду РФ, Товары Прикамья, Чип и Дип, Электронные компоненты. Было установлено, что единой базы датчиков не существует, и каждая база данных является отдельной для каждого магазина. На основании этого анализа предложена базовая модель информационной системы для поиска датчиков, включающая инфологическую модель. Рассмотрены процессы ввода данных о датчиках, предусмотрены возможности дополнения и редактирования информации. Сформированы функциональные и нефункциональные требования для различных видов работ: поиск датчиков, проверка наличия и местонахождения, отправка запроса на добавление и редактирование данных. Создан эскиз пользовательского интерфейса и описаны конкретные пункты для дальнейшей разработки проекта. Выбран стек технологий для реализации системы и приведены отечественные аналоги.

## **ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ»**

### **2.1 Техническое задание на разработку программного обеспечения «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ».**

Техническое задание (ТЗ) – главный технический документ, который определяет четкий набор требований к системе и утвержден как заказчиком или пользователем, так и исполнителем системы. Техническое задание является начальным этапом работ и составляется на все разработки и виды работ, необходимые для создания новой системы.

ТЗ должно устанавливать следующие показатели разрабатываемой системы:

- основное назначение, технические характеристики
- уровень стандартизации и унификации
- технико-экономические показатели
- патентно-правовые показатели;
- специальные требования к изделию и др.

В технических заданиях оговариваются этапы разработки и сроки выполнения каждого этапа, сроки разработки в целом.

При разработке используются следующие материалы:

- научно-техническая информация;
- патентная информация;
- характеристика рынка сбыта;

Техническое задание разрабатывается, как правило, организацией разработчиком. Сформулировать задачу максимально полно и грамотно, обосновать необходимость её решения – главная цель ТЗ. Исполнитель

выполняет его в контакте с заказчиком. Обязанность заказчика – предъявить разработчику исходные данные для разработки.

Полный текст технического задания, разработанного для программного обеспечения «схемотехническое и программное обеспечение реализации робототехнических решений» размещен в Приложении № 1.

## **2.2 Проектирование программного обеспечения «Схемотехническое и программное обеспечение реализации робототехнических решений».**

Проектирование информационной системы состоит из следующих этапов:

- Объектное проектирование;
- Концептуальное проектирование;

### **2.2.1 Объектное проектирование**

Объектное проектирование включает в себя проектирование объектов с атрибутами, связей между объектами с учетом семантики рассматриваемой предметной области (Таблицы 2-4) и заканчивается разработкой стратегической модели базы данных в виде ER – диаграммы.

В вашей базе данных определены следующие сущности: City, Shop, Category, и Product.

Ниже описаны их атрибуты в стиле примера:

#### **City**

Информация, хранимая о городах:

1. `name` (Название города)
2. `shops` (Набор магазинов в городе)

## **Shop**

Информация, хранимая о магазинах:

1. `name` (Название магазина)
2. `city\_id` (ИД города)
3. `website` (Веб-сайт магазина)
4. `address` (Адрес магазина)
5. `products` (Набор продуктов, доступных в магазине)

## **Category**

Информация, хранимая о категориях продуктов:

1. `name` (Название категории)
2. `products` (Набор продуктов в категории)

## **Product**

Информация, хранимая о продуктах:

1. `name` (Название продукта)
2. `number` (Номер продукта)
3. `category\_id` (ИД категории)
4. `shop\_id` (ИД магазина)
5. `price` (Цена продукта)
6. `product\_link` (Ссылка на продукт)
7. `image` (Изображение продукта)



Таблица 2. Объекты и атрибуты

Таблица	атрибут	Первичный ключ
<b>City</b> (город)	`name` (Название города)	<b>name</b> (Название города) - <b>уникальный ключ</b>
	`shops` (Набор магазинов в городе)	
<b>Shop</b> (магазин)	`name` (Название магазина)	<b>city_id</b> (ИД города) - <b>внешний ключ</b> (связан с <b>City</b> )
	`city_id` (ИД города)	
	`website` (Веб-сайт магазина)	
	`address` (Адрес магазина)	
	`products` (Набор продуктов, доступных в магазине)	
<b>Category</b> (категория продукта)	`name` (Название категории)	<b>name</b> (Название категории) - <b>уникальный ключ</b>
	`products` (Набор продуктов в категории)	
<b>Product</b> (продукт)	`name` (Название продукта)	<b>category_id</b> (ИД категории) - <b>внешний ключ</b> (связан с <b>Category</b> )  <b>shop_id</b> (ИД магазина) - <b>внешний ключ</b> (связан с <b>Shop</b> )
	`number` (Номер продукта)	
	`category_id` (ИД категории)	

	`shop_id` (ИД магазина)	
	`price` (Цена продукта)	
	`product_link` (Ссылка на продукт)	
	`image` (Изображение продукта)	

Таблица 3. Спецификация объектов

таблица	атрибут	Первичный ключ
<b>City</b> (город)	`name` (Название города)	Идентификационный атрибут
	`shops` (Набор магазинов в городе)	Описательный атрибут
<b>Shop</b> (магазин)	`name` (Название магазина)	Описательный атрибут
	`city_id` (ИД города)	Идентификационный атрибут
	`website` (Веб-сайт магазина)	Описательный атрибут
	`address` (Адрес магазина)	Описательный атрибут

	`products` (Набор продуктов, доступных в магазине)	Описательный атрибут
<b>Category</b> (категория продукта)	`name` (Название категории)	Идентификационный атрибут
	`products` (Набор продуктов в категории)	Описательный атрибут
<b>Product</b> (продукт)	`name` (Название продукта)	Описательный атрибут
	`number` (Номер продукта)	Описательный атрибут
	`category_id` (ИД категории)	Идентификационный атрибут
	`shop_id` (ИД магазина)	Идентификационный атрибут
	`price` (Цена продукта)	Описательный атрибут
	`product_link` (Ссылка на продукт)	Описательный атрибут
	`image` (Изображение продукта)	Описательный атрибут

### Концептуальное проектирование

Концептуальное проектирование – это первый этап проектирования информационной системы. На данном этапе описывается, что должна включать в себя проектируемая информационная система, определяются

объекты, их атрибуты и связи, с помощью которых составляется ER диаграмма предметной области (рис. 2).

Таблица 4. Описание связей

Наименование связи	Таблицы	Показатель кардинальности	Атрибут для связи
Выбор магазинов	City Shop	1 0	<b>city_id</b> (ИД города) - <b>внешний ключ</b> (связан с <b>City</b> )
Подключение к магазином	Shop Product	1 0	<b>shop_id</b> (ИД магазина) - <b>внешний ключ</b> (связан с <b>Shop</b> )
Подключение к категории	Category Product	1 0	<b>category_id</b> (ИД категории) - <b>внешний ключ</b> (связан с <b>Category</b> )

### Функциональное проектирование

Функциональное проектирование проводилось в среде RAMUS. Деятельность системы представлена контекстной диаграммой (рис. 5).

Деятельность регламентируется следующими гостами:

1. ГОСТ 33707-2016 Информационные технологии. Словарь.
2. ГОСТ 34.201-2020 «Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем»;

3. ГОСТ 34.603-92 «Информационная технология. Виды испытаний автоматизированных систем»;

Обеспечивают деятельность системы модераторы-системные администраторы. Началом для деятельности является текущее время и запросы пользователей к системе, а результатом деятельности вывод чека с выбранными датчиками.

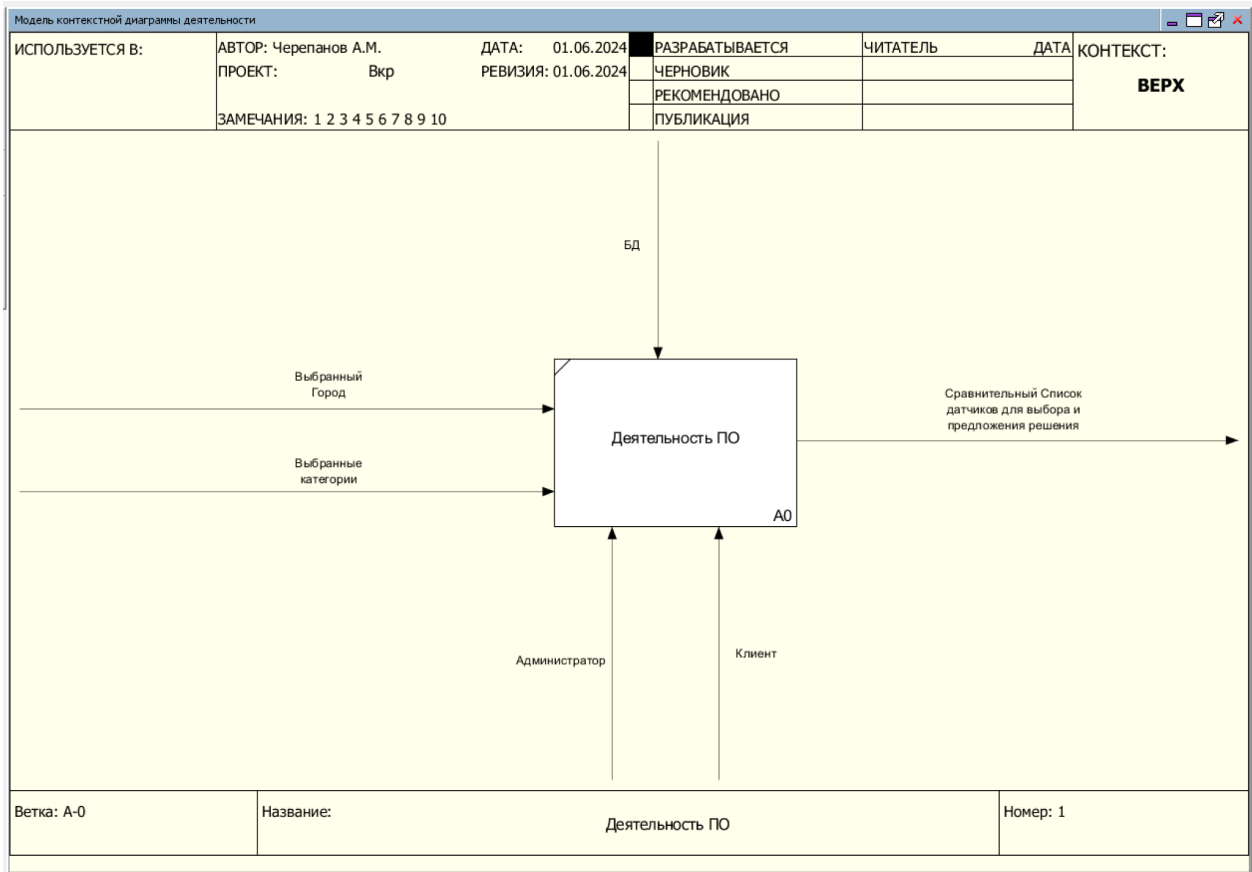


Рисунок 6. Модель контекстной диаграммы деятельности.

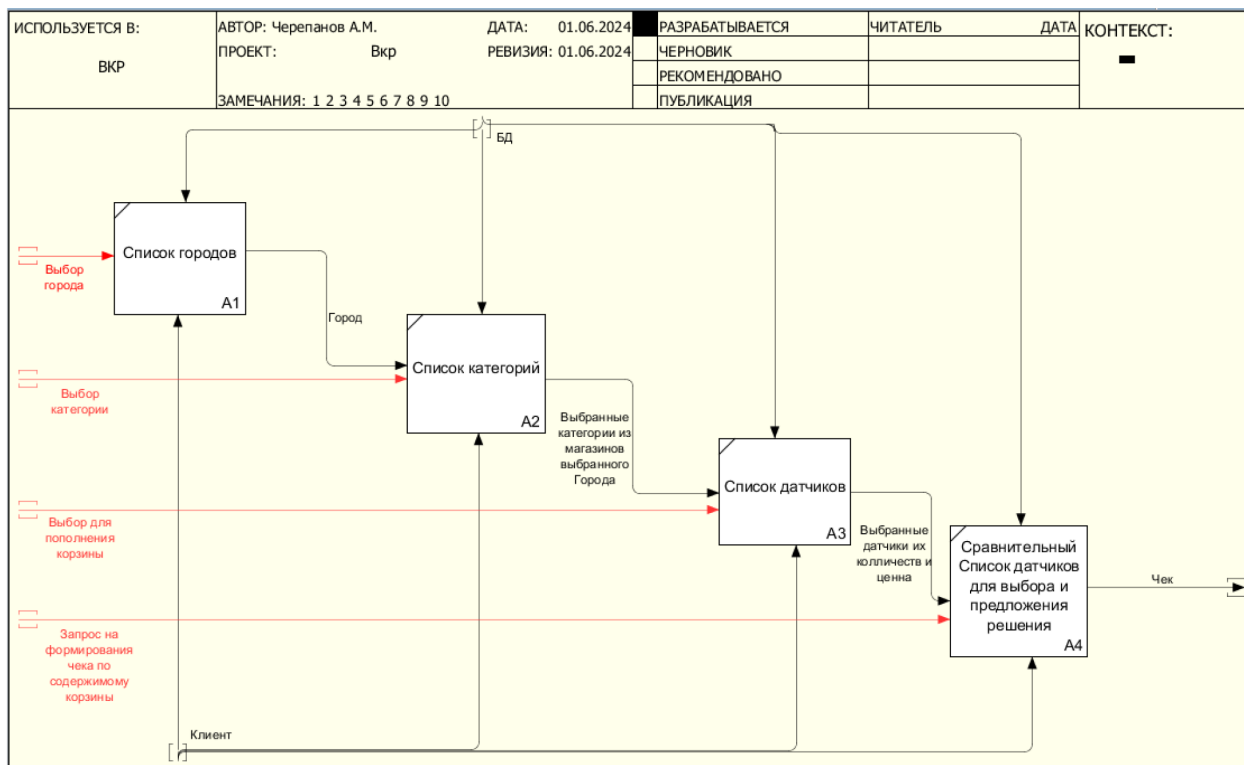


Рисунок 7. Модель декомпозиции контекстной диаграммы деятельности

Для автоматизации деятельности компании в проектируемой информационной системе рассматриваются бизнес-процессы. На рисунке 7 представлена диаграмма декомпозиции контекстной диаграммы.

Пользователь обращается к системе и производит выбор города и категорий. Далее выводится список на основе выбранных пунктов для выбора товара. При завершении выбора можно сформировать чек.

#### Диаграмма прецедентов

Пользователь:

- a) Выбор города
- b) Выбор категорий
- c) Выбор датчиков
- d) Формирование (Сравнительного Списка датчиков для выбора и предложения решения)

Модератор-системный администратор:

- a) Добавление Города
- b) Добавление Магазина

Таблица 5. Описание прецедентов

Прецедент	Краткое описание
Выбор города	Позволяет выбрать населенный пункт
Выбор категорий	Позволяет выбрать категории датчиков
Выбор датчиков	Просматривает информацию о датчиках из БД и позволяет добавить товары в корзину
Формирование (Сравнительного Списка датчиков для выбора и предложения решения)	Формирует чек на основе корзины пользователя
Добавление Города	Запускает модератор-системный администратор. Позволяет добавить в БД населённый пункт
Добавление Магазина	Запускает модератор-системный администратор. Позволяет добавить в БД магазин и дополнить код для парсинга данных с него

Созданная главная диаграмма прецедентов показана на рис.8:

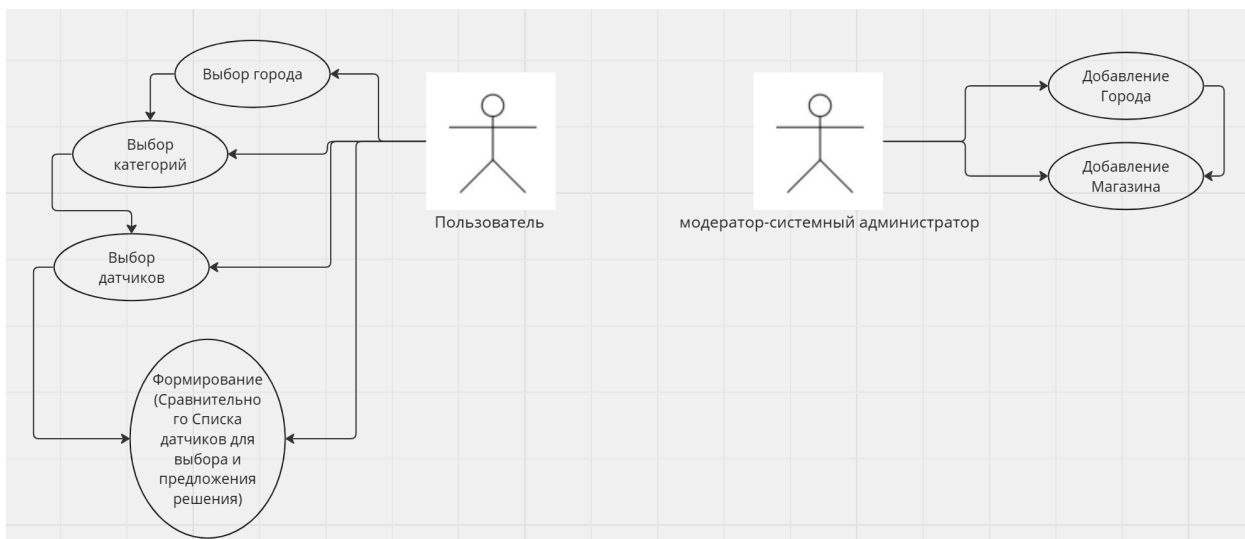


Рисунок 8. Главная диаграмма прецедентов

Пользователю предоставляются следующие возможности: определение города, категорий датчиков, выбор датчиков и формирование чека. Модератору-системному администратору доступна добавление и редактирование данных о городе и магазине, а также дополнение кода для парсинга данных с сайтов магазинов.

В данном пункте проведено проектирование информационной системы, включая разработку структуры базы данных и создание диаграмм по методологии IDEF0. Созданы главная и вспомогательные диаграммы прецедентов, отображающие функциональные возможности системы и взаимодействие пользователей с приложением.



## **2.3 Разработка программного обеспечения «Схемотехническое и программное обеспечение реализации робототехнических решений».**

В данной главе была рассмотрена реализация всех основных компонентов разрабатываемой информационной системы.

Система реализована с помощью фреймворка Django 5 для клиентской части и платформы PonyORM для серверной части приложения.

### **2.3.1 Разработка программы-парсера и реализация базы данных**

Программа-парсер непосредственно работает с базой данных приложения, в этом пункте будет рассмотрена конкретная реализация базы данных, программы-парсера и интерфейса их взаимодействия.

Для обновления БД необходимо запарсить (скачать, расшифровать и систематизировать) данные с сайтов, для этого надо запустить код в командной строке (`python app.py`)

База данных имеет две конечные точки. Формирование базы данных данных происходит на основе данных полученных с сайтов магазинов партнеров. Данные структурируются в соответствии со схемой данных, описанной в пункте 2.3.1. Модельное описание функциональной части базы данных средствами фреймворка PonyORM представлена в листинге 1.

## Листинг 1. Описание сущностей базы данных PonyORM

```
1. # Инициализация базы данных
2. db = Database()
3.
4. # Определение сущностей
5. class City(db.Entity):
6.     _table_ = 'product_city'
7.     name = Required(str)
8.     shops = Set('Shop')
9.
10. class Shop(db.Entity):
11.     _table_ = 'product_shop'
12.     name = Required(str)
13.     city_id = Required(City)
14.     website = Optional(str)
15.     address = Optional(str)
16.     products = Set('Product')
17.
18. class Category(db.Entity):
19.     _table_ = 'product_category'
20.     name = Required(str)
21.     products = Set('Product')
22.
23. class Product(db.Entity):
24.     _table_ = 'product_product'
25.     name = Required(str)
26.     number = Required(int)
27.     category_id = Required(Category)
28.     shop_id = Required(Shop)
29.     price = Optional(str)
30.     product_link = Optional(str)
31.     image = Optional(str)
32.
33. # Привязываем модели к базе данных
34. db.bind(provider='sqlite', filename='db.sqlite3',
35.         create_db=True)
36. # Генерируем схему базы данных
37. db.generate_mapping(create_tables=True)
```

Такие же модели определены в серверной части непосредственного приложения, средствами фреймворка Django5, что позволяет отделить сервис, получающий данные, от сервиса, реализующего непосредственно клиентское

приложение, используя общую базу данных. Такая схема позволяет достичь нужной гибкости в представлении данных, исключает ошибки синхронизации базы данных и, при своевременном создании резервных копий, сохраняет достаточную отказоустойчивость. Аналогичное описание моделей данных средствами Django5 представлено в листинге 2.

Листинг 2. Описание сущностей базы данных Django5

```
1. from django.db import models
2.
3. class City(models.Model):
4.     name = models.CharField(max_length=250,
5.                             unique=True)
6.
7.     class Meta:
8.         verbose_name_plural = 'Cities'
9.
10. class Category(models.Model):
11.     name = models.CharField(max_length=250,
12.                             unique=True)
13.
14.     class Meta:
15.         verbose_name_plural = 'Categories'
16.
17. class Shop(models.Model):
18.     name = models.CharField(max_length=250)
19.     city = models.ForeignKey(City,
20.                             on_delete=models.CASCADE,
21.                             related_name='shops')
22.     website = models.CharField(max_length=250,
23.                                blank=True, null=True)
24.     address = models.CharField(max_length=250,
25.                                 blank=True, null=True)
26.
27. class Product(models.Model):
28.     name = models.CharField(max_length=250)
29.     number = models.IntegerField()
30.     category = models.ForeignKey(Category,
31.                                  on_delete=models.CASCADE,
32.                                  related_name='products')
33.     shop = models.ForeignKey(Shop,
34.                              on_delete=models.CASCADE,
35.                              related_name='products')
```

```
35.     price = models.CharField(max_length=100,  
36.                               blank=True, null=True)  
37.     product_link = models.CharField(max_length=250,  
38.                                     blank=True,  
    null=True)  
39.     image = models.CharField(max_length=255,  
40.                               blank=True,  
41.                               null=True)
```

Непосредственное заполнение базы данных осуществляется с помощью технологического стека библиотек Selenium и Requests, а также вспомогательных BS4, PIL, ChromeDriverManager. Программа-парсер запускает сессию браузера (в нашем случае Google Chrome) и поочередно загружает в нем необходимые страницы из списка, формируемого администратором. Загрузка страниц в сессии браузера необходима для успешной подгрузки динамически-формируемого контента страницы, например категорий товаров, цен, информации о наличии и количестве товаров. С помощью библиотеки BeautifulSoup контент страницы разделяется на объекты согласно HTML-тегам, что позволяет эффективно найти нужную информацию о товарах, используя заготовленные CSS-селекторы. Для каждого магазина пишется своя функция-парсер, извлекающая данные о товарах. Функции запускаются последовательно в управляющем цикле, в который также подается общая информация о магазине. Устройство управляющего цикла показано в листинге 3.

### Листинг 3. Управляющий цикл программы-парсера

```
1. #Словарь с метайнформацией о магазинах
2. cities = ("Пермь",)
3. shops = {cities[0]:[
4.     ("Арду.рф", URL, "Пермь, ул. Пушкина, 17", urls_1,
5.     process_products_1),
6.     ("Радиодетали на Петропавловской",
7.     "https://www.radiodetali.perm.ru", "г.Пермь,
8.     ул.Петропавловская, 15", urls_2, process_products_2),
9. ]}
10. with db_session:
11.     for city in cities:
12.         city_obj = City.select(lambda c: c.name ==
13.         city).first()
14.         if not city_obj:
15.             city_obj = City(name=city)
16.         for shop in shops.get(city):
17.             name, website, address, links, parser = shop
18.             shop_obj = Shop.select(lambda s:
19.             s.city_id==city_obj and s.name==name and
20.             s.address==address).first()
21.             if not shop_obj:
22.                 shop_obj = Shop(name=name,
23.                 city_id=city_obj, website=website, address=address)
24.             for url in links:
25.                 driver.get(url)
26.                 # Даем время на загрузку динамического
27.                 контента
28.                 driver.refresh()
29.                 WebDriverWait(driver,
30.                 20).until(EC.presence_of_element_located((By.CSS_SELECTOR,
31.                 'div'))))
32.                 print(url)
33.                 #Парсим данные
34.                 parser(driver.page_source, shop_obj,
35.                 website)
```

Как видно из листинга 3, управляющая функция передает контент страницы в функцию-парсер, определенную для каждого объекта магазина. Уникальность функций обусловлена различиями в структуре сайтов и

уникальностью CSS-селекторов, которыми извлекаются данные. В каждой функции обеспечена достаточная отказоустойчивость, предусмотрены варианты отсутствия некоторых данных, в соответствии с практикой магазина. Пример функции-парсера для сайта магазина АРДУ.РФ представлен в листинге 4.

Листинг 4. Пример функции-парсера

```
1. # Функция для обработки товаров на странице АРДУ.РФ и
   # добавления их в базу данных
2. @db_session
3. def process_products_1(page_source, shop_obj, website=URL):
4.     soup = BeautifulSoup(page_source, 'html.parser')
5.     category = soup.find('h3', {'id': 'cat-title'}).text
6.     cat_obj = Category.select(lambda c:
7.                               c.name==category).first()
8.     if not cat_obj:
9.         cat_obj = Category(name=category)
10.    product_divs = soup.find_all('div', {'field':
11.                                     'link'})
12.    if not product_divs:
13.        return
14.
15.    product_number = 1 # Первый номер продукта
16.    for product_div in product_divs:
17.        product_name = product_div.text.strip()
18.        if not product_name:
19.            continue
20.        exists_tag =
21.            product_div.find_next_sibling('span', {'field': 'exists'})
22.        prod_obj = Product.select(lambda p:
23.                                   p.name==product_name and p.shop_id==shop_obj and
24.                                   p.category_id==cat_obj).first()
25.        if prod_obj and not (exists_tag and "Товар в
26.                             наличии" in exists_tag.text):
27.            prod_obj.delete()
28.
29.        elif exists_tag and "Товар в наличии" in
30.            exists_tag.text:
31.            image_div = product_div.find_previous('div',
32.                                                    {'field': 'picture'})
```

```

27.         media_path = ''
28.         if image_div:
29.             image_link =
30.                 website+image_div.find_next('img').get('src')
31.                 media_path = f'{product_name}.jpg'
32.                 media_path = 'media/' +
33.                 media_path.translate(str.maketrans(' ', '',
34.                 ''.join(set(string.punctuation) - set(safe_chars))))
35.                 with open(media_path, 'wb') as f:
36.                     f.write(requests.get(image_link).content)
37.                     image = Image.open(media_path)
38.                     image = image.resize((200,200))
39.                     image.save(media_path)
40.             price_span = product_div.find_next('span',
41.             {'field': 'price'})
42.             if price_span:
43.                 price = price_span.text.strip()
44.             else:
45.                 price = "Цена не указана."
46.
47.             # Извлекаем ссылку на товар
48.             product_link_element = product_div.find('a')
49.             if product_link_element:
50.                 product_link =
51.                 product_link_element['href']
52.                 full_product_link = website +
53.                 product_link
54.
55.             # Добавляем товар в базу данных
56.             if not prod_obj:
57.                 Product(name=product_name,
58.                 number=product_number, category_id = cat_obj,
59.                 shop_id=shop_obj, price=price,
60.                 product_link=full_product_link, image=media_path)
61.             else:
62.                 prod_obj.number = product_number
63.                 prod_obj.price = price
64.                 prod_obj.product_link =
65.                 full_product_link
66.                 prod_obj.image = media_path
67.             else:
68.                 # Добавляем товар в базу данных без
69.                 ссылки
70.                 if not prod_obj:
71.                     Product(name=product_name,
72.                     number=product_number, category_id=cat_obj,
73.                     shop_id=shop_obj, price=price, image=media_path)

```

```

61.                 else:
62.                     prod.obj.number = product_number
63.                     prod_obj.price = price
64.                     prod_obj.image = media_path
65.
66.                 product_number += 1 # Увеличиваем номер
    продукта

```

Как видно из примера, функция создает объект продукта, он сразу сохраняет все данные в базе, кроме того происходит загрузка и форматирование изображения товара. При возникновении ошибок в процессе загрузки данных, парсер прекращает свою работу, но это не влияет на работу клиентского приложения, так как архитектурно они разделены. На этом реализация программы парсера завершена, дальше будет рассмотрена реализация клиентской части приложения.

### 2.3.2 Разработка клиентского приложения

Клиентское приложение основано на фреймворке Django5. Приложение основано на трёх компонентах: модели, представлении и контроллере. Разберем каждый из них отдельно.

Модель это описание базы данных языком Django, она нужна для связи приложения со сформированной БД. Реализация модели представлена в листинге 2. Кроме того, Django позволяет делать запросы в базу данных прямо из браузера, для этого у пользователя должны быть права администратора, а модель должна быть зарегистрирована в административной панели Django. В процессе регистрации явно указываются поля модели, которые будут отображены в панели администратора, по которым будет производится поиск и фильтрация данных. Процесс регистрации показан в листинге 5.

Листинг 5. Регистрация моделей в панели администратора



```

1. from django.contrib import admin
2. from .models import City, Shop, Product, Category
3.
4. @admin.register(City)
5. class CityAdmin(admin.ModelAdmin):
6.     search_fields = ['name',]
7.     list_display = ['id', 'name', ]
8.
9. @admin.register(Shop)
10. class ShopAdmin(admin.ModelAdmin):
11.     search_fields = ['name', 'address', ]
12.     list_display = ['name', 'address', ]
13.
14. @admin.register(Product)
15. class ProductAdmin(admin.ModelAdmin):
16.     search_fields = ['name', ]
17.     list_display = ['name', 'number', 'price', ]
18.
19. @admin.register(Category)
20. class CategoryAdmin(admin.ModelAdmin):
21.     list_display = ['name', ]

```

Вид панели администратора показан на рисунке 9.

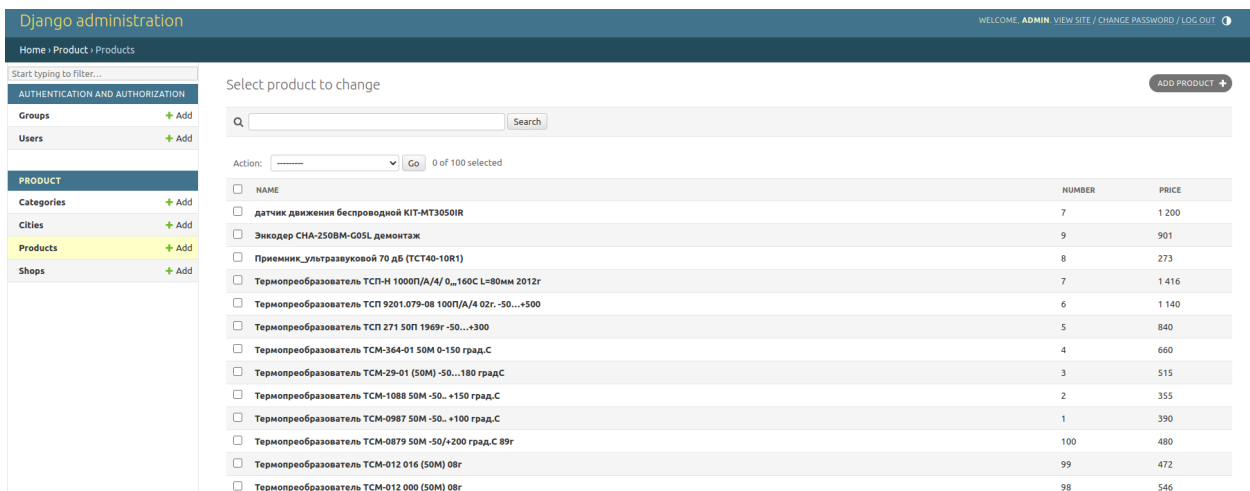


Рисунок 9. Вид панели администратора с открытой моделью “Продукты”

Контроллер это интерпретатор пользовательских действий. С его помощью приложение принимает команды от пользователя и отправляет ему ответ. В нашем случае командами служат HTTP-запросы, а контроллер представлен URL-диспетчером, возвращающим необходимые представления на конкретные запросы. Реализация контроллера представлена в листинге 6.

Листинг 6. Реализация контроллера

```
1. #product.urls
2. urlpatterns = [
3.     path('', views.city_list, name='city_list'),
4.     path('products/', views.product_list,
5.         name='product_list'),
6.     path('check/', views.form_check, name='check')
7. ]
8.
9.
10. #urls
11. urlpatterns = [
12.     path('admin/', admin.site.urls),
13.     path('', include('product.urls', namespace="product")),
14. ] + static(settings.MEDIA_URL,
15.     document_root=settings.MEDIA_ROOT)
```

Представление отвечает за ответ пользователю. Оно получает данные из модели и наполняет возвращаемые шаблоны страниц динамическим контентом. Представление - самая обширная часть кода приложения, так как оно включает в себя шаблоны страниц, обработку данных, связь с моделью и непосредственное представление динамического контента. Последовательно рассмотрим каждую функцию представление и возвращаемые веб-страницы.

Функция представления `city_list` (листинг 7) отвечает за вывод главной страницы сайта, представленной на рисунке 10. На этой странице представлен список городов, в которых доступны магазины.

## Листинг 7. Функция-представление главной страницы

```
1. def city_list(request):
2.     city_list = City.objects.all()
3.     category_list = set([prod.category for prod in
4.         Product.objects.all()])
5.     form = CategoriesForm()
6.     return render(request,
7.         'product/city_list.html',
8.         {'cities': city_list,
9.         'title': 'Доступные города',
10.        'form': form})
```

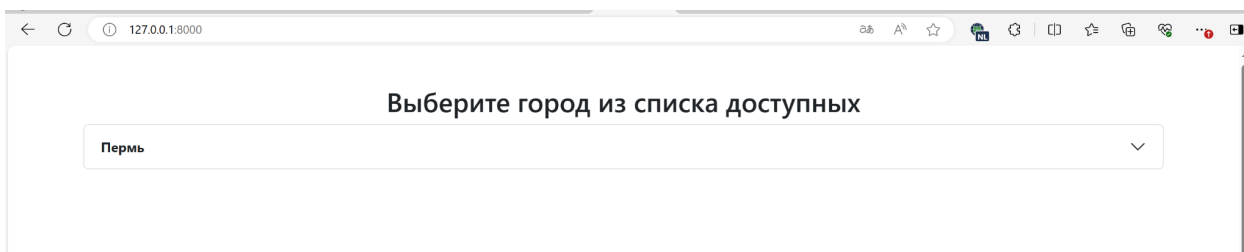


Рисунок 10. Главная страница сайта

На главной странице (рисунок 11) расположены Аккордеоны (вертикально сложенные выпадающие списки) с три в которых содержатся категории товаров.

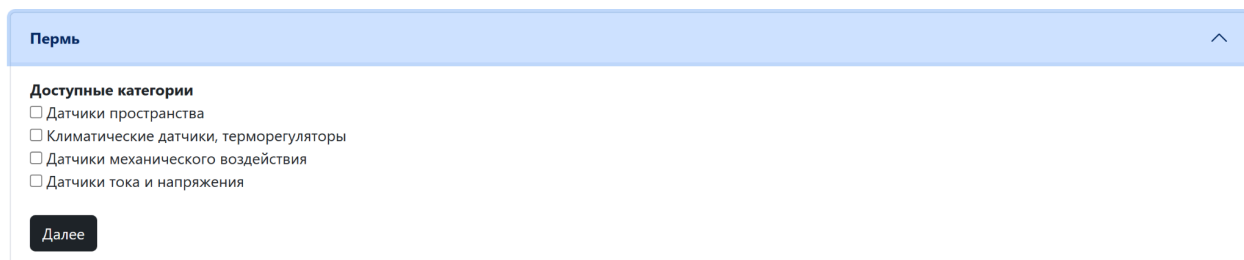


Рисунок 11. Категории товаров

На данной странице (рисунок 11) пользователь должен выбрать нужные ему категории и при нажатии на кнопку далее перейти к списку датчиков.

Шаблон страницы представлен на листинге 8. Шаблон наследует базовый шаблон сайта, включающий элементы, общие для всех страниц сайта (основная разметка, контейнер для содержимого, футер). Базовый шаблон представлен на листинге 9.

Листинг 8. Шаблон главной страницы сайта

```
1. {% extends "main.html" %}
2.
3. {% block title %} {{ title }} {% endblock %}
4.
5. {% block content %}
6.     {% if cities %}
7.         <h2 class="text-center">Выберите город из списка
           доступных</h2>
8.         <div class="accordion" id="CitiesAccordion">
9.             {% for city in cities %}
10.                 <div class="accordion-item">
11.                     <h2 class="accordion-header" id="heading{{ city.id }}">
12.                         <button class="accordion-button collapsed"
13.                             type="button" data-bs-toggle="collapse"
14.                             data-bs-target="#collapse{{ city.id }}" aria-expanded="false"
15.                             aria-controls="collapse{{ city.id }}">
16.                             <strong>{{ city.name }}</strong>
17.                         </button>
18.                     </h2>
19.                     <div id="collapse{{ city.id }}"
20.                         class="accordion-collapse collapse" aria-labelledby="heading{{ city.id }}" data-bs-parent="#CitiesAccordion">
21.                         <div class="accordion-body">
22.                             <strong>Доступные категории</strong>
23.                             <form method="POST" action="{% url
24.                                 'product:product_list' %}?city={{ city.id }}">
25.                                 {{ form.Categories }}
26.                                 {% csrf_token %}
27.                                 <br>
28.                                 <input class="btn btn-primary btn-dark"
29.                                     type="submit" value="Далее">
30.                             </form>
31.                         </div>
32.                     </div>
33.                 </div>
34.             {% endfor %}
35.         </div>
36.     {% else %}
```

```
31.     <p>Извините, в данный момент доступных городов нет</p>
32.     {% endif %}
33. {% endblock %}
```

Листинг 9. Базовый шаблон сайта

```
1. <!DOCTYPE html>
2. <html lang="en">
3.   <head>
4.     <meta charset="utf-8">
5.     <title>{% block title %}{% endblock %}</title>
6.     {% load django_bootstrap5 %}
7.     {% bootstrap_css %}
8.     {% bootstrap_javascript %}
9.     {% block links %}{% endblock %}
10.  </head>
11.  <body>
12.    <div>
13.      <!-- Page content -->
14.      <div class="container mt-5 h-75 min-vh-100">
15.        {% block content %}
16.        {% endblock %}
17.      </div>
18.
19.      <!-- Footer -->
20.      <footer class="py-5 bg-dark mt-2">
21.        <div class="container">
22.          <p class="m-0 text-center text-white">Copyright
    &copy; My project 2024, Powered by Django 5</p>
23.        </div>
24.      </footer>
25.    </div>
26.  </body>
27. </html>
```

Как можно видеть из листингов 7 и 8, функция наполняет шаблон списком городов динамически. Также в шаблон передается форма для выбора категорий, получающая список всех возможных категорий из базы данных. Код формы представлен на листинге 10. Django предоставляет инструментарий для автоматического создания форм и представления их в шаблонах. Данные о выбранных категориях передаются в теле POST-запроса,

а в самом запросе параметрами передается идентификатор выбранного города (типичный HTTP запрос при таком действии будет выглядеть как POST `http://localhost:8000/products/?city=1`, а в теле запроса будет следовать список выбранных категорий).

Листинг 10. Форма для выбора категорий

```
1. from django import forms
2. from .models import Category
3.
4. class CategoriesForm(forms.Form):
5.     choices = tuple(enumerate([cat.name for cat in
6.     Category.objects.all()], 1))
7.     Categories = forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple,
8.     choices=choices)
```

При получении такого POST-запроса программа возвращает пользователю страницу со списком датчиков из выбранных категорий (рисунок 12). Шаблон данной страницы показан на листинге 11.



	цена	количество
	Датчик тока 20А	242р. <input type="text" value="0"/>
	Датчик тока 5А	205р. <input type="text" value="0"/>
Вернуться к списку городов		Сформировать чек

Рисунок 12. Список датчиков

## Листинг 11. Шаблон страницы со списком датчиков

```

1. {% extends "main.html" %}
2. {% load static %}
3. {% block title %}{{ title }}{% endblock %}
4. {% block content %}
5.     <div class="container">
6.         <form action="{% url "product:check" %}" method="POST">
7.             {% csrf_token %}
8.             <table class="table table-striped">
9.                 <thead class="table-dark">
10.                     <tr>
11.                         <th scope="col"></th>
12.                         <th scope="col"></th>
13.                         <th scope="col">цена</th>
14.                         <th scope="col">количество</th>
15.                     </tr>
16.                 </thead>
17.                 <tbody>
18.                     {% for product in products %}
19.                         <tr class="align-middle">
20.                             <th scope="row"></th>
22.                             <td><a href="{% product.product_link %}"
23.                                 class="link-dark">{{ product.name }}</a></td>
24.                             <td>{{ product.price }}p.</td>
25.                             <td>
26.                                 <div data-mdb-input-init class="form-outline">
27.                                     <input class="text-center" type="number"
28.                                         name="typeNumber{% product.id %}" value="0"/>
29.                                 </div>
30.                             </td>
31.                         </tr>
32.                         <td class="justify-content-end" colspan="2">
33.                             <div class="row justify-content-end">
34.                                 <a href="/" class="btn btn-primary btn-dark">
35.                                     Вернуться к списку городов
36.                                 </a>
37.                             </div>
38.                         </td>
39.                         <td class="justify-content-end" colspan="2">
40.                             <div class="row justify-content-end">
41.                                 <input class="btn btn-primary btn-dark"
42.                                     type="submit" name="form-check" value="Сформировать чек">
43.                             </div>

```

```

43.         </td>
44.     </tr>
45. </tbody>
46. </table>
47. </form>
48. </div>
49.
50. {% endblock %}

```

Как видно, в шаблон передается информация о продуктах, это происходит через функцию представление. Код функции представлен на листинге 12.

Листинг 12. Функция-представление для списка датчиков

```

1. def product_list(request):
2.     if request.method == "POST":
3.         form = CategoriesForm(request.POST)
4.         city_id = int(request.GET.get('city'))
5.         if form.is_valid():
6.
7.             products = []
8.             for category in
form.cleaned_data.get('Categories'):
9. products.extend(list(Product.objects.filter(category_id=int(category))))
10.
11.         return render(request,
12.                        'product/product_list.html',
13.                        {'title': 'Список продукции',
14.                        'products': products})

```

Функция проверяет, является ли полученный запрос POST-запросом. Далее происходит валидация формы, проверяется находятся ли полученные значения в допустимых рамках. Затем происходит формирование списка датчиков из заявленных категорий. Данные передаются в шаблон страницы (листинг 11).



На данной странице (рисунок 12) посередине расположены сами товары и их счетчики количества. В нижней части страницы расположены две кнопки: первая для возврата на главную страницу, вторая для формирования чека (Сравнительного списка датчиков для выбора и предложения решения) (рисунок 13). Кнопка возврата на главную страницу задействует ту же команду от пользователя, что и при первом переходе на страницу сайта. Кнопка формирования чека передает POST-запрос, в теле которого передается список и количество датчиков, которое пользователь выбрал на странице.

### **Сравнительный список датчиков для выбора и предложения решения**

Арду.рф: Пермь, ул. Пушкина, 17

Товар	Цена	Количество	Итого
Датчик расхода воды YF-S201, 1/2", до 30 л/мин	304р.	3	912р.
Датчик уровня воды поплавковый, вертикальное крепление	205р.	1	205р.
Датчик дождя (капель, утечки) с платой контроля	133р.	5	665р.
Датчик влажности почвы	133р.	2	266р.
Датчик газа MQ-4	205р.	1	205р.
Датчик газа MQ-5	236р.	1	236р.
Датчик газа MQ-6	205р.	3	615р.
Датчик газа MQ-8	236р.	1	236р.
Датчик света пороговый	108р.	1	108р.
Датчик пламени	157р.	1	157р.
		Сумма	3605р.

Рисунок 13. Сравнительный список датчиков для выбора и предложения решения

В отличие от предыдущих команд, команда формирования чека не отправляет пользователю в ответ страницу. Ответ представляет из себя сформированный файл в формате pdf. Для работы с pdf была выбрана библиотека Reportlab, как удобный и функциональный инструмент для создания документов, также эта библиотека рекомендована в официальной документации Django [2]. Формирование страницы происходит прямо внутри функции-представления, отвечающей за вывод чека. Код функции представлен листингом 13.

Листинг 13. Функция-представление для вывода чека

```
1. def form_check(request):
2.     if request.method == "POST":
3.         basket = []
4.         for key, value in request.POST.dict().items():
5.             if key.startswith("typeNumber") and int(value) >
6.                 0:
7.                 basket.append((int(key.replace('typeNumber',
8. '')), int(value)))
9.         if not basket:
10.            return HttpResponse(status=203)
11.            buffer = io.BytesIO()
12.            p = SimpleDocTemplate(buffer)
13.            pdfmetrics.registerFont(TTFont('golos-text',
14. 'fonts/golos-text_regular.ttf'))
15.            pdfmetrics.registerFont(TTFont('golos-text-bold',
16. 'fonts/golos-text_bold.ttf'))
17.            elements = []
18.            data = []
19.            data.append(['Товар', 'Цена', 'Количество',
20. 'Итого'])
21.            sumprice = 0
22.            addresses = set()
23.            for id, count in basket:
24.                product = Product.objects.get(id=id)
25.                data.append([product.name, product.price+'p.',
26. str(count), str(int(product.price.replace(' ',
27. ''))*count)+'p.'])
28.                sumprice += int(product.price.replace(' ',
29. ''))*count
```

```

24.         addresses.add(product.shop.name + ': '
+product.shop.address)
25.         data.append(['', '', 'Сумма', str(sumprice)+'p.'])
26.         data = [[], [], [], ['\n'.join(addresses)]] + data
27.         t = Table(data)
28.         t.setStyle(TableStyle([
29.             ('FONTNAME', (0,0), (-1,-1), 'golos-text'),
30.             ('FONTSIZE', (0,0), (-1,-1), 10),
31.             ('FONTSIZE', (0,0), (-1, len(addresses)+3),
12),
32.             ('FONTSIZE', (0,-1), (-1,-1), 12),
33.         ]))
34.
35.         header_style = ParagraphStyle(
36.             name="Header",
37.             fontName='golos-text-bold',
38.             fontSize=16,
39.             alignment=1
40.         )
41.         header = Paragraph("Сравнительный список датчиков
для выбора и предложения решения\n\n", style=header_style)
42.         elements.append(header)
43.         elements.append(t)
44.         p.build(elements)
45.         buffer.seek(0)
46.         return FileResponse(buffer, as_attachment=True,
filename=f"check {datetime.now().strftime('%d.%m.%Y
%H:%M:%S')}.pdf")
47.         return redirect('/')

```

В процессе выполнения формирование pdf файла происходит в оперативной памяти сервера, он не сохраняется на диск. Для работы с кириллицей был выбран свободно распространяемый шрифт golos text. Функция рассчитывает цены выбранных датчиков, передает адреса магазинов. В чеке формируется таблица для сравнительного списка датчиков для выбора и предложения решения.

На этом заканчивается рассмотрение функциональной части клиентского приложения. Далее будет рассмотрена серверная часть, обеспечивающая работу внутреннего сервера.

Рассмотрим файл конфигурации (листинг 14)

#### Листинг 14. Файл конфигурации приложения

```
1. from dataclasses import dataclass
2. from environs import Env
3.
4. @dataclass
5. class Server:
6.     secret_key: str
7.
8. @dataclass
9. class Config:
10.     server: Server
11.
12. def load_config(path=None):
13.     env=Env()
14.     env.read_env(path)
15.
16.     return
    Config(server=Server(secret_key=env('SECRET_KEY')))
```

Файл конфигурации необходим, чтобы скрывать непосредственно из кода приложения закрытые данные, такие как уникальные ключи аутентификации, ssh-ключи, логины и пароли баз данных и другие. В нашем случае скрыт секретный ключ сервера Django (при формировании приложения он генерируется непосредственно в настроечном файле сервера и при публикации кода может попасть в открытые источники), он подгружается из окружения установленного на сервере (файл .env), скрытого для публикации. Это необходимо для обеспечения безопасности сервера. При масштабировании системы, предпочтителен переход на другую СУБД, в таком случае логин и пароль от сервера базы данных также необходимо подгружать из окружения через файл конфигурации.

Рассмотрим настроечный файл сервера (листинг 15).

```

1. """
2. Django settings for vcr project.
3.
4. Generated by 'django-admin startproject' using Django 5.0.
5.
6. For more information on this file, see
7. https://docs.djangoproject.com/en/5.0/topics/settings/
8.
9. For the full list of settings and their values, see
10. https://docs.djangoproject.com/en/5.0/ref/settings/
11. """
12.
13. #loading configuration
14. from .config import load_config
15. config = load_config()
16.
17. from pathlib import Path
18.
19. # Build paths inside the project like this: BASE_DIR /
    'subdir'.
20. BASE_DIR = Path(__file__).resolve().parent.parent
21.
22.
23. # Quick-start development settings - unsuitable for
    production
24. # See
    https://docs.djangoproject.com/en/5.0/howto/deployment/checkli
        st/
25.
26. # SECURITY WARNING: keep the secret key used in production
    secret!
27. SECRET_KEY = config.server.secret_key
28.
29. # SECURITY WARNING: don't run with debug turned on in
    production!
30. DEBUG = True
31.
32. ALLOWED_HOSTS = []
33.
34.
35. # Application definition
36.
37. INSTALLED_APPS = [
38.     'django.contrib.admin',
39.     'django.contrib.auth',
40.     'django.contrib.contenttypes',

```

```

41.     'django.contrib.sessions',
42.     'django.contrib.messages',
43.     'django.contrib.staticfiles',
44.     'django_bootstrap5',
45.     'product.apps.ProductConfig',
46. ]
47.
48. MIDDLEWARE = [
49.     'django.middleware.security.SecurityMiddleware',
50.     'django.contrib.sessions.middleware.SessionMiddleware',
51.     'django.middleware.common.CommonMiddleware',
52.     'django.middleware.csrf.CsrfViewMiddleware',
53.
54.     'django.contrib.auth.middleware.AuthenticationMiddleware',
55.     'django.contrib.messages.middleware.MessageMiddleware',
56.
57.     'django.middleware.clickjacking.XFrameOptionsMiddleware',
58. ]
59.
60. ROOT_URLCONF = 'vcr.urls'
61.
62. TEMPLATES = [
63.     {
64.         'BACKEND':
65.         'django.template.backends.django.DjangoTemplates',
66.         'DIRS': [BASE_DIR / 'templates'],
67.         'APP_DIRS': True,
68.         'OPTIONS': {
69.             'context_processors': [
70.                 'django.template.context_processors.debug',
71.                 'django.template.context_processors.request',
72.                 'django.contrib.auth.context_processors.auth',
73.                 'django.contrib.messages.context_processors.messages',
74.             ],
75.         },
76.     ],
77. ]
78.
79. WSGI_APPLICATION = 'vcr.wsgi.application'
80.
81. # Database
82. #
83. https://docs.djangoproject.com/en/5.0/ref/settings/#databases
84.
85. DATABASES = {
86.     'default': {

```

```

84.         'ENGINE': 'django.db.backends.sqlite3',
85.         'NAME': BASE_DIR / 'db.sqlite3',
86.     }
87. }
88.
89.
90. # Password validation
91. #
92. https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
93. AUTH_PASSWORD_VALIDATORS = [
94.     {
95.         'NAME':
96.         'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
97.     },
98.     {
99.         'NAME':
100.        'django.contrib.auth.password_validation.MinimumLengthValidator',
101.     },
102.     {
103.         'NAME':
104.        'django.contrib.auth.password_validation.CommonPasswordValidator',
105.     },
106.     {
107.         'NAME':
108.        'django.contrib.auth.password_validation.NumericPasswordValidator',
109.     },
110. ]
111.
112. # Internationalization
113. # https://docs.djangoproject.com/en/5.0/topics/i18n/
114. LANGUAGE_CODE = 'en-us'
115.
116. TIME_ZONE = 'UTC'
117.
118. USE_I18N = True
119.
120. USE_TZ = True
121.
122. # Static files (CSS, JavaScript, Images)
123. # https://docs.djangoproject.com/en/5.0/howto/static-files/

```

```
124. STATIC_URL = 'static/'
125.
126. # Default primary key field type
127. #
    https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
128.
129. DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
130.
131.
132. #media
133. # Media storage settings
134. MEDIA_ROOT = BASE_DIR/'media'
135. MEDIA_URL = '/media/'
```

В данном файле мы должны обратить внимание на список установленных приложений (список `INSTALLED_APPS`), в нем содержатся приложения, задействованные в проекте, в том числе: системные (`django.contrib.admin`, `django.contrib.auth`, `django.contrib.contenttypes`, `django.contrib.sessions`), приложение для браузерных уведомлений (`django.contrib.messages`), приложение для подключения статических файлов к шаблонам (`django.contrib.staticfiles`), приложение для подключения к шаблонам фронтенд-фреймворка Bootstrap (`django_bootstrap5`), и основное приложение проекта Product (`product.apps.ProductConfig`).

Дальше идет список установленных мидлваров (связующих ПО), в нашем приложении используются только стандартные мидлвары Django.

Далее следует обратить внимание на настройки базы данных. В словаре `DATABASES` содержится информация о работающих БД: название файла и движок СУБД. Текущая база данных приложения работает на движке `sqlite`, при масштабировании предпочтителен переход на PostgreSQL, MariaDB или аналогичные реляционные СУБД.



Далее в файле идет список валидаторов паролей, в приложении используются стандартные валидаторы Django, так как не предполагается аутентификация пользователя при работе с приложением.

В конце настроечного файла находится информация о директориях для хранения статических файлов (STATIC\_URL) и загружаемых медиа-файлов (MEDIA\_ROOT, MEDIA\_URL), в нашем случае медиа-файлами служат картинки датчиков, загруженные с сайтов магазинов.

Наконец, рассмотрим системную базу данных приложения. Таблицы серверной части базы данных можно увидеть на рисунке 14.

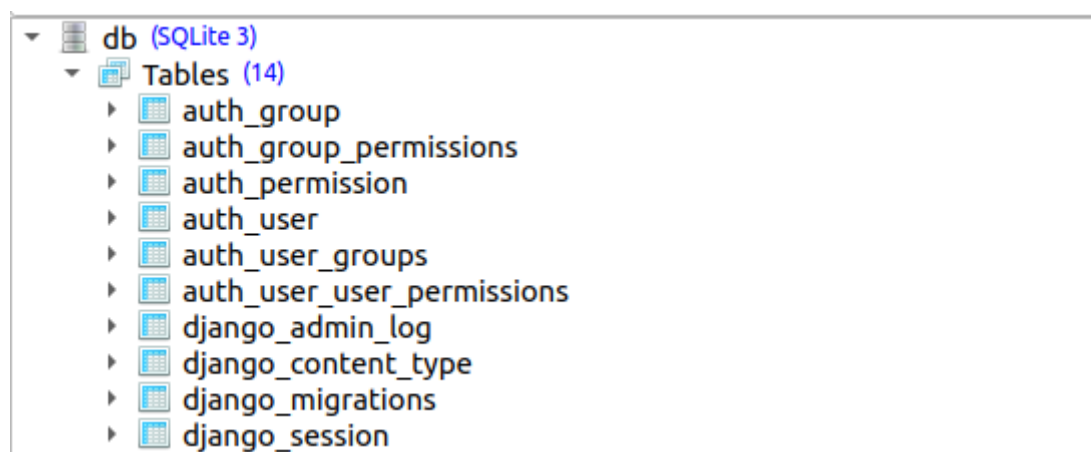


Рисунок 14. Таблицы серверной части БД

Как видно из рисунка, таблицы глобально делятся на две группы: таблицы аутентификации и системные таблицы Django.

Рассмотрим таблицы аутентификации. Таблицы `auth_group` `auth_group_permissions` отвечают соответственно за группы пользователей и права этих групп. В нашем приложении эти таблицы не содержат данные, так как пользователи не нуждаются в аутентификации. Таблица `auth_permissions` содержит информацию о различных правах по изменению моделей. Права автоматически формируется Django для всех системных и пользовательских

моделей. Данная таблица позволяет гибкую настройку прав для групп и отдельных авторизованных пользователей. Таблица `auth_user` содержит информацию о зарегистрированных в системе пользователях, в нашем приложении к таким относятся только администраторы системы, которым предоставлены полные права по изменению моделей. Таблица `auth_user_group` распределяет пользователей по группам. Таблица `auth_user_user_permissions` хранит информацию о правах пользователей.

Рассмотрим системные таблицы Django. Таблица `django_admin_log` содержит системный лог всех манипуляций с моделью данных через административную панель. В ней сохраняется информация о изменении, времени его осуществления и пользователе, его производившем. Лог позволяет отслеживать изменения и служит аналогом системы контроля версий для базы данных. Лог не может заменить резервное копирование, но может быть использован для ручного отката незначительных изменений в базе данных. Таблица `django_content_type` соотносит таблицы базы с приложениями Django. Содержимое таблицы `content_type` можно видеть на рисунке 15, на нем наглядно видно разделение пользовательских и системных таблиц.

	id	app_label	model
1	1	admin	logentry
2	2	auth	permission
3	3	auth	group
4	4	auth	user
5	5	contenttypes	contenttype
6	6	sessions	session
7	7	product	city
8	8	product	shop
9	9	product	product
10	10	product	category

Рисунок 15. Содержимое таблицы `django_content_type`

Таблица `django_migrations` содержит информацию о миграциях базы данных. Миграции Django - специальный инструмент для распространения изменений, внесенных в структуру базы данных в ходе разработки и эксплуатации приложения. Этот инструмент позволяет перерабатывать структуру базы данных в процессе её использования и обеспечивает полную совместимость разных версий БД. При работе с бэкендом SQLite, движок Django имитирует изменение схемы БД, из-за ограниченных возможностей данного бэкенда по изменению схемы. Этот процесс происходит по такой схеме:

- Создание новой таблицы с новой схемой
- Копирование данных
- Удаление старой таблицы
- Переименование новой таблицы в соответствии с исходным именем

Такой процесс обычно происходит без ошибок, но при росте объемов данных может быть достаточно медленным. Именно поэтому при масштабировании проекта предлагается перейти на другой движок базы данных.

Таблица `django_sessions` содержит информацию обо всех сеансах работы сервера.

На этом заканчивается рассмотрение разработки клиентского приложения, в данном пункте были обозначены все основные моменты разработки и настройки приложения, также обозначены некоторые моменты дальнейшей эксплуатации системы и её масштабирования.

В пункте 2.3 была рассмотрена реализация всех основных компонентов разрабатываемой информационной системы. Общая файловая структура проекта представлена на рисунке 16.

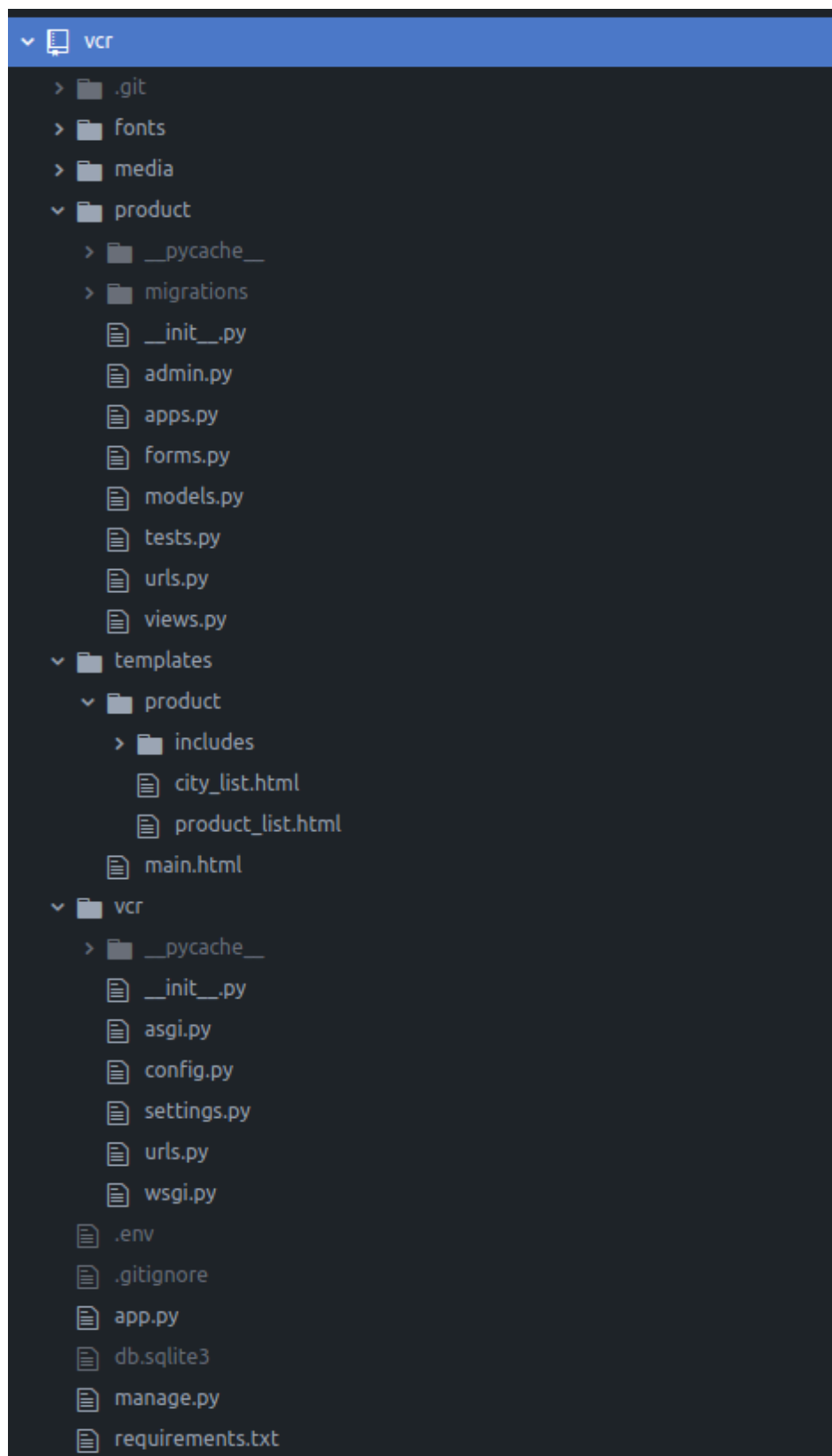


Рисунок 16. файловая структура проекта

## **2.4 Подготовка, ввод и систематизация базы данных датчиков по выбранным магазинам.**

Для создания приложения по поиску датчиков в базах данных магазинов было проведено исследование, в ходе которого составлены основные категории датчиков, к которым относятся датчики, например: датчики температуры, датчики влажности, датчики давления, датчики движения и т.д.. Также изучены научные статьи и техническая документация, в которых описываются современные датчики и их применения.

Для каждого датчика была добавлена информация о наличии и местонахождении в каждом магазине, чтобы пользователи могли легко найти и приобрести необходимые компоненты.

## **Вывод по главе 2**

В данной главе проведено проектирование и разработка ПО. Были спроектированы таблицы базы данных, установлены свойства атрибутов таблиц (таблицы 2-4) и прописаны все основные бизнес-процессы, задействованные в системе (рис. 6-7). Спроектированы диаграммы прецедентов (рис. 8). Реализован пользовательский интерфейс (рис. 10-12) в соответствии с действующим техническим заданием.

### **ГЛАВА 3. ФИНАНСОВОЕ И ДОКУМЕНТАЦИОННОЕ ОБЕСПЕЧЕНИЕ РАЗРАБОТКИ ПО «СХЕМОТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕАЛИЗАЦИИ РОБОТОТЕХНИЧЕСКИХ РЕШЕНИЙ». СОСТАВЛЕНИЕ БЮДЖЕТА ПРОЕКТА. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ ПО. НАПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОЙ ДОКУМЕНТАЦИИ**

#### **3.1 Документационное обеспечение**

Цель документационного обеспечения обеспечить понимание принципов работы ПО для пользователей и администраторов. При разработке ПО были разработаны: руководство администратора, руководство пользователя и программа методики испытаний.

Программа и методика испытаний (содержится в приложении 3) разработанной ПО предназначена для проверки выполнения заданных функций ПО и соответствия обозначенных задач и функций системы.

Программа и методика испытаний включает в себя:

1. Общие положения
2. Объект испытаний
3. Цель испытаний
4. Общие положения
5. Объем испытаний
6. Условия и порядок проведения испытаний
7. Требования к ПО
8. Материально-техническое обеспечение испытаний
9. Метрологическое обеспечение испытаний

В следующей таблице представлены основные процессы проверки и выполняемые действия, которые необходимо произвести для тестирования ПО на функциональность.



<b>№</b>	<b>Наименование проверки</b>	<b>Исполнитель</b>	<b>Выполняемые действия</b>	<b>Ожидаемый результат</b>
1	Вход в приложение	Пользователь; Сис. админ.	Выполняется вход в приложение через ярлык на рабочем столе	Произошёл вход в приложение, активность не прерывается
2	Создание проекта	Пользователь; Сис. админ.	Пользователь нажимает кнопку «+ Новый проект»	Происходит успешное создание нового проекта

Таблица 6. Методика испытаний

Для ускорения процесса обучения технических специалистов в аудитории, снижения нагрузки на администратора и повышения эффективности использования продукта были разработаны документы: руководство пользователя (содержится в приложении 1) и руководство администратора (содержится в приложении 2).

Руководство пользователя содержит следующие разделы:

1. Введение
2. Назначение и условия применения
3. Подготовка к работе
4. Описание основных операций
5. Аварийные ситуации
6. Рекомендации по освоению

ПО разрабатывалась с учетом различного уровня подготовки владения компьютером, однако, пользователи должны обладать базовыми навыками работы с компьютерной техникой, а в частности работы с операционными системами и прикладным ПО. Далее приведен фрагмент из руководства пользователя, в котором обозначен необходимый уровень подготовки пользователя при работе с ПО (в приложении 1 разъясняются более подробные аспекты руководства пользователя, представленного в тексте).

### **Уровень подготовки пользователя**

Для освоения приложения, пользователю достаточно базовых навыков работы с ПК, умение пользоваться браузером. Подробнее с уровнем подготовки пользователя можно ознакомиться в приложенном руководстве пользователя (Приложение 1).

### **Требования к компетенциям администратора**

Для администрирования ПО пользователь должен обладать следующими компетенциями:

- Умение работать с персональным компьютером
- Умение работать и настраивать серверную составляющую вычислительной машины
- Понимание работы Desktop- и Web-приложений

Документационное обеспечение имеет ключевое значение для успешного внедрения, использования и поддержания ПО, обеспечивая пользователям и специалистам необходимую информацию для эффективной работы. Подробнее с необходимыми компетенциями администратора можно ознакомиться в приложенном руководстве администратора (Приложение 2).

## **ЗАКЛЮЧЕНИЕ**

В рамках выполнения данной выпускной квалификационной работы было спроектировано и разработано приложение «Схемотехническое и программное обеспечение реализации робототехнических решений». Система предоставляет возможность для быстрого и эффективного поиска электронных датчиков в базах данных магазинов радиоэлектроники, дает возможность получить актуальную информацию о наличии и местонахождении товаров.

Реализована система администрирования проекта и пользовательская часть приложения. Пользовательский интерфейс представляет собой удобную и интуитивную веб-страницу, к которой каждый пользователь имеет доступ без необходимости авторизации.

Перед началом работы над проектом было составлено и утверждено Техническое задание. Затем разработана Программа и методика испытаний для тестирования системы. Также подготовлено документационное обеспечение в составе Руководства пользователя и Руководства администратора.

Таким образом, система удовлетворяет поставленным требованиям, благодаря грамотному проектированию и тестированию были решены все задачи и достигнута поставленная цель. Созданная система обеспечивает надежный и эффективный инструмент для поиска и подбора датчиков в различных магазинах.

Кроме того, разработанная система имеет перспективы для дальнейшего развития и улучшения. Планируется расширять список магазинов-партнеров, предоставляющих данные о наличии электронных датчиков, система предусматривает возможность добавления данных из иногородних магазинов. Планируется добавление сравнительной статистики

цен и сроков доставки компонентов. Планируется возможность приобретения датчиков непосредственно через систему.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bootstrap documentation - Текст : электронный // Bootstrap Team : [сайт]. - URL: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> (дата обращения: 20.04.2024)
2. Django documentation - Текст : электронный // Django Software Foundation : [сайт]. - URL: <https://docs.djangoproject.com/en/5.0/> (дата обращения: 12.04.2024)
3. Pony ORM documentation - Текст : электронный // Pony ORM : [сайт]. - URL: <https://docs.ponyorm.org/toc.html> (дата обращения: 20.10.2023)
4. ReportLab documentation - Текст : электронный // ReportLab, Inc : [сайт]. - URL: <https://docs.reportlab.com/> (дата обращения: 25.04.2024)
5. Selenium documentation - Текст : электронный // Software Freedom Conservancy : [сайт]. - URL: <https://www.selenium.dev/documentation/> (дата обращения: 15.10.2023)
6. ГОСТ 34.201-2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем. – Текст : электронный // Техэксперт : [сайт]. – URL: <https://docs.cntd.ru/document/1200181803> (дата обращения: 12.01.2024).
7. ГОСТ Р 59792-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды испытаний автоматизированных систем. - Текст : электронный // Техэксперт : [сайт]. - URL: <https://docs.cntd.ru/document/1200181348> (дата обращения: 12.01.2024)
8. ГОСТ Р 59795-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов. – Текст : электронный

// Техэксперт : [сайт]. – URL: <https://docs.cntd.ru/document/1200181351>  
(дата обращения: 12.01.2024).

## ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

УТВЕРЖДЕН

XXXXXXXXX.XXXXXXX.XXX.ИЗ-ЛУ

УТВЕРЖДАЮ

руководитель проекта

\_\_\_\_\_ / А. М. Черепанов.

«\_\_» \_\_\_\_\_ 2024 г.

СОГЛАСОВАНО

Директор  
ЧОУ ДПЛ “Кванториум фатоника”

\_\_\_\_\_ / В. Г. Былинкина.

«\_\_» \_\_\_\_\_ 2024 г.

**Разработка ПО «Схемотехническое и программное обеспечение реализации  
робототехнических решений»**

Руководство пользователя

XXXXXXXXX.XXXXXXX.XXX.ПМ

## Оглавление

<b>1. ВВЕДЕНИЕ</b>	<b>3</b>
Описание	3
Уровень подготовки пользователя	3
Перечень эксплуатационной документации	3
<b>2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ</b>	<b>4</b>
Назначение	4
Условия применения	4
<b>3. ПОДГОТОВКА К РАБОТЕ</b>	<b>5</b>
Первоначальная настройка	5
<b>4. ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ</b>	<b>6</b>
<b>5. АВАРИЙНЫЕ СИТУАЦИИ</b>	<b>7</b>
Недоступность системы	7
Недоступность парсинга данных	7
Аппаратные сбои	7
<b>6. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ</b>	<b>8</b>



# **1. ВВЕДЕНИЕ**

## **Описание**

ПО «Схемотехническое и программное обеспечение реализации робототехнических решений» — это ПО, предназначенное для поиска доступных датчиков для робототехники которые есть в наличии в г.Перми.

Основное предназначение ПО - это подключение к различным источникам данных, сбор, хранение и дальнейшая обработка данных.

ПО обеспечивает выполнения функций парсинга данных, выбора категорий датчиков, вывод подходящих датчиков в наличии, вывод примеров использования датчиков и вывод местонахождения магазина.

## **Уровень подготовки пользователя**

Пользователи ПО должны обладать базовыми навыками работы с ПК.

## **Перечень эксплуатационной документации**

Для работы в ПО пользователь должен ознакомиться с настоящим руководством.

## **2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ**

### **Назначение**

ПО нацелена на сокращение времени поиска подходящих датчиков в короткие сроки. ПО позволяет выполнить поиск необходимых датчиков среди магазинов г. Перми, узнать наличие датчиков.

### **Условия применения**

ПО может эксплуатироваться и выполнять заданные функции при соблюдении требований, предъявляемых к техническому, системному и прикладному программному обеспечению.

### **3. ПОДГОТОВКА К РАБОТЕ**

#### **Первоначальная настройка**

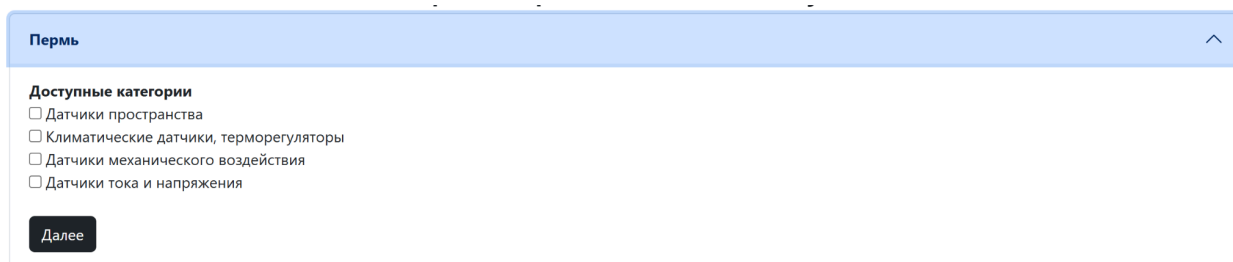
Первоначальная настройка ПО пользователем не требуется. Взаимодействие пользователя с Системой осуществляется посредством графического интерфейса.

До начала работы с ПО необходимо:

1. Ознакомиться с настоящим руководством.
2. Перейти по заданному url-адресу для доступа к web-сервису. Авторизация пользователя в системе не требуется

## 4. ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ



Все критерии поиска необходимых датчиков в ПО осуществляются на Главном окне (Рисунок 1).



The screenshot shows a window titled "Пермь" with a blue header. Below the header, there is a section titled "Доступные категории" (Available categories) with four checkboxes: "Датчики пространства" (Space sensors), "Климатические датчики, терморегуляторы" (Climate sensors, thermostats), "Датчики механического воздействия" (Mechanical impact sensors), and "Датчики тока и напряжения" (Current and voltage sensors). A "Далее" (Next) button is located at the bottom left of the list.

Рисунок 1. Главное окно ПО

Датчики которые подходят под критерии, отображаются в виде списка и короткого описания к ним (местонахождение магазина, ссылка на пример применения) (Рисунок 2).

	цена	количество
	<a href="#">Датчик тока 20А</a> 242р.	<input type="text" value="0"/>
	<a href="#">Датчик тока 5А</a> 205р.	<input type="text" value="0"/>

ВЕРНУТЬСЯ К СПИСКУ ГОРОДОВ СФОРМИРОВАТЬ ЧЕК

Рисунок 2. Список подходящих датчиков

Для завершения работы системы необходимо закрыть основное окно программы.

## **5. АВАРИЙНЫЕ СИТУАЦИИ**

### **Недоступность системы**

При недоступности системы необходимо установить или переустановить систему (смотри руководство Администратора).

### **Сбой подключения к системе**

При сбое текущего подключения, следует проверить соединение с интернетом и устранить возникшие неполадки

### **Аппаратные сбои**

При возникновении аппаратных сбоев, т.к. отказ жесткого диска, центрального процессора, оперативной памяти и т.д., необходимо обратиться к системному администратору для устранения возникших сбоев.

## **6. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ**

Для успешной работы с ПО необходимо: получить навыки работы с программным обеспечением, указанным в настоящем Руководстве; ознакомиться с настоящим Руководством.

## ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]

## СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

## СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата



## ПРИЛОЖЕНИЕ 2 РУКОВОДСТВО АДМИНИСТРАТОРА

УТВЕРЖДЕН

XXXXXXXXX.XXXXXXX.XXX.ИЗ-ЛУ

УТВЕРЖДАЮ

руководитель проекта

\_\_\_\_\_ / А. М. Черепанов.

«\_\_» \_\_\_\_\_ 2024 г.

СОГЛАСОВАНО

Директор  
ЧОУ ДПЛ "Кванториум фатоника"

\_\_\_\_\_ / В. Г. Былинкина.

«\_\_» \_\_\_\_\_ 2024 г.

**Разработка ПО «Схемотехническое и программное обеспечение реализации  
робототехнических решений»**

Руководство пользователя

XXXXXXXXX.XXXXXXX.XXX.ПМ

## СОДЕРЖАНИЕ

<b>1. НАЗНАЧЕНИЕ СИСТЕМЫ</b>	<b>4</b>
Назначение ПО	4
Функции системы	4
<b>2. УСЛОВИЯ РАБОТЫ СИСТЕМЫ</b>	<b>5</b>
Требования к аппаратуре и программным средствам	5
Требования к компетенциям администратора	5
<b>3. УСТАНОВКА И НАСТРОЙКА СИСТЕМЫ</b>	<b>6</b>
<b>4. СОЗДАНИЕ РЕЗЕРВНОЙ КОПИИ И ВОССТАНОВЛЕНИЕ СИСТЕМЫ</b>	<b>7</b>
Создание резервной копии	7
Восстановление работоспособности	7
<b>5. ПОРЯДОК ЗАПУСКА И ОСТАНОВКИ</b>	<b>8</b>
Порядок запуска	8
Порядок остановки	8
<b>6. АВАРИЙНЫЕ СИТУАЦИИ</b>	<b>9</b>
Ошибка установки соединения	9
Ошибка установки сторонних пакетов пакетным менеджером	9
<b>7. СООБЩЕНИЯ АДМИНИСТРАТОРУ</b>	<b>10</b>

## **АННОТАЦИЯ**

Настоящий документ представляет собой руководство администратора ПО «Схемотехническое и программное обеспечение реализации робототехнических решений».

Руководство определяет правила администрирования системы, а именно установку, настройку и сопровождение ПО, устранение возможных ошибок.

Перед работой администратора в системе рекомендуется внимательно ознакомиться с настоящим Руководством.

# **1. НАЗНАЧЕНИЕ СИСТЕМЫ**

## **Назначение ПО**

ПО нацелена на сокращение времени поиска подходящих датчиков в короткие сроки. Система представляет из себя Web приложение. ПО используется на аппаратно-технических средствах заказчика.

## **Функции системы**

Описание возможностей программного комплекса:

- Интеграция с сайтами магазинов по продаже электронных датчиков г.Перми.
- Сбор и хранение данных о датчиках: название, применимость, стоимость, адрес магазина, ссылка на характеристики.
- Формирование отчетности: вывод результатов поиска, фильтр результатов поиска, экспорт результатов

## **2. УСЛОВИЯ РАБОТЫ СИСТЕМЫ**

### **Требования к аппаратуре и программным средствам**

Для корректной работы ПО требуется наличие оборудования в виде компьютера включающего:

- процессор с архитектурой x86-64 (AMD, Intel);
- оперативная память — не менее 8 ГБ;
- объем свободного дискового пространства — не менее 100 ГБ;
- стандартный монитор;
- совместимость с оборудованием (штатное, предусмотренное документацией);
- Широкополосное подключение к интернету.

А также программных средств:

- Браузер Google Chrome, версии не ниже 80 (или другой с аналогичными характеристиками)

### **Требования к компетенциям администратора**

Для администрирования системы пользователь должен обладать следующими компетенциями:

- Умение работать с персональным компьютером
- Умение работать и настраивать серверную составляющую вычислительной машины
- Понимание работы Desktop- и Web-приложений
- Средний уровень работы с python версии 3.10.4 и выше
- Умение работать с административной панелью фреймворка Django
- Понимание концепций реляционных баз данных, знание языка запросов SQL

### **3. УСТАНОВКА И НАСТРОЙКА СИСТЕМЫ**

Во избежание непредвиденных ситуаций и предотвращения разного рода ошибок при установке и эксплуатации системы защиты, следует предварительно подготовить рабочее место. Для этого необходимо провести ряд процедур:

- проверить оперативную память компьютера, а также его жесткий диск на отсутствие вирусов;
- убедиться, что на компьютере в данный момент не работают какие-либо программы, препятствующие работе с системным реестром, выполняющие функции защиты от шпионского программного обеспечения и так далее;

Установка дополнительных компонентов на рабочую станцию администратора не требуется. Всё взаимодействие с системой производится через web-интерфейс или через удаленное соединение с сервером.

Перед началом работы с системой необходимо убедиться, что администратор зарегистрирован в системе и имеет к ней доступ, в соответствии со своими полномочиями.

## **4. СОЗДАНИЕ РЕЗЕРВНОЙ КОПИИ И ВОССТАНОВЛЕНИЕ СИСТЕМЫ**

### **Создание резервной копии**

Создание резервной копии базы данных производится администратором, через прямое подключение к серверу с системой. Для предотвращения возможных ошибок, работа системы должна быть приостановлена на время создания резервной копии. Для создания резервной копии администратор должен ввести команду `python manage.py dumpdata --output=<название резервной копии>.json` в панели управления проектом. Для предотвращения потери резервных данных при отказе основного сервера, файл с резервной копией базы данных должен быть скопирован на резервный сервер или на локальную машину администратора.

### **Восстановление работоспособности**

Для восстановления работоспособности, необходимо удалить текущую версию системы и установить ее заново используя поставляемый установочный пакет. Восстановление данных из резервной копии базы данных производится администратором в панели управления проектом, через прямое подключение к основному серверу системы. Для восстановления данных, администратор должен ввести команду `python manage.py loaddata --output=<название резервной копии>.json`, предварительно переместив последнюю стабильную резервную копию в корневой каталог проекта системы.

## **5. ПОРЯДОК ЗАПУСКА И ОСТАНОВКИ**

### **Порядок запуска**

Запуск системы производится администратором через прямое подключение к серверу. Для запуска системы администратору достаточно ввести в панели управления проектом команды `python manage.py runserver`

### **Порядок остановки**

Для завершения работы программы администратору необходимо остановить текущий процесс системы, напрямую подключившись к серверу. В операционной системе Linux остановка процесса происходит через сочетание клавиш CTRL + C.



## **6. АВАРИЙНЫЕ СИТУАЦИИ**

### **Ошибка установки соединения**

Возможные причины: отсутствует соединение с сетью интернет, либо ограничен доступ к ресурсам глобальной сети; аппаратная или программная ошибка сервера (имеется в виду ошибка на стороне хостинга, не связанная с работой сервера)

### **Ошибка установки сторонних пакетов пакетным менеджером**

Возможные причины: не указаны версии пакетов, неверный синтаксис конфигурационного файла, устаревшая версия пакетного менеджера

### **Ошибка получения данных**

Возможные причины: изменения структуры страниц магазинов, с которых загружаются данные, в том числе изменение CSS-селекторов важных структурных элементов страницы.

## 7. СООБЩЕНИЯ АДМИНИСТРАТОРУ

При сообщении **«Фатальная ошибка»**: Администратору следует перезапустить программу

При сообщении **«Нет доступа к сети»**: Администратору следует проверить наличие подключения к сети и параметры сетевого подключения.

При сообщении **«Ошибка подключения к БД»**: Администратору следует проверить наличие подключения к БД.

## ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]

## СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

## СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

### ПРИЛОЖЕНИЕ 3. МЕТОДИКА ИСПЫТАНИЙ

УТВЕРЖДЕН

XXXXXXXXX.XXXXXXX.XXX.ИЗ-ЛУ

УТВЕРЖДАЮ

руководитель проекта

\_\_\_\_\_ / А. М. Черепанов.

«\_\_» \_\_\_\_\_ 2024 г.

СОГЛАСОВАНО

Директор  
ЧОУ ДПЛ “Кванториум фатоника”

\_\_\_\_\_ / В. Г. Былинкина.

«\_\_» \_\_\_\_\_ 2024 г.

**Разработка ПО «Схемотехническое и программное обеспечение реализации  
робототехнических решений»**

Методика испытаний

XXXXXXXXX.XXXXXXX.XXX.ПМ

## СОДЕРЖАНИЕ

<b>1. ОБЩИЕ ПОЛОЖЕНИЯ</b>	<b>4</b>
1.1. Основные понятия и определения	4
<b>2. ОБЪЕКТ ИСПЫТАНИЙ</b>	<b>5</b>
2.1. Наименование и условные обозначения	5
2.2. Цели, назначение системы	5
2.3. Комплектность испытательной системы	5
<b>3. Цель испытаний</b>	<b>6</b>
<b>4. Общие положения</b>	<b>7</b>
4.1. Перечень руководящих документов, на основании которых проводятся испытания	7
4.2. Место проведения и продолжительность испытаний	7
4.3. Ведомства и организации, участвующие в испытаниях	7
4.4. Перечень предъявляемых на испытания документов	7
<b>5. ОБЪЕМ ИСПЫТАНИЙ</b>	<b>8</b>
5.1. Перечень испытаний	8
5.1.1. Перечень проводимых проверок по документации	8
5.1.2. Количественные и качественные характеристики, подлежащие оценке	8
5.2. Последовательность проведения испытаний	8
5.3. Перечень работ, проводимых после завершения испытаний	8
<b>6. УСЛОВИЯ И ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЙ</b>	<b>10</b>
6.1. Условия проведения испытаний	10
6.2. Условия начала и завершения отдельных этапов испытаний	10
6.3. Ограничения в условиях проведения испытаний	10
6.4. Требования к техническому обслуживанию	10
6.5. Меры, обеспечивающие безопасность и безаварийность проведения испытаний	10
6.6. Порядок взаимодействия организаций, участвующих в испытаниях	10
6.7. Требования к персоналу, проводящему испытания	11
<b>7. ТРЕБОВАНИЯ ПО</b>	<b>12</b>
<b>8. Материально-техническое обеспечение испытаний</b>	<b>13</b>
8.1. Технические средства, используемые во время испытаний	13
8.2. Программные средства, используемые во время испытаний	13
<b>9. Метрологическое обеспечение испытаний</b>	<b>14</b>

## **ВВЕДЕНИЕ**

В настоящем документе приведена «Разработка ПО «Схемотехническое и программное обеспечение реализации робототехнических решений»» на этапе опытного функционирования в рабочем режиме эксплуатации. Программа предназначена для установления нормативных данных, подлежащих проверке при испытании комплекса средств автоматизации, эксплуатационного и обслуживающего персонала, а также порядок испытаний и методы их контроля. Проводится проверка наличия необходимой эксплуатационной документации, как непосредственно на рабочих местах, так у административного персонала.

Документ содержит следующие основные разделы:

- Объект испытаний
- Цель испытаний
- Общие положения
- Объём испытаний
- Условия и порядок проведения испытаний
- Материально-техническое обеспечение испытаний
- Отчетность.

# 1. ОБЩИЕ ПОЛОЖЕНИЯ

## 1.1. Основные понятия и определения

Основные понятия и определения приведены в таблице 1.

Таблица 1. Определение основных понятий

Термин	Описание
ИТ	Информационные технологии
АРМ	Автоматизированное рабочее место
ПК	Персональный компьютер
ПО	Программное обеспечение
ОС	Операционная система

Штатная ситуация – все компоненты ПО обработают действия пользователей и администраторов в соответствии с алгоритмами, приведенными в документах технических проектов.

Нештатная ситуация – один или более компонент ПО обработают действия пользователей и администраторов по алгоритмам, отличающимся от приведенных в документах технических проектов.

Отказоустойчивость – свойство ПО выполнять свои основные функции при выходе из строя одного или более компонентов.

Резервирование – свойство ПО передавать выполнение функций компонента, вышедшего из строя, другой идентичной или аналогичной компоненте.

Репрезентативность — соответствие характеристик выборки характеристикам популяции или генеральной совокупности в целом. Репрезентативность определяет возможность обобщать результаты исследования с привлечением определённой выборки на всю генеральную совокупность, из которой она была собрана.



## **2. ОБЪЕКТ ИСПЫТАНИЙ**

### **2.1. Наименование и условные обозначения**

Полное наименование – «Разработка ПО «Схемотехническое и программное обеспечение реализации робототехнических решений»».

### **2.2. Цели, назначение системы**

Основными целями создания ПО:

- Интеграция с сайтами магазинов по продаже электронных датчиков г.Перми.
- Сбор и хранение данных о датчиках: название, применимость, стоимость, адрес магазина, ссылка на характеристики.
- Формирование отчетности: вывод результатов поиска, фильтр результатов поиска, экспорт результатов

### **2.3. Комплектность испытательной системы**

ПО «Схемотехническое и программное обеспечение реализации робототехнических решений» содержит в себе такие системы как БД и Внутренний сервер.

### **3. Цель испытаний**

Целью проводимых по настоящей программе и методике испытаний ПО является определение функциональной работоспособности системы на этапе проведения испытаний. Программа испытаний должна удостоверить работоспособность ПО в соответствии с функциональным назначением. Ключевые моменты, при проведении испытаний на которые необходимо обратить внимание:

- Авторизация
- Доступность сайтов магазинов
- Доступность внутренней БД
- Работоспособность системы подотчетности

## **4. Общие положения**

### **4.1. Перечень руководящих документов, на основании которых проводятся испытания**

Для восстановления работоспособности необходимо удалить текущую версию системы и установить ее заново используя поставляемый установочный пакет.

Настоящая программа и методика испытаний разработана в соответствии со следующими документами:

- ГОСТ Р 59795—2021 Комплекс стандартов на автоматизированные системы;
- техническое задание на разработку системы;
- РД 50-34.698-90 Автоматизированные системы, требования к содержанию документов;
- ГОСТ 34.603-92 Виды испытаний автоматизированных систем;
- Перечень эксплуатационных параметров автоматизированных информационных систем;
- ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.

### **4.2. Место проведения и продолжительность испытаний**

Приёмочные испытания должны проводиться на объекте Заказчика в течение одного календарного месяца. Приёмочные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной с Заказчиком Программы, и методики испытаний.

### **4.3. Ведомства и организации, участвующие в испытаниях**

Нагрузочные испытания ПО проводятся комиссией, в состав которой входят представители организаций Заказчика и Исполнителя. Состав комиссии утверждается Приказом.

### **4.4. Перечень предъявляемых на испытания документов**

- Программа и методика испытаний
- Руководство администратора
- Руководство пользователя

– Акт приёмочных испытаний

## **5. ОБЪЕМ ИСПЫТАНИЙ**

### **5.1. Перечень испытаний**

Испытания состоят из:

- Документации;
- Программных и технических средств;
- Тестирования системы;
- Сравнения устанавливаемого ПО.

Более детальная методика испытаний отображена в Приложении 1 — Методика испытаний. Там отображены основные функции, которые будут учитываться при тестировании системы на функциональность.

#### **5.1.1.Перечень проводимых проверок по документации**

Состав документации, представляемой на испытания, ее комплектность, качество разработки, соответствие нормативно-техническим требованиям.

Состав программных технических средств, входящих в состав опытного образца ПО, их комплектность и работоспособность.

#### **5.1.2.Количественные и качественные характеристики, подлежащие оценке**

Количественные характеристики — физические величины, характеризующие какое-нибудь свойство технического устройства, системы, явления или процесса.

Качественные — не требующие проведения измерений.

5.1.2.1. Корректность установки ПО.

5.1.2.2. Успешный запуск.

5.1.2.3. Соответствие интерфейса и работоспособность основных функций ПО.

### **5.2. Последовательность проведения испытаний**

Испытания проводятся в последовательности, указанной в пункте 5.1

### **5.3. Перечень работ, проводимых после завершения испытаний**

После завершения испытаний системы ПО все результаты испытаний фиксируются в отчетах о нагрузочных испытаниях. Общая цель подобного отчета - документирование результатов и характеристик работы системы под различными нагрузками.

В случае успешного проведения проверки функциональности и нагрузочных испытаний в полном объеме Исполнитель совместно с Заказчиком на основании Протокола испытаний утверждают Акт приемки-сдачи работ. Исполнитель передает заказчику систему на баланс.

В случае выявления несоответствия разработанной программы отдельным требованиям технического задания Исполнитель проводит корректировку Программы испытаний и документации по результатам испытаний в сроки, согласованные с Заказчиком.

По завершении корректировки Программы испытаний и документации Исполнитель и Заказчик проводят повторные испытания согласно настоящей Программе испытаний и Методике в объеме, требуемом для проверки проведения корректировок.

Мелкие, несущественные замечания могут быть устранены в рабочем порядке.

## **6. УСЛОВИЯ И ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЙ**

### **6.1. Условия проведения испытаний**

Испытания должны проводиться в нормальных климатических условиях по ГОСТ 22261-94. Условия проведения испытаний приведены ниже:

- Температура окружающего воздуха, °C -  $20 \pm 5$ ;
- Относительная влажность, % - от 30 до 80;
- Атмосферное давление, кПа - от 84 до 106. Требования к электропитанию:
- Частота питающей электросети, Гц -  $50 \pm 0,5$ ;
- Напряжение питающей сети переменного тока, В -  $220 \pm 4,4$ .

### **6.2. Условия начала и завершения отдельных этапов испытаний**

Необходимым и достаточным условием завершения испытаний является успешное завершение проверок (см. пункт 5.1.1).

### **6.3. Ограничения в условиях проведения испытаний**

Отсутствуют.

### **6.4. Требования к техническому обслуживанию**

Отсутствуют.

### **6.5. Меры, обеспечивающие безопасность и безаварийность проведения испытаний**

При проведении испытаний Заказчик должен обеспечить соблюдение требований безопасности, установленных ГОСТ 12.2.007.0–75, ГОСТ 12.2.007.3 – 75, «Правилами техники безопасности при эксплуатации электроустановок потребителей», и «Правилами технической эксплуатации электроустановок потребителей».

### **6.6. Порядок взаимодействия организаций, участвующих в испытаниях**

Участие в испытаниях принимают сотрудники:

- Заказчика; (Былинкина В. Г. Директор ЧОУ ДПО "Кванториум Фатоника")
- Исполнитель. (Черепанов А. М. Студент)

Далее участники испытаний совместно именуются Стороны.

Заказчик Приказом назначает срок проведения испытаний и Приемочную комиссию, которая должна включать в свой состав представителей Сторон.

Заказчик письменно извещает подчиненные и сторонние организации, которые должны принять участие в испытаниях.

Представители Заказчика и Исполнителя проводят все подготовительные мероприятия для проведения испытаний на объекте испытаний.

Испытания проводятся в соответствии с настоящим документом.

Заказчик осуществляет контроль за проведением испытаний, а также документирует ход проведения проверок в «Протоколе проведения испытаний».

## **6.7. Требования к персоналу, проводящему испытания**

Специальные требования не предъявляются



## 7. ТРЕБОВАНИЯ ПО

ПО должна сохранять целевое назначение при следующих значениях вероятностно-временных характеристик:

- Режим функционирования 24x7x365;
- Единый пользовательский интерфейс в виде одного приложения для доступа к функциям системы;
- Возможность смены версий специального ПО, обслуживание, подключение и отключение АРМ пользователей;
- Возможность расширения БД, а также масштабирования вычислительных ресурсов без повторного проектирования.

## **8. Материально-техническое обеспечение испытаний**

### **8.1. Технические средства, используемые во время испытаний**

Состав технических средств ПО определяется Спецификацией, Описанием комплексных технических средств.

### **8.2. Программные средства, используемые во время испытаний**

Для проведения испытаний на АРМ должны быть установлены необходимые Библиотеки.

## **9. Метрологическое обеспечение испытаний**

Программа испытаний не требует использования специализированного измерительного оборудования.

Таблица 2. Методика испытаний

<b>№</b>	<b>Наименование проверки</b>	<b>Исполнитель</b>	<b>Выполняемые действия</b>	<b>Ожидаемый результат</b>
1	Вход в приложение	Пользователь; Сис. админ.	Выполняется вход в приложение через ярлык на рабочем столе	Произошёл вход в приложение, активность не прерывается
2	Создание проекта	Пользователь; Сис. админ.	Пользователь нажимает кнопку «+ Новый проект»	Происходит успешное создание нового проекта