**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им. Э.Т. Кренкеля**

Допустить к защите

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_ Н.В. Калинина

14 июня 2023 г.

|  |  |  |  |
| --- | --- | --- | --- |
| **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА** | | | |
| **Разработка автоматизированной информационной** | | | |
| **системы сервисного ИТ-центра** | | | |
|  | | | |
|  | *(тема ВКР)* | | |
| Вид выпускной квалификационной работы | | | *Дипломный проект* |
|  | | | *(дипломный проект, дипломная работа)* |
|  | | | |
| Специальность | | 09.02.03 Программирование в компьютерных системах | |
|  | | (код и наименование специальности) | |
| Квалификация | | Техник-программист | |
|  | | (наименование квалификации в соответствии с ФГОС СПО) | |

|  |  |  |
| --- | --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Е.А. Самойлов |
|  | (подпись) | 10 июня 2023 г. |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | И.В. Карелина |
|  | (подпись) | 10 июня 2023 г. |

Санкт-Петербург

2023

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Работа написана мною самостоятельно и не содержит неправомерных заимствований. Работа может быть размещена в электронно-библиотечной системе университета) | | | | | | | |
| 10.06.2023 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Е.А. Самойлов | | | |
| *(дата)* | | *(подпись студента)* | | *(ИОФ студента)* | | | |
|  | |  | |  | | | |
| Текст ВКР размещен в электронно-библиотечной системе университета | | | | | | | |
|  | | | | | | | |
| Руководитель отдела комплектования библиотеки | | | | | |  | |
| 10.06.2023 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | Д.В. Маслов | | |
| *(дата)* | | *(подпись)* | | | *(ИОФ)* | | |
|  | |  | | |  | | |
| Коэффициент оригинальности ВКР | | | \_\_\_\_\_\_\_ % | | | |  |
|  | |  | | | | | |
| Проверил: | Председатель цикловой комиссии № 5 Н.В. Кривоносова | | | | | | |
| *(Должность, ИОФ)* | | | | | | | |
| 10.06.2023 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |
| *(дата)* | | *(подпись)* | | | | | |

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им. Э.Т. Кренкеля**

УТВЕРЖДАЮ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Калинина

14 марта 2023 г.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ЗАДАНИЕ**  **на выполнение выпускной квалификационной работы (ВКР)** | | | | |
|  | | | | |
| 1. Студент | *Самойлов Евгений Александрович* | | № группа | *К501* |
|  | *(фамилия, имя, отчество)* | |  |  |
| 2. Руководитель | *Карелина Ирина Владимировна, преподаватель СПБГУТ* | | | |
|  | *(фамилия, имя, отчество, должность, уч. степень и звание)* | | | |
| 3. Квалификация | *Техник-программист* | | | |
|  | *(наименование квалификации в соответствии с ФГОС СПО)* | | | |
| 4. Вид ВКР | *Дипломный проект* | | | |
|  | *( дипломный проект, дипломная работа)* | | | |
| 5. Тема ВКР | *Разработка автоматизированной информационной* | | | |
| *системы сервисного ИТ-центра* | | | | |
|  | | | | |
| утверждена приказом ректора университета от 20 марта 2023 г. № 341/кс | | | | |
|  | | | | |
| 6. Исходные данные (технические требования): | |  | | |
| *Техническое задание на разработку, документация отдела кадров,* | | | | |
| *Методология управления проектами ГОСТ 7.32 – 2001* | | | | |
|  | | | | |
| 7. Содержание работы (анализ состояния проблемы, проведение исследований, разработка, расчеты параметров, экономическое обоснование и др.) | | | | |
| *Постановка задачи, Проектирование программных модулей, Разработка* | | | | |
| *программного продукта, Тестирование программного обеспечения, Информационная* | | | | |
| *безопасность, Техническая документация, Мероприятия по технике безопасности и* | | | | |
| *охране труда, Экономическое обоснование проекта, Заключение.* | | | | |
|  | | | | |
| 8. Вид отчетных материалов, представляемых в ГЭК (пояснительная записка, перечень, графического материала, отчет о НИР, технический проект, образцы и др.) | | | | |
| *Пояснительная записка, презентация, технологическая документация, экономическое* | | | | |
| *обоснование проекта.* | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9. Консультанты по ВКР с указанием относящихся к ним разделов | | | |
|  | | | |
| Раздел | Консультант  (фамилия имя отчество) | Подпись дата | |
| Задание  выдал | Задание  принял |
| 1. Постановка задачи | Карелина И.В. | 20.04.2023 | 20.04.2023 |
| 2. Проектирование программных модулей | Карелина И.В. | 25.04.2023 | 25.04.2023 |
| 3. Разработка программного продукта | Карелина И.В. | 10.05.2023 | 10.05.2023 |
| 4. Тестирование программного обеспечения | Карелина И.В. | 15.05.2023 | 15.05.2023 |
| 5. Информационная безопасность | Карелина И.В. | 20.05.2023 | 20.05.2023 |
| 6. Техническая документация на программный продукт | Карелина И.В. | 30.05.2023 | 30.05.2023 |
| 7. Мероприятия по технике безопасности и охране труда | Ожерельева Н.А. | 01.06.2023 | 01.06.2023 |
| 8. Экономическое обоснование проекта | Корина М.Н | 02.06.2023 | 02.06.2023 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  | | | | |
| Дата выдачи задания | | 20 апреля 2023 г. | | | | |
|  | |  | | | | |
| Дата представления ВКР к защите | | 10 июня 2023 г. | | | | |
|  | |  | | | | |
| Председатель предметной (цикловой) комиссии | | |  | | | |
|  | | |  | | |  |
|  | | | | | | |
|  | | |  | | | Н.В. Кривоносова |
|  | | | *(подпись)* | | *(И.О. Фамилия)* | |
| Руководитель ВКР |  | | |  | | |
|  | *(подпись)* | | | *(И.О. Фамилия)* | | |
| Студент |  | | |  | | |
|  | *(подпись)* | | | *(И.О. Фамилия)* | | |

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им. Э.Т. Кренкеля**

**Календарный план выполнения ВКР**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов выпускной квалификационной работы (ВКР) | Срок выполнения этапов ВКР | Примечание |
| 1. | Постановка цели выполнения ВКР и задач | *20.04.2023* |  |
| 2. | Работа с теоретическим материалом | *21.04.2023-17.05.2023* |  |
| 3. | Сбор информации, необходимой для написания работы |
| 4. | Систематизация и обработка материалов ВКР |
| 5. | Анализ полученных в работе результатов, обобщение |
| 6. | Подготовка отчетных материалов, представляемых в государственную экзаменационную комиссию, доклада к защите и презентации | *18.05.2023-29.06.2023* |  |
| 7. | Консультации с руководителем ВКР |
| 8. | Представление выполненной ВКР руководителю для подготовки отзыва |
| 9. | Подготовка к защите ВКР, включая подготовку к процедуре защиты и процедуру защиты |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Председатель предметной (цикловой) комиссии | |  | | | |
|  | | | | | |
|  | |  | | | Н.В. Кривоносова |
|  | | *(подпись)* | | *(И.О. Фамилия)* | |
| Руководитель ВКР |  | |  | | |
|  | *(подпись)* | | *(И.О. Фамилия)* | | |
| Студент |  | |  | | |
|  | *(подпись)* | | *(И.О. Фамилия)* | | |

**Оглавление**

[ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ: 8](#_Toc137603223)

[ВВЕДЕНИЕ 9](#_Toc137603224)

[1. ПОСТАНОВКА ЗАДАЧИ 10](#_Toc137603225)

[**1.1. Обзор аналогов** 10](#_Toc137603226)

[**1.2 SWOT-анализ** 13](#_Toc137603227)

[**1.3 PEST-анализ** 14](#_Toc137603228)

[**1.4 Формирование требований к программному продукту** 14](#_Toc137603229)

[**1.4.1 Бизнес-требования** 14](#_Toc137603230)

[**1.4.2 Пользовательские требования** 15](#_Toc137603231)

[**1.4.3 Функциональные требования к программному продукту** 15](#_Toc137603232)

[**1.4.4 Нефункциональные требования** 16](#_Toc137603233)

[**1.4.5 Ограничения** 16](#_Toc137603234)

[**1.4.6 Требования к интерфейсам** 17](#_Toc137603235)

[**1.4.7 Требования к данным** 17](#_Toc137603236)

[**1.5 Программные средства разработки** 19](#_Toc137603237)

[**1.6 Аппаратные средства разработки** 20](#_Toc137603238)

[2. ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ 21](#_Toc137603239)

[**2.1. Архитектура системы** 21](#_Toc137603240)

[**2.2. Моделирование основных сценариев системы** 21](#_Toc137603241)

[**2.3. Проектирование графического интерфейса пользователя** 32](#_Toc137603242)

[**2.4. Проектирование и разработка модели данных** 36](#_Toc137603243)

[3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 43](#_Toc137603244)

[**3. 1. Порядок разработки программных модулей** 43](#_Toc137603245)

[**3. 2. Структурное программирование** 44](#_Toc137603246)

[**3. 3. Реализация серверной части** 45](#_Toc137603247)

[**3. 4. Организация взаимодействия с БД** 46](#_Toc137603248)

[4. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 49](#_Toc137603249)

[**4.1. История изменений** 49](#_Toc137603250)

[**4.2. Терминология** 50](#_Toc137603251)

[**4.3. Стратегия тестирования** 50](#_Toc137603252)

[**4.4. Определение объектов тестирования** 50](#_Toc137603253)

[**4.5. Архитектура тестируемой системы** 52](#_Toc137603254)

[**4.6. Описание процесса тестирования** 53](#_Toc137603255)

[5. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ 55](#_Toc137603256)

[**5.1. Определение защищаемых информационных активов** 55](#_Toc137603257)

[**5.2. Вид и содержание персональных данных** 55](#_Toc137603258)

[**5.3. Модель угроз ПНД** 55](#_Toc137603259)

[**5.4. Возможные сценарии угроз** 55](#_Toc137603260)

[**5.5. Организация защиты данных в проекте** 55](#_Toc137603261)

[**5.6. Защита программного продукта** 55](#_Toc137603262)

[6. ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ НА ПРОГРАММНЫЙ ПРОДУКТ 56](#_Toc137603263)

[**6.1. Руководство пользователя** 56](#_Toc137603264)

[**6.2. Руководство администратора** 56](#_Toc137603265)

[**6.3. Определение качества программного продукта по метрикам** 56](#_Toc137603266)

[7. МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ТРУДА 58](#_Toc137603267)

[8. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА 60](#_Toc137603268)

[**8.1 Расчет затрат на разработку программного обеспечения** 60](#_Toc137603269)

[**8.2. Эксплуатационные затраты на оборудование** 61](#_Toc137603270)

[**8.3. Затраты на материалы** 62](#_Toc137603271)

[**8.4.**  **Себестоимость разработки программного обеспечения** 63](#_Toc137603272)

[**8.5. Расчет экономического эффекта разработчика и пользователя (заказчика) программного обеспечения** 63](#_Toc137603273)

[ЗАКЛЮЧЕНИЕ 65](#_Toc137603274)

[СПИСОК ИСТОЧНИКОВ ИНФОРМАЦИИ 66](#_Toc137603275)

[ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММНЫХ МОДУЛЕЙ 67](#_Toc137603276)

[ПРИЛОЖЕНИЕ 2. ПОЛИТИКА БЕЗОПАСНОСТИ 68](#_Toc137603277)

[ПРИЛОЖЕНИЕ 3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 69](#_Toc137603278)

[ПРИЛОЖЕНИЕ 4. РУКОВОДСТВО АДМИНИСТРАТОРА 74](#_Toc137603279)

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ:**

**SQL — Structured Query Language (язык структурированных запросов) ОЗУ – оперативное запоминающее устройство.**

**UML — Unified Modeling Language (унифицированный язык моделирования).**

**ПК — персональный компьютер.**

**СУБД — система управления базами данных.**

**ПО — программное обеспечение.**

**ERD — диаграмма «сущность-связь».**

**API — интерфейс прикладного программирования.**

**Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.**

**Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.**

**ЯП — язык программирования.**

**IDE (Integrated development environment) — внутренняя среда разработки.**

**ORM (Object-Relational Mapping) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».**

**АИС (Автоматизированная информационная система) - организационно-техническая система, состоящая из средств автоматизации определенного вида или нескольких видов деятельности людей и персонала, осуществляющего эту деятельность.**

**CRM (Система управления взаимоотношениями с клиентами) - прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками, в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов**

**ВВЕДЕНИЕ**

**Актуальность состоит в том, что в современных условиях работы предприятий возникает потребность автоматизировать процессы.** **В качестве примера: информационные системы позволяют людям, не отходя от рабочего места, узнать все необходимые данные о документации и наличии изделий, хранящихся на складе, и полностью, или частично, избавиться от бумажного документооборота, который имеет ограниченный срок службы, в отличие от электронных средств хранения данных.**

**Целью дипломной работы является разработка АИС, обладающей функционалом с достаточными показателями скорости работы, эффективности и удобства для администрирования и взаимодействия с сотрудниками, а также соответствующего всем действующим на момент написания требованиям к системе.**

* **Задачи, таким образом, подразделяются на:**
* **Формирование требований к программному продукту**
* **Проектирование и моделирование архитектуры, сценариев взаимодействия внутри проекта**
* **Непосредственно разработка и написание кода, разработка базы данных**
* **Тестирование системы**
* **Написание документации**

**1. ПОСТАНОВКА ЗАДАЧИ**

В современном бизнесе нельзя обойтись без технической поддержки в виде программ для ведения бизнеса, учета клиентов, поставщиков и поставок, а также учет продаж. Для удобства ведения коммерческих взаимоотношений, современные технологии предлагают использование онлайн-касс и автоматизированных информационных систем

## **1.1. Обзор аналогов**

Наиболее популярные CRM системы:

1. **AmoCRM** — система управления взаимоотношениями с клиентами, позволяющая автоматизировать продажи и организовать работу отдела маркетинга внутри небольшого предприятия. (Рис 1.1.)

Для менеджеров компании доступны:

* Удобный интерфейс для отслеживания целей и выполнения задач.
* Общение с клиентами непосредственно из CRM-карточки.
* Упрощенное ведение документооборота и запуск рассылок.
* Глобальный мессенджер для общения с сотрудниками внутри компании.

В свою очередь, для руководителя отдела продаж предусмотрены:

* Наглядная аналитика работы отдела.
* Настройка сложносоставных отчетов под нужды компании.
* Детализированный анализ работы колл-центра.
* Контроль над работой со сделками, в том числе и просроченными.
* Координирование работы менеджеров, распределение нагрузки на персонал.
* Планирование и разработка маркетинговой стратегии и настройка воронок продаж.

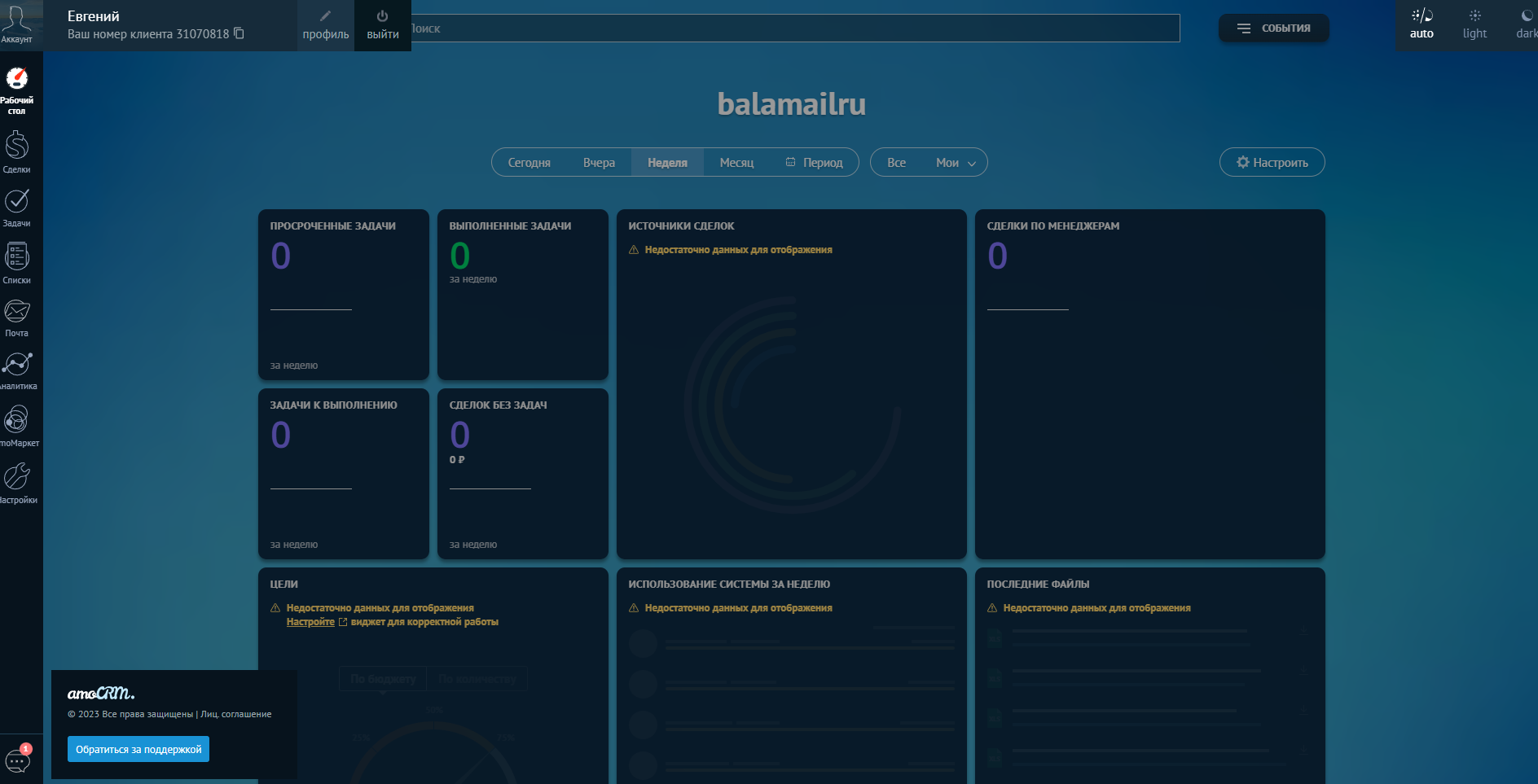


Рис. 1.1. AmoCRM

Функциональность программы определяется ее тарифным планом и количеством проведенных интеграций. AmoCRM предусматривает возможность тонкой настройки инструментария под требования компании благодаря открытому API и обширному каталогу приложений во встроенном магазине.

Основные инструменты системы:

* Рабочий стол
* Сделки
* Задачи и цели
* Списки(контакты)
* Почта
* Аналитика
* Настройка

1. **CRM Мегаплан** — мощная система для управления бизнесом с расширенными возможностями в области проектов и заказов. Продукт позволяет автоматизировать внутренние и внешние процессы, наладить взаимоотношение с клиентами, а также управлять и координировать персонал.

Мегаплан — СРМ-система, созданная в первую очередь для управления проектами и обеспечения контроля над сотрудниками. Инструментарий направлен на организацию труда, оптимизацию рутинных обязанностей и автоматизацию бизнес-процессов. (Рис 1.2.)

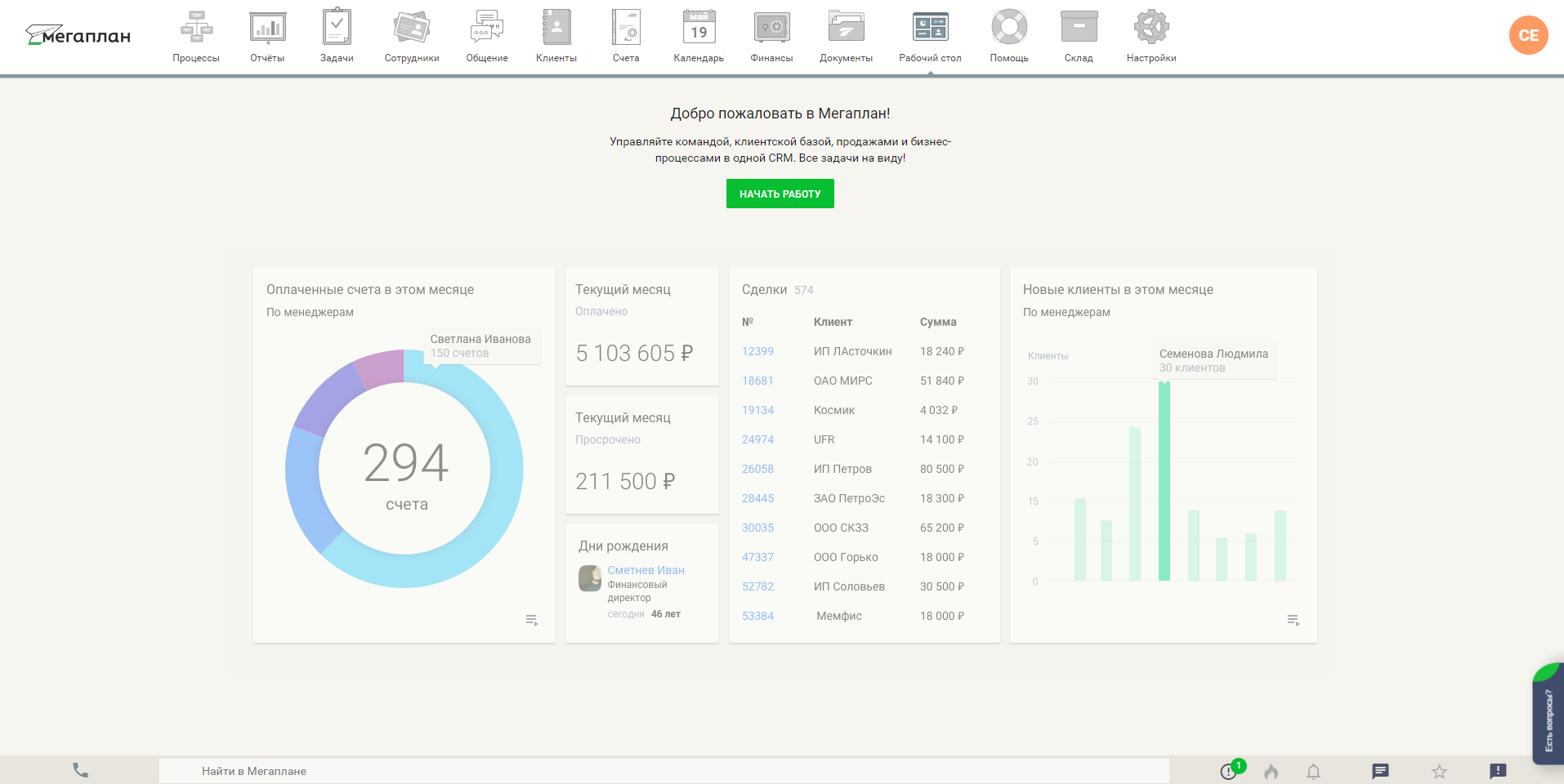


Рис. 1.2. Мегаплан

Задачи предприятия и онлайн-офис

Основной инструментарий направлен на проектную работу и постановку заданий, что позволяет полностью организовать бизнес-процессы предприятия с большим штатом. Система Мегаплан формирует единое рабочее пространство, где объединены:

* Рабочий
* Список заданий.
* Информер-панель
* Облачное хранилище

Департаменты в программе нужны, чтобы организовать все отделы предприятия, а также ограничить доступ и защитить конфиденциальные данные. Кроме того, все версии Мегаплан поддерживают мобильное приложение, через которое удобно контролировать дела, назначать ответственных и координировать подчиненных.

Управление продажами

В старших версиях предусмотрена CRM-система Мегаплан, позволяющая организовать отдел продаж и контакт с покупателями. Софт интегрируется с сайтом или интернет-магазином, предусматривает подключение телефонии и почты, а также все популярные мессенджеры или соцсети.

Основные инструменты:

* Воронка продаж
* База заказчиков и контрагентов
* Обработка счетов
* База сделок
* Учет движения финансов

1. **Salesforce** − это CRM-система, которая предназначена для управления самыми разными видами бизнес-процессов, отношениями с клиентами, аналитикой, маркетингом, продажами и прочее. (Рис 1.3.)

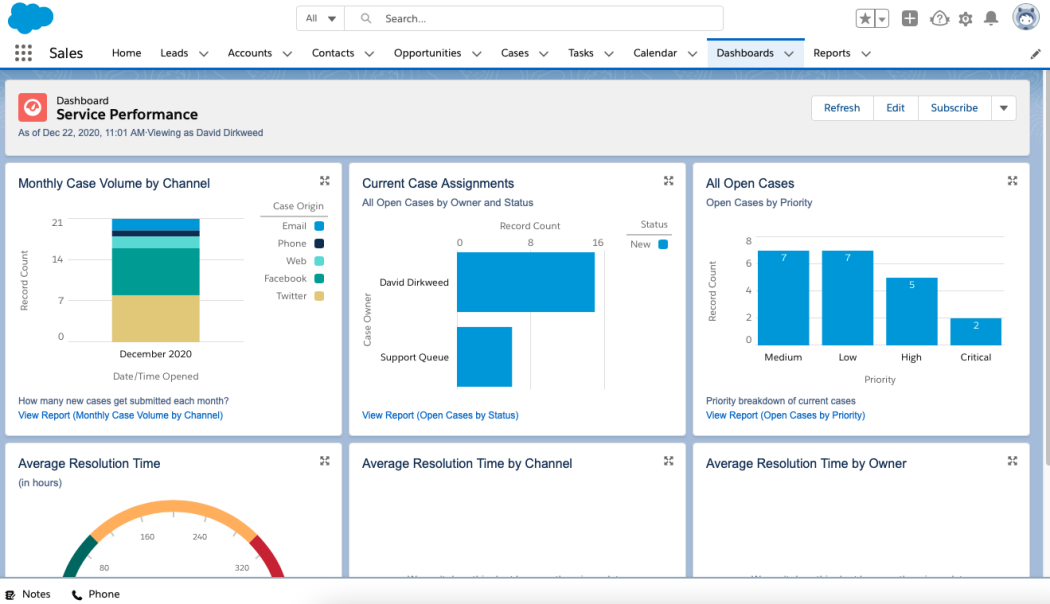


Рис. 1.3. Salesforce

Работает Salesforce CRM по модели SaaS (форме вычислений в облаке). Пользователям CRM-системы предоставляется готовое программное обеспечение, доступ к которому открытый через браузер или мобильное приложение. CRM-система ориентирована на малый, средний и крупный бизнес. Предлагает все необходимое для поиска, удержания клиентской базы, совершения сделки и прочее.

Предоставляет Salesforce в России следующие возможности:

* управление контактами;
* лидогенерация;
* прогнозирование;
* ведение отчетов;
* сотрудничество;
* автоматизация рабочего процесса;
* мобильность доступа.

Управление продажами автоматизировано комплексно, поэтому директор может получить отчет о продажах в любое удобное время, а продавец имеет возможность совершать сделки удаленно, используя смартфон. Система интегрирована с различными решениями по телефонии.

Таблица 1.1. Сравнение аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | AmoCRM | Мегаплан | Salesforce |
| Управление задачами и проектами | + | + | + |
| Управление сотрудниками | + | + | + |
| Отчеты и аналитика | + | + | + |
| Склад | + | + | + |
| Товары и услуги | + | + | + |
| Цены | Базовый - 499₽  Расширенный - 999₽  Профессиональный - 1499₽ | Базовый - 329₽  Расширенный - 599₽  Профессиональный - 839₽ | Lightning Professional - $75  Lightning Enterprise − $150  Lightning Unlimited − $300 |

Вывод:

Анализируя таблицу полученную таблицу, можно сопоставить результат для реализации АИС, где будут включены все удобства в функциональности и их потребности. Тем самым, система, которая будет разработана, будет иметь множество плюсов для использования в предприятии.

## **1.2 SWOT-анализ**

SWOT - анализ показывает, какие сильные и слабые стороны могут быть у сервиса во внутренней и внешней среде – Таблица 1.2. SWOT-анализ.

Таблица 1.2. SWOT - анализ

|  |  |  |
| --- | --- | --- |
|  | Положительное влияние/Сильные стороны | Отрицательное влияние/Слабые стороны |
| Внутренняя среда | Цены на услуги более низкие чем у конкурентов | Потребность в расходах на «раскрутку», для узнаваемости системы |
| Внешняя среда | Возможность быстрого роста благодаря всем нужным функционалам | Высокая конкуренция, многим пользователем привычнее пользоваться другими системами |

## **1.3 PEST-анализ**

Из PEST-анализа (Таблица 1.3.) понятно, какие риски могут быть у сервиса в политической, экономической, социальной и технологической сферах.

Таблица 1.3. PEST-анализа

|  |  |
| --- | --- |
| Political  Может быть ужесточение ограничений, что приведет к ограничению функционала системы | Economic  Из-за роста курса валют может не хватать средств для поддержания функционирования системы |
| Social  Центры, использующие другие системы могут создать мнение о недостаточности функционала. | Technological  Развитие систем конкурентов может привести к потери актуальности |

## **1.4 Формирование требований к программному продукту**

**1.4.1 Бизнес-требования**

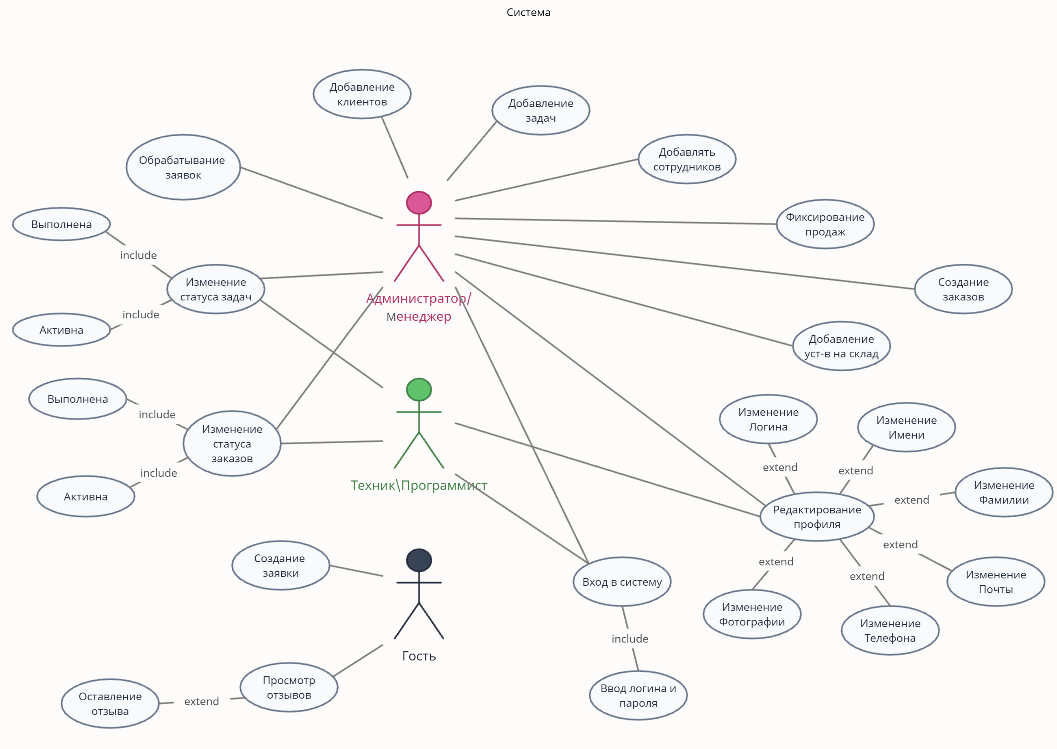
В настоящее время вся информация о принятых заказах хранится в электронных таблицах Microsoft Excel, что существенно затрудняло одновременную работу двум и более пользователям с данными, и тем самым отрицательно влияло на динамику работы сотрудников в условиях дефицита рабочего времени.

Таким образом, большое количество заявок и длительное время обработки может привести к множеству проблем: неправильно введенные данные или их потеря, дублирование информации, невозможность отслеживания хода ремонта, так как невозможно быстро найти нужную информацию, а клиентам сложнее узнать состояние ремонта. Точно так же ручная документация увеличивает вероятность пропуска важных моментов при приеме ремонта. Не исключены случаи подделки документации.

Без централизованного сбора информации труднее получить информацию о конкретном клиенте, а руководству трудно понять, какой объем работы проделал сотрудник за определенный период времени. АИС позволяет решить все эти проблемы.

### **1.4.2 Пользовательские требования**

Диаграмма вариантов использования (use-case) в UML — диаграмма, показывающая отношения между “актерами” и прецедентами, и являющаяся составной частью общей модели прецедентов пользования, позволяющей описать систему на концептуальном уровне. Построенная в рамках проектирования дипломной работы диаграмма для роли всех типов (“Гость”, “Техник” или “Программист”, “Администратор” или “Менеджер”). На рисунке 1.1 изображена диаграмма вариантов использования (диаграмма use-case)

Рис. 1.4. Диаграмма use-case

### **1.4.3 Функциональные требования к программному продукту**

Приложение для учёта заявок строиться на основании нескольких функциональных ролей:

* Администратор\Менеджер – авторизованный сотрудник, наделенный правами администратора, которому доступны все возможности
* Техник\Программист – авторизованный сотрудник, который имеет ограниченный спектр функционала для своей работы
* Гость\Клиент – неавторизованный пользователь, который может оставлять и просматривать отзывы, а также писать заявки

Система должна обладать следующими функциями:

* Авторизация в системе с использованием логина и пароля;
* Регистрация новых сотрудников;
* Авторизация в админ-панели;
* Добавление и обработка заявок;
* Добавление и редактирование задач, заказов, клиентов, сотрудников и продаж;
* Экспорт;
* Управление учетной записью
* Просмотр справки о странице

Перечень функций и use-case диаграмма приведены в пунктах 1.1.3.1 и 1.1.2.

Распределение функций по ролям указано в use-case диаграмме.

### **1.4.4 Нефункциональные требования**

Данные должны передаваться между серверной и клиентской частью приложения по протоколу HTTPS в формате JSON.

Для работы с клиентской частью web-приложения необходимо современный браузер, с поддержкой HTML5, CSS 3 и ECMAScript 2015 (ES6). Например:

* Google Chrome;
* Yandex браузер;
* Opera;
* Safari.

### **1.4.5 Ограничения**

Ограничения функциональности:

* Только администратор может зайти в Админ-панель;
* Гость никак не может попасть в систему;
* Техник или программист имеют ограниченный функционал;

Ограничения на ввод:

Имя пользователя может в себе содержать:

* Латиницу;
* Цифры;
* Знаки ! # $ % & ' \* + — / =? ^ \_ ` { | } ~
* Точку, за исключением первого и последнего знака, которая не может
* повторяться.

Ограничения на ввод текста:

* Длина заявки должна быть не менее 1 и не более 300 символов;
* Длина отзыва должна быть не менее 1 и не более 150 символов;
* Длина описания заказа должна быть не менее 1 и не более 1000 символов;
* Длина описания задачи должна быть не менее 1 и не более 1000 символов;

Ограничения на ввод логина:

* Длина должны быть не менее 3 и не более 40 символов.

Ограничения на ввод пароля:

* Длина должны быть не менее 8 символов;
* Должен содержать хотя бы одну прописную букву;
* Должен содержать хотя бы одну заглавную букву;
* Должен содержать хотя бы одну цифру.

### **1.4.6 Требования к интерфейсам**

Для хранения информации нужен сервер базы данных. Сведения должны выбираться, изменяться, добавляться с помощью API с использованием ЯП Python и фреймворка Django.

Также соблюдаются определенные стилевые установки:

Цветовая гамма программного продукта белый и темно-серо-синий;

Большая часть элементов управления имеет анимацию для визуальной наглядности;

Шрифт: Trebuchet MS;

Цвет шрифта: светло-серый, белый, черный, темно-бордовый, #252930, #eb4f37, #44cc66

Размер шрифта на сайте– от 20 px до 40 px;

### **1.4.7 Требования к данным**

Данные, которые будет вносить пользователь должны соблюдать тип данных, которому он соответствует в базе данных:

1. Для таблицы пользователей:

* id — уникальный идентификатор пользователя.
* first\_name, last\_name — поля, предназначенные для хранения имени и фамилии пользователя.
* username — поле, предназначенное для хранения логина пользователя. Это необходимо для идентификации пользователя во время входа в приложение.
* email — поле почты, предназначенное для хранения электронной почты пользователя.
* password — поле, предназначенное для хранения пароля пользователя. Это необходимо для идентификации пользователя во время входа в приложение.
* user\_phone — поле, предназначенное для хранения телефона пользователя.
* user\_note — поле, предназначенное для хранения примечаний о пользователе.
* user\_time — поле, предназначенное для хранения даты начала работы пользователя.
* user\_photo — поле, предназначенное для хранения Фото профиля пользователя.
* user\_role — поле выбора, предназначенное для хранения должности пользователя. (“Администратор”, “ Техник”, “ Программист”, “ Менеджер”).

1. Для таблицы задач:

* id — уникальный идентификатор задачи.
* task\_name— поле, предназначенное для хранения названия задачи.
* task\_description — поле, предназначенное для хранения описания задачи.
* task\_time— поле времени, предназначенное для хранения крайнего срока.
* task\_active— поле выбора, предназначенное для хранения статуса задачи (“Активна”, “Выполнена”.)

1. Для таблицы клиентов:

* id — уникальный идентификатор клиента.
* client\_type— поле выбора, предназначенное для хранения типа клиента (“Физ. Лицо”, “ Компания”).
* client\_name— поле, предназначенное для хранения имени клиента.
* client\_phone— поле, предназначенное для хранения номера телефона клиента.
* client\_email— поле, предназначенное для хранения почты клиента.
* client\_address— поле времени, предназначенное для адреса клиента.
* client\_found — поле выбора, предназначенное для хранения информации (“Знакомые”, “ Интернет”, “ Реклама”).

1. Для таблицы заказов:

* id — уникальный идентификатор заказа.
* order\_name— поле, предназначенное для хранения названия заказа.
* order\_description — поле, предназначенное для хранения описания заказа.
* order\_time — поле времени, предназначенное для хранения крайнего срока.
* order\_active — поле выбора, предназначенное для хранения статуса задачи (“Активна”, “Выполнена”.)
* order\_price — поле, предназначенное для хранения стоимости заказа.
* order\_client — поле выбора, предназначенное для хранения клиента.

1. Для таблицы склада:

* id — уникальный идентификатор вещи.
* inventory\_type— поле выбора, предназначенное для хранения типа вещи (“Устройство”, “ Запчасти”).
* inventory\_article— поле, предназначенное для хранения артикула.
* inventory\_name — поле, предназначенное для хранения названия вещи.
* inventory\_color — поле, предназначенное для хранения цвета.
* inventory\_condition — поле, предназначенное для хранения состояния.
* inventory\_description — поле, предназначенное для хранения описания.
* inventory\_client — поле, предназначенное для хранения владельца.

1. Для таблицы продаж:

* id — уникальный идентификатор продажи.
* sales\_name— поле, предназначенное для хранения названия продажи.
* sales\_comment — поле, предназначенное для хранения описания продажи.
* sales\_time — поле времени, предназначенное для хранения времени продажи.
* sales\_price — поле, предназначенное для хранения цены.
* sales\_client — поле выбора, предназначенное для хранения клиента.

1. Для таблицы отзывов:

* id — уникальный идентификатор отзыва.
* review\_fio — поле, предназначенное для хранения ФИО.
* review\_gender — поле выбора, предназначенное для хранения пола ('Мужчина', 'Женщина').
* review\_email — поле, предназначенное для хранения почты.
* review\_comment — поле, предназначенное для хранения отзыва.
* review\_grade — поле выбора, предназначенное для хранения оценки ('Отлично!', 'Очень хорошо', 'Хорошо', 'Плохо', 'Очень плохо').

1. Для таблицы заявок:

* id — уникальный идентификатор заявки.
* application\_fio — поле, предназначенное для хранения ФИО.
* application\_phone — поле, предназначенное для хранения телефона
* application\_email — поле почты, предназначенное для хранения электронной почты
* application\_address — поле, предназначенное для хранения адреса
* application\_found — поле выбора, предназначенное для хранения информации (“Знакомые”, “ Интернет”, “ Реклама”).
* application\_comment — поле, предназначенное для хранения заявки
* application\_active — поле выбора, предназначенное для хранения статуса заявки (“Активна”, “Выполнена”.)

### **1.5 Программные средства разработки**

Работа была выполнена с использованием следующих средств разработки программных продуктов:

1. PyCharm — это кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA. Предоставляет пользователю комплекс средств для графических отладчиков и работы с кодом. Продукт доступен в двух версиях: PyCharm Community Edition - бесплатная версия, находится под лицензией Apache License, и PyCharm Professional Edition - расширенная версия продукта, обладающая дополнительной функциональностью, является проприетарным ПО.

Возможности:

• Отладка кода при помощи PyDev;

• Рефакторинг кода

• Поддержка Git, SVN, Mercurial и других систем контроля версиями;

• Автодополнение кода

1. Django

В разработке был использован популярный фреймворк Django языка Python. В отличие от подобных фреймворков, обработчики ссылок в Django конфигурируются с помощью регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель описывается классами Python, и по ней генерируется схема БД.

1. SQLite — компактная (тесно связанная с программой и работающая, не требуя профессионального администрирования) встраиваемая СУБД.

Сама библиотека написана на ЯП “C”; существует большое количество привязок к другим языкам программирования, в том числе “Apple Swift”, “Delphi”, “C++”, “Java”, “C#”, “VB.NET”, “Python”, “Perl”, “PHP”, “PureBasic”, “Tcl”, “Ruby”, “Haskell”, “Scheme”, а также ко многим другим.

1. HTML (HyperText Markup Language) — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Элементами HTML являются строительные блоки на страницах. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую браузером страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов.
2. CSS (Cascading Style Sheets) — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

1. JS (JavaScript) — ­­­ мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений.

### **1.6 Аппаратные средства разработки**

В ходе разработки дипломной работы использовались персональные компьютеры на базе архитектуры x64 со следующими характеристиками:

* Стационарный ПК следующей конфигурации:
* Процессор AMD Ryzen 5 2600 Six-Core Processor 3.40 GHz
* Видеоадаптер NVIDIA GeForce GTX 1060 объемом 3GB
* Объем оперативной памяти DDR3 объемом 16 ГБ
* Объем магнитного накопителя (HDD) 500 ГБ
* Операционная система Windows 10.

**Вывод первой главы:**

Проведенное исследование позволило сделать следующее:

* Сформулировать требования к системе;
* Описать назначения программного продукта;
* Определить пользователей системы (их тип, роли, категории);
* Проанализировать входные и выходные сообщения системы.

**2. ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ**

**2.1. Архитектура системы**

В ходе разработки была выбрана трехуровневая архитектура, состоящая из:

* Уровня представления — представляет из себя интерфейс пользователя, который работает через браузер. На этом уровне располагается простейшая бизнес-логика.
* Уровня приложения — на этом уровне реализуется основная бизнес-логика программного продукта;
* Уровня данных — состоит из базы данных с, непосредственно, данными, находящимися в ней.

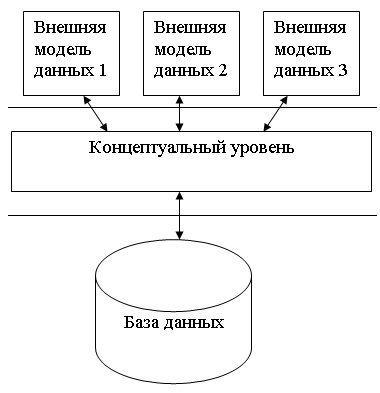


Рис. 2.1. Трехуровневая архитектура системы

**2.2. Моделирование основных сценариев системы**

На рис. 2.2 и 2.3 изображены схемы IDEF0 первого, второго уровня соответственно, отображающие основные функции системы.

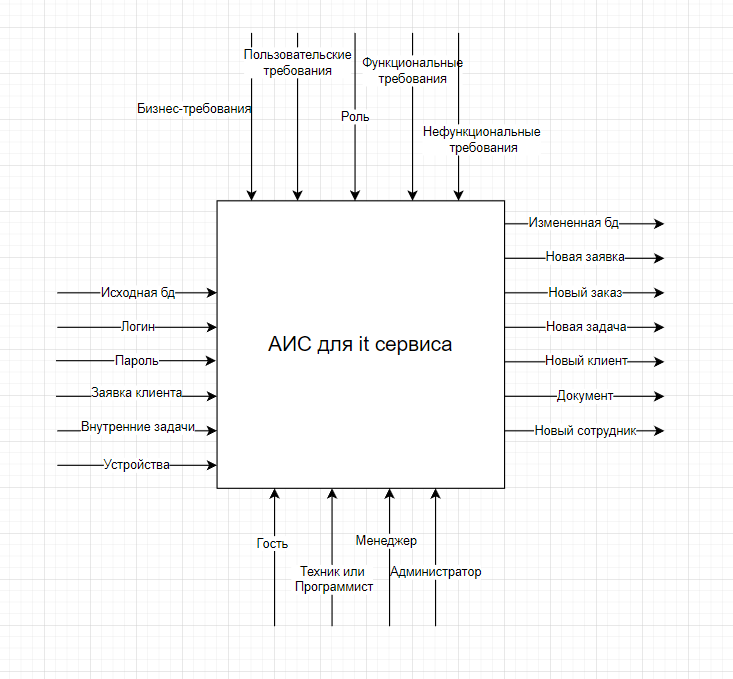


Рис. 2.2. Диаграмма IDEF0 первого уровня для разрабатываемого ПО

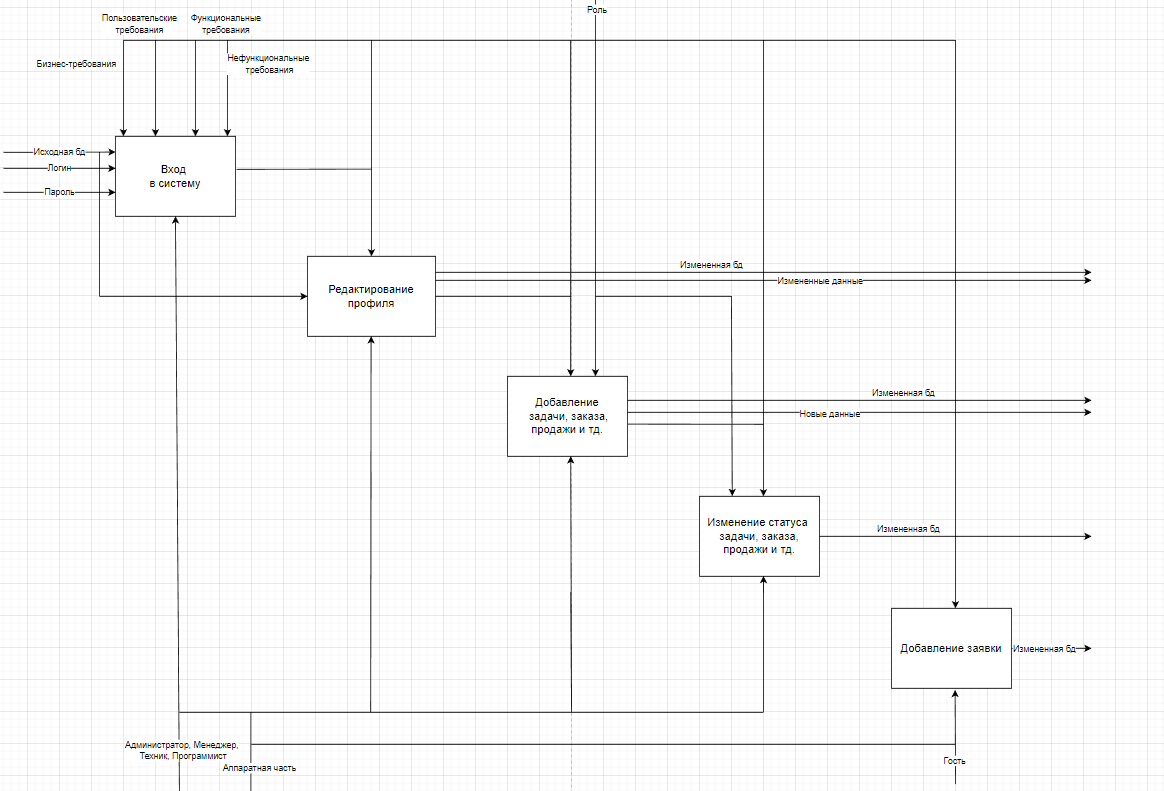


Рис. 2.3. Диаграмма IDEF0 второго уровня для разрабатываемого ПО

Так же была разработана диаграмма третьего уровня для блока “Добавление продажи” изображенная на рис. 2.4.

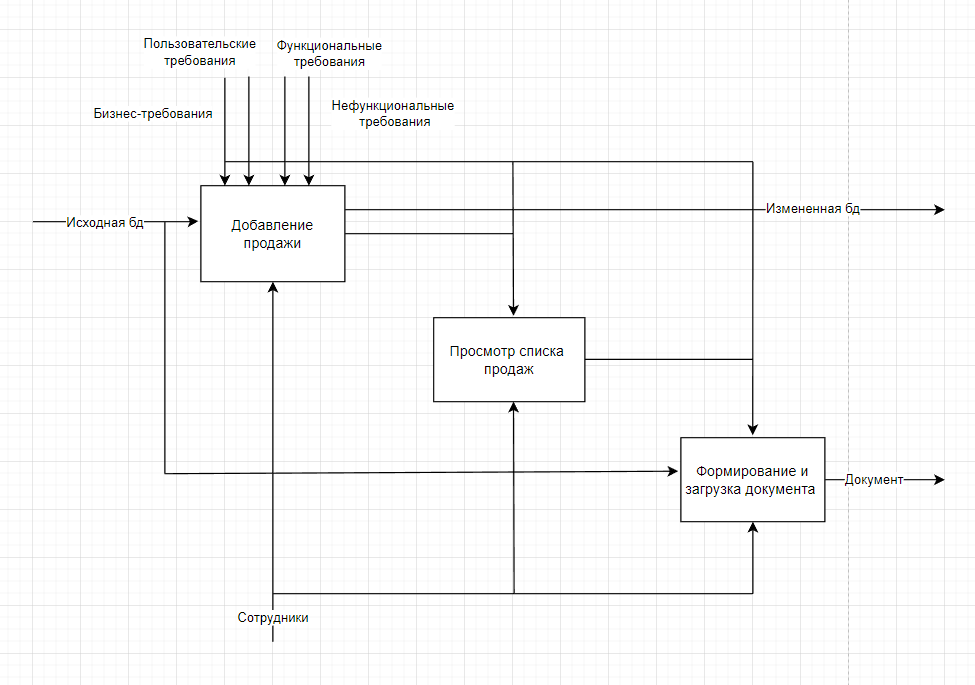


Рис. 2.4. Диаграмма IDEF0 третьего уровня для добавления продажи и формирования отчетного файла Excel

По диаграммам видно, что основной функционал приложения – это инвентаризация оборудования. В свою очередь для реализации данной задачи приложение содержит ряд функций: авторизация, редактирование профиля, добавление и изменения статуса записей.

Глобально ограничениями приложения выступают: бизнес-требования, пользовательские, функциональные и нефункциональные требования.

В системе реализуется 5 ролей: Администратор, Менеджер, Техник, Программист и Гость. Различие между ролями заключается в доступности функционала.

Диаграмма последовательности — это UML-диаграмма, на которой для некоторого набора объектов на временной оси показан жизненный цикл объекта и взаимодействие “актеров” информационной системы в рамках конкретного прецедента.

На рисунке 2.5. изображена диаграмма последовательности для прецедента функции «Добавление нового сотрудника».

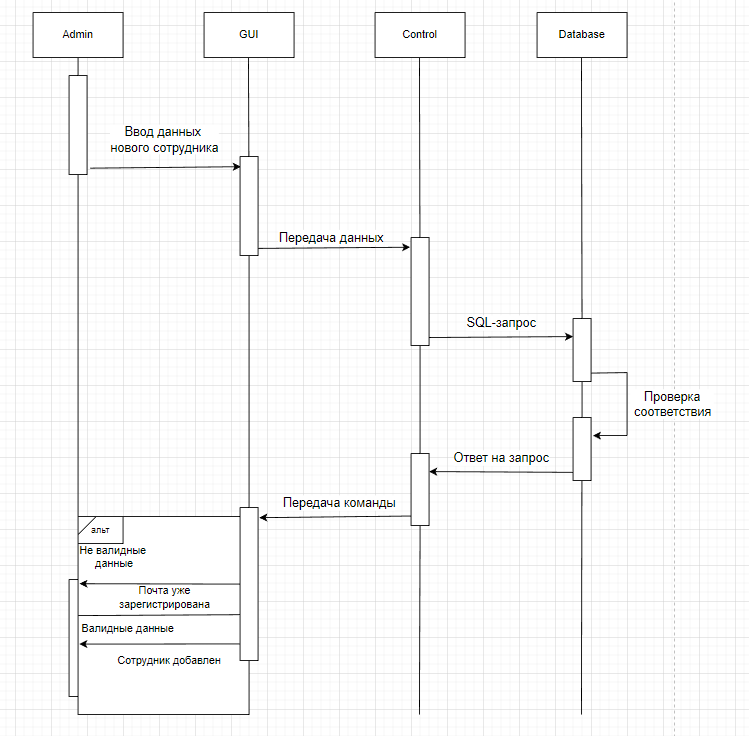


Рис. 2.5. Диаграмма прецедента последовательности «Добавление нового сотрудника»

На рисунке 2.6. изображена диаграмма последовательности для прецедента функции «Добавление новой задачи».

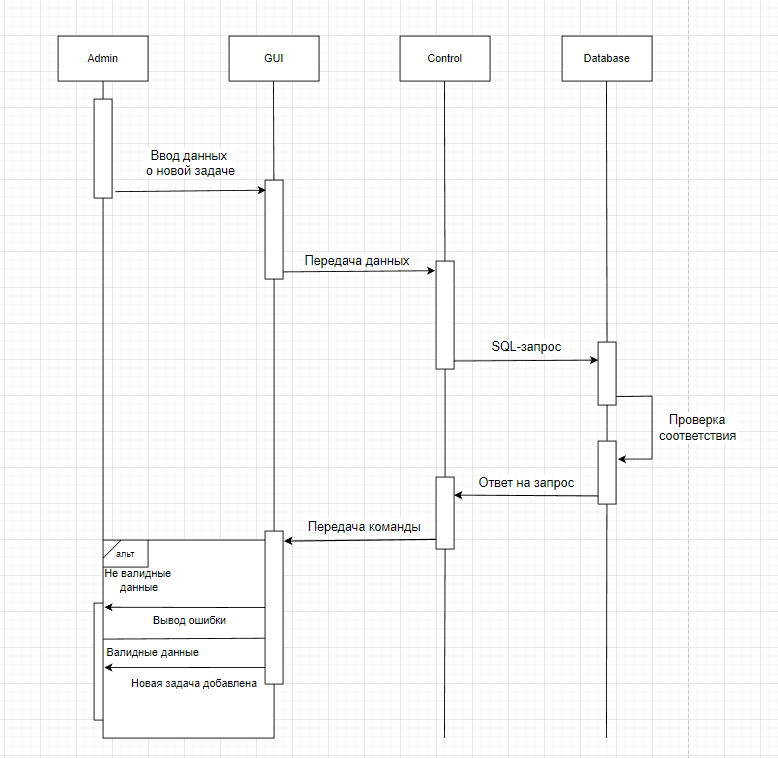


Рис. 2.6. Диаграмма прецедента последовательности «Добавление новой задачи»

На рисунке 2.7. изображена диаграмма последовательности для прецедента функции «Добавление нового заказа».

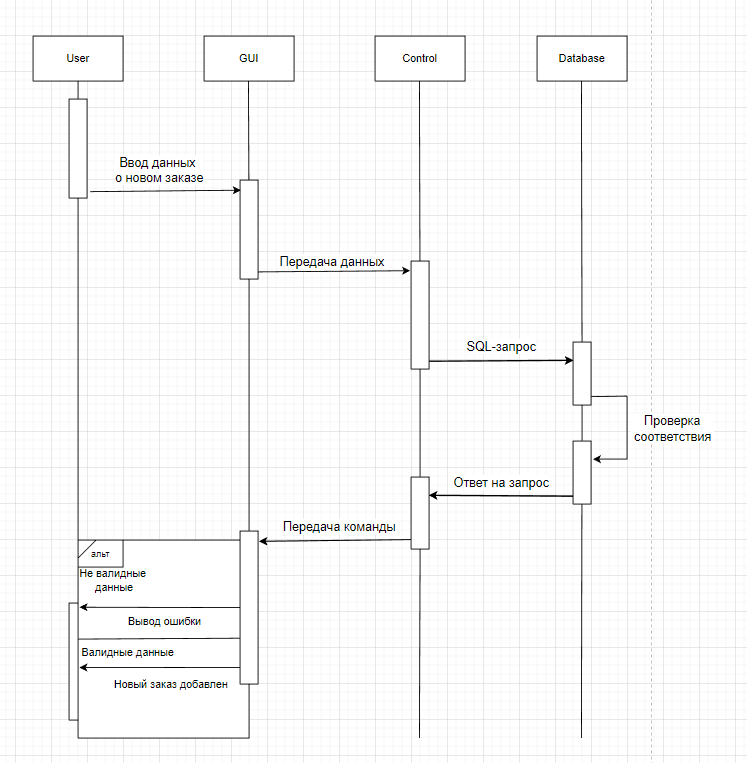


Рис. 2.7. Диаграмма прецедента последовательности «Добавление нового заказа»

На рисунке 2.8. изображена диаграмма последовательности для прецедента функции «Добавление нового клиента».

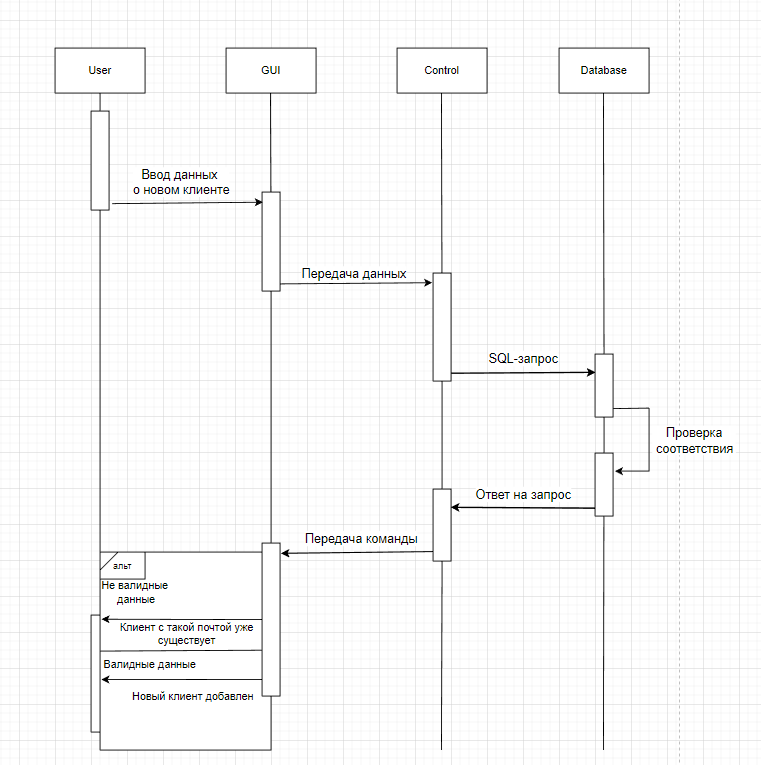


Рис. 2.8. Диаграмма прецедента последовательности «Добавление нового клиента»

На рисунке 2.9. изображена диаграмма последовательности для прецедента функции «Изменение личных данных».

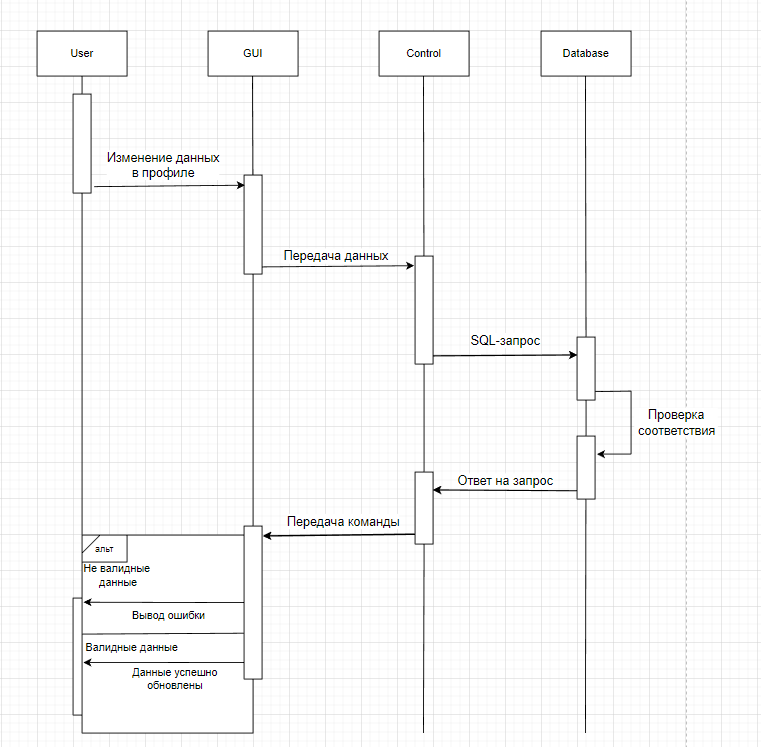


Рис. 2.9. Диаграмма прецедента последовательности «Изменение личных данных»

Диаграмма активностей (видов деятельности) — диаграмма, отражающая динамические моменты поведения проектируемой системы. На ней наглядным образом показан процесс перехода потока управления от одного компонента к другому.

На рисунке 2.10 изображена диаграмма активностей для прецедента «Добавление задачи». На диаграмме представлены следующие объекты: Администратор, Система и База данных.

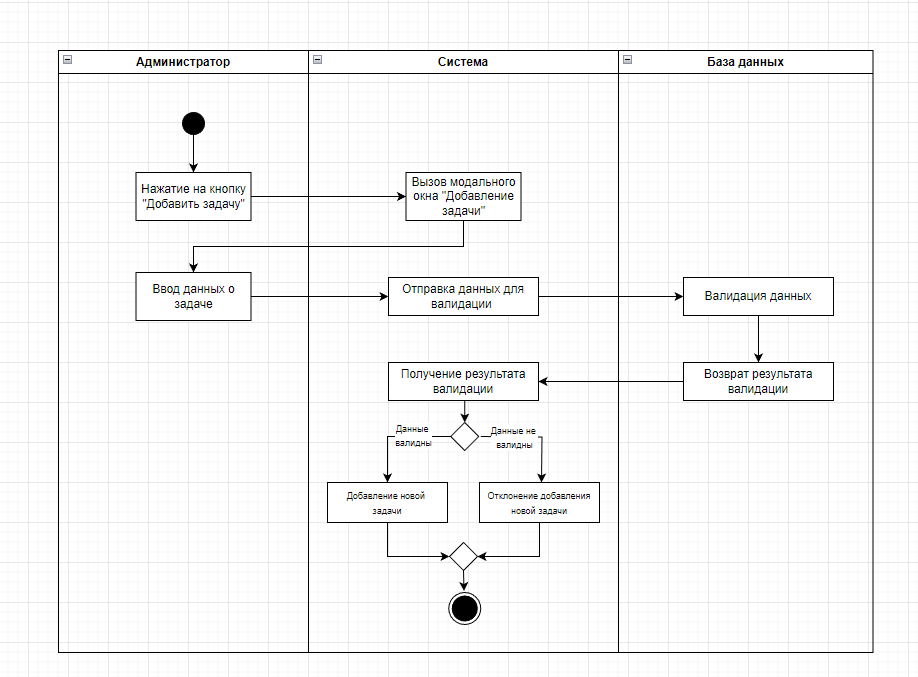


Рис. 2.10. Диаграмма прецедента активностей " Добавление задачи ".

На рисунке 2.11 изображена диаграмма активностей для прецедента «Добавление нового сотрудника». На диаграмме представлены следующие объекты: Администратор, Система и База данных.

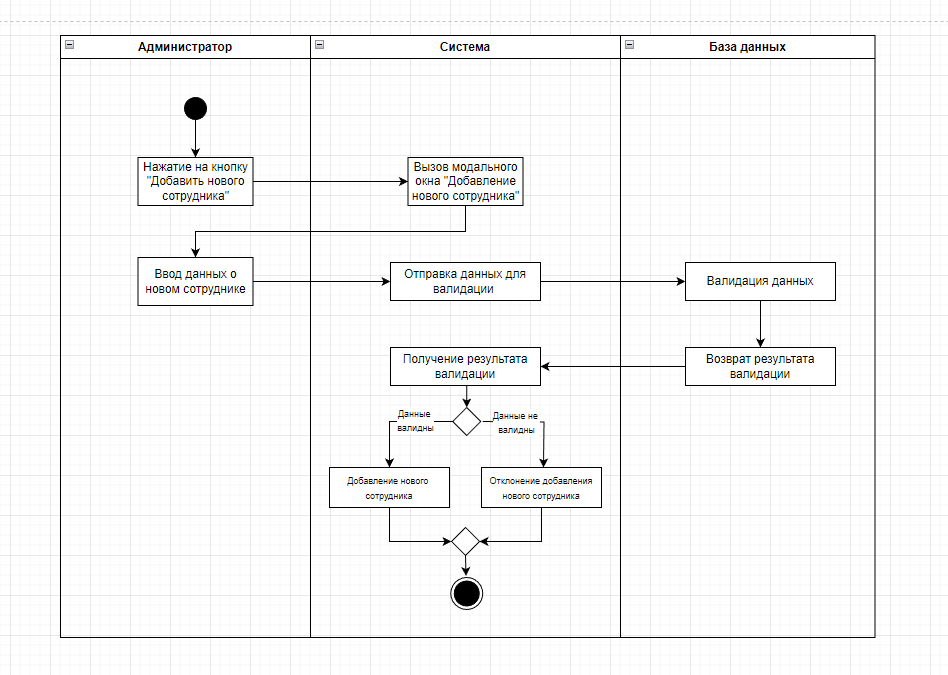


Рис. 2.11. Диаграмма прецедента активностей «Добавление нового сотрудника".

На рисунке 2.12 изображена диаграмма активностей для прецедента «Редактирование профиля». На диаграмме представлены следующие объекты: Сотрудник, Система и База данных.

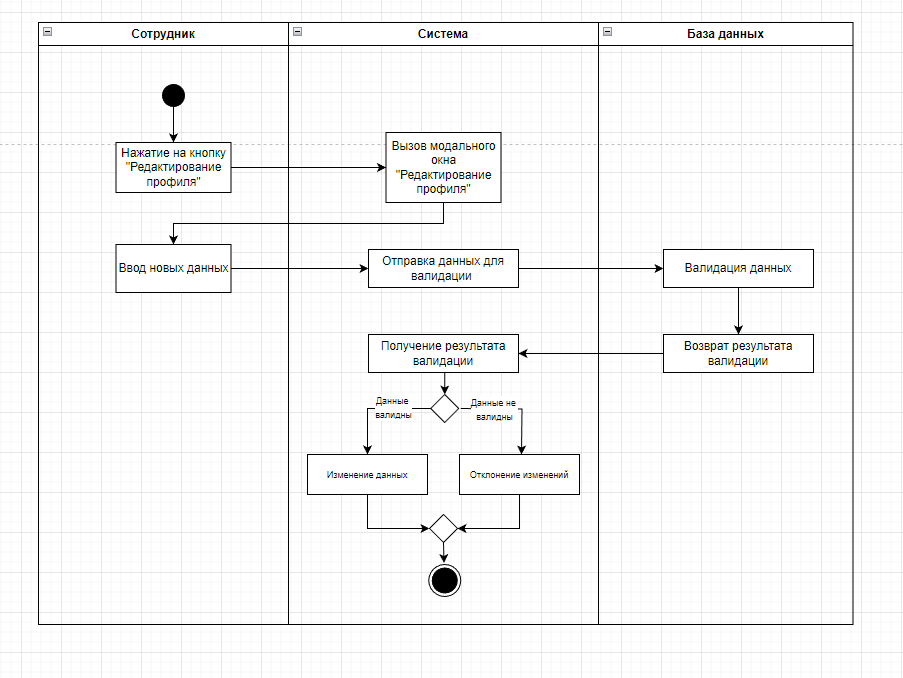


Рис. 2.12. Диаграмма прецедента активностей "Редактирование профиля".

На рисунке 2.13 изображена диаграмма активностей для прецедента «Авторизация». На диаграмме представлены следующие объекты: Сотрудник, Система и База данных.

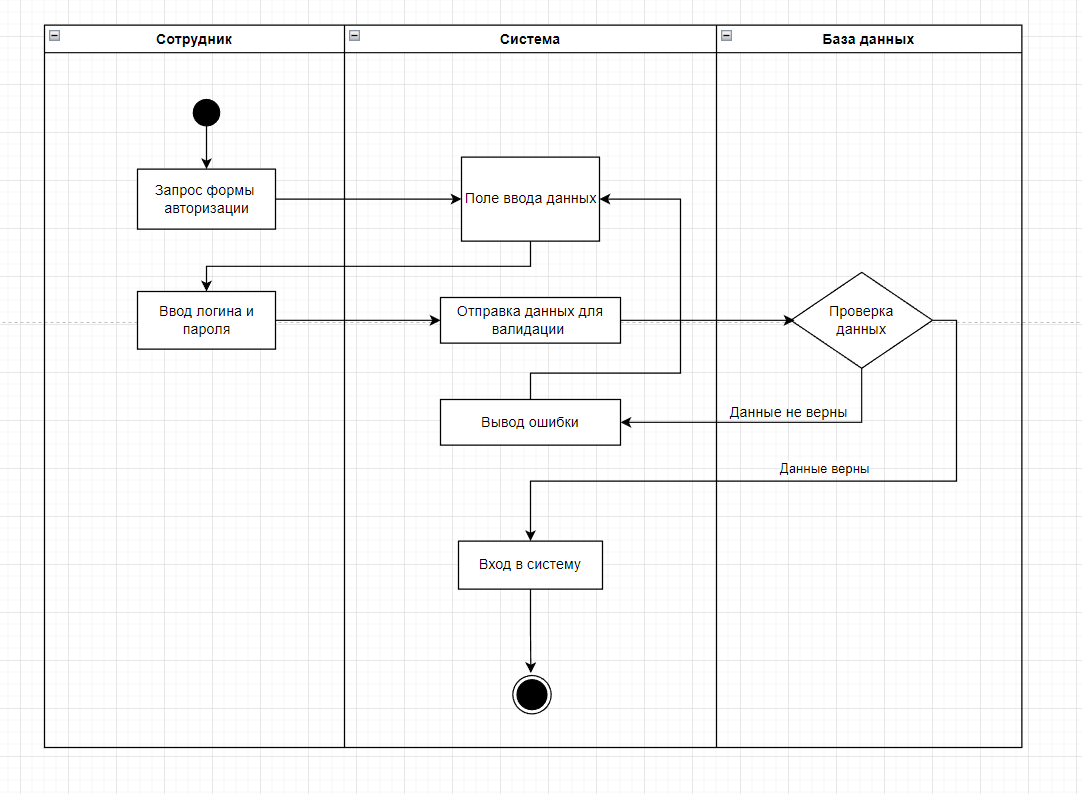


Рис. 2.13. Диаграмма прецедента активностей "Авторизация".

На рисунке 2.14 изображена диаграмма активностей для прецедента «Добавление нового отзыва». На диаграмме представлены следующие объекты: Гость, Система.

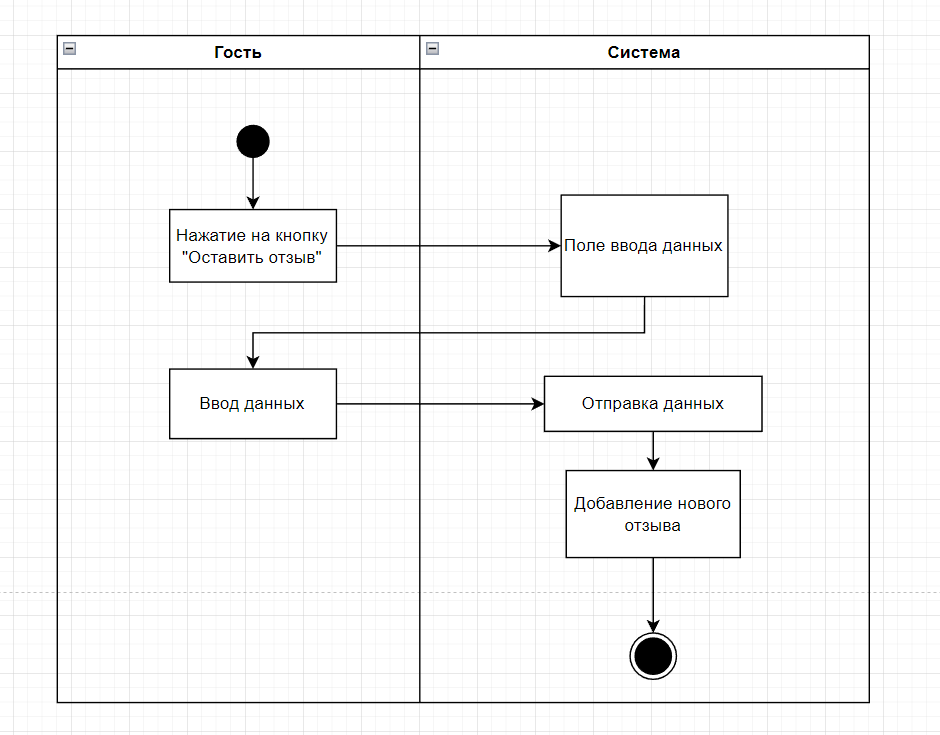


Рис. 2.14. Диаграмма прецедента активностей "Добавление нового отзыва ".

**2.3. Проектирование графического интерфейса пользователя**

После согласования всех требований, задач и функционала программного продукта, построения всех диаграмм по стандартам UML и IDEF, разработки базы данных необходимо выполнить разработку самого программного продукта.

Графический интерфейс пользователя должен быть понятен, не запутан и в то же время не перегружен контентом. Это достигается путем использования привычных для пользователя элементов управления.

Разрабатываемый программный продукт для роли «Администратор» или «Менеджер» имеет следующие страницы: «Профиль», «Задачи», «Заказы», «Продажи», «Склад», «Клиенты», «Сотрудники», «Заявки» и «Админ». Все это видно на Рисунке 2.15.

Страница «Профиль» (Рис 2.15.) выводит актуальные данные о сотруднике с фотографией и должностью, так же имеются две кнопки «Редактировать профиль» и «Выйти из системы».

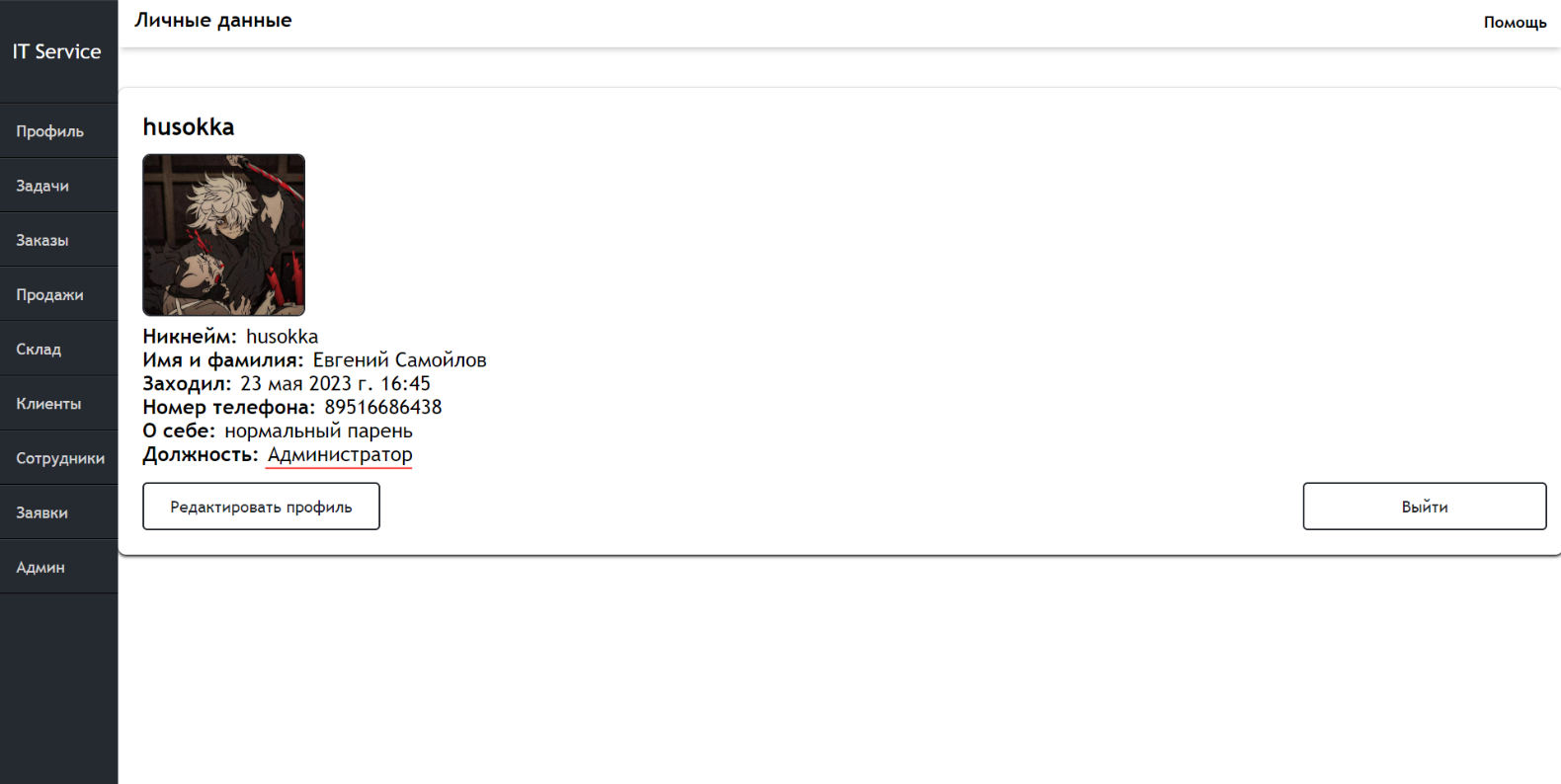


Рис. 2.15. Личный профиль Администратора.

Роли «Техник» и «Программист» имеют следующие страницы: «Профиль», «Задачи», «Заказы», «Продажи», «Склад» и «Клиенты», так как они имеют ограниченный функционал. Все это видно на Рисунке 2.16.

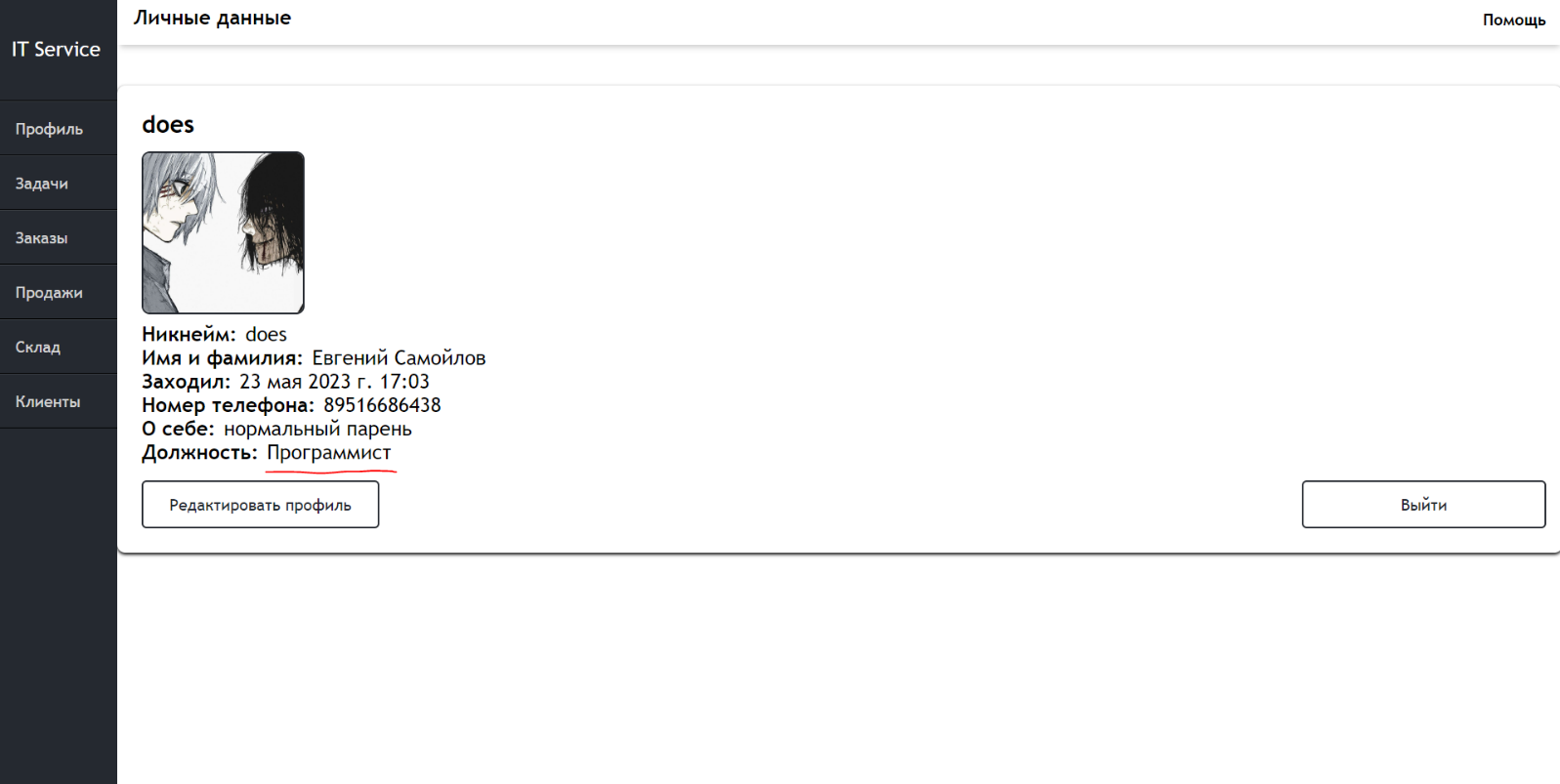


Рис. 2.15. Личный профиль Программиста.

Гость никак не сможет попасть в систему, не зная логина и пароля, так как будет выводится страница с оповещением (Рис. 2.16.)

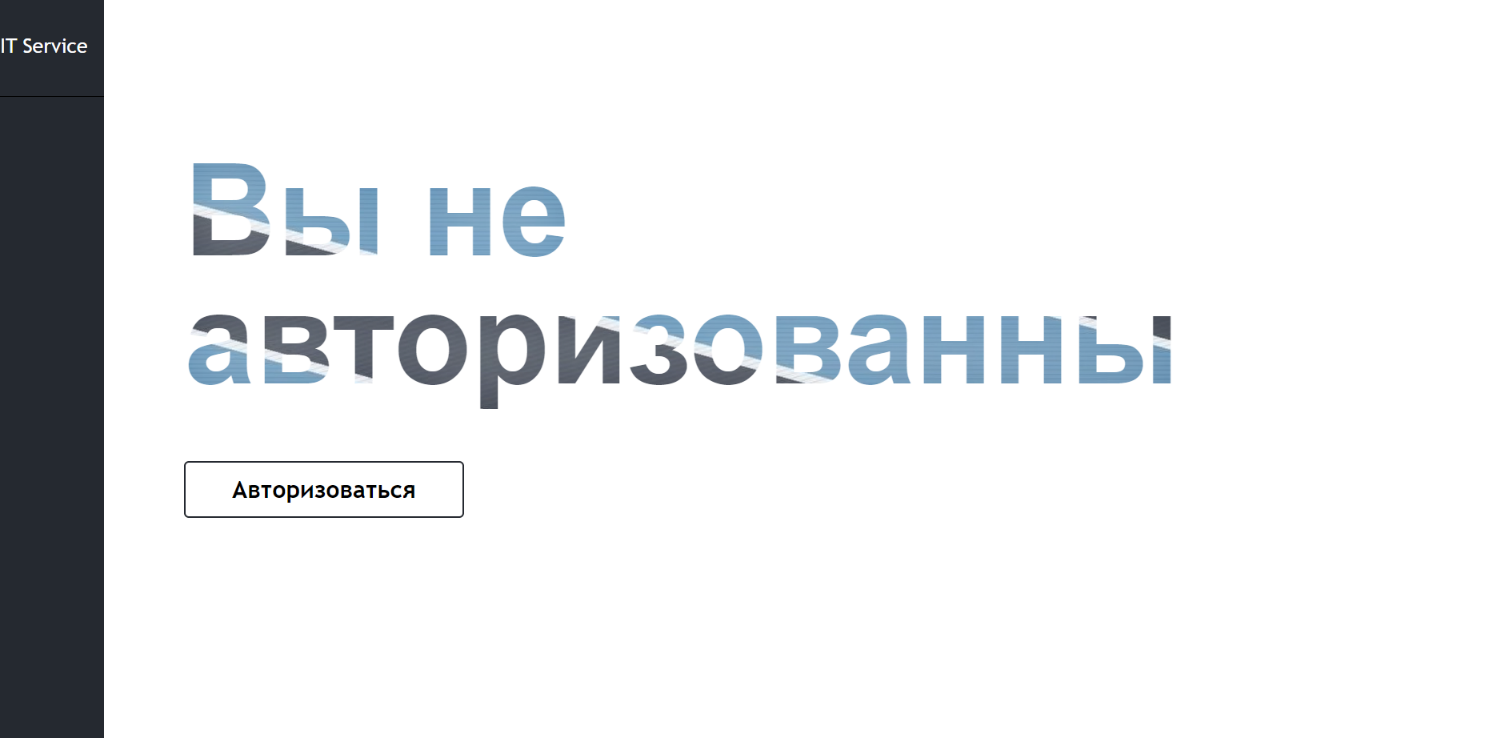


Рис. 2.16. Любая страница системы, когда сотрудник не авторизован.

Гость, заходя на сайт, видит совершенно другую страницу, на которой он может посмотреть или написать отзывы о работе IT Сервиса, так же имеется возможность написать заявку с проблемой и отправить ее на рассмотрение (Рис. 2.17 – 2.20)

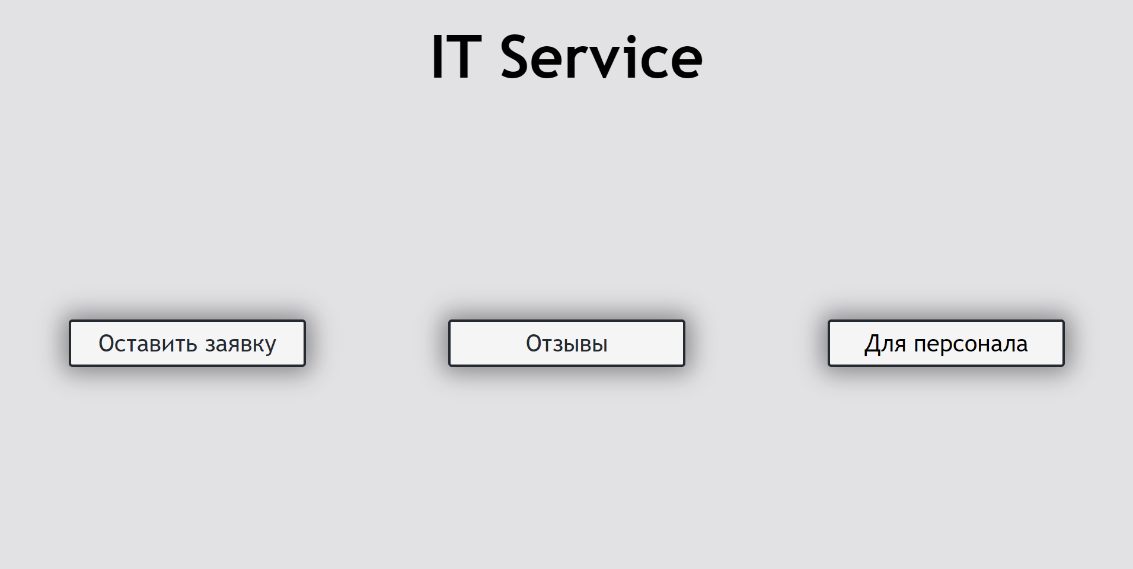


Рис. 2.17. Главная страница для Гостя.

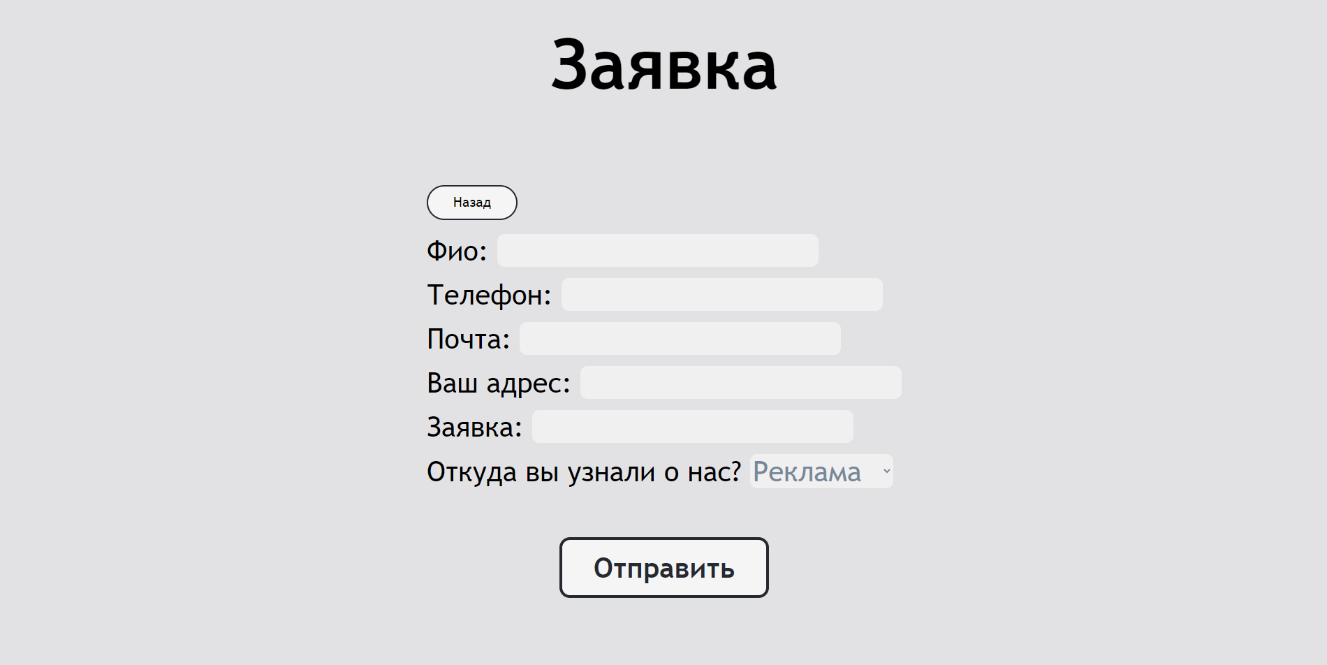


Рис. 2.18. Страница добавления новой заявки.



Рис. 2.19. Страница просмотра отзывов.

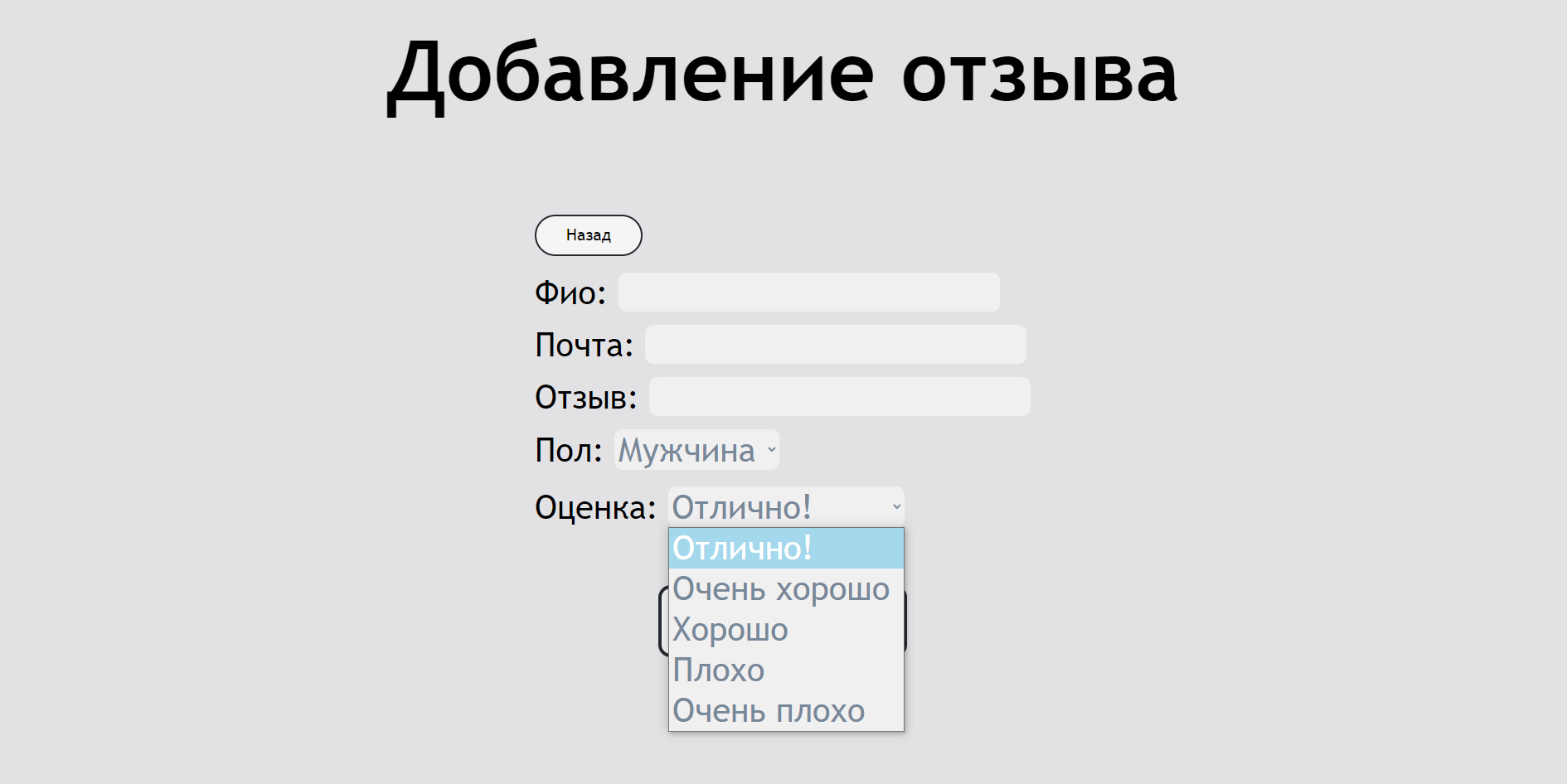


Рис. 2.20 Страница добавления отзыва.

**2.4. Проектирование и разработка модели данных**

При проектировании программного продукта была создана база данных, таблицы и поля, между которыми затем были созданы зависимости и диаграмма “сущность-связь”.

БД содержит следующие таблицы:

1. Таблица “users” – расширенная таблица от AbstractUser, содержит информацию о сотрудниках.

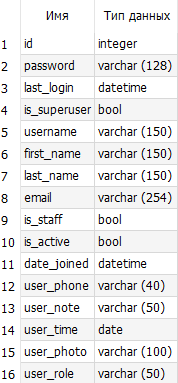


Рис 2.2. Таблица “users”

Описание полей таблицы “ users”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор сотрудника | int |
| password | Пароль сотрудника | varchar(128) |
| last\_login | Дата последней авторизации | datetime |
| is\_superuser | Статус суперпользователя | bool |
| username | Логин сотрудника | varchar(150) |
| first\_name | Настоящее имя сотрудника | varchar(150) |
| last\_name | Настоящая фамилия сотрудника | varchar(150) |
| email | Электронная почта сотрудника | varchar(254) |
| is\_staff | Статус, определяющий, является ли сотрудник администратором | bool |
| is\_active | Статус доступа пользователя (заблокирован или нет) | bool |
| data\_joined | Дата регистрации сотрудника | datetime |
| user\_phone | Номер телефона сотрудника | varchar(40) |
| user\_note | Примечание о сотруднике | varchar(50) |
| user\_time | Дата начала работы сотрудника | date |
| user\_photo | Фото сотрудника | varchar(100) |
| user\_role | Должность сотрудника | varchar(50) |

1. Таблица “ tasks” – содержит информацию о задачах.

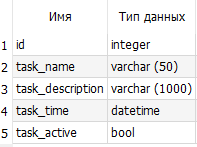


Рис 2.3. Таблица “ tasks”

Описание полей таблицы “ tasks”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор задачи | int |
| task\_name | Название задачи | varchar(50) |
| task\_description | Описание задачи | varchar(1000) |
| task\_time | Крайнего срок задачи | datetime |
| task\_active | Статус | bool |

1. Таблица “ client” – содержит информацию о клиентах.

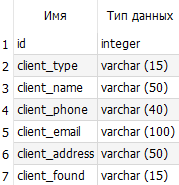


Рис 2.4. Таблица “ client”

Описание полей таблицы “ client”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор клиента | int |
| client\_type | Тип клиента | varchar(15) |
| client\_name | Имя клиента | varchar(50) |
| client\_phone | Телефон клиента | varchar(40) |
| client\_email | Электронная почта клиента | varchar(100) |
| client\_address | Адрес клиента | varchar(50) |
| client\_found | Откуда о нас узнал клиент | varchar(15) |

1. Таблица “ orders” – содержит информацию о клиентах.

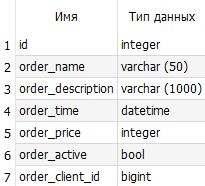


Рис 2.5. Таблица “ orders”

Описание полей таблицы “ orders”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор заказа | int |
| order\_name | Название заказа | varchar(50) |
| order\_description | Описание заказа | varchar(1000) |
| order\_time | Крайний срок заказа | datetime |
| order\_active | Статус заказа | bool |
| order\_price | Цена заказа | int |
| order\_client | Внешний ключ, для клиента | bigint |

1. Таблица “ inventory” – содержит информацию о клиентах.

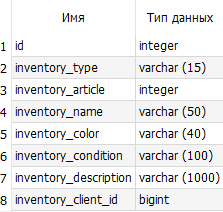


Рис 2.6. Таблица “ inventory”

Описание полей таблицы “ inventory”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор вещи | int |
| inventory\_type | Тип вещи | varchar(15) |
| inventory\_article | Артикул вещи | int |
| inventory\_name | Название вещи | varchar(50) |
| inventory\_color | Цвет вещи | varchar(40) |
| inventory\_condition | Состояние вещи | varchar(100) |
| inventory\_description | Для описания | varchar(1000) |
| inventory\_client | Внешний ключ, для клиента | bigint |

1. Таблица “ sales” – содержит информацию о клиентах.

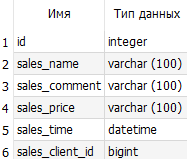


Рис 2.6. Таблица “ sales”

Описание полей таблицы “ sales”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор продажи | int |
| sales\_name | Название продажи | varchar(100) |
| sales\_comment | Описание продажи | varchar(100) |
| sales\_time | Время продажи | datetime |
| sales\_price | Цена продажи | varchar(100) |
| sales\_client | Внешний ключ, для клиента | bigint |

1. Таблица “ review” – содержит информацию о клиентах.

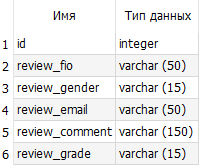


Рис 2.6. Таблица “ review”

Описание полей таблицы “ review”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор отзыва | int |
| review\_fio | Фио | varchar(50) |
| review\_gender | Пол | varchar(15) |
| review\_email | Почта | varchar(50) |
| review\_comment | Отзыв | varchar(150) |
| review\_grade | Оценка | varchar(15) |

1. Таблица “ application” – содержит информацию о клиентах.

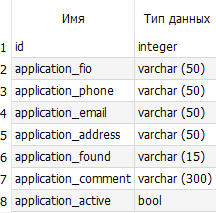


Рис 2.6. Таблица “ application”

Описание полей таблицы “ application”

|  |  |  |
| --- | --- | --- |
| Название столбца | Описание | Тип данных |
| id | Уникальный идентификатор заявки | int |
| application\_fio | Фио | varchar(50) |
| application\_phone | Телефон | varchar(50) |
| application\_email | Почта | varchar(50) |
| application\_address | Адрес | varchar(50) |
| application\_found | Откуда нашел? | varchar(15) |
| application\_comment | Заявка | varchar(300) |
| application\_active | Статус | bool |

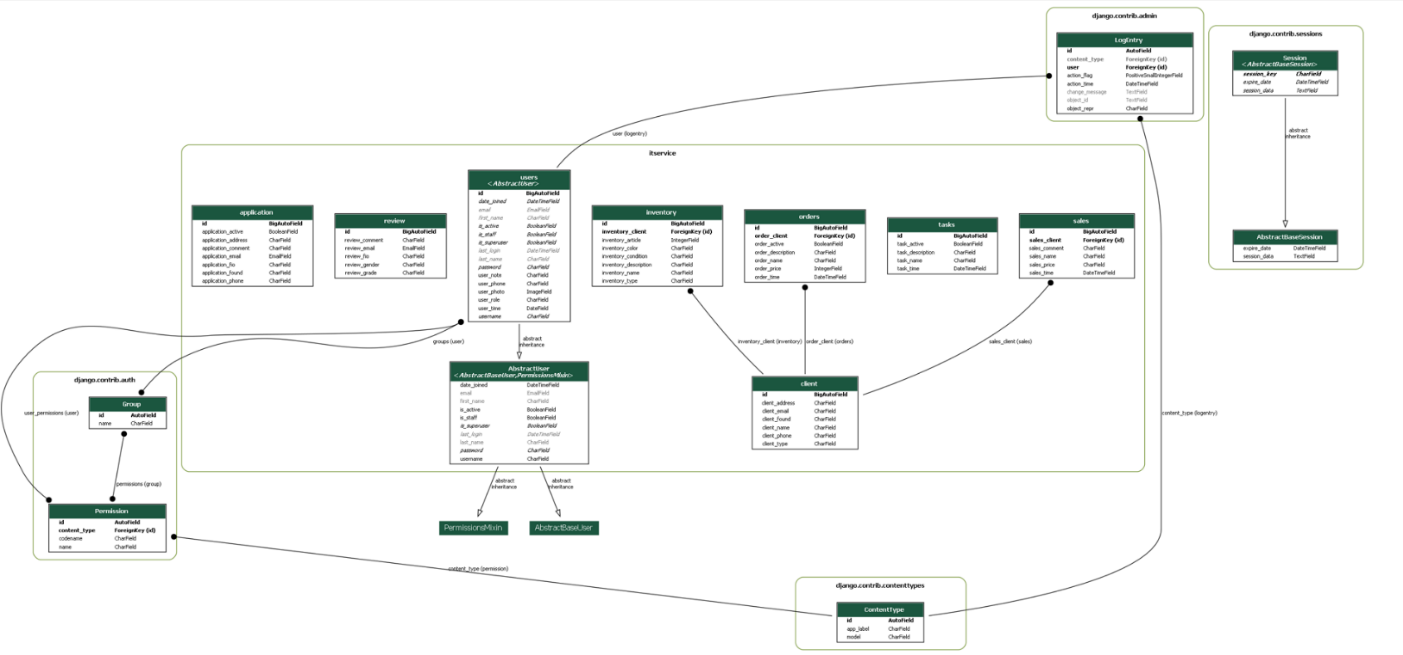


Рис 2.7. ER-диаграмма базы данных программного продукта.

**Выводы второй главы:**

Подводя промежуточные итоги, можно сказать, что разработано следующее:

* представлен вариант использования системы;
* спроектирована архитектура системы;
* разработан графический интерфейс системы;
* реализована база данных программного продукта.

**3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

## **3. 1. Порядок разработки программных модулей**

При создания программного модуля желательно следовать порядку, представленному ниже:

* анализ классификации модуля, выбор языка программирования;
* разработка алгоритма и структуры данных;
* программирование программного модуля;
* шлифовка текста программного модуля;
* проверка программного модуля;
* компиляция программного модуля.

После согласования всех требований, задач и функционала программного продукта, построения всех диаграмм по стандартам UML и IDEF, разработки базы данных необходимо выполнить разработку самого программного продукта.

Проект выполнен при использовании языка программирования Python с Фреймворком Django в среде разработки PyCharm (Рис. 3.1).

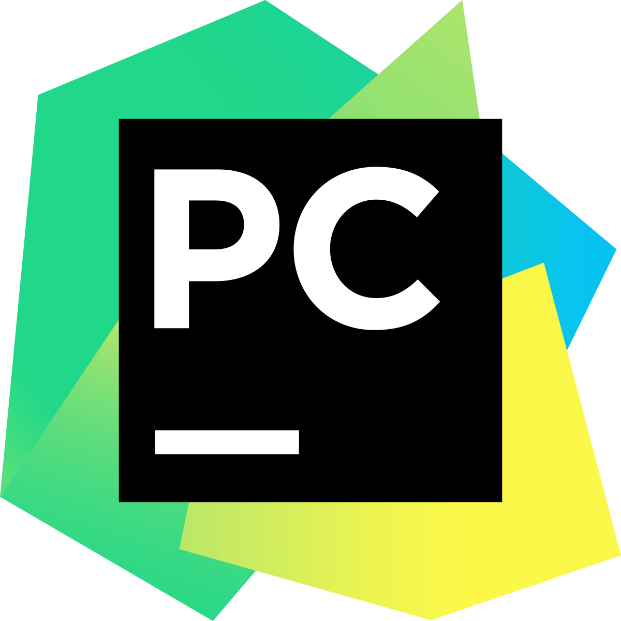


Рис 3.1. Среда разработки PyCharm

## **3. 2. Структурное программирование**

При разработке модуля на языке программирования Python важно помнить, что программа должна быть понятна не только компьютеру, но и людям, которые будут ее использовать и поддерживать. Это включает в себя разработчиков, тестировщиков и системных администраторов, которые будут заниматься отладкой и редактированием модуля.

В современных и новых языках программирования существует множество инструментов, которые могут запутать логику работы модуля и привести к тому, что реализованный модуль будет сложен для понимания и использования. Поэтому при выборе языка программирования и подхода к разработке модуля необходимо учитывать конкретную дисциплину программирования и основываться на мерах, которые обеспечат безопасность и понятность работы модуля.

Структурное программирование - это методология программирования, которая основывается на использовании структурных блоков для организации кода. Она подразумевает разделение программы на небольшие управляющие структуры, такие как последовательность, ветвление и циклы. Эти структуры могут быть объединены в модули, что позволяет создавать более крупные и сложные программы. Главной целью структурного программирования является создание кода, который легко читается, тестируется и сопровождается, а также минимизация количества ошибок в программе.

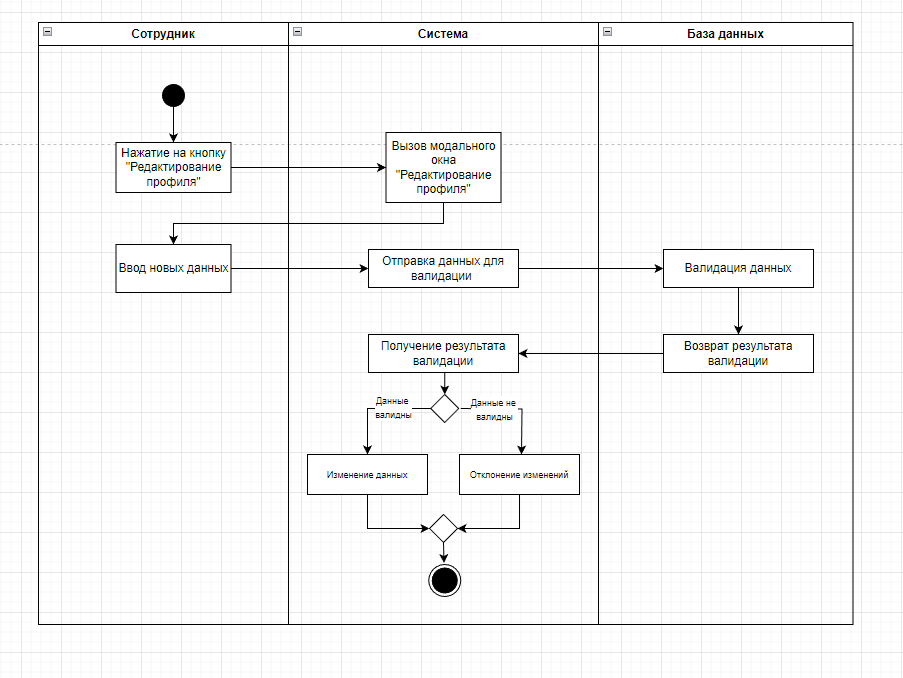


Рис. 3.2. Процесс редактирования данных сотрудника

Сотрудник, который зарегистрирован в системе, изменяет свои данные, например, меняет фотографию профиля, система принимает данные и производит проверку. Если данные не валидны, то все изменения отклоняются и все возвращается к исходным данным, иначе система сохраняет изменения в базе данных.

## **3. 3. Реализация серверной части**

Python - это язык программирования, который используется для написания серверной части системы. PyCharm - это интегрированная среда разработки (IDE), которая поможет создать и отладить код.

Фреймворк Django предоставляет набор инструментов для быстрого и удобного создания веб-приложений. Он позволяет легко работать с базами данных, формами, авторизацией пользователей и другими функциями, которые необходимы для создания полноценного веб-сайта.

Sqlite - это легкая встроенная база данных, которую можно использовать для начала работы с системой. Она не требует установки дополнительного ПО, что делает ее удобной для использования в небольших проектах.

Таким образом, можно использовать Python с помощью PyCharm и фреймворка Django для создания веб-приложений, которые будут хранить свои данные в базе данных Sqlite.

Конечная цель создания моделей в Django - это определение структуры таблиц в базе данных, которые будут хранить данные вашего приложения. Модели в Django - это классы Python, которые наследуются от базового класса «django.db.models.Model». Каждый атрибут класса-модели соответствует полю таблицы в базе данных. В файле models.py создаются модели, которые с помощью миграций перенесутся в базу данных (Рис. 3.3.)

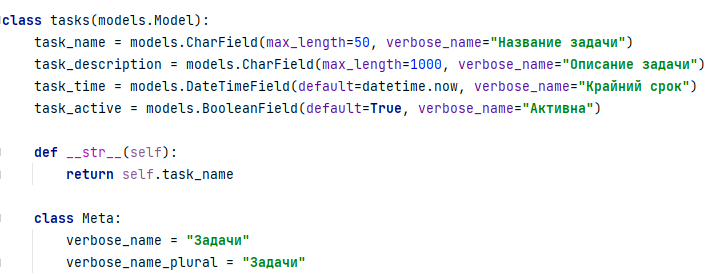


Рис. 3.3. Создание модели на примере класса «tasks»

С помощью «python manage.py makemigrations» мы создаем файл миграции, и он сохраняется в папке проекта «migrations». При успешном создании миграций, с помощью команды «python manage.py migrate» мы переносим наши миграции в базу данных (Рис 3.4.)

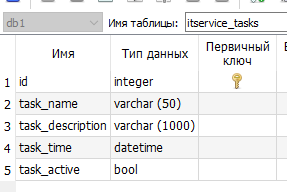


Рис. 3.4. Созданная таблица в бд

## **3. 4. Организация взаимодействия с БД**

Организация взаимодействия моделей базы данных с сайтом на Python в PyCharm с Django заключается в том, что в приложении Django вы можете определить модели базы данных, которые представляют данные, которые нужно хранить. Эти модели могут быть связаны между собой, чтобы создать более сложные структуры данных.

Далее, когда Сотрудник делает запрос на сайт на Python, Django будет использовать модели базы данных для получения необходимых данных из базы данных и передавать их в виде объектов Python веб-страницам. Когда пользователь отправляет данные с веб-страницы, Django использует модели базы данных для сохранения этих данных в базу данных.

Таким образом, организация взаимодействия моделей базы данных с сайтом на Python в PyCharm с Django обеспечивает эффективное хранение данных, быстрый доступ к ним и простое управление данными веб-приложения.

Способ организации взаимодействия моделей базы данных с сайтом на примере страницы «Задачи». В файле views.py создается представление страницы, так же импортируются все модели из models.py, далее создается представление для страницы, в которой мы передаем информацию о конкретной модели (Рис. 3.5.).

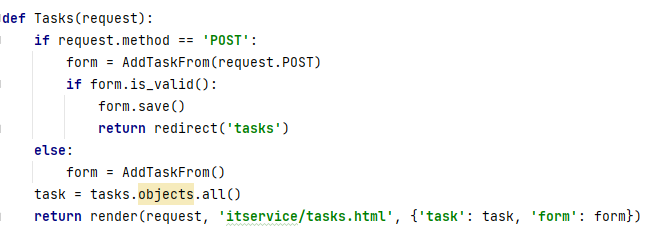


Рис. 3.5. Представление для страницы Задач

После этого, уже в файле tasks.html мы обращаемся к модели по имени как к массиву значений и берем оттуда то, что нам нужно



Рис. 3.6. Обращение к модели task

**Выводы третьей главы:**

Подводя итоги вышесказанному необходимо отметить следующее:

* Описан порядок разработки программных модулей;
* Приведен пример структурного программирования;
* Реализована серверная часть;
* Предоставлена организация взаимодействия с базой данных системы.

**4. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

После разработки программного продукта следует начать его тестировать. В этой главе отображена хронология изменений, исправлений и доработок программного обеспечения, проверена разработанного ПО, определены объекты тестирования, а также описан процесс тестирования.

**4.1. История изменений**

Таблица 4.1. Версионность программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Автор** | **Описание изменений** |
| 21.04.23 | 1.1 | Евгений Самойлов | Создание проекта, добавление виртуального пространства, установка всех необходимых инструментов. |
| 23.04.23 | 1.2-1.3 | Евгений Самойлов | Создание главного шаблона для всех страниц сайта. |
| 27.04.23 | 1.4-1.5 | Евгений Самойлов | Создание моделей базы данных, а так же добавление новых вкладок. |
| 01.05.23 | 1.6 | Евгений Самойлов | Проработка стилей ранее добавленных вкладок. |
| 04.05.23 | 1.7 | Евгений Самойлов | Подключение ко вкладкам базы данных и реализация вывода данных на сайте. |
| 05.05.23 | 1.8 | Евгений Самойлов | Создание модальных окон для добавления новых данных в БД. |
| 10.05.23 | 1.9 | Евгений Самойлов | Добавление стилей и модальных окон на все вкладки. |
| 11.05.23 | 2.0 | Евгений Самойлов | Создание вкладки для редактирования данных пользователя. |
| 14.05.23 | 2.1 | Евгений Самойлов | Проработка стилей, так же добавление возможности изменять статус данных. |
| 16.05.23 | 2.2-2.3 | Евгений Самойлов | Реализация возможности добавления новых сотрудников в систему;  Окно логина в систему;  Проработка стилей. |
| 19.05.23 | 2.4 | Евгений Самойлов | Создание окон для Заявок и Отзывов;  Добавление новых таблиц в бд. |
| 23.05.23 | 2.5 | Евгений Самойлов | Мелкие доработки;  Добавление Фавикона;  Создание ER диаграммы. |

**4.2. Терминология**

Список терминов и определений, используемых далее в документе и в тестировании в целом:

* Производительность - количество выполняемых операций за период времени (N операций за M часов);
* Тестирование стабильности – это проведение тестирования со средней нагрузкой в течение длительного времени. Выявляет проблемы связанные с утечками памяти и некорректными настройками ПО;
* Архитектура “клиент - сервер” – это сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами;
* Нагрузка - совокупное выполнение операций на общем ресурсе (тр./сек, хитов/сек);
* Виртуальный пользователь – программный процесс, циклически выполняющий моделируемые операции;
* Стратегия тестирования — это план проведения работ по тестированию системы или её модуля, учитывающий специфику функциональности и зависимости с другими компонентами системы и платформы;
* Позитивное тестирование – это тестирование с применением сценариев, которые соответствуют нормальному поведению системы;
* Негативное тестирование - это применение сценарий, которые соответствуют внештатному поведению тестируемой системы;
* Адаптивность — это способность сайта “подстраиваться” под различные технические условия (под размеры экрана);
* Тестирование методом “открытого ящика” – это метод тестирования программного продукта, который предполагает, что внутренняя структура, устройство, реализация системы известны тестировщику.

**4.3. Стратегия тестирования**

Стратегия тестирования программного обеспечения заключается в использовании метода тестирования “открытого ящика” для анализа и оценки защиты информационных потоков данных, а также отслежка выходных в входных данных на прецедент ошибки, пустого значения.

**4.4. Определение объектов тестирования**

Таблица 4.2. Определение объектов тестирования

| **Тесты** | **Объект тестирования** |
| --- | --- |
| Функциональное тестирование | * Работа системы авторизации; * Работа системы регистрации новых сотрудников; * Работа системы добавления новых задач и заказов; * Работа системы добавления новых продаж; * Работа системы добавления клиентов и устройств; * Работа системы добавления заявок и отзывов; * Работа системы скачивания документов; * Работа системы редактирования профиля; * Работа системы изменения статуса элементов; |
| Тестирование производительности | * Скорость перехода по формам; * Скорость сохранения данных в БД; * Скорость отображения информации из БД |
| Нагрузочное тестирование | * Попытка добавления огромного количества продаж, а далее скачивание сформированного файла Excel; * Изменение фотографии сотрудника на фотографию с очень большим разрешением; |
| Тестирование совместимости | * Корректное отображение программы в разных версиях ОС Windows (8, 10); * Корректное отображение программы на различных соотношениях сторон экрана: 4:3, 16:10, 16:8; * Корректное отображение программы на различных разрешениях экрана: 1024x768, 1280x720, 1600x900, 1920x1080, 2560x1600. |
| Позитивные тесты | * Добавление сотрудника, используя почту, которой еще нет в системе; * Попытка изменения фотографии сотрудника на другую, используя форматы .jpg и .png; * Добавление заявки длинной не более 150 символов. |
| Негативные тесты | * Добавление сотрудника, используя почту, которая уже есть в системе; * Попытка изменения фотографии сотрудника на другую, используя видео, .gif, .txt и т.д.; * Добавление заявки длинной более 150 символов. |
| Модульное тестирование | * Тестирование возможностей Администратора/Менеджера; * Тестирование возможностей Техника/Программиста * Тестирование возможностей Гостя |
| Интеграционное тестирование | * Переход со стартовой страницы на страницу авторизации; * Переход со страницы авторизации на страницу добавления сотрудников; * Переход со страницы добавления сотрудников на страницу редактирования профиля; |
| Системное тестирование | * Попытка обойти средства защиты программы; * Скорость отклика программы; * Способность системы эффективно использовать оперативную память. |

**4.5. Архитектура тестируемой системы**

Правило функционирования двухуровневой архитектуры содействия клиент-сервер заключается в обработке запросов, происходящих на одной станции ЭВМ, без использования иных ресурсов. Данная архитектура выражает серьёзные требования к коэффициент полезного действия сервера, но является весьма устойчивой (рис. 4.1).

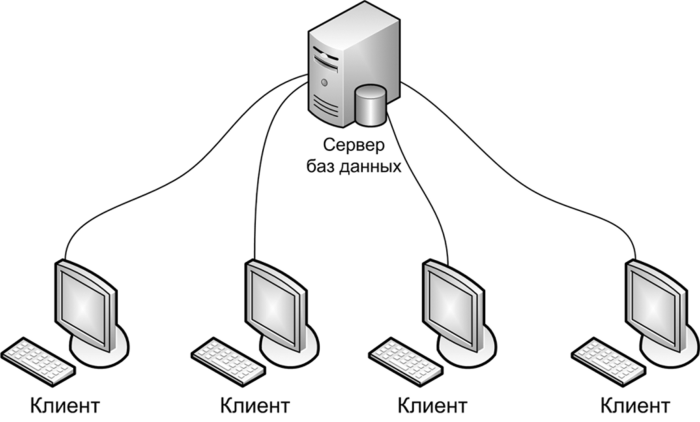


Рис 4.1. Двухзвенная архитектура тестируемой системы

Достоинство реализации модели клиент-сервер является то, что системный код клиентской и серверной программы отделён. Если говорить о локальных компьютерных сетях, то к достоинствам архитектуры клиент-сервер является низкие требования к станциям ЭВМ клиентов, т.к. большая часть вычислительных процессов производиться на сервере, а также архитектура клиент-сервер в меру подвижная и допускает администратору реализовать локальную сеть более безопасной.

**4.6. Описание процесса тестирования**

Таблица 4.3. Тестовые сценарии с указанием конкретных тестовых наборов

| **Название** | **Тестовые данные** | **Ожидаемый результат** | **Фактический результат** | **Статус** |
| --- | --- | --- | --- | --- |
| Вход с вводом существующего логина и пароля | Логин: admin  Пароль: static123  Роль сотрудника: Менеджер | Сотрудник успешно вошел в систему | Сотрудник успешно вошел в систему | Пройден |
| Вход с вводом несуществующего логина и пароля | Логин: user  Пароль: user  Роль сотрудника: неизвестна | Система оповещает о неправильном логине и пароле | Сотрудник не вошел в систему, сообщение получено | Пройден |
| Вход с вводом только логина | Логин: admin  Пароль: -  Роль сотрудника: Менеджер | Система оповещает о необходимости ввода пароля | Система оповещает о необходимости ввода пароля | Пройден |
| Вход с вводом только пароля | Логин: -  Пароль: static123  Роль сотрудника: Менеджер | Система оповещает о необходимости ввода логина | Система оповещает о необходимости ввода логина | Пройден |
| Кнопка “Выйти” | Кнопка на странице “Главная” | Выход пользователя из аккаунта | Выход пользователя из аккаунта | Пройден |
| Функционирование личного кабинета | Кнопка «Профиль» на странице | Открытие личного кабинета | Открытие личного кабинета | Пройден |
| Изменение статуса задачи | Переключатель на задаче | Изменение статуса задачи | Изменение статуса задачи | Пройден |
| Функционирование добавления новой задачи | Кнопка «Добавить задачу» на странице | Создание новой задачи | Создание новой задачи | Пройден |
| Удобство пользования. | Графический интерфейс системы. | Удобный и интуитивно понятный интерфейс системы. | Адаптивный программный продукт, единый стиль интерфейса, удобная и понятная навигация по системы. | Пройден |

**Выводы четвертой главы:**

Все вышесказанное доказывает, что:

* описана история изменения системы;
* разработана стратегия тестирования программного обеспечения;
* определены объекты тестирования;
* приведена архитектура тестируемой системы и описание процесса тестирования.

**5. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ**

**5.1. Определение защищаемых информационных активов**

**5.2. Вид и содержание персональных данных**

**5.3. Модель угроз ПНД**

**5.4. Возможные сценарии угроз**

**5.5. Организация защиты данных в проекте**

**5.6. Защита программного продукта**

**6. ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ НА ПРОГРАММНЫЙ ПРОДУКТ**

**6.1. Руководство пользователя**

Руководство пользователя создавалось по ГОСТ 19 и 34.

* ГОСТ 34 - Автоматизированные системы
* ГОСТ 19 - Единая система программной документации (ЕСПД)

Руководство пользователя расположено в приложении 3.

**6.2. Руководство администратора**

Руководство администратора не имеет собственного стандарта оформления. Выполняется этот документ по РД 50-34.698-90, содержащему общие требования к содержанию документации на автоматизированные системы.

Руководство администратора расположено в приложении 4.

**6.3. Определение качества программного продукта по метрикам**

Код файла sales.html

|  |  |  |
| --- | --- | --- |
| № строки | Строка кода | Номер сегманта |
| 1 | {% extends 'itservice/main.html' %} | 1 |
| 2 | {% block content %} |
| 3 | <div class="header-name"> |
| 4 | Продажи |
| 5 | </div> |
| 6 | <div class="help"> |
| 7 | <span class="tooltip" data-title="Это окно ваших продаж. |
| 8 | Продавайте товары, отслеживайте их оборот и анализируйте доход.">Помощь</span><!--Окно помощи--> |
| 9 | </div> |
| 10 | {% endblock %} |
| 11 | {% block content1 %} | 2 |
| 12 | <script><!--Скрипт для загрузки таблицы--> |
| 13 | const Download\_table = ()=> { |
| 14 | // xmlHttp.open("GET", "{% url 'sales-down' %}" , false); |
| 15 | // xmlHttp.send(); |
| 16 | window.location="{% url 'sales-down' %}" |
| 17 | } |
| 18 | </script> |
| 19 | <div class="main-window"> |
| 20 | {% if user.user\_role == 'Администратор' or user.user\_role == 'Менеджер' %} |
| 21 | <button class="btn" id="open">Добавить продажу</button><!--Кнопка добавления продажи--> | 3 |
| 22 | {% else %} |
| 23 | <div class="order\_"></div> |
| 24 | {%endif%} |
| 25 | <div class="modal" id="modal\_task"><!--Модальное окно добавления продажи--> |
| 26 | <div class="modal\_box"> |
| 27 | <div class="modal\_header"> |
| 28 | <h1>Добавление продажи</h1> |
| 29 | <button class="but\_close" id="close">Закрыть</button><!--Кнопка закрытия модального окна--> |
| 30 | </div> |
| 31 | <form action="{% url 'sales' %}" method="post"> | 4 |
| 32 | {% csrf\_token %} <!--Вывод формы добавления продажи--> |
| 33 | {{form.non\_field\_errors}} |
| 34 | <p> |
| 35 | <label for="id\_sales\_name">Наименование работы:</label> |
| 36 | <input class="input" type="text" name="sales\_name" maxlength="100" required="" id="id\_sales\_name"> |
| 37 | </p> |
| 38 | <p> |
| 39 | <label for="id\_sales\_comment">Комментарий:</label> |
| 40 | <input class="input" type="text" name="sales\_comment" maxlength="100" required="" id="id\_sales\_comment"> |
| 41 | </p> | 5 |
| 42 | <p> |
| 43 | <label for="id\_sales\_price">Цена:</label> |
| 44 | <input class="input" type="text" name="sales\_price" maxlength="100" required="" id="id\_sales\_price"> |
| 45 | </p> |
| 46 | <p> |
| 47 | <label for="id\_sales\_time">Время продажи:</label> |
| 48 | <input class="input" type="datetime-local" name="sales\_time" value="" required="" id="id\_sales\_time"><input type="hidden" name="initial-sales\_time" value="2023-05-12 18:06:18" id="initial-id\_sales\_time"> |
| 49 | </p> |
| 50 | <p> |
| 51 | <label>Клиент:</label> | 6 |
| 52 | {{form.sales\_client}} |
| 53 | </p> |
| 54 | <button class="but" type="submit">Добавить</button><!--Кнопка добавления продажи--> |
| 55 | </form> |
| 56 | </div> |
| 57 | </div> |
| 58 | <div class="ok"></div> |
| 59 | <div class="warpper"> |
| 60 | <div class="panels"> |
| 61 | <button class="btn\_down" onclick="Download\_table()" >Скачать таблицу</button><!--Кнопка для загрузки Excel файла с продажами--> | 7 |
| 62 | <div class="panel-title">Продажи</div><!--Таблица продаж--> |
| 63 | <table class="table"> |
| 64 | <thead> |
| 65 | <tr> |
| 66 | <th>Наименование работы</th> |
| 67 | <th>Комментарий</th> |
| 68 | <th>Цена</th> |
| 69 | <th>Время продажи</th> |
| 70 | <th>Клиент</th> |
| 71 | </tr> | 8 |
| 72 | </thead> |
| 73 | <tbody> |
| 74 | {% for s in sale%} |
| 75 | <a href="{% url 'sales' %}"> |
| 76 | <tr> |
| 77 | <td>{{s.sales\_name}}</td> |
| 78 | <td>{{s.sales\_comment}}</td> |
| 79 | <td>{{s.sales\_price}}</td> |
| 80 | <td>{{s.sales\_time}}</td> |
| 81 | <td>{{s.sales\_client}}</td> | 9 |
| 82 | </tr> |
| 83 | </a> |
| 84 | {% endfor %} |
| 85 | </tbody> |
| 86 | </table> |
| 87 | </div> |
| 88 | </div> |
| 89 | </div> |
| 90 | {% endblock %} |

Таблица 6.1. Расчет метрик стилистики и понятности системы.

|  |  |  |
| --- | --- | --- |
| Количество строк программы | 90 | |
| Строк в каждом сегменте | 10 | |
| Кол-во сегментов программы | 9 | |
|  | Количество  комментариев |  |
| 1 сегмент | 1 | 1 |
| 2 сегмент | 1 | 1 |
| 3 сегмент | 3 | 1 |
| 4 сегмент | 1 | 1 |
| 5 сегмент | 0 | -1 |
| 6 сегмент | 1 | 1 |
| 7 сегмент | 2 | 1 |
| 8 сегмент | 0 | -1 |
| 9 сегмент | 0 | -1 |
|  |  | 3 |

**Расчет:**

Сегмент 1. F1 = sgn(1/9 – 0,1) = 1

Сегмент 2. F2 = sgn(1/9 – 0,1) = 1

Сегмент 3. F3 = sgn(3/9 – 0,1) = 1

Сегмент 4. F4 = sgn(1/9 – 0,1) = 1

Сегмент 5. F5 = sgn(0/9 – 0,1) = -1

Сегмент 6. F6 = sgn(1/9 – 0,1) = 1

Сегмент 7. F7 = sgn(2/9 – 0,1) = 1

Сегмент 8. F8 = sgn(0/9 – 0,1) = -1

Сегмент 9. F9 = sgn(0/9 – 0,1) = -1

**Вывод:**

Так как уровень комментированности программы F=3, а число сегментов n=9, то условие нормального уровня комментированности F=n не выполняется. При F<n уровень комментированности программы считается низким.

**Выводы шестой главы:**

Подводя итоги вышесказанному необходимо отметить следующее:

* Рассчитана метрика стилистики и понятности программы;
* Разработано руководство пользователя и администратора.

**7. МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ТРУДА**

Техника безопасности при работе с ПК на предприятии предусматривает наличие инструкции, в которой описаны обязательные требования к рабочему месту и процессу использования техники. Эти правила общие для всех организаций, их выполнение контролируется соответствующими органами государственного аппарата.

Основными нормативными документами, которые содержат обязательные требования, можно считать:

* Трудовой кодекс;
* Санитарно-гигиенические нормы и правила СанПиН 2.2.2/2.4.1340-03, утвержденные постановлением Главного государственного санврача от 13 июня 2003 года N 118;
* Межгосударственный стандарт ГОСТ 12.0.003-2015, который содержит список основных вредных и опасных производственные факторов и их классифицирующие признаки;
* Типовая инструкция ТОИ Р-45-084-01, в которой содержатся основные правила для работников, использующих компьютер;
* Другие нормативные документы.

Требования перечисленных правовых актов, в которых содержатся общие правила защиты персонала при выполнении трудовых обязанностей и техника безопасности во время работы за компьютером, обязательны к исполнению всеми работниками этой категории.

На специалиста, работающего за компьютером, могут воздействовать вредные факторы, такие как:

* Высокая температура, характерная для отдельных элементов компьютерной техники и создающая общий повышенный температурный фон в рабочем помещении;
* Значительный уровень зрительный нагрузок, испытываемых работником;
* Повышенный уровень напряженности электрического и магнитного полей, генерируемых работающей техникой;
* Неравномерность распределения яркости в поле зрения;
* Другие факторы.

Чтобы донести до работника требования, которые устанавливает техника безопасности при использовании компьютера, работодатель обязан организовать инструктаж. Он проводится на основании специально составленной инструкции, которую обычно разрабатывает специалист по охране труда или, непосредственно, руководитель работника. Проведение инструктажа нужно организовать до начала самостоятельной работы: специалист допускается к работе только после его прохождения. Инструкция-памятка по технике безопасности за компьютером должна быть выдана каждому работнику в бумажном или электронном виде.

Техника безопасности в начале работы на компьютере требует выполнения следующих операций:

1. Проверить исправность элементов электросистемы, обеспечивающей питание компьютера, включая электропроводку, выключатели, вилки и розетки, при помощи которых аппаратура подключается к сети;
2. Проконтролировать заземление компьютера;
3. Проверить его работоспособность.

При выполнении работ нужно соблюдать следующие правила:

1. Запрещается класть на корпус и дисплей компьютера посторонние предметы, прикасаться к элементам аппаратуры мокрыми руками, производить чистку корпуса оборудования, находящегося под напряжением, располагать технику близко к жилищно-коммунальным инженерным системам;
2. В случае обнаружения неисправности компьютера немедленно прекратить работу и сообщить об этом непосредственному руководителю;
3. Избегать частого и необоснованного включения и выключения компьютера во время работы.

Требования к расположению работника за компьютером нацелены на обеспечение его комфорта в течение всей рабочей смены и отсутствие негативных следствий длительной работы. Они действуют для любых рабочих мест и включают следующие правила:

1. Использование компьютерной мебели, отвечающей нормам СанПиН 2.2.2/2.4.1340-03;
2. Отказ от скрещивания конечностей, способного затруднить кровообращение;
3. Соблюдение расстояние до монитора компьютера не меньше 45 сантиметров;
4. Правильная установка освещения, которое не должно светить в глаза и оставлять блики на рабочем мониторе;
5. Клавиатура размещается за 20-30 сантиметров от края стола;
6. Стул стоит таким образом, чтобы спина лишь немного упиралась в его спинку;
7. Локти согнуты под прямым углом, а в кистях рук, лежащих на столе, не чувствуется напряжения;
8. Локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки;
9. Ноги упираются в твердую поверхность, распрямлены вперед, а не подогнуты под себя;

Длительность непрерывного труда и отдыха для сотрудников, постоянно работающих за персональной офисной техникой, регламентирована на законодательном уровне. Соответствующие нормы приводятся в СанПиН 2.2.2/2.4.1340-03, который устанавливает обязательную продолжительность перерывов в такой работе. Они зависят от длины рабочей смены сотрудника и определяются так: при 8-часовой рабочей смене в зависимости от тяжести и напряженности труда работника общая продолжительность перерывов за рабочий день должна составлять от 50 до 90 минут; при 12-часовой рабочей смене – от 80 до 140 минут.

**Выводы седьмой главы:**

В данной части выпускной квалификационной работы были рассмотрены мероприятия по технике безопасности и выделены требования охраны труда.

Так же были рассмотрены мероприятия противопожарной защиты, были определены категория и степень пожарной безопасности объекта и приняты решения для уменьшения вероятности пожара и его предотвращение.

**8. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА**

## **8.1 Расчет затрат на разработку программного обеспечения**

Затраты на оплату (ЗОТ) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной паты

Затраты на оплату труда разработчика ПО складывается из двух

составляющих: основной заработной платы и дополнительной заработной платы.

Таблица 8.1

Расчет текущих тарифных ставок оплаты труда персонала

|  |  |  |  |
| --- | --- | --- | --- |
| Категории работников- исполнителей  работ | Тарифная ставка  руб./чел.-ч | Количество рабочих часов | Всего |
| Архитектор программного обеспечения | 427,00 | 7 | 2 989 |
| US – дизайнер (инженер) | 395,00 | 15 | 5 925 |
| Аналитик программного обеспечения | 361,00 | 8 | 2 888 |
| Специалист -тестировщик | 328,00 | 5 | 1 640 |
| Администратор баз данных | 296,00 | 8 | 2 368 |
| Веб-программист | 238,00 | 30 | 7 140 |
| IT-администратор | 215,00 | 20 | 4 300 |
| ИТОГО | х | ∑t =93 | Зосн = 27 430 руб. |

Дополнительная заработная плата составляет 20% от основной заработной платы, рассчитывается по формуле (8.1):

Здоп = (20 /100) ⋅ Зосн, (8.1)

Здоп = (20 /100) ⋅ 27 430 = 5 486 руб.

Затраты на оплату труда разработчика ПО в рублях (ФОТ - фонд оплаты труда) представляет собой сумму основной и дополнительной заработной платы и рассчитывается по формуле (8.2):

ФОТ = Зосн + Здоп, (8.2)

ФОТ = Зосн + Здоп = 32 916 руб.

Отчисления от фонда оплаты труда включают страховые взносы:

Таблица 8.2

Страховые взносы с зарплаты

|  |  |
| --- | --- |
| **Взносы** | **Ставка** |
| Пенсионное страхование | 22% |
| Социальное страхование | 2,9% |
| Медицинское страхование | 5,1% |
| **Всего** | 30 % |

Отчисления от фонда оплаты труда рассчитываются по формуле (8.3):

ОФОТ = (30 / 100) ⋅ ФОТ, (8.3)

ОФОТ = (30 / 100) ⋅ 32 916 = 9 875 руб.

Затраты на оплату труда разработчика ПО рассчитываются по формуле (8.4):

ЗОТ = ФОТ + ОФОТ, (8.4)

ЗОТ = 32 916 + 9 875 = 42 791 руб.

## **8.2. Эксплуатационные затраты на оборудование**

Суммарная годовая стоимость эксплуатационных затрат Cэз рассчитывается по формуле (8.5):

Cэз = СТО +СК +СЭЭ + Агод,  (8.5)

где СТО - затраты на техническое обслуживание и ремонт оборудования, руб.,

СК – затраты на комплектующие, руб.,

СЭЭ - годовая стоимость электроэнергии, руб.,

Агод - годовые амортизационные отчисления, руб.

Затраты на техническое обслуживание и ремонт (СТО) составляют 3 % от

стоимости оборудования по формуле (8.6):

СТО = 0,03 ⋅ Собор, (8.6)

СТО = 0,03 ⋅ 105 324 = 3 160 руб.

Затраты на комплектующие принимаются в размере 2% от стоимости оборудования по формуле (8.7):

СК = 0,02 ⋅ Собор, (8.7)

СК = 0,02 ⋅ 105 324 = 2 107 руб.

Амортизационные отчисления производятся по установленным нормам амортизации, выражаются, в процентах к стоимости оборудования и рассчитываются по формуле (8.8):

Агод = Собор ⋅ (НА / 100), (8.8)

Агод = 105 324 ⋅ (20 / 100) = 21 065 руб.,

где Cобор – стоимость компьютера;

НА – норма амортизации, которая рассчитывается по формуле (8.9):

НА = 100 / Тнорм,, (8.9)

НА = 100 / 5 = 20.

где Тнорм – нормативный срок службы (для персонального компьютера Тнорм = 5 лет)

Стоимость электроэнергии вычисляется по формуле (8.10):

СЭЭ = M ⋅ kз ⋅ Fэф ⋅ Сквт.ч ⋅ КС, (8.10)

СЭЭ = 0,5 ⋅ 0,6 ⋅ 2000 ⋅ 7,56 ⋅ 1,05 = 4 763 руб.

где M – мощность компьютера, Квт (емкость аккумулятора);

kз – коэффициент загрузки, учитывающий использование оборудования по

времени (0,8);

Fэф – эффективный фонд рабочего времени (от 1 700 – 2000 час);

Cквт.ч – стоимость 1 квт-час электроэнергии (от 7,27- 7,79 руб./квт по состоянию на март 2023 года);

КС - коэффициент, учитывающий потери в сети (Kc=1,05).

Рассчитываем суммарную годовую стоимость эксплуатационных затрат Cэз используя формулу 8.5:

Cэз = 3 160 + 2 107 + 4 763 + 21 065 = 31 095 руб.

Cэз - суммарная годовая стоимость эксплуатационных затрат, вычисленный по формуле (8.5).

## **8.3. Затраты на материалы**

Возможен расчет затрат на материалы (ЗМ ) прямым счетом (по факту).

Таблица 8.3

Расчет затрат на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| **Материалы** | **Количество, шт.** | **Цена за единицу, руб.** | **Всего, руб.** |
| Оплата услуг провайдера | 2 | 600 | 1 200 |
| Источник бесперебойного питания | 1 | 5 210 | 5 210 |
| Бумага | 1000 | 2 | 2 000 |
| Картридж | 2 | 1 500 | 3 000 |
| Итого(ЗМ): х х | | | 11 410 |

**Накладные расходы**, связанные с управлением, организационными расходами и прочими дополнительными затратами, составляют 40% от фонда оплаты труда, вычисляются по формуле (8.11):

Cнакл = 0,4 ⋅ ФОТ, (8.11)

Cнакл = 0,4 ⋅ 32 916 = 13 166 руб.

## **8.4.** **Себестоимость разработки программного обеспечения**

Таблица 8.4

Стоимость программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **п/п** | **Наименование статей затрат** | **Обозначение** | **Сумма, руб.** |
| 1 | Затраты на материалы | ЗМ | 11 410 |
| 2 | Эксплуатационные затраты (ф.5) | Эз | 31 095 |
| 3 | Фонд оплаты труда (ф.2) | ФОТ | 32 916 |
| 4 | Отчисления от фонда оплаты труда (ф.3) | ОФОТ | 9 875 |
| 5 | Накладные расходы (ф.12) | Cнакл | 13 166 |
| 6 | Себестоимость ПО без учета расходов на сопровождение и адаптацию | п.1+п.2+п.3+п.4+п.5 | 98 462 |
| 7 | Расходы на сопровождение и адаптацию | 10% от п. 6 | 9 846 |
| 8 | Полная (плановая) себестоимость ПО | Спол.= п.6 + п.7 | 108 308 |

## **8.5. Расчет экономического эффекта разработчика и пользователя (заказчика) программного обеспечения**

**Экономический эффект у разработчика программного обеспечения.**

Заказчик оплачивает разработчику всю сумму расходов по проекту (полная себестоимость ПО из таблицы 8.4 с учетом прибыли разработчика и налога на добавленную стоимость с учетом качества, потребительских свойств продукции (ПО) и конъюнктуры рынка.

Таким образом, в дипломном проекте отпускная цена программного обеспечения, представляет собой не цену за единицу продукции, а цену проекта вместе с его исходными кодами и документацией, за которую его можно продать и получить определенную выгоду. Прогнозируемая отпускная цена ПО (ЦПО) с учетом НДС рассчитывается по формуле (8.12):

ЦПО = , (8.12)

где

ЦПО -отпускная цена программного обеспечения;

Сполн – полная (плановая) себестоимость ПО, руб., (из таблицы 4);

П – прибыль разработчика ПО, руб.;

СТНДС – ставка налога на добавленную стоимость (=20%), в %.

ЦПО = ((108 308 + 21 662) ⋅ (100 + 20))/100 = 155 964 руб.

Прибыль закладывается в цену исходя из уровня рентабельности (**устанавливается студентом самостоятельно**), расчет производится по формуле (8.13):

П = , (8.13)

где

П – прибыль, руб;

R – уровень рентабельности, % (в рамках дипломного проекта рекомендуемый уровень рентабельности ≈ **15-20-30%**).

Сполн – полная (плановая) себестоимость ПО, руб., (из таблицы 8.4).

П = (20 ⋅ 108 308)/100 = 21 662 руб.

Ввиду того, что программное обеспечение разрабатывается для одного объекта, в качестве экономического эффекта разработчика от реализованного программного обеспечения можно рассматривать чистую прибыль (ЧП), которая рассчитывается по формуле (8.14):

ЧП = , или ЧП = (8.14)

где

ЧП - чистая прибыль, руб.;

П – прибыль, руб;

СТП – ставка налогообложения прибыли составляет 20%. (100% - 20%)

ЧП = 21 662 ⋅ 0,8 = 17 330 руб.

**Вывод:** Таким образом, разработчик программного обеспечения может продать заказчику программное обеспечение по рассчитанной цене 155 964 руб., что покроет затраты на разработку ПО 108 308 руб. и обеспечит получение чистой прибыли при его реализации заказчику 17 330 руб.

**ЗАКЛЮЧЕНИЕ**

Цель дипломной работы - это автоматизированная информационная система для IT сервиса, разработанная на языке Python с использованием фреймворка Django в среде PyCharm. В результате работы был создан функциональный сайт, который позволяет сотрудникам быстро и удобно получать информацию, клиентам оставлять заявки на обслуживание и писать отзывы, а администраторам контролировать весь процесс работы.

В процессе анализа существующих решений были выявлены преимущества и недостатки конкурирующих систем. Было замечено, что проект отличается более простым и понятным интерфейсом, а также более широким спектром функций.

В ходе разработки были использованы современные технические решения, такие как Django ORM, JavaScript, jSON, Pandas и другие. Также была проведена тестирование приложения и исправлены найденные ошибки.

В результате работы были достигнуты все поставленные цели, и создана полноценная информационная система для IT сервисов. Выводы по результатам работы указывают на то, что проект имеет большой потенциал для дальнейшего развития и улучшения.

**СПИСОК ИСТОЧНИКОВ ИНФОРМАЦИИ**

1. Web и DHTML : учебное пособие / С. И. Солодушкин, 2018;
2. Макфарланд Д. М17 Новая большая книга CSS. — СПб.: Питер, 2021.
3. Использование библиотеки Django OAuth Toolkit в Django Python: журнал / под ред.: Неустроев Айаал Васильевич, 2017.
4. Сравнение времени отклика репликации mysql: журнал / под ред.: Зарипов А.К., 2021.
5. Робсон, Э. Изучаем HTML, XHTML и CSS : практическое руководство / Э. Робсон, Э. Фримен ; перевод В. В. Черника. - 2-е изд. - Санкт-Петербург : Питер, 2021.
6. Дронов, В. А. Django: практика создания Web-сайтов на Python : практическое руководство / В. А. Дронов. - Санкт-Петербург : БХВ-Петербург, 2016.
7. Рощин, С. М. Современные интернет-технологии. Семь главных трендов : научно-популярное издание / С. М. Рощин. - 2-е изд. - Москва : Дашков и К, 2022.
8. Никитченко, И. И. Основы web-технологий : учебное пособие / И. И. Никитченко, К. Н. Мезенцев, О. В. Зинюк. - Москва : РИО Российской таможенной академии, 2020.
9. Руководство по веб-фреймворку Django[Электронный ресурс] – режим доступ: <https://metanit.com/python/django>. Дата доступа: 14.05.23.
10. Уроки Django. [Электронный ресурс] – режим доступ: <https://itproger.com/course/django>. Дата доступа: 24.05.23.

**ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММНЫХ МОДУЛЕЙ**

Код файла home.css, используется на главной странице, которую видит неавторизованный пользователь:

body {

width: 100%;

height: 100%;

display: block;

font-weight: initial;

font-family: 'Trebuchet MS', Helvetica, sans-serif;

background: #e2e2e5;

}

.main-title{

width: 100%;

height: 20%;

background: #e2e2e5;

display: flex;

align-items: center;

justify-content: center;

font-size: 100px;

font-weight: bold;

}

.main{

width: 100%;

height: 80%;

background: #e2e2e5;

display: flex;

align-items: center;

justify-content: center

}

.btn{

background-color: whitesmoke;

width: 400px;

height: 80px;

margin: 0 auto;

display: block;

position: center;

font-size: 40px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding-top: 12px;

cursor: pointer;

color: #252930;

border: 4px solid #252930;

border-radius: 6px;

text-align: center;

box-shadow: 0px 0px 40px 10px rgba(0,0,0,0.40);

}

.btn:hover {

background-color: #252930;

color: white;

box-shadow: none;

}

.form\_app{

height: 80%;

font-size:40px;

}

.input\_a:focus{

border: 2px solid lightslategrey;

border-radius: 5px;

}

.input\_a{

font-size: 40px;

border-radius: 10px;

margin-bottom: 16px;

padding-left: 5px;

color: lightslategrey;

background-color: #F0F0F0;

/\*border: 2px solid lightslategrey;\*/

/\*border-radius: 5px;\*/

}

select{

color: lightslategrey;

background-color: #F0F0F0;

border-radius: 10px;

margin-bottom: 20px;

}

.butt{

background-color: whitesmoke;

width: 300px;

margin: 0 auto;

display: block;

position: center;

font-weight: bolder;

font-size: 40px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 4px solid #252930;

border-radius: 15px;

margin-top: 50px;

}

.butt:hover {

background-color: #252930;

color: white;

}

.butt-close{

padding-top: 10px;

text-align: center;

background-color: whitesmoke;

display: block;

position: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

width: 130px;

height: 50px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 40px;

margin-bottom: 20px;

}

.butt-close:hover {

background-color: #252930;

color: white;

}

.window{

height: 80%;

width: 100%;

}

.review{

display: block;

height: 100%;

width: 100%;

padding: 100px;

}

.grade{

color:white;

font-size: 20px;

display: block;

float: left;

margin: 30px;

padding: 40px;

width: 500px;

min-height: 280px;

border-radius: 40px;

background-color: darkgrey;

}

.grade\_header{

padding-right: 10px;

float: left;

}

.img{

width: 80px;

height: 80px;

border-radius: 40px;

}

.buttons{

display: inline-flex;

float:right;

}

.butt-rev{

margin-right: 30px ;

text-align: center;

background-color: whitesmoke;

display: block;

position: center;

font-size: 25px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

width: 150px;

height: 70px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 40px;

}

.butt-rev:hover {

background-color: #252930;

color: white;

}

Код файла login.сss, используется на странице входа в систему, которую видит неавторизованный сотрудник:

::selection {

background: #2D2F36;

}

::-moz-selection {

background: #2D2F36;

}

body {

background: white;

font-family: 'Trebuchet MS', Helvetica, sans-serif;

margin: 0;

}

.page {

background: #e2e2e5;

display: flex;

flex-direction: column;

height: 100%;

position: absolute;

place-content: center;

width: 100%;

}

@media (max-width: 767px) {

.page {

height: auto;

margin-bottom: 20px;

padding-bottom: 20px;

}

}

.container {

display: flex;

height: 370px;

margin: 0 auto;

width: 640px;

}

@media (max-width: 767px) {

.container {

flex-direction: column;

height: 630px;

width: 320px;

}

}

.left {

background: white;

height: calc(100% - 40px);

top: 20px;

position: relative;

width: 50%;

}

@media (max-width: 767px) {

.left {

height: 100%;

left: 20px;

width: calc(100% - 40px);

max-height: 270px;

}

}

.login {

font-size: 50px;

font-weight: 900;

margin: 50px 40px 40px;

}

.eula {

color: #999;

font-size: 17px;

line-height: 1.5;

margin: 40px;

}

.right {

background: #474A59;

box-shadow: 0px 0px 40px 16px rgba(0,0,0,0.22);

color: #F1F1F2;

position: relative;

width: 50%;

}

@media (max-width: 767px) {

.right {

flex-shrink: 0;

height: 100%;

width: 100%;

max-height: 350px;

}

}

svg {

position: absolute;

width: 320px;

}

.form {

margin-left: 40px;

margin-top: 20px;

position: absolute;

}

label {

color: #c2c2c5;

display: block;

font-size: 17px;

height: 16px;

margin-top: 10px;

margin-bottom: 10px;

}

input {

padding: 2px;

border-radius: 5px;

background: #2D2F36;

color: #f2f2f2;

font-size: 20px;

height: 30px;

line-height: 30px;

width: 100%;

}

input:focus {

box-shadow: inset 0 -2px 0 0 #c2c2c5;

}

input::-moz-focus-inner {

border: 0;

}

.btn{

margin-top: 30px;

background-color: white;

display: block;

position: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 40px;

box-shadow: 0px 0px 40px 7px rgba(0,0,0,0.40);

}

.btn:hover {

background-color: #252930;

color: white;

box-shadow: none;

}

.btn-close{

margin-top: 20px;

margin-left: 200px;

background-color: white;

display: block;

position: center;

font-size: 17px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

width: 80px;

height: 30px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 40px;

box-shadow: 0px 0px 40px 5px rgba(0,0,0,0.40);

}

.btn-close:hover {

background-color: #252930;

color: white;

box-shadow: none;

}

Код файла main.сss, используется на всех остальных страницах системы:

body {

width: 100%;

height: 100%;

display: flex;

font-weight: initial;

font-family: 'Trebuchet MS', Helvetica, sans-serif;

}

/\*Хедер\*/

.header{

box-shadow: 0 7px 7px -4px rgba(0, 0, 0, .2);

height: 6%;

min-height: 50px;

width: 100%;

background-color: white;

display: flex;

}

.header-name{

font-size: 25px;

padding-left: 20px;

padding-top: 10px;

font-weight: bold;

}

/\*Помощь на сайте, находится в левом верхнем углу\*/

.help{

position: absolute;

right: 0;

padding-right: 20px;

padding-top: 15px;

font-weight: bold;

font-size: 20px;

}

.tooltip{

right: 0;

border-bottom: 1px;

position: relative;

cursor: pointer;

}

.tooltip:before,

.tooltip:after {

display: block;

position: absolute;

opacity:0;

visibility: hidden;

margin-top: 15px;

transition: 0.7s;

}

.tooltip:before{

background-color: #252930;

content: attr(data-title);

color:lightgray;

font-weight: initial;

min-width: 200px;

min-height: 50px;

border-radius: 5px;

margin-left: -160px;

top:50px;

padding: 10px;

box-sizing: border-box;

text-align: center;

font-size: 20px;

}

.tooltip:after {

content: "";

}

.tooltip:hover:before{

visibility: visible;

opacity:1;

margin-top: 0;

}

/\*Стили сайтбара, он имеет фиксированное позиционирование \*/

.sidebar{

width: 145px;

height: 100%;

position: fixed;

min-width: 145px;

background-color: #252930;

color: lightgray;

}

.sidebar-name{

color:white;

font-size: 25px;

display: block;

line-height: 65px;

padding-left: 15px;

padding-top: 30px;

padding-bottom: 30px;

box-sizing: border-box;

border-top:1px solid rgba(255,255,255,.1);

border-bottom:1px solid black;

}

.sidebar-name p{

}

.sidebar ul a{

font-size: 20px;

display: block;

line-height: 65px;

padding-left: 20px;

box-sizing: border-box;

border-top:1px solid rgba(255,255,255,.1);

/\*border-bottom:1px solid rgba(255,255,255,.1);\*/

border-bottom:1px solid black;

transition: 0.4s;

}

ul li:hover a{

padding-left:30px;

}

/\*Стили блока main\*/

.main{

margin-left:145px;

display: inline-block;

width: 100%;

min-height: 100vh;

}

.main-form{

width: 100%;

}

/\*Здесь стили для страниц с двумя вкладками\*/

.warpper {

display: flex;

flex-direction: column;

}

.tab {

cursor: pointer;

padding: 10px 20px;

margin: 0 10px;

background: #252930;

display: inline-block;

color: #fff;

border-radius: 3px 3px 0 0;

box-shadow: 0 0 3px 1px #b0b0b0;

}

.panels {

box-shadow: 0 -6px 4px -4px rgba(0, 0, 0, .2);

background-color: white;

width: 100%;

height: 100%;

border-radius: 3px;

overflow: hidden;

padding: 20px;

}

.panel {

display: none;

animation: fadein 0.8s;

}

@keyframes fadein {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

.panel-title {

font-size: 1.5em;

font-weight: bold;

padding-bottom: 20px ;

}

.radio {

display: none;

}

#one:checked ~ .panels #one-panel,

#two:checked ~ .panels #two-panel {

display: block;

}

#one:checked ~ .tabs #one-tab,

#two:checked ~ .tabs #two-tab{

background: #fff;

color: #000;

border-top: 3px solid #252930;

}

.main-window{

padding-top: 20px;

}

/\*Стили для кнопки главного экрана, который доступен Гостю\*/

.btn{

background-color: transparent;

width: 300px;

margin: 0 auto;

display: block;

position: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.btn:hover {

background-color: #252930;

color: white;

}

/\*Стили для блоков задач и заявок\*/

.tasks {

margin: 0 auto;

display: block;

position: center;

list-style-type: none;

color: black;

margin-bottom: 30px;

font-size: 20px;

padding: 20px 20px 20px 20px;

min-height: 150px;

width: 70%;

min-width: 450px;

background-color: white;

box-shadow: 0 2px 4px black;

border-radius: 10px;

transition: all .3s ease-out;

}

.tasks\_appl {

margin: 0 auto;

display: flex;

position: center;

list-style-type: none;

color: black;

margin-bottom: 30px;

font-size: 20px;

padding: 20px 20px 20px 20px;

min-height: 150px;

min-width: 500px;

width: 70%;

background-color: white;

box-shadow: 0 2px 4px black;

border-radius: 10px;

transition: all .3s ease-out;

}

.task-inf h1{

padding-top: 15px;

font-size: 22px;

font-weight: bold;

}

.desc{

padding-top: 10px;

padding-right: 50px;

}

.applo{

width: 250px;

}

.desc1{

word-wrap: break-word;

min-width: 250px;

max-width: 800px;

font-size: 23px;

padding-top: 35px;

padding-left: 10px;

}

.desc2{

font-size: 15px;

padding-top: 10px;

padding-right: 50px;

}

.time{

padding-top: 10px;

display: flex;

justify-content: right;

color: #eb4f37;

}

.time\_2{

padding-top: 10px;

display: flex;

justify-content: right;

color: #44cc66;

}

/\* Стили чекбоксов у задач и заявок\*/

.toggler-wrapper {

display: block;

width: 45px;

height: 25px;

cursor: pointer;

position: relative;

}

.toggler-wrapper input[type="checkbox"] {

display: none;

}

.toggler-wrapper input[type="checkbox"]:checked+.toggler-slider {

background-color: #44cc66;

}

.toggler-wrapper .toggler-slider {

background-color: #ccc;

position: absolute;

border-radius: 100px;

padding-top:25px;

width: 100%;

height: 100%;

-webkit-transition: all 300ms ease;

transition: all 300ms ease;

}

.toggler-wrapper .toggler-knob {

position: absolute;

-webkit-transition: all 300ms ease;

transition: all 300ms ease;

}

.toggler-wrapper input[type="checkbox"]:checked+.toggler-slider {

background-color: transparent;

border-color: #44cc66;

}

.toggler-wrapper input[type="checkbox"]:checked+.toggler-slider:before {

-webkit-transform: translateY(0);

transform: translateY(0);

opacity: 0.7;

}

.toggler-wrapper input[type="checkbox"]:checked+.toggler-slider:after {

opacity: 0;

-webkit-transform: translateY(20px);

transform: translateY(20px);

}

.toggler-wrapper input[type="checkbox"]:checked+.toggler-slider .toggler-knob {

left: calc(100% - 19px - 3px);

background-color: #44cc66;

}

.toggler-wrapper .toggler-slider {

background-color: transparent;

border: 2px solid #eb4f37;

}

.toggler-wrapper .toggler-slider:before {

content: 'Выполнена';

position: absolute;

top:3px;

left:50px;

font-size: 75%;

text-transform: uppercase;

font-weight: 500;

opacity: 0;

-webkit-transition: all 300ms ease;

transition: all 300ms ease;

-webkit-transform: translateY(-20px);

transform: translateY(-20px);

}

.toggler-wrapper .toggler-slider:after {

content: 'Активна';

position: absolute;

top:3px;

left:50px;

font-size: 75%;

text-transform: uppercase;

font-weight: 500;

opacity: 0.7;

-webkit-transition: all 300ms ease;

transition: all 300ms ease;

}

.toggler-wrapper .toggler-knob {

width: calc(25px - 6px);

height: calc(25px - 6px);

border-radius: 50%;

left: 3px;

top: 3px;

background-color: #eb4f37;

}

/\* Стили модальных окон\*/

.modal{

position: fixed;

left:0;

top:0;

width: 100%;

height: 100vh;

z-index: 99998;

background-color: rgba(0,0,0,.3);

display: grid;

align-items: center;

justify-content: center;

overflow-y: auto;

visibility: hidden;

opacity: 0;

transition: all 0.4s;

}

.modal.open{

visibility: visible;

opacity: 1;

}

.modal\_box{

max-width: 700px;

min-height: 800px;

padding: 45px;

z-index: 1;

margin: 30px 15px;

background-color: white;

box-shadow: 0 2px 4px black;

border-radius: 10px;

}

.modal\_box h1{

margin-right: 30px;

margin-bottom: 20px;

font-weight: bold;

font-size: 40px;

}

.modal\_box p{

font-weight: bold;

font-size: 25px;

}

.modal\_header{

display: inline-flex;

}

/\*Стили для таблиц\*/

.table {

width: 100%;

border: none;

margin-bottom: 20px;

}

.table thead th {

font-weight: bold;

text-align: left;

border: none;

padding: 10px 15px;

background: #d8d8d8;

font-size: 17px;

}

.table thead tr th:first-child {

border-radius: 8px 0 0 8px;

}

.table thead tr th:last-child {

border-radius: 0 8px 8px 0;

}

.table tbody td {

text-align: left;

border: none;

padding: 10px 15px;

font-size: 15px;

vertical-align: top;

}

.table tbody tr:nth-child(even){

background: #f3f3f3;

}

.table tbody tr td:first-child {

border-radius: 8px 0 0 8px;

}

.table tbody tr td:last-child {

border-radius: 0 8px 8px 0;

}

.ok{

height: 45px;

}

/\*Стили для личного профиля\*/

.profile{

margin-top: 50px;

display: block;

list-style-type: none;

color: black;

margin-bottom: 30px;

font-size: 25px;

padding: 30px 30px 30px 30px;

width: 100%;

background-color: white;

box-shadow: 0 2px 4px black;

border-radius: 10px;

transition: all .3s ease-out;

}

.profile-title{

font-size: 30px;

font-weight: bold;

margin-bottom: 16px;

}

.input:focus{

border: 1px solid lightslategrey;

border-radius: 5px;

}

/\*Стили для input текста\*/

.input{

border-radius: 10px;

margin-bottom: 16px;

padding-left: 5px;

color: lightslategrey;

background-color: #F0F0F0;

/\*border: 2px solid lightslategrey;\*/

/\*border-radius: 5px;\*/

}

.profile .photo\_url{

color:maroon;

border-bottom: 2px solid maroon;

}

.error{

font-size: 25px;

color:maroon;

}

.img{

border-radius: 10px;

max-height: 200px;

max-width: 200px;

margin-bottom: 10px;

border: 2px solid #252930;

}

.profile-text{

display: grid;

}

.profile-text li{

display: inline-flex;

}

.profile-text p{

font-weight: bold;

margin-right: 10px;

}

/\*Стили для надписи, которую видит неавторизованный сотрудник\*/

.noauth{

margin-top: 150px;

height: 80%;

width: 90%;

margin-left: 100px;

font-weight: bold;

font-size: 40px;

}

/\*Стили для кнопок\*/

.but{

margin-top: 20px;

background-color: transparent;

width: 350px;

text-align: center;

font-size: 30px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.but:hover {

background-color: #252930;

color: white;

}

/\*Красивая анимация текста "Вы не авторизованы"\*/

.noauth span{

font: 700 4em/1 "Oswald", sans-serif;

letter-spacing: 0;

padding: .25em 0 .325em;

display: block;

margin: 0 auto;

text-shadow: 0 0 80px rgba(255,255,255,.5);

/\* Clip Background Image \*/

background: url("/media/photos/23/05/11/2.jpg") repeat-y;

-webkit-background-clip: text;

/\* Animate Background Image \*/

-webkit-text-fill-color: transparent;

-webkit-animation: aitf 40s linear infinite;

/\* Activate hardware acceleration for smoother animations \*/

-webkit-transform: translate3d(0,0,0);

-webkit-backface-visibility: hidden;

}

@-webkit-keyframes aitf {

0% { background-position: 0% 50%; }

100% { background-position: 100% 50%; }

}

/\*Стили для кнопки "Закрыть"\*/

.but\_close{

text-align: center;

right: 0;

font-weight: bold;

font-size: 20px;

background-color: transparent;

width: 100px;

height: 40px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.but\_close:hover {

background-color: #252930;

color: white;

}

/\*Стили кнопок профиля\*/

.profile\_btn{

align-items: center;

align-content: center;

display: block;

margin: 0 auto;

position: center;

}

.btn\_prof{

margin-top: 20px;

background-color: transparent;

max-width: 300px;

min-width: 100px;

text-align: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.btn\_prof\_exit{

position: absolute;

right: 0;

margin-top: 20px;

margin-right: 20px;

background-color: transparent;

width: 300px;

text-align: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.btn\_prof:hover {

background-color: #252930;

color: white;

}

.btn\_prof\_exit:hover {

background-color: #252930;

color: white;

}

.btn\_down{

position: absolute;

right: 0;

margin-right: 20px;

background-color: transparent;

width: 200px;

height: 40px;

text-align: center;

font-size: 15px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.btn\_down:hover {

background-color: #252930;

color: white;

}

.order\_{

height: 60px;

}

/\*Стили страницы добавления сотрудников\*/

.staff-add{

padding-top: 20px;

padding-left: 30px;

font-size: 25px;

box-shadow: 0 -6px 4px -4px rgba(0, 0, 0, .2);

background-color: white;

width: 100%;

border-radius: 3px;

overflow: hidden;

}

.staff-add\_header{

font-size: 30px;

font-weight: bold;

margin-bottom: 20px;

}

.but\_staff{

margin-top: 20px;

background-color: transparent;

width: 250px;

text-align: center;

font-size: 20px;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

padding: 16px 32px;

cursor: pointer;

color: #252930;

border: 2px solid #252930;

border-radius: 6px;

}

.but\_staff:hover {

background-color: #252930;

color: white;

}

Код файла reset.сss, используется на всех страницах системы, для сброса базовых стилей:

/\* Reset base styles \*/

\* {

padding: 0px;

margin: 0px;

border: none;

}

\*,

\*::before,

\*::after {

box-sizing: border-box;

}

/\* Links \*/

a, a:visited, a:hover {

color: inherit;

text-decoration: none;

display: inline-block;

}

a:focus,

a:active {

outline: none;

}

/\* Common \*/

aside, nav, footer, header, section, main {

display: block;

}

h1, h2, h3, h4, h5, h6, p {

font-size: inherit;

font-weight: inherit;

}

ul {

list-style: none;

}

img, svg {

display: block;

max-width: 100%;

height: auto;

}

Form

input, textarea, button, select {

font-family: inherit;

font-size: inherit;

}

input::-ms-clear {

display: none;

}

button, input[type="submit"] {

display: inline-block;

box-shadow: none;

background-color: transparent;

background: none;

cursor: pointer;

}

input:focus, input:active,

button:focus, button:active {

outline: none;

}

button::-moz-focus-inner {

padding: 0;

border: 0;

}

label {

cursor: pointer;

}

legend {

display: block;

}

Код файла application-add.html, страница добавления заявки:

{% extends 'itservice/home.html' %}

{% block content1 %}

<p>Заявка</p>

{% endblock %}

{% block content %}

<div class="form\_app">

<a class="butt-close"href="{% url 'home' %}">Назад</a>

<form method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label for="id\_application\_fio">Фио:</label>

<input class="input\_a" type="text" name="application\_fio" maxlength="50" required="" id="id\_application\_fio">

</p>

<p>

<label for="id\_application\_phone">Телефон:</label>

<input class="input\_a" type="text" name="application\_phone" maxlength="50" required="" id="id\_application\_phone">

</p>

<p>

<label for="id\_application\_email">Почта:</label>

<input class="input\_a" type="email" name="application\_email" maxlength="50" required="" id="id\_application\_email">

</p>

<p>

<label for="id\_application\_address">Ваш адрес:</label>

<input class="input\_a" type="text" name="application\_address" maxlength="50" required="" id="id\_application\_address">

</p>

<p>

<label for="id\_application\_comment">Заявка:</label>

<input class="input\_a" type="text" name="application\_comment" maxlength="300" required="" id="id\_application\_comment">

</p>

<p>

<label>Откуда вы узнали о нас?</label>

{{form.application\_found}}

</p>

<button class="butt" type="submit">Отправить</button>

</form>

</div>

{% endblock %}

Код файла application.html, страница отображения заявок сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Заявки

</div>

<div class="help">

<span class="tooltip" data-title="Это окно заявок пользователей.

Администратор или менеджер обрабатывают заявки и создают из них заказы.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<script>

const Change\_appl\_status = (application\_id)=> {

var xmlHttp = new XMLHttpRequest();

xmlHttp.open("POST", "{% url 'applications-change' %}" , false);

xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

xmlHttp.setRequestHeader("X-CSRFToken", "{{ csrf\_token }}");

console.log(xmlHttp);

xmlHttp.send(`{"application\_id": ${application\_id}}`);

window.location.reload()

}

</script>

<div class="main-window">

<div class="order\_"></div>

<div class="warpper">

<input class="radio" id="one" name="group" type="radio" checked>

<input class="radio" id="two" name="group" type="radio">

<div class="tabs">

<label class="tab" id="one-tab" for="one">Необработанные</label>

<label class="tab" id="two-tab" for="two">Обработанные</label>

</div>

<div class="panels">

<div class="panel" id="one-panel">

<div class="panel-title">Необработанные заявки</div>

{% for p in appl%}

{% if p.application\_active == 1 %}

<div class="tasks\_appl">

<div class="applo">

<label class="toggler-wrapper ">

{% if p.application\_active %}

<input type="checkbox" onclick="Change\_appl\_status({{p.id}})"/>

{% else %}

<input type="checkbox" onclick="Change\_appl\_status({{p.id}})" checked/>

{%endif%}

<div class="toggler-slider">

<div class="toggler-knob"></div>

</div>

</label>

<li class="task-inf">

<h1>Почта: {{p.application\_email}}</h1>

<p class="desc2">Фио: {{p.application\_fio}}</p>

<p class="desc2">Телефон: {{p.application\_phone}}</p>

<p class="desc2">Адрес: {{p.application\_address}}</p>

<p class="desc2">Откуда узнал: {{p.application\_found}}</p>

</li>

</div>

<div class="desc1">Заявка: {{p.application\_comment}}</div>

</div>

{% else %}

{% endif %}

{% endfor %}

</div>

<div class="panel" id="two-panel">

<div class="panel-title">Обработанные заявки</div>

{% for p in appl%}

{% if p.application\_active == 0 %}

<div class="tasks\_appl">

<div class="applo">

<label class="toggler-wrapper ">

{% if p.application\_active %}

<input type="checkbox" onclick="Change\_appl\_status({{p.id}})"/>

{% else %}

<input type="checkbox" onclick="Change\_appl\_status({{p.id}})" checked/>

{%endif%}

<div class="toggler-slider">

<div class="toggler-knob"></div>

</div>

</label>

<li class="task-inf">

<h1>Почта: {{p.application\_email}}</h1>

<p class="desc2">Фио: {{p.application\_fio}}</p>

<p class="desc2">Телефон: {{p.application\_phone}}</p>

<p class="desc2">Адрес: {{p.application\_address}}</p>

<p class="desc2">Откуда узнал: {{p.application\_found}}</p>

</li>

</div>

<div class="desc1">Заявка: {{p.application\_comment}}</div>

</div>

{% else %}

{% endif %}

{% endfor %}

</div>

</div>

</div>

</div>

{% endblock %}

Код файла clients.html, страница отображения клиентов сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Клиенты

</div>

<div class="help">

<span class="tooltip" data-title="Это окно ваших клиентов.

Создавайте клиентов и поставщиков для отслеживания истории работы с ними.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="main-window">

<button class="btn" id="open">Добавить клиента</button>

<div class="modal" id="modal\_task">

<div class="modal\_box">

<div class="modal\_header">

<h1>Добавление клиента</h1>

<button class="but\_close" id="close">Закрыть</button>

</div>

<form action="{% url 'clients' %}" method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label>Тип клиента:</label>

{{form.client\_type}}

</p>

<p>

<label for="id\_client\_name">Имя клиента:</label>

<input class="input" type="text" name="client\_name" maxlength="50" required="" id="id\_client\_name">

</p>

<p>

<label for="id\_client\_phone">Номер телефона:</label>

<input class="input" type="text" name="client\_phone" maxlength="40" required="" id="id\_client\_phone">

</p>

<p>

<label for="id\_client\_email">Email:</label>

<input class="input" type="text" name="client\_email" maxlength="100" required="" id="id\_client\_email">

</p>

<p>

<label for="id\_client\_address">Адрес:</label>

<input class="input" type="text" name="client\_address" maxlength="50" required="" id="id\_client\_address">

</p>

<p>

<label>Откуда клиент узнал о нас?</label>

{{form.client\_found}}

</p>

<button class="but" type="submit">Добавить</button>

</form>

</div>

</div>

<div class="ok"></div>

<div class="warpper">

<div class="panels">

<div class="panel-title">Клиенты</div>

<table class="table">

<thead>

<tr>

<th>Тип клиента</th>

<th>Имя клиента</th>

<th>Номер телефона</th>

<th>Email</th>

<th>Адрес</th>

<th>Откуда клиент узнал о нас?</th>

</tr>

</thead>

<tbody>

{% for c in clients%}

<tr>

<td><a href="{% url 'clients' %}">{{c.client\_type}}</a></td>

<td>{{c.client\_name}}</td>

<td>{{c.client\_phone}}</td>

<td>{{c.client\_email}}</td>

<td>{{c.client\_address}}</td>

<td>{{c.client\_found}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

{% endblock %}

Код файла home.html, главная страница – окно визитка:

{% load static %}

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">

<!--css файлы-->

<link rel="stylesheet" href="{% static 'itservice/css/reset.css' %}" type="text/css">

<link rel="stylesheet" href="{% static 'itservice/css/home.css' %}" type="text/css">

<!--Иконка Favicon-->

<link rel="shortcut icon" type="image/png" href="{% static 'itservice/static/itservice/favicon/1.png' %}"/>

<title>IT Service</title>

</head>

<body>

<div class="main-title">

{% block content1 %}

<p>IT Service</p>

{% endblock %}

</div>

<div class="main">

{% block content %}

<a class="btn" href="{% url 'applications\_add' %}">Оставить заявку</a>

<a class="btn" href="{% url 'review' %}">Отзывы</a>

<a class="btn" href="{% url 'login' %}">Для персонала</a>

{% endblock %}

</div>

</body>

</html>

Код файла inventory.html, страница отображения склада сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Склад

</div>

<div class="help">

<span class="tooltip" data-title="Это склад товаров.Размещайте продукты на складе,

чтобы организовать свои запасы, отслеживать уровень запасов и

автоматизировать процессы управления.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="main-window">

<button class="btn" id="open">Добавить устройство</button>

<div class="modal" id="modal\_task">

<div class="modal\_box">

<div class="modal\_header">

<h1>Добавление устройства</h1>

<button class="but\_close" id="close">Закрыть</button>

</div>

<form action="{% url 'inventory' %}" method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label>Категория:</label>

{{form.inventory\_type}}

</p>

<p>

<label for="id\_inventory\_article">Артикул:</label>

<input class="input" type="number" name="inventory\_article" required="" id="id\_inventory\_article">

</p>

<p>

<label for="id\_inventory\_name">Название:</label>

<input class="input" type="text" name="inventory\_name" maxlength="50" required="" id="id\_inventory\_name">

</p>

<p>

<label for="id\_inventory\_color">Цвет:</label>

<input class="input" type="text" name="inventory\_color" maxlength="40" required="" id="id\_inventory\_color">

</p>

<p>

<label for="id\_inventory\_condition">Состояние:</label>

<input class="input" type="text" name="inventory\_condition" maxlength="100" required="" id="id\_inventory\_condition">

</p>

<p>

<label for="id\_inventory\_description">Описание:</label>

<input class="input" type="text" name="inventory\_description" maxlength="1000" required="" id="id\_inventory\_description">

</p>

<p>

<label>Клиент:</label>

{{form.inventory\_client}}

</p>

<button class="but" type="submit">Добавить</button>

</form>

</div>

</div>

<div class="ok"></div>

<div class="warpper">

<div class="panels">

<div class="panel-title">Склад</div>

<table class="table">

<thead>

<tr>

<th>Категория</th>

<th>Артикул</th>

<th>Название</th>

<th>Цвет</th>

<th>Состояние</th>

<th>Описание</th>

<th>Клиент</th>

</tr>

</thead>

<tbody>

{% for i in invent%}

<tr>

<td>{{i.inventory\_type}}</td>

<td>{{i.inventory\_article}}</td>

<td>{{i.inventory\_name}}</td>

<td>{{i.inventory\_color}}</td>

<td>{{i.inventory\_condition}}</td>

<td>{{i.inventory\_description}}</td>

<td>{{i.inventory\_client}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

{% endblock %}

Код файла login.html, страница входа в систему:

{% load static %}

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">

<link rel="stylesheet" href="{% static 'itservice/css/reset.css' %}" type="text/css">

<link rel="stylesheet" href="{% static 'itservice/css/login.css' %}" type="text/css">

<link rel="shortcut icon" type="image/png" href="{% static 'itservice/static/itservice/favicon/1.png' %}"/>

<title>IT Service</title>

</head>

<body>

<div class="page">

<div class="container">

<div class="left">

<div class="login">Login</div>

<div class="eula">Входя в систему, вы соглашаетесь на смехотворно длинные условия, которые вы не удосужились прочитать</div>

</div>

<div class="right">

<button class="btn-close"><a href="{% url 'home' %}">Назад</a></button>

<div class="form">

<form method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<label for="id\_username">Имя пользователя:</label>

<input class="input" type="text" name="username" autofocus="" autocapitalize="none" autocomplete="username" maxlength="150" required="" id="id\_username">

<label for="id\_password">Пароль:</label>

<input class="input" type="password" name="password" autocomplete="current-password" required="" id="id\_password">

<button class="btn" id="submit" type="submit">Войти в систему</button>

</form>

</div>

</div>

</div>

</div>

</body>

</html>

Код файла main.html, главный шаблон страницы, из которого созданы все остальные:

{% load static %}

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">

<!--css файлы-->

<link rel="stylesheet" href="{% static 'itservice/css/reset.css' %}" type="text/css">

<link rel="stylesheet" href="{% static 'itservice/css/main.css' %}" type="text/css">

<!--Иконка Favicon-->

<link rel="shortcut icon" type="image/png" href="{% static 'itservice/static/itservice/favicon/1.png' %}"/>

<title>IT Service</title>

</head>

<body>

<!--Левый сайдбар-->

<div class="sidebar">

<div class="sidebar-name">

IT Service

</div>

<!--Если сотрудник авторизован, то-->

{% if user.is\_authenticated %}

<ul>

<li><a href="{% url 'profile' %}">Профиль</a></li>

<li><a href="{% url 'tasks' %}">Задачи</a></li>

<li><a href="{% url 'orders' %}">Заказы</a></li>

<li><a href="{% url 'sales' %}">Продажи</a></li>

<li><a href="{% url 'inventory' %}">Склад</a></li>

<li><a href="{% url 'clients' %}">Клиенты</a></li>

{% if user.user\_role == 'Администратор'%}

<li><a href="{% url 'staff' %}">Сотрудники</a></li>

<li><a href="{% url 'applications' %}">Заявки</a></li>

<li><a href="{% url 'admin:index' %}">Админ</a></li>

{% elif user.user\_role == 'Менеджер' %}

<li><a href="{% url 'staff' %}">Сотрудники</a></li>

<li><a href="{% url 'applications' %}">Заявки</a></li>

{%endif%}

</ul>

{% else %}

{%endif%}

</div>

<main class="main">

<!--Если сотрудник авторизован, то-->

{% if user.is\_authenticated %}

<div class="header">

{% block content %}

{% endblock %}

</div>

<div class="main-form">

{% block content1 %}

{% endblock %}

</div>

{% else %}

<div class="noauth">

<span>Вы не авторизованы</span>

<a class="but" href="{% url 'login' %}">Авторизоваться</a>

</div>

{%endif%}

</main>

<script language="JavaScript">

<!--Скрипт для открытия и закрытия модальных окон-->

document.getElementById("open").addEventListener("click",function () {

document.getElementById("modal\_task").classList.add("open")

})

document.getElementById("close").addEventListener("click",function () {

document.getElementById("modal\_task").classList.remove("open")

})

window.addEventListener('keydown', (e) => {

if (e.key === "Escape"){

document.getElementById("modal\_task").classList.remove("open")

}

})

</script>

</body>

</html>

Код файла orders.html, страница отображения заказов сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Заказы

</div>

<div class="help">

<span class="tooltip" data-title="Это страница рабочих заказов.

Вы можете менять их статус или добавлять новые.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<script>

const Change\_order\_status = (order\_id)=> {

var xmlHttp = new XMLHttpRequest();

xmlHttp.open("POST", "{% url 'orders-change' %}" , false);

xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

xmlHttp.setRequestHeader("X-CSRFToken", "{{ csrf\_token }}");

console.log(xmlHttp);

xmlHttp.send(`{"order\_id": ${order\_id}}`);

window.location.reload()

}

</script>

<div class="main-window">

{% if user.user\_role == 'Администратор' or user.user\_role == 'Менеджер' %}

<button class="btn" id="open">Добавить заказ</button>

{% else %}

<div class="order\_"></div>

{%endif%}

<div class="modal" id="modal\_task">

<div class="modal\_box">

<div class="modal\_header">

<h1>Добавление заказа</h1>

<button class="but\_close" id="close">Закрыть</button>

</div>

<form action="{% url 'orders' %}" method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label for="id\_order\_name">Название заказа:</label>

<input class="input" type="text" name="order\_name" maxlength="50" required="" id="id\_order\_name">

</p>

<p>

<label for="id\_order\_description">Требования к заказу:</label>

<input class="input" type="text" name="order\_description" maxlength="1000" required="" id="id\_order\_description">

</p>

<p>

<label for="id\_order\_time">Крайний срок:</label>

<input class="input" type="datetime-local" name="order\_time" value="" required="" id="id\_order\_time"><input type="hidden" name="initial-order\_time" value="2023-05-12 17:33:18" id="initial-id\_order\_time">

</p>

<p>

<label for="id\_order\_price">Ориентировочная стоимость:</label>

<input class="input" type="number" name="order\_price" required="" id="id\_order\_price">

</p>

<p>

<label >Клиент:</label>

{{form.order\_client}}

</p>

<p>

<label for="id\_order\_active">Активна:</label>

<input class="input" type="checkbox" name="order\_active" id="id\_order\_active" checked="">

</p>

<button class ="but" type="submit">Добавить</button>

</form>

</div>

</div>

<div class="warpper">

<input class="radio" id="one" name="group" type="radio" checked>

<input class="radio" id="two" name="group" type="radio">

<div class="tabs">

<label class="tab" id="one-tab" for="one">Активные</label>

<label class="tab" id="two-tab" for="two">Выполненные</label>

</div>

<div class="panels">

<div class="panel" id="one-panel">

<div class="panel-title">Активные заказы</div>

<div class="table-wrap">

<table class="table">

<thead>

<tr>

<th>Статус</th>

<th>Название заказа</th>

<th>Требования к заказу</th>

<th>Крайний срок</th>

<th>Ориентировочная стоимость</th>

<th>Клиент</th>

</tr>

</thead>

<tbody>

{% for o in order%}

{% if o.order\_active == 1 %}

<tr>

<td>{% if o.order\_active %}

<input type="checkbox" onclick="Change\_order\_status({{o.id}})"/>

{% else%}

<input type="checkbox" onclick="Change\_order\_status({{o.id}})" checked/>

{%endif%}</td>

<td>{{o.order\_name}}</td>

<td>{{o.order\_description}}</td>

<td>{{o.order\_time}}</td>

<td>{{o.order\_price}}</td>

<td>{{o.order\_client}}</td>

</tr>

{% else %}

{% endif %}

{% endfor %}

</tbody>

</table>

</div>

</div>

<div class="panel" id="two-panel">

<div class="panel-title">Выполненные заказы</div>

<table class="table">

<thead>

<tr>

<th>Статус</th>

<th>Название заказа</th>

<th>Требования к заказу</th>

<th>Крайний срок</th>

<th>Ориентировочная стоимость</th>

<th>Клиент</th>

</tr>

</thead>

<tbody>

{% for o in order%}

{% if o.order\_active == 0 %}

<tr>

<td>{% if o.order\_active %}

<input type="checkbox" onclick="Change\_order\_status({{o.id}})"/>

{% else%}

<input type="checkbox" onclick="Change\_order\_status({{o.id}})" checked/>

{%endif%}</td>

<td>{{o.order\_name}}</td>

<td>{{o.order\_description}}</td>

<td>{{o.order\_time}}</td>

<td>{{o.order\_price}}</td>

<td>{{o.order\_client}}</td>

</tr>

{% else %}

{% endif %}

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

</div>

{% endblock %}

Код файла profile.html, страница личного профиля:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Личные данные

</div>

<div class="help">

<span class="tooltip" data-title="Это ваш профиль.

Здесь вы можете отредактировать личную информацию.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="profile">

<div class="profile-body">

<div class="row">

<div class="col-md-3">

<div class="profile-title">

{{ user }}

</div>

<figure>

{% if user.user\_photo %}

<a href="/media/{{user.user\_photo}}"><img class="img" src="{{user.user\_photo.url}}"></a>

{% endif %}

</figure>

</div>

<div>

<ul class="profile-text">

<li><p>Никнейм:</p> {{ user.username }}</li>

<li><p>Имя и фамилия:</p> {{ user.get\_full\_name }}</li>

<li><p>Заходил:</p> {{ user.last\_login }}</li>

<li><p>Номер телефона:</p> {{ user.user\_phone }}</li>

<li><p>О себе:</p> {{ user.user\_note }}</li>

<li><p>Должность:</p> {{ user.user\_role }}</li>

</ul>

<div class="profile\_btn">

<button class="btn\_prof"><a href="{% url 'profile-edit' %}">Редактировать профиль</a></button>

<button class="btn\_prof\_exit"><a href="{% url 'logout' %}">Выйти</a></button>

</div>

</div>

</div>

</div>

</div>

{% endblock %}

Код файла profile\_edit.html, страница редактирования личной информации:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Редактирование личных данных

</div>

<div class="help">

<span class="tooltip" data-title="Это ваш профиль.

Здесь вы можете отредактировать личную информацию.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="profile">

<div class="profile-title">Изменение профиля</div>

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

<p>

<label for="id\_username">Имя пользователя:</label>

<input class="input" type="text" name="username" value="{{user.username}}"

maxlength="150" required="" id="id\_username">

</p>

<p>

<label for="id\_first\_name">Имя:</label>

<input class="input" type="text" name="first\_name" value="{{user.first\_name}}" maxlength="150" id="id\_first\_name">

</p>

<p>

<label for="id\_last\_name">Фамилия:</label>

<input class="input" type="text" name="last\_name" value="{{user.last\_name}}" maxlength="150" id="id\_last\_name">

</p>

<p>

<label for="id\_email">Адрес электронной почты:</label>

<input class="input" type="email" name="email" value="{{user.email}}" maxlength="254" id="id\_email">

</p>

<p>

<label for="id\_user\_phone">Номер телефона:</label>

<input class="input" type="text" name="user\_phone" value="{{user.user\_phone}}" maxlength="40" required="" id="id\_user\_phone">

</p>

<p>

<label for="id\_user\_photo">Фото сотрудника на данный момент:</label>

<a class="photo\_url" href="/media/{{user.user\_photo}}">{{user.user\_photo}}</a><br>

Изменить:

<input type="file" name="user\_photo" accept="image/\*" id="id\_user\_photo">

</p>

<button type="submit" class="btn">Подтвердить изменение профиля</button>

</form>

</div>

{% endblock %}

Код файла review.html, страница отображения отзывов:

{% extends 'itservice/home.html' %}

{% block content1 %}

<p>Отзывы</p>

{% endblock %}

{% block content %}

<div class="window">

<div class="buttons">

<button class="butt-rev"> <a class href="{% url 'review\_add' %}">Оставить отзыв</a></button>

<button class="butt-rev"><a href="{% url 'home' %}">Назад</a></button>

</div>

<div class="review">

{% for r in reviews%}

<div class="grade">

<div class="grade\_header">

{% if r.review\_gender == 'Мужчина'%}

<img class="img" src="/media/photos/review/man.jpg" alt="">

{% else %}

<img class="img" src="/media/photos/review/woman.jpg" alt="">

{% endif %}

</div>

<p>{{r.review\_grade}}</p>

<h1>Почта: {{r.review\_email}}</h1>

<p class="">Фио: {{r.review\_fio}}</p>

<p class="">Пол: {{r.review\_gender}}</p>

<p class="">Отзыв: {{r.review\_comment}}</p>

</div>

{% endfor %}

</div>

</div>

{% endblock %}

Код файла review-add.html, страница добавления отзывов:

{% extends 'itservice/home.html' %}

{% block content1 %}

<p>Добавление отзыва</p>

{% endblock %}

{% block content %}

<div class="form\_app">

<a class="butt-close"href="{% url 'home' %}">Назад</a>

<form method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label for="id\_review\_fio">Фио:</label>

<input class="input\_a" type="text" name="review\_fio" maxlength="50" required="" id="id\_review\_fio">

</p>

<p>

<label for="id\_review\_email">Почта:</label>

<input class="input\_a" type="email" name="review\_email" maxlength="50" required="" id="id\_review\_email">

</p>

<p>

<label for="id\_review\_comment">Отзыв:</label>

<input class="input\_a" type="text" name="review\_comment" maxlength="150" required="" id="id\_review\_comment">

</p>

<p>

<label class="op">Пол:</label>

{{form.review\_gender}}

</p>

<p>

<label class="op">Оценка:</label>

{{form.review\_grade}}

</p>

<button class="butt" type="submit">Отправить</button>

</form>

</div>

{% endblock %}

Код файла sales.html, страница отображения продаж сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Продажи

</div>

<div class="help">

<span class="tooltip" data-title="Это окно ваших продаж.

Продавайте товары, отслеживайте их оборот и анализируйте доход.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<script>

const Download\_table = ()=> {

// var xmlHttp = new XMLHttpRequest();

// xmlHttp.open("GET", "{% url 'sales-down' %}" , false);

// xmlHttp.send();

window.location="{% url 'sales-down' %}"

}

</script>

<div class="main-window">

{% if user.user\_role == 'Администратор' or user.user\_role == 'Менеджер' %}

<button class="btn" id="open">Добавить продажу</button>

{% else %}

<div class="order\_"></div>

{%endif%}

<div class="modal" id="modal\_task">

<div class="modal\_box">

<div class="modal\_header">

<h1>Добавление продажи</h1>

<button class="but\_close" id="close">Закрыть</button>

</div>

<form action="{% url 'sales' %}" method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label for="id\_sales\_name">Наименование работы:</label>

<input class="input" type="text" name="sales\_name" maxlength="100" required="" id="id\_sales\_name">

</p>

<p>

<label for="id\_sales\_comment">Комментарий:</label>

<input class="input" type="text" name="sales\_comment" maxlength="100" required="" id="id\_sales\_comment">

</p>

<p>

<label for="id\_sales\_price">Цена:</label>

<input class="input" type="text" name="sales\_price" maxlength="100" required="" id="id\_sales\_price">

</p>

<p>

<label for="id\_sales\_time">Время продажи:</label>

<input class="input" type="datetime-local" name="sales\_time" value="" required="" id="id\_sales\_time"><input type="hidden" name="initial-sales\_time" value="2023-05-12 18:06:18" id="initial-id\_sales\_time">

</p>

<p>

<label>Клиент:</label>

{{form.sales\_client}}

</p>

<!-- {{form.as\_p}}-->

<button class="but" type="submit">Добавить</button>

</form>

</div>

</div>

<div class="ok"></div>

<div class="warpper">

<div class="panels">

<!--Кнопка для загрузки Excel файла с продажами-->

<button class="btn\_down" onclick="Download\_table()" >Скачать таблицу</button>

<div class="panel-title">Продажи</div>

<table class="table">

<thead>

<tr>

<th>Наименование работы</th>

<th>Комментарий</th>

<th>Цена</th>

<th>Время продажи</th>

<th>Клиент</th>

</tr>

</thead>

<tbody>

{% for s in sale%}

<a href="{% url 'sales' %}">

<tr>

<td>{{s.sales\_name}}</td>

<td>{{s.sales\_comment}}</td>

<td>{{s.sales\_price}}</td>

<td>{{s.sales\_time}}</td>

<td>{{s.sales\_client}}</td>

</tr>

</a>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

{% endblock %}

Код файла staff.html, страница отображения сотрудников в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Сотрудники

</div>

<div class="help">

<span class="tooltip" data-title="Это окно сотрудников.

Администратор или менеджер могут просматривать сотрудников.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="main-window">

<button class="btn" id=""><a href="{% url 'staff-add' %}">Добавить сотрудника</a></button>

<div class="ok"></div>

<div class="warpper">

<div class="panels">

<div class="panel-title">Сотрудники</div>

<table class="table">

<thead>

<tr>

<th>Id</th>

<th>Должность</th>

<th>Имя</th>

<th>Фамилия</th>

<th>Email</th>

<th>Телефон</th>

<th>Примечание</th>

<th>Логин</th>

<th>Админ</th>

</tr>

</thead>

<tbody>

{% for u in us%}

<tr>

<td>{{u.id}}</td>

<td>{{u.user\_role}}</td>

<td>{{u.first\_name}}</td>

<td>{{u.last\_name}}</td>

<td>{{u.email}}</td>

<td>{{u.user\_phone}}</td>

<td>{{u.user\_note}}</td>

<td>{{u.username}}</td>

<td>{{u.is\_superuser}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

{% endblock %}

Код файла staff-add.html, страница добавления сотрудника в систему:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Добавление сотрудника

</div>

<div class="help">

<span class="tooltip" data-title="Это окно добавления сотрудников.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<div class="ok"></div>

<div class="staff-add">

<div class="staff-add\_header">

Добавление сотрудника

</div>

<form method="post">

{% csrf\_token %}

<div class="error">{{form.errors}}</div>

<p>

<label for="id\_username">Логин:</label>

<input class="input" type="text" name="username" maxlength="150" required="" id="id\_username">

</p>

<p>

<label for="id\_password1">Пароль:</label>

<input class="input" type="text" name="password1" maxlength="128" required="" id="id\_password1">

</p>

<p>

<label for="id\_password2">Повтор пароля:</label>

<input class="input" type="text" name="password2" maxlength="128" required="" id="id\_password2">

</p>

<p>

<label for="id\_first\_name">Имя:</label>

<input class="input" type="text" name="first\_name" maxlength="150" id="id\_first\_name">

</p>

<p>

<label for="id\_last\_name">Фамилия:</label>

<input class="input" type="text" name="last\_name" maxlength="150" id="id\_last\_name">

</p>

<p>

<label for="id\_email">Адрес электронной почты:</label>

<input class="input" type="email" name="email" maxlength="254" id="id\_email">

</p>

<p>

<label for="id\_user\_phone">Номер телефона:</label>

<input class="input" type="text" name="user\_phone" maxlength="40" required="" id="id\_user\_phone">

</p>

<p>

<label for="id\_user\_note">Примечание:</label>

<input class="input" type="text" name="user\_note" maxlength="50" required="" id="id\_user\_note">

</p>

<p>

<label>Должность:</label>

{{form.user\_role}}

</p>

<!-- {{form.as\_p}}-->

<div class="profile\_btn">

<button class="but\_staff" type="submit">Зарегистрировать</button>

<button class="btn\_prof\_exit"><a href="{% url 'staff' %}">Назад</a></button>

</div>

</form>

</div>

</div>

{% endblock %}

Код файла tasks.html, страница отображения задач сотруднику в системе:

{% extends 'itservice/main.html' %}

{% block content %}

<div class="header-name">

Задачи

</div>

<div class="help">

<span class="tooltip" data-title="Это окно активных задач коллектива.

Назначайте задачи членам вашей команды или себе, чтобы все сделать вовремя.">Помощь</span>

</div>

{% endblock %}

{% block content1 %}

<script>

<!--Скрипт для внесения изменений в бд, после переключения чекбокса-->

const Change\_task\_status = (task\_id)=> {

var xmlHttp = new XMLHttpRequest();

xmlHttp.open("POST", "{% url 'tasks-change' %}" , false);

xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

xmlHttp.setRequestHeader("X-CSRFToken", "{{ csrf\_token }}");

console.log(xmlHttp);

xmlHttp.send(`{"task\_id": ${task\_id}}`);

window.location.reload()

}

</script>

<div class="main-window">

{% if user.user\_role == 'Администратор' or user.user\_role == 'Менеджер' %}

<button class="btn" id="open">Добавить задачу</button>

{% else %}

<div class="order\_"></div>

{%endif%}

<div class="modal" id="modal\_task">

<div class="modal\_box">

<div class="modal\_header">

<h1>Добавление задачи</h1>

<button class="but\_close" id="close">Закрыть</button>

</div>

<form action="{% url 'tasks' %}" method="post">

{% csrf\_token %}

{{form.non\_field\_errors}}

<p>

<label for="id\_task\_name">Название задачи:</label>

<input class ="input" type="text" name="task\_name" maxlength="50" required="" id="id\_task\_name">

</p>

<p>

<label for="id\_task\_description">Описание задачи:</label>

<input class ="input" type="text" name="task\_description" maxlength="1000" required="" id="id\_task\_description">

</p>

<p>

<label for="id\_task\_time">Крайний срок:</label>

<input class ="input" type="datetime-local" name="task\_time" value="" required="" id="id\_task\_time"><input type="hidden" name="initial-task\_time" value="2023-05-12 17:26:32" id="initial-id\_task\_time">

</p>

<p>

<label for="id\_task\_active">Активна:</label>

<input class ="input" type="checkbox" name="task\_active" id="id\_task\_active" checked="">

</p>

<!-- {{form.as\_p}}-->

<button class="but" type="submit">Добавить</button>

</form>

</div>

</div>

<div class="warpper">

<input class="radio" id="one" name="group" type="radio" checked>

<input class="radio" id="two" name="group" type="radio">

<div class="tabs">

<label class="tab" id="one-tab" for="one">В разработке</label>

<label class="tab" id="two-tab" for="two">Выполненные</label>

</div>

<div class="panels">

<div class="panel" id="one-panel">

<div class="panel-title">Задачи в разработке</div>

{% for p in task%}

{% if p.task\_active == 1 %}

<div class="tasks">

<div>

<label class="toggler-wrapper ">

{% if p.task\_active %}

<input type="checkbox" onclick="Change\_task\_status({{p.id}})"/>

{% else %}

<input type="checkbox" onclick="Change\_task\_status({{p.id}})" checked/>

{%endif%}

<div class="toggler-slider">

<div class="toggler-knob"></div>

</div>

</label>

</div>

<li class="task-inf">

<h1>{{p.task\_name}}</h1>

<p class="desc">{{p.task\_description|linebreaks|truncatewords:25}}</p>

<p class="time">Крайний срок: {{p.task\_time}}</p>

</li>

</div>

{% else %}

{% endif %}

{% endfor %}

</div>

<div class="panel" id="two-panel">

<div class="panel-title">Выполненные задачи</div>

{% for p in task%}

{% if p.task\_active == 0 %}

<div class="tasks">

<div>

<label class="toggler-wrapper ">

{% if p.task\_active %}

<input type="checkbox" onclick="Change\_task\_status({{p.id}})"/>

{% else%}

<input type="checkbox" onclick="Change\_task\_status({{p.id}})" checked/>

{%endif%}

<div class="toggler-slider">

<div class="toggler-knob"></div>

</div>

</label>

</div>

<li class="task-inf">

<h1>{{p.task\_name}}</h1>

<p class="desc">{{p.task\_description|linebreaks|truncatewords:25}}</p>

{% if p.task\_active %}

<p class="time">Крайний срок: {{p.task\_time}}</p>

{% else%}

<p class="time\_2">Крайний срок: {{p.task\_time}}</p>

{%endif%}

</li>

</div>

{% else %}

{% endif %}

{% endfor %}

</div>

</div>

</div>

</div>

{% endblock %}

Код файла forms.py, в котором реализованы все формы добавления и редактирования:

from django import forms

from django.contrib.auth.forms import UserCreationForm, AuthenticationForm

from .models import \*

class ProfileFrom(forms.ModelForm):

class Meta:

model = users

fields = ['username', 'first\_name', 'last\_name', 'email',

'user\_phone', 'user\_photo']

class RegStaff(UserCreationForm):

username = forms.CharField(label='Логин', widget=forms.TextInput(attrs={'class': 'input'}))

password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class': 'input'}))

password2 = forms.CharField(label='Повтор пароля', widget=forms.PasswordInput(attrs={'class': 'input'}))

first\_name = forms.CharField(label='Имя', widget=forms.TextInput(attrs={'class': 'input'}))

last\_name = forms.CharField(label='Фамилия', widget=forms.TextInput(attrs={'class': 'input'}))

email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={'class': 'input'}))

user\_phone = forms.CharField(label='Телефон', widget=forms.TextInput(attrs={'class': 'input'}))

class Meta:

model = users

fields = ['username', 'first\_name', 'last\_name', 'email', 'user\_phone',

'password1', 'password2', 'user\_note', 'user\_role', ]

class Login(AuthenticationForm):

username = forms.CharField(label='Логин', widget=forms.TextInput(attrs={'class': 'input'}))

password = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class': 'input'}))

class ProfileUpdateForm(forms.ModelForm):

class Meta:

model = users

fields = ['username', 'first\_name', 'last\_name', 'email',

'user\_phone', 'user\_photo']

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

for field in self.fields:

self.fields[field].widget.attrs.update({

'class': 'form-control',

'autocomplete': 'off'

})

class AddTaskFrom(forms.ModelForm):

class Meta:

model = tasks

fields = "\_\_all\_\_"

class AddOrdersFrom(forms.ModelForm):

class Meta:

model = orders

fields = "\_\_all\_\_"

class AddInventFrom(forms.ModelForm):

class Meta:

model = inventory

fields = "\_\_all\_\_"

class AddSalesFrom(forms.ModelForm):

class Meta:

model = sales

fields = "\_\_all\_\_"

class AddClientFrom(forms.ModelForm):

class Meta:

model = client

fields = "\_\_all\_\_"

class AddApplication(forms.ModelForm):

class Meta:

model = application

fields = ['application\_fio', 'application\_phone', 'application\_email',

'application\_address', 'application\_found', 'application\_comment']

class AddReview(forms.ModelForm):

class Meta:

model = review

fields = ['review\_fio', 'review\_gender', 'review\_email', 'review\_comment', 'review\_grade']

Код файла models.py, в котором реализованы все модели базы данных:

from datetime import date, datetime

from django.db import models

from django.contrib.auth.models import AbstractUser

class users(AbstractUser):

Type = (

('Администратор', 'Администратор'),

('Техник', 'Техник'),

('Программист', 'Программист'),

('Менеджер', 'Менеджер')

)

user\_phone = models.CharField(max\_length=40, verbose\_name="Номер телефона")

user\_note = models.CharField(max\_length=50, verbose\_name="Примечание")

user\_time = models.DateField(default=date.today, verbose\_name="Дата начала работы")

user\_photo = models.ImageField(upload\_to="photos/%y/%m/%d/", verbose\_name="Фото сотрудника")

user\_role = models.CharField(max\_length=50, choices=Type, default='Программист', verbose\_name="Должность")

class Meta:

verbose\_name = "Пользователи"

verbose\_name\_plural = "Пользователи"

class tasks(models.Model):

task\_name = models.CharField(max\_length=50, verbose\_name="Название задачи")

task\_description = models.CharField(max\_length=1000, verbose\_name="Описание задачи")

task\_time = models.DateTimeField(default=datetime.now, verbose\_name="Крайний срок")

task\_active = models.BooleanField(default=True, verbose\_name="Активна")

def \_\_str\_\_(self):

return self.task\_name

class Meta:

verbose\_name = "Задачи"

verbose\_name\_plural = "Задачи"

class client(models.Model):

Type = (

('Компания', 'Компания'),

('Физ.Лицо', 'Физ.Лицо')

)

Found = (

('Знакомые', 'Знакомые'),

('Интернет', 'Интернет'),

('Реклама', 'Реклама')

)

client\_type = models.CharField(max\_length=15, choices=Type,

default='Физ.лицо', verbose\_name="Тип клиента")

client\_name = models.CharField(max\_length=50, verbose\_name="Имя клиента")

client\_phone = models.CharField(max\_length=40, verbose\_name="Номер телефона")

client\_email = models.CharField(max\_length=100, verbose\_name="Email")

client\_address = models.CharField(max\_length=50, verbose\_name="Адрес")

client\_found = models.CharField(max\_length=15, choices=Found, default='Знакомые',

verbose\_name="Откуда клиент узнал о нас?")

def \_\_str\_\_(self):

return self.client\_name

class Meta:

verbose\_name = "Клиент"

verbose\_name\_plural = "Клиент"

class orders(models.Model):

order\_name = models.CharField(max\_length=50, verbose\_name="Название заказа")

order\_description = models.CharField(max\_length=1000, verbose\_name="Требования к заказу")

order\_time = models.DateTimeField(default=datetime.now, verbose\_name="Крайний срок")

order\_price = models.IntegerField(verbose\_name="Ориентировочная стоимость")

order\_client = models.ForeignKey(client, on\_delete=models.PROTECT, verbose\_name="Клиент")

order\_active = models.BooleanField(default=True, verbose\_name="Активна")

def \_\_str\_\_(self):

return self.order\_name

class Meta:

verbose\_name = "Заказы"

verbose\_name\_plural = "Заказы"

class inventory(models.Model):

Type = (

('Устройство', 'Устройство'),

('Запчасти', 'Запчасти')

)

inventory\_type = models.CharField(max\_length=15, choices=Type,

default='Устройство', verbose\_name="Категория")

inventory\_article = models.IntegerField(verbose\_name="Артикул")

inventory\_name = models.CharField(max\_length=50, verbose\_name="Название")

inventory\_color = models.CharField(max\_length=40, verbose\_name="Цвет")

inventory\_condition = models.CharField(max\_length=100, verbose\_name="Состояние")

inventory\_description = models.CharField(max\_length=1000, verbose\_name="Описание")

inventory\_client = models.ForeignKey(client, on\_delete=models.PROTECT, null=True, verbose\_name="Клиент")

def \_\_str\_\_(self):

return self.inventory\_name

class Meta:

verbose\_name = "Склад"

verbose\_name\_plural = "Склад"

class sales(models.Model):

sales\_name = models.CharField(max\_length=100, verbose\_name="Наименование работы")

sales\_comment = models.CharField(max\_length=100, verbose\_name="Комментарий")

sales\_price = models.CharField(max\_length=100, verbose\_name="Цена")

sales\_time = models.DateTimeField(default=datetime.today, verbose\_name="Время продажи")

sales\_client = models.ForeignKey(client, on\_delete=models.PROTECT)

def \_\_str\_\_(self):

return self.sales\_name

class Meta:

verbose\_name = "Продажи"

verbose\_name\_plural = "Продажи"

class review(models.Model):

Gender = (

('Мужчина', 'Мужчина'),

('Женщина', 'Женщина')

)

Grade = (

('Отлично!', 'Отлично!'),

('Очень хорошо', 'Очень хорошо'),

('Хорошо', 'Хорошо'),

('Плохо', 'Плохо'),

('Очень плохо', 'Очень плохо'),

)

review\_fio = models.CharField(max\_length=50, verbose\_name="Фио")

review\_gender = models.CharField(max\_length=15, choices=Gender, default='Мужчина',

verbose\_name="Пол")

review\_email = models.EmailField(max\_length=50, verbose\_name="Почта")

review\_comment = models.CharField(max\_length=150, verbose\_name="Отзыв")

review\_grade = models.CharField(max\_length=15, choices=Grade, default='Отлично!', verbose\_name="Оценка")

def \_\_str\_\_(self):

return self.review\_email

class Meta:

verbose\_name = "Отзывы"

verbose\_name\_plural = "Отзывы"

class application(models.Model):

Found = (

('Знакомые', 'Знакомые'),

('Интернет', 'Интернет'),

('Реклама', 'Реклама')

)

application\_fio = models.CharField(max\_length=50, verbose\_name="Фио")

application\_phone = models.CharField(max\_length=50, verbose\_name="Телефон")

application\_email = models.EmailField(max\_length=50, verbose\_name="Email")

application\_address = models.CharField(max\_length=50, verbose\_name="Ваш адрес")

application\_found = models.CharField(max\_length=15, choices=Found, default='Реклама',

verbose\_name="Откуда вы узнали о нас?")

application\_comment = models.CharField(max\_length=150, verbose\_name="Заявка")

application\_active = models.BooleanField(default=True, verbose\_name="Активна")

def \_\_str\_\_(self):

return self.application\_email

class Meta:

verbose\_name = "Заявки"

verbose\_name\_plural = "Заявки"

Код файла views.py, в котором реализованы все представления:

from django.contrib.auth import authenticate, logout

from django.contrib.auth.forms import AuthenticationForm

from django.contrib.auth.views import LoginView

from django.db import transaction

from django.shortcuts import render, redirect

from django.http import HttpResponse, HttpResponseNotFound

from django.urls import reverse\_lazy

from django.views import View

from django.views.generic import DetailView, UpdateView, CreateView

from django.core.mail import EmailMessage

import pandas as pd

import codecs

from django.http import FileResponse

from openpyxl.workbook import Workbook

from .forms import \*

from .models import \*

import json

from django.core.mail import send\_mail

# import smtplib

# smtpObj = smtplib.SMTP('smtp.mail.ru', 465)

# smtpObj.starttls()

# smtpObj.login('itservicediplom1@mail.ru', 'mmdSrKEW3YUUu2YpgysL')

def Main(request):

user = request.user

return render(request, 'itservice/main.html', {'user': user})

def Home(request):

return render(request, 'itservice/home.html')

def Review(request):

if request.method == 'POST':

form\_review = AddReview(request.POST)

if form\_review.is\_valid():

form\_review.save()

return redirect('home')

else:

form\_review = AddReview()

reviews = review.objects.all()

return render(request, 'itservice/review-add.html', {'reviews': reviews, 'form': form\_review})

def Rev(request):

reviews = review.objects.all()

return render(request, 'itservice/review.html', {'reviews': reviews})

def Application(request):

if request.method == 'POST':

form\_appl = AddApplication(request.POST)

if form\_appl.is\_valid():

form\_appl.save()

return redirect('home')

else:

form\_appl = AddApplication()

applications = application.objects.all()

return render(request, 'itservice/application-add.html', {'applications': applications, 'form': form\_appl})

def Profile(request):

user = request.user

if request.method == 'POST':

form\_prof = ProfileFrom(request.POST)

if form\_prof.is\_valid():

form\_prof.save()

return redirect('sales')

else:

form\_prof = ProfileFrom()

return render(request, 'itservice/profile.html', {'user': user, 'form': form\_prof})

class ProfileUpdateView(UpdateView):

model = users

form\_class = ProfileUpdateForm

template\_name = 'itservice/profile\_edit.html'

def get\_object(self, queryset=None):

return self.request.user

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

if self.request.POST:

context['user\_form'] = ProfileFrom(self.request.POST, instance=self.request.user)

else:

context['user\_form'] = ProfileFrom(instance=self.request.user)

return context

def form\_valid(self, form):

context = self.get\_context\_data()

user\_form = context['user\_form']

with transaction.atomic():

if all([form.is\_valid(), user\_form.is\_valid()]):

user\_form.save()

form.save()

else:

context.update({'user\_form': user\_form})

return self.render\_to\_response(context)

return super(ProfileUpdateView, self).form\_valid(form)

def get\_success\_url(self):

return reverse\_lazy('profile')

def Clients(request):

if request.method == 'POST':

form\_client = AddClientFrom(request.POST)

if form\_client.is\_valid():

form\_client.save()

return redirect('clients')

else:

form\_client = AddClientFrom()

clients = client.objects.all()

return render(request, 'itservice/clients.html', {'clients': clients, 'form': form\_client})

def Inventory(request):

if request.method == 'POST':

form\_invent = AddInventFrom(request.POST)

if form\_invent.is\_valid():

form\_invent.save()

return redirect('inventory')

else:

form\_invent = AddInventFrom()

invent = inventory.objects.all()

return render(request, 'itservice/inventory.html', {'invent': invent, 'form': form\_invent})

def Orders(request):

if request.method == 'POST':

form\_order = AddOrdersFrom(request.POST)

if form\_order.is\_valid():

form\_order.save()

return redirect('orders')

else:

form\_order = AddOrdersFrom()

order = orders.objects.all()

return render(request, 'itservice/orders.html', {'order': order, 'form': form\_order})

def Sales(request):

if request.method == 'POST':

form\_sales = AddSalesFrom(request.POST)

if form\_sales.is\_valid():

form\_sales.save()

return redirect('sales')

else:

form\_sales = AddSalesFrom()

sale = sales.objects.all()

return render(request, 'itservice/sales.html', {'sale': sale, 'form': form\_sales})

def Tasks(request):

if request.method == 'POST':

form = AddTaskFrom(request.POST)

if form.is\_valid():

form.save()

return redirect('tasks')

else:

form = AddTaskFrom()

task = tasks.objects.all()

return render(request, 'itservice/tasks.html', {'task': task, 'form': form})

def Change\_Task\_st(request):

body = json.loads(request.body)

task = tasks.objects.get(pk=body["task\_id"])

task.task\_active = not task.task\_active

task.save()

return HttpResponse("true")

def Change\_appl\_status(request):

body = json.loads(request.body)

applications = application.objects.get(pk=body["application\_id"])

applications.application\_active = not applications.application\_active

applications.save()

return HttpResponse("true")

def Change\_Order\_st(request):

body = json.loads(request.body)

order = orders.objects.get(pk=body["order\_id"])

order.order\_active = not order.order\_active

order.save()

return HttpResponse("true")

def Аpplications(request):

appl = application.objects.all()

return render(request, 'itservice/applications.html', {'appl': appl})

def st(request):

us = users.objects.all()

return render(request, 'itservice/staff.html', {'us': us})

def Staff(request):

if request.method == 'POST':

form\_staff = RegStaff(request.POST)

if form\_staff.is\_valid():

form\_staff.save()

return redirect('staff')

else:

form\_staff = RegStaff()

user = request.user

us = users.objects.all()

return render(request, 'itservice/staff-add.html', {'user': user, 'us': us, 'form': form\_staff})

class LoginUser(LoginView):

form\_class = Login

template\_name = 'itservice/login.html'

def get\_success\_url(self):

print(1)

return reverse\_lazy('profile')

def Logout\_user(request):

logout(request)

return redirect('tasks')

def pageNotFound(request, exeption):

return HttpResponseNotFound('<h1>Страница не найдена</h1>')

def Download\_table(request):

sale = sales.objects.all()

data = []

columns = ['Наименование работы', 'Комментарий', 'Цена', 'Время продажи', 'Клиент']

for i in sale:

print(i)

data.append([i.sales\_name, i.sales\_comment, i.sales\_price, i.sales\_time, i.sales\_client])

df = pd.DataFrame(data, columns=columns)

df['Время продажи'] = df['Время продажи'].dt.tz\_localize(None)

df.to\_excel('itservice/Exel/Sale.xlsx', index=False)

file = open('itservice/Exel/Sale.xlsx', "rb")

return FileResponse(file)

Код файла settings.py, в котором реализованы все настройки:

"""

Django settings for Management project.

Generated by 'django-admin startproject' using Django 4.2.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/4.2/ref/settings/

"""

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

from django.template.defaultfilters import join

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-c1s5#+bukc^nuimrj+z=9(q^@a#9phaa3jwqi#7z42@h(vv@m='

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ['127.0.0.1']

#

AUTH\_USER\_MODEL = "itservice.users"

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'itservice.apps.ItserviceConfig',

'django\_extensions'

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'Management.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'Management.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'ru'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.2/howto/static-files/

# Default primary key field type

# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

STATIC\_URL = '/static/'

STATIC\_DIR = join(BASE\_DIR, 'static')

STATICFILES\_DIRS = [STATIC\_DIR]

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

# EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

# EMAIL\_USE\_TLS = True

# EMAIL\_HOST = 'smtp.mail.ru'

# EMAIL\_HOST\_USER = 'itservicediplom1@mail.ru'

# EMAIL\_HOST\_PASSWORD = 'itservice533'

# EMAIL\_PORT = 465

GRAPH\_MODELS = {

"all\_applications": True,

"group\_models": True,

}

Код файла urls.py, в котором реализованы все маршруты:

from django.contrib import admin

from django.conf.urls.static import static

from django.template.defaulttags import url

from django.urls import path

from Management import settings

from itservice import views

from itservice.views import \*

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.Home, name="home"),

path('login/', LoginUser.as\_view(), name="login"),

path('logout/', Logout\_user, name="logout"),

path('profile/', views.Profile, name="profile"),

path('profile-edit/', ProfileUpdateView.as\_view(), name="profile-edit"),

path('clients/', views.Clients, name="clients"),

path('sales/', views.Sales, name="sales"),

path('orders/', views.Orders, name="orders"),

path('orders-change/', views.Change\_Order\_st, name="orders-change"),

path('inventory/', views.Inventory, name="inventory"),

path('tasks/', views.Tasks, name="tasks"),

path('tasks-change/', views.Change\_Task\_st, name="tasks-change"),

path('applications/', views.Аpplications, name="applications"),

path('staff/', views.st, name="staff"),

path('staff-add/', views.Staff, name="staff-add"),

path('applications-add/', views.Application, name="applications\_add"),

path('applications-change/', views.Change\_appl\_status, name="applications-change"),

path('home/', views.Home, name="home"),

path('review/', views.Rev, name="review"),

path('review-add/', views.Review, name="review\_add"),

path('sales-down/', views.Download\_table, name="sales-down"),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

handler404 = pageNotFound

**ПРИЛОЖЕНИЕ 2. ПОЛИТИКА БЕЗОПАСНОСТИ**

**ПРИЛОЖЕНИЕ 3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Документ «Руководство пользователя» относится к пакету эксплуатационной документации. Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой или автоматизированной системой. Таким образом, документ Руководство пользователя должен отвечать на следующие вопросы: что это за программа, что она может, что необходимо для обеспечения ее корректного функционирования и что делать в случае отказа системы.

**1. ВВЕДЕНИЕ**

**1.1 Область применения**

Система управления взаимоотношениями с клиентами, позволяющая автоматизировать процесс работы и организовать работу отдела маркетинга внутри небольшого предприятия.

**1.2 Краткое описание возможностей**

Система обладает следующими функциями:

• Авторизация в системе с использованием логина и пароля;

• Регистрация новых сотрудников;

• Авторизация в админ-панели;

• Добавление и обработка заявок;

• Добавление и редактирование задач, заказов, клиентов, сотрудников и продаж;

• Экспорт;

• Управление учетной записью

• Просмотр справки о странице

**1.3 Уровень подготовки пользователя**

Пользователь, работающий в информационной системе, должен обладать следующими знаниями:

–  Иметь опыт работы с операционной системой;

–  Иметь опыт с работой ПК и утилитами.

**1.4 Перечень эксплуатационной документации**

Пользователи должны быть ознакомлены с "Руководством пользователю".

**2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ**

Основная цель АИС - хранение, обеспечение эффективного поиска и передачи информации по соответствующим запросам для наиболее полного удовлетворения информационных запросов большого числа пользователей.

Для работы с клиентской частью web-приложения необходимо современный браузер, с поддержкой HTML5, CSS 3 и ECMAScript 2015 (ES6).

**3. ПОДГОТОВКА К РАБОТЕ**

**3.1 Состав и содержание носителя данных**

Папка программного продукта содержит файлы разрешений HTML, CSS, Python, JavaScript.

**3.2 Порядок загрузки программ и данных**

Систему устанавливает на предприятие Администратор и сам добавляет всех сотрудников в систему.

**3.3 Порядок проверки работоспособности**

Проверка работоспособности автоматизированной информационной системы IT-сервиса обычно включает в себя следующие шаги:

1. Проверка наличия необходимого программного обеспечения и его актуальности. Это включает в себя проверку версий используемых компонентов системы, проверку наличия необходимых библиотек и других зависимостей.

2. Проверка настроек системы. Необходимо убедиться, что конфигурационные файлы и параметры системы настроены правильно.

3. Проверка доступности ресурсов. Необходимо убедиться, что все используемые ресурсы, такие как базы данных, серверы и сетевое оборудование, доступны и работают нормально.

4. Проведение тестирования производительности. Необходимо убедиться, что система может обрабатывать большие объемы данных и нагрузки без существенных задержек или ошибок.

5. Проверка функциональности системы. Необходимо убедиться, что все функциональные возможности системы работают корректно и соответствуют требованиям заказчика.

6. Проверка авторизации. Необходимо убедиться, что система должным образом регистрирует все события и ошибки, чтобы обеспечить возможность быстрой диагностики проблем в случае их возникновения.

**4. ОПИСАНИЕ ОПЕРАЦИЙ**

**1) Редактирование профиля**

Для этого необходимо нажать на кнопку «Редактировать профиль» на странице Профиля (Рис. 4.1.)

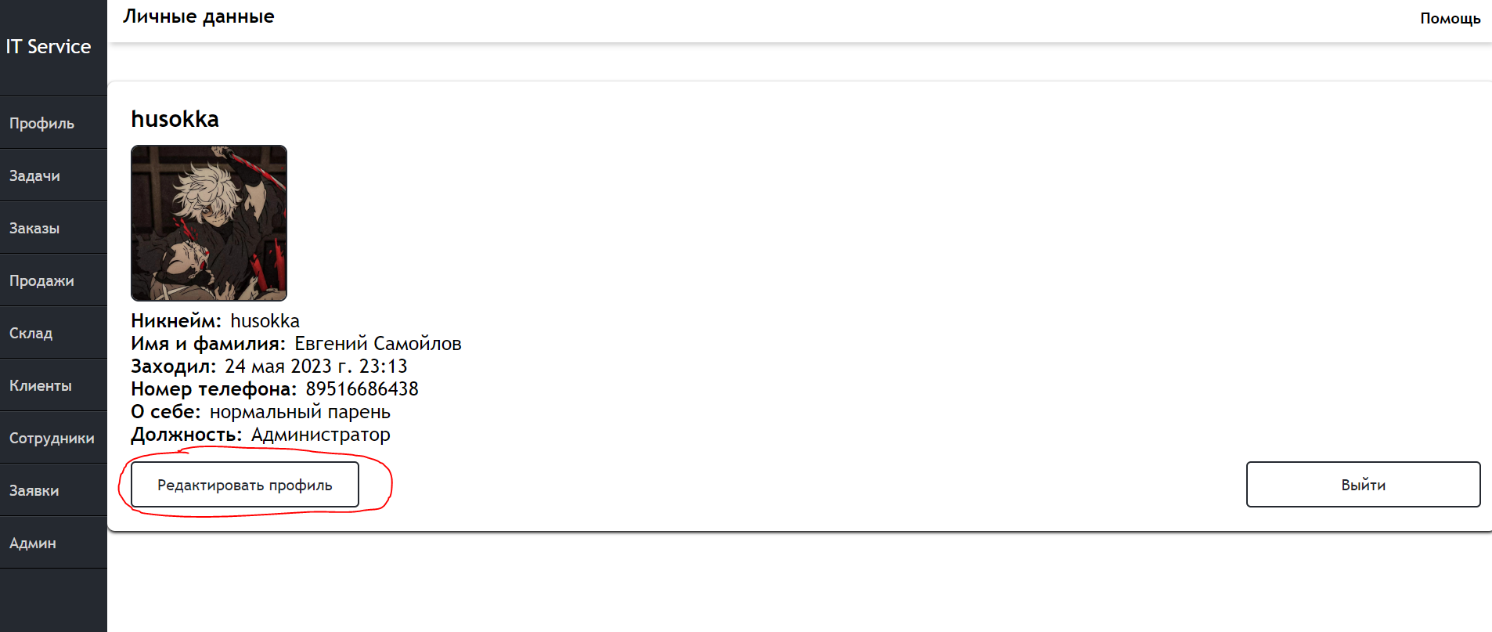


Рис 4.1. Кнопка «Редактировать профиль»

Происходит переход на страницу редактирования профиля (Рис 4.2.), нужно внести нужные изменения и нажать на кнопку подтверждения изменений.

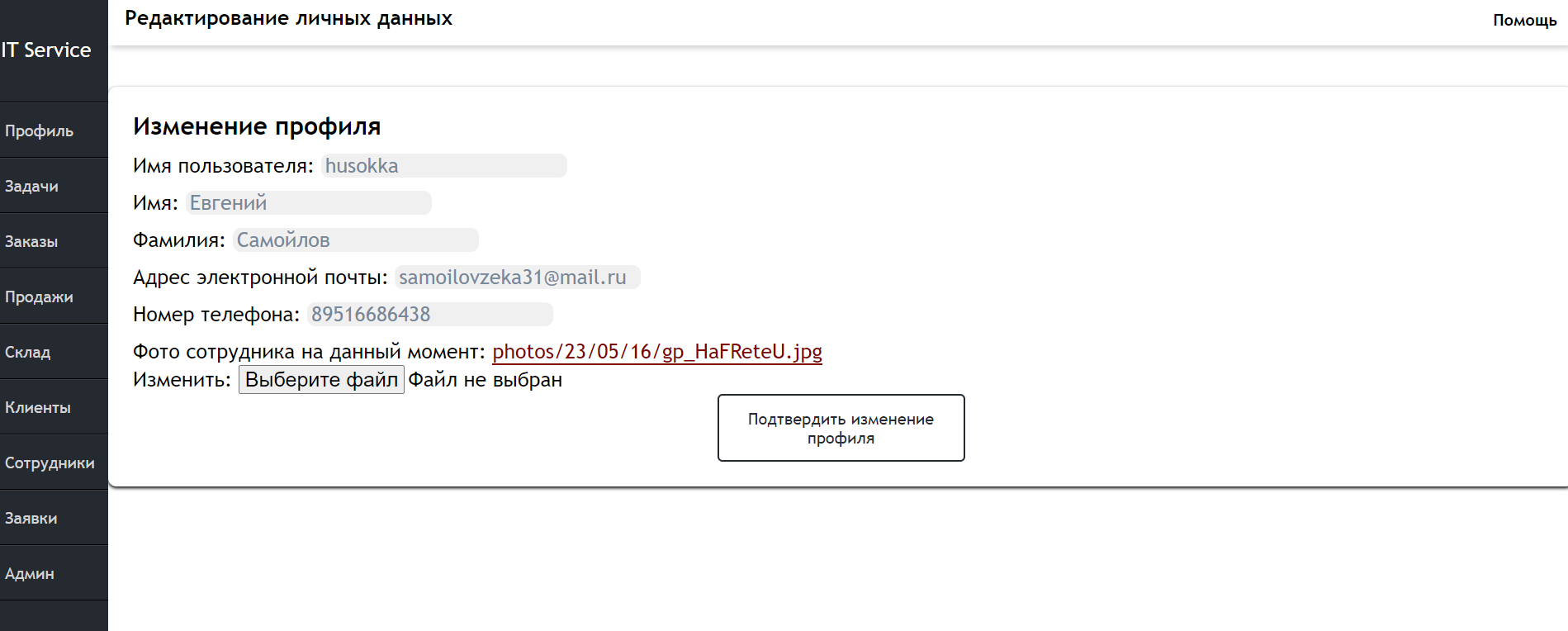


Рис 4.2. Страница редактирования профиля

**2)Добавление задачи**

Нужно перейти на вкладку «задачи» и нажать на кнопку «Добавить задачу» (Рис. 4.3.), после этого откроется модальное окно добавления задачи (Рис. 4.4.)

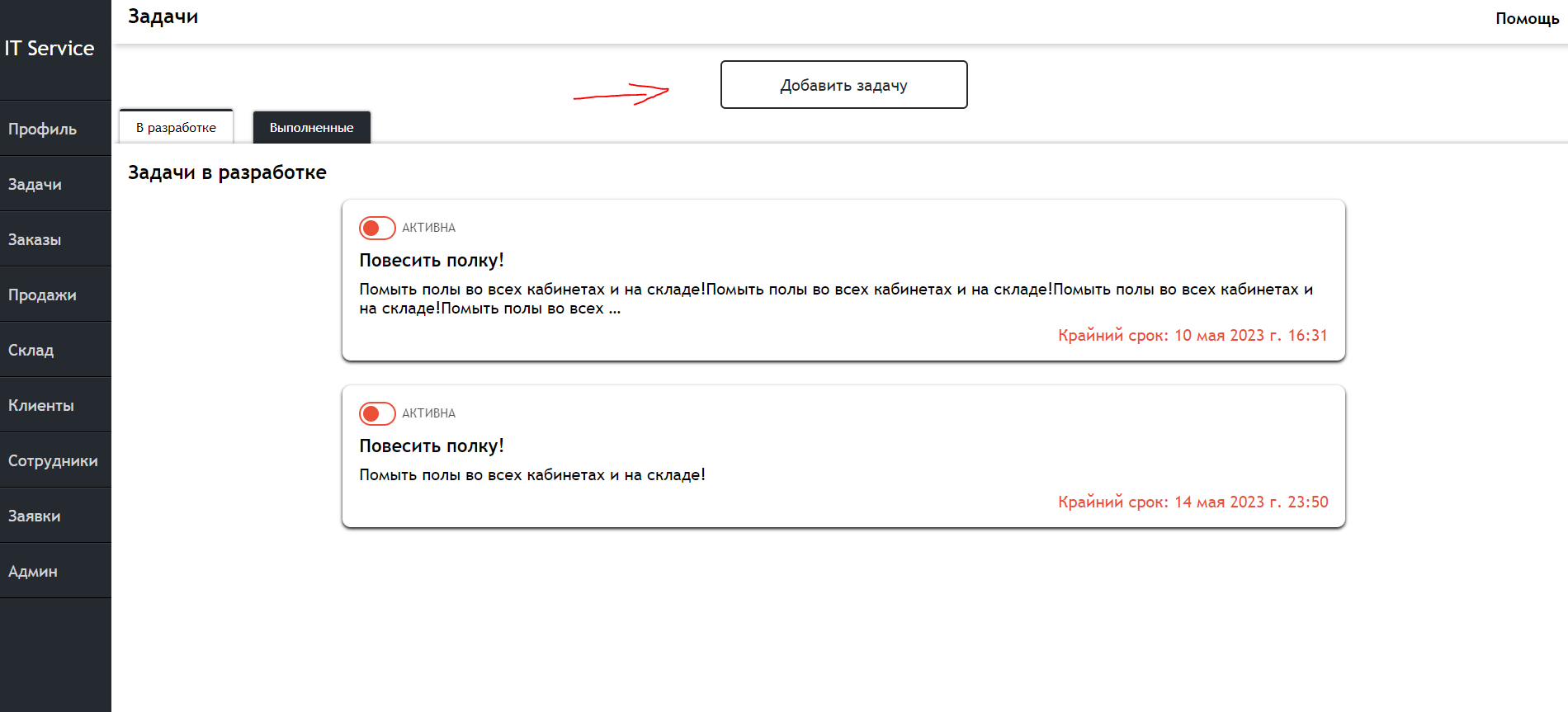


Рис 4.3. Страница Задач

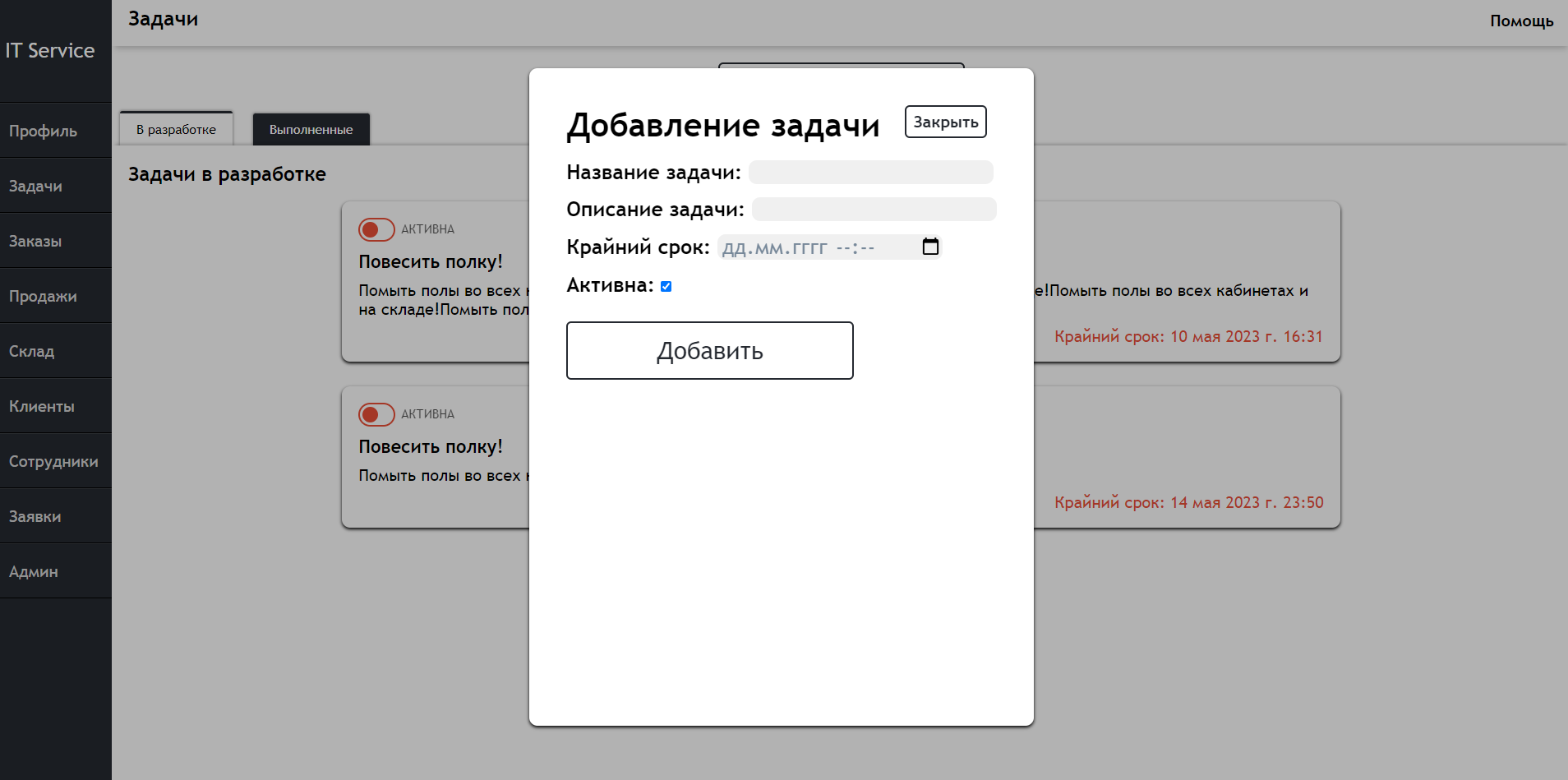


Рис 4.3. Модальное окно добавления задач

**5 АВАРИЙНЫЕ СИТУАЦИИ**

При возникновении аварийных ситуаций сообщите об этом администрации сайта, чтобы оперативно принять необходимые меры для устранения проблемы и предотвратить ее повторное возникновение в будущем.

Если вы обнаружили ошибку на сайте или заметили необычное поведение системы, пожалуйста, свяжитесь с администратором как можно скорее.

**6. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ**

Для успешной работы с системой, сотрудник должен:

–  Иметь опыт работы с операционной системой;

–  Иметь опыт с работой ПК и утилитами.

**ПРИЛОЖЕНИЕ 4. РУКОВОДСТВО АДМИНИСТРАТОРА**

Руководство администратора – это составная часть эксплуатационной документации, которая разрабатывается на любую программу или автоматизированную систему. При помощи руководства администратора ответственные пользователи системы получают возможность управлять ее функционированием – выполнять определенные операции по обеспечению порядка работы АСУ, распределять права доступа к ней, редактировать данные и исправлять ошибки.

Руководство администратора адресовано лицу, задача которого — обеспечить определенный порядок функционирования системы. Обычно администратор считается пользователем системы, однако, при этом он наделен как особыми обязанностями, так и необходимыми для их выполнения привилегиями.

**1.НАЗНАЧЕНИЕ ПРОГРАММЫ**

**1.1 Назначение программы**

Цель системы – автоматизация рабочего процесса сервиса, с целью экономии времени

**1.2 Функции программы**

**1.2.1 Авторизация**

Авторизация в системе с использованием логина и пароля.

**1.2.2 Регистрация**

Добавление администратором новых сотрудников в систему.

**1.2.3 Админ-панель**

Авторизация в админ-панели.

**1.2.4 Добавление и изменение информации**

Добавление и редактирование задач, заказов, клиентов, сотрудников и продаж.

**1.2.5 Обработка заявок**

Добавление и обработка заявок.

**1.2.6 Экспорт файлов**

Экспорт информации о продажах

**1.2.7 Управление учетной записью**

Редактирование профиля в личном кабинете

**2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

**2.1 Аппаратные требования**

Для работы с клиентской частью web-приложения необходимо современный браузер, с поддержкой HTML5, CSS 3 и ECMAScript 2015 (ES6). Например:

• Google Chrome;

• Yandex браузер;

• Opera;

• Safari.

**2.2 Программные требования**

Для использования системы необходимы следующие системные требования:

ОС: Window 7, 8,10,11

Процессор: 1.66 GHz intel 5i

Оперативная память: 1024 MB ОЗУ

Видеокарта: HD Graphics

DirectX: Версии 10

**3. ВЫПОЛНЕНИЕ ПРОГРАММЫ**

**3.1. Загрузка и запуск программы**

**3.1.1 Загрузка программы**

Система находится на сервере. Добавляется Администратор, далее Администратор добавляет остальных сотрудников в систему

**3.1.2 Запуск программы**

Запуск выполняется системы выполняется администратором.

**3.2 Выполнение программы**

**3.2.1 Главное окно**

На главном экране отображаются 2 кнопки для «Отзывов», «Заявок» и кнопка для персонала «Вход в систему»

**3.2.1 Страница Профиля**

На странице Профиля представлена информация о сотруднике, так же есть кнопка редактирования профиля и кнопка выхода.

**3.2.2 Страница Задачи**

На странице рекомендаций представлены задачи коллектива, они подразделены на «В разработке» и «Выполненные», с возможностью их добавления.

**3.2.3 Страница Заказы**

На странице представлены заказы предприятия, они подразделены на «Активные» и «Выполненные», с возможностью их добавления.

**3.2.4 Страница Продажи**

На странице представлены продажи, с возможностью скачать Excel файл с информацией о продажах, с возможностью их добавления.

**3.2.5 Страница Склад**

На странице представлена информация о складе предприятия, с возможностью их добавления.

**3.2.6 Страница Клиенты**

На странице представлена информация о клиентах предприятия, с возможностью их добавления.

**3.2.7 Страница Сотрудники**

На странице представлена информация о сотрудниках предприятия, с возможностью их добавления Администратором.

**3.2.8 Страница Заявки**

На странице представлены заявки клиентов, они подразделены на «Необработанные» и «Обработанные».

**4.** **СООБЩЕНИЯ АДМИНИСТРАТОРУ**

Сообщения администратору, посредством графического интерфейса пользователя, появляются во всплывающем окне.

Типы сообщений об ошибках:

1) Некорректный пароль при регистрации нового сотрудника

Текст: «Введенные пароль слишком похож на имя пользователя»

Действия: Придумать более сложный пароль.

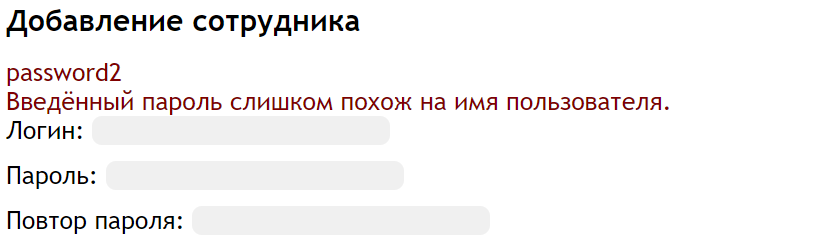


Рисунок 1. Отображение «Защита легкого пароля»

1) Некорректный логин при регистрации нового сотрудника

Текст: «Пользователь с таким именем уже существует»;

Действия: Использовать другое имя пользователя.

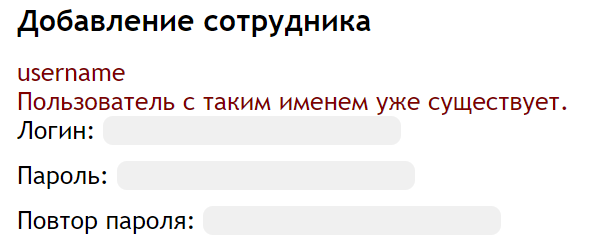


Рисунок 2. Ошибка «Пользователь с таким именем уже существует»

2) Попытка перейти на страницу системы неавторизованным

Текст: «Вы не авторизованы»;

Действия: Перейти на окно логина и авторизоваться.

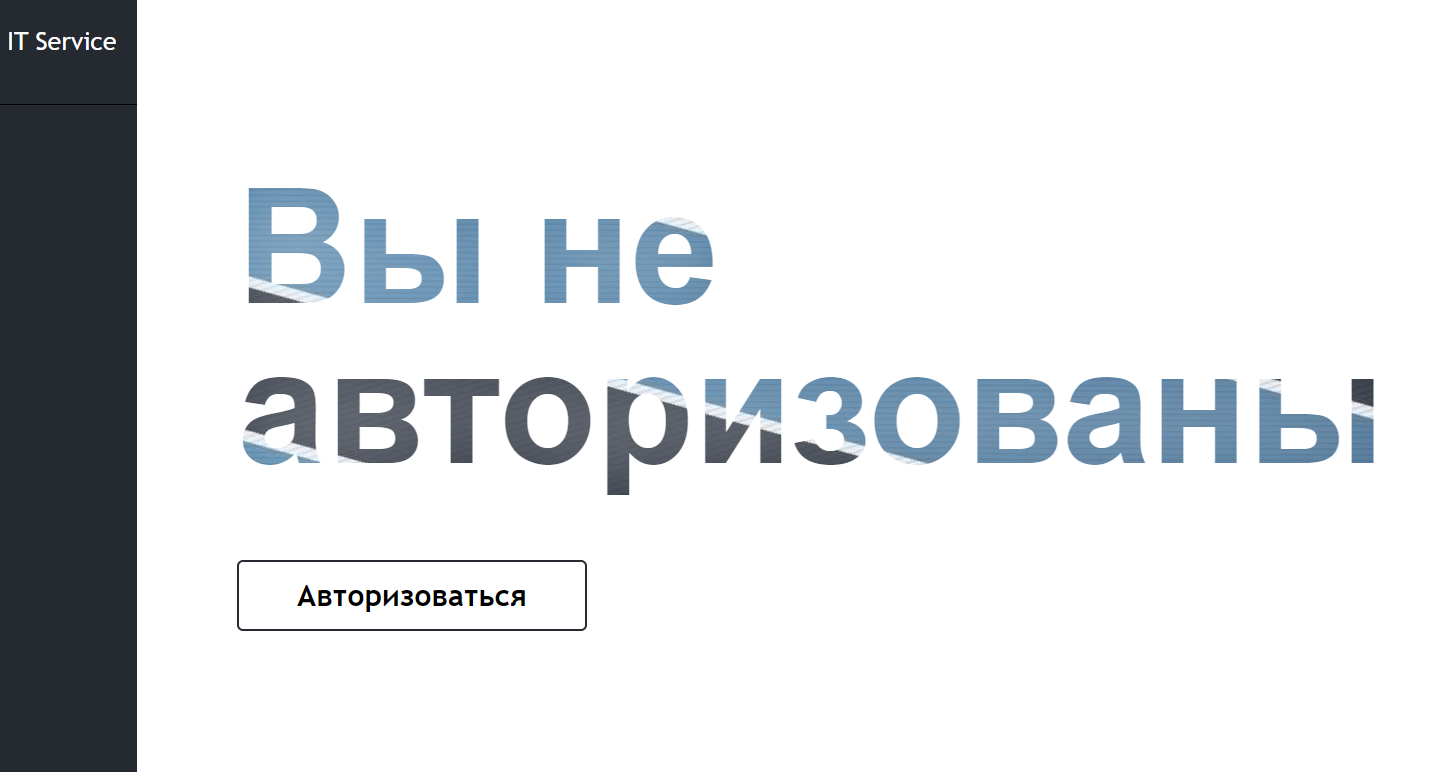


Рисунок 3. «Неавторизованный сотрудник»