

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования «Казанский национальный исследовательский
технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации
(наименование института (факультета), филиала)

Кафедра Прикладной математики и информатики
(наименование кафедры)

(01.03.02) Прикладная математика и информатика
(шифр и наименование направления подготовки (специальности))

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

по дисциплине: «Проектирование и архитектура программных систем»
на тему: «Автоматизированная система учета аренды коммерческой
недвижимости»

Выполнили: студенты группы 4318

Данилаева С.Д.

Трифонов С.А.

Проверил:

ст. преподаватель каф. ПМИ

Александров А. Ю.

Подпись: _____

Дата сдачи: _____

Казань 2024

Оглавление

1. Введение	3
1.1 Анализ предметной области	3
1.2 Области применения системы	4
1.3 Бизнес преимущество и аналоги	5
1.3.1 Анализ аналогов	5
1.3.2 Преимущество нашей АИС	6
2. Цель и задача проекта	7
2.1 Цель разработки	7
2.2 Постановка задачи.....	7
3. Требования к системе.....	8
3.1 Общие требования.....	8
3.2 Бизнес требования.....	8
3.3 Функциональные требования	9
3.4 Требования к безопасности	10
3.5 Требования к надежности	10
3.6 Требования к лингвистическому обеспечению	10
3.7 Требования к программному обеспечению	11
3.8 Требования к техническому обеспечению	11
3.9 Требования к сопровождению	11
3.10 Проектные ограничения	11
4. Моделирование предметной области.....	12
4.1 Поиск акторов и вариантов использования.....	12
4.1.1 Выявление акторов.....	12
4.1.2 Выявление вариантов использования.....	13
4.1.3 Разработка диаграмм вариантов использования	16
4.2 Диаграмма потоков данных	17
4.3 SADT Диаграммы.....	22
4.3.1 Создание контекстной диаграммы.....	22
4.3.2 Декомпозиция контекстной диаграммы	23
4.4 ER Диаграмма.....	28
4.4.1 Список сущностей предметной области	28
4.4.2 Список атрибутов сущностей	31

4.4.3	Описание взаимосвязей между сущностями	34
4.4.4	Концептуальная диаграмма (ER – диаграмма)	35
4.5	Диаграмма классов	36
5.	Проектирование общей архитектуры системы.....	38
6.	Описание программы.....	40
6.1	Общие сведения.....	40
6.1.1	Обозначение и наименование программы	40
6.1.2	Программное обеспечение необходимое для функционирования программы	40
6.2	Функциональное назначение	41
6.2.1	Классы решаемых задач	41
6.2.2	Назначение программы	41
6.2.3	Сведения о функциональных ограничениях на применение	41
6.3	Описание логической структуры.....	41
6.3.1	Алгоритм программы	41
6.3.2	Используемые методы.....	43
6.3.3	Структура программы с описанием функций составных частей и связи между ними	43
6.3.4	Связи программы с другими программами	44
6.4	Используемые технические средства	44
6.5	Вызов и загрузка	44
7.	Руководство пользователя.....	45
8.	Заключение	53

1. Введение

1.1 Анализ предметной области

Область аренды коммерческих помещений включает в себя систематический контроль, учет и анализ всех операций, направленных на сдачу в аренду торговых и офисных площадей. Данный процесс состоит из ключевых этапов: регистрация арендаторов и объектов аренды, заключение и регулярное обновление договоров, мониторинг состояния арендных объектов, генерация отчетности и аналитики.

При заключении арендного договора арендатор предоставляет ряд документов. На основе предоставленных данных создается карточка арендатора, содержащая основные сведения о компании, условиях аренды, возможных сроках договора и прочей существенной информации.

Сотрудники отдела учета аренды отслеживают изменения в условиях договоров, такие как пересмотр арендной ставки, изменение сроков аренды, а также регистрируют любые дополнительные соглашения между сторонами. Эти данные аккумулируются в системе учета и поддерживаются в актуальном состоянии.

В процессе аренды площади фиксируются все платежи арендаторов, включая арендные платежи, коммунальные услуги и другие сопутствующие расходы. Система учета автоматически формирует отчеты о финансовых потоках и состоянии задолженности.

Кроме того, отдел занимается контролем состояния объектов аренды, проводит инвентаризацию торговых и офисных помещений, регистрирует неисправности и принимает меры по их устранению.

Основная проблема менеджеров, которые работают в отделе учета аренды — большой поток данных. Договоры, сроки внесения арендной платы, данные заказчика и самого объекта сложно систематизировать в привычных офисных программах. С расширением клиентской базы ситуация с ее ведением только усугубляется.

Автоматизация процессов аренды коммерческих помещений позволит значительно упростить и ускорить все вышеперечисленные операции, минимизировать ручной труд и повысить точность учета. Создание специализированной информационной системы обеспечит повышение эффективности бизнеса, облегчит мониторинг финансовых потоков и обеспечит возможность оперативного принятия управленческих решений.

1.2 Области применения системы

1. Управляющие компании и девелоперы: для компаний, занимающихся разработкой, управлением и эксплуатацией недвижимости, предоставляя инструменты для учета арендаторов, договоров аренды и финансовых потоков.

2. Малый и средний бизнес: предприниматели и владельцы малого бизнеса, сдающие в аренду площади для торговли и услуг могут использовать систему для оптимизации учета и управления арендными отношениями.

1.3 Бизнес преимущество и аналоги

1.3.1 Анализ аналогов

На рынке присутствует достаточное количество готовых решений по автоматизации учета аренды коммерческой недвижимости. В данный момент на рынке программных средств учета аренды недвижимости лидируют три компании: АрендаSoft, Фирма 1С и SOFTPRO.

Фирма 1С – российская компания, является крупным производителем различного программного обеспечения. Наиболее известным продуктом компании является «1С: Предприятие», часто используемое небольшими фирмами в качестве основы для собственных разработок.

SOFTPRO – украинская компания, специализирующаяся на разработке веб-ресурсов, веб-интеграции и модернизации различного ПО.

ArendaSoft – российская компания, разработавшая одноименное программное средство. Особенностью ценовой политики данной фирмы является возможность ежемесячной оплаты предоставляемых услуг и программного продукта.

В таблице 1.1 представлены аналоги разрабатываемого средства автоматизации процессов аренды объектов коммерческой недвижимости. Стоимость рассчитывается для 10 рабочих мест.

Табл. 1.1. Достоинства и недостатки рассматриваемых аналогов

Наименование	1С: Аренда и управление недвижимостью	ArendaSoft	SOFTPRO:Аренда+	RentSoft
Необходимость интернет-подключения	Нет	Да	Да	Да
Ведение бухгалтерского учета	Да	Нет	Нет	Нет
Ведение учета аренды	Да	Да	Да	Да
Визуализация объектов недвижимости	Да	Да	Да	Да
Возможность работы в WEB-интерфейсе	Да	Да	Да	Да
Автоматическое информирование клиентов о платежах и задолженностях	Нет	Да	Да	Да
Автоматическая подготовка и печать документов (договоры, счета, отчеты)	Да	Да	Нет	Да
Добавление нескольких зданий и информации о них в одном аккаунте	Да	Нет	Нет	Да
Пробный период	30 дней	-	20 дней	30 дней
Стоимость, руб.	203200	119900	145500	40000

В результате анализа мы обнаружили, что хотя 1С: Аренда и управление недвижимостью имеет более высокую стоимость, она предлагает дополнительные функциональные возможности, которые важны для больших компаний. Выбор между продуктами будет зависеть от конкретных потребностей и приоритетов организации.

1.3.2 Преимущество нашей АИС

Привлекательная экономичность и ориентированность на мелкий бизнес и частных предпринимателей делают нашу автоматизированную систему учета аренды торговых помещений уникальной на рынке. Наш фокус на отсутствие встроенной CRM системы и отсутствии бухгалтерского учета значительно снижает стоимость внедрения, что позволяет нашим клиентам с легкостью использовать современные технологии учета без излишних затрат. Также мы предлагаем альтернативу покупке программного продукта – подписка на АИС.

2. Цель и задача проекта

2.1 Цель разработки

Цель разработки - облегчить работу сотрудникам отдела учета аренды, обеспечить оперативный доступ ко всем операциям и аналитике из любого места и с любого устройства и как следствие, повысить эффективность бизнес-процесса.

2.2 Постановка задачи

Задача состоит в разработке и реализации программной системы типа “клиент-сервер” для отдела учета аренды помещений. Для выполнения задачи были определены следующие подзадачи:

- создание базы данных отдела учета аренды (серверная часть)
- разработка и создание web-сайта для работы с базой данных (клиентская часть)

Создание информационной системы в виде сайта для сотрудника отдела учета аренды и базы данных позволит автоматизировать работу отдела учета аренды недвижимости.

3. Требования к системе

3.1 Общие требования

Автоматизированная информационная система «RentSoft» должна быть организована по архитектуре «клиент-сервер», соответствовать требованиям надёжности и безопасности, иметь интуитивно понятный пользовательский интерфейс.

3.2 Бизнес требования

1. Повышение эффективности работы сотрудников: Сокращение времени на выполнение типовых операций и минимизация ошибок за счет автоматизации и упрощения рабочих процессов.

2. Автоматизация учета аренды: Система должна обеспечить полную автоматизацию процессов учета арендных операций, включая регистрацию арендаторов, объектов аренды, платежей и договоров.

3. Управление финансами: Эффективное управление и мониторинг финансовых потоков, связанных с арендной деятельностью, включая автоматизацию расчетов, учет коммунальных платежей и создание финансовых отчетов.

4. Доступ к актуальной информации в реальном времени: Повышение оперативности принятия решений за счет предоставления сотрудникам мгновенного доступа к актуальным данным о состоянии арендных объектов, договорных обязательствах и финансовых потоках.

5. Оптимизация взаимодействия с арендаторами: Автоматизация процессов коммуникации с арендаторами, включая отправку уведомлений о сроках платежей, продлении договоров и других важных моментах, что улучшает качество обслуживания и удовлетворенность клиентов.

3.3 Функциональные требования

Программный продукт должен удовлетворять следующему ряду функциональных требований:

1. Авторизация сотрудников
2. Добавление арендаторов

3. Регистрация и отслеживания технического состояния арендуемых объектов, включая инвентаризацию и учет неисправностей

4. Автоматическое информирование клиентов о платежах и задолженностях (СМС- и e-mail-рассылки)

5. Просмотр и поиск объектов недвижимости, клиентов, договоров

6. Автоматическая подготовка и печать документов (договоры, счета, отчеты)

7. Добавление нескольких зданий и информации о них в одном аккаунте

8. Отслеживание аренды нескольких зданий компании если они находятся по разным адресам

9. Добавление план-схемы зданий с этажами и секциями

10. Визуальное отображение сдаваемых помещений на поэтажных схемах и планах территорий

11. Визуальный контроль занятости площадей

12. Ведение учета арендных, коммунальных платежей и других переменных затрат

13. Автоматическое составление различных финансовых отчетов и их генерация в виде графиков, диаграмм и таблиц:

- Реестр объектов недвижимости
- Формирование диаграммы состояний объектов недвижимости
- Отчет по начислениям и выплатам арендной платы
- Анализ задержек платежей по аренде
- Анализ эффективности использования площадей
- Сведения о расходах на эксплуатацию объектов недвижимости
- План-фактный анализ расходов на эксплуатацию

3.4 Требования к безопасности

- Организация резервирования информации на внешних серверах для предотвращения ошибок в результате системных сбоев или физического повреждения оборудования;

- Утверждение внутренних нормативных документов, регламентирующих обработку данных и доступ к АИС;
- Обучение персонала всем аспектам работы с системой
- Блокировка ошибочных или преднамеренных вредоносных операций
- Контроль доступа, авторизация пользователей.

3.5 Требования к надежности

- Система должна быть способна поддерживать минимум 10 одновременно работающих пользователей.
- Минимальное время доступности должно составлять не менее 97%

3.6 Требования к лингвистическому обеспечению

- Языки программирования должны быть широко распространенными и иметь обширное сообщество разработчиков.
- Обеспечить поддержку стандартных кодировок данных, таких как UTF-8, для корректной обработки текстовой информации
- Реализовать механизмы для кодирования и декодирования данных при передаче и хранении информации
- Система должна поддерживать разнообразные форматы данных для ввода и вывода, включая текстовые, бинарные, XML, JSON и другие

3.7 Требования к программному обеспечению

Операционная система, поддерживающая работу с браузерами: Google Chrome, Yandex Browser, Mozilla Firefox, DuckDuck, Microsoft Edge последних версий.

3.8 Требования к техническому обеспечению

Для корректной работы системы требуется ПК или мобильное устройство с минимальными характеристиками:

Для ПК:

- процессор не менее 2 GHz
- не менее 4 ГБ оперативной памяти

Для мобильного устройства:

- не менее 4 ГБ оперативной памяти
- Операционная система: Android 7.0 и выше/iOS 11 и выше

Также требуется наличие стабильного интернет-соединения со скоростью не менее 10 Мбит/с для обеспечения доступа к системе и ее бесперебойной работы.

3.9 Требования к сопровождению

- Мониторинг работы системы: Непрерывный мониторинг состояния системы для своевременного выявления и устранения проблем с производительностью, доступностью и безопасностью.
- Обновление программного обеспечения: Регулярное обновление системы для устранения обнаруженных ошибок, повышения безопасности и расширения функционала в соответствии с изменяющимися потребностями пользователей и требованиями законодательства.

3.10 Проектные ограничения

Табл. 3.1. Этапы и сроки исполнения этапов проекта

Этапы работ	Сроки выполнения
Начало проекта	07.02.2024
Анализ предметной области	12.02.2024
Анализ конкурентов	14.02.2024
Разработка технических требований	16.02.2024
Разработка структуры системы	10.03.2024
Разработка прототипа	20.03.2024
Разработка серверной части	06.04.2024
Создание дизайн-макета	13.04.2024
Верстка макета	20.04.2024
Тестирование	27.04.2024
Создание руководства пользователя	03.05.2024
Развертывание проекта	03.05.2024
Завершение проекта	05.05.2024

Бюджет проекта: 200 000 (двести тысяч) рублей.

4. Моделирование предметной области

4.1 Поиск акторов и вариантов использования

4.1.1 Выявление акторов

На рисунке 4.1.1 представлены основные кандидаты в акторы системы.

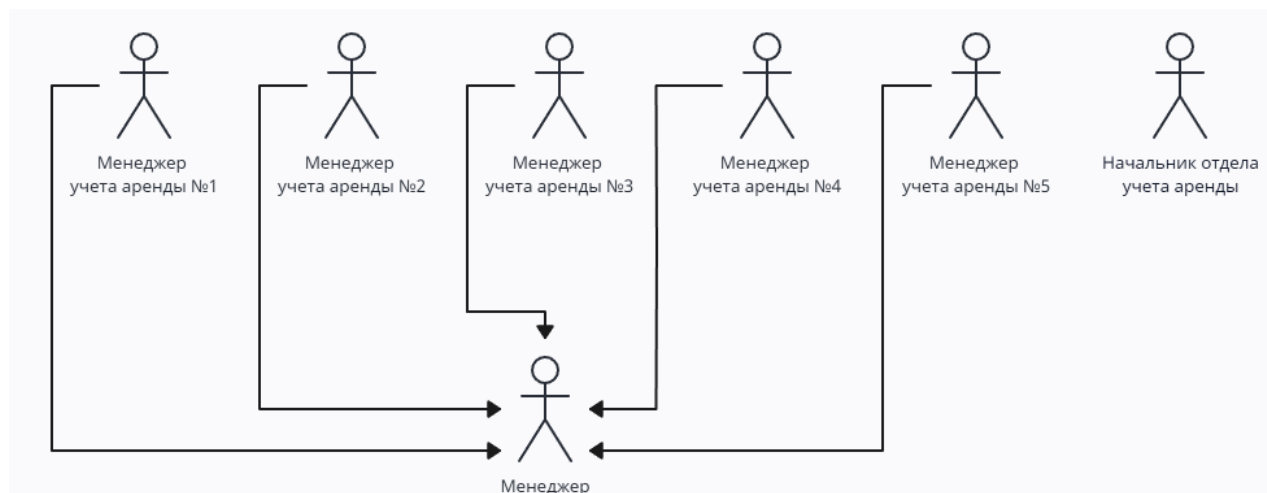


Рис. 4.1.1 Анализ акторов системы

Анализ сотрудников, которые будут пользоваться системой, показал, что начальник отдела учета аренды будет иметь больше прав доступа и функционал, чем менеджер учета аренды. Тем самым должности “Начальник отдела учета аренды” и “Менеджер учета аренды” не объединяются в одну сущность, см. рис. 1.

Краткое описание акторов представлено в таблице 4.1.1

Табл. 4.1.1. Выявление акторов

Актор	Краткое описание
Начальник отдела учета аренды	Добавление и контроль сотрудников, назначение сотрудникам объектов учета аренды, возможность выполнения рядовых задач
Менеджер	Выполнение всех рядовых задач: работа с клиентской базой (проверка договоров, оповещение об оплатах, анализ выплат), работа с базой объектов недвижимости (редактирование объектов и их состояния, сведения о расходах на содержание объекта, анализ прибыли)

4.1.2 Выявление вариантов использования

Выявленные варианты использования сведены в таблицу 4.1.2.

Табл. 4.1.2. Выявление вариантов использования

Основной актер	Наименование	Формулировка
Менеджер/ Начальник отдела учета аренды	Добавление арендаторов	Этот вариант использования позволяет менеджеру вносить данные о арендаторе (реквизиты компании), создав карточку арендатора
Менеджер/ Начальник отдела учета аренды	Изменение данных о арендаторе	Менеджер может откорректировать информацию о арендаторе
Менеджер/ Начальник отдела учета аренды	Удаление арендатора	При необходимости менеджер может удалить данные о арендаторе
Менеджер/ Начальник отдела учета аренды	Поиск арендатора	Используется менеджером для поиска нужной информации о арендаторе
Менеджер/ Начальник отдела учета аренды	Изменение данных об объекте аренды	Этот вариант использования позволяет менеджеру редактировать объект аренды
Менеджер/ Начальник отдела учета аренды	Поиск объекта аренды	При необходимости менеджер может найти информацию об объекте аренды
Менеджер/ Начальник отдела учета аренды	Заключение договора аренды	Данный вариант использования позволяет актору составить договор аренды с арендатором
Менеджер/ Начальник отдела учета аренды	Контроль за арендными платежами	Данный вариант использования позволяет актору следить за своевременной оплатой аренды
Менеджер/ Начальник отдела учета аренды	Информирование клиентов о платежах и задолженностях	Менеджер совершает СМС- и e-mail-рассылки с определенной информацией
Менеджер/ Начальник отдела учета аренды	Печать документов (договоров, счетов)	Менеджер с помощью функции «Пакетная выгрузка документов» экспортирует файлы с программы и имеет возможность распечатать их
Менеджер/ Начальник отдела учета аренды	Диаграмма состояний объектов недвижимости	Менеджер имеет визуальное представление о степени занятости объектов аренды

Менеджер/ Начальник отдела учета аренды	Анализ данных всех выплат, состояния объектов недвижимости, расходов на их эксплуатацию	Используется менеджером для составления отчетности по взаиморасчетам с арендаторами с аналитикой по периодам, услугам, объектам аренды, счетам на оплату и возможностью анализа расчетов по пени
Начальник отдела учета аренды	Предоставление доступа к объектам аренды	Начальник отдела учета аренды может дать доступ к определенным объектам недвижимости менеджерам своего отдела
Начальник отдела учета аренды	Ограничение доступа к объектам аренды	Начальник отдела учета аренды может ограничить доступ к определенным объектам недвижимости менеджерам своего отдела
Начальник отдела учета аренды	Добавление сотрудников	Добавление сотрудников в систему
Начальник отдела учета аренды	Удаление сотрудников	Удаление сотрудников из системы
Начальник отдела учета аренды	Редактирование данных о сотрудниках	Начальник отдела учета аренды может редактировать данные о сотрудниках
Начальник отдела учета аренды	Добавление объектов аренды	Менеджер может вносить данные об объектах аренды
Начальник отдела учета аренды	Удаление объекта аренды	Используется менеджером для удаления объекта аренды

4.1.3 Разработка диаграмм вариантов использования

Все варианты использования показаны на рис. 4.1.2



Рис. 4.1.2 Диаграмма прецедентов системы

4.2 Диаграмма потоков данных

В результате проектирования была разработана информационная модель системы, которая представляет из себя модель организации работы системы и схематично поясняет, как происходит функционирование автоматизированной информационной системы учета аренды помещений и получение выходных данных, т.е. процесс преобразования данных в информационной системе.

На рис. 4.2.1 Приведена контекстная диаграмма системы с единственным процессом ОБСЛУЖИТЬ, идентифицирующая внешние сущности КЛИЕНТ и МЕНЕДЖЕР.

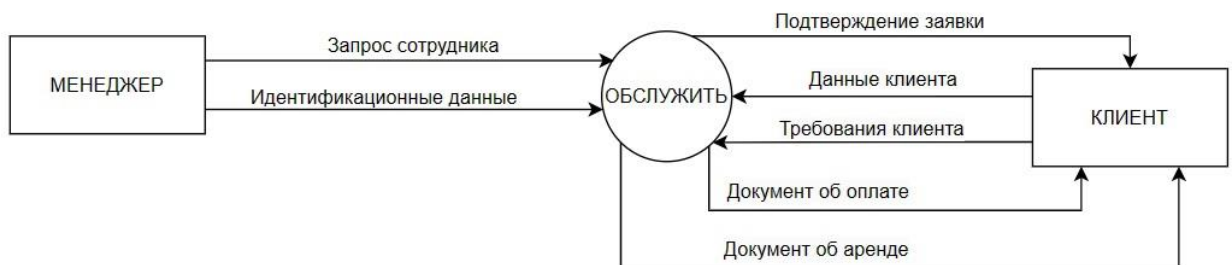


Рис. 4.2.1 Контекстная диаграмма.

Для аренды помещения клиенту необходимо предоставить ДАННЫЕ КЛИЕНТА (ФИО, номер телефона, реквизиты организации), а также свои требования. Обслуживание с позиций клиента, в свою очередь, должно обеспечить следующее:

- ПОДТВЕРЖДЕНИЕ ЗАЯВКИ
- выдать ДОКУМЕНТ ОБ АРЕНДЕ, ДОКУМЕНТ ОБ ОПЛАТЕ;

Опишем потоки данных, которыми обменивается проектируемая система с внешними объектами:

Контекстный процесс и МЕНЕДЖЕР должны обмениваться следующей информацией:

- ИДЕНТИФИКАЦИОННЫЕ ДАННЫЕ сотрудников;
- ЗАПРОС СОТРУДНИКА;

Контекстный процесс может быть детализирован DFD первого уровня как показано на рис. 4.2.2

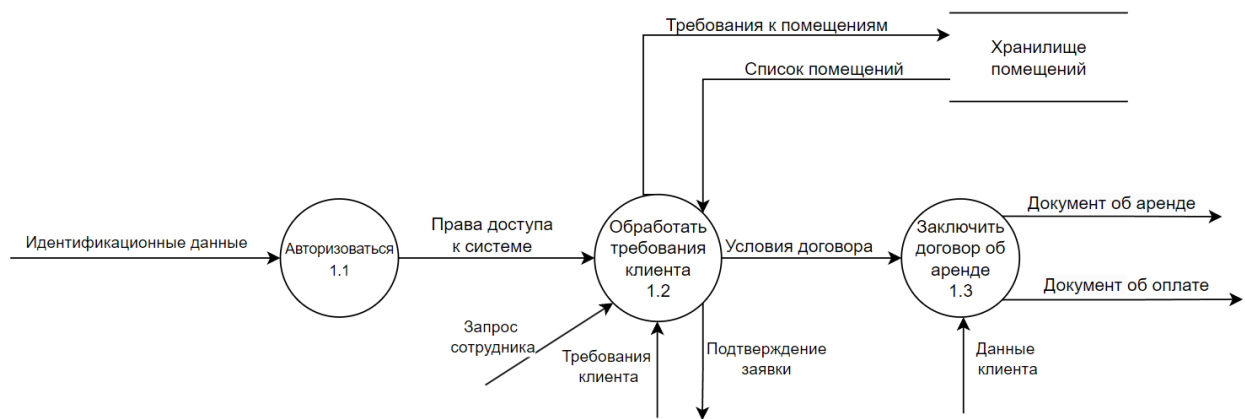


Рис. 4.2.2 DFD 1 уровня.

Данная диаграмма содержит 3 процесса и 1 хранилище.

Процесс 1.1 (АВТОРИЗОВАТЬСЯ) осуществляет авторизацию сотрудника в системе с возможностью дальнейшей отправки запросов к системе и имеет на входе/выходе следующие потоки:

- внешние входные потоки **ИДЕНТИФИКАЦИОННЫЕ ДАННЫЕ**;
- выходные **ПРАВА ДОСТУПА К СИСТЕМЕ**;

Процесс 1.2 (ОБРАБОТАТЬ ТРЕБОВАНИЯ КЛИЕНТА) осуществляет опрос и выявление конкретных требований клиента касательно объектов аренды и условий договора и имеет на входе/выходе следующие потоки:

- внешние входные **ТРЕБОВАНИЯ КЛИЕНТА, ЗАПРОС СОТРУДНИКА**;
- входные потоки **ПРАВА ДОСТУПА К СИСТЕМЕ, СПИСОК ПОМЕЩЕНИЙ**;
- выходные потоки **ТРЕБОВАНИЯ К ПОМЕЩЕНИЯМ, УСЛОВИЯ ДОГОВОРА, ПОДТВЕРЖДЕНИЕ ЗАЯВКИ**;

Процесс 1.3 (ЗАКЛЮЧИТЬ ДОГОВОР ОБ АРЕНДЕ) осуществляет заключение с клиентом договора, после согласования всех требований и имеет на входе/выходе следующие потоки:

- внешние входные потоки **ДАННЫЕ КЛИЕНТА**;
- входные потоки **УСЛОВИЯ ДОГОВОРА**;
- внешние выходные потоки **ДОКУМЕНТ ОБ АРЕНДЕ, ДОКУМЕНТ ОБ ОПЛАТЕ**;

Процесс 1.1 может быть детализирован с помощью DFD диаграммы 2-го уровня как показано на рис. 4.2.3

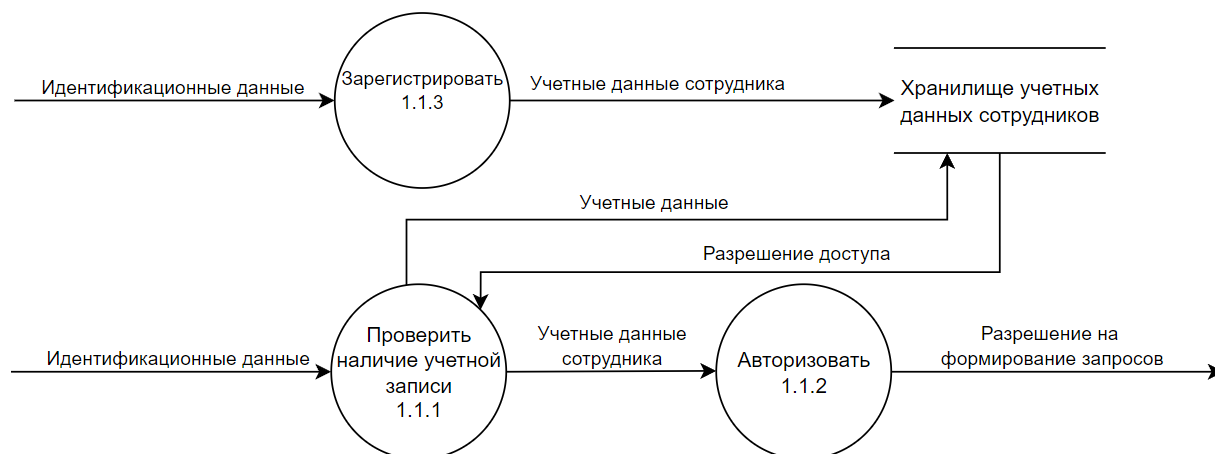


Рис.4.2.3

Данная диаграмма содержит 3 процесса и 1 хранилище.

Процесс 1.1.1 (ПРОВЕРИТЬ НАЛИЧИЕ УЧЕТНОЙ ЗАПИСИ) осуществляет проверку на наличие учетной записи сотрудника и имеет на входе/выходе следующие потоки:

- внешние входные потоки ИДЕНТИФИКАЦИОННЫЕ ДАННЫЕ;
- входные потоки РАЗРЕШЕНИЕ ДОСТУПА
- выходной поток УЧЕТНЫЕ ДАННЫЕ СОТРУДНИКА;

Процесс 1.1.2 (АВТОРИЗИРОВАТЬ) осуществляет авторизацию сотрудника в его аккаунте и имеет на входе/выходе следующие потоки:

- входные потоки ИДЕНТИФИКАЦИОННЫЕ ДАННЫЕ;
- выходной поток РАЗРЕШЕНИЕ НА ФОРМИРОВАНИЕ ЗАПРОСОВ;

Процесс 1.1.3 (ЗАРЕГИСТРИРОВАТЬ) осуществляет регистрацию начальником отдела нового сотрудника и имеет на входе/выходе следующие потоки:

- внешние входные потоки УЧЕТНЫЕ ДАННЫЕ СОТРУДНИКА;
- входные потоки ОТВЕТ УСПЕШНОЙ РЕГИСТРАЦИИ;
- выходной поток УЧЕТНЫЕ ДАННЫЕ СОТРУДНИКА;

Процесс 1.2 может быть детализирован с помощью DFD диаграммы 2-го уровня как показано на рис. 4.2.4

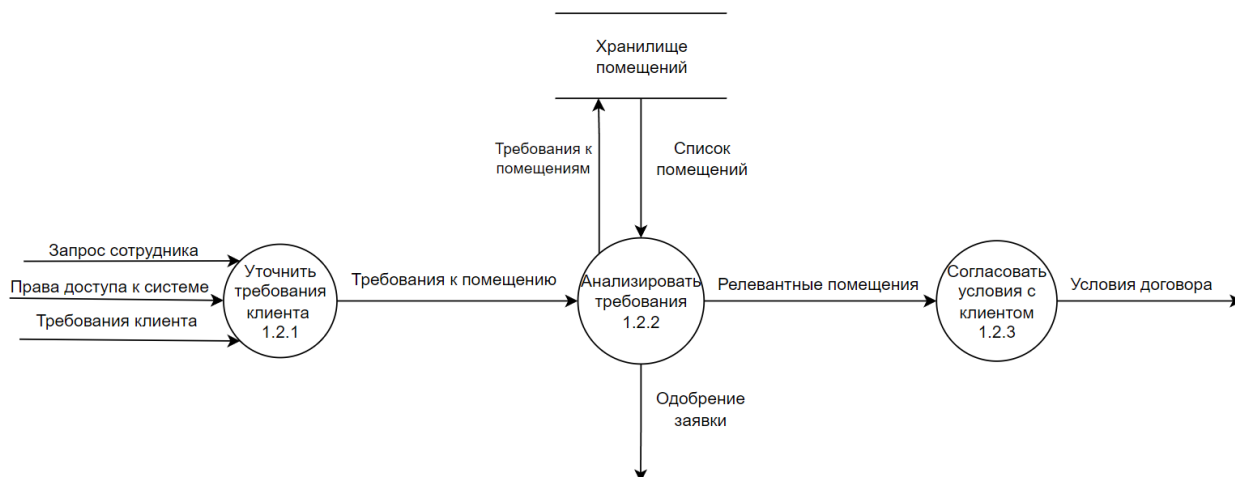


Рис. 4.2.4

Данная диаграмма содержит 3 процесса и 1 хранилище.

Процесс 1.2.1 (УТОЧНИТЬ ТРЕБОВАНИЯ КЛИЕНТА) осуществляет опрос и выявление конкретных требований клиента касательно объектов аренды и условий договора и имеет на входе/выходе следующие потоки:

- внешние входные потоки ПРАВА ДОСТУПА К СИСТЕМЕ, ЗАПРОС ОТ СОТРУДНИКА ОТДЕЛА, ТРЕБОВАНИЯ КЛИЕНТА;
- выходной поток ТРЕБОВАНИЯ К ПОМЕЩЕНИЮ;

Процесс 1.2.2 (АНАЛИЗИРОВАТЬ ТРЕБОВАНИЯ) осуществляет анализ текущего состояния объектов недвижимости и подбор наиболее подходящих под выявленные требования площадей и имеет на входе/выходе следующие потоки:

- входные потоки ТРЕБОВАНИЯ К ПОМЕЩЕНИЮ, СПИСОК ПОМЕЩЕНИЙ;
- выходной поток ТРЕБОВАНИЯ К ПОМЕЩЕНИЮ, РЕЛЕВАНТНЫЕ ПОМЕЩЕНИЯ, ОДОБРЕНИЕ ЗАЯВКИ;

Процесс 1.2.3 (СОГЛАСОВАТЬ УСЛОВИЯ С КЛИЕНТОМ) осуществляет согласование условий аренды с клиентом и имеет на входе/выходе следующие потоки:

- входные поток РЕЛЕВАНТНЫЕ ПОМЕЩЕНИЯ;
- внешний выходной поток УСЛОВИЯ ДОГОВОРА;

Процесс 1.3 может быть детализирован с помощью DFD диаграммы 2-го уровня как показано на рис. 4.2.5

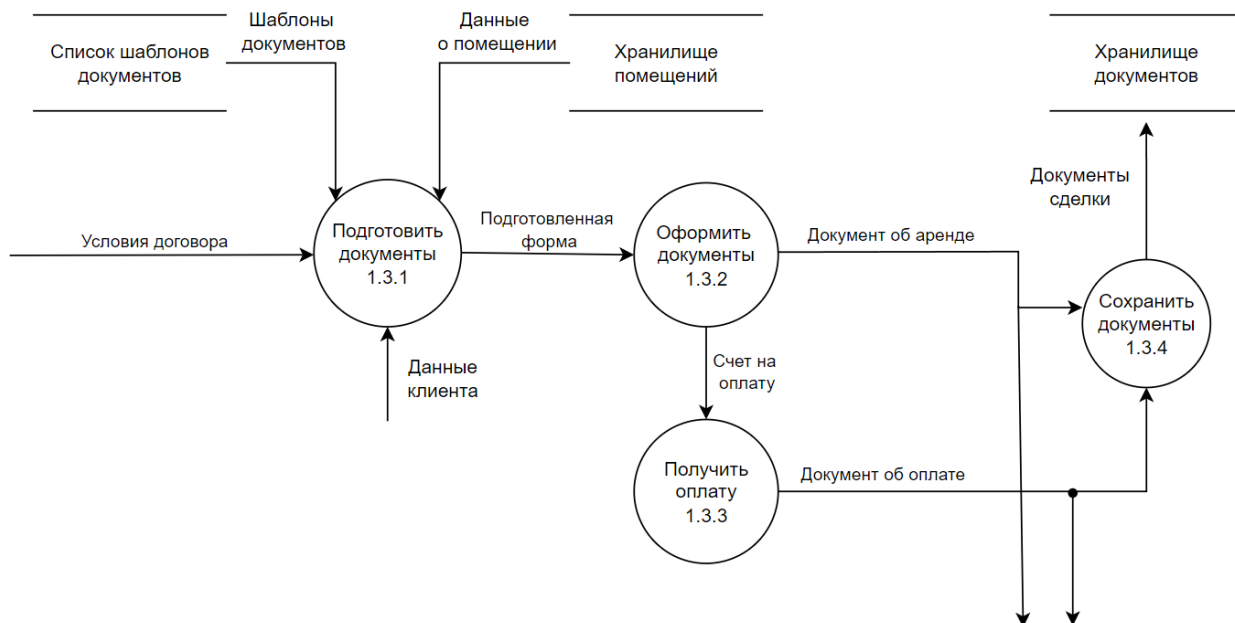


Рис. 4.2.5

Данная диаграмма содержит 4 процесса и 3 хранилища.

Процесс 1.3.1 (ПОДГОТОВИТЬ ДОКУМЕНТЫ) осуществляет подготовку документа для заключения договора и имеет на входе/выходе следующие потоки:

- внешние входные потоки УСЛОВИЯ ДОГОВОРА, ДАННЫЕ КЛИЕНТА;
- входные потоки ДАННЫЕ О ПОМЕЩЕНИИ, ШАБЛОНЫ ДОКУМЕНТОВ;
- выходной поток ПОДГОТОВЛЕННАЯ ФОРМА;

Процесс 1.3.2 (ОФОРМИТЬ ДОКУМЕНТЫ) осуществляет подписание и согласование договора обеими сторонами и имеет на входе/выходе следующие потоки:

- входные потоки ПОДГОТОВЛЕННАЯ ФОРМА;
- выходной поток ДОКУМЕНТ ОБ АРЕНДЕ, СЧЕТ НА ОПЛАТУ;

Процесс 1.3.3 (ПОЛУЧИТЬ ОПЛАТУ) осуществляет получение и подтверждение оплаты клиента и имеет на входе/выходе следующие потоки:

- входные поток СЧЕТ НА ОПЛАТУ;

- выходной поток ДОКУМЕНТ ОБ ОПЛАТЕ;

Процесс 1.3.4 (СОХРАНИТЬ ДОКУМЕНТЫ) осуществляет добавление в базу данных документов сделки и имеет на входе/выходе следующие потоки:

- входные поток ДОКУМЕНТ ОБ АРЕНДЕ, ДОКУМЕНТ ОБ ОПЛАТЕ;
- внешний выходной поток ДОКУМЕНТЫ СДЕЛКИ;

4.3 SADT Диаграммы

4.3.1 Создание контекстной диаграммы

Рассматривается автоматизированная система учета аренды коммерческой недвижимости. Программа позволяет автоматизировать работу отдела учета аренды, что облегчит труд сотрудников, обеспечит оперативный доступ ко всем операциям и аналитике из любого места и с любого устройства и как следствие, повысит эффективность бизнес-процесса.

Основные процедуры в системе таковы:

Менеджеры с помощью АИС:

1. Добавляют, изменяют и удаляют сведения об арендаторах
2. Добавляют, изменяют и удаляют данные об объекте аренды
3. Заключают договора аренды и создают договор об оплате
4. Контролируют оплату арендных платежей
5. Информировать клиентов о платежах и задолженностях
6. Анализируют данные всех выплат, состояние объектов недвижимости, расходы на их эксплуатацию

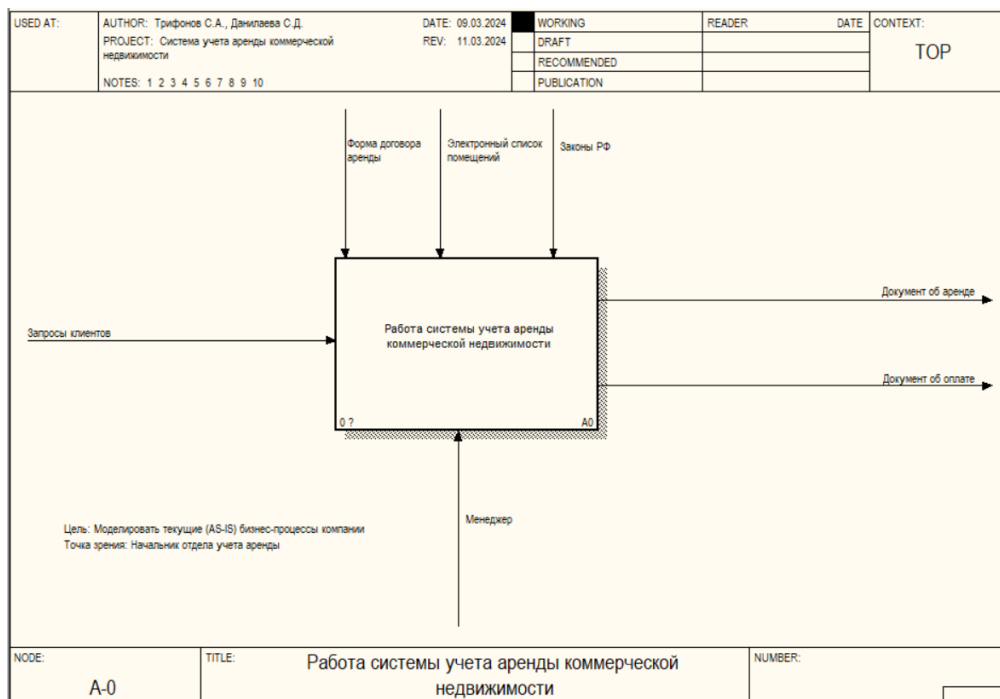


Рис. 4.3.1 Диаграмма А-0.

Табл. 4.3.1. Стрелки контекстной диаграммы

Имя стрелки (Arrow Name)	Определение стрелки (Arrow Definition)	Тип стрелки (Arrow Type)
Менеджер	Работа с клиентской базой (проверка договоров, оповещение об оплатах, анализ выплат), работа с базой объектов недвижимости (редактирование объектов и их состояния, сведения о расходах на содержание объекта, анализ прибыли), работа с системой посредством персонального компьютера	Mechanism
Запросы клиентов	Запросы информации, заказы, техподдержка и т. д.	Input
Формы договора аренды	Совокупность реквизитов, расположенных в определенной последовательности в документе	Control
Электронный список помещений	Совокупность всех арендуемых и свободных помещений	Control
Законы РФ	Совокупность правовых норм и положений, которые регулируют отношения между арендодателем и арендатором	Control
Документ об аренде	Официальное письменное соглашение между арендодателем и арендатором, в котором указываются права и обязанности сторон по использованию арендуемого имущества	Output
Документ об оплате	Официальный документ, подтверждающий факт оплаты аренды или иных платежей, связанных с арендой недвижимости	Output

4.3.2 Декомпозиция контекстной диаграммы

В таблицах 4.3.2 и 4.3.3 приведено определение элементов декомпозированной контекстной диаграммы.

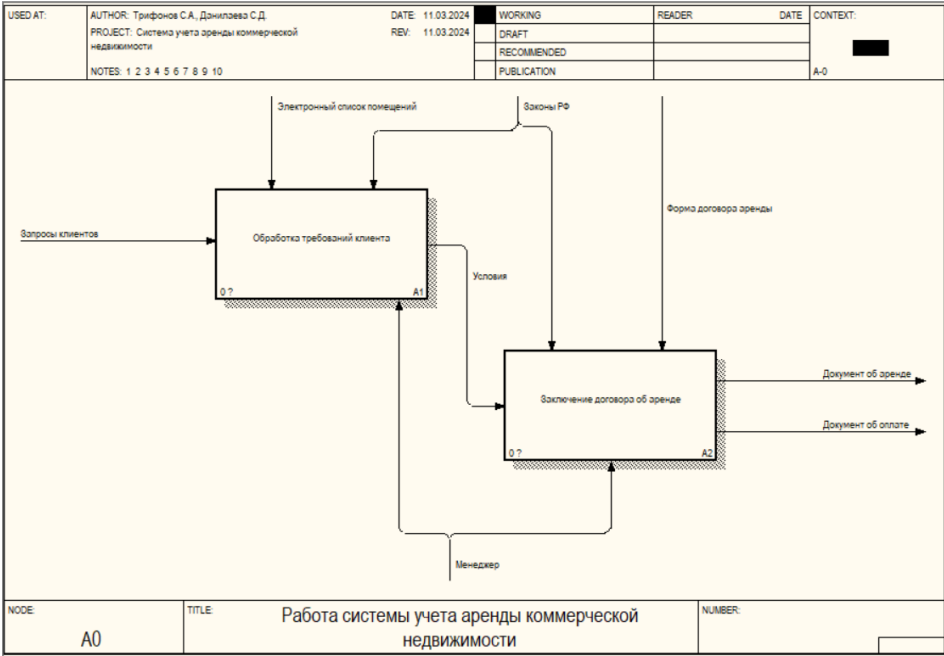


Рис. 4.3.2 Декомпозиция контекстной диаграммы A0

Табл. 4.3.2. Работы диаграммы декомпозиции A0

Имя работы (Activity Name)	Определение (Definition)
Обработка требований клиента	Опрос и выявление конкретных требований клиента касательно объектов аренды и условий договора
Заключение договора об аренде	Заключение с клиентом договора, после согласования всех требований

Табл. 4.3.3. Стрелки декомпозированной контекстной диаграммы A0

Имя стрелки (Arrow Name)	Источник стрелки (Arrow Source)	Тип источника стрелки (Arrow Source Type)	Назначение стрелки (Arrow Dest.)	Тип назначения стрелки (Arrow Dest. Type)
Менеджер	Граница диаграммы		Заключение договора об аренде	Mechanism
			Обработка требований клиента	Mechanism
Запросы клиентов	Граница диаграммы		Обработка требований клиента	Input
Электронный список помещений	Граница диаграммы		Обработка требований клиента	Control
Законы РФ	Граница диаграммы		Обработка требований клиента	Control

			Заклучение договора об аренде	Control
Форма договора аренды	Границы диаграммы		Заклучение договора об аренде	Control
Условия	Обработка требований клиента	Output	Заклучение договора об аренде	Input
Документ об аренде	Заклучение договора об аренде	Output		
Документ об оплате	Заклучение договора об аренде	Output		

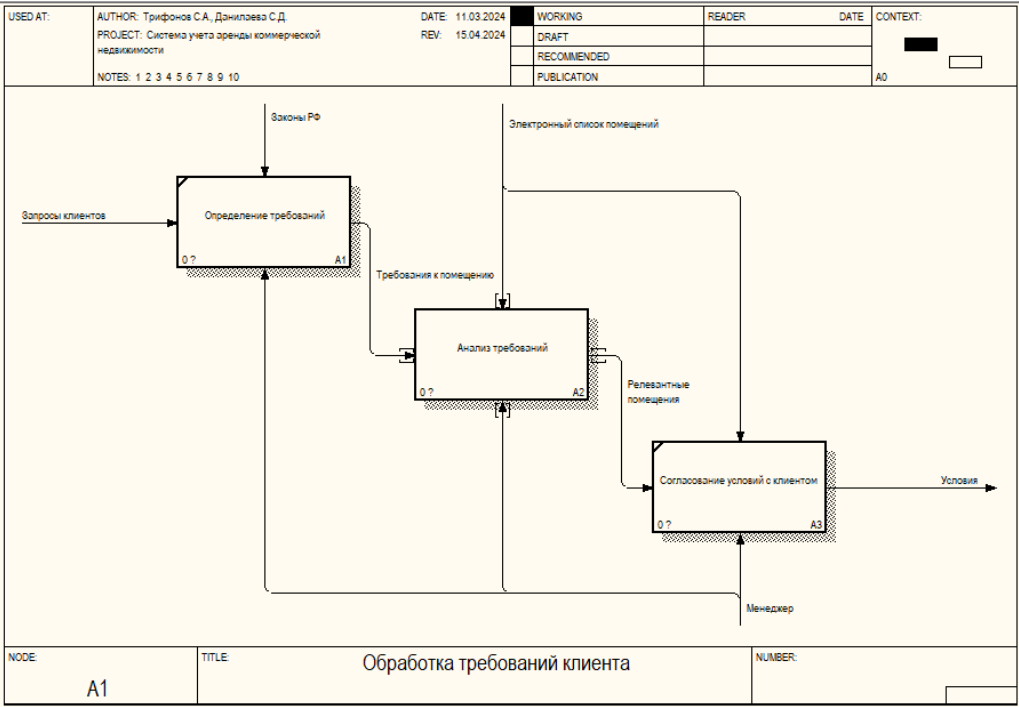


Рис. 4.3.3 Диаграмма декомпозиции A1

Табл. 4.3.4. Работы диаграммы декомпозиции A1

Имя работы (Activity Name)	Определение (Definition)
Определение требований	Опрос и выявление конкретных требований клиента касательно объектов аренды и условий договора
Анализ требований	Анализ текущего состояния объектов недвижимости и подбор наиболее подходящих под выявленные требования площадей
Согласование условий с клиентом	Согласование условий аренды с клиентом

Табл. 4.3.5. Стрелки диаграммы декомпозиции A1

Имя стрелки (Arrow Name)	Источник стрелки (Arrow Source)	Тип источника стрелки (Arrow Source Type)	Назначение стрелки (Arrow Dest.)	Тип назначения стрелки (Arrow Dest. Type)
Менеджер	Граница диаграммы		Определение требований	Mechanism
			Анализ требований	Mechanism
			Согласование условий с клиентом	Mechanism
Запросы клиентов	Граница диаграммы		Определение требований	Input
Электронный список помещений	Граница диаграммы		Анализ требований	Control
			Согласование условий с клиентом	Control
Законы РФ	Граница диаграммы		Определение требований	Control
Требования к помещению	Определение требований	Output	Анализ требований	Input
Релевантные помещения	Анализ требований	Output	Согласование условий с клиентом	Input
Условия	Согласование условий с клиентом	Output		

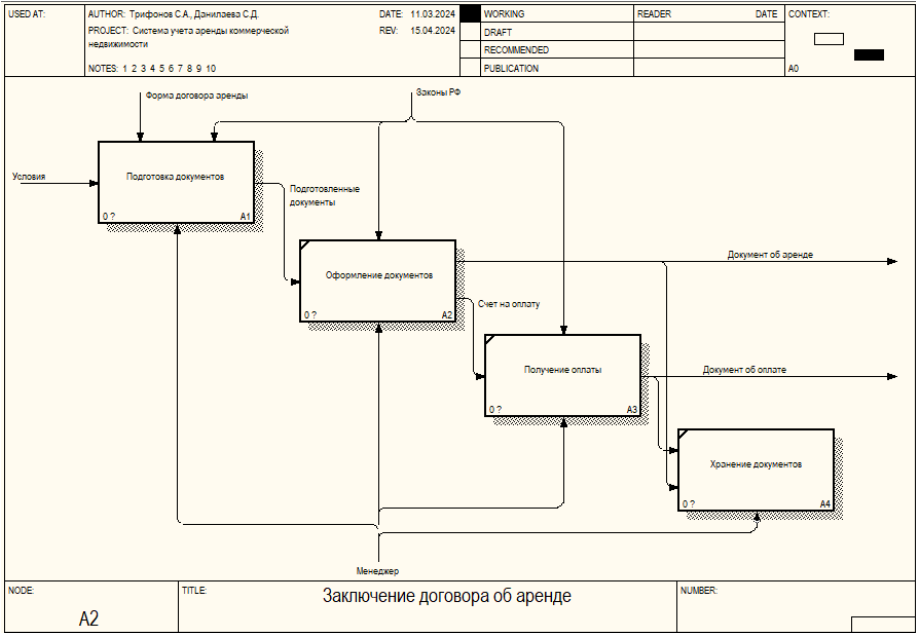


Рис. 4.3.4 Диаграмма декомпозиции A2

Табл. 4.3.6. Работы декомпозированной контекстной диаграммы A2

Имя работы (Activity Name)	Определение (Definition)
Подготовка документов	Подготовка пакета документов для заключения договора об аренде
Оформление документов	Процесс обработки, проверки и оформления всех документов
Получение оплаты	Получение оплаты за арендуемое клиентом помещение
Хранение документов	Сохранение документов сделки в системе

Табл. 4.3.7. Стрелки декомпозированной контекстной диаграммы A2

Имя стрелки (Arrow Name)	Источник стрелки (Arrow Source)	Тип источника стрелки (Arrow Source Type)	Назначение стрелки (Arrow Dest.)	Тип назначения стрелки (Arrow Dest. Type)
Менеджер	Граница диаграммы		Подготовка документов	Mechanism
			Оформление документов	Mechanism
			Получение оплаты	Mechanism
			Хранение документов	Mechanism

Условия	Граница диаграммы		Подготовка документов	Input
Форма договора аренды	Граница диаграммы		Подготовка документов	Control
Законы РФ	Граница диаграммы		Подготовка документов	Control
			Оформление документов	Control
			Получение оплаты	Control
Подготовленные документы	Подготовка документов	Output	Оформление документов	Input
Счет на оплату	Оформление документов	Output	Получение оплаты	Input
Документ об аренде	Оформление документов	Output		
Документ об аренде	Оформление документов	Output	Хранение документов	Input
Документ об оплате	Получение оплаты аренде	Output		
Документ об оплате	Оформление документов	Output	Хранение документов	Input

4.4 ER Диаграмма

4.4.1 Список сущностей предметной области

Проектируемая система должна выполнять следующие действия:

- Хранить информацию о сотрудниках системы;
- Хранить информация о пользователях, для доступа к системе;
- Хранить информацию о зданиях, содержащих объекты аренды;
- Хранить информацию об арендных помещениях;
- Отслеживать статус помещения (свободно, в ремонте и пр.)
- Хранить информацию об арендаторах;
- Хранить информацию об арендных соглашениях;
- Хранить информацию о счетах за обслуживание помещения;
- Хранить заключенные договоры;

- Хранить внесенные платежи;
- Отслеживать категорию платежа (платеж за свет, воду, газ и пр.)

Чтобы выделить сущности системы и определить, как они будут связаны, проанализируем работу проектируемой системы. Каждый сотрудник перед использованием системы должен быть внесен в базу начальником отдела, который регистрирует сотрудника, указав его данные (ФИО, номер телефона, фото при наличии) и пользователя, выдав ему логин и пароль. Учетная запись начальника отдела формируется при проектировании системы на этапе заполнения ее баз данных. Каждый пользователь может быть ассоциирован только с одним сотрудником.

Здания и арендные помещения регистрируются в системе на этапе ее проектирования. Дальнейшее их редактирование, а именно добавление и удаление, доступно только начальнику отдела. Сотрудники в свою очередь назначаются кураторами конкретных помещений и имеют доступ к редактированию информации об объектах аренды, а именно их описания и информации о текущем статусе помещения, его площади. Каждое помещение имеет только одного куратора, в то время как каждый куратор имеет в распоряжении несколько помещений.

Для каждого помещения ведется учет текущих счетов за коммунальные и прочие услуги. Учет включает в себя счет за свет, газ, воду, отопление, а также ремонтные работы. Одно помещение может иметь несколько счетов за обслуживание, в то время как каждый счет присваивается конкретному помещению.

Каждый объект аренды находится в своем здании. Информация о здании содержит адрес, количество арендных помещений и их общую площадь. Каждое здание имеет несколько помещений, но помещение принадлежит только одному зданию.

Каждое помещение имеет один конкретный статус текущего состояния. У нескольких помещений может быть один и тот же статус.

У объектов аренды может быть только один арендатор или не быть вовсе. Арендатор в свою очередь может иметь несколько помещений.

Информация об арендаторах вносится сотрудниками при заключении арендного соглашения. При этом необходимо указать следующую информацию: ФИО арендатора, его почту, наименование организации, ОГРН организации, ИНН организации, паспортные данные арендатора, БИК организации и расчетный счет банка организации. После добавления арендатора ему присваиваются так же поля: задолженность по арендным выплатам и выплатам по обслуживанию помещения, а также общая сумма выплат.

Арендное соглашение заключается на одно помещение для одного арендатора. В свою очередь помещение может иметь несколько арендных соглашений за разные периоды, арендатор может иметь несколько арендных соглашений за раз.

При заключении соглашения утверждается и вносится в базу следующая информация: дата заключения и завершения договора, дата списания арендного ежемесячного платежа (конкретное число для каждого месяца).

После заключения арендного соглашения, необходимо после подписания занести в базу данных копию договора, подписанного обеими сторонами. Каждое арендное соглашение содержит один договор, точно, как и договор заключается на базе одного конкретного арендного соглашения.

Договор может иметь один изменяемый статус его актуальности. Один и тот же статус может иметь несколько договоров.

После подписания договора необходимо начать учет ежемесячных арендных платежей и платежей за обслуживание помещения. Для этого сотрудниками в базу данных вносится каждый новый платеж арендатора.

Платеж должен быть категорирован (ежемесячный арендный платеж, платеж за свет, за газ и пр.). Платеж привязывается к одному конкретному арендному соглашению, на основании которого он был совершен. Следовательно, на основе одного арендного соглашения может быть заключено множество платежей.

Таким образом, список сущностей предметной области:

- *Сотрудник;*
- *Пользователь;*
- *Здание;*
- *Арендное помещение;*
- *Статус помещения;*
- *Арендатор;*
- *Счет за обслуживание помещения;*
- *Арендное соглашение;*
- *Договор;*
- *Платеж;*
- *Категория платежа;*

4.4.2 Список атрибутов сущностей

Сущность *Сотрудник* имеет следующие атрибуты:

- Идентификатор сотрудника;
- Фамилия сотрудника;
- Имя сотрудника;
- Отчество сотрудника;
- Номер телефона;
- Фото сотрудника.

Сущность *Пользователь* имеет следующие атрибуты:

- Идентификатор пользователя;
- Логин пользователя;
- Пароль пользователя.

Сущность *Здание* имеет следующие атрибуты:

- Идентификатор здания;
- Название;
- Адрес;
- Общая площадь помещений;

- Общее количество помещений;

Сущность *Арендное помещение* имеет следующие атрибуты:

- Идентификатор арендного помещения;
- Площадь помещения;
- Описание помещения;
- Этаж;
- Номер помещения;
- Куратор помещения;
- Статус помещения.

Сущность *Статус помещения* имеет следующие атрибуты:

- Идентификатор статуса;
- Описание статуса;

Сущность *Арендатор* имеет следующие атрибуты:

- Идентификатор арендатора;
- Фамилия арендатора;
- Имя арендатора;
- Отчество арендатора;
- Почта арендатора;
- Номер арендатора;
- Наименование организации;
- ОГРН организации;
- ИНН организации;
- Серия паспорта арендатора;
- Номер паспорта арендатора;
- БИК банка;
- Расчетный счет банка;
- Общая сумма выплат;
- Текущая задолженность по арендной(ым) плате(ам);
- Текущая задолженность по обслуживанию помещения(ий).

Сущность *Счет за обслуживание помещения* имеет следующие атрибуты:

- Идентификатор счета;
- Идентификатор помещения;
- Счет за газ;
- Счет за свет;
- Счет за воду;
- Счет за отопление;
- Счет за ремонтные работы;
- Дата создания счета.

Сущность *Арендное соглашение* имеет следующие атрибуты:

- Идентификатор арендного соглашения;
- Идентификатор арендатора;
- Идентификатор помещения.
- Сумма ежемесячных выплат;
- Дата заключения соглашения;
- Дата завершения соглашения;
- Дата списания ежемесячного арендного платежа.

Сущность *Договор* имеет следующие атрибуты:

- Идентификатор договора;
- Идентификатор арендного соглашения;
- Дата подписания договора;
- Копия договора;
- Статус договора.

Сущность *Платеж* имеет следующие атрибуты:

- Идентификатор платежа;
- Идентификатор арендного соглашения;
- Сумма платежа;
- Дата платежа;
- Копия платежа;

- Категория платежа.

Сущность *Категория платежа* имеет следующие атрибуты:

- Идентификатор категории;
- Наименование категории платежа.

4.4.3 Описание взаимосвязей между сущностями

Сущности «Сотрудник» и «Пользователь» связаны друг с другом отношением типа один-к-одному, т.е. «Каждый сотрудник должен являться пользователем» и «Каждый пользователь может представлять собой сотрудника».

Сущности «Здание» и «Арендное помещение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждое здание содержит несколько арендных помещений» и «Каждое арендное помещение находится в одном здании».

Сущности «Статус помещения» и «Арендное помещение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждое помещение имеет один конкретный статус» и «Каждый статус может принадлежать множеству помещений».

Сущности «Пользователь» и «Арендное помещение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждый пользователь курирует несколько помещений» и «Каждое помещение находится в распоряжении одного пользователя».

Сущности «Арендатор» и «Арендное помещение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждый арендатор может иметь несколько помещений» и «Каждое помещение принадлежит только одному арендатору».

Сущности «Арендатор» и «Арендное соглашение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждый арендатор может заключить несколько арендных соглашений» и «Каждое соглашение может быть заключено только одним арендатором».

Сущности «Арендное помещение» и «Арендное соглашение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждое арендное соглашение имеет одно помещение» и «Каждое помещение может иметь множество арендных соглашений (за разный период времени)».

Сущности «Арендное соглашение» и «Договор» связаны друг с другом отношением типа один-к-одному, т.е. «Каждый договор составлен на основе одного арендного соглашения» и «Каждое арендное соглашение имеет один договор».

Сущности «Счет за обслуживание помещения» и «Арендное помещение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждое помещение имеет множество счетов (за разный период времени)» и «Каждый счет принадлежит только одному помещению».

Сущности «Платеж» и «Арендное соглашение» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждое арендное соглашение может иметь несколько платежей» и «Каждый платеж ссылается на конкретное арендное соглашение».

Сущности «Категория платежа» и «Платеж» связаны друг с другом отношением типа один-ко-многим, т.е. «Каждый платеж имеет конкретную категорию» и «Каждая категория может присваиваться множеству платежей».

4.4.4 Концептуальная диаграмма (ER – диаграмма)

Таким образом, концептуальная диаграмма (ER – диаграмма) выглядит следующим образом (см. **Ошибка! Источник ссылки не найден.**).

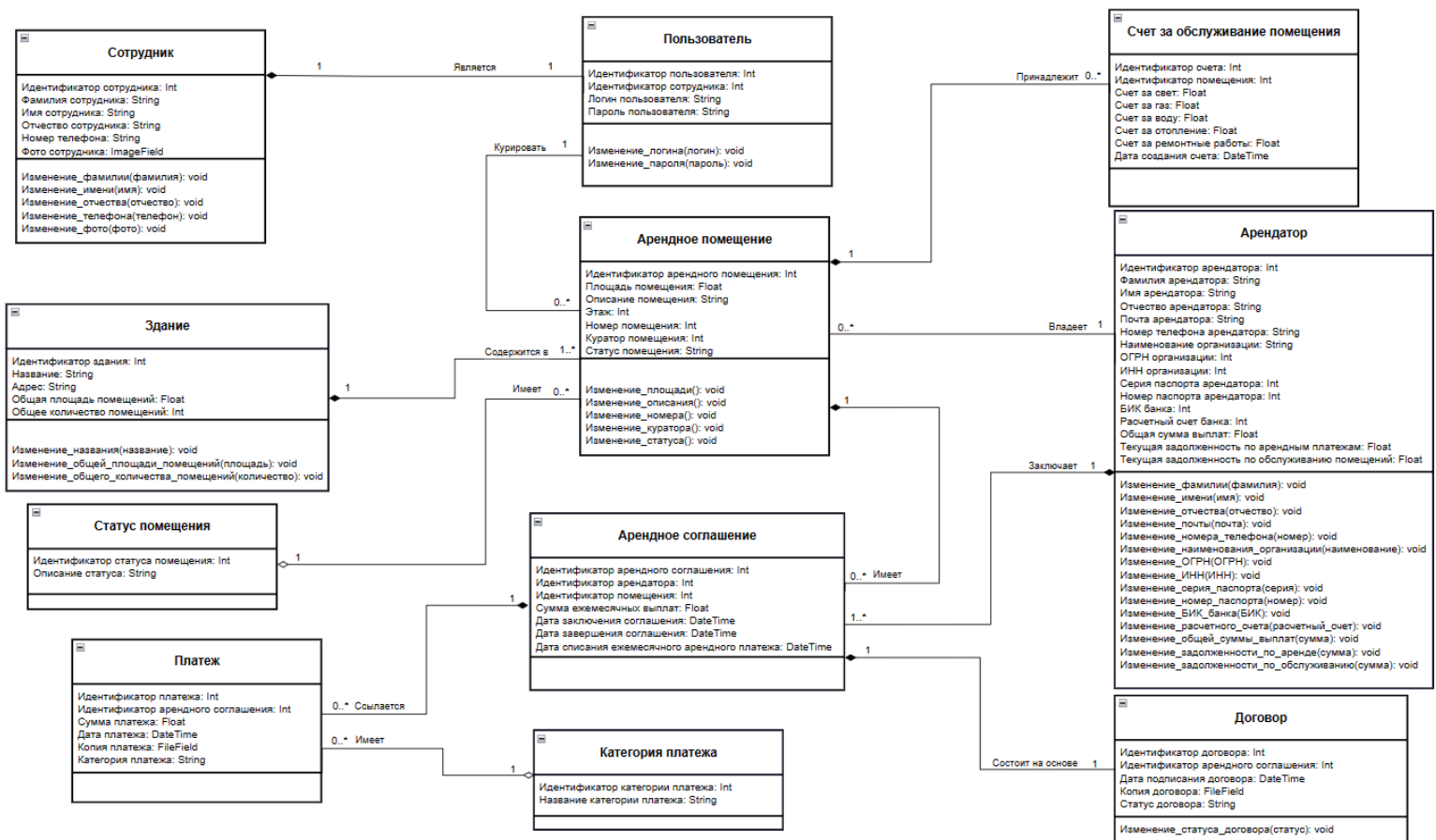
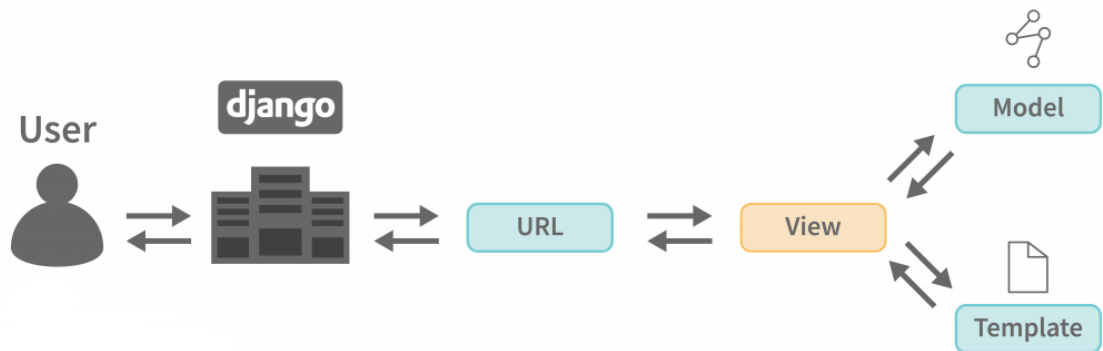


Рис. 4.4.2 – Диаграмма классов «АС учета аренды коммерческой недвижимости»

5. Проектирование общей архитектуры системы

При разработке применялась стандартная архитектура, используемая в Django – MVC (Model – View – Controller)



Ключевые компоненты архитектуры:

Models: Модели представляют собой структуру данных и определяют поля и их типы, а также отношения между различными моделями. Они используются для взаимодействия с базой данных и предоставляют API для создания, чтения, обновления и удаления записей.

Views: Образы отвечают за обработку запросов и формирование ответов. Они могут получать данные из моделей, обрабатывать ввод пользователя, выполнять логику приложения и рендерить шаблоны для формирования HTML-страниц.

Templates: Шаблоны представляют собой HTML-файлы с дополнительными тегами и фильтрами, которые позволяют динамически генерировать контент. Они используются для разделения представления и логики приложения, чтобы облегчить разработку и сопровождение.

URL dispatcher: URL-диспетчер отвечает за маршрутизацию запросов к соответствующим образам. Он использует шаблоны URL для сопоставления запросов с определенными образами и передачи им необходимых параметров.

Объекты технических и программных средств

Технические средства:

- *Сервер: Обработка запросов и взаимодействие между приложением и базой данных.*

- Рабочие станции: *ПК или смартфоны пользователей с доступом в интернет.*

Программные средства:

- Веб-приложение: Интерфейс для менеджеров.
- Серверное ПО: Логика обработки данных и взаимодействие с Базой данных.
- База данных: Хранение данных о помещениях, договорах, счетах, арендаторах и платежах.

Взаимодействие между объектами:

- Веб-приложение ↔ Сервер: Передача каких-либо данных для взаимодействия с Базой данных.
- Сервер ↔ База данных: Чтение и запись данных из Базы данных.
- ПК или смартфон пользователей ↔ Веб-приложение: Доступ сотрудников к системе учета аренды помещений.

Распределение требований к системе:

- Функциональные требования:
 - Вход в систему: Веб-приложение, Сервер.
 - Редактирование данных: Веб-приложение, Сервер, База данных.
 - Просмотр аналитики: Веб-приложение, Сервер, База данных.
- Нефункциональные требования:
 - Производительность: Сервер, База данных.
 - Безопасность: Веб-приложение (авторизация), Сервер (обработка запросов), База данных (шифрование данных)

Диаграмма архитектуры (в UML, кратко):

- Сервер (обрабатывает запросы) и обращается к Базе данных (хранит данные).
- Веб-приложение (интерфейс) обращается к Серверу.
- ПК пользователей (используют интерфейс) взаимодействуют с Веб-приложением.

6. Описание программы

6.1 Общие сведения

6.1.1 Обозначение и наименование программы

Название программы – «RentSoft».

6.1.2 Программное обеспечение необходимое для функционирования программы

Для функционирования веб-приложения, со стороны сервера и клиента требуется стабильное подключение к интернету. Со стороны клиента необходимо:

1. Убедиться, что браузер поддерживает последние веб-стандарты, для этого рекомендуется использовать последние версии браузеров Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.
2. Разрешить использование cookies в настройках браузера, чтобы обеспечить корректную работу приложения.
3. Убедиться, что есть доступ к порту, на котором запущено приложение (по умолчанию это порт 8000), и что сетевой администратор не блокирует доступ к этому порту.

Со стороны сервера необходимо:

1. Убедиться, что сервер поддерживает последние версии Python и Django, для этого рекомендуется использовать последние версии этих программных продуктов.
2. Убедиться, что сервер имеет достаточное количество памяти (рекомендуется не меньше 20Гб).
3. Настроить веб-сервер для работы с приложением Django, для этого необходимо указать путь к файлам приложения, настроить виртуальное окружение, если оно используется, и указать другие необходимые параметры.

6.1.3 Языки программирования, на которых написана программа

Программа написана с использованием языка Python и декларативного языка программирования SQL.

6.2 Функциональное назначение

6.2.1 Классы решаемых задач

Система позволяет отслеживать статистику: текущие и предыдущие арендные соглашения, состояния выплат и задолженностей арендаторов, состояние помещений и расходов на них. Также система позволяет добавлять новых арендаторов, новые выплаты и арендные соглашения.

6.2.2 Назначение программы

«RentSoft» — это программная система, предназначенная для автоматизации учета аренды коммерческой недвижимости. К назначению системы относится:

- Повышение эффективности работы компании
- Оптимизация труда менеджеров
- Устранение рутинных операций

6.2.3 Сведения о функциональных ограничениях на применение

«RentSoft» может работать только с устройствами, имеющими доступ к сети интернет.

Операционная система устройства должна поддерживать работу с браузерами: Google Chrome, Yandex Browser, Mozilla Firefox, DuckDuck, Microsoft Edge последних версий.

6.3 Описание логической структуры

6.3.1 Алгоритм программы

Пользователь получает доступ к контенту приложения при открытии его страницы в браузере. Веб-приложение, в зависимости от текущей страницы и действий пользователя инициирует запросы к серверу, которые запрашивает данные из базы данных. Для получения полного контента приложения пользователю необходимо пройти процесс авторизации.

Структурная схема алгоритма работы приложения приведена на рис. 6.1 и рис. 6.2.

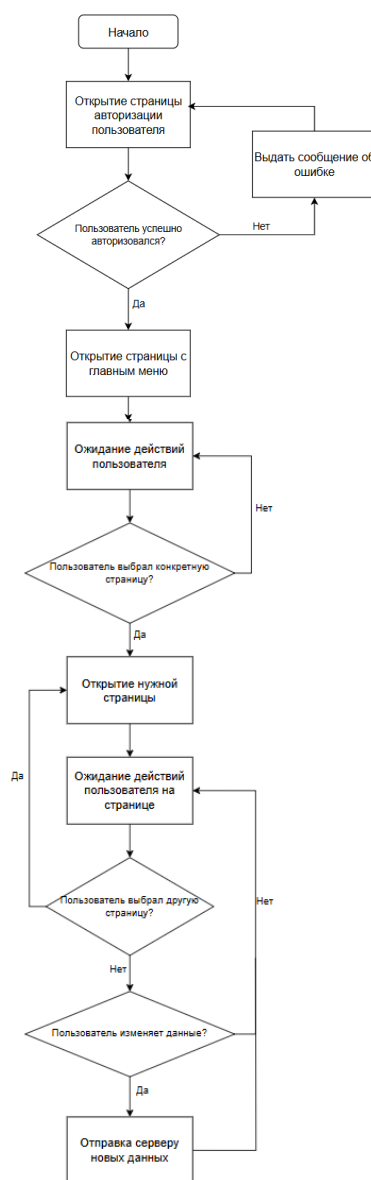


Рис. 6.1 Структурная схема алгоритма работы приложения

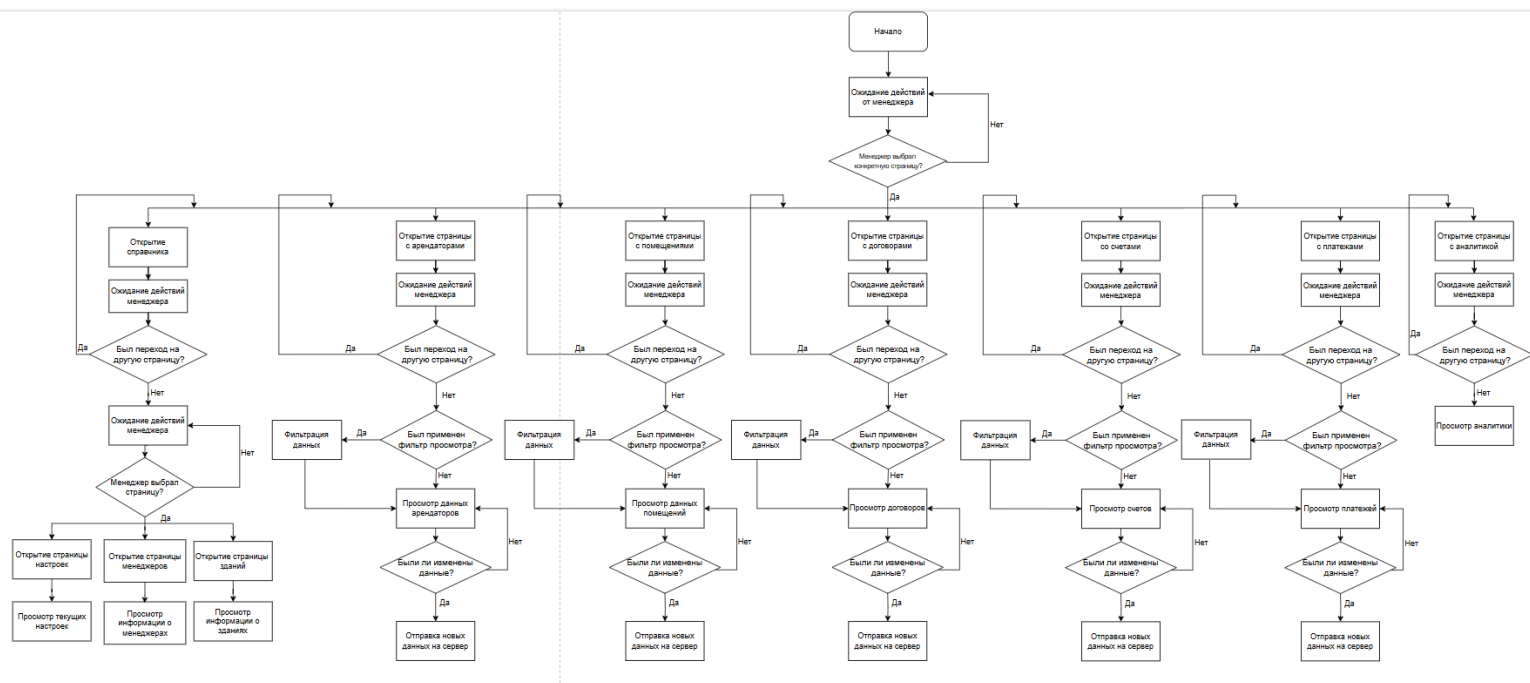


Рис. 6.2 Структурная схема работы менеджера в системе

6.3.2 Используемые методы

- Использование методов СУБД PostgreSQL для обработки различных запросов.
- Использование фреймворка Django для написания полноценного веб-приложения.

6.3.3 Структура программы с описанием функций составных частей и связи между ними

Программа состоит из нескольких частей: клиентская часть, серверная часть и сама База данных. Все данные хранятся в Базе данных и достаются по запросам сервера, которому клиент отправляет ту или иную информацию.

Взаимодействие клиентской и серверной частей в Django осуществляется по принципу "запрос-ответ". Клиентская часть, которая представляет собой веб-браузер пользователя, отправляет HTTP-запрос на сервер, где запущено приложение Django.

Серверная часть, которая представляет собой набор Python-модулей и библиотек, обрабатывает полученный запрос. Для этого она выполняет следующие действия:

1. Анализирует URL-адрес запроса и определяет, какому представлению (view) соответствует этот адрес.
2. Вызывает соответствующее представление и передает ему необходимые данные, которые были получены из запроса (например, параметры в строке запроса или данные из формы).
3. Представление выполняет необходимые действия, такие как обращение к базе данных, вычисления, обработка файлов, и формирует ответ, который будет отправлен клиенту.
4. Серверная часть отправляет ответ клиенту, который отображает его в веб-браузере.

6.3.4 Связи программы с другими программами

Клиентское приложение взаимодействует с несколькими внешними программными компонентами и сервисами в процессе работы. Во-первых, это СУБД PostgreSQL, которая позволяет выполнять запросы, а также читать и записывать данные. Во-вторых, это библиотеки и фреймворки Django, благодаря которым система взаимодействует с пользователем.

6.4 Используемые технические средства

Программное решение может эксплуатироваться на любом персональном компьютере (ПК) или смартфоне.

6.5 Вызов и загрузка

Вызов и загрузка программы осуществляется посредством перехода в браузере на соответствующую веб-страницу. Также необходимо, чтобы сервер был включен, иначе приложение не будет работать.

7. Руководство пользователя

После подключения к интернету и переходу к веб-странице, приложение становится доступным для использования.

1. Необходимо пройти авторизацию. Для этого пользователю необходимо указать свой логин и пароль в соответствующих полях и нажать кнопку «Войти».


Авторизация

Логин	<input type="text" value="Введите логин"/>
Пароль	<input type="password" value="Введите пароль"/>
<input type="button" value="Войти"/>	

Рис. 7.1 Авторизация

2. После авторизации пользователю открывается его карточка сотрудника. Менеджер может редактировать только свое фото, т.к. остальные данные сотрудников должны регулироваться начальником отдела.

Профиль



Логин:	sddanilaeva
Фамилия:	Данилаева
Имя:	Софья
Отчество:	Дмитриевна

Загрузить новое фото

Выбор файла

Не выбран ни один файл

Отправить фото

Рис. 7.2 Карточка сотрудника

3. Так же пользователю становятся доступны окна, для работы в системе.

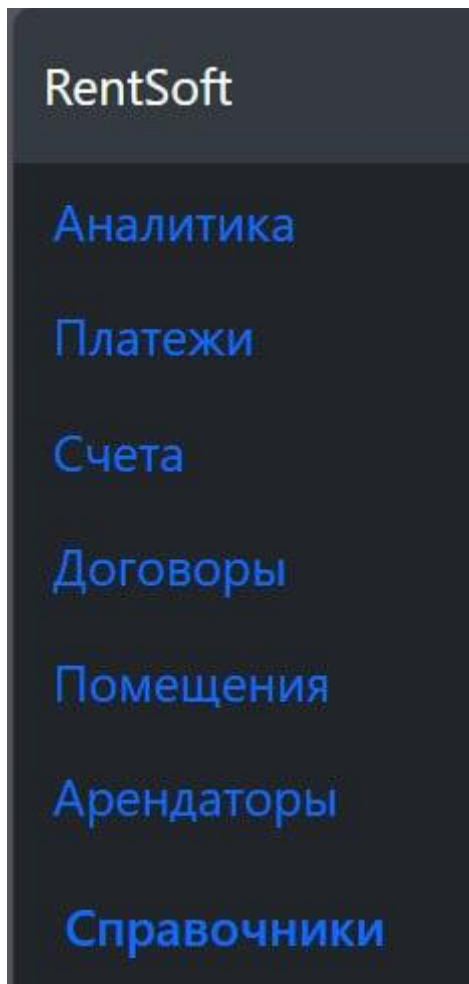


Рис. 7.3 Доступные окна

4. К карточке сотрудника можно перейти, нажав кнопку «Личный кабинет». Рядом находится кнопка «Выход» для выхода из учетной записи.

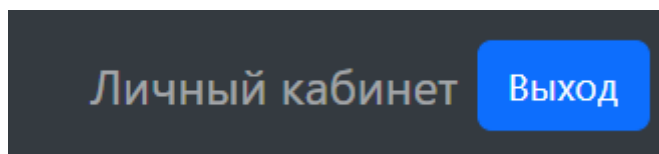


Рис. 7.4

5. В разделе счета менеджер ежемесячно создает счет на обслуживание конкретного помещения. Для этого необходимо нажать кнопку «Создать счет» и заполнить все поля в новом окне.

Так же для пользователя доступна фильтрация поиска. Для этого необходимо заполнить нужные ячейки и нажать кнопку воронки, для установления маски.

Счета							
Создать счет							
Помещение	Счет за свет	Счет за газ	Счет за воду	Счет за отопление	Счет за ремонтные работы	Дата создания счета	Действия
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	ДД.ММ.ГГГГ	
Птичий рынок - 13	123.00 Р	321.00 Р	454.00 Р	6579.00 Р	0.00 Р	29.05.24	

Рис. 7.5 Окно Счета

Добавление счета

Арендное помещение

Птичий рынок - 11

▼

Птичий рынок - 11

Птичий рынок - 12

Птичий рынок - 13

Птичий рынок - 14

Счет за газ

Счет за воду

Счет за отопление

Счет за ремонт

Дата создания

ДД.ММ.ГГГГ

Сохранить

Рис. 7.6 Создание нового счета

6. Окно договоров аренды выглядит и устроено аналогично.

При добавлении договора необходимо загрузить pdf файл, с копией договора.

Если договор заключается с новым арендатором, его можно добавить, не выходя из окна создания договора с помощью кнопки «Добавить арендатора».

Договоры аренды									
Заклучить договор аренды									
№ док.	Арендатор	Помещение	Дата заключения	Дата завершения	Оплачено до	Действующие	Период	Тариф за месяц	Действия
			ДД.ММ.ГГГГ	ДД.ММ.ГГГГ	ДД.ММ.ГГГГ				
№14	ООО "Клубничное поле"	11	01.05.24	01.08.24		да	01.05.24 - 01.08.24	140000.00 Р	
№15	АО "ПРАП"	12	31.05.24	01.07.24		да	31.05.24 - 01.07.24	80000.00 Р	
№16	ООО "ГОРМОН"	13	29.05.24	29.09.24	28.06.24	да	29.05.24 - 29.09.24	76000.00 Р	
№17	ОАО "МАНДАРИН"	14	29.05.24	29.06.24		да	29.05.24 - 29.06.24	200000.00 Р	

Рис. 7.7 Окно Договоры Аренды

Заклучение договора аренды	
<div>Арендатор</div> <div>ООО "Клубничное поле" </div> <div> Добавить арендатора</div>	<div>Документы</div> <div> Договор PDF Выгрузка файлов и документов</div>
<div>Арендное помещение</div> <div>Птичий рынок -11 </div>	
<div>Дата заключения соглашения</div> <div>ДД.ММ.ГГГГ </div>	
<div>Дата завершения соглашения</div> <div>ДД.ММ.ГГГГ </div>	
<div>Период аренды (дней)</div> <div></div>	
<div>Сумма ежемесячных платежей</div> <div></div>	
<div>Дата списания ежемесячного арендного платежа</div> <div>ДД.ММ.ГГГГ </div>	
<div>Сохранить</div>	

Рис. 7.8. Заключение договора аренды

7. Окно арендных помещений выглядит и устроено аналогично предыдущим.

Добавлять новые помещения можно только в учетной записи начальника отдела.

Арендные помещения

Добавить помещение

Здание	Этаж	Номер помещения	Статус	Площадь	Оплата до	Текущий договор	Текущий клиент	Менеджер	Комментарий	Действия
			Все							
Птичий рынок	1	1	Свободно	250.00	-		-	Трифонов Сергей Алексеевич		
Птичий рынок	1	2	Свободно	200.00	-		-	Трифонов Сергей Алексеевич		
Птичий рынок	1	3	Свободно	300.00	-		-	Трифонов Сергей Алексеевич		
Птичий рынок	1	4	Свободно	270.00	-		-	Зяц Александр Сергеевич		

Арендаторы

Добавить арендатора

ID	Компания	Фамилия	Имя	Телефон	Менеджер	Договор	Помещение	Задолженность по арендным платежам	Задолженность по обслуживанию помещений	Действия
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>			
33	ООО "Клубничное поле"	Яковенко	Анастасия	+7 (999) 123-45-67		-		0.00	0.00	
34	АО "ПРАП"	Давлетьяров	Ильяр	+7 (495) 987-65-43		-		0.00	0.00	
35	ООО "ГОРМОН"	Романов	Кирилл	+7 (812) 333-77-77		-		0.00	0.00	
36	ОАО "МАНДАРИН"	Бисякова	Мария	+7 (903) 555-33-33		-		0.00	0.00	

Рис. 7.11 Окно Арендаторы

Добавление арендатора

Фамилия:

Имя:

Отчество:

Электронная почта:

Номер телефона:

Название организации:

ОГРН:

ИНН:

Серия паспорта:

Номер паспорта:

БИК банка:

Расчетный счет:

Сохранить

Рис. 7.12 Добавление арендатора

9. Окно справочника разделяется на 3 дополнительных окна.

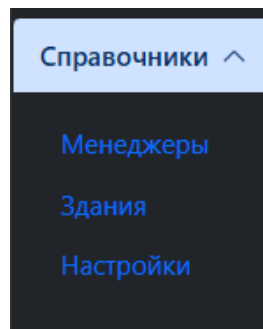


Рис. 7.13 Справочники

10. Окно менеджеров выглядит и устроено аналогично предыдущим.
Добавлять нового менеджера может только начальник отдела.







Менеджеры					Добавить менеджера
Фамилия	Имя	Отчество	Телефон	Действия	
Трифонов	Сергей	Алексеевич	79178956374	 	
Заяц	Александр	Сергеевич	999	 	
Данилаева	Софья	Дмитриевна	89179784565	 	

Рис. 7.14 Окно Менеджеры

Добавление менеджера

Логин:

Пароль:

Подтверждение пароля:

Фамилия:

Имя:

Отчество:

Номер телефона:

Фотография:

Выбор файла

Не выбран ни один файл

Сохранить

Рис. 7.15 Добавление Менеджера

11. Окно зданий выглядит и устроено аналогично предыдущим. Добавлять новое здание может только начальник отдела.

Здания

Добавить здание



Название	Адрес	Общая площадь	Общее количество помещений	Действия
Птичий рынок	г. Казань, ул. Белинского д. 18	4000.00	20	 

Рис. 7.16 Окно Здания

Добавить здание

Название:

Адрес:

Общая площадь:

Общее количество помещений:

Сохранить

Рис. 7.17 Добавление Здания

Заключение

В рамках курсовой работы был произведен анализ предметной области и выявлены основные требования к системе учета аренды коммерческой недвижимости. Были выделены основные пользователи и определены их сценарии использования программной системы.

Для более детального описания функционирования системы были разработаны SADT диаграммы и диаграмма потоков данных, описывающие процессы, протекающие в системе. При проектировании системы были разработаны ER диаграмма, детализирующая необходимые структуры данных,

и диаграмма классов предметной области, описывающая необходимые классы, методы и данные в разрабатываемой системе.

Для системы была выбрана клиент-серверная архитектура, с использованием фреймворка Django и СУБД PostgreSQL. Был разработан прототип системы, демонстрирующий часть функционала программной системы.

Последним этапом написания курсовой работы была разработка руководства пользователя системы учета аренды коммерческой недвижимости, подробно описывающего процесс использования разработанной системы.

Приложение 1

Файл account.html

```
{% extends 'base.html' %}

{% block content %}
<style>
    .bg-grey {
        background-color: #ffffff; /* Серый фон колонок */
        border-radius: 10px; /* Скругленные углы */
        padding: 20px; /* Отступ внутри колонок */
        margin-bottom: 20px; /* Отступ между колонками */
        margin-right: 50px; /* Отступ между колонками */
        margin-left: 10px; /* Отступ между колонками */
    }
    .row {
        display: flex;
        align-items: flex-start; /* Выравнивание по верхнему краю */
    }
</style>

<div class="container">
    <div class="row">
        <h3>Профиль</h3>
        <div class="col-md-5 bg-grey">
            <form method="post" enctype="multipart/form-data">
                {% csrf_token %}
                <div class="row">
                    <div class="col-4 text-center">
                        
```

```

</div>
<div class="col-8">
  <div class="row">
    <div class="col-4 font-weight-bold">Логин:</div>
    <div class="col-8">{{ user_info.username }}</div>
  </div>
  <div class="row">
    <div class="col-4 font-weight-bold">Фамилия:</div>
    <div class="col-8">{{ employee_info.surname }}</div>
  </div>
  <div class="row">
    <div class="col-4 font-weight-bold">Имя:</div>
    <div class="col-8">{{ employee_info.name }}</div>
  </div>
  <div class="row">
    <div class="col-4 font-weight-bold">Отчество:</div>
    <div class="col-8">{{ employee_info.middlename }}</div>
  </div>
</div>
</div>
<label for="formFile" class="form-label mt-3 mb-2">Загрузить новое фото</label>
<input class="form-control mb-3" name="user_photo_upload" type="file" id="formFile"
accept=".jpg,.jpeg,.png">
<button type="submit" class="btn btn-primary btn-sm">Отправить фото</button>
</form>
</div>
</div>
</div>
{% endblock %}

```

Файл add_building.html

```

{% extends 'base.html' %}

{% block content %}
<style>
  .bg-grey {
    background-color: #ffffff;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 20px;
  }

  .row {
    display: flex;
    align-items: flex-start;
  }

  .errorlist {
    display: none;
    color: red;
  }
</style>

```

```

<div class="container">
  <h2 class="my-4">Добавить здание</h2>
  <div class="row">
    <div class="col-md-6 bg-grey">
      <form method="post" class="w-100 mx-auto" style="max-width: 600px;">
        {% csrf_token %}
        <div class="mb-3">
          <label for="name">Название:</label>
          <input type="text" id="name" name="name" class="form-control" required>
        </div>
        <div class="mb-3">
          <label for="address">Адрес:</label>
          <input type="text" id="address" name="address" class="form-control" required>
        </div>
        <div class="mb-3">
          <label for="total_area">Общая площадь:</label>
          <input type="number" step="0.01" id="total_area" name="total_area" class="form-control"
required>
        </div>
        <div class="mb-3">
          <label for="total_rooms">Общее количество помещений:</label>
          <input type="number" id="total_rooms" name="total_rooms" class="form-control"
required>
        </div>
        <button type="submit" class="btn btn-success btn-block mt-3">Сохранить</button>
      </form>
    </div>
  </div>
</div>
{% endblock %}

```

Файл add_manager.html

```

{% extends 'base.html' %}

{% block content %}
<style>
  .bg-grey {
    background-color: #ffffff;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 20px;
  }

  .row {
    display: flex;
    align-items: flex-start;
  }

  .errorlist {
    display: none;
    color: red;
  }
</style>

<div class="container">
  <h2 class="my-4">Добавление менеджера</h2>
  <div class="row">
    <div class="col-md-6 bg-grey">

```

```

<form method="post" enctype="multipart/form-data">
  {% csrf_token %}
  <div class="mb-3">
    {{ user_form.username.label_tag }}
    {{ user_form.username }}
  </div>
  <div class="mb-3">
    {{ user_form.password1.label_tag }}
    {{ user_form.password1 }}
  </div>
  <div class="mb-3">
    {{ user_form.password2.label_tag }}
    {{ user_form.password2 }}
  </div>
  <div class="mb-3">
    {{ employee_form.surname.label_tag }}
    {{ employee_form.surname }}
  </div>
  <div class="mb-3">
    {{ employee_form.name.label_tag }}
    {{ employee_form.name }}
  </div>
  <div class="mb-3">
    {{ employee_form.middlename.label_tag }}
    {{ employee_form.middlename }}
  </div>
  <div class="mb-3">
    {{ employee_form.phone_number.label_tag }}
    {{ employee_form.phone_number }}
  </div>
  <div class="mb-3">
    {{ employee_form.photo.label_tag }}
    {{ employee_form.photo }}
  </div>
  <button type="submit" class="btn btn-success">Сохранить</button>
</form>
</div>
</div>
</div>

<script>
  // Скрыть сообщения об ошибках по умолчанию
  document.addEventListener("DOMContentLoaded", function() {
    var passwordErrors = document.querySelectorAll(".errorlist");
    passwordErrors.forEach(function(error) {
      error.style.display = 'none';
    });

    // Показать сообщения об ошибках при неправильном вводе пароля
    var password1 = document.getElementById("id_password1");
    var password2 = document.getElementById("id_password2");

    password1.addEventListener("input", function() {
      passwordErrors.forEach(function(error) {
        error.style.display = 'none';
      });
    });
  });

```



```

        password2.addEventListener("input", function() {
            passwordErrors.forEach(function(error) {
                error.style.display = 'none';
            });
        });
    });
</script>
{% endblock %}

```

Файл add_rental_space.html

```

{% extends 'base.html' %}

{% block content %}
<style>
    .bg-grey {
        background-color: #ffffff;
        border-radius: 10px;
        padding: 20px;
        margin-bottom: 20px;
    }

    .row {
        display: flex;
        align-items: flex-start;
    }

    .errorlist {
        display: none;
        color: red;
    }
</style>

<div class="container">
    <h2 class="my-4">Добавить помещение</h2>
    <div class="row">
        <div class="col-md-6 bg-grey">
            <form method="post" class="w-100 mx-auto" style="max-width: 600px;">
                {% csrf_token %}
                <div class="mb-3">
                    {{ form.building.label_tag }}
                    {{ form.building }}
                </div>
                <div class="mb-3">
                    {{ form.floor.label_tag }}
                    {{ form.floor }}
                </div>
                <div class="mb-3">
                    {{ form.number.label_tag }}
                    {{ form.number }}
                </div>
                <div class="mb-3">
                    {{ form.status.label_tag }}
                    {{ form.status }}
                </div>
                <div class="mb-3">
                    {{ form.area.label_tag }}
                    {{ form.area }}
                </div>
            </form>
        </div>
    </div>
</div>

```

```

    </div>
    <div class="mb-3">
      {{ form.curator.label_tag }}
      {{ form.curator }}
    </div>
    <div class="mb-3">
      {{ form.description.label_tag }}
      {{ form.description }}
    </div>
    <button type="submit" class="btn btn-success btn-block mt-3">Сохранить</button>
  </form>
</div>
</div>
</div>
{% endblock %}

```

Файл analytics.html

```

{% extends 'base.html' %}

{% block content %}
<div class="container-fluid">
  <h3>Аналитика</h3>
  <div class="row">
    <h5>Поступление платежей по месяцам</h5>
    <!-- Поступление платежей по месяцам -->
    <div class="col-sm-6">
      <canvas id="paymentsChart"></canvas>
    </div>

    <h5>Заполненность по количеству</h5>
    <!-- Круговая диаграмма состояний объектов недвижимости -->
    <div class="col-sm-6">
      <canvas id="statusPieChart"></canvas>
    </div>
  </div>
</div>
</div>
<script>
  // Пример данных для диаграммы состояний объектов недвижимости
  const statusData = {
    labels: ['Этаж-1', 'Этаж-2', 'Этаж-3'],
    datasets: [
      {
        label: 'Свободно',
        data: [3, 2, 1],
        backgroundColor: 'red',
        borderWidth: 1
      },
      {
        label: 'Занято',
        data: [2, 3, 4],
        backgroundColor: 'green',
        borderWidth: 1
      },
      {
        label: 'Ремонт',
        data: [1, 2, 3],
        backgroundColor: 'gray',
        borderWidth: 1
      }
    ]
  }
</script>

```

```
    }  
  ]  
};
```

```
// Настройки для линейной диаграммы состояний объектов недвижимости
```

```
const statusConfig = {  
  type: 'bar',  
  data: statusData,  
  options: {  
    scales: {  
      x: {  
        beginAtZero: true  
      },  
      y: {  
        beginAtZero: true  
      }  
    }  
  }  
};
```

```
// Рендеринг диаграммы состояний объектов недвижимости
```

```
const statusChart = new Chart(  
  document.getElementById('statusChart'),  
  statusConfig  
);
```

```
// Пример данных для поступления платежей по месяцам
```

```
const paymentsData = {  
  labels: ['Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь',  
'Октябрь', 'Ноябрь', 'Декабрь'],  
  datasets: [{  
    label: 'Сумма платежей',  
    data: [0, 0, 0, 170000, 147777, 280000, 0, 0, 0, 0, 0, 0],  
    backgroundColor: 'blue',  
    borderWidth: 1  
  }]  
};
```

```
// Настройки для гистограммы поступления платежей по месяцам
```

```
const paymentsConfig = {  
  type: 'bar',  
  data: paymentsData,  
  options: {  
    scales: {  
      x: {  
        beginAtZero: true  
      },  
      y: {  
        beginAtZero: true  
      }  
    }  
  }  
};
```

```
// Рендеринг гистограммы поступления платежей по месяцам
```

```
const paymentsChart = new Chart(  
  document.getElementById('paymentsChart'),
```

```

    paymentsConfig
  );

  // Пример данных для круговой диаграммы состояний объектов недвижимости
  const statusPieData = {
    labels: ['Свободно', 'Занято', 'Ремонт'],
    datasets: [{
      data: [1, 5, 1],
      backgroundColor: ['red', 'green', 'gray'],
      hoverOffset: 4
    }]
  };

  // Настройки для круговой диаграммы состояний объектов недвижимости
  const statusPieConfig = {
    type: 'pie',
    data: statusPieData,
  };

  // Рендеринг круговой диаграммы состояний объектов недвижимости
  const statusPieChart = new Chart(
    document.getElementById('statusPieChart'),
    statusPieConfig
  );
</script>
{% endblock %}

```

Файл auth.html

```

{% extends 'baseauthreg.html' %}
{% block content %}
<div class="h-100 d-flex align-items-center justify-content-center">
  <div class="row row-cols-1">
    <div class="col">
      <h1>Авторизация</h1>
    </div>
    <form method="post" action="">
      <div class="col">
        <div class="mb-3 row">
          <label for="staticEmail" class="col-sm-2 col-form-label">Логин</label>
          <div class="col-sm-10">
            <input type="text" class="form-control" name = "authloginame" id="authloginid"
placeholder="Введите логин">
          </div>
        </div>
        <div class="mb-3 row">
          <label for="staticEmail" class="col-sm-2 col-form-label">Пароль</label>
          <div class="col-sm-10">
            <input type="password" class="form-control" name = "authpasswordname"
id="authpasswordid" placeholder="Введите пароль">
          </div>
        </div>
        {% csrf_token %}
        <div class="mb-3 row">
          <button type="submit" class="btn btn-primary mb-3">Войти</button>
        </div>
      </form>
    </div>
  </div>

```

```
</div>
{% endblock %}
```

Файл base.html

```
{% load static %}
```

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>RentSoft</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
crossorigin="anonymous">
  <link rel="stylesheet" href="{% static 'css/style.css' %}">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-
icons.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #343a40;">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">RentSoft</a>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item">
              <a class="nav-link" href="{% url 'account' %}">Личный кабинет</a>
            </li>
            <li class="nav-item align-self-center">
              <a class="btn btn-primary" href="{% url 'logout' %}">Выход</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
  </header>

  <div class="d-flex" style="min-height: 100%;">
    <nav class="bg-dark sidebar" style="width: 200px; height: auto;">
      <div class="sidebar-sticky">
        <ul class="nav flex-column">
          <li class="nav-item">
            <a class="nav-link" href="{% url 'analytics' %}">Аналитика</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'payments' %}">Платежи</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'bills' %}">Счета</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'contracts' %}">Договоры</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'rental_objects' %}">Помещения</a>
          </li>
        </ul>
      </div>
    </nav>
  </div>
</body>
</html>
```

```

<li class="nav-item">
  <a class="nav-link" href="{% url 'tenant_list' %}">Арендаторы</a>
</li>
{% comment %} <li class="nav-item">
  <a class="nav-link" href="#">Рассылки</a>
</li> {% endcomment %}

{% if user.is_superuser %}
<li class="nav-item accordion">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed nav-link" type="button" data-bs-toggle="collapse"
data-bs-target="#collapseOne" aria-expanded="false" aria-controls="collapseOne">
        Справочники
      </button>
    </h2>
    <div id="collapseOne" class="accordion-collapse collapse" aria-labelledby="headingOne" data-
bs-parent="#accordionExample">
      <div class="accordion-body">
        <ul class="nav flex-column">
          <li class="nav-item">
            <a class="nav-link" href="{% url 'managers' %}">Менеджеры</a>
            <a class="nav-link" href="{% url 'buildings' %}">Здания</a>
            {% comment %} <a class="nav-link" href="{% url 'settings' %}">Настройки</a> {%
endcomment %}
          </li>
        </ul>
      </div>
    </div>
  </li>
{% endif %}
</ul>
</div>
</div>
</nav>
<main class="flex-fill p-5">
  {% block content %}
  {% endblock %}
</main>
</div>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"
integrity="sha384-..." crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</body>
</html>

```

Файл bill_confirm_delete.html

```

{% extends 'base.html' %}

{% block content %}
<div class="container">
  <h3>Вы уверены, что хотите удалить этот счет?</h3>
  <form method="post">
    {% csrf_token %}

```

```

        <button type="submit" class="btn btn-secondary">Удалить</button>
        <a href="{% url 'bills' %}" class="btn btn-danger">Отмена</a>
    </form>
</div>
{% endblock %}

```

Файл bill_form.html

```

{% extends 'base.html' %}

{% block content %}
<style>
    .bg-grey {
        background-color: #ffffff;
        border-radius: 10px;
        padding: 20px;
        margin-bottom: 20px;
    }
    .row {
        display: flex;
        align-items: flex-start;
    }
    .errorlist {
        display: none;
        color: red;
    }
</style>

<div class="container mt-5">
    <h2>
        {% if form.instance.pk %}
            Редактирование счета
        {% else %}
            Добавление счета
        {% endif %}
    </h2>
    <div class="row">
        <div class="col-md-6 bg-grey">
            <form method="post">
                {% csrf_token %}
                <div class="mb-3">
                    <label for="id_rental_space" class="form-label">Арендное помещение</label>
                    <select id="id_rental_space" name="rental_space" class="form-select">
                        {% for space in form.fields.rental_space.queryset %}
                            <option value="{{ space.pk }}">{{ space.building.name }} - {{ space.floor }} {{
space.number }}</option>
                        {% endfor %}
                    </select>
                </div>
                <div class="mb-3">
                    <label for="id_electricity_bill" class="form-label">Счет за электричество</label>
                    {{ form.electricity_bill }}
                </div>
                <div class="mb-3">
                    <label for="id_gas_bill" class="form-label">Счет за газ</label>
                    {{ form.gas_bill }}
                </div>
                <div class="mb-3">
                    <label for="id_water_bill" class="form-label">Счет за воду</label>

```

```

        {{ form.water_bill }}
    </div>
    <div class="mb-3">
        <label for="id_heating_bill" class="form-label">Счет за отопление</label>
        {{ form.heating_bill }}
    </div>
    <div class="mb-3">
        <label for="id_repair_bill" class="form-label">Счет за ремонт</label>
        {{ form.repair_bill }}
    </div>
    <div class="mb-3">
        <label for="id_creation_date" class="form-label">Дата создания</label>
        {{ form.creation_date }}
    </div>
    <button type="submit" class="btn btn-success">Сохранить</button>
</form>
</div>
</div>
</div>
{% endblock %}

```

Файл bills.html

```

{% extends 'base.html' %}
{% block content %}

<div class="container-fluid">
    <h2>Счета</h2>
    <div class="row">
        <div class="col d-flex justify-content-end">
            <a href="{% url 'add_bill' %}" class="btn btn-success mb-3">Создать счет</a>
        </div>
    </div>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Помещение</th>
                <th scope="col">Счет за свет</th>
                <th scope="col">Счет за газ</th>
                <th scope="col">Счет за воду</th>
                <th scope="col">Счет за отопление</th>
                <th scope="col">Счет за ремонтные работы</th>
                <th scope="col">Дата создания счета</th>
                <th scope="col">Действия</th>
            </tr>
            <tr>
                <form method="get" class="mb-3">
                    <th><input type="text" name="rental_space" class="form-control" value="{ {
request.GET.rental_space } }"></th>
                    <th></th>
                    <th></th>
                    <th></th>
                    <th></th>
                    <th></th>
                    <th><input type="date" id="creation_date" name="creation_date" class="form-control"
value="{ { request.GET.creation_date } }"></th>
                    <th>
                        <button type="submit" class="btn btn-primary btn-sm">

```



```

        <a href="{% url 'add_building' %}" class="btn btn-success mb-3">Добавить здание</a>
    </div>
</div>
<table class="table">
    <thead>
        <tr>
            <th scope="col">Название</th>
            <th scope="col">Адрес</th>
            <th scope="col">Общая площадь</th>
            <th scope="col">Общее количество помещений</th>
            <th scope="col">Действия</th>
        </tr>
    </thead>
    <tbody>
        {% for building in buildings %}
        <tr>
            <td>{{ building.name }}</td>
            <td>{{ building.address }}</td>
            <td>{{ building.total_area }}</td>
            <td>{{ building.total_rooms }}</td>
            <td>
                <a href="{% url 'edit_building' building.building_id %}" class="btn btn-primary">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-pencil-fill" viewBox="0 0 16 16">
                        <path d="M12.854 14.646 5.5 0 0-.707 0L10.5 1.793 14.207 5.511 6.47-1.646 5.5 0 0 0-.708 1-
3-3zm.646 6.061L9.793 2.5 3.293 9H3.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1
.5.5v.5h.5a.5.5 0 0 1 .5.5v.207l6.5-6.5zm-7.468 7.468A.5.5 0 0 1 6 13.5V13h-.5a.5.5 0 0 1-.5-.5V12h-
.5a.5.5 0 0 1-.5-.5V11h-.5a.5.5 0 0 1-.5-.5V10h-.5a.499.499 0 0 1-.175-.032l-.179.178a.5.5 0 0 0-
.11 1.168l-2 5a.5.5 0 0 0 .65.65l5-2a.5.5 0 0 0 .168-.11l.178-.178z"/>
                    </svg>
                </a>
                <a href="{% url 'delete_building' building.building_id %}" class="btn btn-danger">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-trash-fill" viewBox="0 0 16 16">
                        <path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 1 1H3v9a2 2 0 0 0 2 2h6a2 2 0 0 0 2 2V4h.5a1 1 0 0
0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1 1H2.5zm3 4a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 1 0v-
7a.5.5 0 0 1 .5-.5zm8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 1 0v-7a.5.5
0 0 1 1 1 0z"/>
                    </svg>
                </a>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
</div>
{% endblock %}

```

Файл contracts.html

```

{% extends "base.html" %}

{% block content %}
<div class="container-fluid">
    <h2>Договоры аренды</h2>
    <div class="row">
        <div class="col d-flex justify-content-end">
            <a href="{% url 'create_agreement' %}" class="btn btn-success mb-3">Заключить договор
аренды</a>

```

```

</div>
</div>
<table class="table ">
  <thead>
    <tr>
      <th>№ док.</th>
      <th>Арендатор</th>
      <th>Помещение</th>
      <th>Дата заключения</th>
      <th>Дата завершения</th>
      <th>Оплачено до</th>
      <th>Действующие</th>
      <th>Период</th>
      <th>Тариф за месяц</th>
      <th>Действия</th>
    </tr>
    <tr>
      <th>
        <input type="text" id="agreement_id" name="agreement_id" class="form-control"
value="{{ request.GET.agreement_id }}">
      </th>
      <th>
        <input type="text" id="organization_name" name="organization_name" class="form-
control" value="{{ request.GET.organization_name }}">
      </th>
      <th></th>
      <th>
        <input type="date" id="start_date" name="start_date" class="form-control" value="{{
request.GET.start_date }}">
      </th>
      <th>
        <input type="date" id="end_date" name="end_date" class="form-control" value="{{
request.GET.end_date }}">
      </th>
      <th>
        <input type="date" id="paid_until" name="paid_until" class="form-control" value="{{
request.GET.paid_until }}">
      </th>
      <th></th>
      <th></th>
      <th></th>
      <th><button type="submit" class="btn btn-primary btn-sm">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-funnel-fill" viewBox="0 0 16 16">
          <path d="M1.5 1.5A.5.5 0 0 1 2 1h12a.5.5 0 0 1 .5.5v2a.5.5 0 0 1-.128.334L10
8.692V13.5a.5.5 0 0 1-.342.474l-3 1A.5.5 0 0 1 6 14.5V8.692L1.628 3.834A.5.5 0 0 1 1.5 3.5v-2z"/>
        </svg>
      </button>
      </th>
    </tr>
  </thead>
  <tbody>
    {% for agreement in agreements %}

```


</style>

<div class="container mt-5">

<h2>Заключение договора аренды</h2>

<div class="row">

<div class="col-md-6 bg-grey">

<form method="post">

{% csrf_token %}

<div class="mb-3">

<label for="id_tenant" class="form-label">Арендатор</label>

<div class="input-group">

<select id="id_tenant" name="tenant" class="form-select">

{% for tenant in form.fields.tenant.queryset %}

<option value="{{ tenant.pk }}" {% if tenant.pk == request.GET.tenant_id

%}selected{% endif %}>

{{ tenant.organization_name }}

</option>

{% endfor %}

</select>

Добавить арендатора

</div>

</div>

<div class="mb-3">

<label for="id_rental_space" class="form-label">Арендное помещение</label>

<select id="id_rental_space" name="rental_space" class="form-select">

{% for space in form.fields.rental_space.queryset %}

<option value="{{ space.pk }}">{{ space.building.name }} - {{ space.floor }}. {{

space.number }}

{% endfor %}

</select>

</div>

<div class="mb-3">

<label for="id_start_date" class="form-label">Дата заключения соглашения</label>

{{ form.start_date }}

</div>

<div class="mb-3">

<label for="id_end_date" class="form-label">Дата завершения соглашения</label>

{{ form.end_date }}

</div>

<div class="mb-3">

<label for="id_period" class="form-label">Период аренды (дней)</label>

<input type="text" id="id_period" class="form-control" readonly>

</div>

<div class="mb-3">

<label for="id_monthly_payment" class="form-label">Сумма ежемесячных платежей</label>

{{ form.monthly_payment }}

</div>

<div class="mb-3">

<label for="id_payment_date" class="form-label">День списания ежемесячного арендного платежа</label>

{{ form.payment_date }}

</div>

<button type="submit" class="btn btn-success">Сохранить</button>

</form>

</div>

```

<div class="col-md-5 bg-grey">
  <h3>Документы</h3>
  <button type="button" class="btn btn-primary">Договор PDF</button>
  <button type="button" class="btn btn-primary">Выгрузка файлов и документов</button>
</div>
</div>
</div>

<!-- Modal для добавления арендатора -->
<div class="modal fade" id="addTenantModal" tabindex="-1" aria-labelledby="addTenantModalLabel"
aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="addTenantModalLabel">Добавить арендатора</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        <iframe id="addTenantFrame" src="{% url 'add_tenant_modal' %}" style="width: 100%;
height: 400px; border: none;"></iframe>
      </div>
    </div>
  </div>
</div>

</div>
</div>

<script>
document.addEventListener("DOMContentLoaded", function() {
  function calculatePeriod() {
    var startDate = document.getElementById("id_start_date").value;
    var endDate = document.getElementById("id_end_date").value;
    if (startDate && endDate) {
      var start = new Date(startDate);
      var end = new Date(endDate);
      var timeDiff = end - start;
      var dayDiff = timeDiff / (1000 * 3600 * 24);
      document.getElementById("id_period").value = dayDiff > 0 ? dayDiff : 0;
    }
  }

  document.getElementById("id_start_date").addEventListener("change", calculatePeriod);
  document.getElementById("id_end_date").addEventListener("change", calculatePeriod);

  // Initial calculation
  calculatePeriod();
});
</script>
{% endblock %}

Файл create_payment.html

{% extends 'base.html' %}

{% block content %}
<style>
  .bg-grey {

```

```

        background-color: #ffffff;
        border-radius: 10px;
        padding: 20px;
        margin-bottom: 20px;
    }
    .row {
        display: flex;
        align-items: flex-start;
    }
    .errorlist {
        display: none;
        color: red;
    }
}
</style>

<div class="container mt-5">
    <h2>Добавление платежа</h2>
    <div class="row">
        <div class="col-md-6 bg-grey">
            <form method="post">
                {% csrf_token %}
                <div class="mb-3">
                    <label for="id_rental_agreement" class="form-label">Арендное соглашение</label>
                    <select id="id_rental_agreement" name="rental_agreement" class="form-select">
                        {% for agreement in form.fields.rental_agreement.queryset %}
                            <option value="{{ agreement.pk }}">№ {{ agreement.agreement_id }} - {{
agreement.rental_space.building.name }} - {{ agreement.rental_space.floor }} {{
agreement.rental_space.number }}</option>
                        {% endfor %}
                    </select>
                </div>
                <div class="mb-3">
                    <label for="id_amount" class="form-label">Сумма платежа</label>
                    {{ form.amount }}
                </div>
                <div class="mb-3">
                    <label for="id_payment_date" class="form-label">Дата платежа</label>
                    {{ form.payment_date }}
                </div>
                <div class="mb-3">
                    <label for="id_payment_file" class="form-label">Копия платежа</label>
                    {{ form.payment_file }}
                </div>
                <div class="mb-3">
                    <label for="id_category" class="form-label">Категория платежа</label>
                    <select id="id_category" name="category" class="form-select">
                        {% for category in form.fields.category.queryset %}
                            <option value="{{ category.pk }}">{{ category }}</option>
                        {% endfor %}
                    </select>
                </div>
                <button type="submit" class="btn btn-success">Сохранить</button>
            </form>
        </div>
    </div>
</div>
{% endblock %}

```

Файл forms.py

```
from django import forms
from .models import *
from django.contrib.auth.forms import UserCreationForm, UserChangeForm
```

```
class BuildingForm(forms.ModelForm):
```

```
    class Meta:
        model = Building
        fields = ('name', 'address', 'total_area', 'total_rooms')
```

```
class CustomUserCreationForm(UserCreationForm):
```

```
    password1 = forms.CharField(
        label="Пароль",
        strip=False,
        widget=forms.PasswordInput(attrs={'class': 'form-control'}),
    )
    password2 = forms.CharField(
        label="Подтверждение пароля",
        widget=forms.PasswordInput(attrs={'class': 'form-control'}),
        strip=False,
    )
```

```
    class Meta(UserCreationForm.Meta):
```

```
        model = User
        fields = ('username',)
        labels = {
            'username': 'Логин',
        }
        widgets = {
            'username': forms.TextInput(attrs={'class': 'form-control'}),
        }
```

```
class CustomClearableFileInput(forms.ClearableFileInput):
```

```
    template_name = 'custom_clearable_file_input.html'
```

```
class EmployeeForm(forms.ModelForm):
```

```
    class Meta:
        model = Employee
        fields = ('surname', 'name', 'middlename', 'phone_number', 'photo')
        labels = {
            'surname': 'Фамилия',
            'name': 'Имя',
            'middlename': 'Отчество',
            'phone_number': 'Номер телефона',
            'photo': 'Фотография',
        }
        widgets = {
            'surname': forms.TextInput(attrs={'class': 'form-control'}),
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'middlename': forms.TextInput(attrs={'class': 'form-control'}),
            'phone_number': forms.TextInput(attrs={'class': 'form-control'}),
            'photo': forms.ClearableFileInput(attrs={'class': 'form-control'}),
        }
```

```
class CustomUserChangeForm(UserChangeForm):
```

```
    password1 = forms.CharField(
        label="Новый пароль",
```



```

strip=False,
widget=forms.PasswordInput(attrs={'class': 'form-control'}),
required=False,
)
password2 = forms.CharField(
    label="Подтверждение пароля",
    widget=forms.PasswordInput(attrs={'class': 'form-control'}),
    strip=False,
    required=False,
)

```

```

class Meta(UserChangeForm.Meta):
    model = User
    fields = ('username', 'password1', 'password2')
    labels = {
        'username': 'Логин',
    }
    widgets = {
        'username': forms.TextInput(attrs={'class': 'form-control'}),
    }

```

```

def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.fields['username'].initial = self.instance.username

```

```

class RentalSpaceForm(forms.ModelForm):
    class Meta:
        model = RentalSpace
        fields = ['building', 'floor', 'number', 'status', 'area', 'tenant', 'curator', 'description']
        labels = {
            'building': 'Здание',
            'floor': 'Этаж',
            'number': 'Номер помещения',
            'status': 'Статус',
            'area': 'Площадь',
            'tenant': 'Арендатор',
            'curator': 'Менеджер',
            'description': 'Комментарий',
        }
        widgets = {
            'building': forms.Select(attrs={'class': 'form-control'}),
            'floor': forms.NumberInput(attrs={'class': 'form-control'}),
            'number': forms.TextInput(attrs={'class': 'form-control'}),
            'status': forms.Select(attrs={'class': 'form-control'}),
            'area': forms.NumberInput(attrs={'class': 'form-control'}),
            'tenant': forms.Select(attrs={'class': 'form-control'}),
            'curator': forms.Select(attrs={'class': 'form-control'}),
            'description': forms.Textarea(attrs={'class': 'form-control'}),
        }

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['tenant'].required = False # Сделать поле tenant необязательным

```

```

class TenantForm(forms.ModelForm):
    class Meta:

```

```

model = Tenant
fields = [
    'surname',
    'name',
    'middlename',
    'phone_number',
    'email',
    'organization_name',
    'ogrn',
    'inn',
    'passport_series',
    'passport_number',
    'bank_bik',
    'bank_account',
]
labels = {
    'surname': 'Фамилия',
    'name': 'Имя',
    'middlename': 'Отчество',
    'phone_number': 'Номер телефона',
    'email': 'Электронная почта',
    'organization_name': 'Название организации',
    'ogrn': 'ОГРН',
    'inn': 'ИНН',
    'passport_series': 'Серия паспорта',
    'passport_number': 'Номер паспорта',
    'bank_bik': 'БИК банка',
    'bank_account': 'Расчетный счет',
}
widgets = {
    'surname': forms.TextInput(attrs={'class': 'form-control'}),
    'name': forms.TextInput(attrs={'class': 'form-control'}),
    'middlename': forms.TextInput(attrs={'class': 'form-control'}),
    'phone_number': forms.TextInput(attrs={'class': 'form-control'}),
    'email': forms.EmailInput(attrs={'class': 'form-control'}),
    'organization_name': forms.TextInput(attrs={'class': 'form-control'}),
    'ogrn': forms.TextInput(attrs={'class': 'form-control'}),
    'inn': forms.TextInput(attrs={'class': 'form-control'}),
    'passport_series': forms.TextInput(attrs={'class': 'form-control'}),
    'passport_number': forms.TextInput(attrs={'class': 'form-control'}),
    'bank_bik': forms.TextInput(attrs={'class': 'form-control'}),
    'bank_account': forms.TextInput(attrs={'class': 'form-control'}),
}

```

```

class TenantFilterForm(forms.Form):
    tenant_id = forms.CharField(max_length=10, required=False, label='ID')
    organization_name = forms.CharField(max_length=200, required=False, label='Компания')
    surname = forms.CharField(max_length=100, required=False, label='Фамилия')
    name = forms.CharField(max_length=100, required=False, label='Имя')
    phone_number = forms.CharField(max_length=20, required=False, label='Телефон')
    manager = forms.ModelChoiceField(queryset=User.objects.all(), required=False, label='Менеджер')
    rental_debt = forms.DecimalField(max_digits=15, decimal_places=2, required=False,
label='Задолженность по арендным платежам')
    maintenance_debt = forms.DecimalField(max_digits=15, decimal_places=2, required=False,
label='Задолженность по обслуживанию помещений')

```

```

class RentalAgreementForm(forms.ModelForm):
    class Meta:
        model = RentalAgreement
        fields = [
            'tenant', 'rental_space', 'start_date', 'end_date',
            'monthly_payment', 'payment_date'
        ]
        widgets = {
            'tenant': forms.Select(attrs={'class': 'form-select'}),
            'rental_space': forms.Select(attrs={'class': 'form-select'}),
            'start_date': forms.DateInput(attrs={'class': 'form-control', 'type': 'date'}),
            'end_date': forms.DateInput(attrs={'class': 'form-control', 'type': 'date'}),
            'monthly_payment': forms.NumberInput(attrs={'class': 'form-control'}),
            'payment_date': forms.DateInput(attrs={'class': 'form-control', 'type': 'date'}),
        }

```

```

class MaintenanceBillForm(forms.ModelForm):
    class Meta:
        model = MaintenanceBill
        fields = [
            'rental_space',
            'electricity_bill',
            'gas_bill',
            'water_bill',
            'heating_bill',
            'repair_bill',
            'creation_date'
        ]
        widgets = {
            'rental_space': forms.Select(attrs={'class': 'form-select'}),
            'electricity_bill': forms.NumberInput(attrs={'class': 'form-control'}),
            'gas_bill': forms.NumberInput(attrs={'class': 'form-control'}),
            'water_bill': forms.NumberInput(attrs={'class': 'form-control'}),
            'heating_bill': forms.NumberInput(attrs={'class': 'form-control'}),
            'repair_bill': forms.NumberInput(attrs={'class': 'form-control'}),
            'creation_date': forms.DateInput(attrs={'class': 'form-control', 'type': 'date'}),
        }

```

```

class PaymentForm(forms.ModelForm):
    class Meta:
        model = Payment
        fields = [
            'rental_agreement',
            'amount',
            'payment_date',
            'payment_file',
            'category'
        ]
        widgets = {
            'rental_agreement': forms.Select(attrs={'class': 'form-select'}),
            'amount': forms.NumberInput(attrs={'class': 'form-control'}),
            'payment_date': forms.DateInput(attrs={'class': 'form-control', 'type': 'date'}),
            'payment_file': forms.FileInput(attrs={'class': 'form-control'}),
            'category': forms.Select(attrs={'class': 'form-select'}),
        }

```

Файл models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.core.validators import MinValueValidator
```

СОТРУДНИК

```
class Employee(models.Model):
    employee_id = models.AutoField(primary_key=True)
    surname = models.CharField(max_length=100)
    name = models.CharField(max_length=100)
    middlename = models.CharField(max_length=100, blank=True, null=True)
    phone_number = models.CharField(max_length=20)
    photo = models.ImageField(upload_to='userphotos/%Y-%m-%d/',
                              default='images/photo_not_found.jpg', null=True, blank=True)

    def __str__(self):
        return f'{self.surname} {self.name} {self.middlename} {self.phone_number}'

    def get_photo_filename(self):
        return f'userphotos/{self.photo}' if self.photo else 'images/photo_not_found.jpg'

    def full_name(self):
        return f'{self.surname} {self.name} {self.middlename}'

    def fio(self):
        return f'{self.surname} {self.name[0]}.{self.middlename[0:1]}'
```

ПОЛЬЗОВАТЕЛЬ

```
class User(AbstractUser):
    user_id = models.AutoField(primary_key=True)
    employee = models.OneToOneField(Employee, on_delete=models.CASCADE)
    username = models.CharField(max_length=150, unique=True)
    password = models.CharField(max_length=150)

    def __str__(self):
        return f'{self.employee.full_name()}'
```

ЗДАНИЕ

```
class Building(models.Model):
    building_id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200, verbose_name='Название здания')
    address = models.CharField(max_length=200, verbose_name='Адрес здания')
    total_area = models.DecimalField(max_digits=10, decimal_places=2,
                                     validators=[MinValueValidator(0)], verbose_name='Общая площадь помещений')
    total_rooms = models.IntegerField(validators=[MinValueValidator(0)], verbose_name='Общее количество помещений')

    def __str__(self):
        return (f'{self.name} {self.address}')
```

АРЕНДНОЕ ПОМЕЩЕНИЕ

```
class RentalSpace(models.Model):
```

```

rental_space_id = models.AutoField(primary_key=True)
building = models.ForeignKey(Building, on_delete=models.CASCADE)
area = models.DecimalField(max_digits=10, decimal_places=2, validators=[MinValueValidator(0)],
verbose_name='Площадь помещения')
description = models.TextField(blank=True, null=True, verbose_name='Описание помещения')
floor = models.IntegerField(verbose_name='Этаж')
number = models.CharField(max_length=10, verbose_name='Номер помещения')
curator = models.ForeignKey(User, on_delete=models.CASCADE, related_name='curated_spaces',
verbose_name='Куратор помещения')
status = models.ForeignKey('SpaceStatus', on_delete=models.CASCADE)
tenant = models.ForeignKey('Tenant', on_delete=models.CASCADE, related_name='rented_spaces',
blank=True, null=True,)

```

```

def __str__(self):
    return (f'{self.rental_space_id} {self.building} {self.area} {self.floor} {self.number} '
            f'{self.curator} {self.status} {self.tenant}')

```

СТАТУС ПОМЕЩЕНИЯ

```

class SpaceStatus(models.Model):
    status_id = models.AutoField(primary_key=True)
    description = models.CharField(max_length=50, verbose_name='Статус помещения')

    def __str__(self):
        return self.description

```

АРЕНДАТОР

```

class Tenant(models.Model):
    tenant_id = models.AutoField(primary_key=True)
    surname = models.CharField(max_length=100, verbose_name='Фамилия арендатора')
    name = models.CharField(max_length=100, verbose_name='Имя арендатора')
    middlename = models.CharField(max_length=100, null=True, verbose_name='Отчество арендатора')
    email = models.EmailField(unique=True, verbose_name='Почта арендатора')
    phone_number = models.CharField(max_length=20, verbose_name='Номер телефона арендатора')
    organization_name = models.CharField(max_length=200, verbose_name='Наименование
организации')
    ogrn = models.CharField(max_length=13, verbose_name='ОГРН организации')
    inn = models.CharField(max_length=12, verbose_name='ИНН организации')
    passport_series = models.CharField(max_length=4, verbose_name='Серия паспорта арендатора')
    passport_number = models.CharField(max_length=6, verbose_name='Номер паспорта арендатора')
    bank_bik = models.CharField(max_length=9, verbose_name='БИК банка')
    bank_account = models.CharField(max_length=20, verbose_name='Расчетный счёт банка')
    total_payments = models.DecimalField(max_digits=15, decimal_places=2, blank=True, null=True,
verbose_name='Общая сумма выплат')
    rental_debt = models.DecimalField(max_digits=15, decimal_places=2, blank=True, null=True,
verbose_name='Текущая задолженность по арендным платежам')
    maintenance_debt = models.DecimalField(max_digits=15, decimal_places=2, blank=True, null=True,
verbose_name='Текущая задолженность по обслуживанию помещений')

```

```

def __str__(self):
    return (f'{self.surname} {self.name} {self.middlename} {self.email} '
            f'{self.phone_number} {self.organization_name} {self.ogrn} {self.inn} {self.bank_bik} '
            f'{self.total_payments} {self.rental_debt} {self.maintenance_debt}')

```

СЧЁТ ЗА ОБСЛУЖИВАНИЕ ПОМЕЩЕНИЯ

```

class MaintenanceBill(models.Model):
    bill_id = models.AutoField(primary_key=True)
    rental_space = models.ForeignKey(RentalSpace, on_delete=models.CASCADE)
    gas_bill = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Счет за газ')
    electricity_bill = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Счет за
электричество')
    water_bill = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Счет за воду')
    heating_bill = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Счет за
отопление')
    repair_bill = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Счет за
ремонтные работы')
    creation_date = models.DateField(verbose_name='Дата создания счёта')

    def __str__(self):
        return (f'{self.rental_space} {self.gas_bill} {self.electricity_bill} {self.water_bill} '
            f'{self.heating_bill} {self.repair_bill} {self.creation_date}')

```

АРЕНДНОЕ СОГЛАШЕНИЕ

```

class RentalAgreement(models.Model):
    agreement_id = models.AutoField(primary_key=True)
    tenant = models.ForeignKey(Tenant, on_delete=models.CASCADE)
    rental_space = models.ForeignKey(RentalSpace, on_delete=models.CASCADE)
    monthly_payment = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Сумма
ежемесячных выплат')
    start_date = models.DateField(verbose_name='Дата заключения соглашения')
    end_date = models.DateField(verbose_name='Дата завершения соглашения')
    payment_date = models.DateField(verbose_name='Дата списания ежемесячного арендного
платежа')

    def __str__(self):
        return (f'{self.tenant} {self.tenant.surname} {self.tenant.name} {self.tenant.middlename}'
            f'{self.rental_space.number} {self.monthly_payment} {self.start_date} {self.end_date}
{self.payment_date}')

```

ДОГОВОР

```

class Contract(models.Model):
    contract_id = models.AutoField(primary_key=True)
    rental_agreement = models.OneToOneField(RentalAgreement, on_delete=models.CASCADE)
    signing_date = models.DateField(verbose_name='Дата подписания договора')
    contract_file = models.FileField(upload_to='contracts/', blank=True, null=True, verbose_name='Копия
договора')
    status = models.CharField(max_length=50, verbose_name='Статус договора')

    def __str__(self):
        return f'{self.contract_id} {self.signing_date} {self.contract_file} {self.status}'

```

ПЛАТЕЖ

```

class Payment(models.Model):
    payment_id = models.AutoField(primary_key=True)
    rental_agreement = models.ForeignKey(RentalAgreement, on_delete=models.CASCADE)
    amount = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Сумма платежа')
    payment_date = models.DateField(verbose_name='Дата платежа')

```

```

    payment_file = models.FileField(upload_to='payments/', blank=True, null=True,
verbose_name='Копия платежа')
    category = models.ForeignKey('PaymentCategory', on_delete=models.CASCADE,
verbose_name='Категория договора')

    def __str__(self):
        return f'{self.payment_id} {self.payment_date} {self.amount} {self.category.category_name}'

# КАТЕГОРИЯ ПЛАТЕЖА
class PaymentCategory(models.Model):
    category_id = models.AutoField(primary_key=True)
    category_name = models.CharField(max_length=100, verbose_name='Название категории платежа')

    def __str__(self):
        return f'{self.category_name}'

```

Файл urls.py

```

from django.urls import path
from django.views.generic import TemplateView
from . import views
from .views import *

from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("", views.auth, name='auth'),

    path('analytics/', analytics_view, name='analytics'),

    path('payments/', views.payments, name='payments'),
    path('payments/create/', views.create_payment, name='create_payment'),
    path('payments/edit/<int:bill_id>', views.edit_payment, name='edit_payment'),
    path('payments/delete/<int:bill_id>', views.delete_payment, name='delete_payment'),

    path('account/logout', views.logout_, name='logout'),
    path('account', views.account, name='account'),

    path('rental/', views.rental_objects, name='rental_objects'),
    path('rental/add/', views.add_rental_space, name='add_rental_space'),
    path('rental/edit/<int:pk>', views.edit_rental_space, name='edit_rental_space'),
    path('rental/delete/<int:pk>', views.delete_rental_space, name='delete_rental_space'),

    path('contracts/', views.contracts, name='contracts'),
    path('contracts/delete/<int:pk>', views.delete_contract, name='delete_contract'),

    path('create_agreement/', views.create_agreement, name='create_agreement'),
    path('add_tenant_modal/', views.add_tenant_modal, name='add_tenant_modal'),

    path('settings/', views.settings, name='settings'),

    path('buildings/', buildings, name='buildings'),
    path('buildings/add/', add_building, name='add_building'),
    path('buildings/edit/<int:building_id>', edit_building, name='edit_building'),
    path('buildings/delete/<int:building_id>', delete_building, name='delete_building'),

```

```

path('managers/', views.managers, name='managers'),
path('add_manager/', views.add_manager, name='add_manager'),
path('edit_manager/<int:pk>/', views.edit_manager, name='edit_manager'),
path('delete_manager/<int:pk>/', views.delete_manager, name='delete_manager'),

```

```

path('tenants/', views.tenant_list, name='tenant_list'),
path('tenants/add/', views.tenant_add, name='tenant_add'),
path('tenants/edit/<int:tenant_id>/', views.tenant_edit, name='tenant_edit'),
path('tenants/delete/<int:tenant_id>/', views.tenant_delete, name='tenant_delete'),

```

```

path('bills/', views.bills_view, name='bills'),
path('bills/add/', views.add_bill_view, name='add_bill'),
path('bills/edit/<int:bill_id>/', views.edit_bill_view, name='edit_bill'),
path('bills/delete/<int:bill_id>/', views.delete_bill_view, name='delete_bill'),

```

```
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Файл views.py

```
import os
```

```

from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.core.exceptions import ValidationError
from django.http import HttpResponseRedirect, HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse
from .models import *
from rsapp.forms import *
from django.shortcuts import render
from django.utils import timezone
import pandas as pd
import matplotlib.pyplot as plt
import base64
import matplotlib.pyplot as plt
import io
from PIL import Image

```

```

from django.shortcuts import render
from .models import RentalSpace, Payment
from django.db.models import Count, Sum
import pandas as pd

```

```

@login_required
def payments(request):
    return render(request, 'payments.html', {'title': 'RentSoft | Оплата'})

```

```

def auth(request):
    if request.user.is_authenticated:
        return redirect('account')

    if request.method == 'POST':
        u_login = request.POST['authloginame']
        password = request.POST['authpasswordname']

```



```

        user = authenticate(request, username=u_login, password=password)
        if user and user.is_active:
            login(request, user)
            return HttpResponseRedirect(reverse('account'))

    return render(request, 'auth.html', {'title': 'RentSoft | Авторизация'})

@login_required
def account(request):
    user_info = request.user
    employee_info = user_info.employee

    if request.method == 'POST':
        user_photo_uploaded = request.FILES['user_photo_upload']
        allowed_extensions = ['.jpg', '.png']
        ext = os.path.splitext(user_photo_uploaded.name)[1]

        if ext.lower() not in allowed_extensions:
            raise ValidationError('Only JPG, and PNG files are allowed.')
        else:
            employee_info.photo = user_photo_uploaded
            employee_info.save()

    return render(request, 'account.html', {'user_info': user_info, 'employee_info': employee_info, 'title':
'RentSoft | Профиль'})

@login_required
def tenant_list(request):
    tenants = Tenant.objects.all()

    if request.method == 'GET':
        filter_form = TenantFilterForm(request.GET)
        if filter_form.is_valid():
            if filter_form.cleaned_data['tenant_id']:
                tenants = tenants.filter(tenant_id=filter_form.cleaned_data['tenant_id'])
            if filter_form.cleaned_data['organization_name']:
                tenants =
tenants.filter(organization_name__icontains=filter_form.cleaned_data['organization_name'])
            if filter_form.cleaned_data['surname']:
                tenants = tenants.filter(surname__icontains=filter_form.cleaned_data['surname'])
            if filter_form.cleaned_data['name']:
                tenants = tenants.filter(name__icontains=filter_form.cleaned_data['name'])
            if filter_form.cleaned_data['phone_number']:
                tenants = tenants.filter(phone_number__icontains=filter_form.cleaned_data['phone_number'])
            if filter_form.cleaned_data['manager']:
                tenants = tenants.filter(rented_spaces__curator=filter_form.cleaned_data['manager'])
            if filter_form.cleaned_data['rental_debt']:
                tenants = tenants.filter(rental_debt__gte=filter_form.cleaned_data['rental_debt'])
            if filter_form.cleaned_data['maintenance_debt']:
                tenants = tenants.filter(maintenance_debt__gte=filter_form.cleaned_data['maintenance_debt'])
        else:
            filter_form = TenantFilterForm()

    return render(request, 'tenant_list.html', {'tenants': tenants, 'filter_form': filter_form})

```

```

@login_required
def tenant_add(request):
    if request.method == 'POST':
        form = TenantForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('tenant_list')
    else:
        form = TenantForm()
    return render(request, 'tenant_form.html', {'form': form})

@login_required
def create_agreement(request):
    if request.method == 'POST':
        form = RentalAgreementForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('contracts') # Замените на ваш список договоров
    else:
        form = RentalAgreementForm()
    return render(request, 'create_agreement.html', {'form': form})

```

```

@login_required
def add_tenant_modal(request):
    if request.method == 'POST':
        form = TenantForm(request.POST)
        if form.is_valid():
            tenant = form.save()
            return redirect(f'{reverse("create_agreement")}?tenant_id={tenant.tenant_id}')
    else:
        form = TenantForm()

    return render(request, 'tenant_form_modal.html', {'form': form, 'is_modal': True})

```

```

@login_required
def tenant_edit(request, tenant_id):
    tenant = get_object_or_404(Tenant, tenant_id=tenant_id)
    if request.method == 'POST':
        form = TenantForm(request.POST, instance=tenant)
        if form.is_valid():
            form.save()
            return redirect('tenant_list')
    else:
        form = TenantForm(instance=tenant)
    return render(request, 'tenant_form.html', {'form': form})

```

```

@login_required
def tenant_delete(request, tenant_id):
    tenant = get_object_or_404(Tenant, tenant_id=tenant_id)
    if request.method == 'POST':
        tenant.delete()
        return redirect('tenant_list')
    return render(request, 'tenant_confirm_delete.html', {'tenant': tenant})

```

```

# Работа с помещениями
# -----
def rental_objects(request):
    rental_spaces = RentalSpace.objects.all()
    building_filter = request.GET.get('building')
    floor_filter = request.GET.get('floor')
    number_filter = request.GET.get('number')
    status_filter = request.GET.get('status')
    area_filter = request.GET.get('area')
    tenant_filter = request.GET.get('tenant')
    curator_filter = request.GET.get('curator')

    if building_filter:
        rental_spaces = rental_spaces.filter(building__name__icontains=building_filter)
    if floor_filter:
        rental_spaces = rental_spaces.filter(floor=floor_filter)
    if number_filter:
        rental_spaces = rental_spaces.filter(number__icontains=number_filter)
    if status_filter:
        try:
            status = SpaceStatus.objects.get(description=status_filter)
            rental_spaces = rental_spaces.filter(status=status.status_id)
        except SpaceStatus.DoesNotExist:
            rental_spaces = rental_spaces.none()
    if area_filter:
        rental_spaces = rental_spaces.filter(area=area_filter)
    if tenant_filter:
        rental_spaces = rental_spaces.filter(tenant__surname__icontains=tenant_filter)
    if curator_filter:
        rental_spaces = rental_spaces.filter(
            models.Q(curator__employee__surname__icontains=curator_filter) |
            models.Q(curator__employee__name__icontains=curator_filter) |
            models.Q(curator__employee__middlename__icontains=curator_filter)
        )

    return render(request, 'rental_objects.html', {'rental_spaces': rental_spaces})

@login_required
def add_rental_space(request):
    if request.method == 'POST':
        form = RentalSpaceForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('rental_objects')
    else:
        form = RentalSpaceForm()
    return render(request, 'add_rental_space.html', {'form': form})

@login_required
def edit_rental_space(request, pk):
    rental_space = get_object_or_404(RentalSpace, pk=pk)
    if request.method == 'POST':
        form = RentalSpaceForm(request.POST, instance=rental_space)
        if form.is_valid():
            form.save()
            return redirect('rental_objects')

```

```

else:
    form = RentalSpaceForm(instance=rental_space)
    return render(request, 'edit_rental_space.html', {'form': form})

```

```

@login_required
def delete_rental_space(request, pk):
    rental_space = get_object_or_404(RentalSpace, pk=pk)
    if request.method == 'POST':
        rental_space.delete()
        return redirect('rental_objects')
    return render(request, 'delete_rental_space.html', {'rental_space': rental_space})
# -----

```

```

@login_required
def contracts(request):
    agreements = RentalAgreement.objects.all()

    # Фильтрация данных по GET-параметрам
    doc_num_filter = request.GET.get('agreement_id')
    start_date_filter = request.GET.get('start_date')
    end_date_filter = request.GET.get('end_date')
    paid_until_filter = request.GET.get('paid_until')
    organization_name_filter = request.GET.get('organization_name')

    if doc_num_filter:
        agreements = agreements.filter(agreement_id=doc_num_filter)
    if start_date_filter:
        agreements = agreements.filter(start_date__gte=start_date_filter)
    if end_date_filter:
        agreements = agreements.filter(end_date__lte=end_date_filter)
    if paid_until_filter:
        paid_until_date = timezone.datetime.strptime(paid_until_filter, '%Y-%m-%d').date()
        agreements = agreements.filter(payment_date__lte=paid_until_date)
    if organization_name_filter:
        agreements = agreements.filter(tenant__organization_name__icontains=organization_name_filter)

    # Дополнительные поля
    for agreement in agreements:
        latest_payment = Payment.objects.filter(rental_agreement=agreement).order_by('-payment_date').first()
        agreement.paid_until = latest_payment.payment_date + timezone.timedelta(days=30) if latest_payment else None
        agreement.is_active = "да" if agreement.end_date >= timezone.now().date() else "нет"

    return render(request, 'contracts.html', {'agreements': agreements})

```

```

def delete_contract(request, pk):
    contract = get_object_or_404(Contract, pk=pk)

    if request.method == 'POST':
        contract.delete()
        return redirect('contracts')
    context = {
        'contract': contract
    }

```

```
return render(request, 'delete_contract.html', context)
```

```
@login_required
def settings(request):
    return render(request, 'settings.html', {'title': 'RentSoft | Настройки'})
```

```
@login_required
# Список зданий
def buildings(request):
    buildings = Building.objects.all()
    return render(request, 'buildings.html', {'buildings': buildings})
```

```
# Добавление здания
```

```
@login_required
def add_building(request):
    if request.method == 'POST':
        name = request.POST['name']
        address = request.POST['address']
        total_area = request.POST['total_area']
        total_rooms = request.POST['total_rooms']

        Building.objects.create(
            name=name,
            address=address,
            total_area=total_area,
            total_rooms=total_rooms
        )
        return redirect('buildings')

    return render(request, 'add_building.html')
```

```
# Изменение здания
```

```
@login_required
def edit_building(request, building_id):
    building = get_object_or_404(Building, pk=building_id)
    if request.method == 'POST':
        building.name = request.POST['name']
        building.address = request.POST['address']
        building.total_area = request.POST['total_area']
        building.total_rooms = request.POST['total_rooms']
        building.save()
        return redirect('buildings')

    return render(request, 'edit_building.html', {'building': building})
```

```
# Удаление здания
```

```
@login_required
def delete_building(request, building_id):
    building = get_object_or_404(Building, pk=building_id)
    if request.method == 'POST':
        building.delete()
        return redirect('buildings')

    return render(request, 'delete_building.html', {'building': building})
```

```

# Менеджеры
@login_required
def managers(request):
    managers = User.objects.all()
    return render(request, 'managers.html', {'managers': managers})

@login_required
def add_manager(request):
    if request.method == 'POST':
        employee_form = EmployeeForm(request.POST, request.FILES)
        user_form = CustomUserCreationForm(request.POST)

        if employee_form.is_valid() and user_form.is_valid():
            employee = employee_form.save()
            user = user_form.save(commit=False)
            user.employee = employee
            user.set_password(user_form.cleaned_data['password1'])
            user.save()
            return redirect('managers')

    else:
        employee_form = EmployeeForm()
        user_form = CustomUserCreationForm()

    return render(request, 'add_manager.html', {'employee_form': employee_form, 'user_form':
user_form})

@login_required
def edit_manager(request, pk):
    employee = get_object_or_404(Employee, pk=pk)
    user = get_object_or_404(User, employee=employee)

    if request.method == 'POST':
        employee_form = EmployeeForm(request.POST, request.FILES, instance=employee)
        user_form = CustomUserChangeForm(request.POST, instance=user)

        if employee_form.is_valid() and user_form.is_valid():
            employee_form.save()
            user_form.save()
            return redirect('managers')

    else:
        employee_form = EmployeeForm(instance=employee)
        user_form = CustomUserChangeForm(instance=user)

    return render(request, 'edit_manager.html', {
        'employee_form': employee_form,
        'user_form': user_form,
    })

@login_required
def delete_manager(request, pk):
    employee = get_object_or_404(Employee, pk=pk)
    user = employee.user

```

```

if request.method == 'POST':
    employee.delete()
    user.delete()
    return redirect('managers')

return render(request, 'delete_confirmation.html', {'employee': employee, 'user': user})

```

```

@login_required
def bills_view(request):
    bills = MaintenanceBill.objects.all()

    rental_space_filter = request.GET.get('rental_space')
    creation_date_filter = request.GET.get('creation_date')

    if rental_space_filter:
        bills = bills.filter(rental_space__number__icontains=rental_space_filter)
    if creation_date_filter:
        bills = bills.filter(creation_date__gte=creation_date_filter)

    context = {
        'bills': bills,
    }
    return render(request, 'bills.html', context)

```

```

@login_required
def add_bill_view(request):
    if request.method == "POST":
        form = MaintenanceBillForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('bills')
    else:
        form = MaintenanceBillForm()
    return render(request, 'bill_form.html', {'form': form})

```

```

@login_required
def edit_bill_view(request, bill_id):
    bill = get_object_or_404(MaintenanceBill, pk=bill_id)
    if request.method == "POST":
        form = MaintenanceBillForm(request.POST, instance=bill)
        if form.is_valid():
            form.save()
            return redirect('bills')
    else:
        form = MaintenanceBillForm(instance=bill)
    return render(request, 'bill_form.html', {'form': form})

```

```

@login_required
def delete_bill_view(request, bill_id):
    bill = get_object_or_404(MaintenanceBill, pk=bill_id)
    if request.method == "POST":
        bill.delete()
        return redirect('bills')
    return render(request, 'bill_confirm_delete.html', {'bill': bill})

```

```

@login_required
def payments(request):
    payments = Payment.objects.all()

    # Фильтрация данных по GET-параметрам
    rental_space_filter = request.GET.get('rental_space')
    tenant_filter = request.GET.get('organization_name')
    payment_date_filter = request.GET.get('payment_date')

    if rental_space_filter:
        payments = payments.filter(rental_agreement__rental_space__icontains=rental_space_filter)
    if tenant_filter:
        payments =
payments.filter(rental_agreement__tenant__organization_name__icontains=tenant_filter)
    if payment_date_filter:
        payments = payments.filter(payment_date=payment_date_filter)

    return render(request, 'payments.html', {'payments': payments})

@login_required
def create_payment(request):
    if request.method == 'POST':
        form = PaymentForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('payments') # Перенаправляем пользователя на страницу с платежами после
создания платежа
        else:
            form = PaymentForm()
    return render(request, 'create_payment.html', {'form': form})

@login_required
def edit_payment(request, payment_id):
    payment = get_object_or_404(Payment, pk=payment_id)
    if request.method == 'POST':
        form = PaymentForm(request.POST, instance=payment)
        if form.is_valid():
            form.save()
            return redirect('payments') # Перенаправляем пользователя на страницу с платежами после
редактирования платежа
        else:
            form = PaymentForm(instance=payment)
    return render(request, 'edit_payment.html', {'form': form})

def delete_payment(request, payment_id):
    payment = get_object_or_404(Payment, pk=payment_id)
    if request.method == 'POST':
        payment.delete()
        return redirect('payments') # Перенаправляем пользователя на страницу с платежами после
удаления платежа
    return render(request, 'delete_payment.html', {'payment': payment})

@login_required
def analytics_view(request):
    return render(request, 'analytics.html')

```



```

def analytics(request):
    # Подготовка данных для круговой диаграммы состояний объектов недвижимости
    status_counts = RentalSpace.objects.values('status__description').annotate(count=Count('status'))
    pie_labels = [item['status__description'] for item in status_counts]
    pie_sizes = [item['count'] for item in status_counts]

    # Подготовка данных для гистограммы поступлений платежей по месяцам
    payments = Payment.objects.all().values('payment_date', 'amount')
    df = pd.DataFrame(payments)
    df['payment_date'] = pd.to_datetime(df['payment_date'])
    df = df.set_index('payment_date')
    monthly_sums = df.resample('M').sum().reset_index()
    bar_labels = monthly_sums['payment_date'].dt.strftime('%Y-%m').tolist()
    bar_data = monthly_sums['amount'].tolist()

    # Подготовка данных для диаграммы состояний объектов недвижимости по статусу
    rental_spaces = RentalSpace.objects.select_related('status').all()
    timeline_data = []
    for space in rental_spaces:
        timeline_data.append({
            'label': f'{space.floor} - {space.number}',
            'status': space.status.description,
            'start_date': space.rentalagreement_set.first().start_date.strftime('%Y-%m-%d') if
space.rentalagreement_set.exists() else None,
            'end_date': space.rentalagreement_set.first().end_date.strftime('%Y-%m-%d') if
space.rentalagreement_set.exists() else None
        })

    context = {
        'pie_labels': pie_labels,
        'pie_sizes': pie_sizes,
        'bar_labels': bar_labels,
        'bar_data': bar_data,
        'timeline_data': timeline_data
    }
    return render(request, 'analytics.html', context)

@login_required
def logout(request):
    logout(request)
    return redirect('auth')

```

Файл tenant_list.html

```

<!-- tenant_list.html -->
{% extends 'base.html' %}
{% block content %}
<div class="container-fluid">
    <h2>Арендаторы</h2>
    <div class="d-flex justify-content-end mb-3">
        <a href="{% url 'tenant_add' %}" class="btn btn-success">Добавить арендатора</a>
    </div>
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Компания</th>
                <th>Фамилия</th>

```

```
<th>Имя</th>  
<th>Телефон</th>  
<th>Эл.почта</th>  
<th>Действия</th>  
</tr>  
<tr>  
    <form method="get" class="mb-3">  
        <th><input type="text" name="tenant_id" class="form-control" value="{{  
request.GET.tenant_id }}"></th>  
        <th><input type="text" name="organization_name" class="form-control" value="{{  
request.GET.organization_name }}"></th>  
        <th><input type="text" name="surname" class="form-control" value="{{  
request.GET.surname }}"></th>  
        <th><input type="text" name="name" step="0.01" class="form-control" value="{{  
request.GET.name }}"></th>  
        <th><input type="number" name="phone_number" step="0.01" class="form-control"  
value="{{ request.GET.phone_number }}"></th>  
        <th></th>  
        <th><button type="submit" class="btn btn-primary btn-sm">  
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"  
class="bi bi-funnel-fill" viewBox="0 0 16 16">  
                <path d="M1.5 1.5A5.5 0 0 1 2 1h12a.5.5 0 0 1 .5.5v2a.5.5 0 0 1-.128.334L10  
8.692V13.5a.5.5 0 0 1-.342.474l-3 1A.5.5 0 0 1 6 14.5V8.692L1.628 3.834A.5.5 0 0 1 1.5 3.5v-2z"/>  
            </svg>  
        </button>  
        </th>  
    </form>  
</tr>  
</thead>  
<tbody>  
    {% for tenant in tenants %}  
    <tr>  
        <td>{{ tenant.tenant_id }}</td>  
        <td>{{ tenant.organization_name }}</td>  
        <td>{{ tenant.surname }}</td>  
        <td>{{ tenant.name }}</td>  
        <td>{{ tenant.phone_number }}</td>  
        <td>{{ tenant.email }}</td>  
        <td>  
            <a href="{% url 'tenant_edit' tenant.tenant_id %" class="btn btn-sm btn-primary">  
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"  
class="bi bi-pencil-fill" viewBox="0 0 16 16">  
                    <path d="M12.854 1.46a.5.5 0 0 0-.707 0L10.5 1.793 14.207 5.5l1.647-1.646a.5.5 0 0 0  
0-.708l-3-3zm.646 6.061L9.793 2.5 3.293 9H3.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1  
.5.5v.5h.5a.5.5 0 0 1 .5.5v.207l6.5-6.5zm-7.468 7.468A.5.5 0 0 1 6 13.5V13h-.5a.5.5 0 0 1-.5-.5V12h-  
.5a.5.5 0 0 1-.5-.5V11h-.5a.5.5 0 0 1-.5-.5V10h-.5a.499.499 0 0 1-.175-.032l-.179.178a.5.5 0 0 0-  
.11.168l-2 .5a.5.5 0 0 0 .65.65l5-2a.5.5 0 0 0 .168-.11l.178-.178z"/>  
                </svg>  
            </a>  
            <a href="{% url 'tenant_delete' tenant.tenant_id %" class="btn btn-sm btn-danger">  
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"  
class="bi bi-trash-fill" viewBox="0 0 16 16">  
                    <path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 1 1H3v9a2 2 0 0 0 2h6a2 2 0 0 0 2-  
2V4h.5a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1-1H2.5zm3 4a.5.5 0 0 1  
.5.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 -.5-.5zM8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 8 5zm3  
.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 1 0z"/>  
                </svg>  
            </a>  
        </td>  
    </tr>  
    </tbody>  
</table>
```

```

        </a>
    </td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
{% endblock %}

```

Файл rental_objects.html

```

{% extends 'base.html' %}
{% block content %}

```

```

<div class="container">
    <h2>Арендные помещения</h2>
    <div class="row">
        <div class="col d-flex justify-content-end">
            <a href="{% url 'add_rental_space' %}" class="btn btn-success mb-3">Добавить
помещение</a>
        </div>
    </div>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Здание</th>
                <th scope="col">Этаж</th>
                <th scope="col">Номер помещения</th>
                <th scope="col">Статус</th>
                <th scope="col">Площадь</th>
                <th scope="col">Менеджер</th>
                <th scope="col">Комментарий</th>
                <th scope="col">Действия</th>
            </tr>
            <tr>
                <form method="get" class="mb-3">
                    <th><input type="text" name="building" class="form-control" placeholder="{ {
request.GET.building } }"></th>
                    <th><input type="number" name="floor" class="form-control" placeholder="{ {
request.GET.floor } }"></th>
                    <th><input type="number" name="number" class="form-control" placeholder="{ {
request.GET.number } }"></th>
                    <th>
                        <select name="status" class="form-control">
                            <option value="">Все</option>
                            <option value="Свободно" {% if request.GET.status == "Свободно" %}>selected{%
endif %}>Свободно</option>
                            <option value="Занято" {% if request.GET.status == "Занято" %}>selected{% endif
%}>Занято</option>
                            <option value="Ремонт" {% if request.GET.status == "Ремонт" %}>selected{% endif
%}>Ремонт</option>
                            <option value="Забронировано" {% if request.GET.status == "Забронировано"
%}>selected{% endif %}>Забронировано</option>
                            <option value="Ожидает съезда" {% if request.GET.status == "Ожидает съезда"
%}>selected{% endif %}>Ожидает съезда</option>
                        </select>
                    </th>

```

```

        </th>
        <th><input type="number" name="area" step="0.01" class="form-control" placeholder="{{
request.GET.area }}"></th>
        <th><input type="text" name="curator" class="form-control" placeholder="{{
request.GET.curator }}"></th>
        <th></th>
        <th><button type="submit" class="btn btn-primary btn-sm">
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-funnel-fill" viewBox="0 0 16 16">
                <path d="M1.5 1.5A.5.5 0 0 1 2 1h12a.5.5 0 0 1 .5.5v2a.5.5 0 0 1-.128.334L10
8.692V13.5a.5.5 0 0 1-.342.474l-3 1A.5.5 0 0 1 6 14.5V8.692L1.628 3.834A.5.5 0 0 1 1.5 3.5v-2z"/>
            </svg>
        </button>
        </th>
    </form>
</tr>
</thead>
<tbody>
    {% for rental_space in rental_spaces %}
    <tr>
        <td>{{ rental_space.building.name }}</td>
        <td>{{ rental_space.floor }}</td>
        <td>{{ rental_space.number }}</td>
        <td>{{ rental_space.status }}</td>
        <td>{{ rental_space.area }}</td>
        <td>{{ rental_space.curator.employee.fio }}</td>
        <td>{{ rental_space.description }}</td>
        <td>
            <a href="{% url 'edit_rental_space' rental_space.rental_space_id %}" class="btn btn-sm btn-
primary">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-pencil-fill" viewBox="0 0 16 16">
                    <path d="M12.854 1.46a.5.5 0 0 0-.707 0L10.5 1.793 14.207 5.51l1.647-1.646a.5.5 0 0 0
0-.708l-3-3zm.646 6.061L9.793 2.5 3.293 9H3.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1
.5.5v.5h.5a.5.5 0 0 1 .5.5v.207l6.5-6.5zm-7.468 7.468A.5.5 0 0 1 6 13.5V13h-.5a.5.5 0 0 1-.5-.5V12h-
.5a.5.5 0 0 1-.5-.5V11h-.5a.5.5 0 0 1-.5-.5V10h-.5a.499.499 0 0 1-.175-.032l-.179.178a.5.5 0 0 0-
.11.168l-2 5a.5.5 0 0 0 .65.65l5-2a.5.5 0 0 0 .168-.11l.178-.178z"/>
                </svg>
            </a>
            <a href="{% url 'delete_rental_space' rental_space.rental_space_id %}" class="btn btn-sm
btn-danger">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-trash-fill" viewBox="0 0 16 16">
                    <path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 1 1H3v9a2 2 0 0 0 2 2h6a2 2 0 0 0 2-
2V4h.5a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1-1H2.5zm3 4a.5.5 0 0 1
.5.5v7a.5.5 0 0 1-1 1v-7a.5.5 0 0 1-.5-.5zm8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 1v-7a.5.5 0 0 1-.5-.5zm3
.5v7a.5.5 0 0 1-1 1v-7a.5.5 0 0 1 1 1v7z"/>
                </svg>
            </a>
        </td>
    </tr>
    {% endfor %}
</tbody>
</table>
</div>
{% endblock %}

```

