МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Разработка социальных и экономических информационных систем
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка приложения «Коммунальные услуги. Личный кабинет потребителя»

Обучающийся И.А. Лимарь
(Инициалы Фамилия) (личная подпись)

Руководитель к. п. н., доцент, О.Ю. Копша
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к. п. н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Темой данной выпускной квалификационной работы является "Коммунальные услуги. Личный кабинет потребителя".

Бакалаврская работа описана на 48 страницах, включающие 23 рисунка, 2 таблицы и списка литературы из 20 пунктов, в том числе 5 иностранных источников

Объектом исследования в данной работе является деятельность компании «Квартплата 24». Предметом исследования бакалаврской работы – усовершенствование процесса взаимодействия пользователя со своими коммунальными услугами посредством информационных технологий.

Ключевым вопросом дипломной работы является разработка веб-приложения, которое позволит решить проблему отсутствия автоматизации процесса управления своими коммунальными услугами.

Дипломную работу можно разделить на несколько взаимосвязанных частей: анализ существующих решений в данной области в рамках модели "AS IS", разработка контекстной модели "TO BE" для выявления необходимых улучшений, логическое проектирование веб-приложения, включая сценарии использования и диаграммы, а также выбор технологий и программных средств для разработки.

Наконец, в практической части мы рассматриваем разработку веб-приложения, его интерфейса и серверной части с последующим тестированием. Описана реализация таких функциональных возможностей, как регистрация и аутентификация пользователей, управление профилем, ввод показаний счетчиков, управление платежами, запросы в экстренные службы.

В заключение хотелось бы подчеркнуть, что данная работа актуальна для решения проблемы автоматизации взаимодействия пользователя с коммунальными службами и может существенно повысить эффективность управления лицевыми счетами за коммунальные услуги.

Abstract

The title of the graduation work is Development of a web application for the Personal Account of Utility Service Consumers."

The graduation work consists of an explanatory note on 48 pages, introduction, including 23 figures, 2 tables, the list of 20 references including 5 foreign sources.

The object of the graduation work is the interaction of users with public utilities.

The subject of the graduation work is a web application that automates the process of users interacting with their utilities.

The key issue of the thesis is the development of a custom application that will solve the problem of lack of automation in the process of managing one's utilities.

The graduation work may be divided into several connected parts which are: the analysis of existing solutions in the domain within the framework of the "AS IS" model, development of the contextual model "TO BE" to identify necessary improvements, logical design of the web application including use cases and diagrams, and the selection of technologies and software tools for development.

Finally, in the practical part, we consider the development of the web application, its interface, and the server part, followed by testing. We describe the implementation of functionalities such as user registration and authentication, profile management, meter readings input, payment management, and emergency service requests.

In conclusion, we'd like to stress that this work is relevant in solving the problem of automation in user-utility interactions and can significantly improve the efficiency of managing personal utility accounts.

Оглавление

Введение

В современном мире самой важной частью повседневной жизни человека стали информационные технологии. Их можно использовать для быстрого получения информации, ведения бизнеса, создания контента и многих других вещей. Но, пожалуй, главным преимуществом информационных технологий является оптимизация и облегчение ежедневных рутинных задач. И администрирование своего жилищно-коммунального хозяйства, несомненно, является такой задачей.

Для удобной оплаты, внесения и изменение показаний счетчиков, отправки заявок и прочих жилищно-коммунальных услуг, несомненно, потребуется веб-приложение. С помощью веб-приложения есть возможность реализовать весь функционал, необходимый для личного кабинета рядового потребителя жилищно-коммунальных услуг.

Целью выпускной квалификационной работы является разработка личного кабинета для потребителя жилищно-коммунальных услуг, который имел бы функционал авторизации и регистрации, управления личными данными, внесения показателей счетчиков, вывод истории платежей и показаний приборов, функционал отправки заявки в аварийную службу и раздел с помощью.

Объектом исследования в данной работе является деятельность компании «Квартплата 24». Предметом исследования бакалаврской работы – усовершенствование процесса взаимодействия пользователя со своими коммунальными услугами посредством информационных технологий.

Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ предметной области и компании «Квартплата 24»;

– рассмотреть аналогичные решения на рынке;

– выбрать технологии и программные средства для разработки;

– разработать клиентскую и серверную часть веб-приложения.

Глава 1 Анализ предметной области

1.1 Характеристика отрасли ЖКХ

Перед началом анализа предметной области необходимо четко определить, что такое ЖКХ и какова его структура.

В общих чертах, жилищно-коммунальное хозяйство – это самостоятельная многоуровневая система, которая отвечает за услуги, необходимые для жизнедеятельности граждан и предприятий. Эти услуги имеют широкий спектр и могут быть выражаться в материальном виде, как предоставление воды и электричества, так и в виде обслуживания домов, уборки мусора и т.д. На рисунке 1 представлена общая схема устройства жилищно-коммунальных служб.

Рисунок 1 – Схема устройства ЖКХ

ЖКХ необходимо для полноценного и комфортабельного функционирования любых населенных пунктов и имеет специфическую

структуру в зависимости от их типа и размера. В составе ЖКХ можно выделить много пунктов работ, но в основном они подразделяются на две главные категории:

– снабжение,

– поддержание.

Снабжение представляет собой предоставление населению благ, необходимых для утоления их нужд. Основными из них являются водопровод (в т.ч. канализация), газопровод, электропровод и теплоэнергетика. Поддержание выражается в сохранении и благоустройстве структурами ЖКХ уже готовых объектов. Структурно оно состоит из планового ремонта, уборки и утилизации отходов, технического обслуживания инфраструктурных сооружений, озеленения и дорожных служб [20].

1.2 Характеристика компании «ООО Квартплата 24»

«Квартплата 24» — ИТ-компания, эффективно решающая задачи расчета и учета платы за жилищно-коммунальные услуги, приема и распределения платежей, востребования долгов в полном соответствии с законодательством для товариществ собственников жилья, управляющих и ресурсоснабжающих организаций, информационно-расчетных центров на всей территории РФ [6].

Организация ведет свою деятельность с 1996 года и за это время набрала богатый опыт в области информационных технологий. На счету компании более 1 миллиона лицевых счетов из 70 субъектов РФ. Это было достигнуто благодаря современной экосистеме из более чем десяти интегрированных между собой облачных сервисов.

Компания предлагает более 200 способов оплаты, в том числе электронную, и является единственный в РФ платежным сервисом, обеспечивающим моментальным прием и распределение платежей за ЖКУ на всех этапах его прохождения.

Среди прочих, основными преимуществами организации значатся: высокая степень автоматизации; команда, состоящая из экспертов; высокая платежная дисциплина; соответствие расчета законам Российской Федерации. Также компания числится резидентом современных инновационных центров и технопарков, таких как «Сколково» и «Жигулевская долина». Помимо этого, Минстрой России включил компанию в современные проекты «Банк решений Умного города» и «Банк эффективных технологий в ЖКХ».

На рисунке 2 представлена организационная структура организации.

Рисунок 2 – Организационная структура «Квартплата 24»

Организационная структура предприятия начинается с генерального директора, который ответственен за общее виденье и стратегию компании. Он руководит командой, делегирует полномочия и настраивает эффективную работу. В его обязанности также входит анализ финансового состояния компании, контроль бюджета и обеспечение финансовой устойчивости.

Генеральный директор следит за тенденциями и инновациями в своей сфере, находит инвесторов и ведет переговоры с партнерами.

Исполнительный директор отвечает за своих прямых подчинённых – директоров отдельных областей. Директор по разработке следит за полным процессом создания и поддержания разрабатываемого программного продукта. У него в подчинении находится отдел разработки ПО, тестировщики и отдел безопасности. Административный директор отвечает за координацию трех небольших, но ключевых отделов – бухгалтерии, отдела кадров и юридического отдела. Директор по маркетингу ответственен за разрабатываемые маркетинговые стратегии, распространением рекламных материалов и PR-активностями.

1.3 Концептуальное моделирование предметной области

1.3.1 Разработка и анализ модели бизнес-процесса «Как есть»

Модель «Как есть» позволяет систематизировать протекающие в данный момент процессы, а также используемые информационные объекты. На основе этого выявляются в организации и взаимодействии бизнес-процессов, определяется необходимость тех или иных изменения в существующей структуре. Такую модель часто называют функциональной и выполняют с использованием различных графических нотаций и case-средств. На этапе построения модели AS-IS важным считается строить максимально приближенную к действительности модель, основанную на реальных потоках процессов, а не на их идеализированном представлении. Проектирование информационных систем и управление процессами подразумевает построение модели AS IS и дальнейший переход к модели TO-BE, что является залогом автоматизации «правильных», усовершенствованных процессов [5].

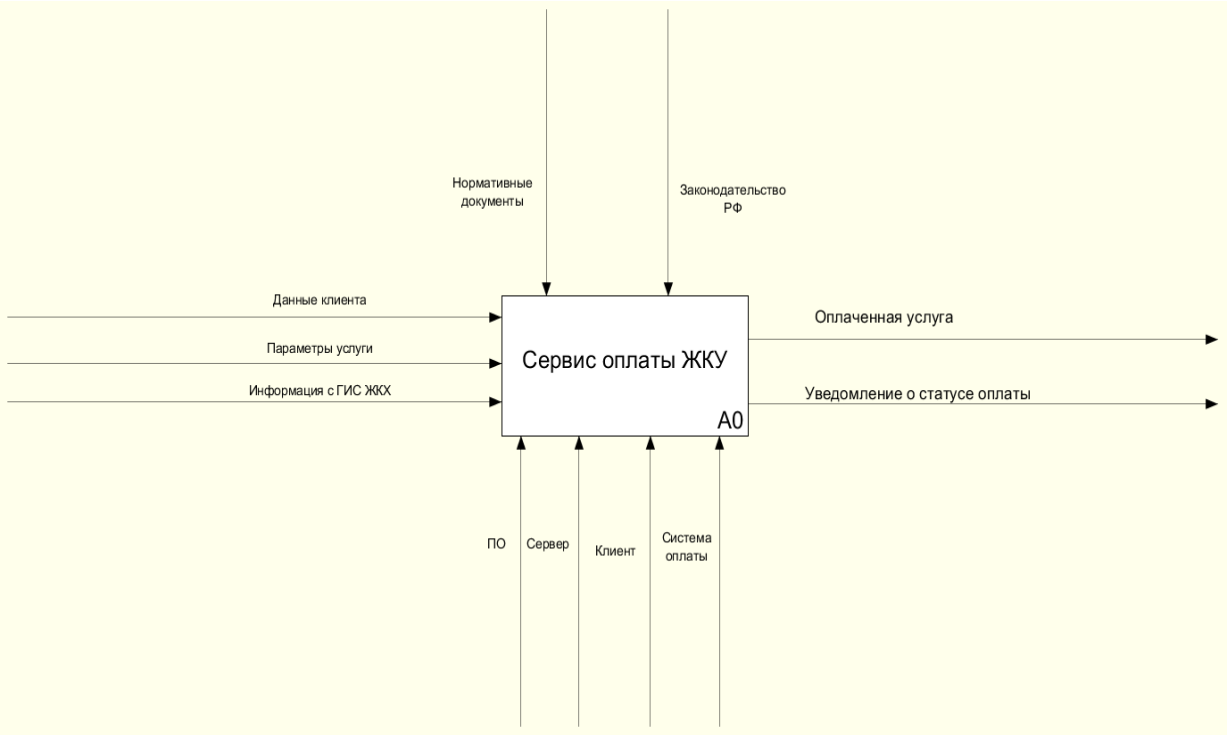

Рисунок 3 – Диаграмма IDEF0 верхнего уровня модели «Как есть»

В данной модели есть 3 входных потока: данные клиента, параметры услуги и информация с ГИС ЖКХ. Рассмотрим каждый поток отдельно:

– данные клиента: в этом потоке описывается все известные данные, необходимые для оплаты услуг (ФИО, лицевой счет, место проживания);

– параметры услуги: тут описывается необходимая сумма к оплате, период оплаты и т.п.;

– информация с ГИС: в компании интегрирован государственный сервис расчета, другими словами, обмен данными происходит без участия клиента.

Механизмы в данной бизнес-модели – программное обеспечение, серверная архитектура и клиенты.

Основная деятельность процесса «Оплата ЖКУ» заключается в предоставлении клиентам возможности оплаты через номер лицевого счета на сайте и представлена на рисунке 4.

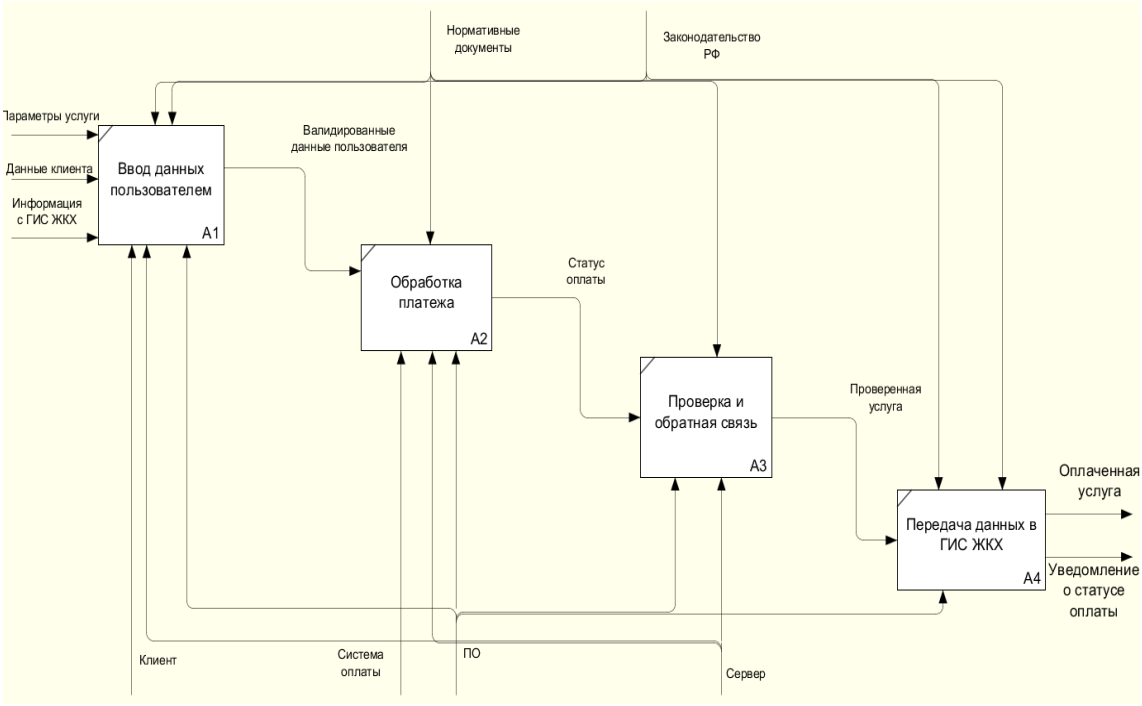

Рисунок 4 – Декомпозиция первого уровня модели «Как есть»

Распишем процессы декомпозиции модели первого уровня:

– получение и валидация данных пользователя: обработка вводимых пользователем данных, проверка на корректность;

– обработка платежа и взаимодействие с системой оплаты: валидация данных в систему оплаты, взаимодействие с платежной системой для выполнения транзакции;

– управление ошибками и обратная связь: обработка возможных ошибок в процессе оплаты и предоставление пользователю соответствующих уведомлений и дальнейших инструкций;

– передача данных в ГИС ЖКХ: синхронизация показания введённых пользователем показателей счетчиков с общей государственной информационной системой.

1.3.2 Анализ модели и определение недостатков в существующих бизнес-процессах

После детального анализа модели «Как есть» был сделан вывод, что существующая система в данный момент имеет следующие недостатки:

– при оплате через лицевой счет пользователю приходится вручную вводить номер лицевого счета, сумму платежа и другую информацию, это достаточно времязатратно, особенно если у пользователя нет всей необходимой информации под рукой;

– нет возможности автоматического обновления информации о текущем состоянии счета и доступных услуг, пользователю приходится вручную отслеживать свои платежи и остатки;

– доступны только базовые функции оплаты жилищно-коммунальных услуг, в то время как личный кабинет может предоставить более широкий спектр возможностей, такие как просмотр истории платежей;

– поскольку процесс включает ручной ввод данных, есть больше возможностей для возникновения ошибок, такие как неправильно введенные данные или неверно указанные суммы платежей.

Для решения вышеописанных проблем было решено разработать личный кабинет потребителя коммунальных услуг для компании «Квартплата 24». Личный кабинет будет являться веб-приложением с современным удобным интерфейсом и грамотно разработанной backend частью.

1.4 Анализ существующих аналогов

Рассмотрим функционал личного кабинета ikvp, созданный управляющей компанией «Волжские коммунальные системы». Для входа в личный кабинет используется данные с бумажной квитанции об оплате, что не является оптимальным решением.

После успешного входа пользователь попадает в профиль, где есть три формы для добавления трех разных номеров телефона (контактный, сотовый и рабочий) и одна форма для адреса электронной почты. В профиле отсутствует дополнительный функционал, такой как возможность изменить пароль и включить уведомления. Также три формы для разных телефонов является устаревшей практикой не дающей прямой пользы.

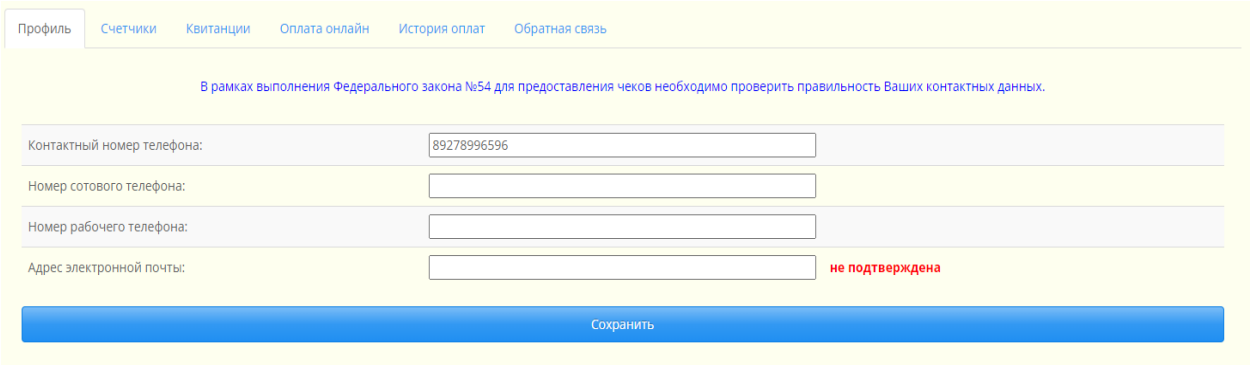

Рисунок 5 – Профиль сайта компании «Волжские коммунальные системы»

Из преимуществ данного личного кабинета можно выделить:

– нет необходимости регистрироваться, вход через готовые данные;

– быстрая скорость работы;

– возможность оставить обратную связь.

Также личный кабинет обладает рядом недостатков:

– сильно ограниченный функционал профиля;

– нет возможности зарегистрироваться отдельно;

– устаревший дизайн;

– нет возможности защитить свою учетную запись;

– нельзя самостоятельно вписывать показания счетчиков.

1.5 Постановка задачи на разработку

Исходя из выполненного анализа существующих разработок, выполненных в прошлой главе, можно сделать вывод, что на данный момент обозначенное веб-приложение не соответствует необходимым функциональным требованиям. Следовательно, вывод, что существует потребность в компетентно выполненном личном кабинете потребителя коммунальных услуг.

Цель представленной работы - разработка веб-приложения «Личный кабинет потребителя коммунальных услуг», который будет отвечать современным стандартам проектирования и дизайна. Поэтому веб-приложение должно соответствовать поставленным критериям:

– user-friendly интерфейс;

– иметь функционал регистрации и авторизации;

– возможность защитить учетную запись;

– иметь функцию добавления показаний счетчиков;

– Вывод истории платежей и показаний;

– Иметь функционал отправки заявки в аварийную службу.

1.6 Разработка и анализ бизнес-модели «Как должно быть»

На основе модели бизнес-процесса «Как есть» была построена новая модель «Как должно быть». Она исправляет недостатки прошлой модели и совершенствует взаимодействие пользователя с системой.

Как видно на рисунках 6 и 7 главным отличием новой модели является внедрение практически в каждый процесс системы функционала веб-приложения. Оно в значительной степени автоматизирует процесс взаимодействия пользователей со своими жилищно-коммунальными услугами.

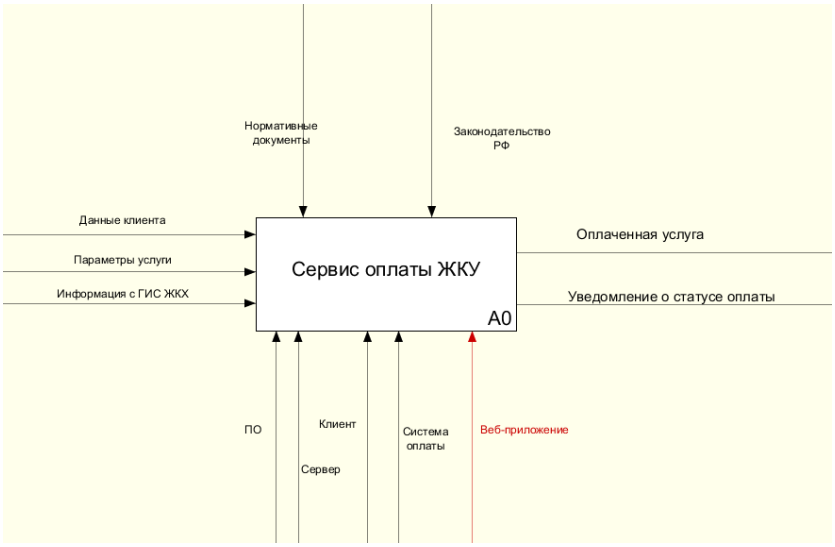

Рисунок 6 – Диаграмма бизнес-процесса модели «Как должно быть»

С помощью личного кабинета, пользователи смогут получать информацию в режиме реального времени. Для клиентов будет доступен полный функционал, где они смогут отслеживать свою историю введенных показаний, историю платежей, изменять и добавлять необходимую им информацию. С помощью данного-веб приложения пользователи смогут оставлять заявки в аварийно-диспетчерские службы, получать необходимую консультацию по работе личного кабинета в соответствующем разделе, следить за поступающими уведомлениями и сообщениями.

Одним из преимуществ новой модели является повышение эффективности. Автоматизация процессов и улучшение пользовательского интерфейса значительно повышают эффективность взаимодействия пользователей с системой, а быстрая и точная обработка данных позволяет предоставлять пользователям более качественное обслуживание.

К недостаткам можно отнести аспекты, относящиеся к затратам и потребности в обучении. Внедрение новой системы потребует инвестиций на начальном этапе, а пользователям и сотрудникам компании потребуется время на освоение новой системы и адаптации к изменениям.

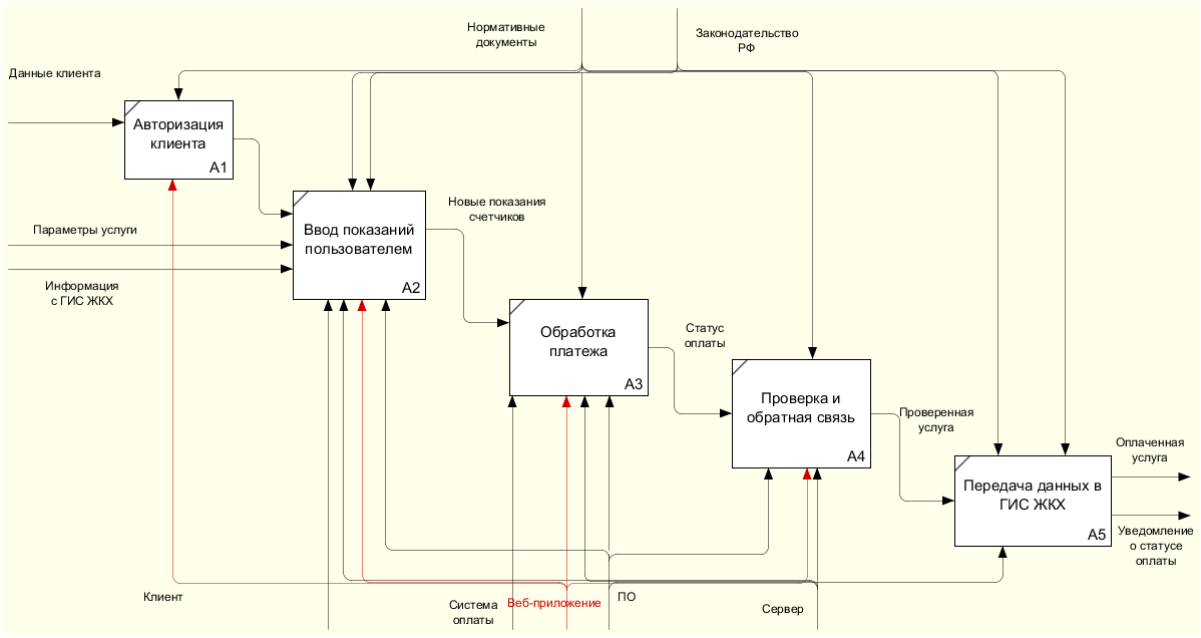

Рисунок 7 – Декомпозиция модели 1-го уровня «Как должно быть»

Благодаря внедрению личного кабинета у клиента пропадет необходимость постоянного ввода лицевого счета и появится возможность контролировать собственные жилищно-коммунальные услуги.

Выводы по главе 1

В первой главе был проведен всесторонний анализ предметной области, который включал исследование особенностей отрасли жилищно-коммунального хозяйства (ЖКХ) и деятельности компании «Квартплата 24». Этот анализ выявил ключевые проблемы и недостатки существующих систем управления коммунальными услугами. Для детального понимания текущего состояния была создана контекстная модель "как есть". На основе этой модели был проведен анализ текущих бизнес-процессов и выявлены основные проблемы, такие как неудобный интерфейс, сложность в использовании и недостаток функциональности.

Далее была разработана контекстная модель "как должно быть", которая определила направления для улучшений и будущих изменений. Эта модель включала в себя улучшение пользовательского интерфейса,

добавление новых функциональных возможностей, таких как управление профилем, внесение показаний счетчиков и управление оплатами. Также был проведен сравнительный анализ существующих на рынке решений, что подтвердило необходимость создания нового веб-приложения. Основные выводы первой главы заключаются в необходимости создания современного личного кабинета потребителя, отвечающего актуальным требованиям функциональности и удобства использования.

Глава 2 Проектирование веб-приложения «Личный кабинет потребителя»

2.1 Логическое моделирование веб-приложения

Логическое моделирование - важный аспект при разработке любого программного продукта. Оно необходимо для постройки логики данных и описания объектовой модели. При построении логической модели данных веб-приложения «Личный кабинет потребителя коммунальных услуг» использовался унифицированный язык моделирования (UML).

2.1.1 Диаграмма вариантов использования

Диаграмма вариантов использования наглядно демонстрирует и описывает, какой функционал будет доступен каждой группе пользователей в разрабатываемой программной среде. В случае с личным кабинетом, основной группой пользователей будут клиенты, которые взаимодействуют с коммунальными услугами внутри личного кабинета с помощью определенных вариантов использования. Но также может быть введен актер «Администратор», который отвечает за управление и администрирование личный кабинетом потребителей при возникновении каких-либо проблем. Администратор имеет расширенные права и работает в определенной среде, позволяющей ему управлять пользователями.

Далее представлены варианты использования для клиента и IT-администратора:

Клиент (пользователь сайта):

– ввод логина и пароля: пользователь вводит свои ученые данные для входа в личный кабинет;

– регистрация: новый пользователь проходит процесс регистрации в личном кабинете;

– просмотр счетов: клиент просматривает список своих счетов за коммунальные услуги;

– просмотр уведомлений: клиент проверяет уведомления о новых услугах, изменениях в работе и т.д.;

– оплата счетов: пользователь осуществляет оплату своих счетов за коммунальные услуги через личный кабинет;

– управление настройками: пользователь управляет настройками своего аккаунта, такие как изменения пароля, контактная информация и т.д.;

– выход из системы: пользователь завершает сеанс работы с личным кабинетом и выходит из системы.

Диаграмма вариантов использования изображена на рисунке 8.

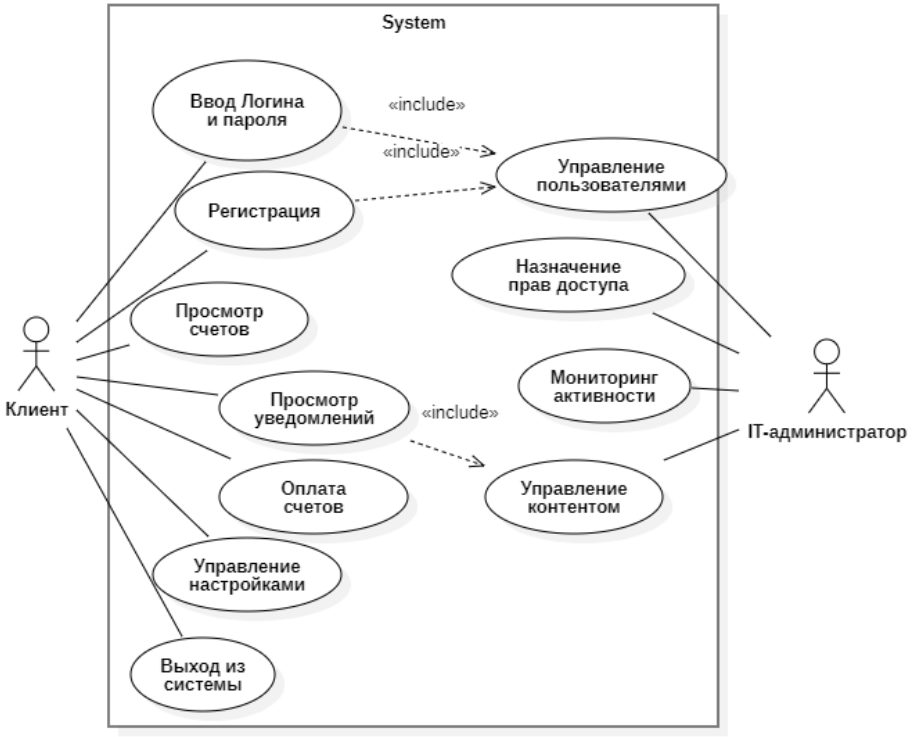

Рисунок 8 – Диаграмма вариантов использования веб-приложения

IT-Администратор компании:

– управление пользователями: администратор может просматривать список зарегистрированных пользователей, добавлять новых, редактировать информацию, а также блокировать и удалять учетные записи;

– назначение прав доступа: у администратора доступен функционал управления правами доступа для каждого из пользователей и назначение им определенных ролей, предусмотренных системой;

– мониторинг активности: администратор может просматривать журналы активности пользователей, анализировать действия и операции, проводимые в личном кабинете;

– управление контентом: администратор имеет возможность управлять информацией, отображаемой в личном кабинете, например добавлять уведомления, редактировать страницу с помощью.

2.1.2 Диаграмма классов личного кабинета потребителя

На этом этапе осуществляется проектирование диаграммы классов UML, которая используется для визуализации структуры объектов и их взаимосвязей в системе. Оно иллюстрирует структуру системы, описывает классы, атрибуты и методы отношения между ними. Классы представляют собой абстракцию для объектов в системе. Каждый класс имеет свои поля данных и операции, которые определяют его поведение.

Диаграмма классов помогает разработчикам понять, как данные будут организованы и взаимосвязаны внутри системы. Она служит важным инструментом для документирования архитектуры системы и является основой для дальнейшей реализации. Важно отметить, что диаграмма классов не только описывает текущую структуру данных, но и помогает выявить возможные улучшения и оптимизации.

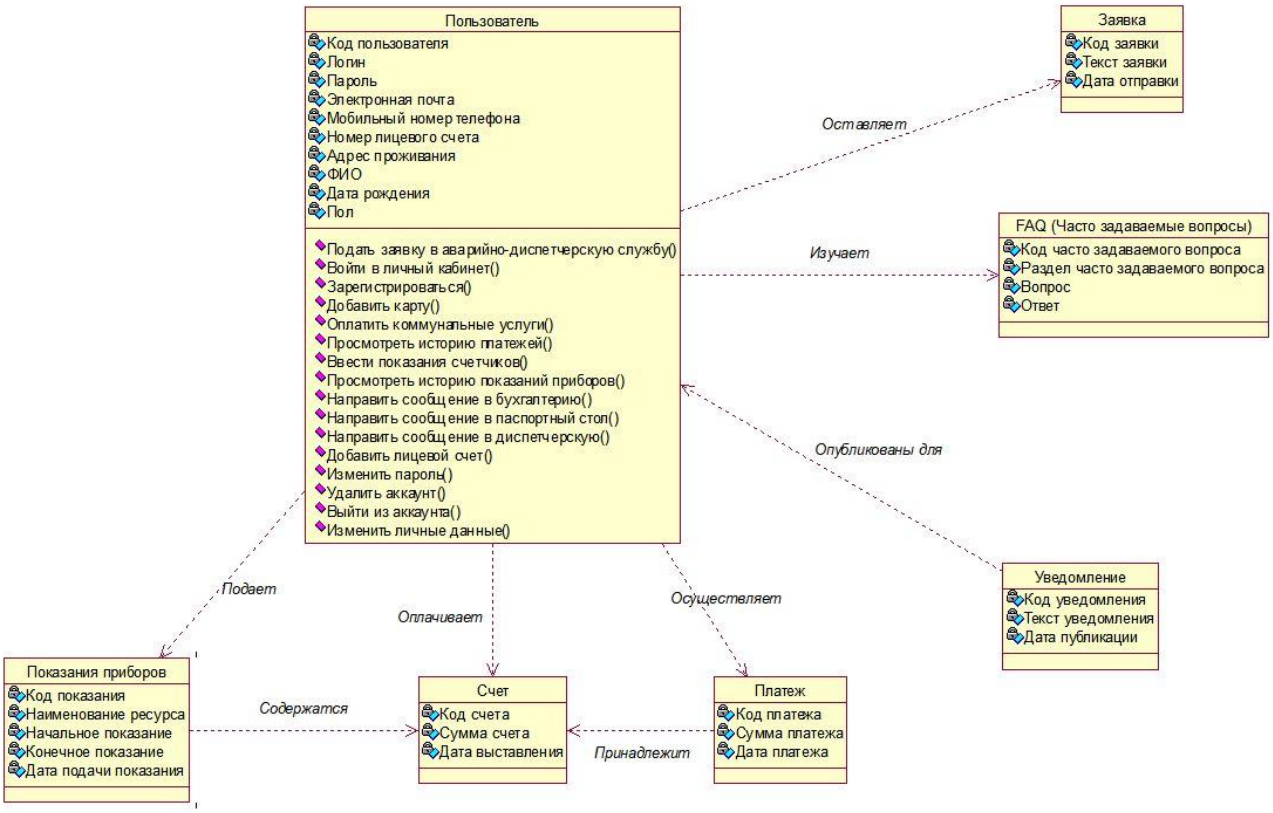

Рисунок 9 – Диаграмма классов модели личного кабинета потребителя коммунальных услуг

Диаграмма классов — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперацией, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования [2].

Диаграмма классов модели личного кабинета потребителя коммунальных услуг представлена на рисунке 9. Она включает классы, такие как «Пользователь», «Профиль», «Счетчик», «Платеж», «Заявка», и описывает их атрибуты и методы. Это помогает создать полную картину структуры данных системы и понять, как различные компоненты будут взаимодействовать друг с другом.

2.2 Разработка логической и физической модели данных для личного кабинета потребителя коммунальных услуг

Логическая модель данных представляет собой структуру данных, которая определяет сущности, их атрибуты и связи между ним без привязки к конкретной реализации. Она является расширением концептуальной модели. С помощью логической модели данных бизнес-аналитики и архитекторы данных могут визуализировать операционные или транзакционные процессы на диаграмме взаимоотношений между сущностями. Логические модели данных определяют, как работают и взаимодействуют объекты данных. Каждая таблица описывает объекты данных и их атрибуты в знакомых для бизнеса терминах. Для всех объектов назначается первичный ключ (PK) для идентификации атрибутов в каждой строке. Некоторые объекты содержат внешние ключи (FK), которые описывают их связь с другими объектами в отношениях «один ко многим» [3].

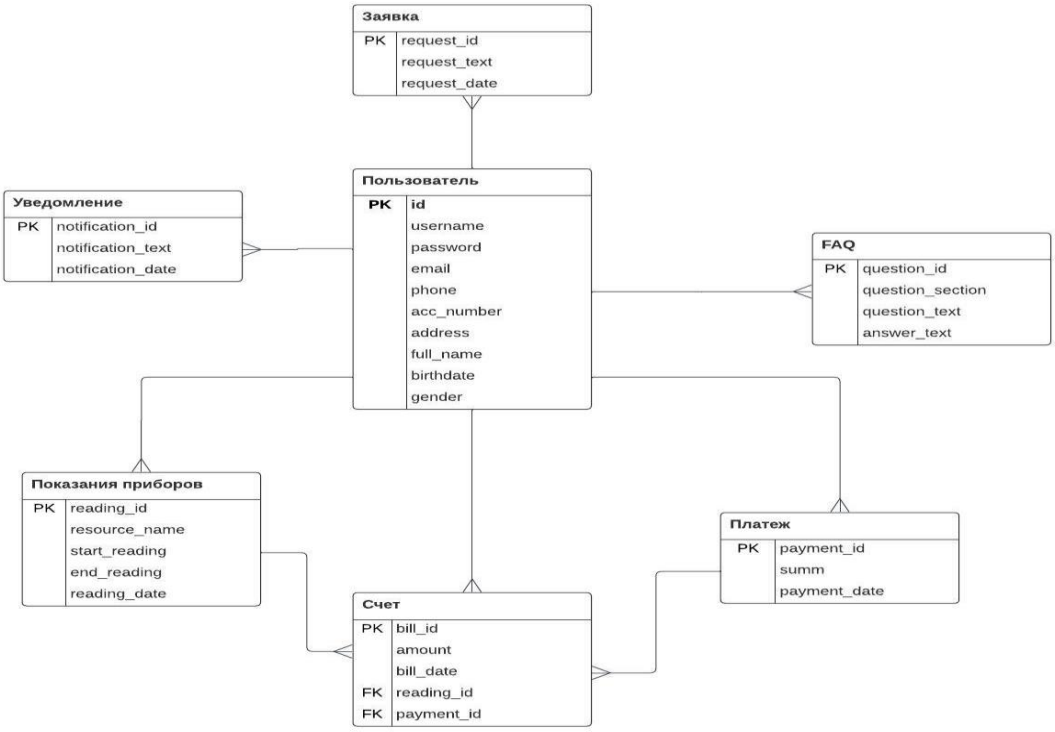

Рисунок 10 – Логическая модель данных веб-приложения

Физическая модель данных демонстрирует как устроены общая структура базы данных, взаимосвязи её объектов, таких как таблицы, индексы, переменные, первичные и внешние ключи. Физическая модель помогает и представляют информацию, которая будет лежать в основе базы данных.

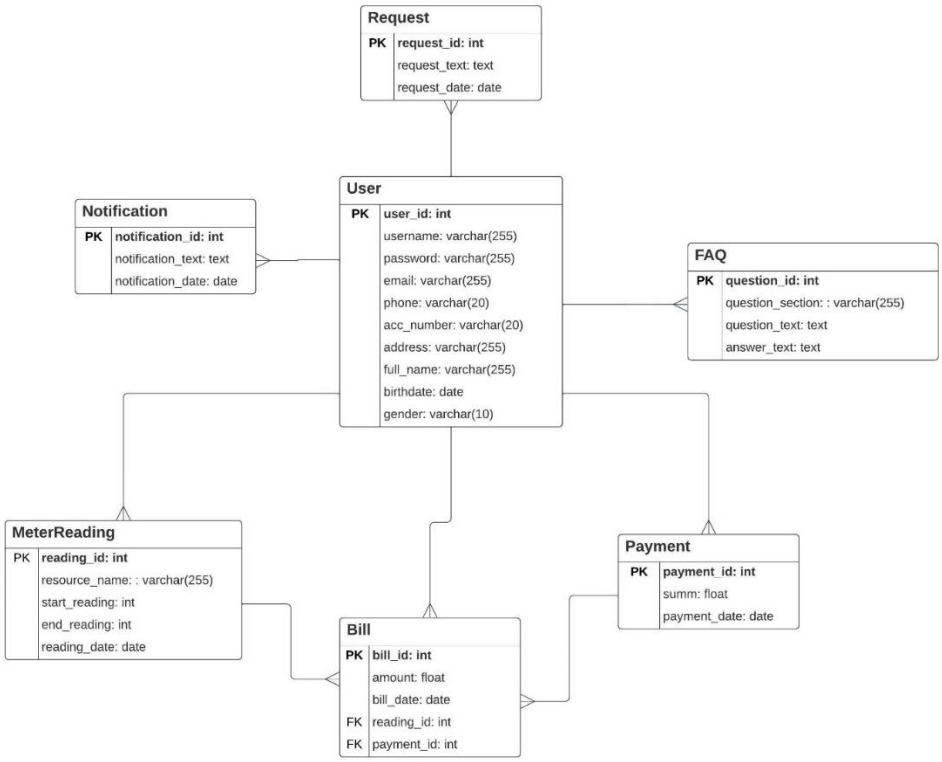

Рисунок 11 – Физическая модель данных веб-приложения

Данные модели хранят дополнительные значения, которые отсутствовали в логической модели данных, например типы данных. Другими словами, без физического представления базы данных невозможно грамотно разработать и реализовать веб-приложение. Рисунок 11 демонстрирует физическую модель данных веб-приложения личного кабинета.

Выводы по главе 2

Во второй главе подробно описано логическое проектирование веб-приложения. На этом этапе были построены логические и физические модели данных, диаграммы вариантов использования и диаграммы классов. Логические модели помогли визуализировать структуру данных и взаимодействие между различными компонентами системы. Были разработаны основные сценарии использования, которые описывают взаимодействие пользователей с системой, такие как регистрация и авторизация, внесение показаний счетчиков и управление оплатами.

Диаграммы классов и последовательностей позволили структурировать данные и определить основные компоненты системы. Было проведено детальное проектирование архитектуры приложения, включая моделирование бизнес-процессов и взаимодействие между различными модулями системы. Это позволило создать структурированную и хорошо спланированную архитектуру приложения, обеспечивающую его стабильную и эффективную работу. Основные выводы второй главы заключаются в успешной разработке архитектуры системы и создании логических моделей, обеспечивающих корректное функционирование всех компонентов приложения.

Глава 3 Разработка веб-приложения «Личный кабинет потребителя»

3.1 Выбор технологий и программных средств для разработки

Для разработки современных веб-приложений необходимо установить среду разработки (IDE). IDE находятся в «высшей категории» текстовых редакторов и обладают большим количеством плюсов. Все инструменты, которые могут пригодится разработчику, либо встроены в пользовательский интерфейс, либо есть возможность установить плагины, сделанные другими разработчиками. В средах разработки присутствует функционал запуска и отладки программы прямо из редактора, что позволяет сильно экономить время на поисках ошибок. Для разработки в составе команды в IDE есть интегрированная система «GIT», которая помогает разработчикам вносить свои правки и синхронизироваться с коллегами.

Для разработки веб-приложения была выбрана среда разработки PyCharm Community Edition. Выбор был сделан на основе сравнительного анализа с другими средами разработки. Для сравнения были вынесены такие критерии, как: удобство использования, функциональность, поддержка языков и модель распространения. Среди разнообразных IDE были выбраны 4 наиболее популярные и эффективные: NetBeans, JetBrains IntelliJ IDEA, Visual Studio и PyCharm.

Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том 34 числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight [18].

PyCharm – интегрированная среда разработки (IDE), используемая для программирования на Python. Она обеспечивает анализ кода, графический отладчик, встроенный модульный тестер, интеграцию с системами контроля версий и поддерживает веб-разработку с помощью Django. PyCharm кроссплатформенный, работает на Windows, macOS, Linux [15].

JetBrains IntelliJ IDEA — это интегрированная среда разработки, разработанная компанией JetBrains для работы с различными языками программирования, такими как Java, Kotlin, Groovy, Scala, JavaScript, TypeScript. IntelliJ IDEA включает в себя интеллектуальные инструменты кодирования (автодополнение кода, рефакторинг), интеграцию с системами контроля версий, такими как Git, Subversion, Mercurial [17].

NetBeans IDE — свободная интегрированная среда разработки приложений, на языках программирования Java, Python, PHP, JavaScript, C, C++. Проект NetBeans IDE поддерживается и спонсируется компанией Oracle. Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода. В версии NetBeans IDE 6.1 декларируется поддержка UML, SOA, языка программирования Ruby, а также средства для создания приложений на J2ME для мобильных телефонов [19].

Таблица 1 – Анализ интегрированных сред разработки

Критерии	Visual Studio	PyCharm	IntelliJ IDEA	NetBeans
Удобство использования	-	+	-	-
Функциональность	+	+	+	+
Поддержка языков	+	+	-	+
Модель распространения	+	+	+	+
Итог	3/4	4/4	2/4	3/4

По итогам сравнительного анализа выбранных сред разработок был сделан вывод, что PyCharm обладает наиболее полноценным функционалом,

удобен и легок для использования, является официальной IDE поддерживающий Django и Python, обладает бесплатной моделью распространения.

Следует подробно описать средства для реализации пользовательского интерфейса веб-приложения. Для реализации клиентской части были использованы язык разметки HTML и таблицы стилей CSS.

HTML (Hyper Text Markup of Language) – язык гипертекстовой разметки. Язык разметки, определяющий как выглядит структура веб-страницы. Он является универсальным языком во всей глобальной сети интернет и используется в основе любого веб-приложения. Благодаря нему мы можем делится информацией, изменять её в интернете, вставлять в текст различные элементы, создавать формы для взаимодействия с различными службами (поисковыми запросами и другими). Основным элементом языка HTML является тег, именно он является своеобразным кирпичом для построения страниц в веб-браузере [1].

CSS (Cascading Style Sheets) – это язык таблицы стилей. Он работает в связке с HTML. CSS определяет стиль всех элементов и описывает внешний вид страницы сайта. Данный язык добавляет на страницы: цвет, анимацию, позволяет менять и структурировать расположение элементов на странице и многое другое [8].

Для серверной части Личного кабинета пользователя был выбран язык программирования Python. Выбор был сделан на основе опыта использования разных языков программирования в процессе обучения в университете. Python, по сравнению с другими языками, обладает рядом сильных преимуществ:

– код на этом языке является не перегруженным и обладает читабельным видом, что сильно упрощает кодинг и дальнейшую поддержку;

– python обладает кроссплатформенностью, что означает, что код написанный на нем может быть запущен на различных операционных системах;

– у python имеется большое активное сообщество разработчиков, и оно обеспечивает поддержку языка, учебные материалы, документацию.

Далее был использован фреймворк Django. Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования. Проект поддерживается организацией Django Software Foundation. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY. Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных [14].

Django является наиболее удобным и универсальным фреймворком для реализации веб приложений любого масштаба. С помощью Django были написаны такие популярные сайты как YouTube, Reddit и DropBox. Отличительными чертами этого фреймворка можно назвать: легкая работа с базой данных, где хранится информация о пользователях; возможность обрабатывать запросы от пользователей и отправлять им нужную информацию; административный интерфейс, который позволяет править веб-приложение без необходимости лезть в код.

3.2 Описание функциональности и реализация проекта

3.2.1 Пользовательский интерфейс и функционал личного кабинета

Далее рассмотрим разработку веб-приложения с точки пользовательского интерфейса. Он должен соответствовать современным стандартам дизайна, быть минималистичным и иметь выдержанную

цветовую гамму. В качестве программы для создания макета сайта использовалась Figma [12].

В первую очередь, необходимо было реализовать страницы Авторизации и Регистрации пользователей. На базовом уровне они представляют собой две интерактивных формы, куда пользователь вводит свой номер телефона и придуманный им пароль. Однако, должен быть добавлен функционал, который позволяет клиенту войти через почту, если ему так удобнее и возможность сбросить пароль. На обеих страницах входа и регистрации имеется общая «шапка» сайта, где расположен логотип компании и ссылки «О компании» и «Связаться с нами», которые переадресуют на страницы с описанием компании и контактными данными соответственно. При регистрации должны выполняться требования к паролю, чтобы обезопасить аккаунт клиента на базовом уровне.

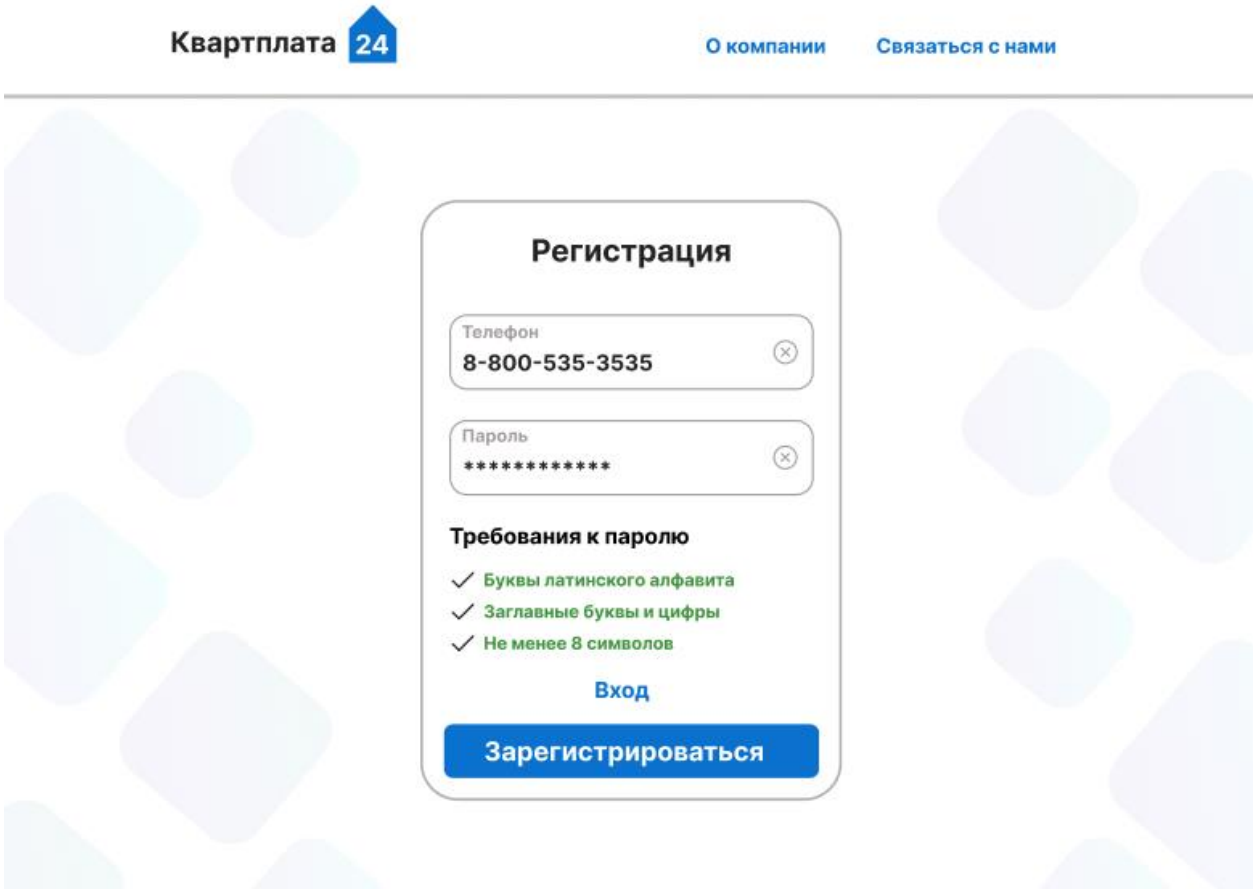

Рисунок 12 - Основной контент страницы «Регистрация»

Дизайн страниц регистрации и авторизации сочетают в функциональность и внешнюю привлекательность, которая достигается благодаря приятной цветовой гамме и читаемому шрифту. Интерфейс сделан относительно простым и понятным, чтобы пользователи легко находили нужную им информацию. Отдельное внимание было уделено формам, кнопкам и интерактивным элементам.

После успешной регистрации пользователь попадает уже в личный кабинет. Первое отличие от страницы со входом и регистрацией заключается в измененной «шапке» сайта. В левой части также располагается кликабельный логотип компании, но теперь хедер представляет собой навигационное меню с шестью пунктами: Оплата, Показания, Заявки, Помощь, Уведомления и Профиль. Каждый пункт является ссылкой и ведет на соответствующую страницу в личном кабинете. Уведомления и Профиль отличаются от остальных пунктов меню тем, что они сделаны с помощью svg-картинок. Это добавляет интерфейсу современности и интерактивности.

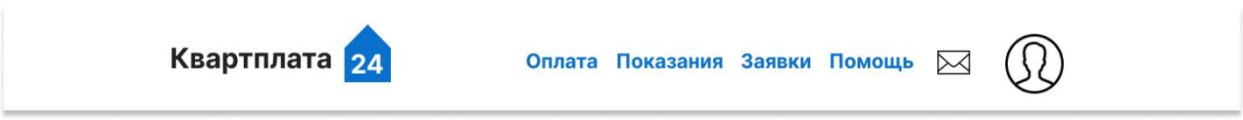

Рисунок 13 - Header всех страниц внутри личного кабинета

Первой страницей, куда перенаправляет веб-приложение пользователя после входа является его профиль аккаунта. В первую очередь пользователь сгруппированный текст, который состоит из названия его обслуживающей компании, номера лицевого счета и адреса. Если же пользователь еще не добавил эту информацию, то на этом месте будет указано, что необходимо это сделать. Соответствующая кнопка размещена в нижней части страницы. При клике на нее появится окно с формами для заполнения. Необходимо выбрать поставщика, ввести номер лицевого счета, номер счетчика или сумму поля к оплате из последней квитанции. Также есть возможность дать

пользовательское название этому счету по усмотрению клиента. Далее на странице идет подгруппа «Личные данные». Тут пользователю необходимо будет указать свои фамилию, имя и отчество, дату рождения и пол. Эта информация может быть полезна компании для отслеживания и анализа своего среднего клиента. Напротив надписи «Личные данные» будет реализована кнопка изменения данных, если такая потребность возникнет у пользователя. Точно такая же подгруппа с таким же функционалом расположена ниже. В «Контактных данных» пользователя попросят ввести свой номер телефона или электронную почту. Это необходимо для связи с пользователем в случае каких-то проблем или для уведомления о капитальном ремонте, отключении воды и т.п. Ниже на странице расположен блок с включением/отключением уведомлений. Если клиенту необходимо в срочном порядке узнавать информацию, он может воспользоваться этой функцией.

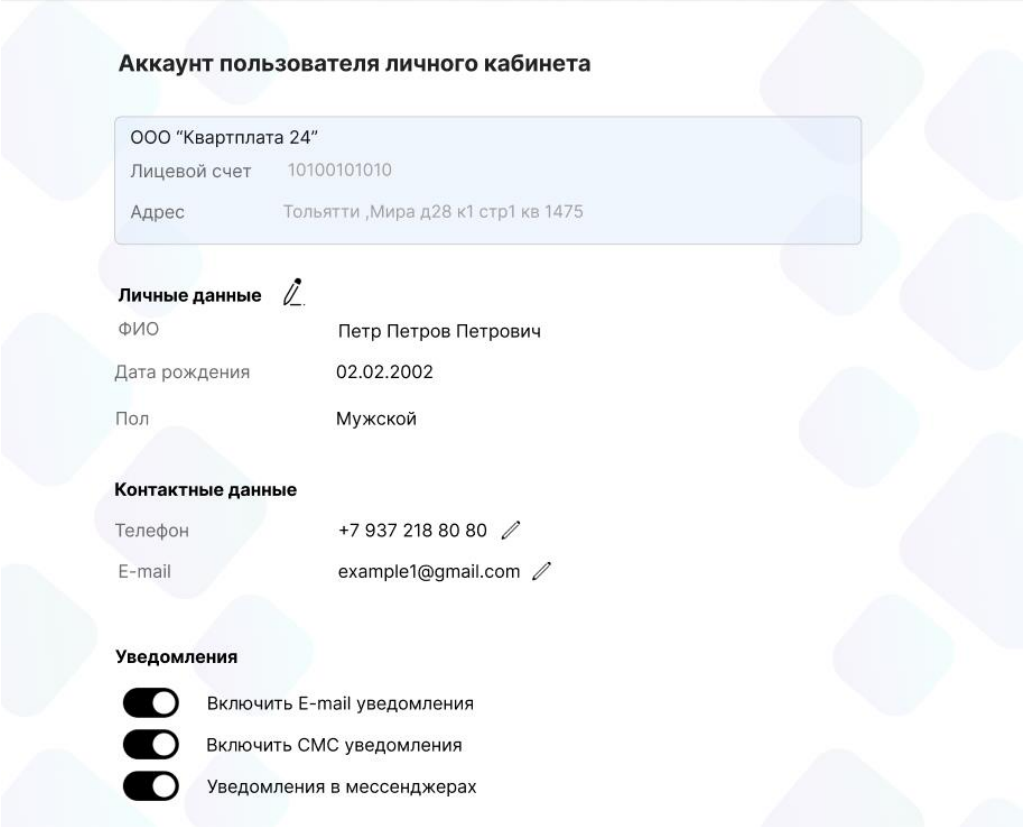

Рисунок 14 - Верхняя часть страницы Профиль

Далее идет кнопка включения двухфакторной аутентификации. Если пользователь попытается зайти в аккаунт с нового браузера или компьютера, ему придет пятизначный код на почту или на телефон. Последняя подгруппа на странице состоит из трёх кнопок, отвечающих за глобальное управление аккаунтом. При активации кнопка «изменить пароль» перенаправляет пользователя на страницу, где ему нужно будет ввести свой старый пароль, новый пароль и его подтверждение. Кнопка удалить аккаунт предусматривает полное удаление учетной записи пользователя без возможности дальнейшего восстановления. Кнопка «Выйти» отвечает за выход из аккаунта пользователя и переадресует на страницу входа.

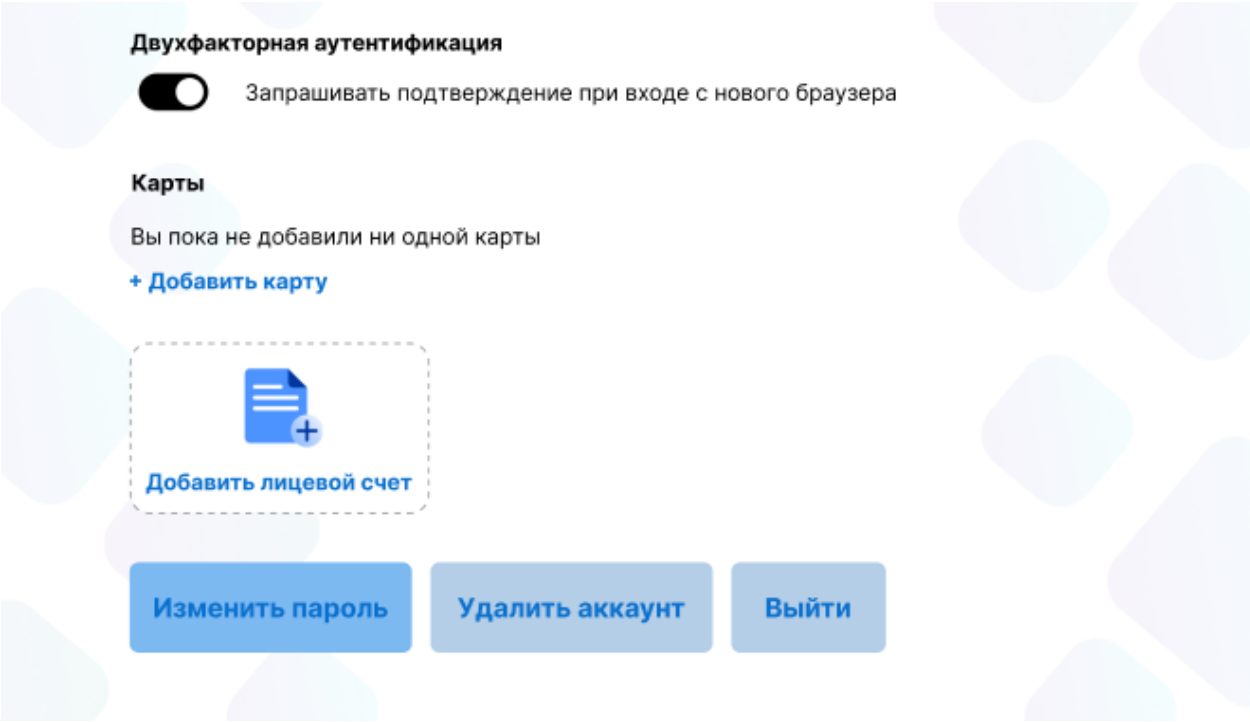

Рисунок 15 - Нижняя часть страницы Профиль

Следующая страница веб-приложения называется «Оплата». Тут клиент оплачивает свои жилищно-коммунальные услуги. Страница состоит из двух блоков: карточки с информацией и оплатой. В карточке описываются важные характеристики, такие как город, номер лицевого счета, адрес клиента. При переходе по ссылке «Детализация» пользователь увидит какая

часть суммы за что была начислена. В верхней части страницы расположена ссылка «История платежей». Она отвечает за переадресацию пользователя на страницу с его прошлым платежами, где он может детально просмотреть, когда и сколько было потрачено на конкретные коммунальные услуги. У клиента может быть несколько жилых помещений и, соответственно, несколько лицевых счетов. Тогда вместо одной карточки с информацией на странице их будет несколько и блок с оплатой будет автоматически складывать все имеющиеся задолженности.

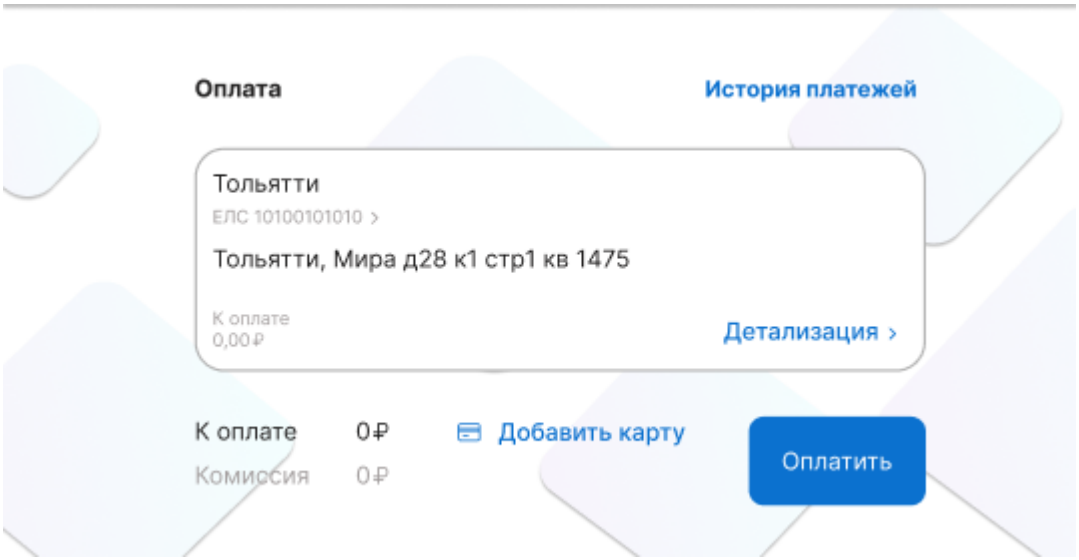

Рисунок 16 - Страница «Оплата» личного кабинета

После оплаты в навигационном меню следует страница «Показания». На этой странице жилищно-коммунальные услуги конкретного помещения группируются в единую карточку, на которой указана основная информация, иконки с используемыми услугами и ссылка «Передать показания», которая ведет на страницу с формами, куда клиент вводит текущие показания счетчиков. Услуги подразделяются на: водоснабжение (холодное, горячее), теплоснабжение и электроэнергия. У каждой услуги указан номер, её характер и прошлое значение счетчика в соответствующей единице

измерения. После ввода пользователем текущих показаний, он нажимает кнопку «Передать показания» и новые данные записываются в базу данных.

Рисунок 17 - Блок с вводом показаний счетчиков

Следующая страница «Заявки» отвечает за создание и отслеживание пользователем обращений в аварийно-диспетчерскую службу. По умолчанию на странице отображается надпись, что текущих заявок нету и кнопка «Создать заявку». При клике по этой кнопке появляется всплывающее окно с тремя формами: выбор типа заявки, тема проблемы и полное описание. После того, как заявка была создана, она появится в списке активных на странице с её текущим статусом. Как на заявку отреагируют ответственные органы, появится ответное сообщение и соответствующий статус.

Страница «Помощь» предназначена для консультирования пользователей с платформой и получения информации по различным темам. Страница разделена на две части, в левой части находятся ответы на часто задаваемые вопросы, сгруппированные по темам. Чтобы не было большего количества текста на странице, по умолчанию отображаются только сами вопросы. Для прочтения ответа клиенту необходимо кликнуть по соответствующей кнопке или просто нажать на вопрос. В правой части страницы находятся блоки информации определенных государственных организаций. Каждый блок состоит из названия, адреса организации, номера телефона, электронной почты, графика работы и кнопки «Направить сообщение».

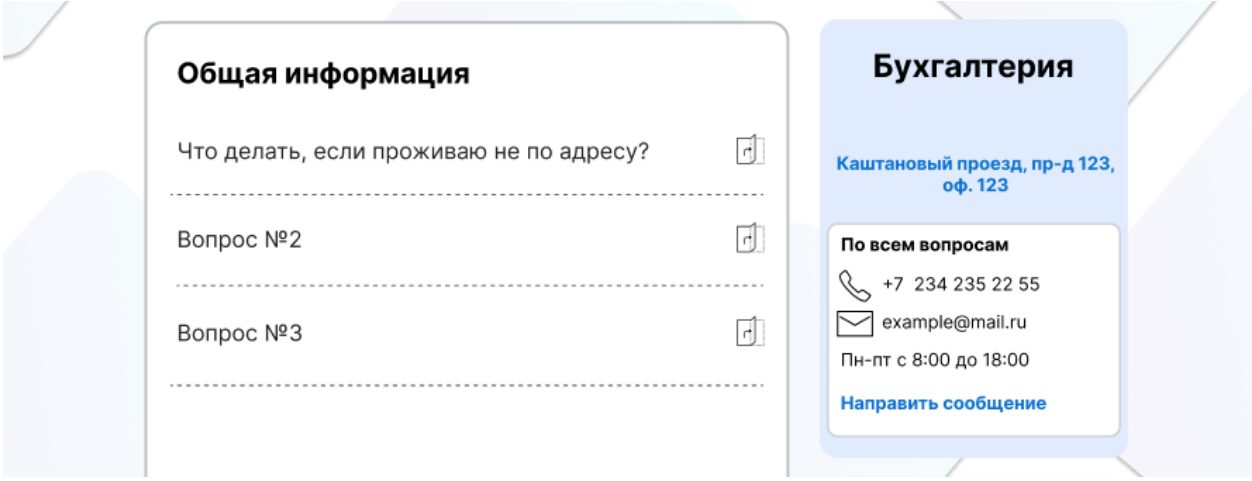

Рисунок 18 - Страница «Помощь» в личном кабинете

Последней из реализованных страниц являются «Уведомления». Страница устроена просто: каждое пришедшее клиенту сообщение отображается в сокращенном виде. По умолчанию видно только название и часть текста. Если пользователь хочет просмотреть сообщение полностью, рядом с каждым блоком сообщения есть кнопка «Детализация», которая раскрывает сообщение полностью.

3.2.2 Реализация backend части веб-приложения

Целью этой части выпускной квалификационной работы было создание системы, которая позволила бы пользователям регистрироваться, входить в личный кабинет, управлять свои данными в разделе профиля, вносить и отслеживать показания счетчиков, создавать счета для оплаты услуг и отправлять заявки в аварийно-диспетчерскую службу.

Структурно проект состоит из нескольких приложений, каждое из которых отвечает за отдельный функциональный блок. В целом, философия разработки серверной части на фреймворке Django следующая: каждая логически не связанная с другими частями веб-приложения страница (группа страниц) должна представлять собой отдельное приложение. В правильно настроенном проекте у разработчиков должна быть возможность перенести

или скопировать какое-то отдельное приложение в другой проект, и оно должно интегрироваться практически без дополнительного вмешательства со стороны. В процессе разработки данного проекта я старался придерживаться данной концепции и разделил веб-приложение на следующую структуру:

– ‘accounts’ приложение, которое отвечает за управление регистрацией и авторизацией пользователей;

– ‘profile’ приложение для управления профилем пользователя;

– ‘readings’ приложение для внесения и хранения показаний счетчиков;

– ‘payments’ приложение для управления и создания счетов для оплаты;

– ‘requests’ приложение для отправки заявок в аварийно-диспетчерскую службу;

– ‘help’ приложение с разделом часто задаваемых вопросов.

Фреймворк использует архитектурный шаблон Model-View-Template (MVT). Он является вариацией паттерна MVC и делит код приложения на данные (model), логику (view) и пользовательский интерфейс (templates) [16].

Модель отвечает за управление данными. Модели определяют структуру базы данных, включающую поля и поведение данных. Django использует объектно-реляционное отображение для создания виртуальной базы данных. Благодаря этому нету необходимости в использовании SQL-запросов, но можно взаимодействовать с помощью объектов и методов.

Представление отвечает за логику отображения данных. Работа происходит по следующему алгоритму: представления получают запросы, взаимодействуют с моделями и передают данные шаблонам для отображения [9].

Шаблоны представляют и определяют, как данные будут отображаться для конечного пользователя на веб-странице.

При создании проекта на Django всегда создается виртуальное окружение. Оно представляет собой изолированные пространство, куда можно установить специфичные для проекта библиотеки, пакеты и

зависимости Python, не влияющие на другие проекты. Это помогает избежать конфликта версий и облегчает одновременную работу над несколькими проектами [4].

Для создания виртуального окружения необходимо выполнить следующую команду: “python -m venv vkr\venv”.

Для активации виртуального окружения необходимо перейти в папку проекта и написать команду активации: “cd kvartplata24“, далее “venv\scripts\activate”.

С активированным виртуальным окружением можно установить Django. Это можно сделать с помощью ввода команды в терминале PyCharm: “pip install Django”.

После установки Django появляется возможность создать новый проект с помощью команды: “django-admin startproject kvartplata24”.

Данная команда создаст структуру папок и файлов для нового проекта. Обычная структура проекта на Django выглядит следующим образом:

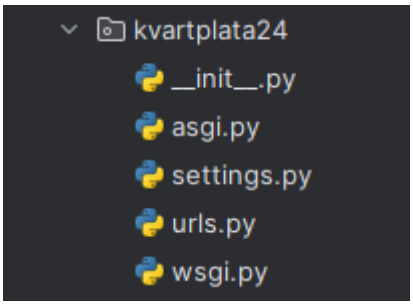

Рисунок 19 - Структура проекта Kvartplata24

Описание функциональности файлов стартовой структуры проекта:

– ‘manage.py’: командный скрипт для управления проектом. С него начинается любая команда в Django. Позволяет запускать локальный сервер, выполнять миграции и совершать другие административные задачи.;

– ‘../init.py’: пустой файл, который обозначает директорию как Python-пакет;

– ‘../settings.py’: файл конфигурации проекта. Здесь находятся настройки базы данных, установленных приложений, middleware и другие конфигурации;

– ‘../urls.py’: файл для маршрутизации URL. Определяется какие представления обрабатывают конкретные URL;

– ‘../wsgi.py’: точка входа для WSGI-совместимых веб-серверов для обслуживания проекта.

В функционал Django входит встроенный локальный сервер. При разработке сервер помогает в реальном времени отслеживать как одни изменения влияют на конкретные страницы в проекте. По умолчанию сервер запускается по адресу ‘http://127.0.0.1:8000/’, но может быть настроен с помощью дополнительных атрибутов. Для запуска встроенного сервера разработке используется команда: “python manage.py runserver”.

Для создания и изменения схемы базы данных в Django используются миграции. Они создаются автоматически для моделей, которые определяются в соответствующем файле.

Основные команды миграций:

– создание миграций: “python manage.py makemigrations”;

– применение миграций: “python manage.py migrate”;

– проверка текущего состояния миграций: “python manage.py showmigrations”.

Как упоминалось ранее, приложения в Django являются самодостаточными модулями, которые имеют собственную структуру из папок и файлов. Для создания нового приложения используется команда: “python manage.py startapp accounts” [11].

После создания приложения его нужно объявить в файле конфигурации settings.py основного приложения.

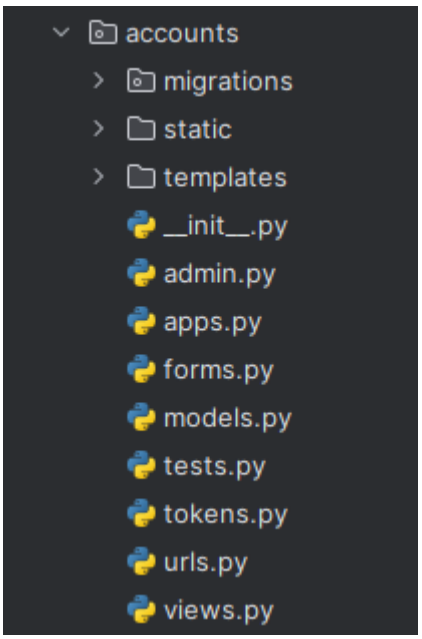

Рисунок 20 - Структура приложения ‘accounts’

Описание файлов приложения:

– ‘admin.py’: конфигурация административной панели для моделей приложения;

– ‘apps.py’: настройки приложения;

– ‘models.py’: определение моделей данных;

– ‘tests.py’: написание тестов для приложения;

– ‘views.py’: определение представлений;

– migrations/: каталог для хранения файлов миграций.

Django поддерживает различные базы данных, например MySQL, Oracle, PostgreSQL и другие [13].

В данном приложении использовалась SQLite, так как она встроена, не требует дополнительной установки и подходит по всем поставленным критериям.

Конфигурация базы данных в settings.py представлена на рисунке 21.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Рисунок 21 – Фрагмент кода settings.py

Опишем каждое приложение подробнее.

Приложение ‘accounts’ отвечает за управление регистрацией, авторизацией и восстановлением пароля пользователей. Оно обеспечивает безопасный доступ к личному кабинету и другим частям приложения, требующим аутентификации. Это приложение является фундаментальным, поскольку без него пользователи не смогут получить доступ к функциональности системы. Была использована встроенная в Django библиотека, обеспечивающая аутентификацию и управление пользователями: django.contrib.auth. Также реализована функция подтверждения электронной почты при регистрации, которая предотвращает создание фальшивых учетных записей. Предусмотрена возможность восстановления пароля, позволяющая пользователям восстанавливать доступ. Таким образом, приложение ‘accounts’ не только обеспечивает аутентификацию и управление пользователями, но и повышает общую безопасность системы.

В интерфейсе приложения предусмотрена возможность редактирования профиля пользователя, где он может обновить свою личную информацию, такие как имя, фамилия, контактные данные и предпочтения по уведомлениям.

```python
def register(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.is_active = False  # Деактивируем пользователя до подтверждения email
            user.save()
            logger.debug('User created and saved, but not active')
            send_activation_email(user, request)
            return render(request, template_name: 'accounts/confirmation_sent.html')
        else:
            logger.debug( msg: 'Form is invalid: %s', *args: form.errors)
    else:
        form = CustomUserCreationForm()
    return render(request, template_name: 'accounts/register.html', context: {'form': form})
```

Рисунок 22 – Представление регистрации accounts/views.py

Приложение profile предназначено для управления профилем пользователя. Оно позволяет пользователям просматривать и редактировать свои личные данные, изменять пароль и управлять уведомлениями. Это приложение обеспечивает пользователям удобный интерфейс для настройки их учетной записи и персонализации опыта работы с системой. Были использованы декораторы для ограничения доступа к представлениям только авторизованным пользователям: django.contrib.auth.decorators.

Приложение readings отвечает за внесение и хранение показаний счетчиков. Оно позволяет пользователям создавать группы счетчиков, вносить текущие показания и просматривать историю показаний. Это приложение обеспечивает прозрачный и удобный способ для пользователей следить за потреблением коммунальных услуг. Для пользователей предусмотрена возможность создания различных типов счетчиков, включая счетчики для воды, электричества и тепла, с возможностью указания текущих показаний в соответствующих единицах измерения. Информация о показаниях автоматически сохраняется и отображается в виде истории, что позволяет пользователям легко отслеживать динамику потребления и контролировать расходы.

```python
class Meter(models.Model):
    METER_TYPES = [
        ('water', 'Водоснабжение'),
        ('heating', 'Теплоснабжение'),
        ('gas', 'Газоснабжение'),
        ('electricity', 'Электроэнергия'),
    ]
    WATER_SUBTYPES = [
        ('cold', 'ХВС'),
        ('hot', 'ГВС'),
    ]
    ELECTRICITY_SUBTYPES = [
        ('day', 'День'),
        ('night', 'Ночь'),
    ]
    meter_group = models.ForeignKey(MeterGroup, related_name='meters', on_delete=models.CASCADE)
    meter_type = models.CharField(max_length=20, choices=METER_TYPES)
    subtype = models.CharField(max_length=20, blank=True, null=True)
    meter_number = models.CharField(max_length=20)
    initial_reading = models.DecimalField(max_digits=10, decimal_places=3)

    def __str__(self):
        return f'{self.get_meter_type_display()} - {self.meter_number}'
```

Рисунок 23 - Модели данных readings/models.py

Приложение payments предназначено для управления оплатами за коммунальные услуги. Оно позволяет пользователям видеть сумму к оплате и детализацию расходов. Тарифы на услуги настраиваются через админку Django. Это приложение обеспечивает прозрачность и контроль над оплатами коммунальных услуг. django.core.exceptions: используется для обработки исключений, таких как отсутствие данных.

Приложение requests отвечает за отправку заявок в аварийно-диспетчерскую службу. Оно позволяет пользователям создавать заявки на ремонт или обслуживание и отслеживать их статус. Это приложение обеспечивает пользователям удобный способ взаимодействия с аварийно-диспетчерской службой. Пользователи могут получать уведомления о статусе своих заявок, что позволяет им быть в курсе всех обновлений и действий. Кроме того, администраторы имеют возможность управлять заявками через административный интерфейс, изменяя их статус и добавляя комментарии.

3.3 Тестирование веб-приложения

Тестирование веб-приложения "Личный кабинет потребителя коммунальных услуг" включает в себя проверку его функциональности, производительности, безопасности и соответствия ожиданиям пользователей. Тестирование данного приложения было выполнено методом «Черного ящика».

Тестирование чёрного ящика или поведенческое тестирование — стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием чёрного ящика занимаются тестировщики, не имеющие доступ к исходному коду приложения. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций [7].

Достоинства метода «черный ящик»:

– тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика»;

– «Черный ящик» позволяет быстро выявить ошибки в функциональных спецификациях (в них описаны не только входные значения, но и то, что мы должны в итоге получить);

– тестировщику не нужна дополнительная квалификация;

– тестирование проходит «с позиции пользователя»;

– составлять тест-кейсы можно сразу после подготовки спецификации.

Недостатки метода:

– основным недостатком метода «черного ящика» является возможность пропуска границ и переходов, которые не очевидны из спецификации, но есть в реализации кода;

– можно протестировать только небольшое количество возможных вводных (входящих) значений [10].

Таблица 2 – Реализация тестирования веб-приложения

№ этапа	Тест	Действие	Результат
1	информирующие сообщения	проверка процесса регистрации	регистрация прошла успешно, учетная запись создана
2	сообщение об успешной операции	проверка входа в кабинет	вход выполнен успешно, редирект на страницу профиля
3	восстановление пароля	проверка функции восстановления пароля	письмо с инструкциями отправлено, пароль изменен
4	изменение профиля	проверка обновления личных данных	данные профиля успешно обновлены
5	внесение показаний	проверка внесения показаний счетчиков	показания успешно внесены и сохранены в базе данных
6	управление оплатами	проверка отображения детализации сумм к оплате	суммы корректно отображены, детализация верна
7	отправка заявок	проверка создания заявки в аварийную службу	заявка успешно создана и отправлена
8	адаптивность интерфейса	проверка отображения на различных устройствах	интерфейс корректно отображается не на всех устройствах
9	ошибка на стороне клиента	проверка обработки ошибок, вызванных пользователями	ошибки корректно обработаны, уведомления отображены
10	ошибка на стороне сервера	проверка обработки ошибок сервера	ошибки сервера не выявлены

Результаты тестирования методом "черного ящика" показали, что веб-приложение "Личный кабинет потребителя коммунальных услуг" функционирует корректно. Все основные функции, такие как регистрация, авторизация, восстановление пароля, управление профилем, внесение показаний и управление оплатами, работали в соответствии с требованиями и спецификациями.

Выводы по главе 3

В третьей главе описаны процессы выбора технологий и программных средств для разработки веб-приложения. Были выбраны язык

программирования Python и фреймворк Django, что обеспечило высокую производительность и гибкость приложения. Для разработки пользовательского интерфейса использовались HTML и CSS, что позволило создать интуитивно понятный и удобный интерфейс.

На этапе реализации были разработаны основные компоненты системы, включая серверную часть и пользовательский интерфейс. Реализованы такие функции, как регистрация и авторизация пользователей, управление профилем, внесение показаний счетчиков, управление оплатами и отправка заявок в аварийную службу. Проведено тестирование методом "черного ящика", которое подтвердило корректность работы всех функциональных компонентов. Тестирование включало проверку функциональности всех основных модулей, а также проверку производительности и безопасности системы.

Результаты тестирования показали, что веб-приложение полностью соответствует заявленным требованиям и готово к использованию. Основные выводы третьей главы заключаются в успешной реализации всех компонентов системы, их тестировании и готовности к внедрению.

Заключение

Целью данной выпускной квалификационной работы была разработка приложения «Коммунальные услуги. Личный кабинет потребителя». Для достижения поставленной цели были решены следующие задачи:

– ознакомление и изучение работы компании «Квартплата 24», был проведен анализ предметной области;

– был проведен анализ веб-приложений аналогичных компаний;

– изучены и выбраны средства разработки;

– разработан пользовательский интерфейс и функционал веб-приложения;

– разработана логика взаимодействия пользователя с приложением и серверная часть.

Анализ предметной области продемонстрировал, что существующие на данный момент решения имеют недостатки. На основе этого анализа и контекстной модели «как есть» было принято решение разработать личный кабинет для пользователей. Во время выполнения работы были поставлены требования к разрабатываемому веб-приложению и на их основе спроектированы диаграмма вариантов использования и диаграмма классов. После чего были построены логические и физические модели данных.

Для разработки веб-приложения был выбран язык программирования Python и фреймворк Django, которые обеспечивают высокую производительность и масштабируемость приложения. Использованы технологии HTML и CSS для создания интерфейса. Были реализованы основные функции, включая регистрацию, авторизацию, управление профилем, внесение показаний счетчиков, управление оплатами и отправку заявок в аварийную службу. Проведено тестирование методом «черного ящика», которое показало, что все основные функции работают корректно.

Все поставленные задачи были достигнуты, веб-приложение разработано и, следовательно, цель работы достигнута.

Список используемой литературы

1. Веб-разработка [Электронный ресурс]. - URL: https://habr.com/ru/articles/357720/ (дата обращения 01.02.24)

2. Диаграмма классов [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/Диаграмма_классов (дата обращения 01.02.24)

3. Логические и физические модели данных [Электронный ресурс]. - URL: https://aws.amazon.com/ru/compare/the-difference-between-logical-and-physical-data-model/ (дата обращения 01.02.24)

4. Марк Саммерфильд. Python на практике. – М.: ДМК Пресс, 2014. – 338 с.

5. Модель «Как есть» [Электронный ресурс]. - URL: https://piter-soft.ru/knowledge/glossary/process/as-is-model.html (дата обращения 01.02.24)

6. ООО «Квартплата 24» [Электронный ресурс]. - URL: https://www.ikvp24.ru/about/ (дата обращения 01.02.24)

7. Особенности тестирования «черного ящика» [Электронный ресурс]. - URL: https://quality-lab.ru/blog/key-principles-of-black-box-testing/ (дата обращения 24.04.2024)

8. Руководство по CSS [Электронный ресурс]. - URL: https://developer.mozilla.org/ru/docs/Web/CSS/Reference (дата обращения 01.02.24)

9. Смит Дж. Django. Полный справочник. – М.: Диалектика, 2018. – 720 с.

10. Тестирование по стратегии чёрного ящика [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/Тестирование_по_стратегии_чёрного_ящика (дата обращения 22.04.2024)

11. Уэсли Дж. Чан. Python. Создание приложений. – М.: Вильямс, 2015. – 816 с.

12. Якоб Нильсен. Веб-дизайн. Книга Якоба Нильсена. – М.: Символ-Плюс, 2006. – 512 с.

13. Brown A. Django Design Patterns and Best Practices. – M.: Packt Publishing Ltd., 2015. – 384 c.

14. Django [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/Django (дата обращения 01.02.24)

15. Django Software Foundation. Django Documentation. [Электронный ресурс]. - URL: https://docs.djangoproject.com/en/3.2/ (дата обращения: 05.03.2024)

16. Hourieh A Learning Website Development with Django / A Hourieh. – Birmingham.: Packt Publishing Ltd., 2008. – 262 c.

17. IntelliJ IDEA [Электронный ресурс]. - URL: https://en.wikipedia.org/wiki/IntelliJ_IDEA (дата обращения 01.02.24)

18. Microsoft Visual Studio [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio (дата обращения 01.02.24)

19. NetBeans [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/NetBeans (дата обращения 01.02.24)

20. Walker P. Public Housing and Urban Development: A Historical Perspective – M.: Harvard University Press, 2019 – 368 c.