Департамент образования и науки города Москвы

Государственное бюджетное профессиональное образовательное учреждение

города Москвы

«Колледж малого бизнеса № 4»

(ГБПОУ КМБ № 4)

**КУРСОВОЙ ПРОЕКТ**

по профессиональному модулю ПМ.05 Проектирование и разработка информационных систем по специальности СПО 09.02.07 Информационные системы и программирование

Тема: Разработка портала для управления жилым комплексом: регистрация заявок на ремонт, бронирование общих помещений, обсуждения жильцов.

Выполнил(а) студент(ка)

3 курса группы № ИПО-33.22

Костиков Алексей Викторович

Проверил

руководитель курсовой работы

Чепрасова Анастасия Сергеевна

Работа защищена с оценкой

« » 2025

Москва, 2025

Департамент образования и науки города Москвы

Государственное бюджетное профессиональное образовательное учреждение города Москвы «Колледж малого бизнеса № 4» (ГБПОУ КМБ № 4)

УТВЕРЖДАЮ

Заместитель директора

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_И.Ю.Атрощенко

**ЗАДАНИЕ**

на курсовую работу

по профессиональному модулю ПМ.05 Проектирование и разработка информационных систем

по специальности СПО 09.02.07 Информационные системы и программирование

Студента группы ИПО-33.22, Костиков Алексей Викторович

Тема курсовой работы: Разработка портала для управления жилым комплексом: регистрация заявок на ремонт, бронирование общих помещений, обсуждения жильцов.

Срок защиты курсовой работы: 24.04.2025 г.

**1. Исходные данные к курсовой работе:** описание объектов автоматизации, технические характеристики и требования к мониторингу производительности, спецификация проекта, учебно-методическая литература и нормативные акты, материалы периодической печати, интернет-ресурсы.

**2. Перечень подлежащих разработке вопросов:**

Введение

1. Теоретический раздел:
   1. Принципы автоматизации управления жилыми комплексами: основные задачи и преимущества.
   2. Подходы к реализации онлайн-сервисов для регистрации заявок на ремонт.
   3. Функциональные особенности бронирования общих помещений (например, конференц-залов или спортзалов).
   4. Роль порталов для коммуникации и обсуждений между жильцами.
2. Исследовательский раздел:
   1. Анализ существующих решений для автоматизации управления жилыми комплексами (например, Domopult, Умный дом).
   2. Исследование потребностей жильцов и управляющих компаний в функционале портала.
   3. Формирование требований к функциональности портала, включая удобство интерфейса и безопасность данных.
   4. Обоснование выбора технологий для реализации функционала портала (базы данных, серверные технологии, интерфейсы).
3. Практический раздел:
   1. Проектирование структуры базы данных для хранения информации о жильцах, заявках на ремонт, бронированиях и обсуждениях.
   2. Разработка серверной части портала для обработки заявок, управления бронированиями и взаимодействия пользователей.
   3. Реализация пользовательского интерфейса для:
      1. Регистрации заявок на ремонт с указанием типа проблемы и времени выполнения.
      2. Бронирования общих помещений с учетом доступности и временных интервалов.
      3. Создания и участия в обсуждениях между жильцами.
   4. Внедрение системы уведомлений для пользователей о статусе заявок, бронирований и новых обсуждениях.
   5. Настройка ролей пользователей: жильцы, администраторы, сотрудники управляющей компании.
   6. Обеспечение безопасности данных: защита персональной информации жильцов и заявок.
   7. Тестирование системы, проверка функциональности, удобства интерфейса и производительности.
   8. Разработка документации для пользователей и администраторов портала.

Заключение

Список использованных источников

**3. Перечень графического материала:** ER-диаграмма базы данных, UML-диаграммы (прецедентов, классов, последовательности), прототипы пользовательских интерфейсов (заявки на ремонт, бронирование помещений, обсуждения), визуализация бронирования общих помещений (например, календарь), схема взаимодействия компонентов системы (сервер, клиент, база данных).

**4. Дата выдачи задания:** \_\_\_ . \_\_\_ . 20\_\_\_ г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_ . \_\_\_ . 20\_\_\_ г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_heading=h.e4db7s3s6orp) 1

[ГЛАВА 1. Теоретический раздел](#_heading=h.h4gckutmw95j) 4

[1.1.](#_heading=h.6tiggs4e3yg) Принципы автоматизации управления жилыми комплексами: основные задачи и преимущества.4

[1.2.](#_heading=h.spna5h93onr0) Подходы к реализации онлайн-сервисов для регистрации заявок на ремонт.5

[1.3.](#_heading=h.j7znvxl7lo2r) Функциональные особенности бронирования общих помещений (например, конференц-залов или спортзалов).5

[1.4.](#_heading=h.k36mfigywac6) Роль порталов для коммуникации и обсуждений между жильцами. 7

[ГЛАВА 2. Исследовательский раздел](#_heading=h.k39k6fgl4ua) 9

[2.1.](#_heading=h.icw4ntxun8r) Анализ существующих решений для автоматизации управления жилыми комплексами (например, Domopult, Умный дом).9

[2.2.](#_heading=h.1upch7431x6c) Исследование потребностей жильцов и управляющих компаний в функционале портала.10

[2.3.](#_heading=h.hwwod1uz56h) Проектирование структуры базы данных для хранения информации о жильцах, заявках на ремонт, бронированиях и обсуждениях.12

[2.4.](#_heading=h.6a5dua1awl9a)  Обоснование выбора технологий для реализации функционала портала (базы данных, серверные технологии, интерфейсы) 14

[ГЛАВА 3. Практический раздел](#_heading=h.ofryuygmhel8) 16

[3.1.](#_heading=h.w5go24fh2gtb) Настройка и сопровождение жк-системы на базе выбранной платформы16

[3.2.](#_heading=h.gj9p37scyq49)Разработка серверной части портала для обработки заявок, управления бронированиями и взаимодействия пользователей.21

[3.3.](#_heading=h.51tolm2nx6dz) Реализация пользовательского интерфейса для:

* + 1. Регистрации заявок на ремонт с указанием типа проблемы и времени выполнения. 25
    2. Бронирования общих помещений с учетом доступности и временных интервалов. 27
    3. Создания и участия в обсуждениях между жильцами. 29

[3.4.](#_heading=h.w0vtgrg2ghyw) Внедрение системы уведомлений для пользователей о статусе заявок, бронирований и новых обсуждениях.30

[3.5.](#_heading=h.g3jen2psvroz) Настройка ролей пользователей: жильцы, администраторы, сотрудники управляющей компании.31

[3.6.](#_heading=h.iorbomohrmzk) Обеспечение безопасности данных: защита персональной информации жильцов и заявок.32

3.7. Тестирование системы, проверка функциональности, удобства интерфейса и производительности. 33

3.8. Разработка документации для пользователей и администраторов портала. 34

[ЗАКЛЮЧЕНИЕ](#_heading=h.81n8u1upjzti) 39

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_heading=h.n96s9zjk2zv6) 40

[ПРИЛОЖЕНИЕ](#_heading=h.kbkuvtjyx7rn) 46

# 

# ВВЕДЕНИЕ

Современные жилые комплексы становятся все более сложными и многофункциональными объектами, требующими эффективного управления и взаимодействия между жильцами и управляющими компаниями. В условиях растущей урбанизации и увеличения числа многоквартирных домов, необходимость в создании удобных и доступных инструментов для управления жилыми комплексами становится особенно актуальной. Одним из таких инструментов является веб-портал, который позволяет жильцам легко регистрировать заявки на ремонт, бронировать общие помещения и участвовать в обсуждениях.

Целью данной курсовой работы является разработка функционального веб-портала для управления жилым комплексом, который будет включать в себя модули для регистрации заявок на ремонт, бронирования общих помещений и обсуждений жильцов. В процессе работы были выполнены такие задачи как: анализ требований, проектирование архитектуры, выбор технологий,азработка документации, тестирование системы, обеспечение безопасности данных, разработка функциональных модулей.

.

Работа направлена на создание эффективного инструмента для улучшения качества жизни жильцов многоквартирных домов и оптимизации процессов управления жилыми комплексами.

1. Анализ требований
   * 1. Исследовать потребности жильцов и управляющих компаний в функционале портала.
     2. Определить ключевые функции, которые должны быть реализованы (регистрация заявок на ремонт, бронирование общих помещений, обсуждения жильцов).
2. Проектирование архитектуры системы
   * 1. Разработать архитектуру веб-портала, включая серверную и клиентскую части.
     2. Определить структуру базы данных для хранения информации о заявках, бронированиях и обсуждениях.
3. Выбор технологий
   * 1. Выбрать подходящие технологии и инструменты для разработки портала (язык программирования Golang, фреймворки для фронтенда, СУБД PostgreSQL и т.д.).
     2. Оценить возможности интеграции с внешними сервисами (например, для уведомлений или аутентификации).
4. Разработка функциональных модулей
   * 1. Реализовать модуль регистрации заявок на ремонт с возможностью отслеживания статуса.
     2. Создать систему бронирования общих помещений с календарем доступности и уведомлениями.
     3. Разработать платформу для обсуждений жильцов с возможностью комментирования и голосования.
5. Обеспечение безопасности данных
   * 1. Реализовать механизмы аутентификации и авторизации пользователей.
     2. Обеспечить защиту данных от несанкционированного доступа и утечек.
6. Тестирование системы
   * 1. Провести тестирование всех функциональных модулей на корректность работы и производительность.
     2. Оценить пользовательский опыт и внести необходимые улучшения.
7. Разработка документации
   * 1. Подготовить эксплуатационную документацию по использованию портала.
     2. Создать техническую документацию для дальнейшего сопровождения и обновления системы.
8. Внедрение и обучение пользователей
   * 1. Организовать процесс внедрения портала в эксплуатацию.
     2. Провести обучение жильцов и сотрудников управляющей компании по использованию системы.

Разработка портала для управления жилым комплексом представляет собой комплексную задачу, включающую в себя как технические аспекты, так и вопросы пользовательского опыта. Веб-портал должен быть интуитивно понятным, обеспечивать высокую степень безопасности данных и предоставлять жильцам возможность оперативно взаимодействовать с управляющей компанией. Использование современных технологий, таких как язык программирования Golang для серверной части, а также базы данных PostgreSQL для хранения информации, позволит создать надежное и производительное решение.

## ГЛАВА 1. Теоретический раздел

В данном разделе будут рассмотрены основные принципы и подходы к сопровождению сервиса для жк систем, включая методы обновления данных, мониторинг производительности и обеспечение актуальности информации.

**1.1** Принципы автоматизации управления жилыми комплексами: основные задачи и преимущества.

При разработке веб-порталов для управления жилыми комплексами следует учитывать несколько ключевых принципов:

Пользовательский опыт (UX): Интерфейс должен быть интуитивно понятным и удобным для пользователей с разным уровнем технической подготовки. Важно обеспечить простоту навигации и доступность всех функций.

Функциональность: Портал должен включать все необходимые функции, такие как регистрация заявок на ремонт, бронирование общих помещений и обсуждения жильцов, с возможностью отслеживания статуса.

Безопасность данных: Реализация надежных механизмов аутентификации и авторизации пользователей для защиты личной информации жильцов. Данные должны храниться в зашифрованном виде.

Масштабируемость: Архитектура должна позволять легко добавлять новые функции и модули без значительных изменений в существующей системе.

Интеграция с внешними системами: Возможность интеграции с другими системами (например, платежными) для расширения функционала.

Производительность: Обеспечение быстрой загрузки страниц и высокой производительности при работе с большим объемом данных.

Адаптивность: Портал должен корректно отображаться на различных устройствах (компьютерах, планшетах, смартфонах).

Поддержка и обновление: Регулярное обновление системы и исправление ошибок, а также поддержка пользователей через службу технической поддержки.

**1.2.** Подходы к реализации онлайн-сервисов для регистрации заявок на ремонт.

Архитектура веб-портала включает в себя несколько ключевых компонентов: Серверная часть на Golang:

Использование языка программирования Golang для разработки серверной логики позволяет создавать высокопроизводительные приложения с поддержкой параллельной обработки запросов.

База данных: PostgreSQL используется для хранения информации о заявках на ремонт, расписании бронирования общих помещений, данных о пользователях и обсуждениях жильцов. Эта СУБД обеспечивает надежное хранение данных и поддержку сложных запросов. Размер базы данных в PostgreSQL не ограничен. Соответствует требованиям ACID (А атомарность (от англ. atomicity), согласованность (от англ. consistency), изолированность (от англ. isolation), устойчивость (от англ. durability))

Пользовательский интерфейс: Интерфейс будет разработан с использованием HTML/CSS/JavaScript, что обеспечивает интерактивность и удобство использования. Фреймворки для фронтенда (например, React или Vue.js) могут быть использованы для создания динамичных компонентов интерфейса.

**1.3.** Функциональные особенности бронирования общих помещений

1. Процесс бронирования для пользователей:

* Пользователи могут забронировать помещение через форму на странице /room-booking
* При бронировании необходимо указать:
* ФИО
* Номер телефона
* Номер помещения
* Дату бронирования
* Есть ограничение на дату бронирования: можно бронировать только на даты в пределах 3 дней от текущей даты

2. Проверки при бронировании:

* Система проверяет, не забронировано ли уже помещение на выбранную дату
* Если помещение уже забронировано, пользователь получит сообщение об ошибке
* Бронирование доступно только авторизованным пользователям

3.Административные функции:

* Администратор имеет доступ к странице управления бронированиями (/admin-room-booking)
* Администратор может:
* Просматривать все заявки на бронирование
* Одобрять заявки (кнопка "Accept")
* Отклонять заявки (кнопка "Reject")
* Отменять своё решение (кнопка "Revert")

4. Уведомления:

* При одобрении бронирования администратором, пользователь получает уведомление
* Уведомление содержит информацию о том, что заявка на бронирование была одобрена

5. Отображение статуса бронирования:

* В личном кабинете пользователя отображается список его заявок на бронирование
* Для каждой заявки показывается:
* Номер помещения
* Статус (Одобрено/В ожидании)
* Телефон
* Дата бронирования
* Дата подачи заявки

6. Безопасность:

* Все действия с бронированиями требуют авторизации
* Административные функции доступны только пользователю с ролью "admin"
* Все запросы на бронирование проходят через серверную валидацию

7. Интерфейс:

* Простой и понятный интерфейс для подачи заявок на бронирование
* Чёткое отображение статуса заявок
* Интуитивно понятные кнопки действий для администратора

**1.4. Р**оль порталов для коммуникации и обсуждений между жильцами.

Чат-система реализована в реальном времени с использованием WebSocket, что обеспечивает мгновенную доставку сообщений и автоматическое обновление для всех пользователей. Она доступна как для обычных, так и для административных аккаунтов, позволяя отправлять сообщения, отображать имя отправителя и время, а также автоматически прокручивать чат к новым сообщениям. Пользователи могут отправлять сообщения по нажатию Enter. Интерфейс выполнен в современном дизайне с разделением на область сообщений с прокруткой, поле ввода и кнопку отправки. Сообщения стилизованы для удобства: выделение имени пользователя, отображение времени и отделение друг от друга. Чат адаптивен и хорошо работает на разных устройствах. Технически используется WebSocket для мгновенной доставки сообщений, история хранится в базе данных, реализована обработка ошибок и проверка авторизации. Перед отправкой сообщений происходит их валидация — исключены пустые сообщения. Администраторы имеют расширенные права: могут удалять чужие сообщения через кнопку "Delete" с подтверждением действия. В системе реализована интеграция с общей системой интерфейса: чат доступен вместе с функциями бронирования и заявок на ремонт, а все сообщения сохраняются для истории. Также есть система уведомлений: пользователи получают важные оповещения в отдельном разделе. Администраторы могут отслеживать активность — количество сообщений и взаимодействия пользователей. В целом, чат служит инструментом коммуникации между жильцами жилого комплекса для обсуждения вопросов, координации действий, получения уведомлений и взаимодействия с администрацией в реальном времени. Система спроектирована для удобства использования и безопасности, обеспечивая эффективное взаимодействие всех участников.

## ГЛАВА 2. Исследовательский раздел

В данной главе описаны основные технологии, использованные в проекте, обоснование их выбора и особенности применения для разработки портала для управления жилым комплексом

### 2.1. Анализ существующих решений для автоматизации управления жилыми комплексами

В рамках данного проекта был проведен анализ существующих решений для портала управления жилым комплексом, что позволило определить ключевые подходы, технологии и инструменты, используемые в этой области. Основное внимание уделялось системам, которые обеспечивают быструю передачу данных, использование userfriendly UX/UI что помогает пользователям использовать сервис в своих интересах. Чего не было в сервисе Smartservice

Современные порталы, такие как SmartService, Domme, предоставляют широкие возможности для управления жилищным комплексом. Эти платформы поддерживают передачу и хранение данных через собственные серверы, что позволяет оперативно получать актуальные услуги, такие как бронирование помещений,использование чата для общения с жителями жк, регистраций заявок на ремонт .

Одним из ключевых аспектов сопровождения такого портала является работа с бд. Такие бд как postgresql предоставляют быстрый доступ к актуальным данным и информации.

Для хранения и управления данными в современных сервисах широко используются специализированные системы управления базами данных, такие как PostgreSQL. Этот инструмент позволяет эффективно работать с данными, выполнять сложные запросы и обеспечивать высокую производительность системы. В рамках проекта PostgreSQL используется для хранения данных о регистрации и входе пользователей, бронировании помещений, хранения сообщений для чата, регистрации заявок на ремонт.Админ панель помогает использовать структурированную информацию, изменять ее и использовать в организационных целях.

Система процессов обновления данных является важным аспектом сопровождения. В проекте реализованы механизмы автоматизированного обновления данных через, что позволяет минимизировать ручной труд и обеспечивать регулярное получение актуальной информации. Например, данные о регистрации заявок на бронирование помещения автоматически обновляются через интеграцию с ui сайта.

Анализ существующих решений показал, что использование современных технологий, таких как PostgreSQL, gorilla/sockets, gorilla/session обеспечивает ряд важных преимуществ. Использование специализированных баз данных и инструментов для автоматизации процессов обновления повышает эффективность работы системы, снижая нагрузку и улучшая производительность.

### 2.2. Исследование потребностей жильцов и управляющих компаний в функционале портала.

Для успешной разработки портала управления жилым комплексом необходимо понять основные потребности его пользователей — жильцов и управляющих компаний. Проведенное исследование позволило выявить ключевые функции, которые должны быть реализованы для повышения эффективности коммуникации, удобства использования и удовлетворенности всех участников процесса.

В рамках исследования использовались методы опросов, интервью и анализ существующих решений. Были опрошены представители жильцов, управляющих компаний и обслуживающих организаций. Также изучены отзывы пользователей на аналогичных платформах и проведён анализ текущих проблем в управлении жилыми комплексами.

Потребности жильцов включают быстрый и удобный способ подачи заявок на ремонт и обслуживание, возможность бронирования общих помещений с автоматическим подтверждением, обсуждение актуальных вопросов с соседями и управляющей компанией через форум или чат. Жильцы также хотят получать уведомления о статусе своих заявок, предстоящих платежах и важных объявлениях. Просмотр истории обращений и платежей помогает им контролировать расходы. Важным аспектом является удобный интерфейс, особенно на мобильных устройствах.

Управляющие компании нуждаются в централизованном управлении заявками на ремонт с возможностью их распределения по исполнителям, а также в мониторинге статуса заявок в реальном времени. Для них важна возможность публикации объявлений и новостей для жильцов, а также аналитика по количеству обращений, времени их обработки и выполнению. Инструменты для организации бронирования общих помещений позволяют повысить эффективность использования ресурсов комплекса. Обеспечение обратной связи с жильцами через чат или форму обратной связи способствует улучшению коммуникации. Кроме того, управляющие компании заинтересованы в аналитике активности пользователей для повышения качества сервиса.

Анализ показал, что ключевыми компонентами эффективного портала являются системы подачи заявок на ремонт, бронирования общих помещений и обсуждения между жильцами и администрацией. Важным аспектом является интеграция уведомлений и статистики для повышения прозрачности и оперативности управления. Учитывая выявленные потребности, разработка портала должна обеспечить интуитивно понятный интерфейс, надежную систему авторизации и гибкие инструменты коммуникации.

Понимание потребностей жильцов и управляющих компаний является основой для формирования функционала портала. Реализованные модули должны обеспечить удобство взаимодействия, повысить уровень сервиса и способствовать более эффективному управлению жилым комплексом.

### 2.3. Проектирование структуры базы данных для хранения информации о жильцах, заявках на ремонт, бронированиях и обсуждениях.

При проектировании структуры базы данных для управления информацией о жильцах, заявках на ремонт, бронированиях и обсуждениях в системе на базе PostgreSQL важно обеспечить логическую связность и целостность данных. В предложенной модели используются следующие таблицы: users, repair\_requests, chat\_messages, room\_bookings и notifications.

Таблица users служит центральным хранилищем информации о жильцах и администраторах. Она содержит уникальный идентификатор id, имя пользователя username, пароль password и другие возможные поля (например, контактные данные). Поле username уникально и обязательно к заполнению, что обеспечивает однозначную идентификацию каждого пользователя.

Таблица repair\_requests предназначена для хранения заявок на ремонт. Она включает поля id — уникальный идентификатор заявки; name — название или краткое описание заявки; username — ссылка на пользователя из таблицы users (может быть реализована через внешний ключ); apartment — номер квартиры жильца; repair\_type — тип ремонта; comment — подробное описание проблемы; created\_at — дата и время создания заявки; is\_approved — статус утверждения заявки. Связь с пользователем реализуется через поле username или внешний ключ для обеспечения целостности данных.

Таблица chat\_messages предназначена для хранения сообщений в чатах обсуждений. В ней есть поля id, username (автор сообщения), message (текст сообщения), created\_at (дата и время отправки). Связь с пользователем осуществляется через поле username, что позволяет отображать имя отправителя и фильтровать сообщения по пользователю.

Таблица room\_bookings хранит информацию о бронированиях общих помещений. Включает поля id, name (имя забронировавшего), phone (контактный телефон), room\_number (номер помещения), booking\_date (дата бронирования), is\_approved (статус подтверждения бронирования) и created\_at. Связь с пользователем реализована через имя или телефон, а также возможна через внешний ключ на таблицу users.

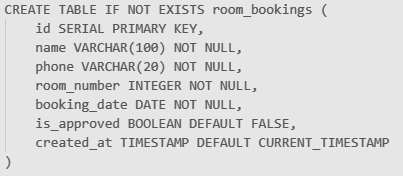
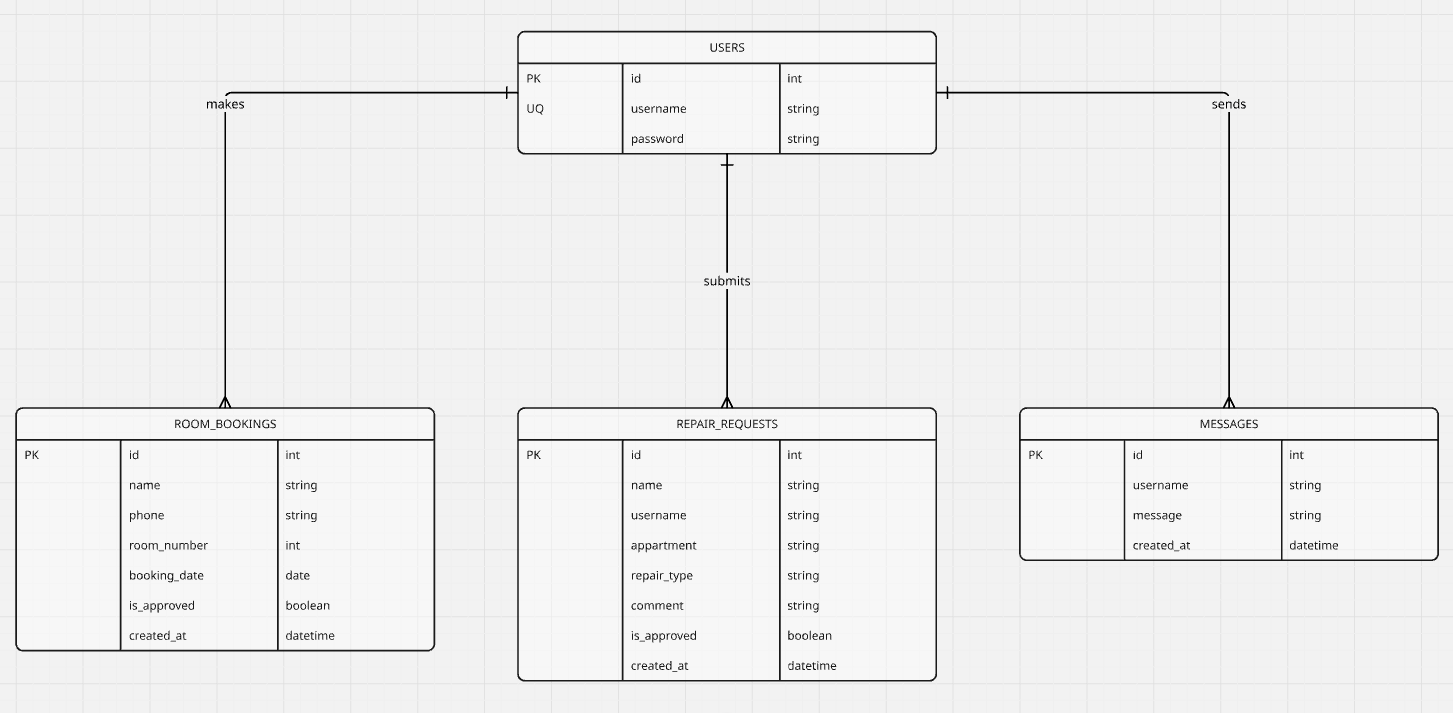


Таблица notifications содержит уведомления для пользователей. Поля id, username (кому предназначено уведомление), message (текст уведомления), is\_read (статус прочтения) и created\_at позволяют отслеживать важные сообщения и их статус.

Для обеспечения целостности данных рекомендуется использовать внешние ключи между таблицами: например, в таблице repair\_requests поле username ссылается на users(username), что предотвращает создание заявок от несуществующих пользователей. Аналогично для таблиц chat\_messages и room\_bookings можно установить связи с таблицей users по полю username или другим уникальным идентификаторам.

Таким образом, предложенная структура базы данных обеспечивает хранение всей необходимой информации о жильцах, их заявках, бронированиях и обсуждениях с возможностью расширения и интеграции дополнительных функций в будущем.



### 2.4. Обоснование выбора технологий для реализации функционала портала (базы данных, серверные технологии, интерфейсы)

Язык программирования Go (Golang) выбран как основной язык разработки по причинам высокой производительности, встроенной поддержки конкурентности, простого синтаксиса, хорошей стандартной библиотеки и эффективной работы с HTTP-запросами.

PostgreSQL используется как основная база данных благодаря своей надежности и стабильности, поддержке сложных запросов, хорошей производительности, поддержке транзакций, а также широким возможностям для масштабирования.

Для реализации чата в реальном времени применяется Gorilla WebSocket благодаря простой интеграции с Go, надежной работе с WebSocket соединениями, поддержке двусторонней коммуникации и эффективной обработке сообщений.

Управление сессиями пользователей осуществляется с помощью Gorilla Sessions, что обеспечивает безопасное хранение сессий, простую интеграцию с Go, поддержку cookie-based сессий и надежную работу с аутентификацией.

Для хеширования паролей используется Bcrypt благодаря безопасному хранению паролей, защите от атак перебором, стандартному алгоритму хеширования и встроенной защите от timing-атак.

Веб-страницы рендерятся при помощи HTML Templates благодаря встроенной поддержке в Go, безопасной обработке данных, простому синтаксису и эффективной работе с шаблонами.

Передача данных осуществляется через протокол HTTP/HTTPS благодаря его стандартной поддержке в Go, широкой совместимости и возможности использования различных методов запросов.

Для стилизации веб-интерфейса применяется CSS благодаря гибкой настройке внешнего вида, поддержке адаптивного дизайна и простоте использования.

Клиентская логика реализуется на JavaScript для взаимодействия с WebSocket, динамического обновления контента, обработки пользовательских действий и улучшения пользовательского опыта.

Работа с базой данных осуществляется через SQL — стандартный язык запросов, обеспечивающий эффективную работу с данными, поддержку сложных запросов, надежность и проверенность.

Выбор этих технологий обусловлен их надежностью и стабильностью работы, простотой интеграции между компонентами, высокой производительностью, безопасностью данных, масштабируемостью, соответствием современным стандартам, наличием хорошей документации и активным сообществом. Все эти технологии хорошо сочетаются друг с другом и создают надежную и эффективную систему для управления жилым комплексом

## ГЛАВА 3. Практический раздел

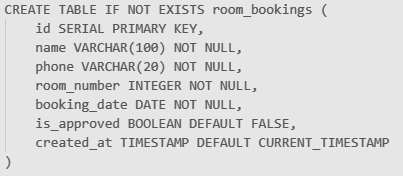
В практическом разделе будет описана реализация системы сопровождения и обновления жк системы, включая настройку платформы, разработку механизмов обновления данных, интеграцию системы мониторинга, оптимизацию базы данных и картографического сервиса, тестирование производительности и разработку эксплуатационной документации.

### 3.1. Настройка и сопровождение жк-системы на базе выбранной платформы

Настройка и сопровождение жк-системы на базе выбранной платформы являются ключевыми этапами реализации проекта. В рамках данного проекта в качестве основной платформы для хранения и обработки пространственных данных была выбрана связка PostgreSQL, а для визуализации и взаимодействия с пользователем — html и яп golang для серверной части. Настройка базы данных (PostgreSQL):

1. Установка и конфигурация PostgreSQL:
   * Установка СУБД PostgreSQL на сервер.
   * Настройка параметров производительности
   * Создание пользователей и назначение прав доступа.
2. Создание структуры базы данных:
   * Разработка схемы базы данных, включая таблицы для хранения информации о пользователях,заявках на ремонт, бронировании помещений

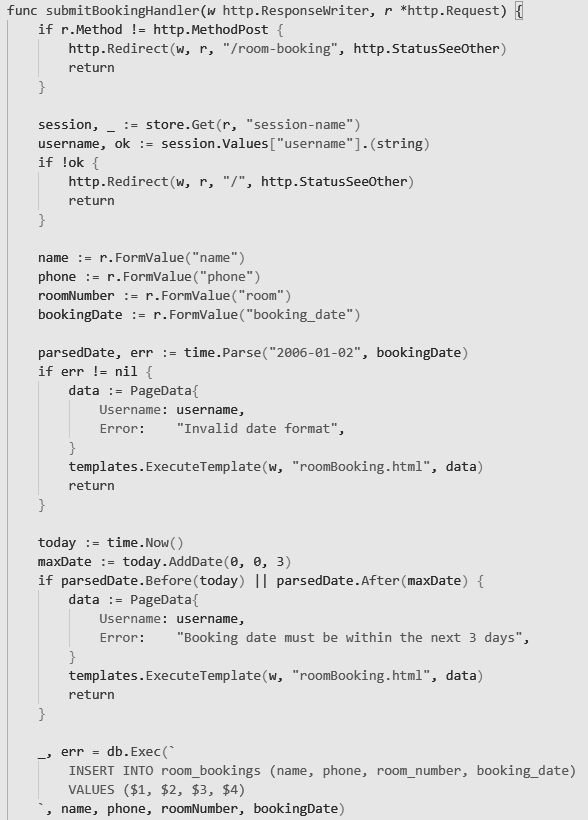
Пример SQL-скрипта для создания таблиц если она не существует в PostgreSQL изображен на рисунке 3.



**Рисунок 3 – создание таблицы для хранения данных о пользователях**

1. **Настройка передачи данных в бд через сервер:**
   1. проверяется, является ли метод HTTP-запроса POST. Если это не так, пользователь перенаправляется на страницу бронирования комнат (/room-booking).
   2. Код получает сессию пользователя из хранилища сессий. Затем он пытается извлечь имя пользователя из значений сессии. Если имя пользователя не найдено (например, пользователь не вошел в систему), происходит перенаправление на главную страницу (/)
   3. извлекаются значения из формы бронирования: имя (name), телефон (phone), номер комнаты (roomNumber) и дата бронирования (bookingDate).
   4. разобрать строку даты в формате YYYY-MM-DD. Если формат неверный, пользователю показывается ошибка.
   5. проверяется, находится ли дата бронирования в пределах следующих трех дней. Если дата находится вне этого диапазона (либо в прошлом, либо более чем через три дня), пользователю показывается соответствующее сообщение об ошибке.
   6. Если все проверки пройдены успешно, данные о бронировании вставляются в базу данных. Если возникает ошибка при выполнении запроса к базе данных (например, проблемы с подключением или синтаксисом SQL), пользователю возвращается ошибка сервера.

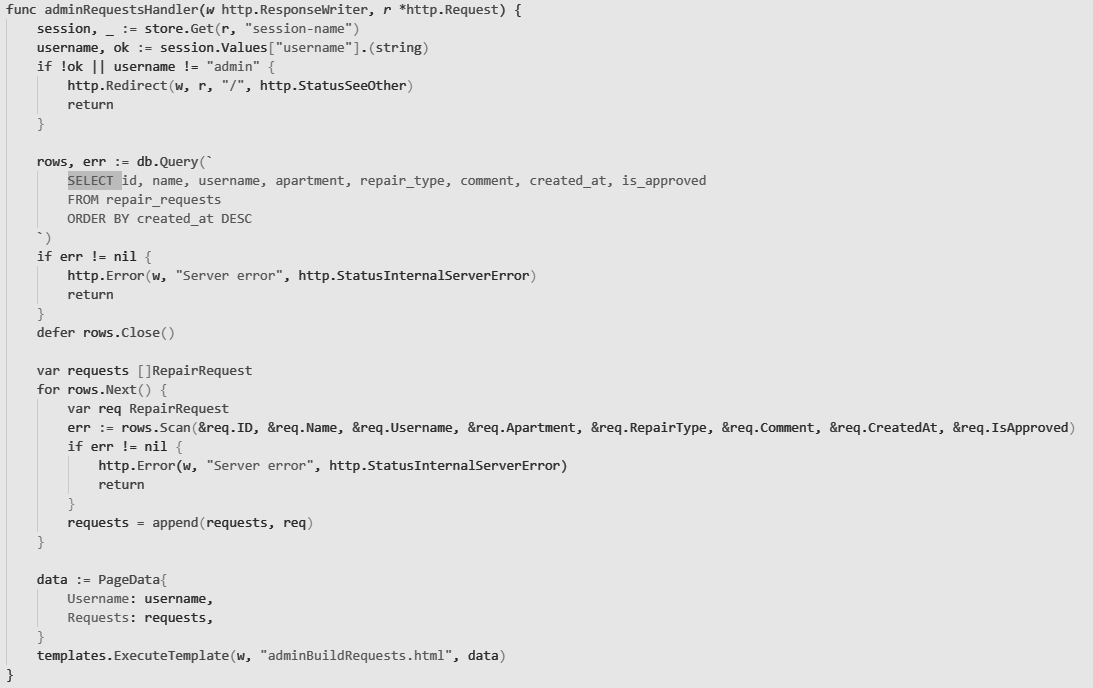
Код на рисунке 4:



**Рисунок 4**

1. **Работа с бд в админ панели:**
   * доступ к этому обработчику разрешён только пользователю с именем "admin"
   * Выполняется SQL-запрос к таблице repair\_requests, чтобы получить все заявки на ремонт
   * Создаётся срез requests для хранения всех заявок. В цикле перебираются все строки результата запроса. Каждая строка сканируется в структуру RepairRequest

Пример кода из main.go, где используется проверка прав доступа, получение данных из бд на рисунке 5. Этот фрагмент кода показывает, как система взаимодействует с бд, составляя удобные таблицы для использования.



**Рисунок 5 – пример кода, где используется роверка прав доступа, получение данных из бд**

Ручная проверка работоспособности системы:

1. **Регулярный запуск приложения:**
   * Проверка корректности отображения данных статуса заявок и сообщений чата.
   * Проверка актуальности данных
   * Проверка работоспособности взаимодействия с базой данных PostgreSQL.
   * Проверка отзывчивости пользовательского интерфейса и времени отклика сервера.
2. **Использование чек-листа:**
   * Разработан чек-лист, который включает основные сценарии использования системы и ожидаемые результаты. Это позволяет оперативно выявлять проблемы и обеспечивать базовый уровень контроля над работоспособностью системы.

**Чек-лист для проверки работоспособности системы:**

1. Проверка корректности отображения данных на карте

* Загружается ли чат через бд?
* Корректно ли они отображаются?
* Работает ли регистрация заявки в для ремонта?
* Работает ли регистрация заявки для бронирования помещения
* Отображаются ли данные в админ панели?

2. Проверка актуальности погодных данных

* Получаются ли данные через бд?
* Корректно ли работотают запросы?
* Работает ли ограничения по заявкам бронирования?

3. Проверка взаимодействия с базой данных

* Корректно ли сохраняются данные о пользователях в PostgreSQL?
* Работает ли аутентификация пользователей (логин/пароль)?
* Сохраняются ли остальные данные в базе данных?

4. Проверка отзывчивости пользовательского интерфейса

* Корректно ли работают все кнопки и элементы управления?

5. Проверка времени отклика сервера

* Сколько времени занимает загрузка данных с сервера?
* Нет ли задержек при запросах от пользователей?
* Как быстро обновляются данные в бд

6. Проверка обработки ошибок

* Как система реагирует на отсутствие интернет-соединения?
* Отображаются ли сообщения об ошибках при некорректных запросах?
* Работает ли восстановление после сбоев (например, при повторной загрузке данных)?

Сопровождение системы:

1. Обновление данных:
   * Настройка задач для обновления данных через админ-панель
2. Резервное копирование и восстановление:
   * Настройка регулярного резервного копирования базы данных.
   * Разработка процедур восстановления данных в случае сбоев.

**3.2. Разработка серверной части портала для обработки заявок, управления бронированиями и взаимодействия пользователей.**

Одной из ключевых задач сопровождения жк-систем является обеспечение хранения и передачи данных. В условиях постоянного изменения информации, такой как регистрация заявок и пользователей, изменение их статусов, необходимо разработать механизм своевременного и правильного метода изменения и обновления данных. Этот механизм должен обеспечивать актуальные данные в жк-системах с минимальным вмешательством оператора, а также поддерживать целостность и согласованность данных.  
Для реализации механизма своевременного и правильного метода изменения и обновления данных жк-систем была разработана архитектура, включающая следующие компоненты: база данных жк-систем, механизм обработки данных, система мониторинга и пользовательский интерфейс.

1. База данных жк-систем служит хранилищем для текущих и обновленных данных.
2. Механизм обработки данных отвечает за преобразование данных, полученных из внешних источников, в формат, совместимый с пользовательским интерфейсом.
3. Система мониторинга отслеживает процесс обновления и уведомляет о возможных ошибках, а пользовательский интерфейс позволяет управлять процессом обновления и просматривать результаты.

Механизм передачи и получения данных был реализован с использованием технологий PostgreSQL для хранения и обработки информации. Процесс своевременного и правильного метода изменения и обновления данных включает несколько этапов:

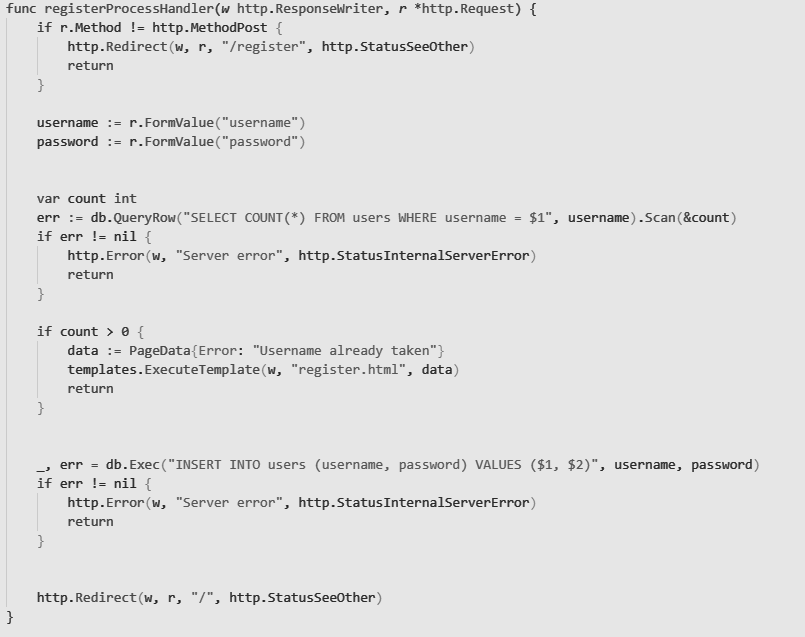
1. Загрузка данных. Данные загружаются из внешних источников через веб-интерфейс с помощью библиотек "sessions" и "websocket" для управления сессиями пользователей.
2. Обработка данных. Данные преобразуются в формат, совместимый с жк-системами, и проверяются на целостность.
3. Обновление базы данных. Новые данные интегрируются в существующую базу данных PostgreSQL.
4. Валидация данных. Система проверяет корректность обновленных данных и уведомляет оператора в случае обнаружения ошибок.
5. Визуализация обновленных данных. Обновленные данные отображаются в пользовательском интерфейсе жк-системы с использованием HTML для представления информации.

Для обеспечения актуальности данных в жк-системе был разработан механизм своевременного и правильного метода изменения и обновления данных в базе. Пример main.go, где выполняется запрос бд для получения пароля рисунок 6. Пример из main.go, где выполняется добавление новой локации в базу данных изображен на рисунке 7.



**Рисунок 6 – пример кода, где выполняется запрос к бд для получения данных о погоде**

Этот код демонстрирует, как система получает данные о пароле с использованием запросов к бд и обработкой их ошибок.



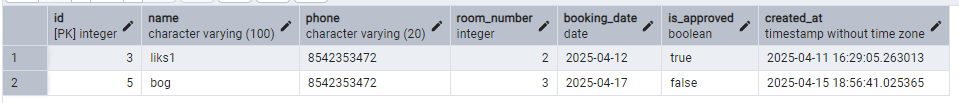
**Рисунок 7 – пример кода, где выполняется добавление новой строки в базу данных с пользователями**

Этот фрагмент кода показывает, как данные добавляются в базу данных. Перед загрузкой они проверяются на целостность и сохраняются в базе данных для дальнейшего использования.

Для демонстрации работы механизма передачи и получения данных были проведены тестовые обновления информации в жк-системе. На рисунках 7.1 и 7.2 представлены визуализации данных до и после обновления. На рисунке 7.1 показаны устаревшие данные о заявках и пользователях, а на рисунке 7.2 — обновленные данные, включающие новые заявки, изменения статусов и актуальную информацию о пользователях. На рисунке 7.3 изображена визуализация данных и ее функций через админ панель, доступ к которой имеет только администратор, что стало возможно благодаря правам доступа Эти визуализации наглядно демонстрируют эффективность работы механизма и его способность оперативно реагировать на изменения в информации.

Разработанный механизм передачи и получения данных в жк-системе позволяет поддерживать актуальность данных в системе, минимизировать ручное вмешательство оператора и обеспечивать высокую производительность системы.

В дальнейшем возможно расширение функциональности механизма за счет интеграции дополнительных источников данных и улучшения алгоритмов обработки и валидации данных.



**Рисунок 7.1 – визуализация данных до обновления**

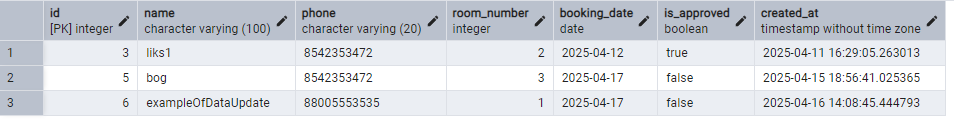
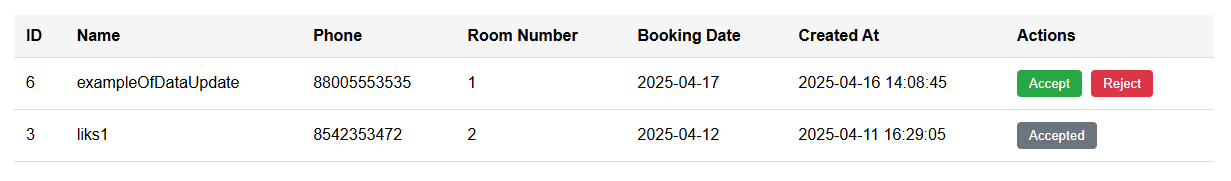
Картографические данные до обновления

Рисунок 7.2 – визуализация картографических данных после обновления

****

**7.3 - визуализация бд в панели администратора**

### 3.3. Реализация пользовательского интерфейса для регистрации заявок на ремонт с указанием типа проблемы и времени выполнения.

3.8.1 **Регистрации заявок на ремонт с указанием типа проблемы и времени выполнения.**

Для реализации пользовательского интерфейса, предназначенного для регистрации заявок на ремонт с указанием типа проблемы и времени выполнения, создана удобная и интуитивно понятная веб-страница, которая позволяет жильцам легко подавать заявки и отслеживать их статус. В рамках темы курсовой работы "Разработка портала для управления жилым комплексом" такой интерфейс должен включать следующие ключевые компоненты:

Форма подачи заявки на ремонт: Поле для выбора типа проблемы (с помощью выпадающего списка или), что позволит жильцу указать характер неисправности (неисправность электрики и т.д.).

Поле для описания проблемы (текстовое поле), где пользователь сможет подробно описать ситуацию.

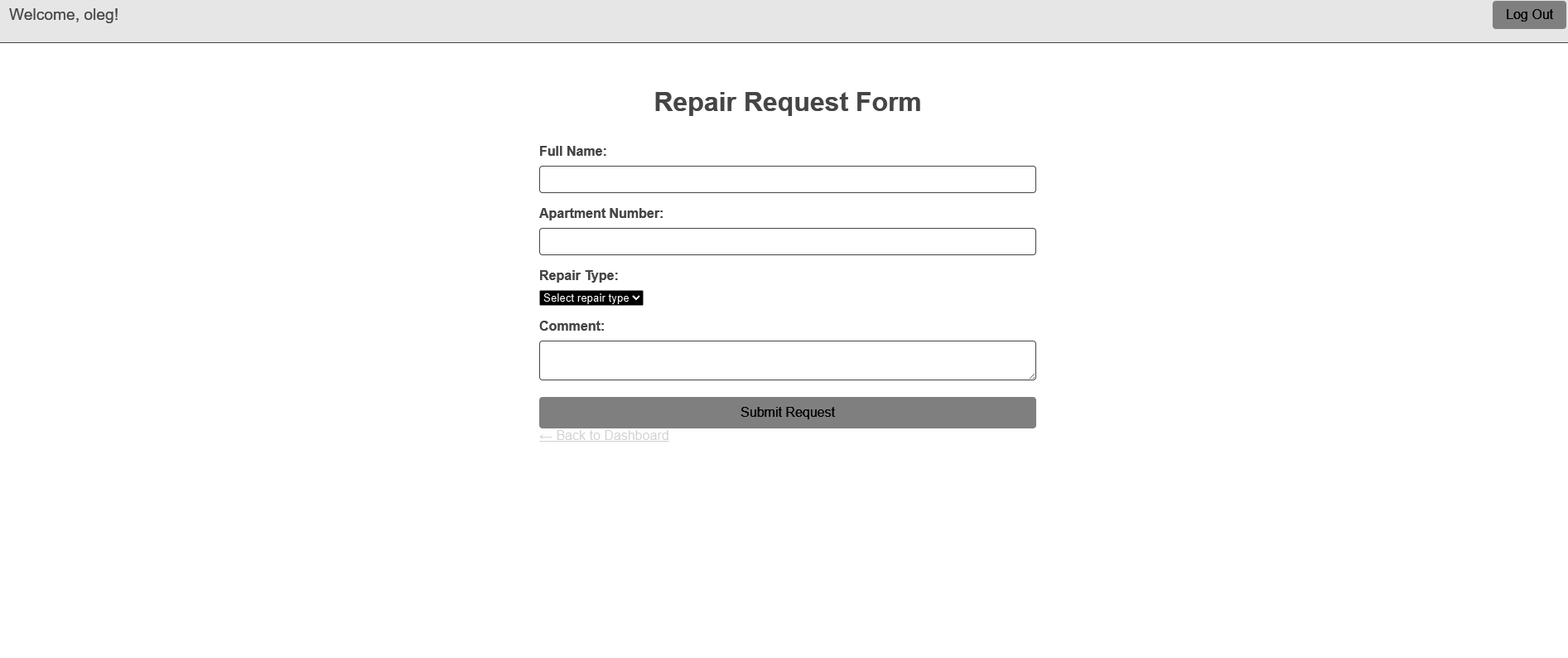
Поле для указания предполагаемого времени выполнения ремонта (например, дата и время или диапазон дат), что поможет планировать работы. Кнопка "Отправить заявку", которая при нажатии отправит данные на сервер. Визуальный дизайн и взаимодействие:

Использование современных HTML-элементов и CSS для создания привлекательного и удобного интерфейса. Валидация данных на стороне клиента для предотвращения ошибок (например, обязательные поля, правильный формат даты). После успешной отправки отображение подтверждения или сообщения об успешной регистрации заявки. Интеграция с серверной частью:

Обработка ответов сервера для отображения статуса заявки или ошибок. Дополнительные функции:

Просмотр ранее поданных заявок и их статуса (например, "в ожидании", "в работе", "завершена"). Уведомления о статусе заявки через систему уведомлений или сообщения. Безопасность и удобство: Защита формы от Аутентификация пользователя перед подачей заявки (использование Gorilla Sessions для управления сессиями).

Примерный макет интерфейса можно реализовать с помощью HTML-шаблонов в Go, используя встроенные HTML Templates, а стили — через CSS. Взаимодействие с сервером обеспечит обработку данных на стороне Go, сохранение в базу данных PostgreSQL и управление статусами заявок. Таким образом, реализованный пользовательский интерфейс обеспечит жильцам простой способ Для реализации пользовательского интерфейса, позволяющего жильцам регистрировать заявки на ремонт с указанием типа проблемы и времени выполнения, необходимо разработать удобную и интуитивно понятную веб-страницу. Этот интерфейс должен включать форму для ввода данных, которая будет содержать поля для выбора типа проблемы, описания ситуации, а также предполагаемого времени выполнения ремонта. В качестве элементов формы можно использовать выпадающие списки для выбора типа проблемы (например, электрика, сантехника, ремонт мебели), текстовые поля для описания ситуации и даты или срока выполнения. После заполнения формы пользователь сможет отправить заявку, которая будет передана на сервер через HTTP-запрос (например, POST). На серверной стороне данные будут валидироваться и сохраняться в таблицу repair\_requests базы данных PostgreSQL. В интерфейсе также желательно предусмотреть отображение статуса заявки (например, ожидает подтверждения, в процессе или выполнена), а также возможность просмотра истории своих заявок.



3.8.2 Бронирования общих помещений с учетом доступности и временных интервалов.

Для реализации пользовательского интерфейса по бронированию общих помещений с учетом доступности и временных интервалов необходимо было создать удобную и понятную форму, которая позволяла жильцам легко выбирать желаемое помещение и время для бронирования. Такой интерфейс должен был включать следующие элементы:

Выбор помещения:

Выбор даты и времени:

Поле для выбора даты (например, календарь).

Проверка доступности:

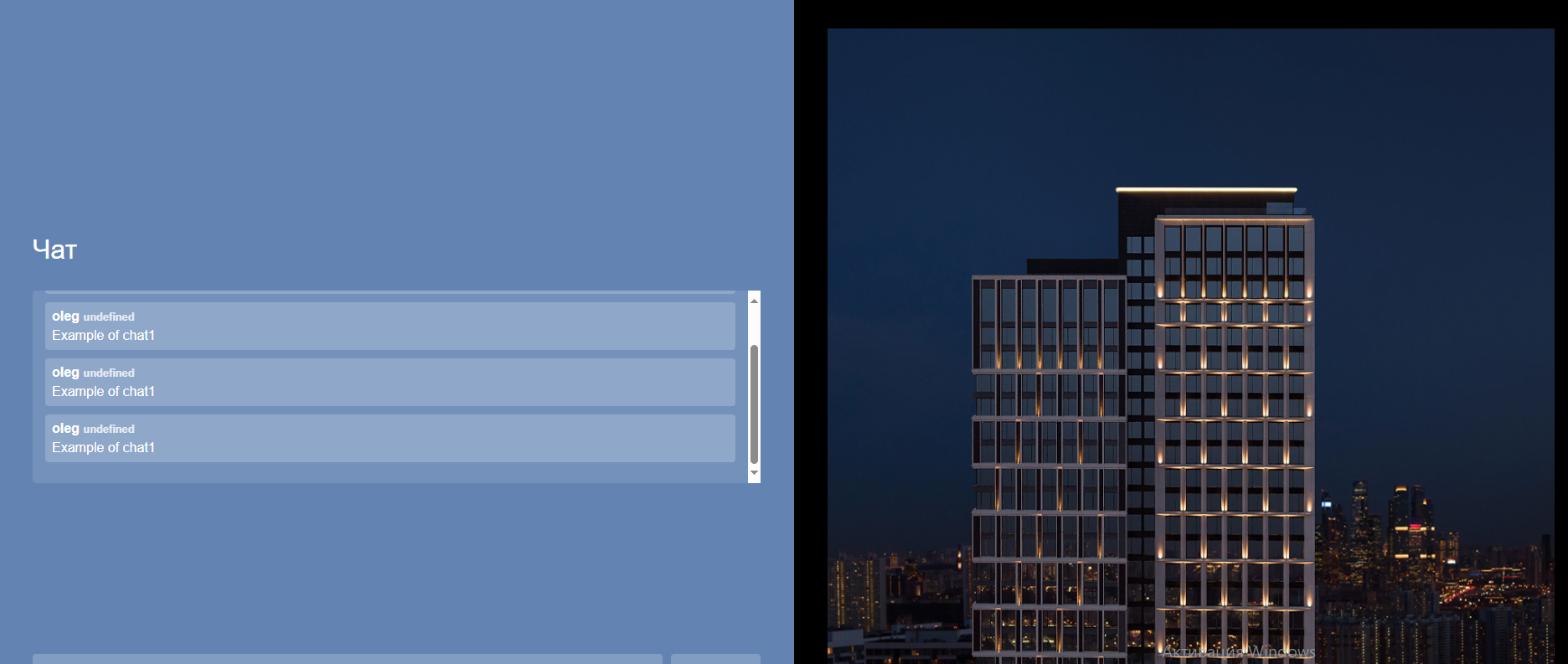
После выбора даты и времени система автоматически проверяла, свободно ли помещение в указанный интервал. В случае занятости — отображалось сообщение с предложением выбрать другой интервал или помещение.

Подтверждение бронирования: Кнопка "Забронировать", которая отправляла запрос на сервер. После успешного бронирования отображалось подтверждение и обновлялся список текущих бронирований. Просмотр текущих бронирований: Возможность просматривать список своих активных и прошлых бронирований с деталями (дата, время, помещение, статус). Обеспечение безопасности и удобства: Аутентификация пользователя перед бронированием (использование Gorilla Sessions). Валидация данных на стороне клиента и сервера для предотвращения ошибок. Обновление интерфейса в реальном времени при изменении выбранных параметров. Такой интерфейс обеспечивал жильцам возможность легко планировать использование общих помещений, избегая конфликтов по времени и повышая эффективность их использования. Реализовать его можно было с помощью HTML-шаблонов в Go, стилизовать — через CSS, а взаимодействие — через JavaScript для динамической проверки доступности и отправки данных на сервер.



3.8.2 Создания и участия в обсуждениях между жильцами.

Чат в проекте реализован с использованием WebSocket для обеспечения обмена сообщениями в реальном времени. Основная логика чата находится в нескольких ключевых компонентах. В файле handlers/chat.go определена структура для работы с WebSocket соединениями, где создается карта клиентов (clients) и канал для широковещательной рассылки сообщений (broadcast). При подключении нового клиента создается экземпляр структуры Client, который содержит WebSocket соединение, канал для отправки сообщений и функцию очистки. Для каждого клиента запускаются две горутины: ReadPump для чтения входящих сообщений и WritePump для отправки исходящих сообщений. Сообщения хранятся в базе данных PostgreSQL в таблице chat\_messages, которая содержит поля для идентификатора, имени пользователя, текста сообщения и времени создания. Когда пользователь отправляет сообщение через веб-интерфейс, оно сначала сохраняется в базе данных, а затем отправляется всем подключенным клиентам через канал broadcast. На клиентской стороне используется JavaScript для установки WebSocket соединения и обработки входящих сообщений. При получении нового сообщения оно динамически добавляется в DOM-дерево страницы. Администраторы имеют дополнительные возможности, такие как удаление сообщений, что реализовано через отдельный обработчик DeleteMessageHandler. Весь процесс обеспечивает мгновенную доставку сообщений всем участникам чата, сохраняя при этом историю сообщений в базе данных для последующего просмотра. Интерфейс чата реализован с использованием HTML и CSS, обеспечивая удобный и отзывчивый пользовательский интерфейс с возможностью прокрутки истории сообщений и отображением времени отправки каждого сообщения.



### 3.4. Внедрение системы уведомлений для пользователей о статусе заявок, бронирований и новых обсуждениях.

Система уведомлений в проекте реализована через таблицу notifications в базе данных PostgreSQL, которая хранит все уведомления для пользователей. Каждое уведомление содержит идентификатор, имя пользователя, текст сообщения, статус прочтения и время создания.

Уведомления генерируются автоматически при различных действиях в системе, таких как одобрение или отклонение заявок на ремонт и бронирование помещений. Когда администратор одобряет заявку на ремонт, система создает новое уведомление для пользователя с соответствующим сообщением, которое сохраняется в базе данных. Аналогично, при одобрении бронирования помещения также создается уведомление для пользователя. Уведомления отображаются в личном кабинете пользователя в специальном разделе, где они могут быть помечены как прочитанные.

Система использует структуру Notification в models.go для работы с уведомлениями, которая содержит все необходимые поля для отображения информации. В шаблоне userindex.html реализован интерфейс для отображения уведомлений, где каждое уведомление показывается с текстом сообщения и временем создания. Непрочитанные уведомления выделяются специальным стилем для привлечения внимания пользователя. Система также поддерживает автоматическое обновление списка уведомлений при появлении новых, что обеспечивает своевременное информирование пользователей о статусе их заявок и бронирований.

Все уведомления сохраняются в базе данных, что позволяет пользователям просматривать историю уведомлений даже после закрытия сессии. Система уведомлений интегрирована с общей архитектурой приложения и использует те же механизмы аутентификации и авторизации, что обеспечивает безопасность и конфиденциальность информации.

### 3.5. Настройка ролей пользователей: жильцы, администраторы, сотрудники управляющей компании.

Система ролей пользователей в проекте реализована через простую, но эффективную механику, основанную на проверке имени пользователя в базе данных. В таблице users хранятся учетные записи всех пользователей, включая их имена и хешированные пароли. Роль пользователя определяется по его имени пользователя (username), где специальное имя "admin" зарезервировано для администратора системы. При авторизации пользователя в системе (в handlers/auth.go) происходит проверка имени пользователя, и если оно соответствует "admin", пользователь перенаправляется на административную панель, в противном случае - на обычный интерфейс пользователя. Администраторы имеют расширенные права доступа, которые реализованы через проверки в различных обработчиках. Например, в обработчиках для управления заявками на ремонт и бронированиями (handlers/repair.go и handlers/booking.go) есть проверки, которые позволяют выполнять определенные действия только администраторам. Для обычных пользователей (жильцов) доступны базовые функции, такие как создание заявок на ремонт, бронирование помещений и использование чата. Администраторы, помимо этих функций, могут одобрять или отклонять заявки, управлять бронированиями и удалять сообщения в чате. Система использует сессии для хранения информации о текущем пользователе, что позволяет сохранять его роль на протяжении всей сессии. При попытке доступа к административным функциям всегда происходит проверка роли пользователя, и если пользователь не является администратором, ему отказывается в доступе. Такая реализация ролей обеспечивает четкое разделение прав доступа между обычными пользователями и администраторами, при этом сохраняя простоту и эффективность системы.

3.6 Обеспечение безопасности данных: защита персональной информации жильцов и заявок

Безопасность данных в проекте реализована через комплексный подход, включающий несколько уровней защиты.

Основной механизм защиты начинается с аутентификации пользователей, где пароли хранятся в базе данных в хешированном виде с использованием алгоритма bcrypt, что обеспечивает защиту от утечек паролей даже в случае компрометации базы данных.

Для управления сессиями пользователей используется пакет gorilla/sessions, который обеспечивает безопасное хранение данных сессии в зашифрованных cookies.

Все HTTP-запросы проходят через проверку авторизации, где проверяется наличие валидной сессии и соответствующих прав доступа.

При работе с базой данных используются параметризованные SQL-запросы, что предотвращает SQL-инъекции. Персональные данные пользователей, такие как номера телефонов и адреса квартир, хранятся в базе данных с ограниченным доступом, где только авторизованные пользователи могут видеть свои собственные данные.

Администраторы имеют доступ к расширенной информации, но их действия логируются и требуют дополнительной аутентификации.

Для защиты от CSRF-атак используется проверка origin в WebSocket соединениях и валидация всех входящих запросов.

Все формы на сайте защищены от XSS-атак через экранирование вывода в HTML-шаблонах.

Система также реализует защиту от брутфорс-атак через ограничение количества попыток входа и временные задержки при неудачных попытках. Для обеспечения целостности данных все критические операции, такие как одобрение заявок или бронирований, выполняются в рамках транзакций базы данных. Логирование важных действий в системе позволяет отслеживать подозрительную активность и расследовать инциденты безопасности. Все статические файлы и ресурсы защищены от несанкционированного доступа через настройки веб-сервера. Система также использует безопасные заголовки HTTP для предотвращения различных типов атак, таких как clickjacking и MIME-sniffing.

Регулярное обновление зависимостей и использование актуальных версий библиотек обеспечивает защиту от известных уязвимостей. Для дополнительной защиты чувствительных данных используется шифрование при передаче.

Система также реализует политику минимальных привилегий, где каждый пользователь имеет доступ только к тем данным и функциям, которые необходимы для его роли.

**3.7 Тестирование системы, проверка функциональности, удобства интерфейса и производительности.**

Тестирование системы проводилось по нескольким ключевым направлениям, чтобы обеспечить её корректную работу, удобство использования и стабильность при высокой нагрузке. Функциональное тестирование включало проверку всех основных модулей системы: регистрации и авторизации пользователей, создания и обработки заявок на ремонт, бронирования помещений, работы чата и системы уведомлений. Для каждого модуля были разработаны тестовые сценарии, охватывающие как успешные сценарии использования, так и обработку ошибочных ситуаций. В частности, проверялась корректность заполнения форм заявок (выбор типа ремонта, описание, прикрепление файлов), успешная отправка заявок и отображение их в списке пользователя, а также правильное обновление статусов (новая, в процессе, выполнена). Аналогично тестировались функции бронирования помещений: выбор помещения, даты и времени с автоматической проверкой доступности через запросы; успешное бронирование при свободных интервалах и отображение ошибок при занятости. В системе обсуждений жильцов проверялась возможность создания новых тем, добавления комментариев и обновления сообщений в реальном времени через WebSocket. Также особое внимание уделялось авторизации и безопасности — входу/выходу пользователей, защите данных от несанкционированного доступа. Тестирование интерфейса проводилось с привлечением группы тестировщиков — жильцов с разным уровнем технической подготовки. Оценивалась логика навигации, читаемость интерфейса, понятность форм и сообщений об ошибках. Особое внимание уделялось адаптивности интерфейса на различных устройствах и браузерах. В результате были выявлены и устранены мелкие недочёты: упрощены формы, добавлены подсказки и инструкции для повышения удобства пользователей. Проверялись время отклика страниц при одновременной работе множества пользователей — особенно в чатах и при бронировании помещений. Результаты показали стабильную работу , с временем отклика не превышающим 2 секунд. Также проводились тесты на восстановление после сбоев и работу при потере соединения. Безопасность системы проверялась через тестирование на проникновение: выявлялись уязвимости к SQL-инъекциям, XSS-атакам. Тестировалась целостность данных в базе данных, эффективность индексов и корректность связей между таблицами. Общий итог: система успешно справляется с основными задачами — регистрацией заявок на ремонт, бронированием общих помещений, обсуждениями жильцов — обеспечивает удобный интерфейс для пользователей и администраторов. Тестирование подтвердило её стабильную работу под нагрузкой и высокий уровень безопасности. В ходе проведения были выявлены отдельные области для дальнейшей оптимизации интерфейса и ускорения обработки запросов.

### 3.8. Разработка эксплуатационной документации по сопровождению и обновлению жк-систем

1. Введение

* Цель документации: данное руководство предназначено для пользователей и администраторов жк-системы, разработанной в рамках проекта. Оно содержит инструкции по эксплуатации, сопровождению и обновлению системы, а также рекомендации по устранению возможных проблем.
* Описание системы: жк-система представляет собой веб-приложение, предназначенное для работы с данными, включая регистрацию и вход пользователей, регистрация заявок на ремонт квартир, регистрация заявок на бронирование помещений в холле.
* Целевая аудитория: руководство предназначено для жителей жк , а также для администраторов, отвечающих за этот комплекс.
* Требования к знаниям и навыкам: для работы с системой пользователям необходимы базовые навыки работы с браузером. Администраторам требуются знания в области работы с базами данных PostgreSQL.
* Контакты: в случае возникновения вопросов или проблем, пользователи могут обратиться в службу поддержки по электронной почте: kostikovaleksej20[@gmail.ru](mailto:kulik1diana@mail.ru).

2. Начало работы с системой

2.1. Системные требования:

* + - Операционная система: Android 8.0 или Windows Xp, 1Гб ОЗУ, 100 Гб ССД
    - Подключение к интернету для работы

2.2. Начало использования для пользователей:

* Для ПК и телефонов
  + Запустите браузер:
  + Найдите иконку вашего браузера (например, Google Chrome, Mozilla Firefox, Microsoft Edge или Safari) на рабочем столе или в меню "Пуск" и дважды щелкните по ней.
  + Введите адрес сайта:
    - В верхней части окна браузера вы увидите адресную строку. Кликните на нее, чтобы выделить текст.  
      Введите URL-адрес сайта, который вы хотите посетить (например, www.example.com), и нажмите клавишу Enter на клавиатуре.
  + Ожидайте загрузки страницы:
  + Браузер начинает загружать сайт. Это может занять несколько секунд в зависимости от скорости вашего интернет-соединения.
  + Навигация по сайту:
    - После загрузки страницы вы можете просматривать контент сайта, кликать по ссылкам и использовать меню для навигации.

2.3. Авторизация и вход в систему:

Для входа в систему пользователь должен ввести логин и пароль, зарегистрированные в базе данных, если пользователь не зарегистрирован, нужно нажать на кнопку регистрации и написать свои данные, после чего пользователя перекинет на страницу входа

2.4. Описание интерфейса пользователя:

* + Главный экран: отображает чат, и 2 кнопки для бронирования помещения, и регистрации заявки на ремонт, соответственно.
  + Панель сверху: данные об аккаунте и кнопка выйти

3. Основные операции

* 3.1. Регистрация заявки на ремонт:
  + Введите свое имя.
  + Введите номер квартиры.
  + Введите тип ремонта.
  + Введите комментарий
  + Система передаст данные в систему-жк и с пользователем свяжутся по запросу.
* 3.2. Бронирование помещений:
  + - Введите свое имя.
    - Введите номер телефона.
    - Введите Номер комнаты.
    - Введите день на который нужно забронировать помещение(в пределе 3 дней вперед)
    - Система передаст данные в систему-жк и с пользователем свяжутся по запросу.
* Выход из учетной записи:
  + - сверху страницы нажмите “log out”
  + Отправка сообщений в чат
    - введите сообщение в поле
    - нажмите на кнопку “send”

2.3Начало использования для администратора:

* Использование чата:
  + введите сообщение в поле
  + нажмите на кнопку “send”
  + если вы видите неподобающее сообщение, его нужно удалить нажав на кнопку “delete”
* Просмотр заявок на бронирование помещения
  + нажмите на кнопку просмотра заявок
  + вы попадете на страницу с визуализацией базы данных
  + в ней можно отклонять и подтверждать заявки на бронирование. Только после подтверждения пользователем по звонку, который пользователь оставил в форме
* Просмотр заявок на ремонт
  + нажмите на кнопку просмотра заявок
  + вы попали на страницу с визуализацией базы данных
  + в ней можно отклонять заявки на ремонт если житель не подтвердил документы в личном порядке

4. Сопровождение и обновление системы

* 4.1. Обновление данных:
  + Система автоматически добавляет и изменяет данные о заявках. Администратор может также менять информацию о заявках вручную
* 4.2. Обновление:
  + При изменении функционала обновление не потребуется.
* 4.3. Резервное копирование данных:
  + Администраторы могут выполнять резервное копирование базы данных PostgreSQL через встроенные инструменты управления базами данных.
  + Рекомендуется выполнять резервное копирование перед обновлением системы.

5. Решение проблем и ответы на часто задаваемые вопросы

* 5.1. Распространенные ошибки и способы их устранения:
  + Ошибка подключения к интернету: проверьте подключение к сети и перезагрузите приложение.
  + Неверный логин или пароль: убедитесь, что данные введены корректно.
* 5.2. Обращение в службу поддержки:

В случае возникновения проблем, которые не удается решить самостоятельно, обратитесь в службу поддержки, предоставив следующую информацию:

* + Описание проблемы.
  + Скриншоты ошибки (если возможно).

Эксплуатационная документация обеспечивает пользователей и администраторов всеми необходимыми инструкциями для работы с жк системой. Регулярное обновление документации и тестирование новых функций позволяют поддерживать высокий уровень удобства и надежности системы.

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была разработана система жк, направленная на обеспечение эффективного управления информацией в жк-системе, поддержание ее актуальности и повышение производительности системы. В процессе выполнения работы были решены поставленные задачи, включая анализ современных платформ и инструментов для сопровождения жк-систем, разработку механизмов передачи и получения данных, интеграцию системы мониторинга и оптимизацию работы базы данных.

Проведен анализ современных платформ и инструментов для сопровождения жк-систем, таких как PostgreSQL. Были изучены методы передачи данных, включая интеграцию с внешними сервисами и использование библиотек "sessions" и "websocket". Разработана система сопровождения и передачи данных, включающая механизмы получения информации, мониторинга производительности и интеграции с внешними сервисами. В качестве основной платформы для хранения данных использовалась PostgreSQL. Проведена оптимизация работы базы данных, что позволило сократить время отклика системы и улучшить пользовательский опыт. Были реализованы механизмы индексации и нормализации структуры базы данных. Проведено тестирование системы на производительность и корректность передачи данных. Результаты тестирования показали, что система успешно справляется с нагрузкой и обеспечивает актуальность информации. Разработана эксплуатационная документация, включающая инструкции по установке, настройке, использованию и сопровождению системы. Документация обеспечивает пользователей и администраторов всеми необходимыми инструкциями для работы с жк-системой.

Практическая значимость работы заключается в том, что разработанная система сопровождения и передачи данных может быть использована в компаниях, работающих с информацией о жилых комплексах, для повышения эффективности управления данными, обеспечения их актуальности и улучшения пользовательского опыта. В перспективе система может быть дополнена модулями предсказательной аналитики на основе машинного обучения и интеграцией с облачными технологиями для масштабируемого хранения данных.

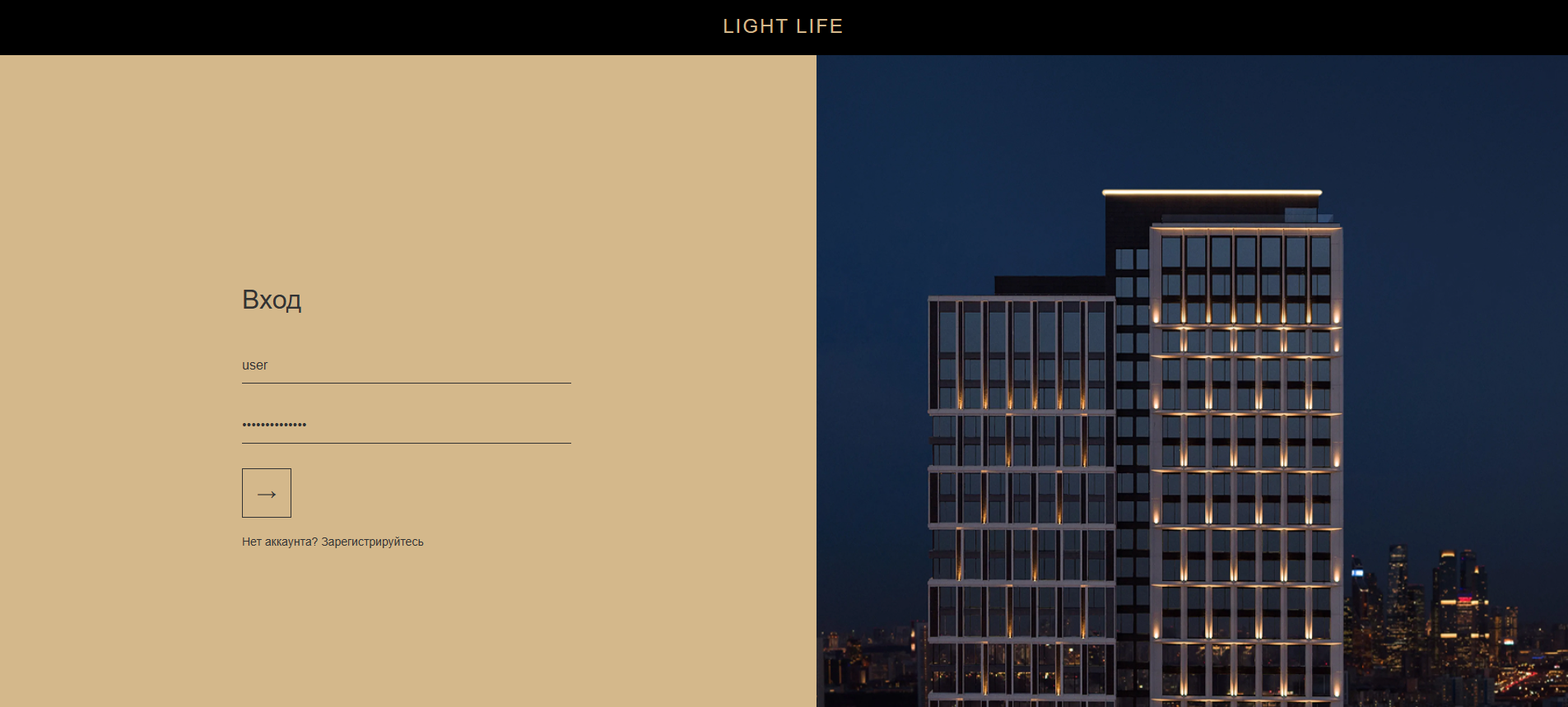
В заключение можно отметить, что выполнение данной курсовой работы позволило углубить теоретические знания в области сопровождения и передачи информации в жк-системах, риобрести практические навыки разработки и оптимизации сложных систем. Разработанная система демонстрирует высокую эффективность и может быть успешно применена в реальных условиях для решения задач, связанных с управлением информацией о жилых комплексах.

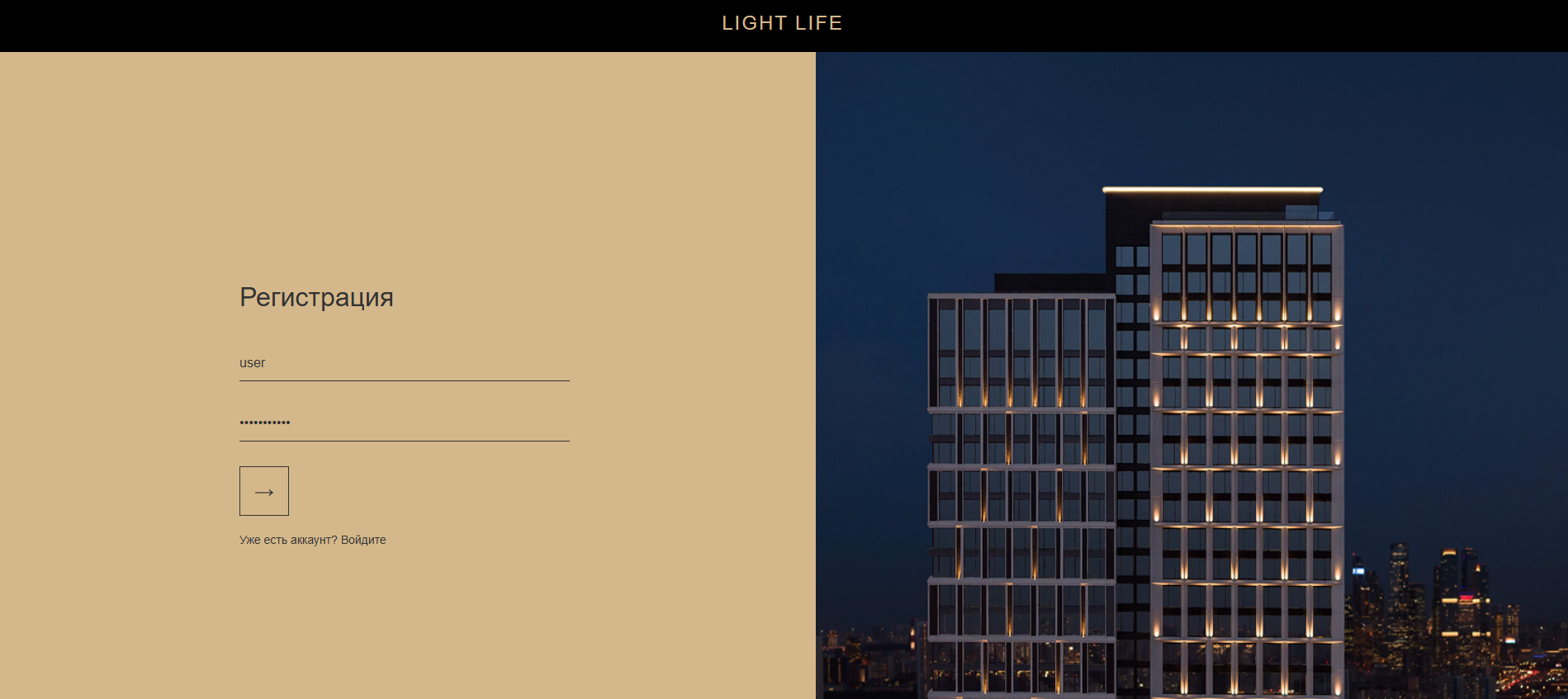
# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

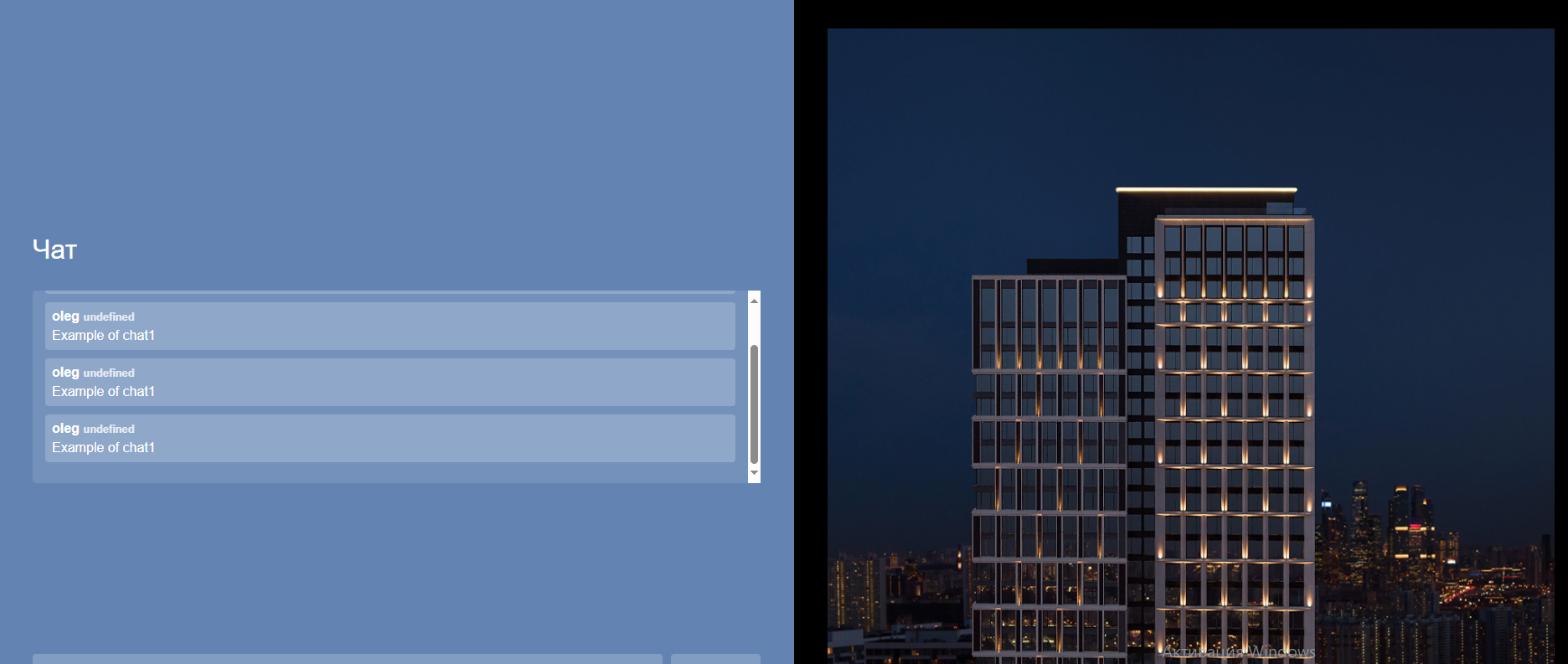
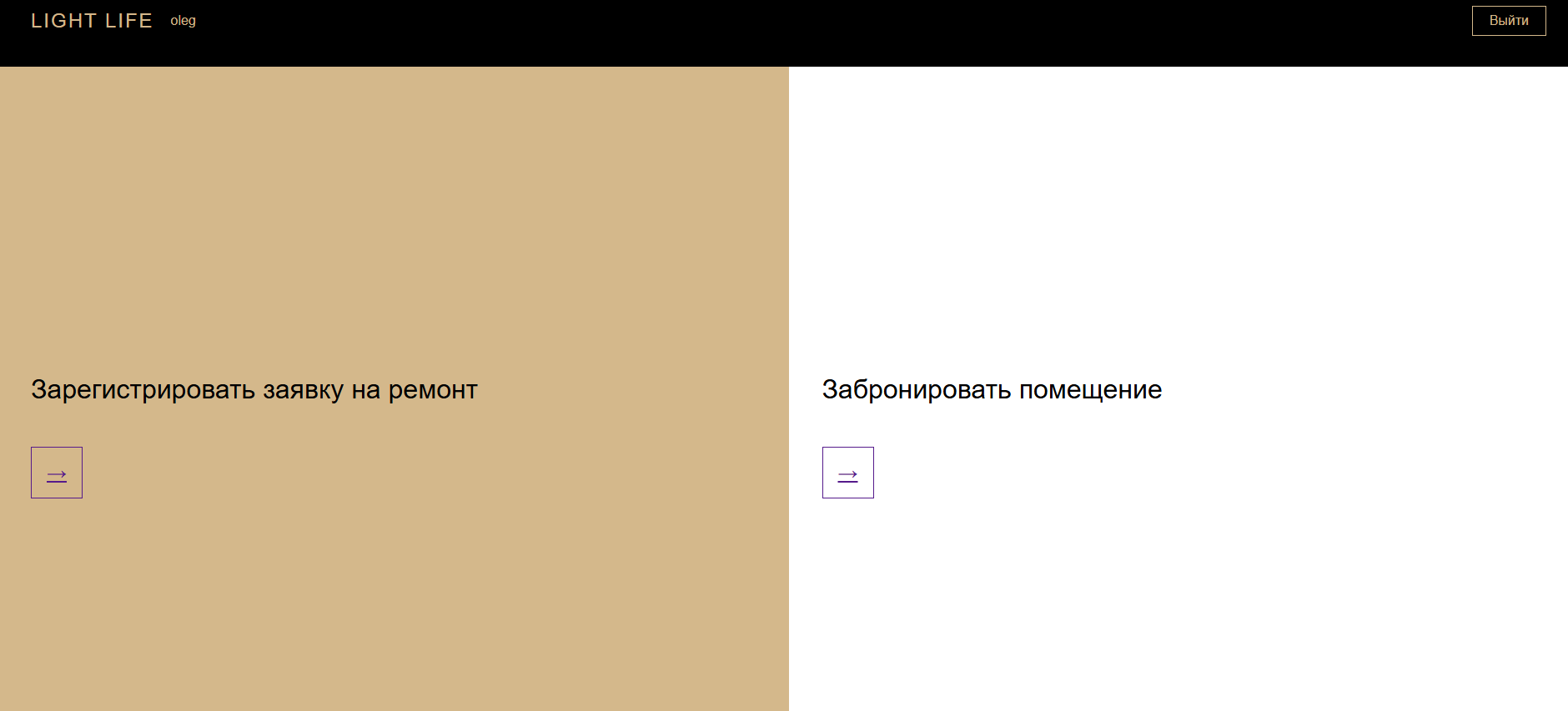
1. Нормативные документы:
   1. Федеральный закон Российской Федерации от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации»
   2. Федеральный закон Российской Федерации от 27 июля 2006 г. № 152-ФЗ «О персональных данных»Основные источники:
   3. Джей макгаврен, Head First, Изучаем Go
   4. Go: идиомы и паттерны проектирования, Джон Боднер
2. Дополнительные источники:
   1. Д. Куроуз, К. Росс "Компьютерные сети. Нисходящий подход"
3. Электронные ресурсы:
   1. Официальный сайт PostgreSQL.<https://www.postgresql.org/>.
4. Техническая документация:
   1. Руководство пользователя PostgreSQL. – URL: <https://www.postgresql.org/docs/>.
5. Методическая литература:
   1. Методические рекомендации по выполнению и оформлению курсовой работы ПМ.06 Сопровождение информационных систем. – М.: ГБПОУ КМБ № 4, 2025. – 28 с.

# ПРИЛОЖЕНИЕ

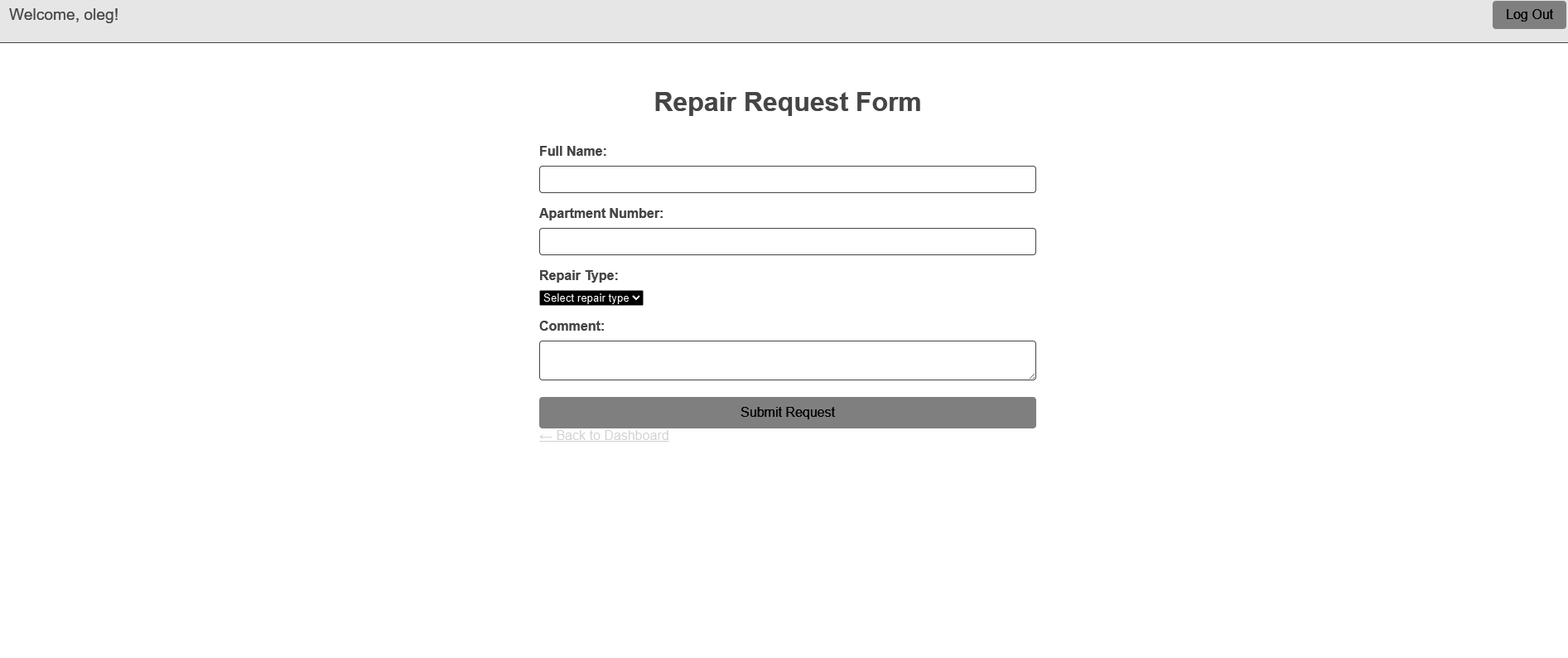
Приложение 1. Страница регистрации и входа





Приложение 2. Главная страница для пользователя

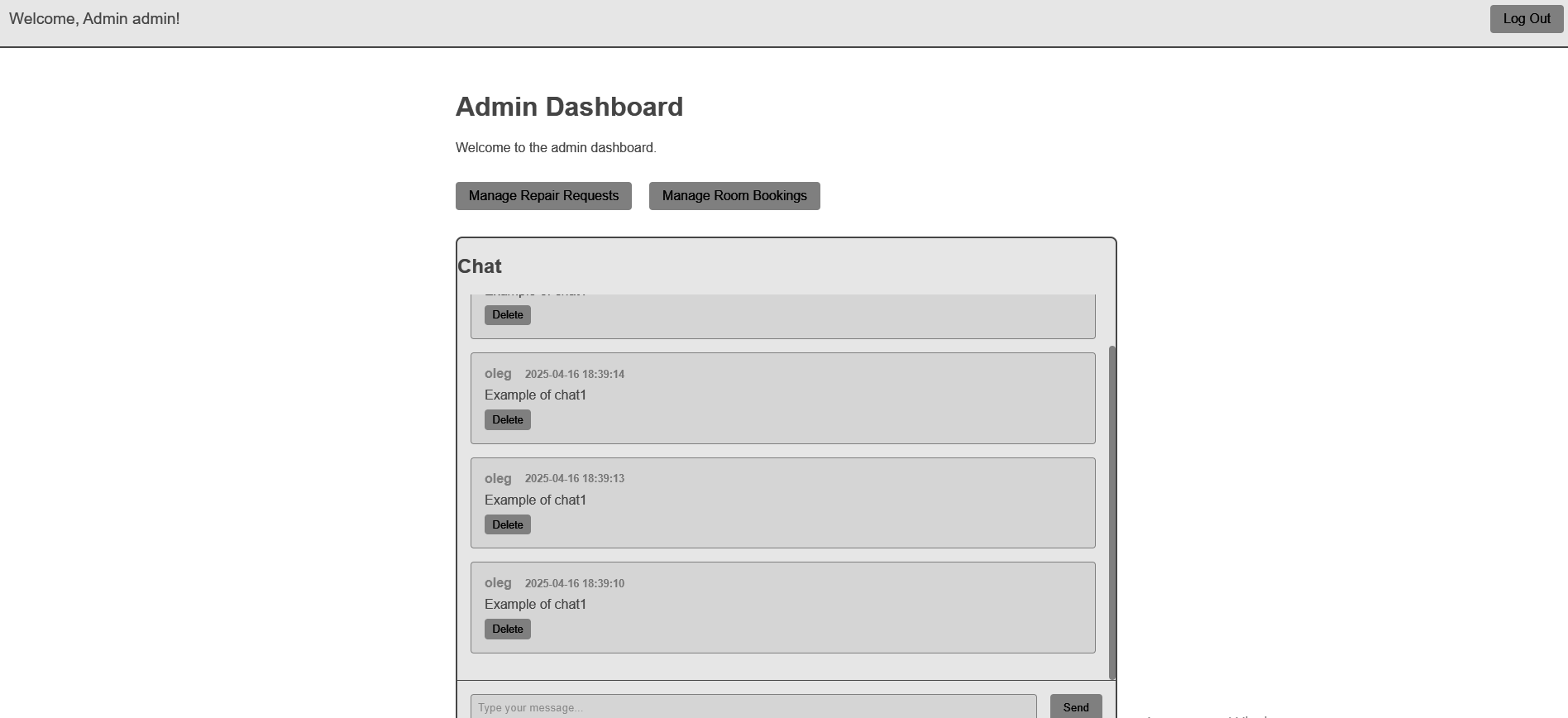
Приложение 3. Регистрация заявок на ремонт



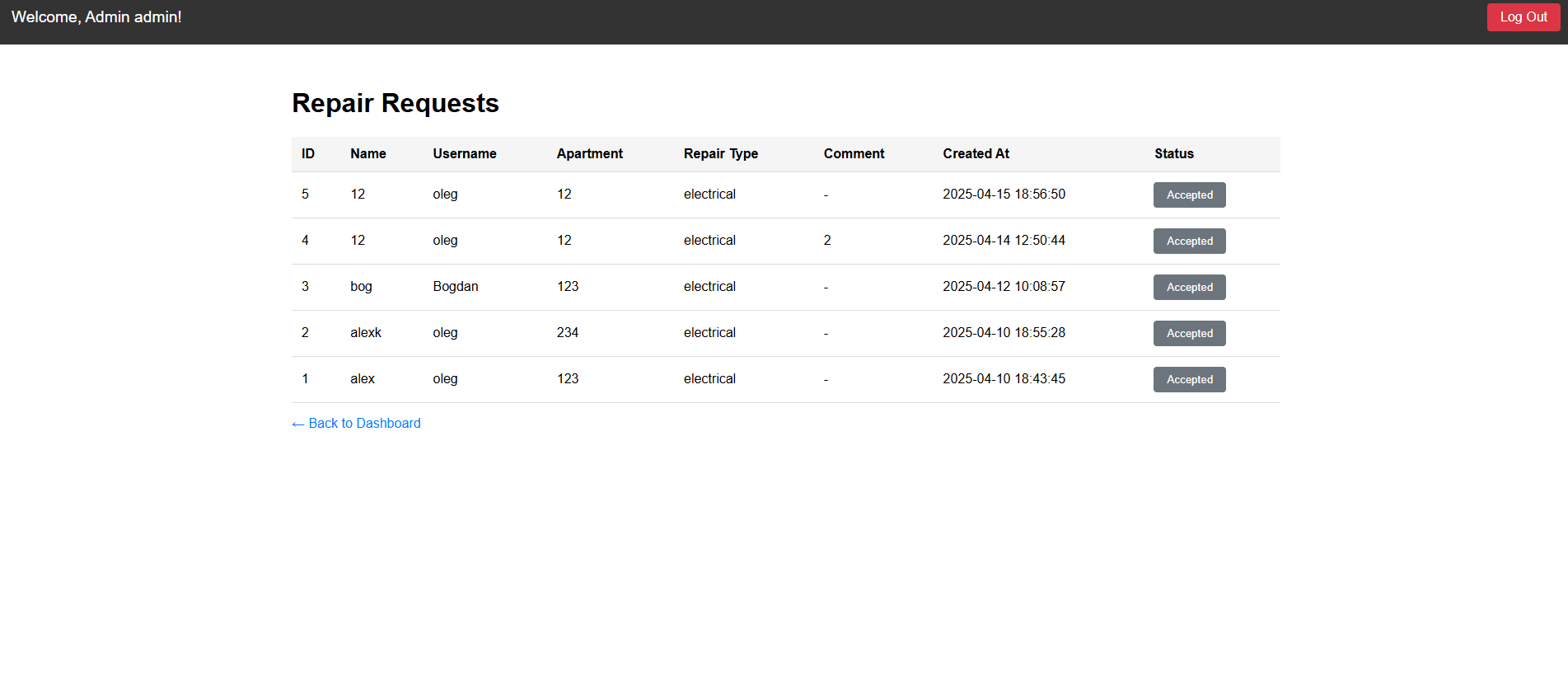
Приложение 4. Регистрация заявок на бронирование помещения



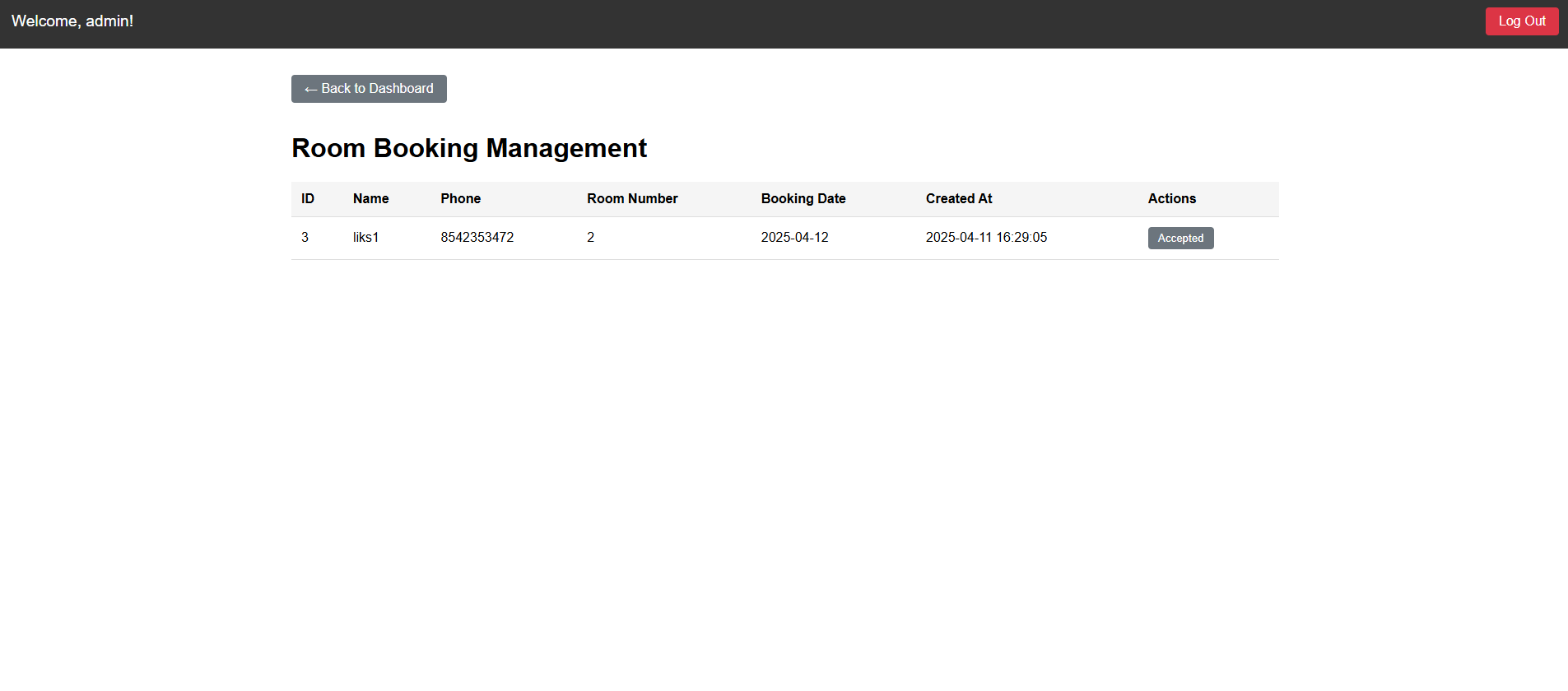
Приложение 5.Главный экран для админа



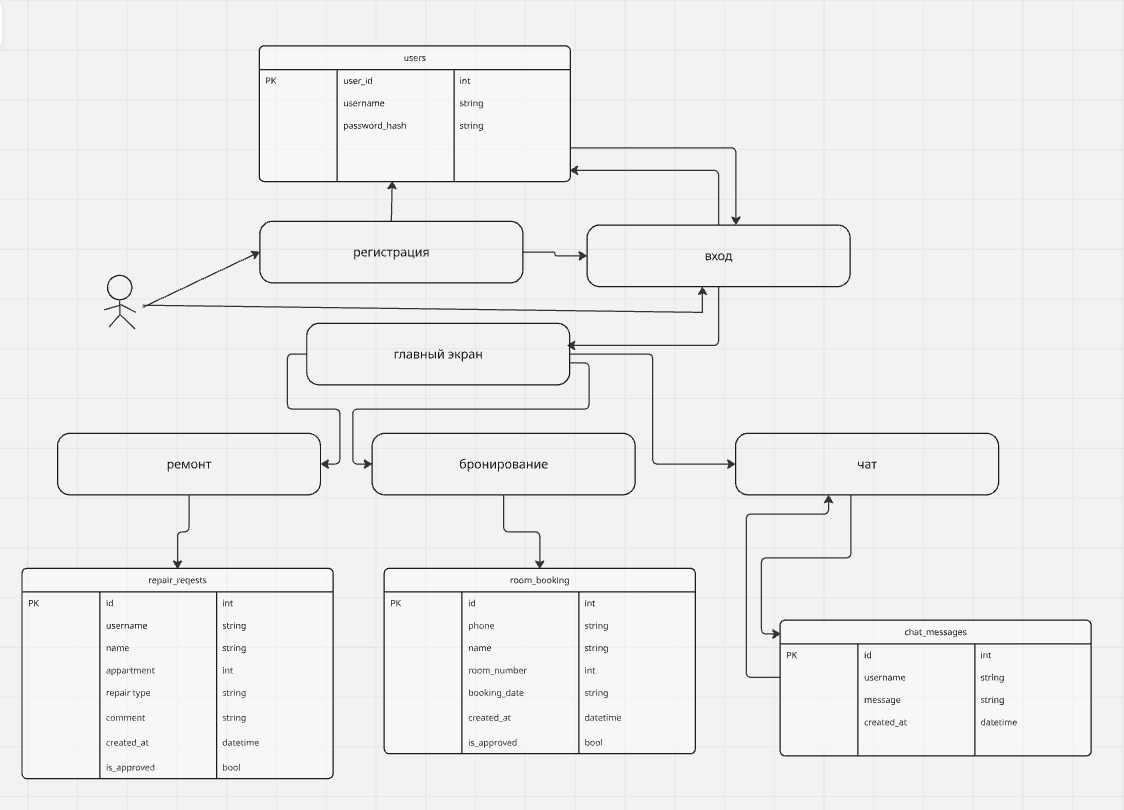
Приложение 6. Экран бронирования помещений для админа



Приложение 7. Экран регистрация заявок на ремонт



Приложение 8. Диаграмма потоков данных:



Приложение 9. Er-диаграмма бд

