

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2023 г.

**Разработка веб-приложения для учета оплаты электроэнергии,
потребляемой от частного генератора**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2023.308-691.ВКР**

Научный руководитель,
ст. преподаватель кафедры СП, к.ф.-м.н.
_____ А.Б.А. Алаасам

Автор работы,
студент группы КЭ-403
_____ А. Алзари

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2023 г.

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
06.02.2023 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Алзари Ахмаду,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. Тема работы (утверждена приказом ректора от 25.04.2023 г. № 753-13/12)
Разработка веб-приложения для учета оплаты электроэнергии, потребляемой
от частного генератора.

2. Срок сдачи студентом законченной работы: 05.06.2023 г.

3. Исходные данные к работе

3.1. Freeman A. Современный учебник Pro ASP.NET MVC. – 2013. –
С. 78–121.

3.2. Phillips B. Современный учебник The Big Nerd. – 2013 – С. 47–233.

3.3. Frain B. Responsive web design with HTML5 and CSS3. – Packt Publishing
Ltd, 2012. – С. 43–62.

3.4. Bob W. SQL Server 2022 Revealed, 2022. – С. 317–350.

4. Перечень подлежащих разработке вопросов

4.1. Анализ предметной области.

4.2. Сформулировать функциональные и нефункциональные требования.

4.3. Разработать диаграмму прецедентов использования для системы.

4.4. Разработать схему базы данных.

4.5. Реализация веб-приложения.

4.6. Реализация Android-приложения.

4.7. Тестирование приложения.

5. Дата выдачи задания: 06.02.2023 г.

Научный руководитель,

ст. преподаватель кафедры СП, к.ф.-м.н.

А.Б.А. Алаасам

Задание принял к исполнению

А. Алзари

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	7
1.1. Анализ аналогичных проектов	7
1.2. Структура пользователей.	11
1.3. Технологии и инструменты	13
2. ДИЗАЙН ПРИЛОЖЕНИЯ	15
2.1. Функциональные и нефункциональные требования.....	15
2.2. Диаграмма вариантов использования	16
2.3. Дизайн базы данных	18
2.4. Дизайн веб-приложений.....	21
2.4.1. Административная часть.....	23
2.4.2. Пользовательская часть.....	26
2.5. Дизайн Android приложений	27
3. РЕАЛИЗАЦИЯ ПРОЕКТА.....	28
3.1. Реализация веб-приложения	28
3.2. Реализация Android-приложений	29
3.3. Несколько фрагментов C#-кода	30
3.4. Несколько фрагментов кода JAVA	33
4. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ.....	35
4.1. Тестирование веб-приложения.....	35
4.2. Тестирование Android-приложений.....	40
4.3. Функциональное тестирование	42
ЗАКЛЮЧЕНИЕ	46
ЛИТЕРАТУРА.....	47

ВВЕДЕНИЕ

Актуальность

Развивающиеся страны, такие как Сирия и Ирак, сталкиваются с многочисленными проблемами, которые препятствуют их росту и технологическому прогрессу. Непрерывающиеся конфликты и войны привели к упадку бизнеса и производства, а сектор электроэнергетики является одним из секторов, которые пострадали и даже были разрушены.

В этих странах государственных услуг по электроснабжению недостаточно, поэтому частные компании и даже частные лица были вынуждены вкладывать средства в генераторы, установленные в городских секторах, и устанавливать для этих секторов частные распределительные сети. Однако, несмотря на эти инвестиции, этим компаниям или частным лицам часто не хватает необходимых технологий для эффективного управления своими ресурсами и услугами. Именно здесь современные технологии могут сыграть решающую роль в улучшении технической стороны их операций и оказании им помощи в предоставлении более качественных услуг своим клиентам и сокращении человеческих ошибок.

Но не все технологии доступны, тем более что эти регионы пострадали от массовой миграции молодежи, в том числе разработчиков приложений, в результате войн и экономических проблем. Следовательно, необходимо предоставить простое в использовании программное обеспечение с открытым исходным кодом для поддержки появляющихся местных инвестиций в этих регионах.

Одной из таких технологий является онлайн-система управления выставлением счетов, которая включает в себя программное обеспечение, позволяющее передавать и контролировать счета и счета-фактуры между клиентами и поставщиками через Интернет. Онлайн-счета дополнительно сохраняются для использования в будущем, а информация, хранящаяся в облаке, остается там для использования в будущем при необходимости. Кроме

того, вся важная информация, включая повторяющиеся счета, счета-фактуры, погашение на основе подписки, а также ежемесячные экономические отчеты о доходах, могут быть получены немедленно.

Постановка задачи

Целью работы является разработка веб-приложения для учета оплаты электроэнергии, потребляемой от частного генератора.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) анализ предметной области;
- 2) сформулировать функциональные и нефункциональные требования;
- 3) разработать диаграмму прецедентов использования для системы;
- 4) разработать схему базы данных;
- 5) реализация веб-приложения;
- 6) реализация Android-приложения;
- 7) тестирование приложения.

Структура и содержание работы

Работа состоит из введения, 5 глав, заключения и списка литературы. Объем работы составляет 48 страниц, объем списка литературы 18 источников.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ аналогичных проектов

В этом разделе описан анализ проектов, механизм работы которых аналогичен механизму работы нашего проекта, посредством которого необходимо определить основные требования к работе проекта, а для пользователей будут определены плюсы и минусы этих проектов.

SAP для коммунальных услуг (SAP for Utilities) [1]

SAP – это компания, которая предоставляет комплексное программное решение, разработанное специально для коммунальной отрасли. Его программное обеспечение помогает электроэнергетическим компаниям управлять различными бизнес-процессами, включая обслуживание клиентов, управление выставлением счетов и доходами, управление активами, торговлю энергией, управление рисками, управление данными измерений, а также аналитику и отчетность.

Программное обеспечение построено на централизованной платформе, которая интегрируется с другими решениями SAP и сторонними системами, чтобы обеспечить единый источник достоверных данных для всех коммунальных данных, позволяя компаниям принимать обоснованные решения на основе данных и идей в режиме реального времени, уменьшая количество ошибок и улучшая эффективность.

SAP предоставляет ряд услуг, в том числе автоматизированное управление выставлением счетов и доходами, централизованное управление данными измерений, а также расширенные возможности аналитики и отчетности для повышения эффективности компаний и повышения уровня их обслуживания (рисунок 1).

Преимущества SAP включают [2].

1. Улучшенное обслуживание клиентов. Благодаря доступу к данным в режиме реального времени и централизованной системе

коммунальные предприятия могут быстро реагировать на запросы клиентов и более эффективно решать проблемы.

2. Повышение эффективности. Автоматизация и стандартизация процессов SAP могут помочь коммунальным службам сократить объем ручного труда и повысить точность, что приведет к повышению эффективности.

3. Лучшее принятие решений. Имея доступ к большому количеству данных и аналитическим возможностям, коммунальные предприятия могут принимать обоснованные решения на основе информации в режиме реального времени.

Увеличение доходов. Автоматизируя процессы выставления счетов и повышая точность, коммунальные службы могут увеличить доходы и сократить расходы, связанные с ошибками, допущенными вручную.

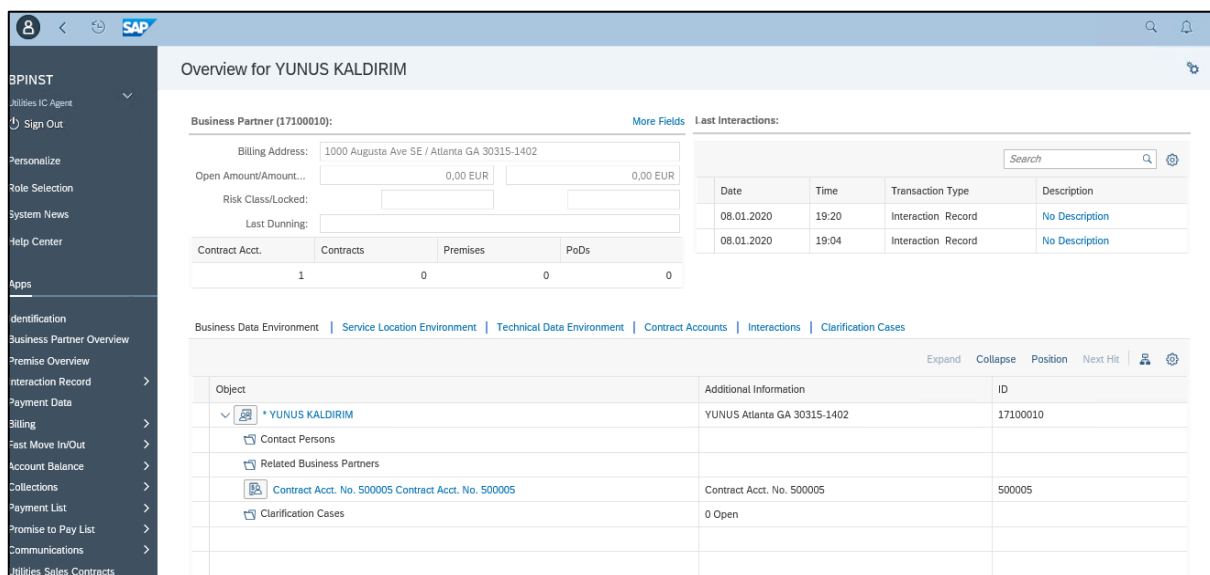


Рисунок 1– Сервис SAP

А что касается недостатков, с которыми сталкиваются пользователи SAP [2].

1. Сложность. SAP – это комплексное решение, которое может быть сложным в реализации и требует специальных знаний для управления.

2. Высокая стоимость. SAP – это решение премиум-класса, приобретение и обслуживание которого может быть дорогостоящим, особенно для малых и средних коммунальных предприятий.

3. Проблемы внедрения. Внедрение SAP for Utilities требует значительных усилий и ресурсов, включая миграцию данных и интеграцию с существующими системами.

4. Сопротивление изменениям. Некоторые сотрудники и заинтересованные стороны могут сопротивляться использованию новой системы, что приводит к сопротивлению изменениям и проблемам внедрения.

Siemens EnergyIP [3]

Siemens EnergyIP – это компания, которая предоставляет интегрированную программную платформу, предназначенную для коммунальных предприятий для управления их энергетическими операциями.

Платформа предлагает ряд возможностей, включая управление данными счетчиков, реагирование на спрос, управление распределенной энергией и управление энергетическим рынком.

Одной из ключевых особенностей EnergyIP является его способность собирать, проверять и анализировать данные из различных источников, включая интеллектуальные счетчики, датчики и другие устройства, а также предоставлять информацию коммунальным предприятиям для оптимизации их операций.

Платформа также предлагает инструменты расширенной аналитики для прогнозирования спроса на энергию, выявления потенциальных проблем и повышения общей эффективности и надежности.

Модуль реагирования на спрос EnergyIP позволяет коммунальным предприятиям стимулировать своих клиентов к сокращению потребления энергии в периоды пикового спроса, тем самым снижая затраты и повышая стабильность сети.

Модуль управления распределенным энергопотреблением позволяет коммунальным службам интегрировать и управлять распределенными энергетическими ресурсами, такими как солнечные панели и системы хранения аккумуляторов, что помогает оптимизировать использование этих ресурсов и повысить надежность и отказоустойчивость сети (рисунок 2).

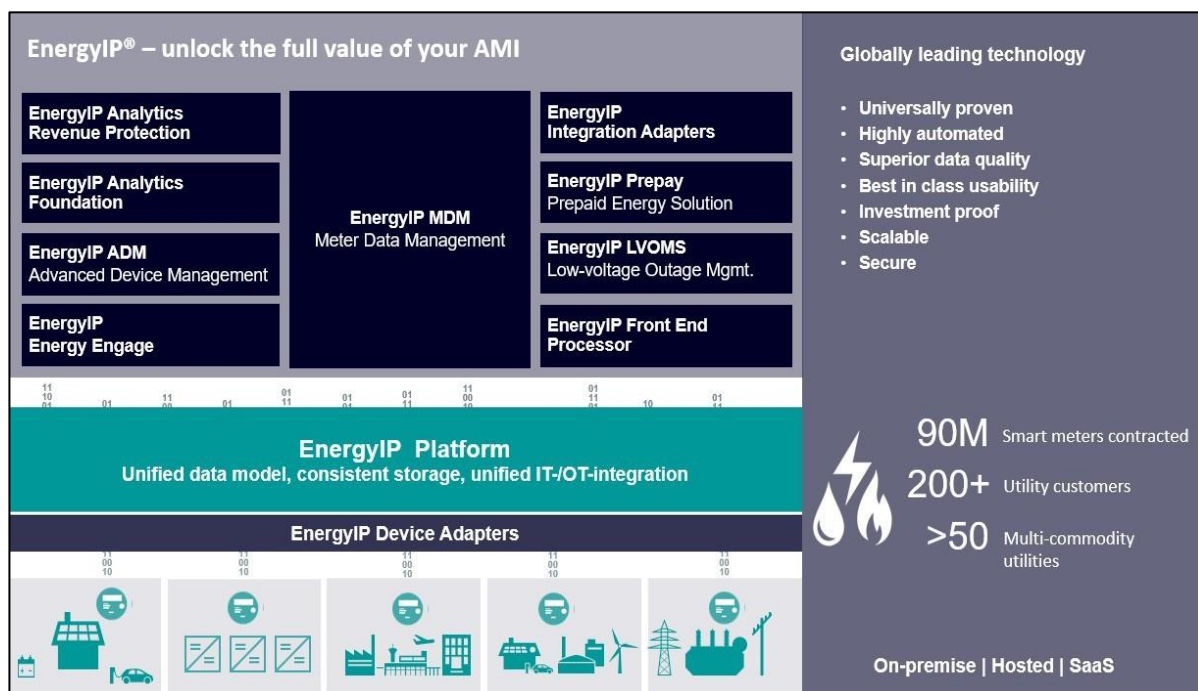


Рисунок 2 – Сервис EnergyIP

Преимущества Siemens EnergyIP [3–4].

1. Улучшенное управление энергопотреблением. Оно предоставляет коммунальным предприятиям различные возможности для управления и оптимизации их энергетических операций. Платформа предлагает инструменты расширенной аналитики для прогнозирования спроса на энергию, выявления потенциальных проблем и повышения общей эффективности и надежности.

2. Повышенная стабильность сети. Модуль реагирования на спрос позволяет коммунальным предприятиям стимулировать своих клиентов к сокращению потребления энергии в периоды пикового спроса, что снижает затраты и повышает стабильность сети.

3. Улучшенная интеграция распределенных энергетических ресурсов. Модуль управления распределенным энергопотреблением позволяет коммунальным предприятиям интегрировать и управлять распределенными энергетическими ресурсами, такими как солнечные батареи и аккумуляторные батареи, оптимизируя использование этих ресурсов и повышая надежность и отказоустойчивость сети.

4. Сбор данных и информация в режиме реального времени. Одной из ключевых функций является способность собирать, проверять и анализировать данные из различных источников в режиме реального времени, предоставляя коммунальным службам информацию для оптимизации их операций.

Недостатки Siemens EnergyIP [4].

1. Стоимость. Siemens EnergyIP может быть дорогим, что затрудняет его доступность для небольших коммунальных предприятий.

2. Сложность. Платформа может быть сложной, и для ее эффективного использования может потребоваться значительное обучение и ресурсы.

3. Проблемы интеграции. Интеграция EnergyIP с другими системами и устройствами может быть сложной задачей, особенно с устаревшими системами.

4. Безопасность. Для любой платформы, работающей с конфиденциальными данными об энергетике, безопасность является проблемой, и важно применять надлежащие меры безопасности для защиты от потенциальных киберугроз.

1.2. Структура пользователей.

В данной системе используется иерархическая структура, которая разделяет рабочий центр по уровням ранга, и это осуществляется в следующем порядке.

Структура пользователей представлена на рисунке 3.

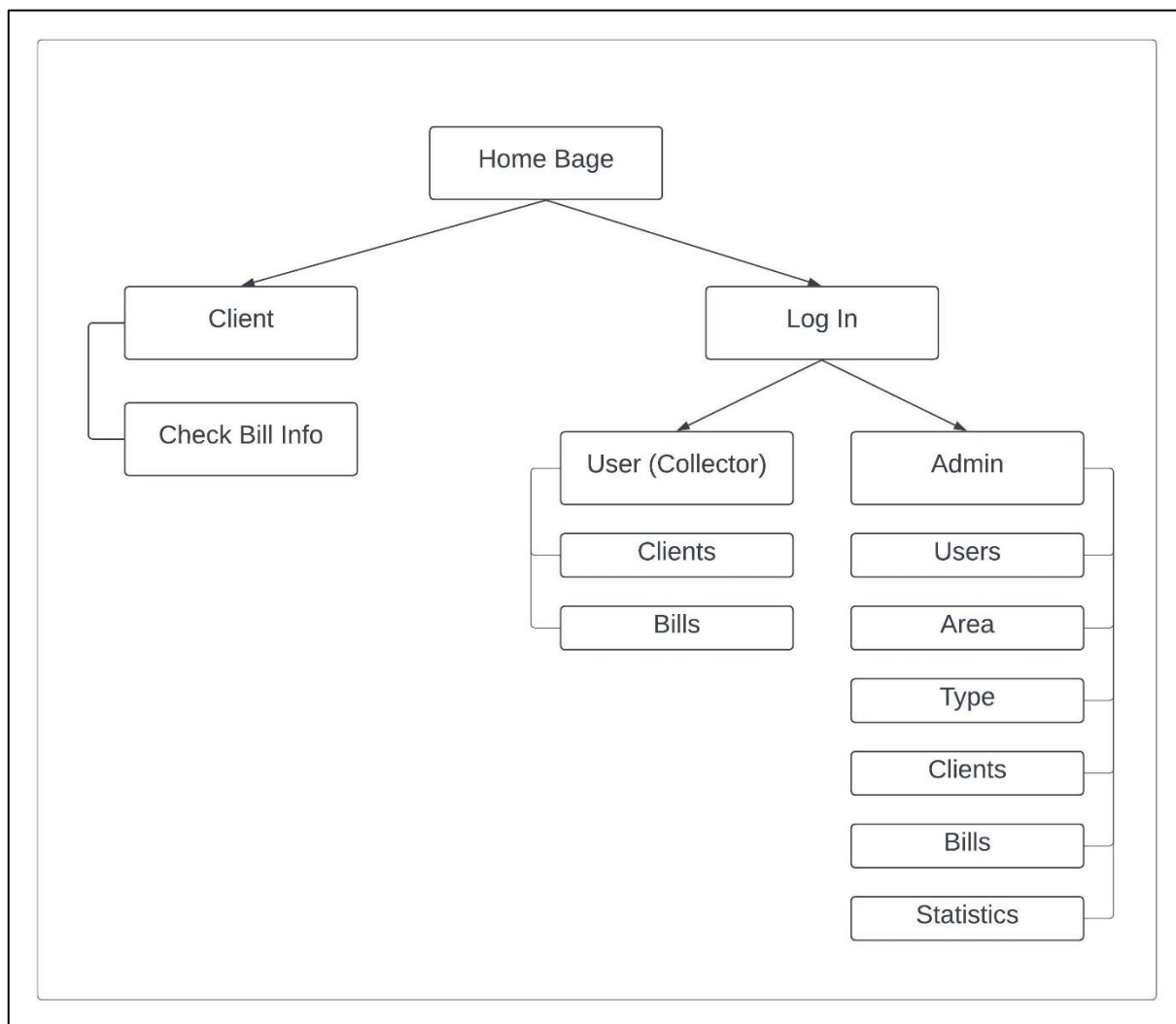


Рисунок 3 – Структура пользователей

Администратор – это человек, который несет на себе ответственность за эффективную работу системы. Он или она должны уметь управлять множеством задач, связанных с системой, а также обеспечивать безопасность и правильное функционирование системы. Кроме того, администраторы имеют доступ ко всем изменениям и дополнениям, сделанным другими пользователями.

Сборщик – это второй по важности человек в системе. Этот человек играет роль связующего звена между компанией и пользователями. Сборщик собирает платежи от пользователей и вносит коррективы в счета для

определения их облачности. Кроме того, он может помогать пользователям в решении различных вопросов и проблем.

Клиент – это человек, который пользуется услугами, предоставляемыми компанией. Клиенты занимают последнее место в данной системе, и их использование системы ограничено только доступом к своим счетам и информации о собственном потреблении. Однако, даже с таким ограниченным доступом, клиенты могут получить доступ к множеству полезной информации о своих счетах и потреблении.

Каждый из этих трех типов пользователей играет важную роль в системе, и без них система не смогла бы функционировать правильно. Имея ясное понимание каждой из ролей, компания может лучше управлять своей системой и обеспечивать более качественное обслуживание своих клиентов.

1.3. Технологии и инструменты

В данном разделе представлен анализ технологий и инструментов, выбранных для внедрения. Технологии были выбраны исходя из их функциональности и актуальности их использования в настоящее время.

Для реализации были выбраны следующие технологии.

ASP.NET MVC Framework – это фреймворк для текущего контроллера-представления-модели-контроллера-представления [5].

Платформа ASP.NET MVC основана на трех компонентах: Контроллерах, модулях и сообщениях. Контроллер управления берет на себя управление, регулирует управление жидкостью, управляет устройством и изменяет его, контролирует управление. Модель предоставляет слой и описание логики организации данных в предложение. Политическая подготовка контролирует и генерирует элементы посредством вмешательства для получения информации о вскрытии [5–6].

HTML – это стандартизированный язык разметки для документов во Всемирной паутине. Большинство веб-страниц описывают разметку в

HTML. Этот язык интерпретируется браузерами; форматированный текст, полученный в результате интерпретации, выводится на экран монитора компьютера или мобильного устройства [7].

CSS – это формальный язык описания внешнего вида документа, написанного с использованием языка разметки [7].

Microsoft SQL Server – это система управления реляционными базами данных, разработанная Microsoft. Как сервер базы данных, это программный продукт с основной функцией хранения и извлечения данных по запросу других программных приложений, которые могут работать либо на том же компьютере, либо на другом компьютере в сети [8].

Bootstrap – это бесплатный набор инструментов для создания веб-сайтов и веб-приложений. Он включает в себя шаблоны дизайна HTML и CSS для печати, веб-формы, кнопки, метки, блоки навигации и другие компоненты веб-интерфейса, включая расширения JavaScript [9].

Microsoft Visual Studio – Это интегрированная среда разработки, используемая для разработки компьютерного программного обеспечения, веб-приложений, мобильных приложений и игр. Visual Studio поддерживает несколько языков программирования, включая C++, C#, VB.NET и F#, и широко используется разработчиками программного обеспечения и организациями для создания и обслуживания программных продуктов [10].

2. ДИЗАЙН ПРИЛОЖЕНИЯ

2.1. Функциональные и нефункциональные требования

Требования к пользователям представляют собой возможность входа и выхода пользователей из системы, а функциональные требования определяют необходимый набор функций, которые должны быть реализованы в проекте для обеспечения его безошибочной работы и защиты от несанкционированного доступа.

Как было упомянуто ранее, в проекте выделяются три типа пользователей: администраторы, сборщики и клиенты. Каждый из этих типов обладает определенными привилегиями и функциональными возможностями в рамках системы.

Учет требований к пользователям и функциональных требований является важным этапом в разработке проекта, поскольку от этого зависит эффективность и надежность работы системы.

С учетом вышеуказанного, в проекте определены следующие функции для каждой из трех категорий пользователей.

1. Функции, доступные для администратора:
 - добавление или удаление сборщика или пользователя;
 - добавление или удаление региона;
 - добавление или удаление типа (типа электросчетчика);
 - добавление или удаление счета;
 - проверка статистики.
2. Функции, доступные для Сборщика:
 - добавление или удаление пользователя;
 - добавление или удаление счета;
 - проверка статистики пользователя.
3. Функции, доступные для клиента:
 - проверка своего собственного счета.

Каждый тип пользователя обладает набором соответствующих функций, которые определены с учетом их потребностей и привилегий в рамках системы. Такой подход обеспечивает эффективное и безопасное использование системы, а также удобство работы для каждой из категорий пользователей.

Что касается не функционала, то он также должен присутствовать в каждом проекте. Разделение его на три части представляется целесообразным.

1. Приложение работает для 3-х пользователей на 3-х разных компьютерах с общей базой данных.

2. В настоящее время на рынке существует множество операционных систем, каждая из которых обладает своими особенностями и программами. Упомянутое программное обеспечение было разработано для доступа к операционной системе Windows и не совместимо с другими операционными системами, такими как Linux и Mac.

3. Существует множество программ для баз данных, каждая из которых обладает своей уникальной функциональностью. Для хранения и поиска информации в настольных приложениях обычно используются специализированные программные решения. В нашем случае, в качестве такого решения была выбрана MsSQL Server, и другие программы данного типа не применяются.

2.2. Диаграмма вариантов использования

Унифицированный язык моделирования UML – это стандартизированный язык моделирования, который состоит из интегрированного набора диаграмм. Он был разработан для помощи разработчикам систем и программного обеспечения в визуализации, конструировании, определении и документировании артефактов программных систем, а также для моделирования заданий и других задач, не связанных с программным обеспечением

системы. UML представляет собой набор лучших инженерных практик, которые имеют определенный успех в моделировании больших и сложных систем [12].

Модель прецедентов использования описывает функциональные требования системы, выраженные в виде прецедентов использования [13]. Это модель предполагаемой функциональности системы (Use Cases) и ее окружения (Actors). Прецеденты использования позволяют связать то, что вам необходимо от системы, с тем, как система удовлетворяет этим потребностям.

Для составления диаграммы прецедентов использования необходимо в первую очередь определить функциональные требования к системе. В этом контексте разработчики должны четко определить, какие функции должна выполнять система. Затем следует определить взаимодействие пользователей с этими функциями.

После того, как функциональные требования определены, разработчики могут составить диаграмму прецедентов использования, которая отображает возможные сценарии использования системы.

Важно отметить, что составление диаграммы прецедентов использования является важным этапом проектирования системы. Оно позволяет разработчикам более глубоко понять требования пользователей и предусмотреть все возможные сценарии использования системы, что в свою очередь может повысить качество и надежность системы.

Данная диаграмма иллюстрирует варианты использования системы управления электроэнергетической компанией (рисунок 4). Она включает 9 вариантов использования, каждый из которых выполняет определенную функцию, а также три типа действующих лиц: администратор, сборщик, клиент. Каждое из этих действующих лиц связано с соответствующими вариантами использования посредством отношения ассоциации, что позволяет продемонстрировать возможности каждого пользователя в системе.

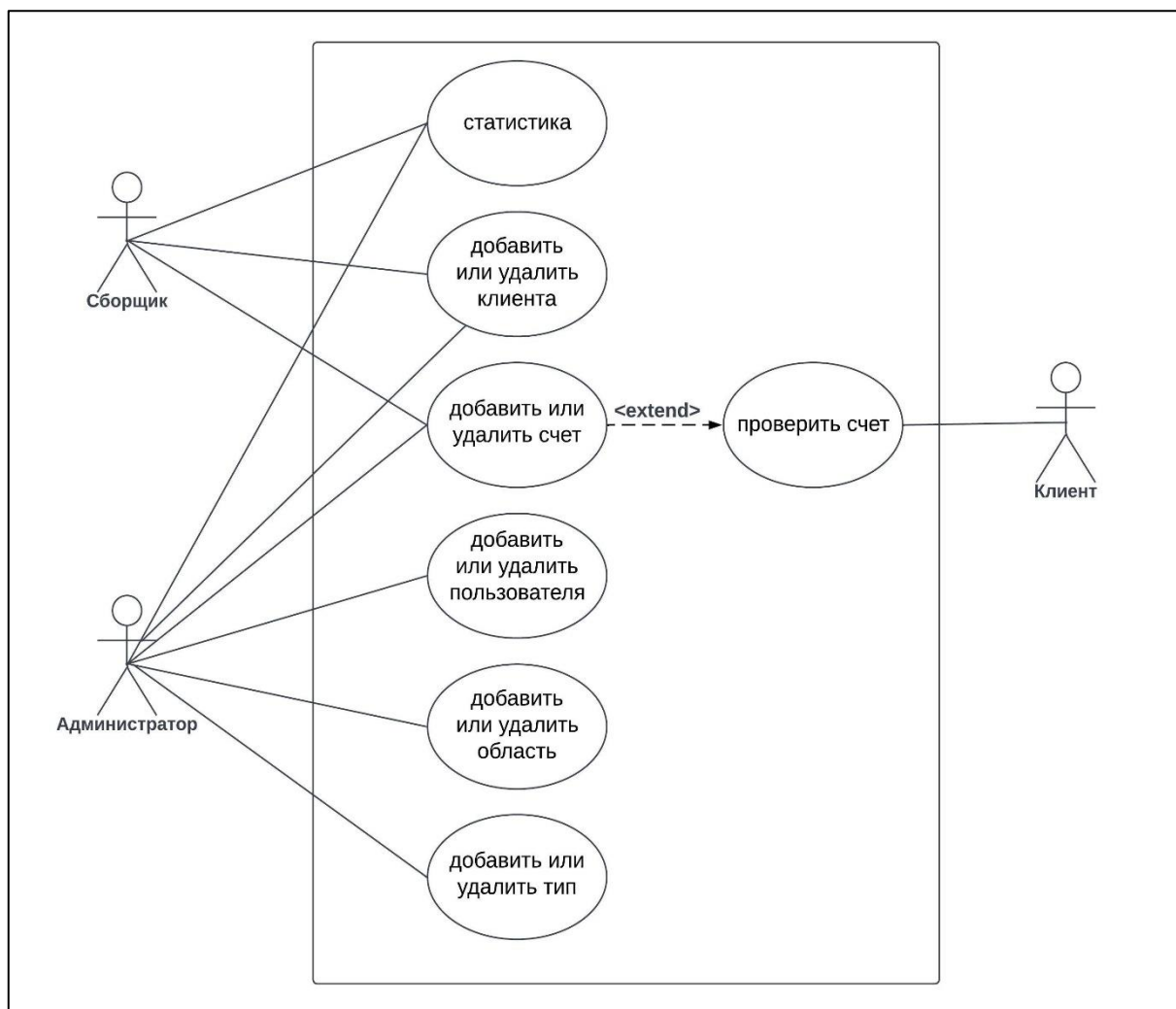


Рисунок 4 – Диаграмма вариантов использования

2.3. Дизайн базы данных

Базы данных сегодня являются неотъемлемой частью любого бизнеса. В сущности, база данных – это собрание информации, которая существует в течение продолжительного времени, часто многих лет. Используя эту информацию, можно начать адаптировать данные к модели базы данных. Проектирование базы данных включает в себя классификацию данных и определение взаимосвязей. Реляционная модель является теоретической основой реляционных баз данных, которая представляет собой технику или способ структурирования данных с использованием отношений, математических структур, состоящих из столбцов и строк, похожих на сетки.

В наши дни базы данных стали неотъемлемой частью любой организации. По сути, база данных представляет собой набор данных, которые хранятся в течение длительного периода времени, часто годами. Имея эту информацию, вы можете начать процесс ее адаптации к модели базы данных. Проектирование базы данных включает в себя классификацию данных и определение отношений. Теоретической основой для реляционных баз данных является реляционная модель, которая представляет собой метод или метод организации данных с использованием отношений, которые представляют собой математические структуры в виде сетки, состоящие из столбцов и строк. В реляционной модели все данные логически организованы в отношения, то есть таблицы.

Каждое отношение имеет имя и состоит из именованных атрибутов или столбцов данных. Каждая группа или строка содержит одно значение для каждого атрибута. Важнейшее преимущество реляционной модели заключается в ее простой логической структуре [14].

Обеспечить высокую независимость данных, чтобы прикладные программы не зависели от изменений во внутреннем представлении данных, таких как изменения в файловой организации или путях доступа.

Обеспечить серьезные основания для решения проблем семантики данных, надежности и избыточности.

Разрешить расширение языков манипулирования данными, ориентированных на наборы.

Базы данных являются общим компонентом многих программных систем, включая системы транзакционной обработки критически важных операций и многоуровневые интернет-приложения. Схема базы данных представляет собой скелетную структуру, отражающую логический вид всей базы данных [15].

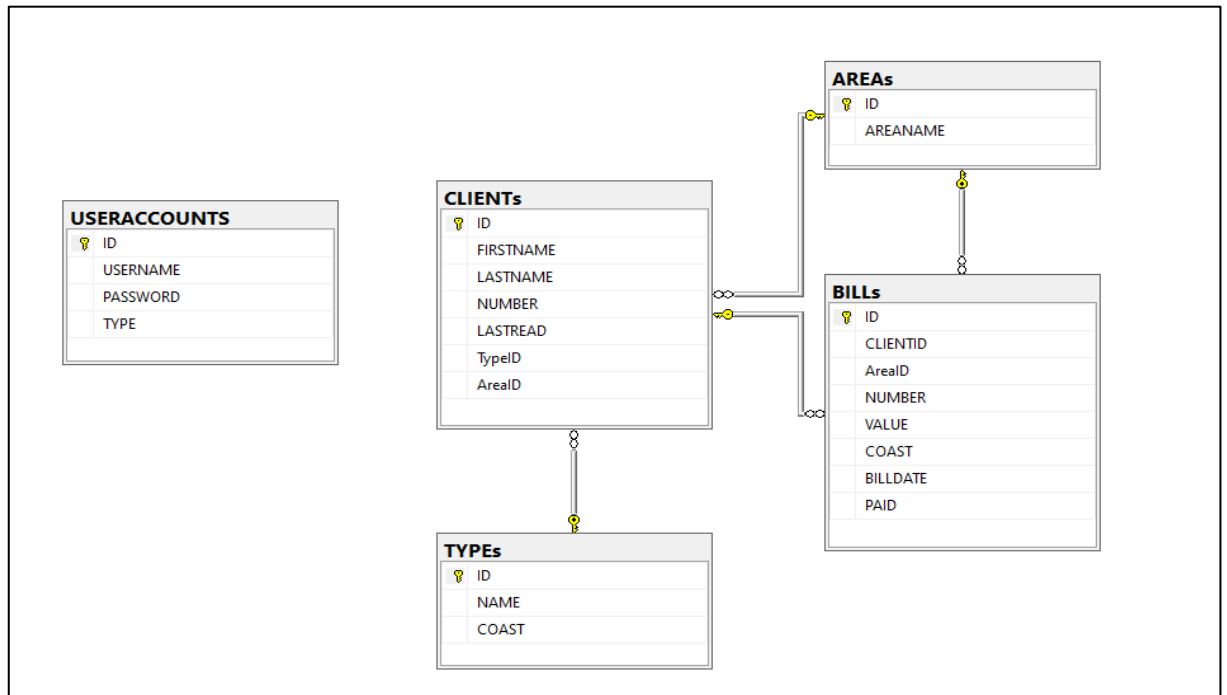


Рисунок 5 – Схема базы данных

Таблица USERACCOUNT сохраняет такую информацию, как ID, USERNAME, PASSWORD и TYPE (рисунок 6).

USERACCOUNT			
ID	USERNAME	PASSWORD	TYPE
1	User1	Pass1	admin
2	User2	Pass2	user

Рисунок 6 – USERACCOUNT

Таблица AREA сохраняет такую информацию, как ID и AREANAME (рисунок 7).

AREA	
ID	AREANAME
1	Area1
2	Area2

Рисунок 7 – AREA

Таблица TYPE сохраняет такую информацию, как ID, NAME и COAST (рисунок 8).

TYPE		
ID	Name	Coast
1	Type1	100
2	Type2	150

Рисунок 8 – TYPE

Таблица CLIENT сохраняет такую информацию, как ID, FIRSTNAME, LASTNAME, AREANAME, NUMBER, LASTREAD и TYPE (рисунок 9).

CLIENT						
ID	FIRSTNAME	LASTNAME	AREANAME	NUMBER	LASTREAD	TYPE
1	Sergey	AA	Area1	111	2022-01-01	Type1
2	Aleksandr	AB	Area2	112	2022-02-01	Type2

Рисунок 9 – CLIENT

Таблица BILL сохраняет такую информацию, как ID, CLIENTID, AREANAME, NUMBER, VALUE, COAST, BILLDATE и PAID (рисунок 10).

BILL							
ID	CLIENTID	AREANAME	NUMBER	VALUE	COAST	BILLDATE	PAID
1	1	Area1	111	500	100	2022-1-1	NO
2	2	Area2	112	750	150	2022-2-1	YES

Рисунок 10 – BILL

2.4. Дизайн веб-приложений

На рисунке 11 показан дизайн основного интерфейса веб-приложения, где у клиента есть два варианта поиска счетов. Клиент может искать

последний счет по его номеру или по полному имени (имя и фамилия). После ввода необходимой информации клиент может нажать кнопку «Показать последний счет», чтобы проверить последний счет. Кнопка входа находится в правом верхнем углу страницы.

The screenshot shows a web page titled "Home Page" in the top-left corner. In the top-right corner, there is a "Log in" link. The main content area contains three input fields: a large one labeled "Enter Your Number", a smaller one labeled "First Name", and another smaller one labeled "Last Name". Below these fields is a button labeled "Show Last Bill". The word "or" is positioned between the "Enter Your Number" field and the "First Name" and "Last Name" fields.

Рисунок 11 – Основная страница

На этом рисунке 12 показан дизайн страницы входа. На странице пользователь должен ввести свое имя пользователя и пароль, а затем нажать кнопку «Войти», чтобы получить доступ к приложению. Доступ к этому приложению ограничен администратором и сборщиком.

The screenshot shows a web page titled "Log in page" in the top-left corner. Below the title, there are two input fields: one labeled "UserName" with the text "username" inside, and another labeled "Password" with the text "*****" inside. Below these fields is a button labeled "Log in".

Рисунок 12 – Войти

2.4.1. Административная часть

На домашней странице администратора отображаются все элементы, доступные в приложении, включая пользователей, область, тип, клиенты, счета и статистику (рисунок 13).

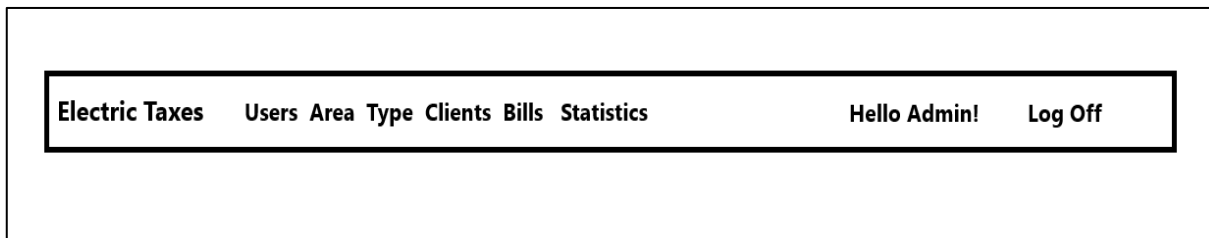


Рисунок 13 – Панель задач администратора

Пользователи (Users)

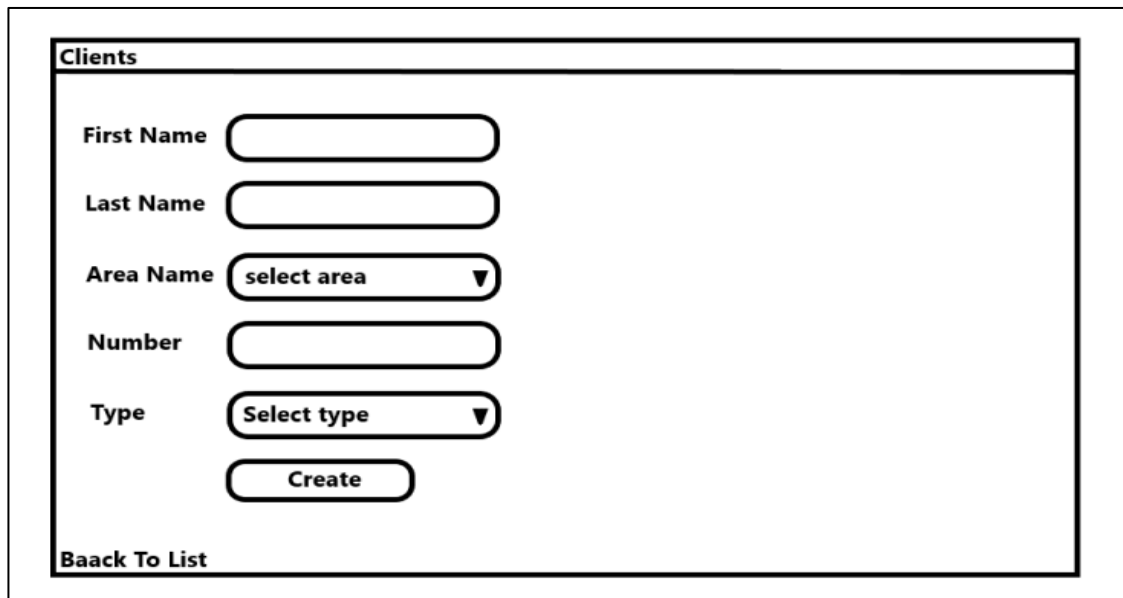
На странице «Пользователи» администратор имеет возможность создать новый файл пользователя, путём указания имени и пароля пользователя, а также выбора его типа (администратор или сборщик) (рисунок 14).

The image shows a form titled "Users" with a black border. Inside the form, there are three input fields: "User Name" with the placeholder text "user name", "Password" with the placeholder text "*****", and "Type" with a dropdown menu showing "Select type" and a downward arrow. Below these fields is a "Create" button. At the bottom left of the form, there is a link labeled "Back To List".

Рисунок 14 – Дизайн страницы пользователей

Клиенты (Clients)

Администратор может выполнить аналогичное действие на странице «Клиенты», добавив нового клиента. Для этого необходимо добавить информацию о клиенте (рисунок 15).

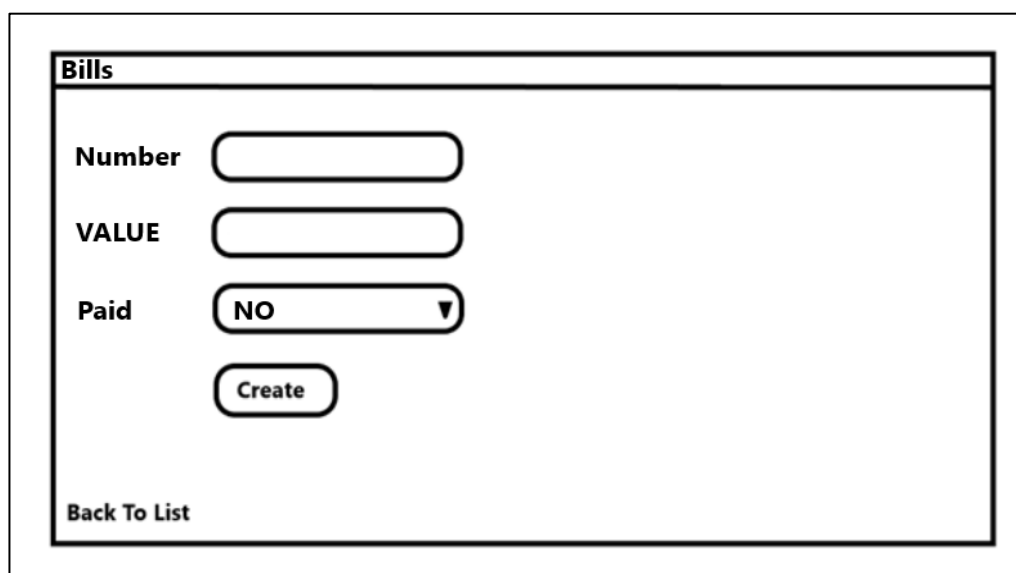


The screenshot shows a web form titled "Clients" enclosed in a rectangular border. Inside the border, the form fields are arranged vertically on the left side, each with a label and an input field to its right. The fields are: "First Name" with a text input, "Last Name" with a text input, "Area Name" with a dropdown menu showing "select area" and a downward arrow, "Number" with a text input, and "Type" with a dropdown menu showing "Select type" and a downward arrow. Below these fields is a "Create" button. At the bottom left of the form area is a link labeled "Baack To List".

Рисунок 15 – Дизайн страницы клиентов

Счета (Bills)

В разделе «Счета», как и в предшествующих разделах, администратор имеет возможность добавлять, изменять или удалять счета, а также определять, был ли счет оплачен (рисунок 16).

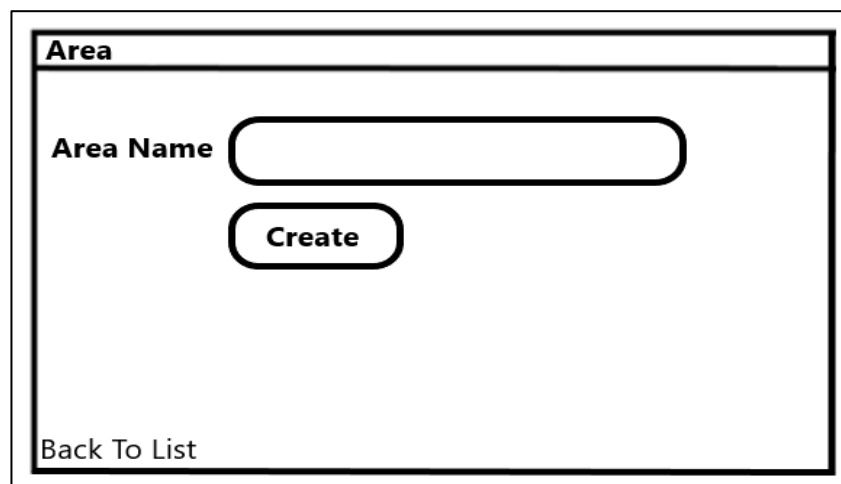


The screenshot shows a web form titled "Bills" enclosed in a rectangular border. Inside the border, the form fields are arranged vertically on the left side, each with a label and an input field to its right. The fields are: "Number" with a text input, "VALUE" with a text input, and "Paid" with a dropdown menu showing "NO" and a downward arrow. Below these fields is a "Create" button. At the bottom left of the form area is a link labeled "Back To List".

Рисунок 16 – Дизайн страницы счетов

Область (Area)

Дизайн веб-страницы «Область» представляет собой простой макет с одним полем ввода, где администратор может ввести название нового региона, который он хочет добавить в систему. Дизайн страницы заключается в ее основной функции, которая заключается в предоставлении пользователю возможности выполнять операции в этой области (рисунок 17).

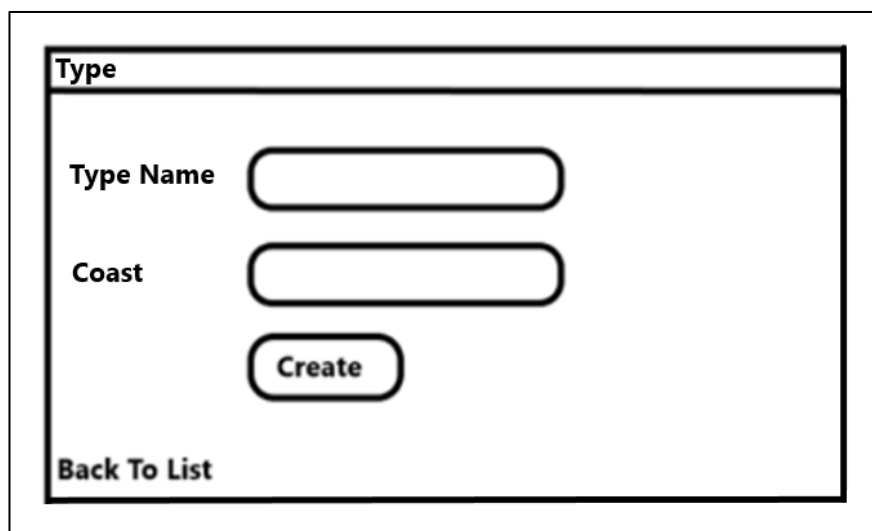


The image shows a wireframe for a web page titled "Area". At the top, there is a header bar with the word "Area". Below the header, on the left, is the label "Area Name". To the right of this label is a rounded rectangular text input field. Directly below the input field is a rounded rectangular button labeled "Create". In the bottom-left corner of the main content area, there is a text link that says "Back To List".

Рисунок 17 – Дизайн страницы области

Тип (Type)

На рисунке 18 показан дизайн страницы «Тип». На этой странице указаны тип и стоимость.



The image shows a wireframe for a web page titled "Type". At the top, there is a header bar with the word "Type". Below the header, on the left, are two labels: "Type Name" and "Coast". To the right of "Type Name" is a rounded rectangular text input field. To the right of "Coast" is another rounded rectangular text input field. Directly below the "Coast" input field is a rounded rectangular button labeled "Create". In the bottom-left corner of the main content area, there is a text link that says "Back To List".

Рисунок 18 – Дизайн страницы Тип

2.4.2. Пользовательская часть

На домашней странице пользователя отображаются все элементы, доступные в приложении, включая клиентов и счета (рисунок 19).

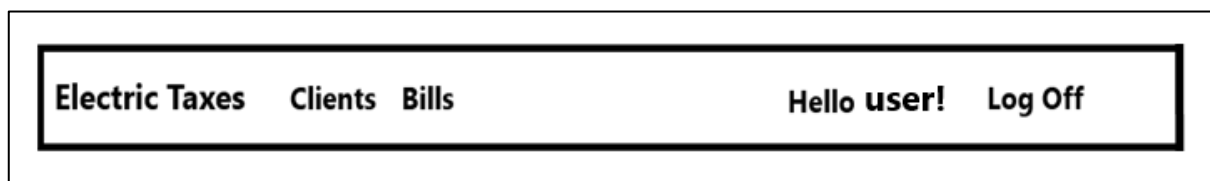


Рисунок 19 – Панель задач пользователя

Как было упомянуто ранее, пользователям доступны функции «Клиенты» и «Счета». Дизайн данных страниц в точности соответствует дизайну страниц администратора, так что различий между ними нет. Удобство использования для пользователей приложения является ключевым аспектом, который учитывается при разработке функционала. Предоставление пользователям схожих интерфейсов способствует простоте в обращении и минимизации возможных ошибок как показано на рисунке 20.

A form titled "Clients" with a header bar. Below the header, there are five labeled input fields: "First Name", "Last Name", "Area Name" (a dropdown menu showing "select area"), "Number", and "Type" (a dropdown menu showing "Select type"). Below these fields is a "Create" button. At the bottom left of the form is a link labeled "Baack To List".

Рисунок 20 – Дизайн страницы клиента для пользователя

2.5. Дизайн Android приложений

Главная страница

На главной странице Android-приложения присутствуют следующие элементы. Поле ввода имени пользователя, предназначенное для ввода имени зарегистрированного пользователя приложения; поле ввода пароля, которое представляет собой защищенную зону, где пользователи могут ввести свой пароль; а также поле ввода, предназначенное для ввода номера счета. Расположенные на странице элементы ввода, такие как поля для ввода имени пользователя, пароля и номера счета, имеют интуитивно понятный дизайн и легко доступны для использования. Кнопка входа расположена в нижней части главной страницы и позволяет пользователям войти в приложение после ввода необходимых данных (рисунок 21).

The diagram illustrates the layout of an Android login screen. It is enclosed in a rectangular frame. At the top, there is a header bar labeled "Android". Below this, the text "Log In" is centered. Following "Log In", there are three input fields, each with a label above it: "User Name", "Password", and "Enter Bill Number". The "Password" field is represented by a horizontal line, indicating a masked input. Below the "Enter Bill Number" field, the word "OR" is centered. At the bottom of the input section, there is a rounded rectangular button labeled "Sign In". The entire login form is contained within a larger rectangular area, which is itself within a frame.

Рисунок 21 – Дизайн Android-приложения

3. РЕАЛИЗАЦИЯ ПРОЕКТА

Для реализации проекта было принято решение использовать язык C# с использованием фреймворка Windows Forms Application (.NET Framework). Этот выбор был сделан после тщательного изучения различных языков программирования и сред с целью создания надежного и эффективного приложения, отвечающего требованиям проекта.

Для эффективного управления данными приложения в качестве системы управления базами данных (СУБД) была выбрана SQL Server 2014 Management Studio. Эта конкретная СУБД широко известна своей надежностью, масштабируемостью и безопасностью, что делает ее идеальным выбором для разработанного веб-приложения.

Кроме того, язык программирования Java также использовался в процессе разработки, в частности, при создании приложения для Android. Для этой цели использовалась популярная программа ANDROID STUDIO, позволяющая эффективно создавать приложения.

3.1. Реализация веб-приложения

C# – это современный, типобезопасный и объектно-ориентированный язык программирования, который широко используется в индустрии разработки программного обеспечения. предоставляет языковые конструкции для поддержки компонентно-ориентированного программирования, который становится все более популярным подходом к разработке программного обеспечения, основанным на автономных и самоописываемых пакетах функций. Эти компоненты предлагают модель программирования со свойствами, методами и событиями и включают атрибуты, предоставляющие декларативную информацию о компоненте, а также собственную документацию.

C# – это естественный язык для создания и использования программных компонентов благодаря встроенной поддержке этих

концепций. Кроме того, некоторые функции C# облегчают создание надежных и долговечных приложений. Например, сборка мусора автоматически освобождает память, занятую неиспользуемыми объектами, а обработка исключений обеспечивает структурированный и расширяемый подход к обнаружению и устранению ошибок. Более того, типобезопасный дизайн языка делает невозможным чтение из неинициализированных переменных или массивов индексов за их границы или выполнение непроверенных приведений типов.

C# имеет унифицированную систему типов, что означает, что все типы, включая примитивные типы, такие как `int` и `double`, наследуются от одного корневого типа объекта. В результате все типы используют набор общих операций, а значения любого типа можно хранить, транспортировать и обрабатывать согласованным образом. Кроме того, C# поддерживает как определяемые пользователем ссылочные типы, так и типы значений, что позволяет динамически размещать объекты и хранить легковесные структуры в потоке.

C# предоставляет богатый набор библиотечных функций и разработан как простой, современный и масштабируемый язык программирования. ориентирован на компоненты, структурирован и поддерживает взаимодействие с другими языками программирования. Кроме того, C# работает быстро благодаря малому времени компиляции и выполнения. Его функции делают его популярным выбором для разработки масштабируемых, функционально совместимых и надежных приложений.

3.2. Реализация Android-приложений

Java – это широко используемый объектно-ориентированный язык программирования, который хорошо подходит для разработки приложений для Android. Одной из причин, по которой Java является хорошим выбором для разработки приложений для Android, является ее гибкость. Код Java

может выполняться на любой платформе с виртуальной машиной Java (JVM), а это означает, что код можно написать один раз и запустить на нескольких устройствах. Такая переносимость упрощает разработку приложений для Android, которые работают на самых разных устройствах.

Еще одним преимуществом использования Java для разработки приложений для Android является большое и активное сообщество разработчиков. Существует множество ресурсов для изучения Java, от онлайн-учебников до книг и форумов. Это сообщество также производит большое количество библиотек и инструментов, которые делают разработку более быстрой и эффективной.

Java также является безопасным языком программирования с такими функциями, как автоматическое управление памятью и обработка исключений, которые помогают предотвратить сбои и другие проблемы безопасности. Кроме того, Java предлагает богатый набор библиотек и сред, таких как Android Studio и Gradle, которые упрощают создание и развертывание высококачественных приложений для Android.

3.3. Несколько фрагментов C#-кода

Контроллер входа в систему (LoginController)

Этот код представляет собой класс контроллера, который обрабатывает аутентификацию пользователя и управление сессией. Код определяет два метода: Index и logout. Метод Index получает запрос POST и проверяет, совпадают ли предоставленные пользователем имя пользователя и пароль со значениями, хранящимися в таблице USERACCOUNTS базы данных. Если пользователь аутентифицирован, переменные сессии «userName» и «role» устанавливаются с данными пользователя, и контроллер перенаправляет пользователя на метод действия «Index» контроллера «CLIENTS». Если пользователь не аутентифицирован, метод возвращает представление. Метод logout удаляет переменные сессии

пользователя и перенаправляет пользователя к методу действия «Индекс» контроллера «По умолчанию». Код использует LINQ to SQL для извлечения и сравнения учетных данных пользователя из базы данных (рисунок 22).

```
1 public class LoginController : Controller
2 {
3     private ETEntities db = new ETEntities();
4
5     // GET: Default
6     public ActionResult Index()
7     {
8         return View();
9     }
10
11     [HttpPost]
12     public ActionResult Index(USERACCOUNT u)
13     {
14         USERACCOUNT userACCOUNTS = (from q in db.USERACCOUNTS where q.USERNAME == u.USERNAME && q.PASSWORD == u.PASSWORD select
15             q).SingleOrDefault();
16         if (userACCOUNTS != null)
17         {
18             Session["userName"] = userACCOUNTS.USERNAME;
19             Session["role"] = userACCOUNTS.TYPE.ToString().Trim();
20             if (Session["role"].ToString() == "admin")
21             {
22                 return RedirectToAction("Index", "CLIENTS");
23             }
24         }
25         return View();
26     }
27
28     public ActionResult logOut()
29     {
30         try { Session["userName"] = null; } catch { }
31         try { Session["role"] = null; } catch { }
32
33         return RedirectToAction("Index", "Default");
34     }
35 }
```

Рисунок 22 – Контроллер входа в систему

Контроллер счетов (BILLSController)

Этот код определяет контроллер для веб-приложения, которое управляет счетами за электроэнергию. Контроллер взаимодействует с базой данных Entity Framework через запросы LINQ для выполнения операций CRUD в таблице «BILLS», которая содержит информацию об использовании электроэнергии клиентами и счетах.

Метод «Индекс» извлекает все счета из базы данных и отображает их в представлении вместе с раскрывающимися списками для фильтрации результатов на основе имени клиента, области, типа и даты счета. Метод «Подробности» извлекает конкретный счет по его идентификатору и отображает его сведения в представлении. Метод «Создать» получает на

вход новый счет и создает новую запись в базе данных после выполнения некоторых расчетов на основе использования клиентом и тарифов на электроэнергию. Метод «Редактировать» обновляет существующий счет новыми значениями, предоставленными пользователем. Метод «Удалить» удаляет счет из базы данных.

Контроллер использует ViewBag для передачи данных в представление и принимает входные данные через HTTP-запросы GET и POST. Запросы LINQ используются для фильтрации счетов на основе ввода пользователя, который затем передается в представление для отображения. Код также включает обработку исключений для обнаружения ошибок, которые могут возникнуть во время транзакций базы данных (рисунок 23).

```
public class BILLSController : Controller
{
    private ETEntities db = new ETEntities();

    // GET: BILLS
    public ActionResult Index()
    {
        ViewBag.CounterTypes = db.TYPES.ToList();
        ViewBag.Areas = db.AREAS.ToList();
        List<BILL> ff = db.BILLS.ToList();
        return View(db.BILLS.ToList());
    }

    [HttpPost]
    public ActionResult Index( string number, string name, string area, string type,string since,string to)
    {
        ViewBag.CounterTypes = db.TYPES.ToList();
        ViewBag.Areas = db.AREAS.ToList();

        ViewBag.name = name;
        ViewBag.number = number;
        ViewBag.area = area;
        ViewBag.type = type;
        ViewBag.since = since;
        ViewBag.to = to;
        List<BILL> b = db.BILLS.ToList();
        //List<BILL> res = new List<BILL>();
        if (!string.IsNullOrEmpty(number))
        {
            b = (from q in b where q.NUMBER.Trim()==number.Trim() select q).ToList();
            return View(b);
        }
    }
}
```

Рисунок 23 – Контроллер счетов

3.4. Несколько фрагментов кода JAVA

Клиент (Customer)

Этот код создает действие под названием «Customer», которое показывает ListView. ListView заполняется данными из веб-службы SOAP, которые используются с помощью библиотеки AndroidHttpTransport.

У действия есть метод onCreate, который начинается с установки политики StrictMode для потока, затем извлекает строковое значение из предыдущего действия и отправляет его в веб-службу в качестве параметра. Затем ответ от веб-службы обрабатывается путем разбиения строковых данных на несколько частей и их сохранения в массивах. Затем эти массивы передаются пользовательскому ArrayAdapter с именем menuAdapter, который используется для заполнения ListView данными.

Класс menuAdapter расширяет класс ArrayAdapter и переопределяет его метод getView для создания пользовательского представления для каждой строки в ListView. Он расширяет макет с именем menu_row и устанавливает значения его TextView с данными из массивов, переданных в конструкторе (рисунок 24).

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

    StrictMode.setThreadPolicy(policy);

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_customer);

    Bundle b=getIntent().getExtras();
    counterN=b.getString("userName");

    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
    request.addProperty("cn", counterN);

    SoapSerializationEnvelope sse = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    sse.dotNet = true;
    sse.setOutputSoapObject(request);
    AndroidHttpTransport aht = new AndroidHttpTransport(URL);

    try {
        aht.call(SOAP_ACTION, sse);
        SoapPrimitive res = (SoapPrimitive) sse.getResponse();
        x=res.toString();
        //Toast.makeText(Customer.this, x, Toast.LENGTH_LONG).show();
        String[] sep=x.split("_");
        for(int i=0;i<sep.length;i++)
        {
            String[] sep1=sep[i].split("#");

            types[i]=sep1[0];
            prices[i]=sep1[1];

            datee[i]=sep1[2];
            paids[i]=sep1[3];
        }
    }
```

Рисунок 24 – Клиент (Customer)

Сборщик налогов (TaxCollector)

Этот код определяет действие под названием TaxCollector и реализует AsyncTask, который используется для выполнения длительных операций в фоновом режиме, отдельно от потока пользовательского интерфейса.

Метод onCreate вызывается при первом создании действия и инициализирует макет действия. Метод onCreateOptionsMenu вызывается для создания меню параметров действия, а метод onOptionsItemSelected вызывается, когда пользователь выбирает элемент в меню параметров.

Метод отправки вызывается, когда пользователь нажимает кнопку отправки в действии. Он создает экземпляр класса AsyncCallWS, который расширяет AsyncTask, и вызывает его метод execute.

В классе AsyncCallWS реализован метод doInBackground, выполняющий основную работу задачи в фоновом режиме, и метод sendMyXY, использующий протокол SOAP для связи с удаленным веб-сервисом.

Метод doInBackground выполняет метод sendMyXY, который считывает данные из двух представлений EditText в действии и отправляет их веб-службе с помощью SOAP. Затем он получает ответ от веб-службы и отображает его во всплывающем сообщении в действии (рисунок 25).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tax_collector);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
    // present.
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement

    return super.onOptionsItemSelected(item);
}
```

Рисунок 25 – Сборщик налогов (TaxCollector)

4. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

4.1. Тестирование веб-приложения

Дизайн пользовательского интерфейса фокусируется на предвидении того, что пользователям может понадобиться сделать, и обеспечении того, чтобы в интерфейсе были элементы, к которым легко получить доступ, понять и использовать для облегчения этих действий [16].

На рисунок 26 показана «Главная страница». На этой странице клиент может напрямую добавить свой номер или свое имя и фамилию, чтобы проверить свой последний счет, а администратор и коллектор могут нажать кнопку входа, чтобы они могли войти в систему.

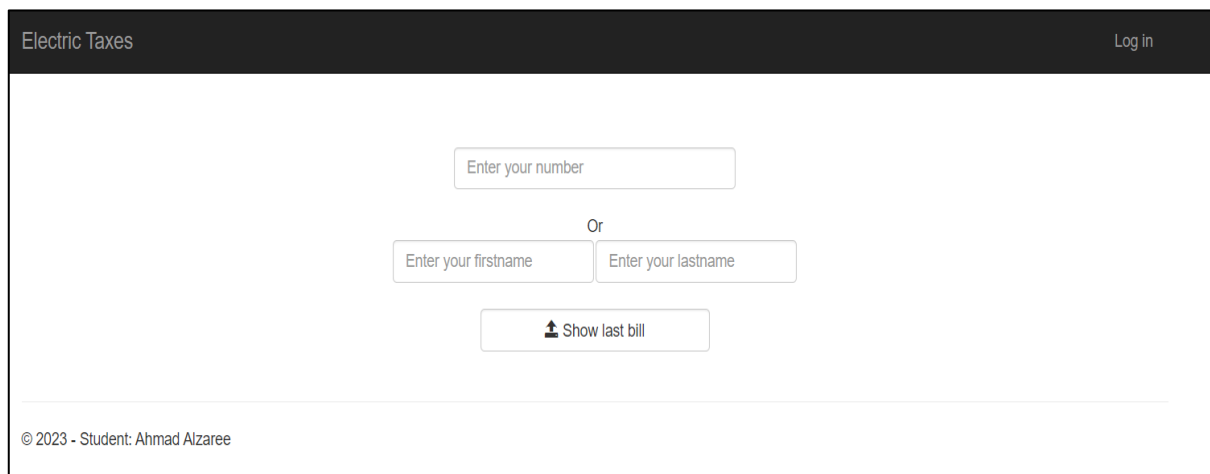


Рисунок 26 – Главная страница

Иллюстрация, изображенная на рисунке 27, отображает «Домашнюю страницу администратора». На изображении представлены несколько отдельных элементов, каждый из которых отвечает за определенные функции. Как показано на изображении, эти элементы включают пользователей, область, тип, клиентов, счета и статистику.

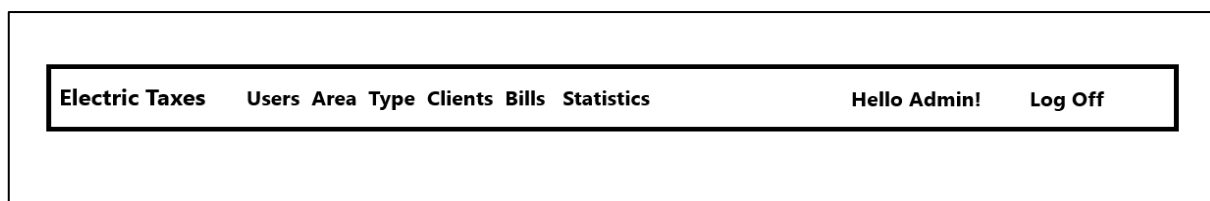


Рисунок 27 – Панель задач администратора

На странице «Пользователи» администратор может добавить, изменить или даже удалить нового пользователя, а также упорядочить пользователей по типу учетной записи (рисунок 28).

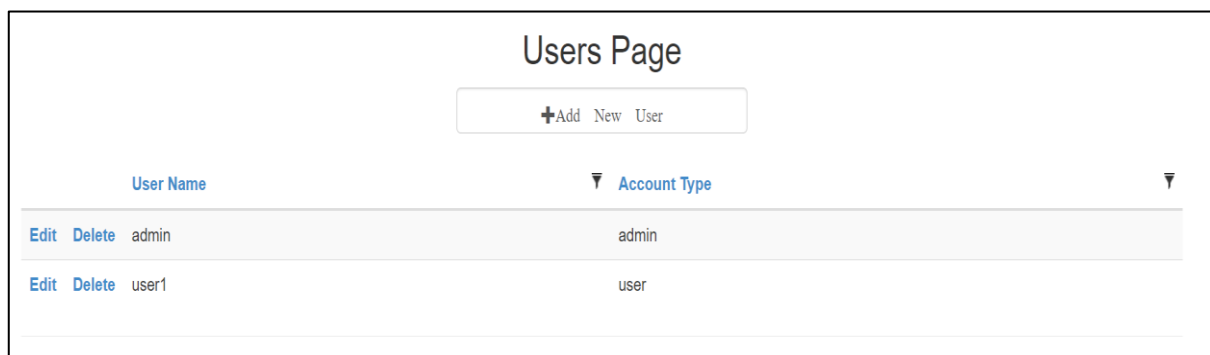


Рисунок 28 – Страница пользователей

На рисунок 29 показана страница «областей». Эта страница служит для администратора платформой для добавления, редактирования или удаления региона в зависимости от требований компании.

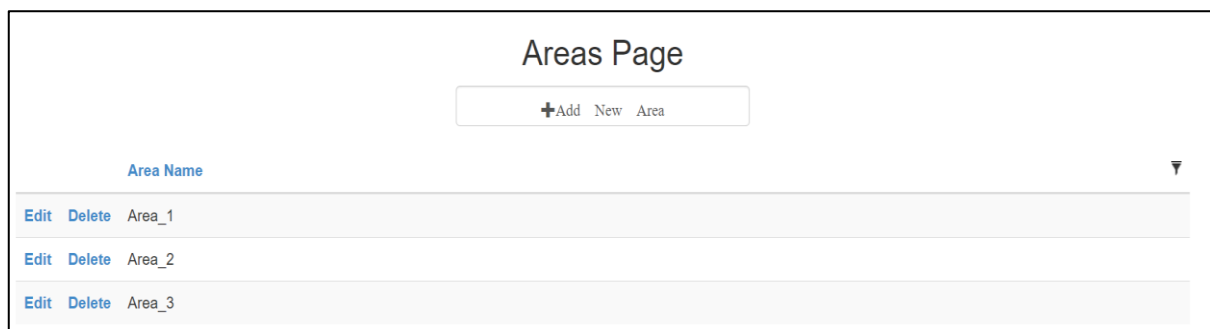


Рисунок 29 – Страницы областей

На странице «Типы» администратор может добавить новый тип счетчика электроэнергии, отредактировать старый тип или удалить его (рисунок 30).

Types Page		
+Add New Type		
Type Number	Coast per Kilo watt	
Edit Delete Type_1	100	
Edit Delete Type_2	150	
Edit Delete Type_3	200	

© 2023 - Student: Ahmad Alzaree

Рисунок 30 – Страницы Типы

На странице «Клиенты» администратор может добавлять новых клиентов, редактировать информацию о клиенте, отображать сведения об использовании или удалять клиентов (рисунок 31).

Clients Page						
+Add New Client						
First Name	Last Name	Area Name	Number	Type	Last Read	
Edit Details Delete AA	AA	Area_1	001	Type_3	200	
Edit Details Delete AB	AB	Area_2	002	Type_2	1500	
Edit Details Delete AC	AC	Area_3	003	Type_1	1500	

Рисунок 31 – Страницы Клиенты

Данные клиента на странице клиента 32.

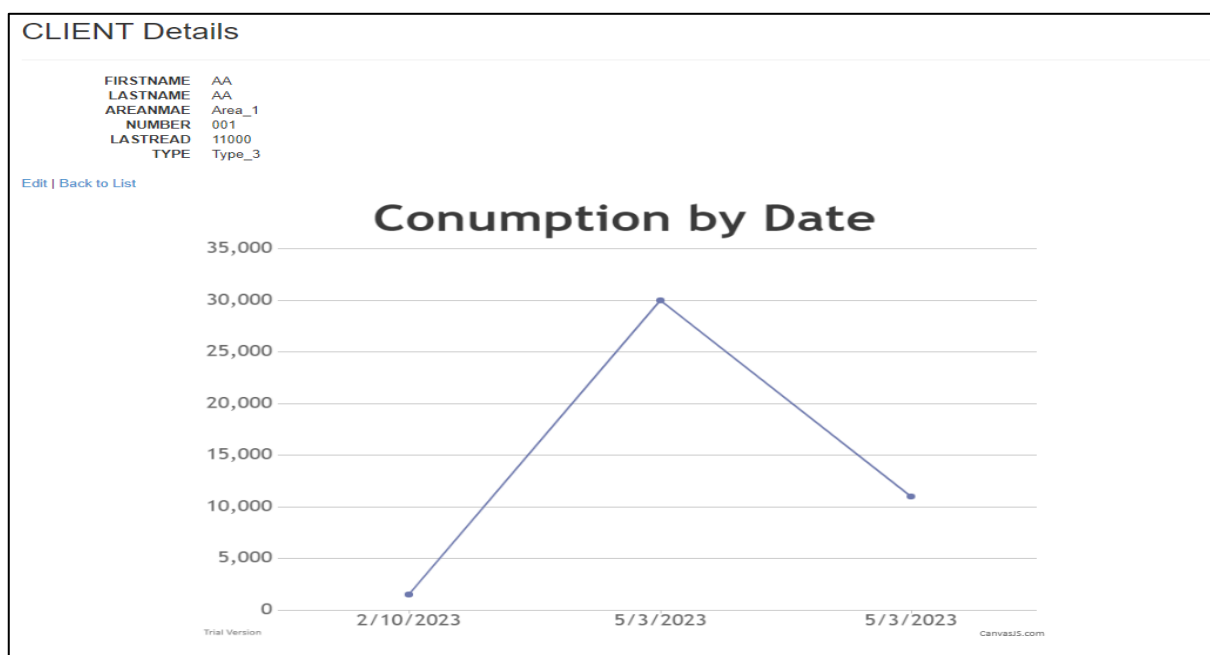


Рисунок 32 – Данные клиента

Страница «Счета» предоставляет администратору возможность добавлять или удалять счета. Кроме того, администратор может проверить, был ли оплачен счет, и использовать кнопку «Оплатить», чтобы обновить статус оплаты (рисунок 33).

+Add New Bill									
Number		Client Name		Area	Type	Since		Until	Q Search
<input type="text"/>		<input type="text"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>
Client Name		Area		Number	Type	Value	Cost	Bill Date	PAID
Delete	PAY	AAAA	Area_1	001	Type_3	1500	300000	2/10/2023 3:35:46 PM	YES
Delete	PAY	AB AB	Area_2	002	Type_2	1500	225000	2/10/2023 3:35:54 PM	YES
Delete	PAY	AC AC	Area_3	003	Type_1	1500	150000	2/10/2023 3:36:01 PM	YES
Delete	PAY	AAAA	Area_1	001	Type_3	30000	5960000	5/3/2023 8:33:38 PM	No
Delete	PAY	AAAA	Area_1	001	Type_3	11000	-3800000	5/3/2023 8:33:51 PM	No

Рисунок 33 – Страница счетов

Страница статистики позволяет администратору отображать две различные формы данных о потреблении: потребление по регионам (рисунок 34) и потребление по типам (рисунок 35). Программа автономно генерирует статистические данные, не требуя вмешательства человека.

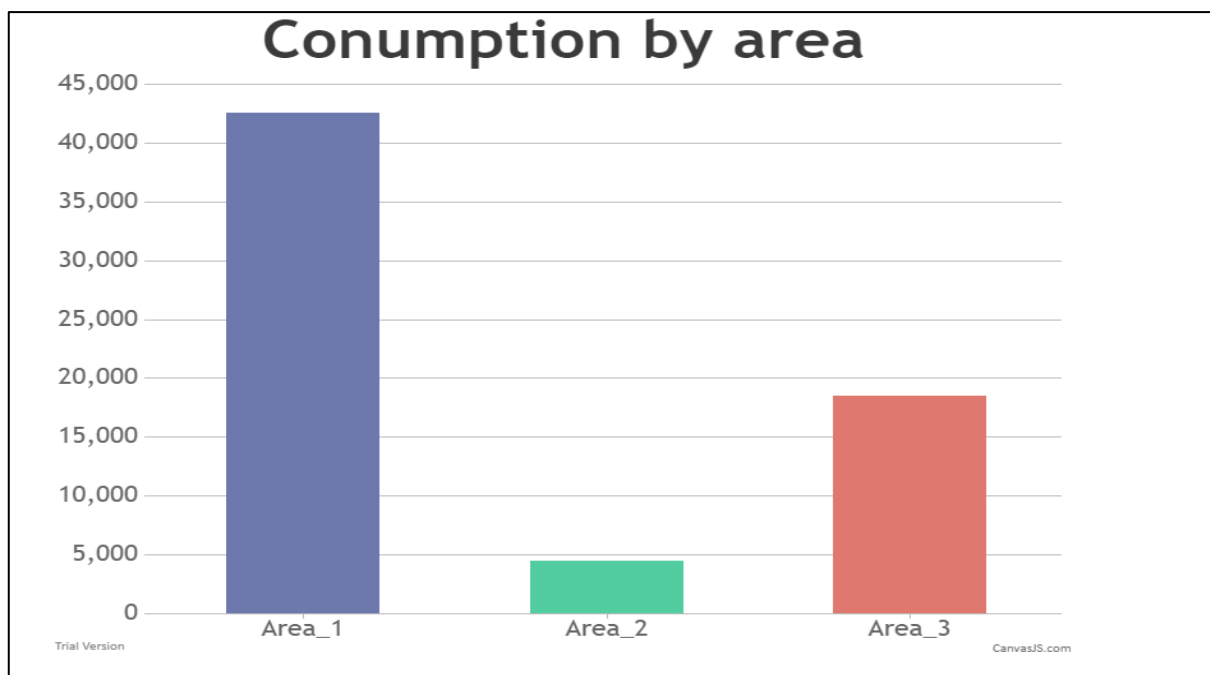


Рисунок 34 – Потребление по регионам

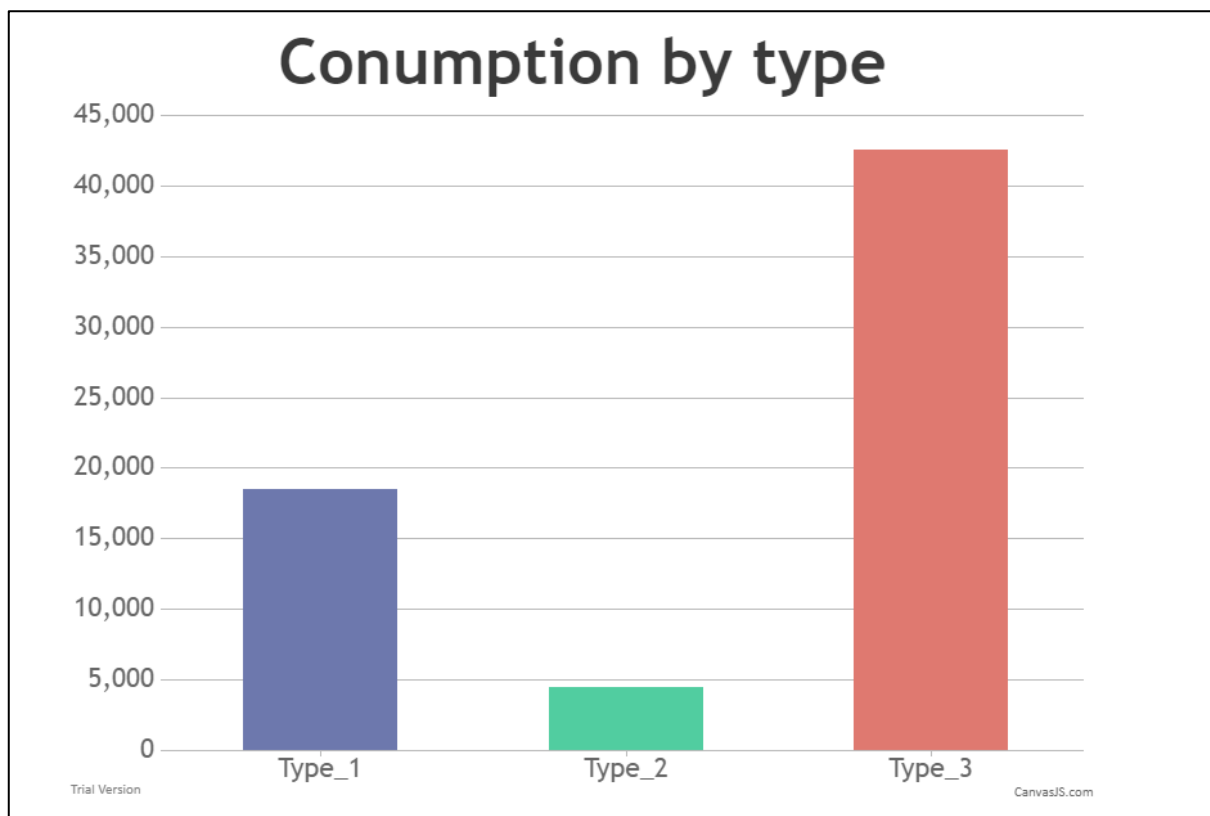


Рисунок 35 – Потребление по типам

На рисунке 36 показана «Домашняя страница пользователя». На изображении показаны несколько отдельных элементов, каждый из которых отвечает за определенные функции. Как показано на изображении, эти элементы включают клиентов и счета.

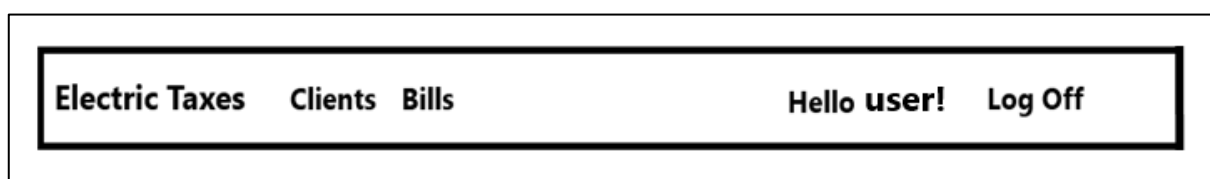


Рисунок 36 – Панель задач пользователя

Разделы «Клиенты» и «Счета» в учетной записи пользователя выполняют те же функции, что и в учетной записи администратора. Эти разделы видны как на рисунках 31 и 33.

4.2. Тестирование Android-приложений

Количество операций, доступных в приложении для Android, сравнительно меньше, чем в веб-приложении. Основная цель разработки приложения для Android заключалась в том, чтобы упростить задачи сборщика, который теперь может использовать свой телефон Android для добавления счетов в систему без необходимости использования отдельного считывающего устройства. Кроме того, была включена дополнительная функция, позволяющая клиентам проверять свой последний счет. Эта функция была легко интегрирована в приложение для повышения его общей функциональности. Кроме того, администратор может войти в систему со своего телефона, чтобы получить доступ к счетам и просмотреть их. Однако следует отметить, что возможность оплачивать счета через приложение для Android недоступна, и для выполнения этой задачи администратор или коллектор должны войти в веб-приложение. В целом, Android-приложение предоставляет коллекторам и администраторам удобное решение для управления счетами, хотя и с ограниченными функциональными возможностями по сравнению с веб-приложением.

После запуска Android-приложения на устройстве, на экране пользователя появляется специальная страница входа, предоставляющая возможность получить доступ к системе как в качестве администратора, так и в качестве сборщика. Кроме того, в состав пользовательского интерфейса входит интуитивно понятная функция отображения номера счета-фактуры, благодаря которой клиенту упрощается процесс проверки своего счета. Данная характеристика приложения значительно облегчает пользовательский опыт и способствует более эффективной работе с системой. Будучи разработанной с учетом пользовательских потребностей, эта функция предоставляет клиентам дополнительный уровень удобства и надежности, что приводит к повышению удовлетворенности пользователей и повышению эффективности работы в целом (рисунок 38).

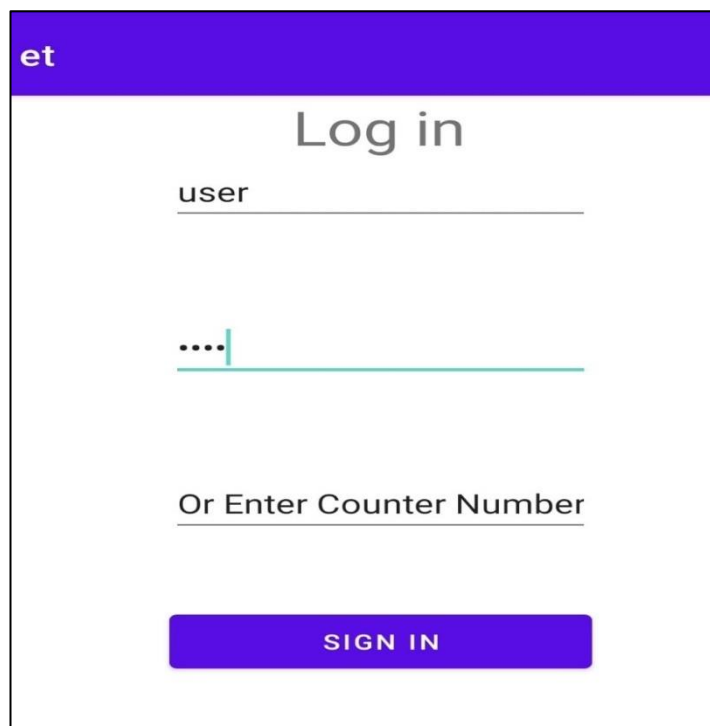


Рисунок 38 – Интерфейс домашнего Android-приложения

При входе в систему пользователю будет представлен интерфейс с обозначенными полями для ввода номера клиента и номера электросчетчика (рисунок 39).

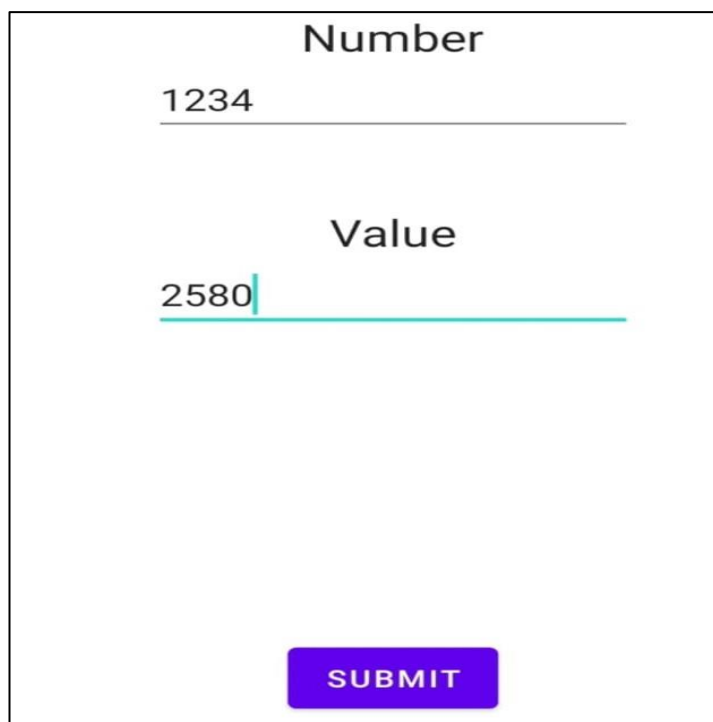


Рисунок 39 – Страница пользователя в приложении для Android

При запуске поиска номера счетчика электроэнергии приложение направляет клиента на специальную страницу, где отображается вся необходимая информация, связанная с номером счетчика. Эта информация включает исчерпывающую информацию о номере счетчика, включая его текущий статус, соответствующую историю потребления и любые соответствующие дополнительные расходы или неоплаченные счета. На этой странице также представлен обзор всех счетов, связанных с учетной записью клиента, что позволяет им удобно просматривать и отслеживать потребление энергии (рисунок 40).

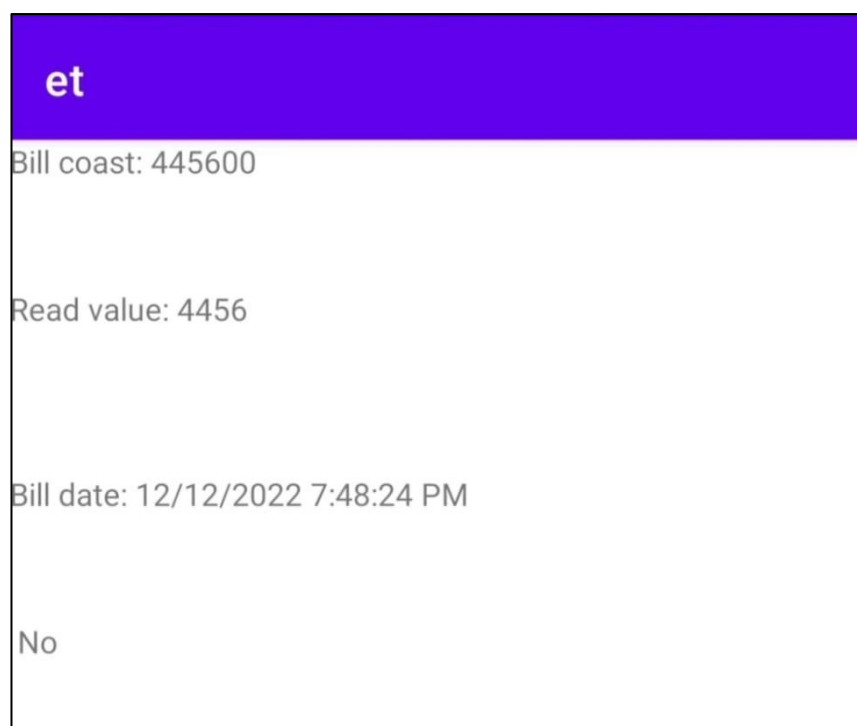


Рисунок 40 – Информация о счете

4.3. Функциональное тестирование

Тестирование программного обеспечения определяется как деятельность по проверке соответствия фактических результатов ожидаемым результатам и обеспечению отсутствия дефектов в программной системе. Он включает в себя выполнение программного компонента или системного компонента для оценки одного или нескольких интересующих свойств.

Функциональное тестирование – это процесс обеспечения качества в рамках цикла разработки программного обеспечения, необходимый для проверки реализуемости функциональных требований, согласно спецификации тестируемого программного обеспечения. Функциональное тестирование проводится для оценки соответствия системы или компонента заданным функциональным требованиям [17].

Поскольку программные приложения становятся все более сложными и взаимосвязанными, а также с большим количеством различных платформ и устройств, которые необходимо протестировать, как никогда важно иметь надежную методологию тестирования, чтобы убедиться, что разрабатываемые программные продукты/системы полностью прошли проверку. протестированы, чтобы убедиться, что они соответствуют заданным требованиям и могут успешно работать во всех предполагаемых средах с требуемым удобством использования и безопасностью [18].

Каждый аспект функциональности системы подвергается тестированию путем предоставления соответствующих входных данных, проверки выходных данных и сравнения фактических результатов с ожидаемыми результатами.

Процесс функционального тестирования:

- 1) определить тестовый ввод;
- 2) вычислить ожидаемые результаты с выбранными входными значениями теста;
- 3) выполнить тест-кейсы;
- 4) сравнить фактический и ожидаемый результат.

Таким образом, проводится сравнение фактических и ожидаемых результатов (таблица 1).

Таблица 1 – Результаты тестирования

№	Функция	Полученный результат	Желаемый результат	Заключение
1	При вводе клиентом своего имени и фамилии или номера.	Клиент имеет возможность увидеть поле для ввода своего имени и фамилии или своего номера.	Клиент имеет возможность увидеть поле для ввода своего имени и фамилии или своего номера.	Проверено
2	Пользователь может ввести имя пользователя и пароль	Пользователь может увидеть «страницу входа» и ввести имя пользователя и пароль для входа в приложение.	Пользователь может увидеть «страницу входа» и ввести имя пользователя и пароль для входа в приложение.	Проверено
3	Чтобы дать пользователю сообщение при вводе неправильного имени пользователя или пароля	Имя пользователя или пароль Любой пользователь увидит сообщение. Если он ввел Неверное имя пользователя или неверный пароль	Имя пользователя или пароль Любой пользователь увидит сообщение. Если он ввел Неверное имя пользователя или неверный пароль	Проверено
4	Когда администратор входит в систему, он может CRUD пользователей	Пользователь «Администратор» может создавать, читать, обновлять и удалять пользователей.	Пользователь «Администратор» может создавать, читать, обновлять и удалять пользователей.	Проверено
5	Когда администратор входит в систему, он может обслуживать CRUD	Пользователь «Администратор» может создавать, читать обновления и удалять услуги	Пользователь «Администратор» может создавать, читать обновления и удалять услуги	Проверено
6	Когда Админ входит в систему, он может прочитать всю информацию	Администратор может прочитать всю информацию в системе	Администратор может прочитать всю информацию в системе	Проверено
7	Сборщик имеет право добавить счет в систему и присвоить ему статус оплаченного или неоплаченного.	Сборщик может добавить счет в систему и присвоить ему статус оплаченного или неоплаченного.	Сборщик может добавить счет в систему и присвоить ему статус оплаченного или неоплаченного.	Проверено
8	Сборщик имеет право добавлять клиента в систему и редактировать или удалять его.	Сборщик может добавить клиента в систему и отредактировать или удалить его	Сборщик может добавить клиента в систему и отредактировать или удалить его	Проверено

После проведения тщательного сопоставления фактических результатов с ожидаемыми, пришли к выводу, что все функции в приложении работают в соответствии с предполагаемыми ожиданиями и успешно прошли все этапы тестирования. Анализу подверглись все восемь функций, описанных в представленной таблице. Первая функция, связанная с возможностью ввода имени, фамилии или номера клиента, позволяет пользователю удобно идентифицировать себя в системе. Вторая функция, позволяющая пользователям вводить имя пользователя и пароль, обеспечивает безопасный доступ к приложению через страницу входа. Третья функция, позволяющая сообщать пользователям об ошибочном вводе имени пользователя или пароля, обеспечивает информативное обратное сообщение и помогает пользователям исправить ошибку. Четвертая и пятая функции, связанные с возможностью администратора управлять пользователями и услугами в системе, позволяют эффективно управлять данными и обеспечивать необходимую функциональность. Шестая функция, предоставляющая администратору полный доступ к всей информации в системе, обеспечивает удобство и оперативность при просмотре данных. Седьмая и восьмая функции, связанные с возможностями сборщика, позволяют добавлять счета и клиентов в систему, а также редактировать или удалять их, обеспечивая полный контроль и гибкость при управлении данными. Таким образом, все функции успешно протестированы и соответствуют ожиданиям, что гарантирует надежность и функциональность приложения.

ЗАКЛЮЧЕНИЕ

Цель данной системы заключается в замене бумажных операций и традиционных устройств сбора платежей на Android-телефон в процессе сбора платежей. Кроме того, система была создана для упрощения автоматизации операций, связанных с электроэнергетическими компаниями и распределением электроэнергии с минимальными потерями.

Для потребителей данная система предоставляет платформу для управления своим электропотреблением и отслеживания его использования. Интуитивно понятный и удобный интерфейс облегчает мониторинг потребления энергии и понимание его размеров.

В целом, система представляет значительный шаг в совершенствовании электроэнергетического сектора и устранении неэффективностей, связанных с традиционными бумажными операциями. Применение новых технологий и автоматизация ключевых процессов позволили повысить эффективность работы электроэнергетической отрасли и обеспечить более гладкую работу для потребителей.

Для достижения данной цели были выполнены следующие задачи.

1. Проведен анализ проблемного заявления приложения.
2. Проанализированы современные технологии разработки настольных приложений и выбрана оптимальная технология для разработки проекта.
3. Разработана диаграмма прецедентов использования системы.
4. Правильно разработаны интерфейсы приложения.
5. Создана база данных приложения.
6. Реализована система.
7. Проведено функциональное тестирование.

ЛИТЕРАТУРА

1. SAP Help Portal (Документация). [Электронный ресурс] URL: <https://help.sap.com/docs> (дата обращения: 13.02.2023 г.).
2. Mindquest (Info). [Электронный ресурс] URL: <https://mindquest.io/blog/news/6429> (дата обращения: 13.02.2023 г.).
3. New.siemens (Документация). [Электронный ресурс] URL: <https://new.siemens.com/global/en/products/energy/grid-software/energyip-meter-data-management/energyip.html> (дата обращения: 13.02.2023 г.).
4. Linkedin (Info). [Электронный ресурс] URL: https://www.linkedin.com/pulse/exploring-siemens-next-generation-meter-data-paul-hobcraft?trk=portfolio_article-card_title (дата обращения: 13.02.2023 г.).
5. Chadwick J., Snyder T., Panda H. Programming ASP. NET MVC 4: Developing Real-World Web Applications with ASP. NET MVC. – O'Reilly Media, Inc., 2012. – С. 345.
6. Frain B. Responsive web design with HTML5 and CSS3. – Packt Publishing Ltd, 2012. – С. 43–62.
7. Duckett J. HTML & CSS: design and build websites. – Indianapolis, IN: Wiley, 2011. – Т.15, – С. 72.
8. Bob W. SQL Server 2022 Revealed, 2022 – С. 317–350.
9. Millett S. Professional ASP. NET Design Patterns. – John Wiley & Sons, 2010. – С. 523.
10. Visual Studio (Документация). [Электронный ресурс] <https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022> (дата обращения: 20.02.2023 г.).
11. Android Studio (Документация). [Электронный ресурс] <https://developer.android.com/studio/intro> (дата обращения: 05.03.2023 г.).
12. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language User Guide. – Pearson Education. First Edit. Addison Wesley 1998. – С. 512.

13. Rumbaugh J., Jacobson J. The Unified Modeling Language Reference Manual. Pearson Higher Education. Second Ed. – Boston: Second Edition, 2004. – С. 742.

14. W3schools (Документация). [Электронный ресурс] URL: <https://www.w3schools.in/dbms/relation-data-model/> (дата обращения: 04.03.2023 г.).

15. Схемы данных. (Документация). [Электронный ресурс] URL: https://www.tutorialspoint.com/dbms/dbms_data_schemas.htm (дата обращения: 05.03.2023 г.).

16. Дизайн пользовательского интерфейса (Документация). [Электронный ресурс] URL: <https://www.usability.gov/what-and-why/user-interface-design.html> (дата обращения: 25.03.2023 г.).

17. Функциональное тестирование Wikipedia (Документация). [Электронный ресурс] URL: ru.wikipedia.org/wiki/Функциональное_тестирование (дата обращения: 15.04.2023 г.).

18. Inflectra (Документация). [Электронный ресурс] URL: <https://www.inflectra.com/ideas/topic/testing-methodologies.aspx> (дата обращения: 15.04.2023 г.).