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)

Цифровая трансформация бизнеса
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка информационной системы для обработки корпоративной информации»

Обучающийся Д.К. Лаврентьев _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд. тех. наук, доцент, О. В. Аникина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. фил. наук, доцент, М. В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Бакалаврская работа выполнена на тему «Разработка информационной системы для корпоративной информации».

Цель работы – разработать приложение для сбора и обработки информации по сбору макулатуры для домов, обслуживаемых компанией «Квартплата 24».

Во введении раскрываются актуальность исследования, объект и предмет работы, а также ее цель и поставленные задачи.

В первой главе рассмотрены вопросы, которые изучают предмет исследования, а именно описание деятельности компании «Квартплата 24». и приводится описание бизнес-процессов по обработке корпоративной информации.

Во второй главе рассмотрены вопросы проектирования и разработки информационной системы для корпоративной информации.

В третьей главе рассчитана экономическая эффективность проекта и описан тестовый пример реализации приложения для корпоративной информации.

В заключении содержатся выводы о работе в целом и по главам.

В приложении приведен код, разработанного прототипа информационной системы.

Бакалаврская работа состоит из 60 страниц и включает 44 рисунка, 10 таблиц, 21 источник литературы и 4 приложения.

Abstract

The title of the graduation work is Developing an information system for corporate information.

The graduation work consists of an introduction, 3 parts, 44 figures, 10 tables, a conclusion, and a list of 21 references including foreign sources.

The aim of this graduation work is to develop an application for gathering and processing information on wastepaper collection for dwelling buildings serviced by OOO "Kvartplata 24" (LLC under the laws of the Russian Federation).

The object of the research is the information collection and processing.

The subject of the study is the process of gathering and processing the information on wastepaper collection.

The graduation work may be divided into several logically connected parts which are: literature review, results and their discussion and experimental part.

The first part describes the design and the development of the information system for collecting and processing information. In this part, the company organizational structure, and its business processes before and after the automation are explored as well. We also consider the analogues of the application being developed and state a technical task.

The second part is devoted to designing and developing the information system for gathering and processing the information on waste paper collection. The third part dwells on assessing the project economic efficiency and describing a test example of implementing the application for gathering and processing the information on waste paper collection. In conclusion, it should be highlighted that the created client-server application is designed to automate the process of accounting and controlling the waste paper collection. Its implementation will significantly reduce the time spent on data processing, minimize errors related to the human factor, and increase the overall efficiency of the enterprise's production processes. The work is of interest for a wide circle of readers interested in developing and designing information systems.

Оглавление

Введение

Актуальность темы исследования заключается в необходимости разработки приложения для обработки собираемой информации из различных источников компании «Квартплата 24».

В работе рассматривается проблема автоматизации обработки информации в сфере ЖКХ.

Объект исследования – сбор и обработка информации.

Предметом исследования - является процесс сбора и обработки информации по сбору макулатуры.

Цель работы - разработать информационную систему для обработки корпоративной информации.

Для достижения поставленной цели необходимо решить ряд ключевых задач:

- изучить предметную область деятельности компании «Квартплата 24», что позволит детально разобраться в специфике её работы, выявить основные требования к информационной системе и определить потенциальные проблемы, которые могут возникнуть при её внедрении;
- описать бизнес-процессы, связанные со сбором, обработкой и анализом корпоративной информации. Это позволит выявить ключевые этапы работы организации, определить возможные узкие места в процессе обработки данных и предложить оптимальные методы их устранения;
- разработать прототип приложения, которое будет учитывать выявленные потребности бизнеса. Данный этап включает в себя проектирование архитектуры системы, создание пользовательского интерфейса, разработку основных функциональных модулей и тестирование их работоспособности;
- оценить стоимость разработки и срок окупаемости проекта. Это позволит определить экономическую целесообразность внедрения

информационной системы, рассчитать возможные затраты и спрогнозировать сроки возврата инвестиций.

Структурно работа состоит из введения, трех глав, заключения, списка использованной литературы и приложения. Введение содержит постановку проблемы, определение целей и задач исследования, обоснование актуальности темы.

В первой главе рассматриваются теоретические аспекты предметной области, анализируются современные подходы к обработке корпоративной информации. Вторая глава посвящена описанию бизнес-процессов компании «Квартплата 24», выявлению проблем и формированию требований к разрабатываемой системе. В третьей главе представлена разработка прототипа приложения, оценка его стоимости и анализ сроков окупаемости.

Практическая значимость исследования заключается в применении полученных теоретических знаний к проектированию и разработке информационной системы, предназначенной для корпоративной информации. В ходе работы были использованы современные методы анализа, моделирования и разработки программного обеспечения, что позволило создать прототип, способный эффективно решать поставленные задачи.

Исследование бизнес процесса могут быть успешно внедрены в любую компанию, в которой актуальны вопросы использования, внедрения и разработки информационной системы для корпоративной информации. Данная разработка позволит автоматизировать процессы сбора и анализа данных, повысить эффективность управленческих решений, минимизировать возможные ошибки и сократить затраты на обработку информации. Таким образом, внедрение предложенной информационной системы может способствовать повышению конкурентоспособности предприятия и оптимизации его бизнес-процессов.

Глава 1 Анализ предметной области

1.1 Описание компании

Выполнение выпускной квалификационной работы проходила на базе компании «Квартплата 24». «Компания 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» [3].

Информация о деятельности компании представлена на рисунке 1.

Рисунок 1 – Экосистема облачных сервисов «Квартплата 24»

«Основной вид деятельности компании по ОКВЭД» [5]:

- Разработка компьютерного программного обеспечения (код по ОКВЭД 62.01);

Дополнительные виды деятельности компании по ОКВЭД:

- 47.7 Торговля розничная прочими товарами в специализированных магазинах;
- 47.91.2 Торговля розничная, осуществляемая непосредственно при помощи информационно-коммуникационной сети Интернет;
- 47.91.3 Торговля розничная через Интернет-аукционы;
- 47.91.4 Торговля розничная, осуществляемая непосредственно при помощи телевидения, радио, телефона;
- 62.02 Деятельность консультативная и работы в области компьютерных технологи.

Организационная структура компании Квартплата 24 представлена на рисунке 2. Численность персонала компании – 50 человек.

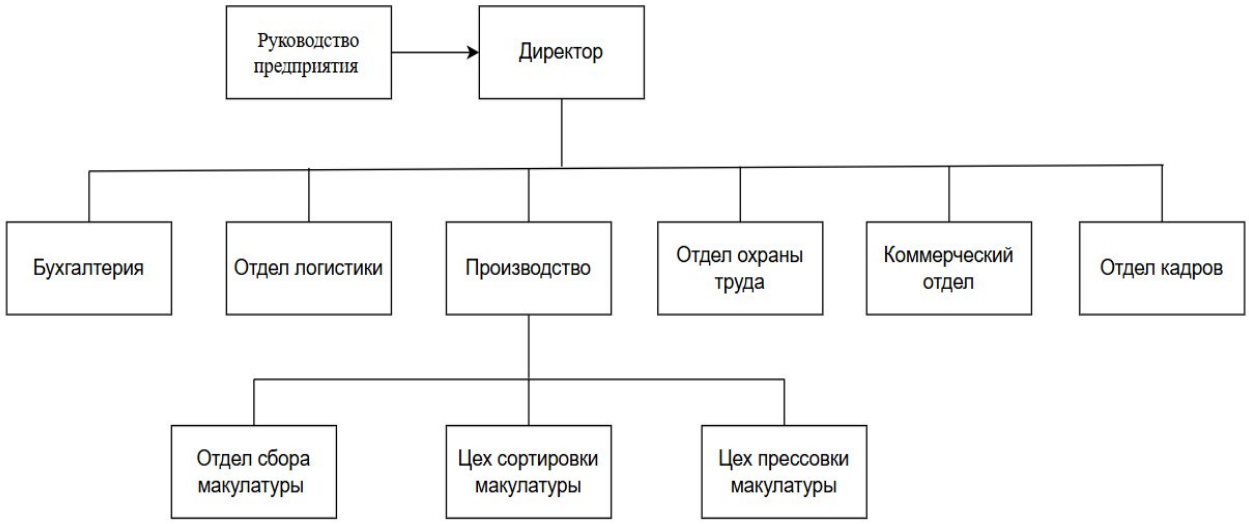

Рисунок 2 - Организационная структура Квартплата 24

В ходе выпускной квалификационной работы рассматривалась работа на производстве, которое есть в компании.

Производственный отдел специализируется на сборе и сортировке

макулатуры для домов, которые обслуживать Квартплата 24.

Его основными функциями являются [18]:

- сбор макулатуры со специализированных контейнеров;
- сбор макулатуры от клиентов и от предприятий;
- сортировка макулатуры;
- прессовка макулатуры для отправки;
- отправка отсортированной макулатуры на предприятия по переработке.

1.2 Описание бизнес-процессов «как есть»

«Сегодня существует две популярные методологии описания функциональной модели организации» [25]:

- «структурный подход к оценке деятельности Компании,
- объектно-ориентированный подход к оценке деятельности Компании» [21].

Структурный подход «основан на принципе алгоритмической декомпозиции. Объектно-ориентированный подход основан на декомпозиции объектов» [16].

В таблице 1 показаны результаты сравнительного анализа двух методологий.

Таблица 1 – Результаты сравнения методологий функционального анализа

Задача по проектированию	Структурный подход	Объектно-ориентированный подход	Методология ARIS
«Описание функциональных задач информационной системы по автоматизации деятельности администратора компании»[3]	«С помощью IDF0 можно лучше понять структуру компании»[3]	« Можно отображать с помощью диаграмм кто выполняет задачу, но не на всех схемах можно отобразить документационный оборот» [1, с.30]	«Предназначена для моделирования экономических процессов, не показывает функциональные задачи» [3, с.78]

Продолжение таблицы 1

Задача по проектированию	Структурный подход	Объектно-ориентированный подход	Методология ARIS
«Определение типов предметов, взаимосвязей между ними и их свойств» [20, с.27]	«С помощью модели IDF1X и DFD можно описать атрибуты сущностей и связи между ними. Подходит для описания реляционных баз данных» [15, с.123]	«UML диаграммы классов. Подходит для описания объектно-ориентированных баз данных» [19, с.38]	«eERM - описывает атрибуты сущностей и связи между ними, подходит для реляционных баз данных» [29, с.77]

Для того, чтобы автоматизировать процесс корпоративной информации нужно описать бизнес-процессы, проходящие на предприятии. Это поможет рассмотреть детально процесс сбора макулатуры, у домов, которые обслуживает компании, и провести аналитику данного процесса для выявления функциональных требований. «Разработка функциональной модели будет осуществлена с помощью методологии моделирования IDEF0.

IDEF0 (Integration Definition for Function Modeling) — это методология функционального моделирования и графическая нотация, разработанная для формализации и описания бизнес-процессов»[6]

IDEF0 [17] используется для создания функциональных блок-схем, которые описывают структуру процесса, его входные и выходные данные, а также потоки данных между блоками. Каждый блок на диаграмме представляет собой функцию, которая может принимать входные данные, выполнять определенные операции и выдавать выходные данные. Связи между блоками представляют собой потоки данных, которые передаются от одного блока к другому.

Диаграмма верхнего уровня (A-0) представлена на рисунке 3, она состоит из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управление и механизмы.

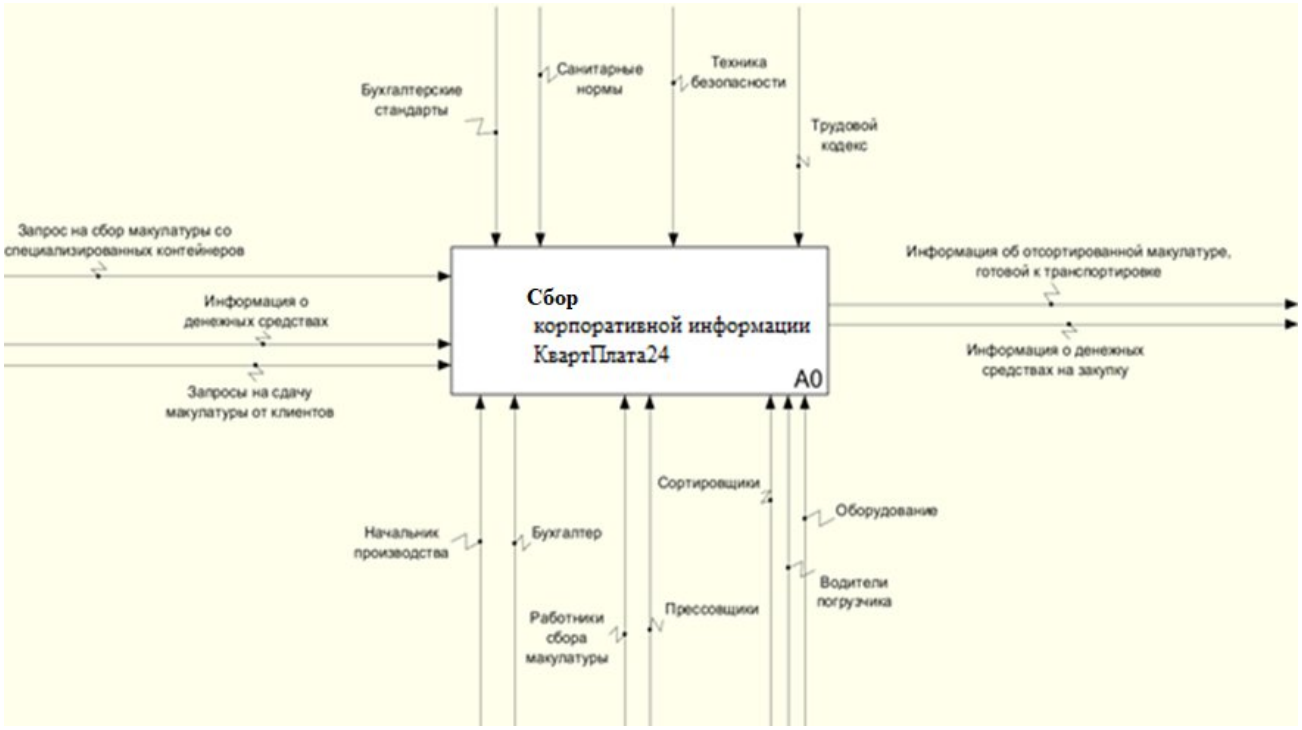

Рисунок 3 - Диаграмма верхнего уровня (А-0). КАК ЕСТЬ

Стрелки на данной диаграмме отображают связи объекта моделирования с окружающей средой. На вход поступают запросы на сдачу макулатуры, информация о денежных средствах, запрос на сбор макулатуры со специализированных контейнеров. А на выход информация об отсортированной макулатуре, готовой к транспортировке и информация о денежных средствах на закупку.

Далее нужно разложить диаграмму А–0 на основные подфункции посредством дочерней диаграммы. Каждая подфункция представляет собой основные процессы, проходящие на предприятии [19] (рисунок 4).

На данной диаграмме показано взаимодействие функциональных блоков между собой, а также отображена входящая и выходящая информация для каждого из них. Теперь можно детальнее рассмотреть отдел сбора макулатуры, определить его основные подфункции (рисунок 5).

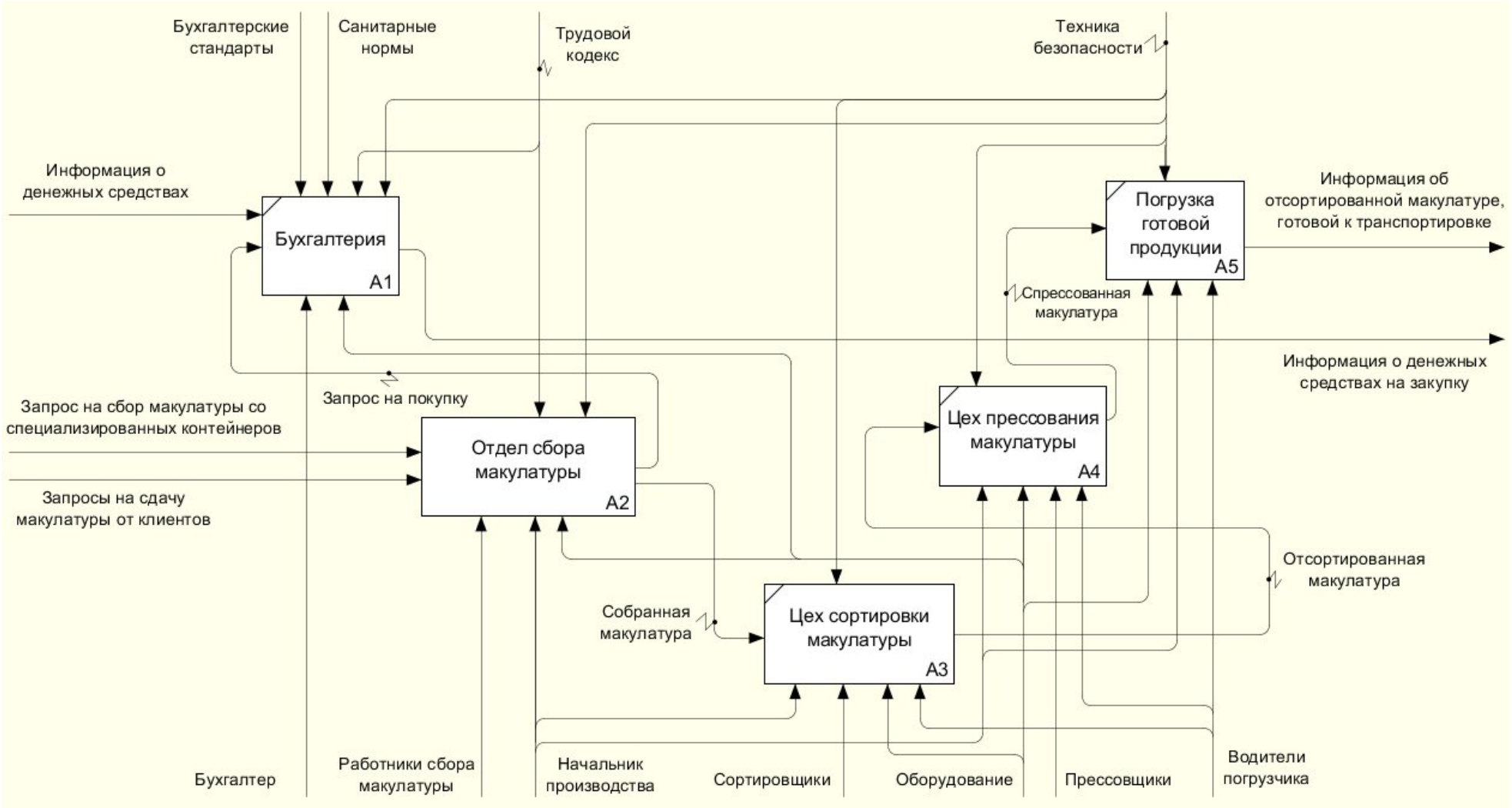

Рисунок 4 – Диаграмма декомпозиции. КАК ЕСТЬ

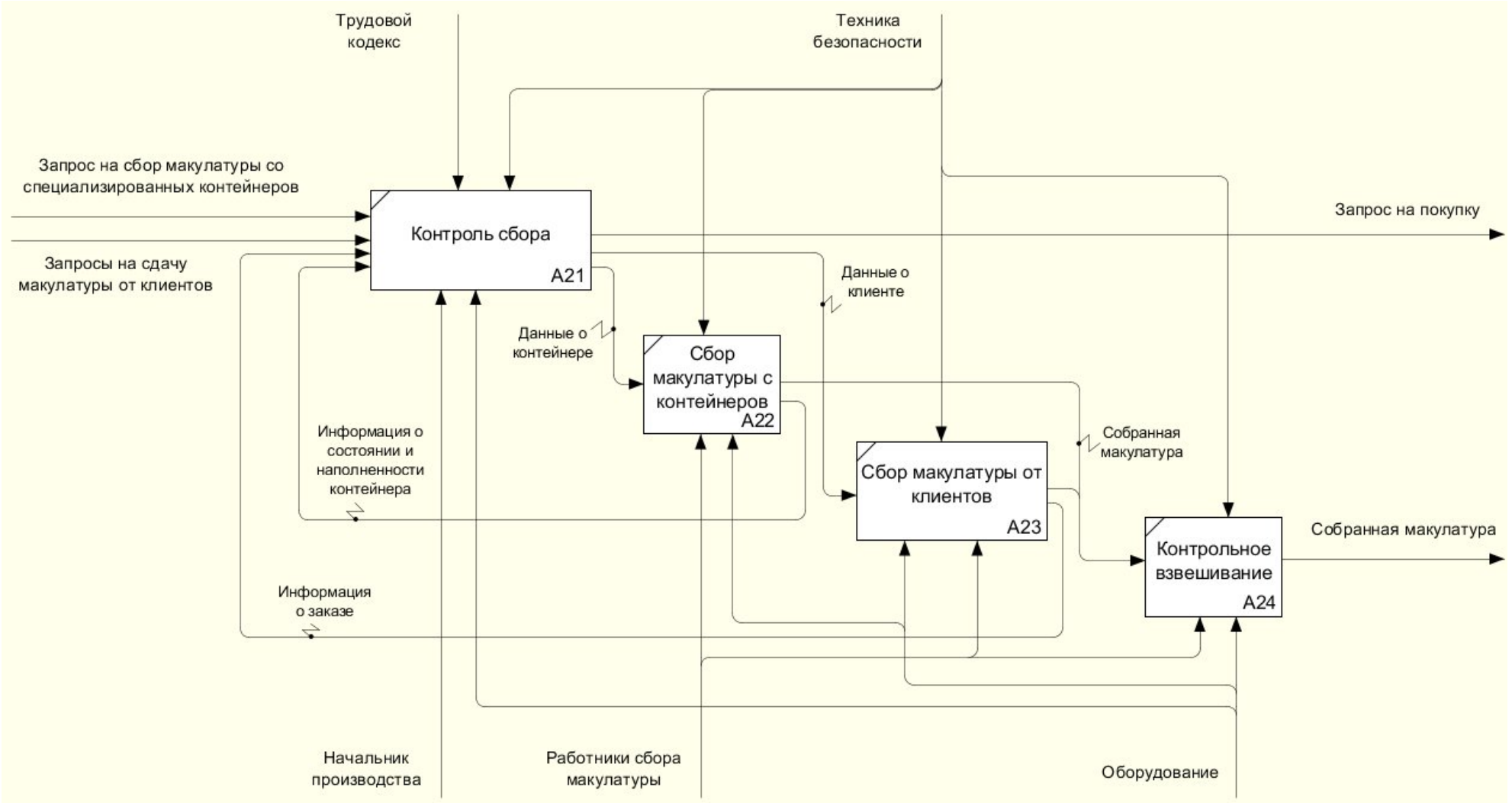

Рисунок 5 – Модель «КАК ЕСТЬ» для отдела сбора макулатуры

На данном этапе стало понятно, как работает бизнес-процесс сбора макулатуры, его основные функции и как они взаимосвязаны между собой.

Контроль сбора: на входе поступают данные о запросах на сдачу макулатуры, далее данные об этих запросах распределяются между работниками сбора макулатуры; также исходит запрос на покупку макулатуры.

Сбор макулатуры с контейнеров: на вход поступают данные о контейнерах, которые необходимо обслужить, а на выход информация о состоянии и наполненности контейнера.

Сбор макулатуры от клиентов: на вход поступают данные о клиентах, которых необходимо обслужить, а на выход информация о заказе.

Контрольное взвешивание: этот этап является завершающим, собранная макулатура взвешивается и переходит дальше в отдел сортировки.

В ходе выпускной квалификационной работы большое внимание уделено процессу аналитической и статистической обработки информации.

Данный процесс планируется автоматизировать для ускорения процесса получения различных отчетов и графических данных по результатам сбора информации из различных источников.

На данном этапе стало понятно, как работает бизнес-процесс, его основные функции и как они взаимосвязаны между собой.

Но выявлены проблемные места, которые необходимо контролировать, требующие сбора информации, а именно отсутствует контроль сбора, где: на входе поступают данные о запросах на сдачу макулатуры, далее данные об этих запросах распределяются между работниками сбора макулатуры; также исходит запрос на покупку макулатуры.

Сбор макулатуры с контейнеров: на вход поступают данные о контейнерах, которые необходимо обслужить, а на выход информация о состоянии и наполненности контейнера.

Сбор макулатуры от клиентов: на вход поступают данные о клиентах, которых необходимо обслужить, а на выход информация о заказе.

Контрольное взвешивание: этот этап является завершающим, собранная макулатура взвешивается и переходит дальше в отдел сортировки.

Т.е. необходимо разделить процессы для пользователей и иметь возможность контролировать и анализировать, т.е для этого требуется клиент-серверной приложение с мобильной версией.

Задачи, которые необходимо решить для достижения цели, можно разделить на три модуля:

- задачи исследования;
- задачи для разработки клиентской части;
- задачи для разработки серверной части системы.

Задачи исследования:

- изучение предметной области, определение потребностей предприятия;
- проведение сравнительного анализа существующих аналогов, в целях выявления преимущества и недостатков;
- определение функциональных и нефункциональных требований, на базе которых разрабатывать приложение;
- определение инструментов для разработки клиент-серверного приложения;
- составление ключевых этапов программной реализации.

После определения требований к разработке можно разработать диаграмму IDEF0, которая поможет отобразить, как будет выглядеть автоматизированный процесс сбора макулатуры.

Диаграмма бизнес-процесса «КАК ДОЛЖНО БЫТЬ» представлена на рисунке 6.

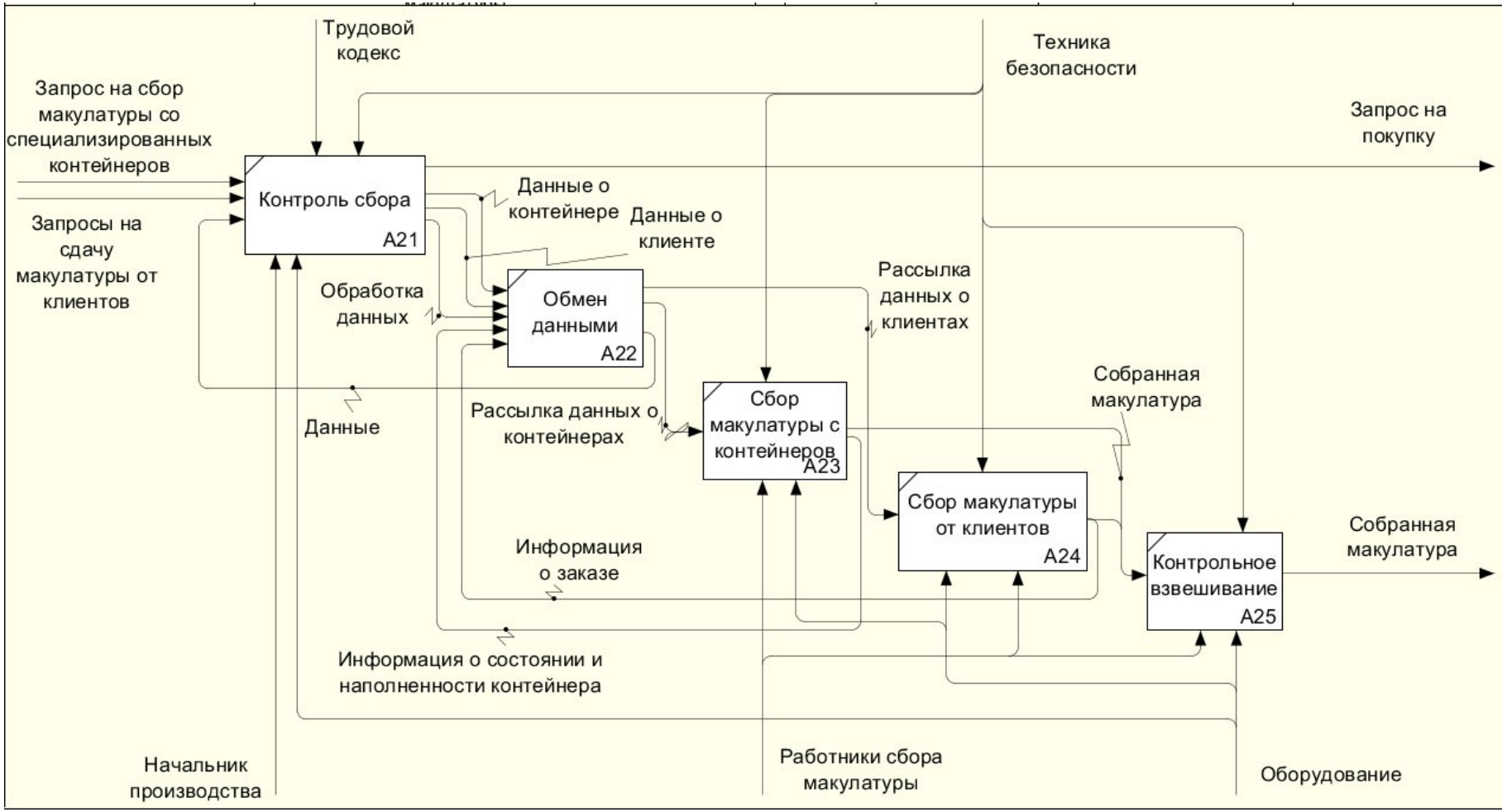

Рисунок 6 – Модель «КАК ДОЛЖНО БЫТЬ»

Задачи для разработки клиентской части приложения:

- разработка дизайна приложения, обеспечивающего удобное и простое использование;
- разработка функционала аутентификации работников предприятия;
- разработка функционала формы о состоянии и наполненности специализированного контейнера;
- разработка функционала приема заявок на сдачу макулатуры от клиентов.

Задачи, связанные с разработкой серверной части и базы данных:

- реализация протокола HTTP для связи между клиентом и сервером;
- создание базы данных для хранения информации о работниках, контейнерах и заявок от клиентов;
- разработка графического интерфейса для связи между пользователем и системными объектами;
- разработка функционала распределения контейнеров между работниками;
- разработка функционала отправки заявок на сдачу макулатуры работниками;
- разработка функционала для генерации отчетов из БД.

Серверная часть, разрабатываемая для начальника предприятия, является десктопным приложением с простым интерфейсом. Основными функциональными требованиями для серверной части будут являться:

- добавление и удаление работников из базы данных;
- распределение контейнеров между работниками предприятия;
- рассылка поступающих заказов на сдачу макулатуры;
- автоматическое занесение данных о специализированных контейнерах в БД;
- автоматическое занесение данных о принятых заказах в БД;
- формирование отчетности предприятия из базы данных.

Клиентская часть, разрабатываемая для работников предприятия, которые обслуживают контейнеры и принимают заказы на сдачу макулатуры, будет является мобильным приложением. Основными функциональными требованиями для клиентской части приложения будут являться:

- отметка специализированного контейнера, который уже опустошили;
- внесение данных о контейнере, его заполненность и состояние;
- прием поступающих заказов на сдачу макулатуры от клиентов и предприятий.

В процесс сбора макулатуры добавлена функция – обмен данными. Теперь весь обмен данными будет происходить автоматизировано и данные будут сохраняться автоматически.

1.3 Анализ существующих разработок для решения задачи по автоматизации процесса тестирования

Программные приложения, которые есть на рынке можно поделить на: платные и бесплатные программы, а также облачные сервисы для ЖКХ, ТСЖ, ОСМД. Рассмотрим наиболее востребованные программные продукты.

«Госуслуги.Дом» (рисунок 7)

«В России создали единое приложение для управления расходами ЖКХ Госуслуги.Дом. Программа уже запущено в пилотном режиме в пяти регионах (в Белгородской, Калужской, Омской и Челябинской областях и Ханты-Мансийском автономном округе), а в третьем квартале 2023 года ожидается ее запуск по всей РФ. Через приложение пользователи смогут передавать показания счетчиков и видеть их статистику, оплачивать коммунальные услуги, видеть свои счета за ЖКХ и подавать заявки в управляющую компанию» [1].

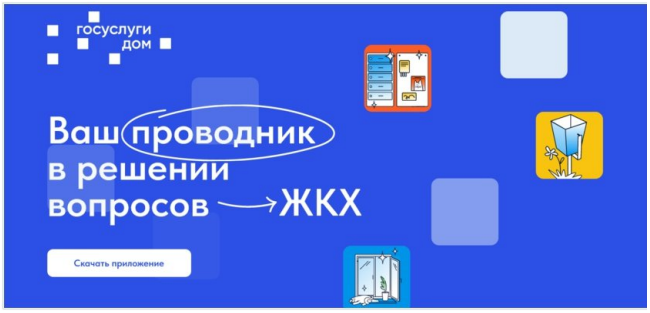

Рисунок 7 - Госуслуги.Дом

Диспетчер 24 (рисунок 8)

«Компания "Удобные решения” разработала решение для автоматизации работы управляющих компаний, ТСЖ и коммерческой недвижимости на основании продуктов: платформа Диспетчер 24, на которой построен контактный центр, и онлайн-сервис для коммуникации с жителями. Плюсы решения для управляющих компаний, которые хотят автоматизировать свою работу - это экономия на сотрудниках, оптимизация затрат на организацию АДС для УК и ТСЖ, повышение лояльности собственников жилья, контроль исполнителей, работа с должниками, проведение собраний собственников. Клиенты оплачивают счета в один клик, получают уведомления о работах в доме и необходимости подать показания счетчиков» [2].

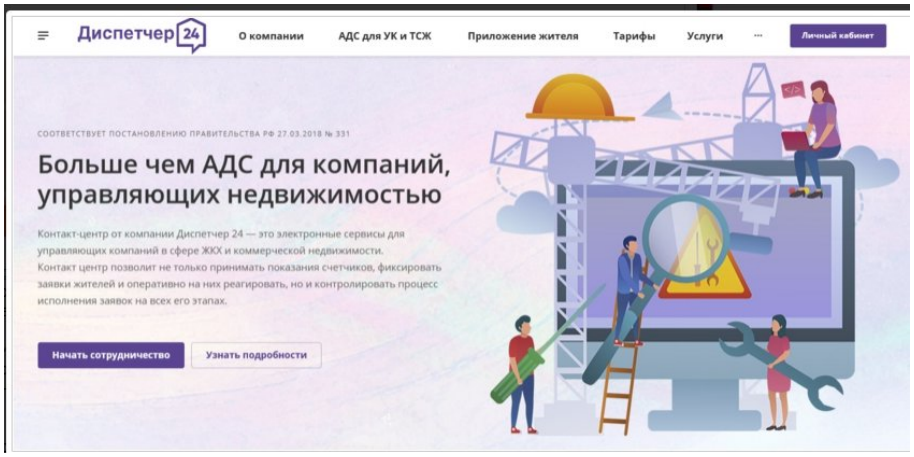

Рисунок 8 - Диспетчер 24

Doma.ai (рисунок 9)

«Сервис Doma.ai. позволяет управляющей компании отслеживать уровень довольства жителей услугами, а жителям — оплачивать «коммуналку» и писать обращения. Есть диспетчер задач, календарь и чат для сотрудников, финансовые отчеты» [7].

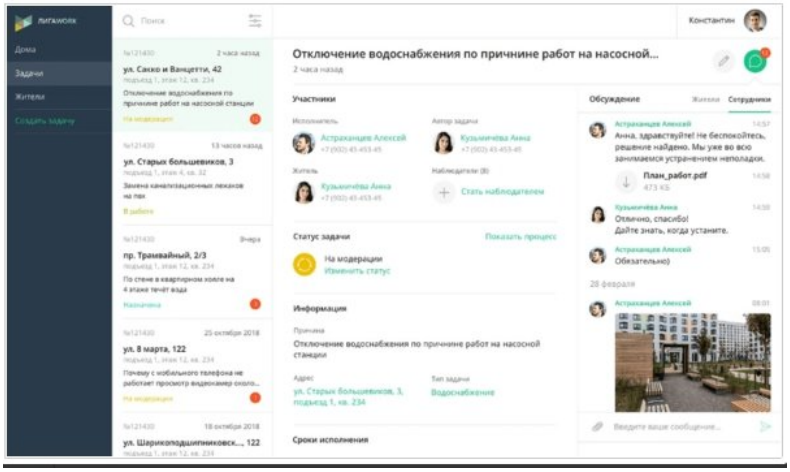

Рисунок 9 - Doma.ai

«Для выбранных программных продуктов разработаем критерии, по которым будет проводиться оценка соответствия ожиданиям от разработки. Критериями выбора программного продукта будут являться:

- простота и удобство,
- безопасность,
- удобство навигации,
- юзабилити,
- функциональность с точки зрения возможности поддержки работы по сбору макулатуры и внесения данных о состоянии контейнеров их наполненности, анализа информации для формирования графика вывоза макулатуры» [8].

Выбранные системы будут оценены по выделенным критериям по шкале, где «1» -полное несоответствие критерию, а «5» - полное соответствие. Оценка программных продуктов приведена в таблице 2.

Таблица 2- Анализ систем, представленных на рынке

Критерий	Госуслуги.Дом	Диспетчер 24	Doma.ai
Простота управления	4	4	4
Безопасность	4	5	3
Удобство навигации	3	3	4
Юзабилити	3	4	3
Функциональность, которая отражает сбор макулатуры	2	1	3
Итого	16	17	17

В результате сделан вывод, что программное обеспечение, представленное на рынке, не соответствует поставленной задаче.

1.4 Формирование бизнес-целей и требований системы для корпоративной информации

Определим требования к информационной системе для корпоративной информации в методологии FURPS+ [4] и отобразим в таблице 3.

Таблица 3 – Требования к информационной системе для корпоративной информации в методологии FURPS+

Вид требований	Содержание требований
«F. Функциональные требования»[3]	− «возможность загружать данные из различных источников, − ввод данных, − поиск данных»[3]
«U. Требования к удобству использования	− эстетика пользовательского интерфейс, − справочная информация
R. Требования надежности »[3]	− верификация данных
«P. Требования к производительности информационной системы»[3]	− «потребление ресурсов, определяется требованиями к персональному компьютеру сотрудника» [9]
«S. Поддерживаемость »[3]	− «тиражирование в филиалах компании или в отрасли, − соответствие международным стандартам»[3]
«+. Ограничения »[3]	− «хранение необходимо реализовать с помощью реляционной БД, − форматы данных входных и выходных документов, должны поддерживать стандарты ЖКХ» [10]

После определение требования требований к разрабатываемой системе переходим к анализу существующих разработок и возможностью их использования для поставленных задач.

Выводы по главе 1

В первой главе выпускной квалификационной работы рассмотрены общие вопросы бизнес-процессов работы компании, которые необходимо автоматизировать. Определена потребность в разработке системы для корпоративной информации.

Проведен сравнительный анализ существующих решений.

Сформулированы требования к функциональности, надежности и удобстве использования разрабатываемой системы по автоматизации тестирования для программного обеспечения.

Глава 2 Концептуальное моделирование информационной системы для корпоративной информации

2.1 Классы и формализация пользователей информационной системы для корпоративной информации

В ходе проектирования информационной системы определено, что информационные потоки поступающих в компанию очень много. Для выпускной работы выделен процесс компании, который отвечает за сбор и анализ информации, которая поступает по сбору макулатуры на контейнерный площадках домов, за которые отвечает компания «Квартплата 24».

В разработке логических моделей будет использована технология моделирования UML [20].

«UML (Unified Modeling Language) — это язык, который позволяет описывать архитектуру и поведение системы в виде диаграмм. UML является стандартом индустрии и широко используется в проектировании различных типов систем.

«Диаграмма вариантов использования (прецедентов) — это графическое представление функциональности системы, основанное на взаимодействии актеров (пользователей) и системы. Диаграмма прецедентов позволяет описать, какой функционал разрабатываемой программной системы доступен каждой группе пользователей»[11]. Диаграмма вариантов использования для клиент-серверного приложения для компании «Квартплата 24» представлена на рисунке 10.

Рисунок 10 – UML диаграмма вариантов использования клиент-серверного приложения

Основные пользователи системы: начальник и работник.

Сценарии для начальника:

- авторизация ;
- добавление работников сбора макулатуры в БД.

Рассылка поступающих заказов на сдачу макулатуры: начальник производства переходит на страницу рассылки заказов, заполняет необходимые поля, такие как дата, ФИО заказчика и адрес.

Далее нажимает на кнопку «Отправить заявку».

Система проверяет корректность введенных данных и рассылает данные на мобильные устройства работников сбора макулатуры.

Распределения контейнеров между работниками предприятия: Начальник производства заходит на страницу распределения контейнеров, выбирает нужного работника и выбирает нужный контейнер.

Далее нажимает на кнопку «Добавить».

Система обновляет данные в БД [21].

Учет состояния контейнеров: Начальник производства заходит на страницу учета состояния контейнеров. На данной странице выводится список всех контейнеров. Можно отфильтровать данные по ФИО и по дате.

Учет состояния заявок на сдачу макулатуры: Начальник производства заходит на страницу учета состояния заявок на сдачу макулатуры. На данной странице отображаются все заявки с их статусом. Можно отфильтровать данные по ФИО и по дате.

Сценарии для работников сбора макулатуры: Авторизация - работник сбора макулатуры заходит в приложение, после чего вводит логин и пароль.

Система проверяет корректность введенных данных и осуществляется переход на главную страницу.

Прием заявок на сдачу макулатуры: Работник сбора макулатуры переходит на страницу заявок и там выбирает список доступных заявок.

Далее нажимает кнопку «Принять» на выбранной заявке.

Внесение данных о контейнере, его состояние и наполненность: Работник сбора макулатуры переходит на страницу контейнеров, выбирает контейнер и заполняет необходимые поля, такие как дата, наполненность контейнера, состояние, комментарий.

Далее нажимает кнопку «Отправить».

2.2 Описание функциональных требований информационной системы для корпоративной информации

Для разработки алгоритма работы системы будет использована диаграмма последовательности. Данная диаграмма позволит описать

последовательность взаимодействия программных компонентов между собой при совершении действий пользователя системы.

Диаграмма последовательности для начальника предприятия представлена на рисунке 11.

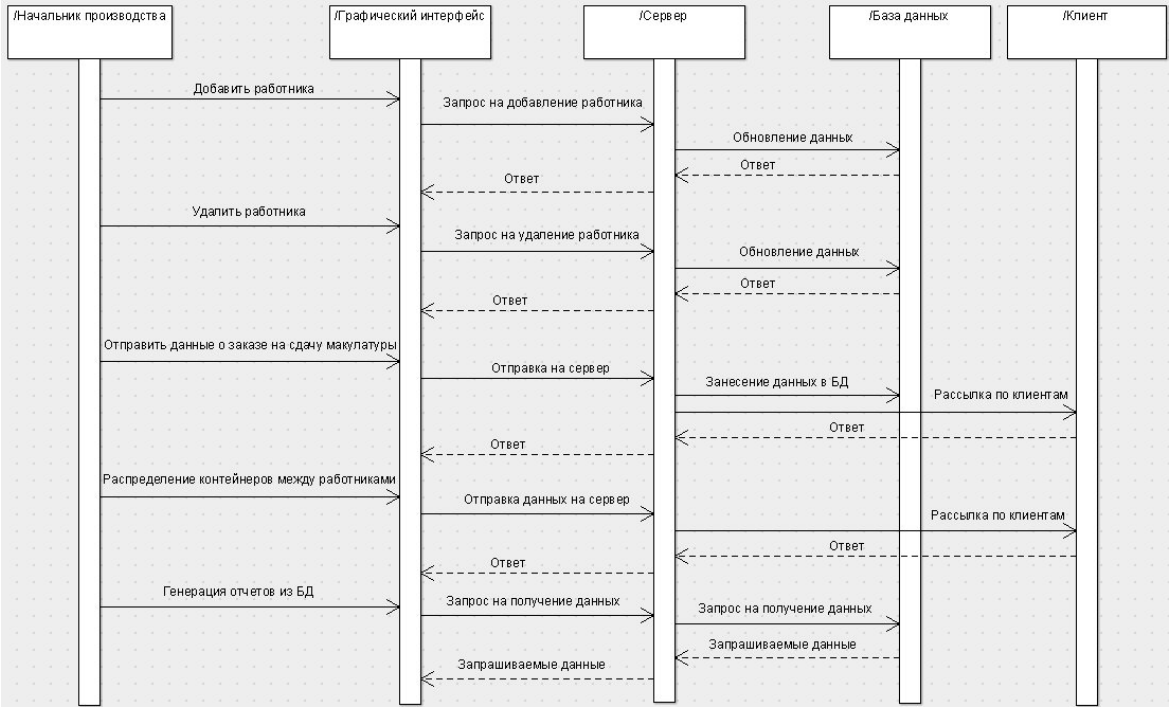

Рисунок 11 – UML диаграмма последовательности для серверной части

Алгоритм работы серверной части приложения для предприятия по сбору макулатуры заключается в том, что начальник производства взаимодействует с приложением через графический интерфейс. И в зависимости от выполняемых действий, будет обрабатываться запрос и данные будут заноситься в базу данных. Также он может отправить данные на клиентскую часть приложения.

Диаграмма последовательности для работников сбора макулатуры представлена на рисунке 12.

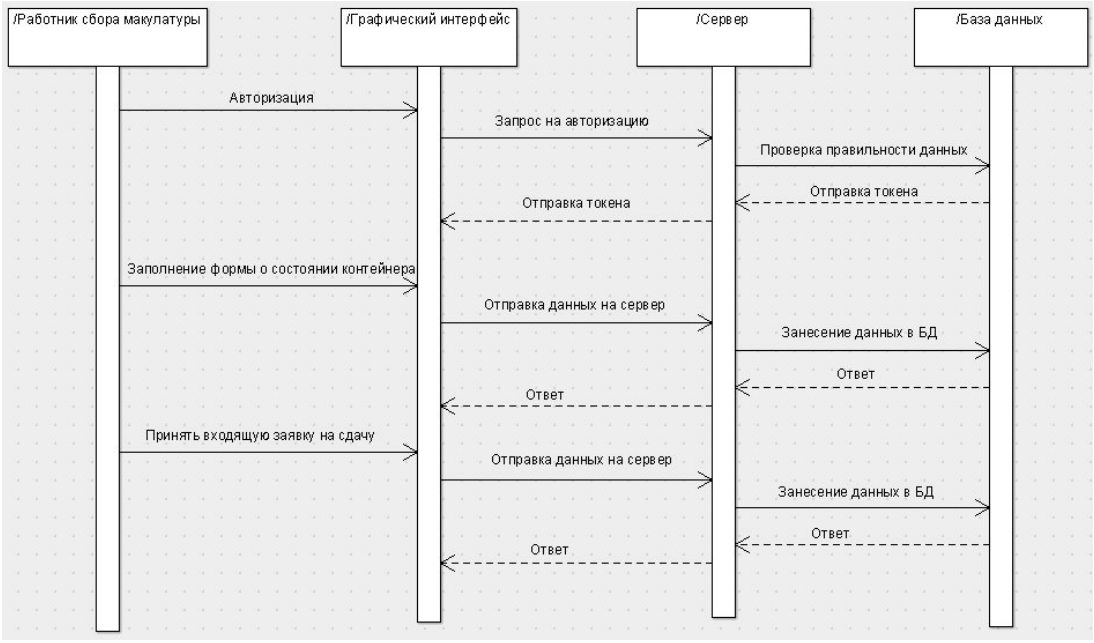

Рисунок 12 – UML диаграмма последовательности для клиентской части

Алгоритм работы клиентской части приложения заключается в том, что работник сбора макулатуры взаимодействует с приложением через графический интерфейс. И в зависимости от выполняемых действий, будет обрабатываться запрос и данные будут заноситься в базу данных. Также он может отправить данные на серверную часть приложения.

После выделения объектов системы и определения основных функций и сценариев работы можно спроектировать диаграмму классов.

Диаграмма классов — это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их кооперaций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

Диаграмма классов серверной части приложения для предприятия по сбору макулатуры представлена на рисунке 13.

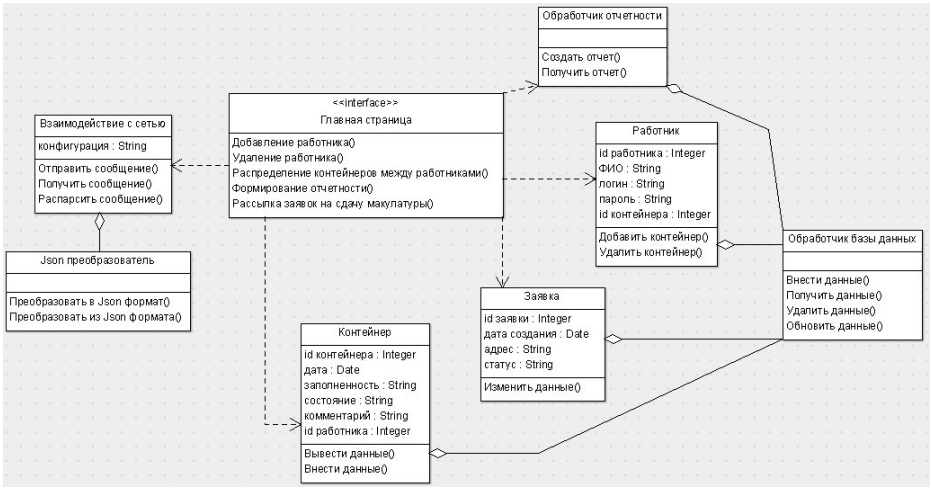

Рисунок 13 –UML диаграмма классов серверной части

Диаграмма классов клиентской части приложения для предприятия по сбору макулатуры представлена на рисунке 14.

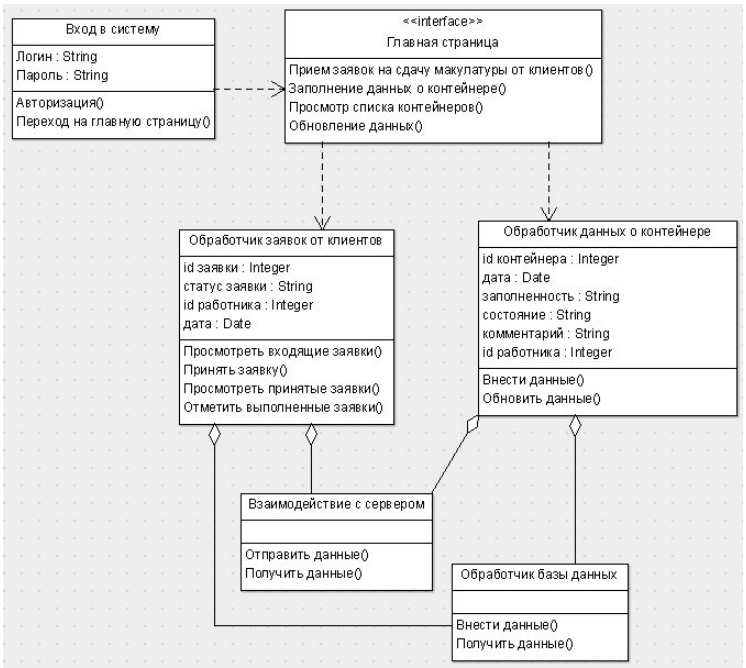

Рисунок 14 – UML диаграмма классов для клиентской части

Диаграмма классов показывает классы, атрибуты и методы проектируемого клиент-серверного приложения.

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Диаграмма развертывания для проектируемой системы представлена на рисунке 15.

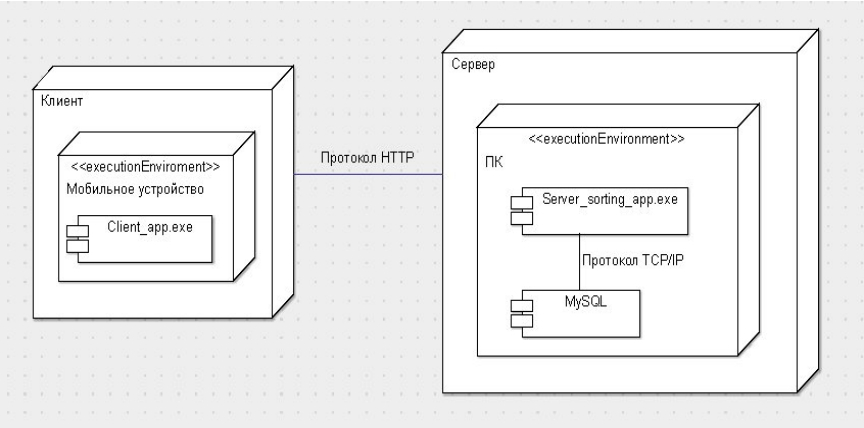

Рисунок 15 – UML диаграмма развертывания клиент-серверного приложения

Данная диаграмма развертывания позволяет понять, как физически должна быть развернута система на аппаратном обеспечении. Клиент и сервер, представленные в виде куба, являются физическими сущностями. Они связаны между собой протоколом HTTP. Клиент — это мобильное устройство, на котором будет развернута клиентская часть приложения для предприятия по сбору макулатуры. Сервер – это персональный компьютер, на котором будет развернута серверная часть приложения, связанная с СУБД посредством протокола TCP/IP.

Выводы по главе 2

Во второй главе были рассмотрены вопросы концептуального моделирования системы для корпоративной информации. На функциональной схеме выделены пользователи системы и основные функции. На диаграмме последовательности определен порядок ввода и вывода данных.

Глава 3 Проектирование системы для корпоративной информации и особенности реализации

3.1 Системная архитектура информационной системы для корпоративной информации

Рассмотрим технологии и средства разработки, которые будут использованы для реализации серверной и клиентской частей приложения.

Чтобы выбрать язык программирования для написания десктопного приложения нужно провести сравнение языков. В сравнение попали языки – Python, Java, C#.

Ниже представлена сравнительная таблица. (Таблица 4)

Таблица 4 – Сравнение языков программирования

Сравниваемые характеристики	Языки программирования		
	Python	C#	Java
Поддержка операционных систем	Поддерживает Windows, macOS и Linux	Поддерживает только Windows	Поддерживает Windows, macOS и Linux
Удобство разработки	Имеет простой и понятный синтаксис	Имеет хорошую интеграцию с Microsoft технологиями	Обладает хорошим инструментарием для разработки на разных платформах
Производительность	Имеет низкую производительность из-за динамической типизации	Имеет высокую производительность	Обладает хорошей производительностью и масштабируемостью

Продолжение таблицы 4

Сравниваемые характеристики	Языки программирования		
	Python	C#	Java
Библиотеки и фреймворки	Имеет много библиотек для разработки десктопных приложений, но их качество может варьироваться	Имеет множество библиотек и фреймворков, связанных с платформой .NET	Имеет обширный набор библиотек и фреймворков для разработки десктопных приложений

В результате сравнения был выбран язык программирования Java.

«Среда разработки IntelliJ IDEA обладает удобным и понятным пользовательским интерфейсом, что делает работу с ней комфортной для разработчиков» [1].

Инструменты автоматического анализа кода, интегрированный отладчик, поддержка автоматической сборки проекта, возможность подключения плагинов и расширений — все это делает IntelliJ IDEA мощным инструментом для разработки программного обеспечения.

Для разработки графического интерфейса можно использовать инструмент JavaFX. Данная библиотека предоставляет инструментарий для создания кроссплатформенных графических приложений.

Для разработки базы данных была выбрана СУБД MySQL.

MySQL — это одна из систем управления базами данных (Databases Management System) или просто СУБД. Простыми словами, программа, с помощью которой создают и контролируют БД. Она предназначена для работы с базами реляционного типа и использует для взаимодействия с ними язык стандартизированных запросов (SQL).

MySQL предлагает пользователям высокий уровень безопасности. В MySQL есть встроенные инструменты безопасности, которые поддерживают управление пользователями и их привилегиями. При недостатке стандартных инструментов пользователь всегда может установить дополнительные плагины.

Перейдем к выбору стека технологий для клиентской части приложения.

Для нее также был выбран язык программирования Java. Разработку мобильного приложения под Android лучше всего осуществлять в Android Studio.

Android Studio – это интегрированная среда разработки (integrated development environment) от Google для создания Android-приложений. В ней можно писать код, проектировать графический интерфейс, проводить отладку и сборку приложений.

Android Studio предлагает широкий набор инструментов и функций, упрощающих процесс разработки приложений. В его состав входит редактор кода с функциями автозаполнения, отладчик, инструменты для разработки пользовательского интерфейса, система сборки приложений и многие другие полезные функции. Он также обеспечивает интеграцию с Android SDK, позволяя разработчикам создавать, тестировать и отлаживать приложения непосредственно в среде разработки.

Использованные технологии ВКР:

- Java;
- IntelliJ IDEA;
- JavaFX;
- MySQL.
- Java;
- Android Studio.

В целом, выбранный стек технологий предоставляет хорошие возможности для создания клиент-серверного приложения на Java, но

требует от разработчика определенного уровня знаний и опыта работы с этими технологиями.

3.2 Информационная модель и ее описание

Логическая модель базы данных клиент-серверного приложения для предприятия по сбору макулатуры представлена на рисунке 16.

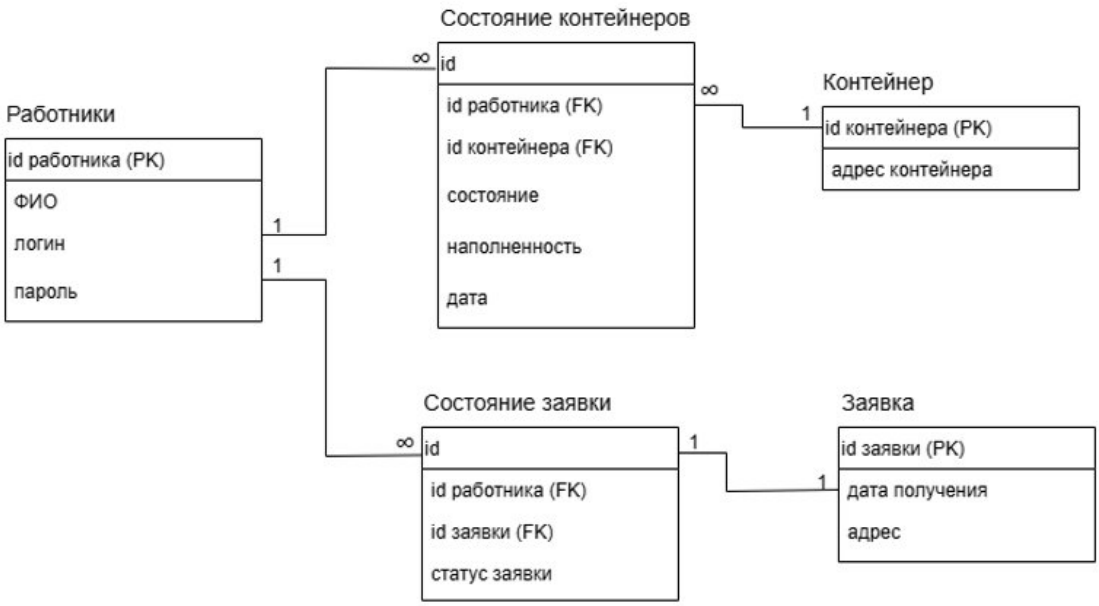

Рисунок 16 – Логическая модель базы данных для клиент-серверного приложения

Логическая модель базы данных не привязана к конкретной СУБД.

Определены основные таблицы, которые будут использоваться в разрабатываемом клиент-серверном приложении:

- таблица «Работник», представленная в таблице 5, в которой будут храниться данные, необходимые для входа работников в систему;
- таблица «Контейнер», представленная в таблице 6, будет хранить в себе информацию о контейнере;

- таблица «Заявки», представленная в таблице 7, будет хранить в себе информацию о поступающих заявках;
- таблица «Состояние контейнеров», представленная в таблице 8, будет хранить в себе данные, поступающие от работников сбора макулатуры о состоянии контейнера;
- таблица «Состояние заявки», представленная в таблице 9, будет хранить в себе данные, поступающие от работников сбора макулатуры о статусе заявки.

Таблица 5 – Атрибуты сущности «Работник»

Поле	Тип данных	Ключ
id работника	int	PK
ФИО	varchar	-
логин	varchar	-
пароль	varchar	-

Таблица 6 – Атрибуты сущности «Контейнер»

Поле	Тип данных	Ключ
id контейнера	int	PK
адрес контейнера	varchar	-

Таблица 7 – Атрибуты сущности «Заявки»

Поле	Тип данных	Ключ
id заявки	Int	PK
ФИО	Varchar	-
дата получения	Varchar	-
адрес	Varchar	-

Таблица 8 – Атрибуты сущности «Состояние контейнеров»

Поле	Тип данных	Ключ
id	Int	PK
id работника	Int	FK
id контейнера	Int	FK
состояние	Varchar	-
наполненность	Varchar	-
дата	Date	-

Таблица 9 – Атрибуты сущности «Состояние заявки»

Поле	Тип данных	Ключ
id	Int	PK
id работника	Varchar	FK
id заявки	Varchar	FK
статус заявки	Varchar	-

Разработка базы данных осуществлена с помощью языка SQL и СУБД MySQL. Изначально было создано соединение с сервером базы данных (рисунок 17).

Рисунок 17 – Соединение с сервером БД

После этого была создана база данных, в нее были добавлены данные таблицы:

- workers;
- container;
- container_status;
- status_request;
- reguest.

Иерархия структуры базы данных представлена на рисунке 18.

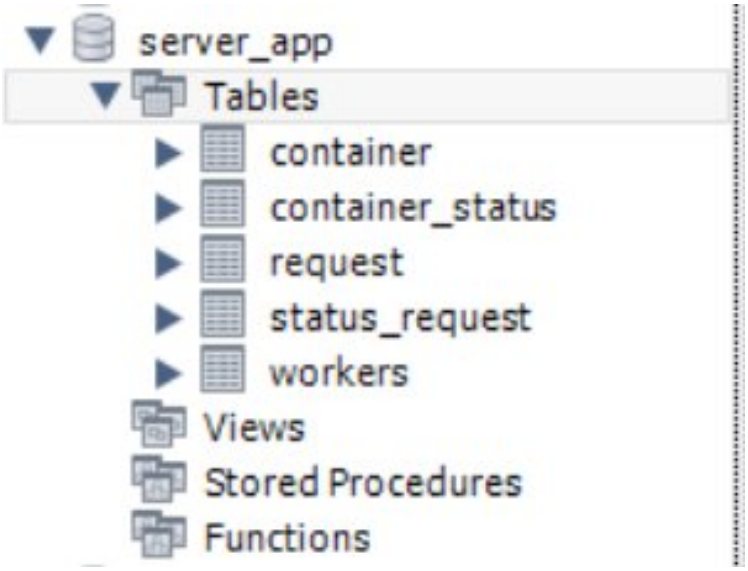

Рисунок 18 – Иерархия структуры БД

Далее были заполнены таблицы, которые хранят сведения о конкретных объектах. Таблицы состоят из записей и полей, а поля в свою очередь содержат различные типы данных.

Таблица «workers» представлена на рисунке 19.

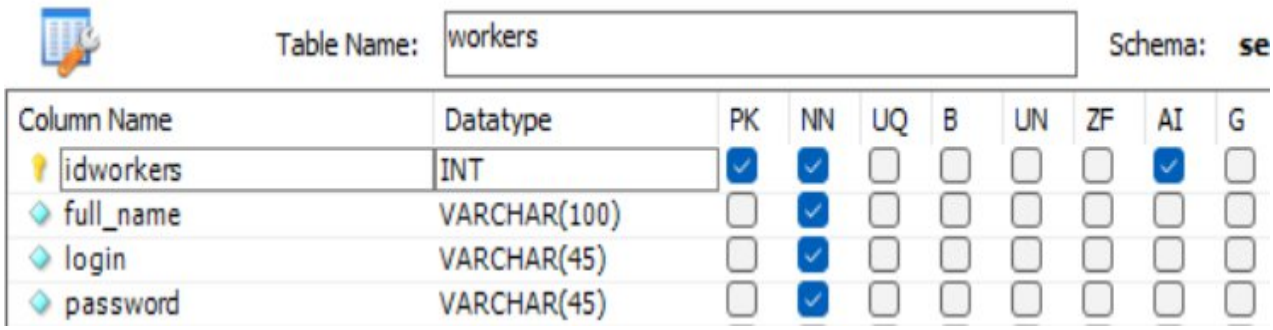

Рисунок 19 – Таблица «workers»

Данная таблица хранит сведения о работниках сбора макулатуры, такие как ФИО, логин и пароль.

На рисунке 20 представлена таблица «container».

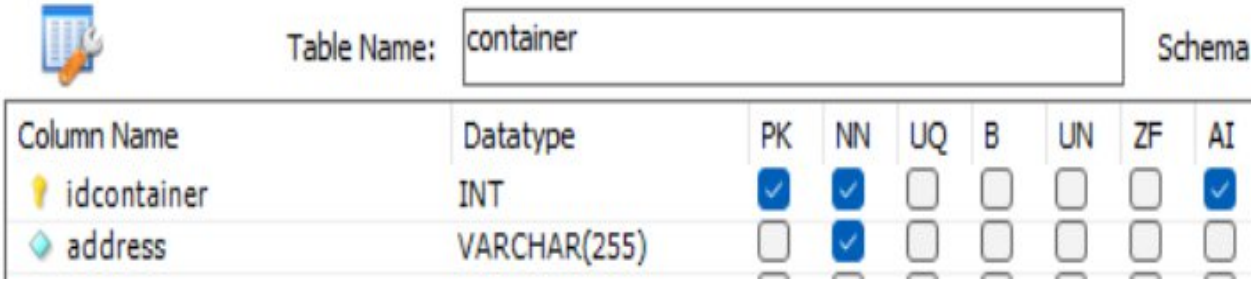

Рисунок 20 –Таблица «container»

Данная таблица хранит в себе сведения о специализированных контейнерах для сбора макулатуры, его id и адрес.

На рисунке 21 представлена таблица «container_status».

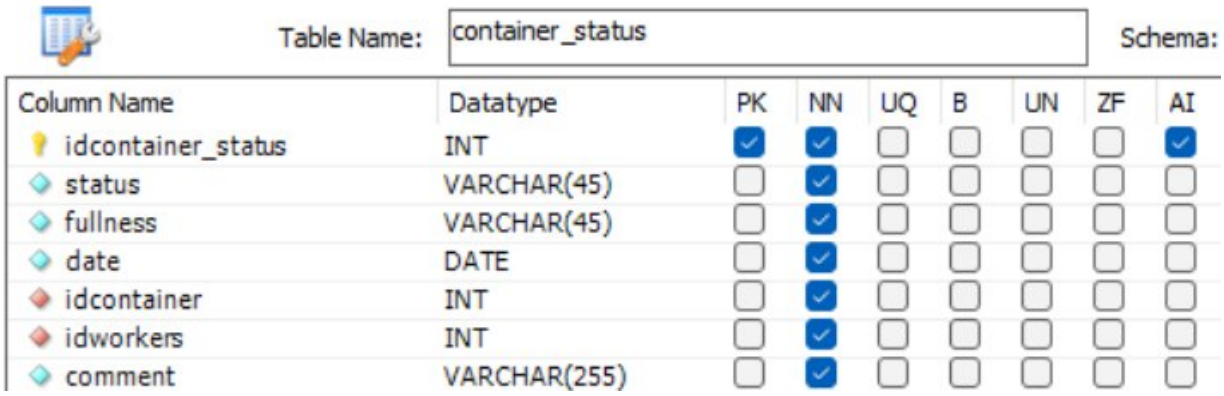

Рисунок 21 – Таблица «container_status»

Данная таблица хранит в себе сведения о состоянии специализированных контейнеров для сбора макулатуры, а также сведения о работнике, который его обслуживает.

На рисунке 22 представлена таблица «status_request».

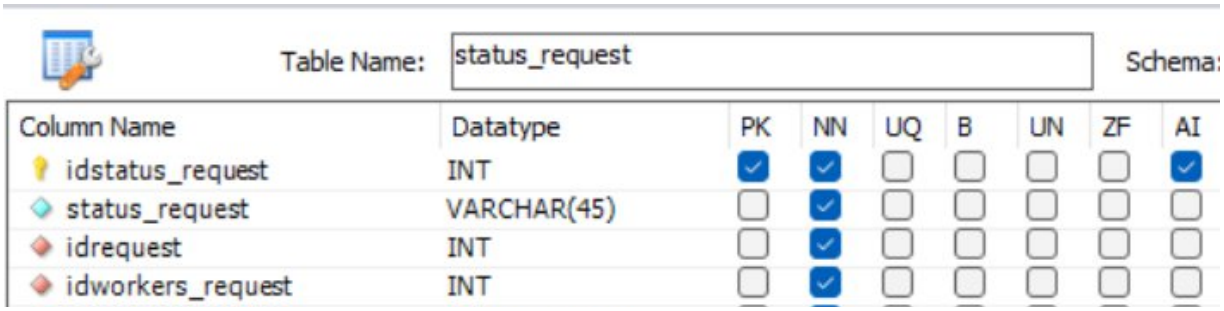

Рисунок 22 – Таблица «status_request»

Данная таблица хранит в себе сведения о статусе заявки на сдачу макулатуры, а также сведения о работнике, который принял заявку.

На рисунке 23 представлена таблица «reguest».

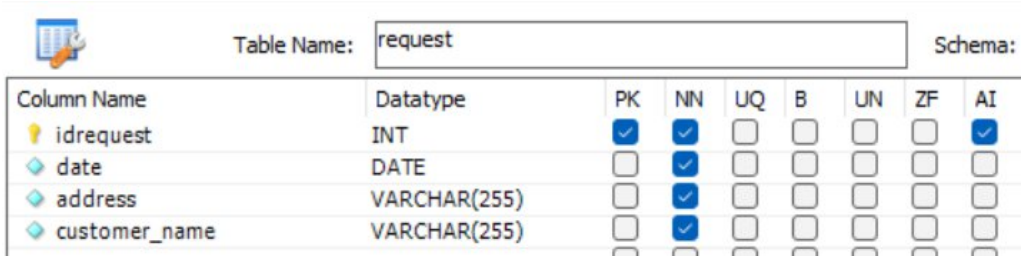

Рисунок 23 – Таблица «reguest»

Данная таблица хранит в себе сведения о заявках на сдачу макулатуры от клиентов, а также дату оформления заявки, ФИО клиента и адрес.

Чтобы показать, как разные «сущности» связаны между собой внутри системы, была использована EER диаграмма. Разработанная база данный клиент-серверного приложения для предприятия по сбору макулатуры представлена на рисунке 24.

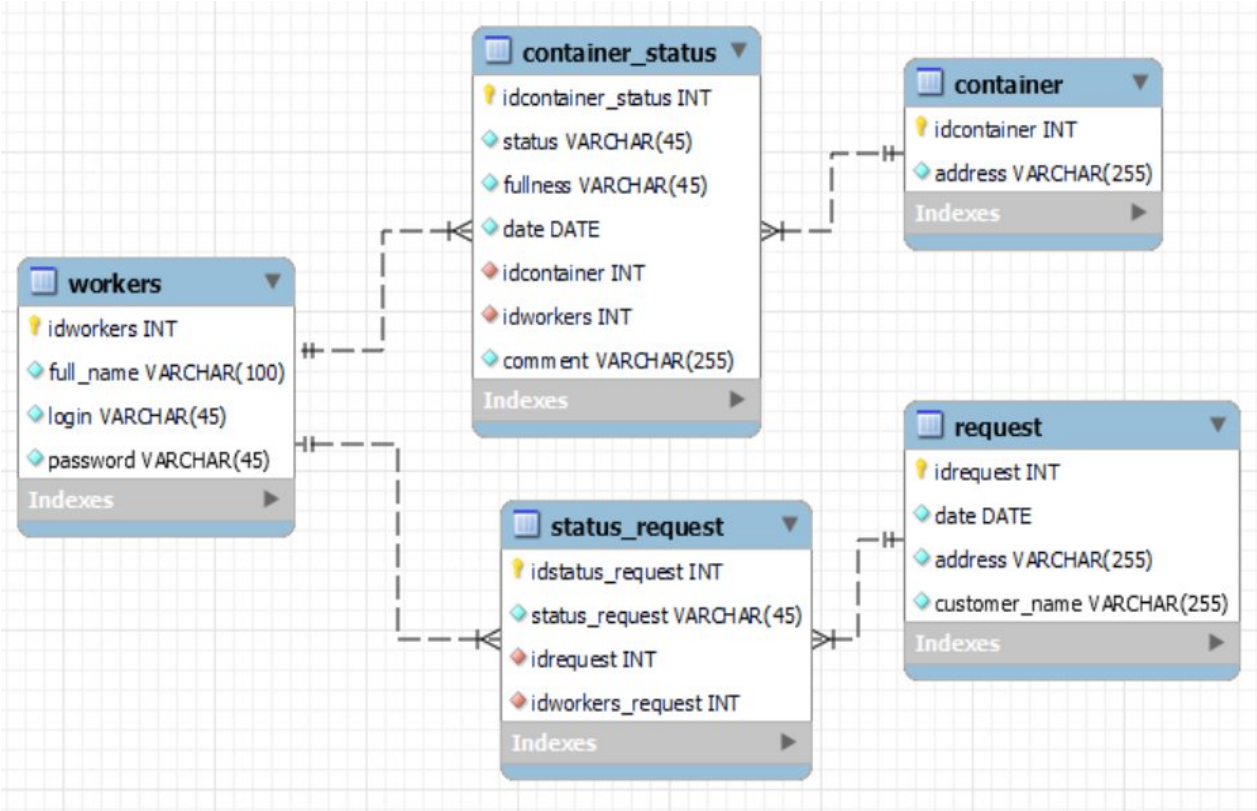

Рисунок 24 – Физическая схема базы данных

После проектирования базы данных, приступим к реализации функционала и пользовательского интерфейса

3.3 Разработка информационной системы для корпоративной информации

Реализация протокола HTTP для связи клиента с сервером

Протокол HTTP (HyperText Transfer Protocol) – это основной протокол передачи данных в интернете. Он будет использоваться для обмена информацией между клиентской и серверной частью приложения.

Для реализации данного протокола разработаны классы, представленные на рисунке 25.

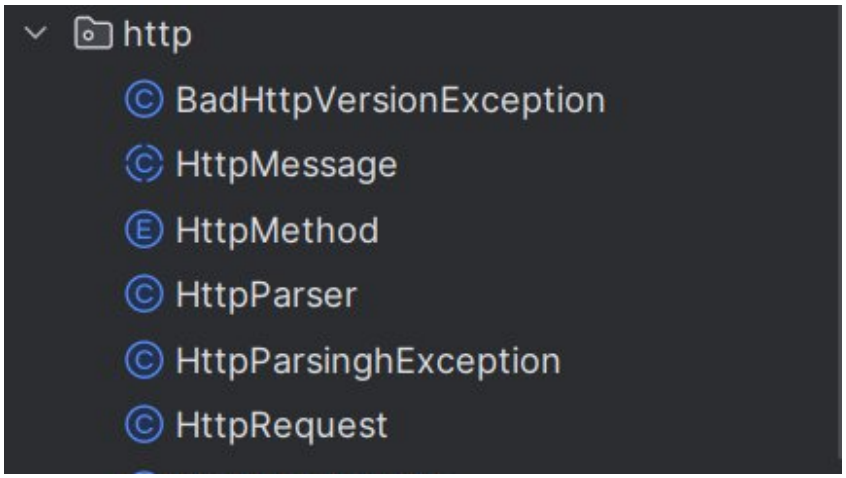

Рисунок 25 – Классы для реализации HTTP протокола

Краткое описание функционала классов:

- BadHttpVersionException – класс, обрабатывающий ситуации, когда ответ на запрос HTTP пришел с ошибкой;
- HttpMessage – класс, содержащий в себе тело HTTP ответа;
- HttpMethod – класс, содержащий перечисление доступных в приложении возможностей;

- HttpParser – класс, который осуществляет обработку HTTP ответа;
- HttpParsingException – класс, обрабатывающий исключительные ситуации при обработке P ответа;
- HttpRequest – класс, который описывает HTTP запросы.

Далее для работы с сетью были разработаны классы конфигурации, которые представлены на рисунке 26.

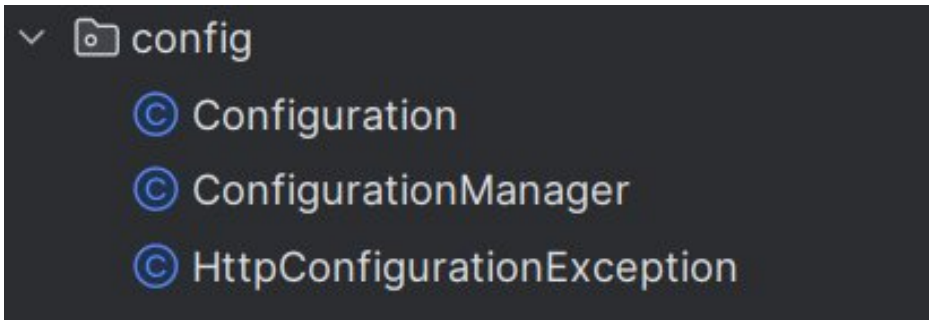

Рисунок 26 – Классы конфигурации для работы с сетью

Краткое описание разработанных классов:

- Configuration – класс, который хранит в себе данные и методы конфигурации;
- ConfigurationManager – класс, осуществляющий загрузку файла конфигурации по указанному пути;
- HttpConfigurationException – класс, обрабатывающий исключительные ситуации при загрузке файла конфигурации.

Программный код класса ConfigurationManager представлен в приложении А, код класса HttpConnectionWorkerThread представлен в приложении Б.

Разработка серверной части приложения производилась в среде разработки IntelliJ IDEA. Иерархия разработанного проекта представлена на рисунке 27.

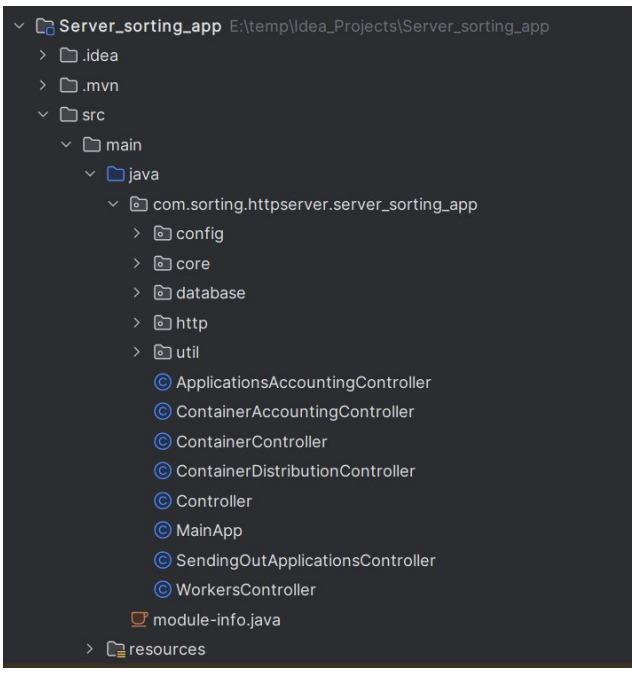

Рисунок 27 – Иерархия разработанной серверной части приложения

На рисунке 28 представлены ресурсы проекта, в которых хранятся файлы FXML.

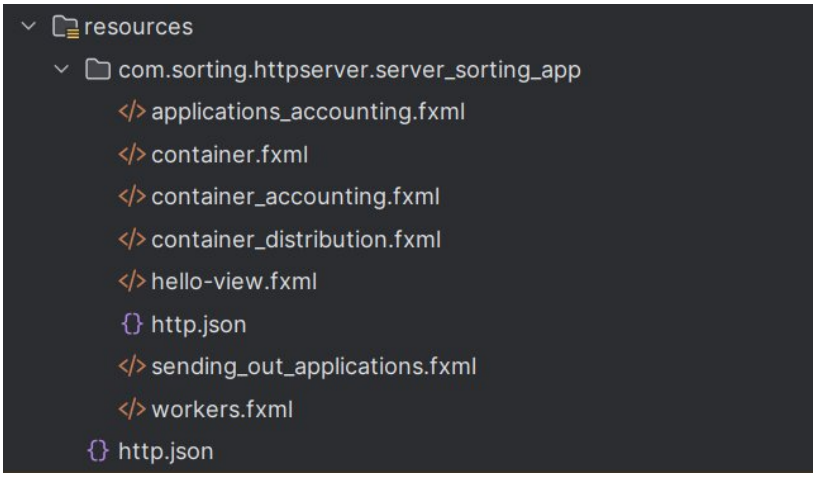

Рисунок 28 – Ресурсы проекта

Файлы FXML нужны для описания иерархии компонентов в пользовательском интерфейсе. Там определены все компоненты приложения и их свойства, а также связь с контроллером, который отвечает за управление

взаимодействием. Класс контроллер является связующим звеном между кодом на java и FXML файлом. На рисунке 29 представлены разработанные классы контроллеры.

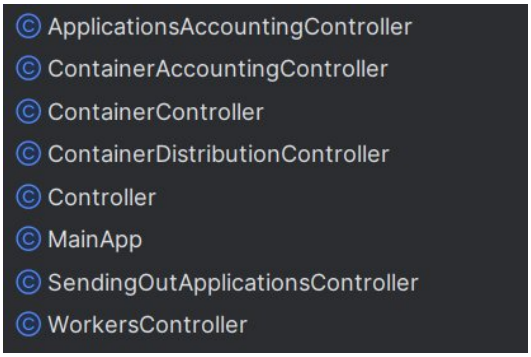

Рисунок 29 – Классы контроллеры

Каждый класс связан со своим FXML файлом. В данных классах описываются функциональные классовые компоненты окна и методы инициализации окна.

Программный код класса ServerListenerThread представлен в приложении В, код класса DbHandler представлен в приложении Г.

Перейдем к описанию функций разработанной серверной части приложения для начальника производства.

При запуске приложения на экране отображается главная страница. Главная страница состоит из кнопки для включения сервера, которая запускает HTTP сервер, для обмена сообщениями с клиентом, и кнопок для перехода по формам. После входа в приложения начальник производства может переходить по следующим формам:

- Работники сбора макулатуры;
- Контейнера;
- Рассылка поступающих заказов на сдачу макулатуры;
- Распределение контейнеров между работниками;
- Учет состояния контейнеров;

– Учет состояния заявок на сдачу макулатуры.

На рисунке 30 изображена главная страница приложения для начальника производства.

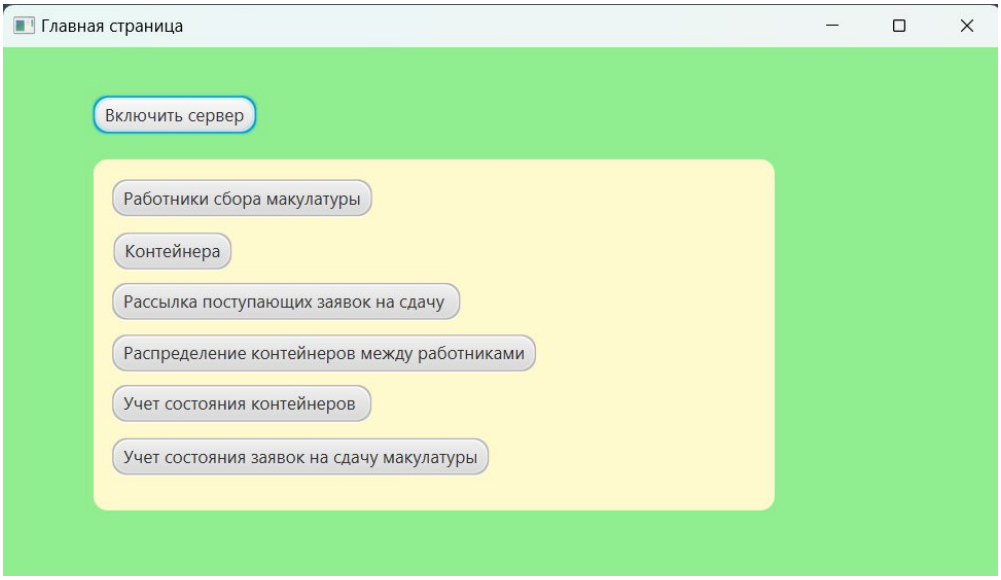

Рисунок 30 – Главная страница приложения

На рисунке 31 представлен функционал, отображающийся при переходе в раздел – Работники сбора макулатуры.

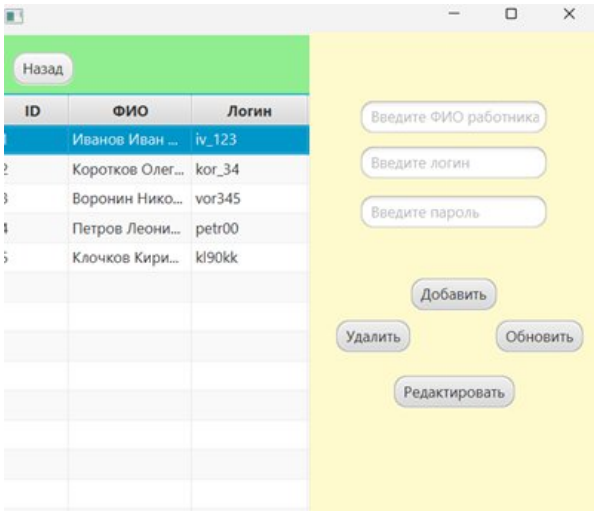

Рисунок 31 – Работники сбора макулатуры

В данном разделе начальник производства может вводить данные о работнике и заносить их в базу данных. А также редактировать, обновлять или удалять из таблицы. Кнопка «Назад» возвращает на главную страницу приложения.

На рисунке 32 представлен функционал, отображающийся при переходе в раздел – Контейнеры.

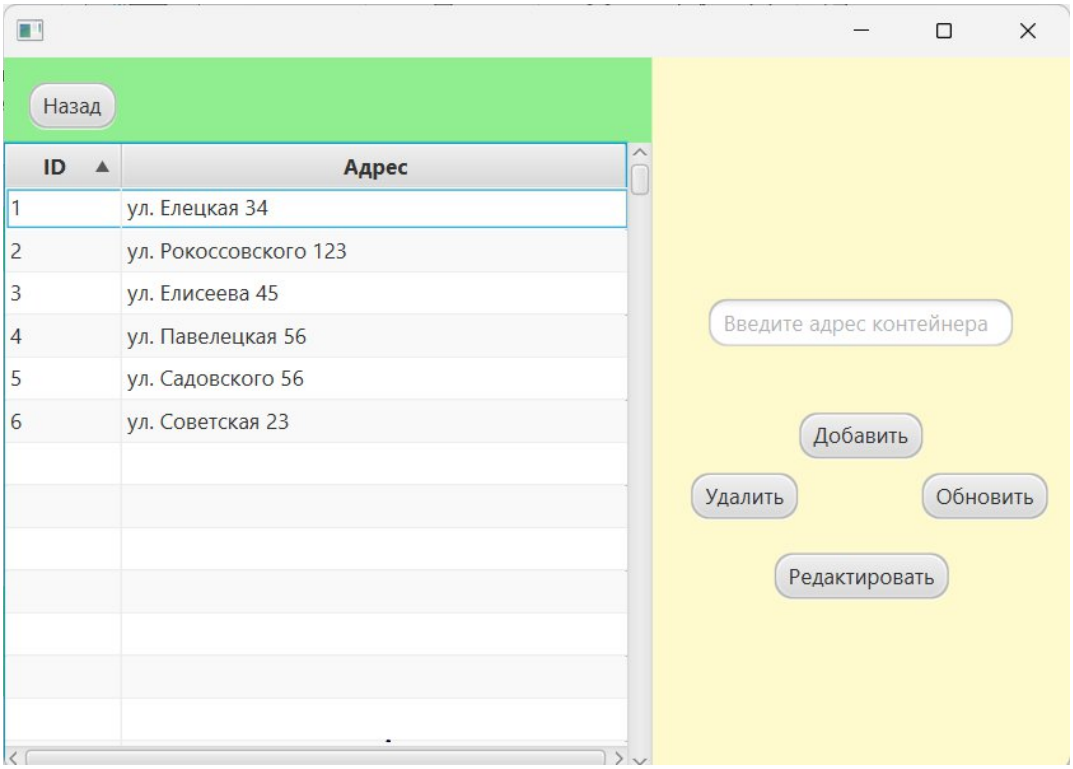

Рисунок 32 – Контейнеры

В данном разделе начальник может вводить данные о контейнере и заносить их в базу данных. А также редактировать, обновлять или удалять из таблицы. Кнопка «Назад» возвращает на главную страницу приложения.

На рисунке 33 представлен функционал, отображающийся при переходе в раздел – Рассылка поступающих заказов на сдачу макулатуры.

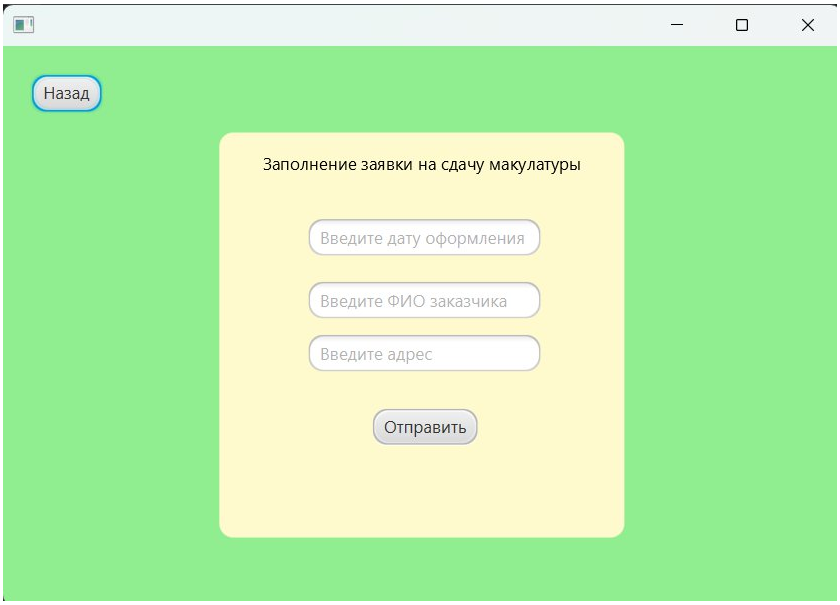

Рисунок 33 – Рассылка поступающих заказов на сдачу макулатуры

На данной странице можно вводить данные о поступающей заявке и производить рассылку по работникам сбора макулатуры.

На рисунке 34 представлен функционал, отображающийся при переходе в раздел – Распределение контейнеров между работниками.

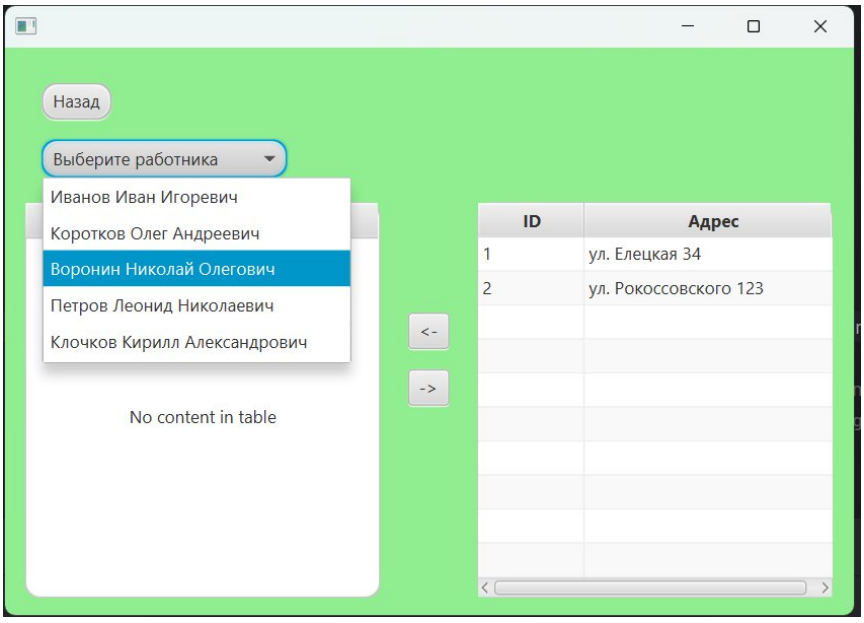

Рисунок 34 – Распределение контейнеров между работниками

Для того, чтобы управлять контейнерами работника, нужно выбрать нужного работника и нажать на него. После появится таблица контейнеров, которые закреплены за этим работником (рисунок 35).

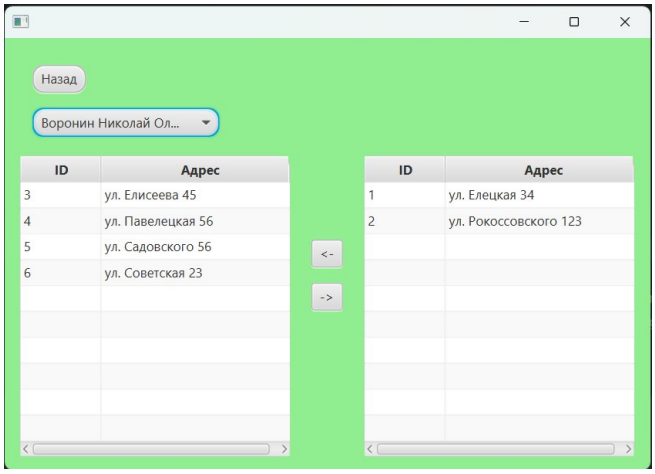

Рисунок 35 – Распределение контейнеров между работниками

В данном разделе можно либо назначить свободный контейнер выбранному работнику, либо убрать контейнер из списка.

На рисунке 36 представлен функционал, отображающийся при переходе в раздел – Учет состояния контейнеров.

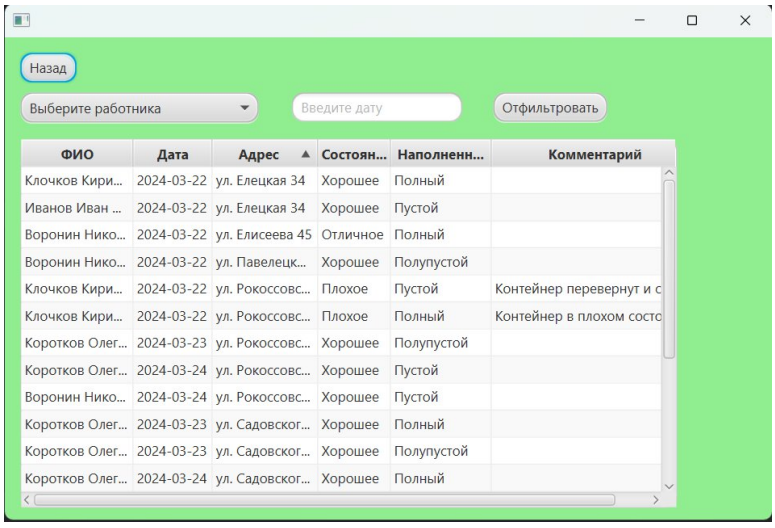

Рисунок 36 – Учет состояния контейнеров

В данной форме выводится таблица данных о контейнерах, а также ФИО работников, которые их обслуживают. Можно отфильтровать данные по ФИО работника и(или) по дате.

Для выгрузки отчетов в Word подключена библиотека microsoft.office.interop.Word (рисунок 37).

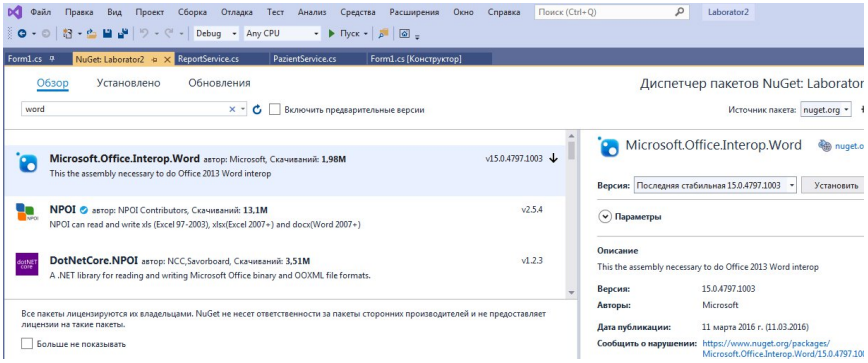

Рисунок 37 – Добавление WORD

Метод для экспорта в Word представлен в коде ниже:

```
public void ExportWord(DataGridView DGV)
   {
      if (DGV.Rows.Count != 0)
      {
         int RowCount = DGV.Rows.Count;
         int ColumnCount = DGV.Columns.Count;
         Object[,] DataArray = new object[RowCount + 1, ColumnCount + 1];

      //добавим поля и колонки
int r = 0;
         for (int c = 0; c <= ColumnCount - 1; c++)
         {
            for (r = 0; r <= RowCount - 1; r++)
            {
               DataArray[r, c] = DGV.Rows[r].Cells[c].Value;
            }
         }
         Microsoft.Office.Interop.Word.Document oDoc = new
Microsoft.Office.Interop.Word.Document();
         oDoc.Application.Visible = true;
```

}

На рисунке 38 представлен функционал, отображающийся при переходе в раздел – Учет состояния заявок на сдачу макулатуры.

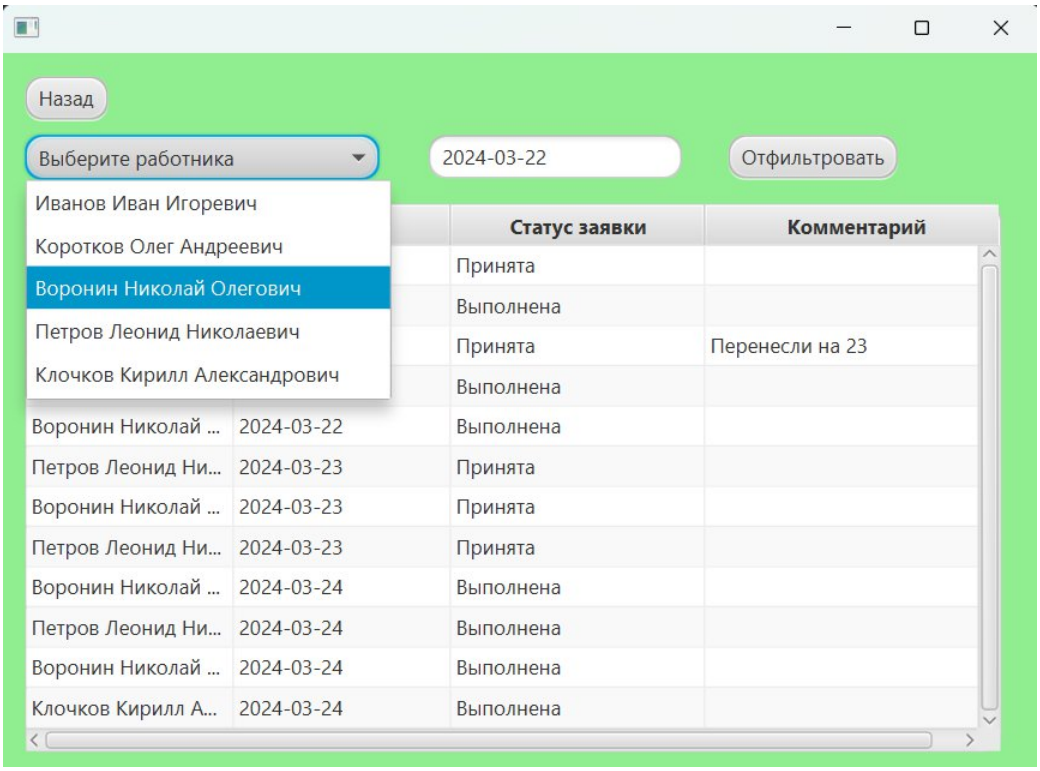

Рисунок 38 – Учет состояния заявок на сдачу макулатуры

В данном разделе выводится таблица данных о контейнерах, а также ФИО работников, которые их обслуживают. Можно отфильтровать данные по ФИО работника и(или) по дате.

При выполнении тестирования серверной части приложения получены следующие результаты:

- успешный вход в приложение;
- таблицы отображаются корректно;
- при нажатии кнопок «Добавить», «Удалить», «Обновить» и «Редактировать» выполняются SQL запросы и данные изменяются;
- рассылка заявок работает корректно;

- функционал добавления и удаления контейнера у работника работает исправно.

Таким образом, проведенное тестирование свидетельствует об успешной реализации приложения для начальника производства.

Перейдем к описанию функций разработанной клиентской части приложения для работников сбора макулатуры.

При запуске приложения на экране отображается страница входа в приложение (рисунок 39).

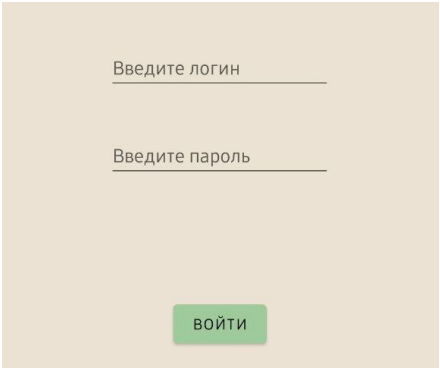

Рисунок 39 – Страница входа в приложение

Далее после введение логина и пароля работник сбора макулатуры переходит на главную страницу (рисунок 40).

Рисунок 40 – Главная страница клиентской части приложения

На главной странице содержится информация: заявки и отправка информации о контейнере.

На рисунке 41 представлен функционал, отображающийся при переходе в раздел – Заявки.

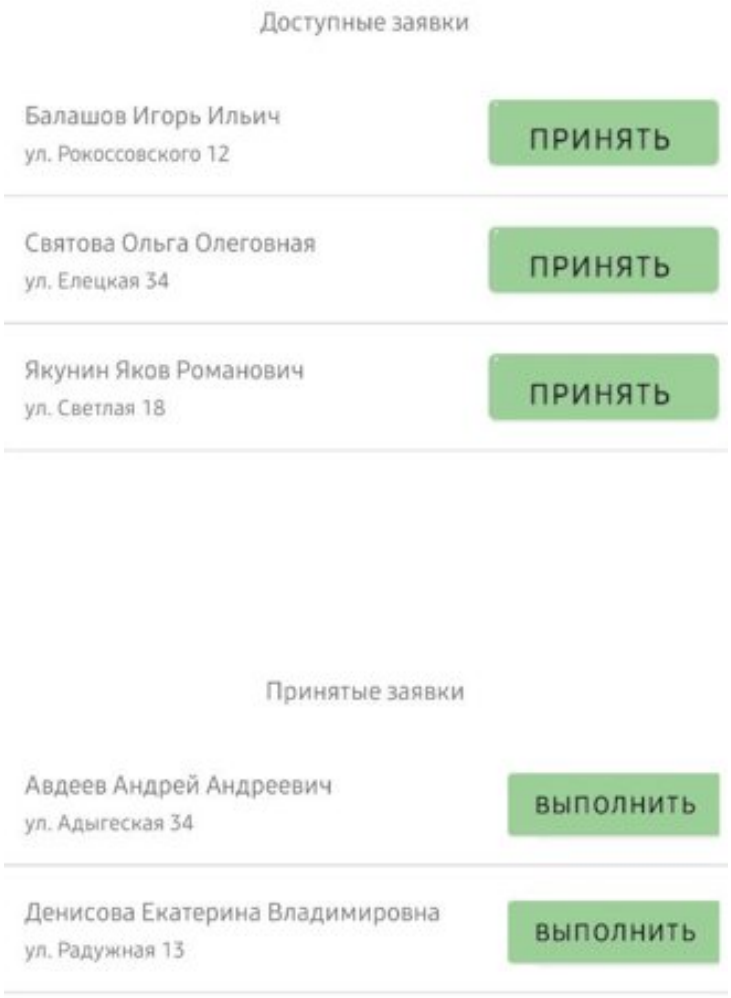

Рисунок 41 – Страница заявок

При принятии доступной заявки она переходит в принятые (рисунок 42)

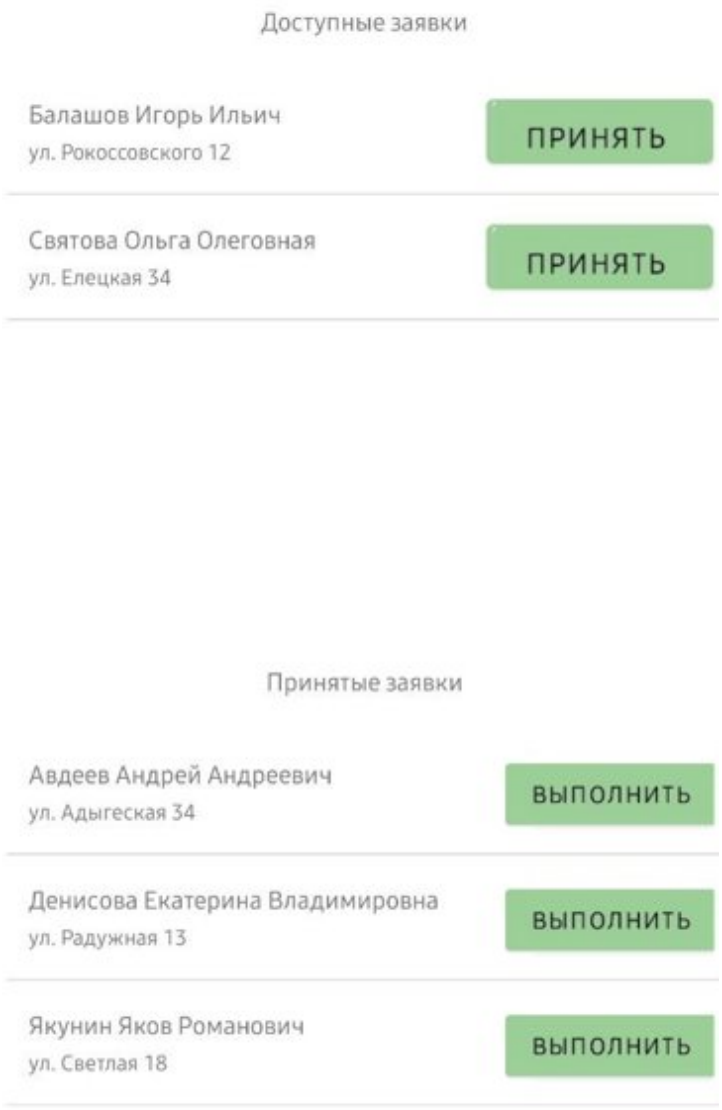

Рисунок 42 – Страница заявок

Далее при выполнении заявки она исчезает и обновленный статус отправляется на сервер. Таким образом, работник сбора макулатуры может принимать заявки на сдачу макулатуры от начальника производства и отмечать их выполнение.

На рисунке 43 представлен функционал, отображающийся при переходе в раздел – Отправка информации о контейнере.

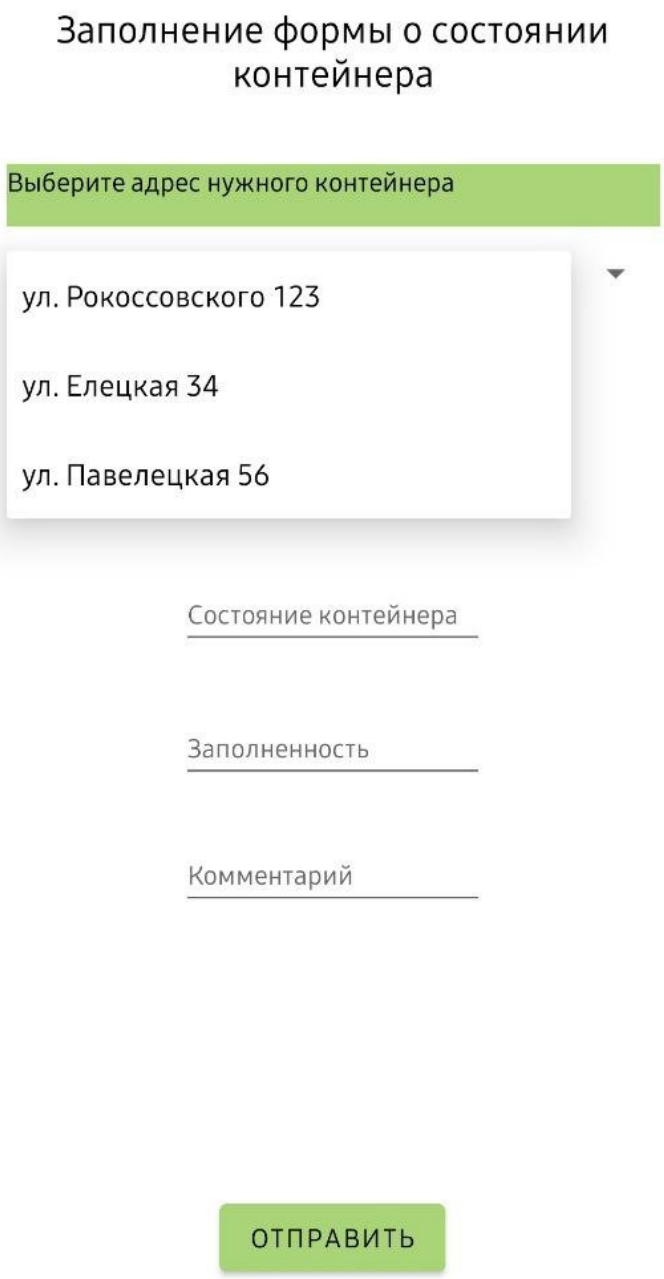

Рисунок 43 – Отправка информации о контейнере

В данном разделе можно заполнить информацию о контейнере. Для начала в выпадающем списке нужно выбрать адрес нужного контейнера и нажать на него. После этого заполнить поля формы (рисунок 44).

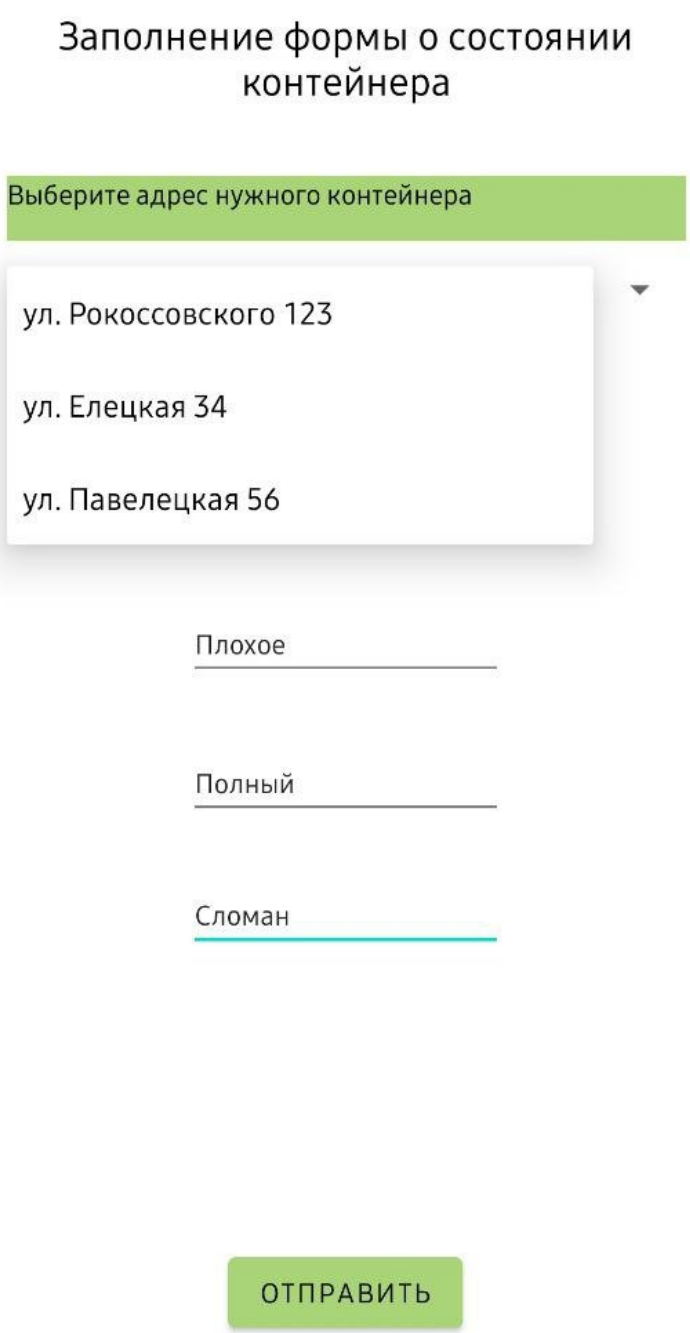

Рисунок 44 – Отправка информации о контейнере

После внесения данных нужно нажать на кнопку «Отправить» и данные отправляются на сервер.

При выполнении тестирования клиентской части приложения получены следующие результаты:

- успешная авторизация в приложении;

- таблицы отображаются корректно;
- при нажатии кнопок «Добавить», «Удалить», «Обновить» и «Редактировать» выполняются SQL запросы и данные изменяются;
- рассылка заявок работает корректно;
- функционал добавления и удаления контейнера у работника работает исправно.

Таким образом, проведенное тестирование свидетельствует об успешной реализации приложения для начальника производства.

3.4 Оценка стоимости и окупаемости

«В рамках оценки стоимости реализации приложения для компании «Квартплата 24» определены следующие категории затрат:

- оплата труда сотрудников, которые участвуют в реализации приложения;
- расходы на запасные части и комплектующие;
- расходы на оплату электроэнергии.

В таблице 10 приведен расчет сумм оплаты труда специалистов» [12].

Таблица 10 – Расчет сумм оплаты труда специалистов, привлеченных к разработке приложения для компании «Квартплата 24»

Должность	Величина оплаты труда специалиста за 1 час., руб.	Величина трудозатрат	Величина заработной платы, руб.
Программист	800	102,5	82000
Специалист компании (Аналитик)	350	80	28000
Директор	460	40	18400
Экономист или бухгалтер	200	5	1000
Итого			129400

«Величина вложений, связанных с оплатой труда специалистов, привлеченных к разработке приложения для компании «Квартплата 24»,

составили 218,4 тыс. руб. С учетом отчислений страховых взносов, тариф которых составляет 30,2%, величина затрат принимает значение» [13]:

$$Z = 128.4 * 1.302 = 167.2 \text{ руб.} \quad (1)$$

«Расчет затрат, связанных с использование компьютерного оборудование, необходимого при разработке приложения для компании «Квартплата 24», производится через расчёт доли времени разработки системы относительно срока амортизации, составляющего для данной категории оборудования 5 лет» [14].

$$S_A = \frac{1.5}{60} * 80000 = 2000 \text{руб.} \quad (2)$$

«Затраты, связанные с оплатой тарифа электроэнергии, составляющего в 6 руб./кВт*ч с учетом 180 часов машинного времени, 0,7 кВт полезной мощности используемого оборудования, составят: $S_E = 6 * 180 * 0.7 =$ 756 руб.

Таким образом, оценка полученного эффекта составляет 275150 руб.

Срок окупаемости приложения для компании «Квартплата 24» составит» [15]:

$$T_{OK} = \frac{174932}{275150} * 12 = 7.6 \text{ мес.} \quad (3)$$

Окупаемость разработки приложения составила 7,6 месяцев

Выводы по главе 3

В третьей главе были рассмотрены вопросы практической реализации системы корпоративной информации и особенности реализации.

Заключение

В рамках выполнения выпускной квалификационной работы реализован полный цикл разработки клиент-серверного программного обеспечения, предназначенного для автоматизации бизнес-процесса по сбору и анализу информации о приеме макулатуры на предприятии КвартПлата24.

В первой главе проведена аналитическая работа: выполнено исследование предметной области, сформулированы цели и задачи проекта, детально описан текущий бизнес-процесс предприятия.

Проведён сравнительный анализ существующих решений на рынке, что позволило выделить их сильные и слабые стороны и обосновать необходимость разработки собственного программного продукта. На основе полученных данных сформулированы функциональные и нефункциональные требования к разрабатываемой системе.

Во второй главе в процессе проектирования системы разработаны логические и функциональные модели, спроектирована структура базы данных с учётом предполагаемой нагрузки и объёмов обрабатываемой информации, а также определён и обоснован стек технологий, наиболее подходящий для реализации проекта. Выбор технологий обеспечил баланс между производительностью, надёжностью и удобством дальнейшего сопровождения системы.

В третьей главе описан процесс разработки информационной системы. Разработка велась по модульному принципу и включала реализацию трёх основных компонентов: базы данных, серверной части (backend) и клиентского интерфейса (frontend).

Особое внимание было уделено удобству взаимодействия пользователя с приложением — пользовательский интерфейс выполнен в соответствии с принципами UI/UX-дизайна, что обеспечивает интуитивно понятную навигацию и доступ к основным функциям системы.

Созданное клиент-серверное приложение предназначено для автоматизации процесса учёта и контроля сбора макулатуры.

Его внедрение позволит существенно сократить временные затраты на обработку данных, минимизировать ошибки, связанные с человеческим фактором, и повысить общую эффективность производственных процессов предприятия.

На завершающем этапе разработки было проведено комплексное тестирование системы.

По результатам тестирования было подтверждено соответствие системы заявленным требованиям, её стабильная работа и готовность к эксплуатации в реальных условиях.

Таким образом, цель и задачи бакалаврской работы выполнены.

Список используемой литературы и используемых источников

1. Буч Г. Введение в UML от создателей языка. Гради Буч, Джеймс Рамбо, Ивар Якобсон — 2-е издание — М.: ДМК Пресс, 2015. — 496 с.

2. Буч Г. Объективно-ориентированное проектирование. С примерами применения – М.:Диалектика, 1992 – 520 с.

3. Влияние переработки макулатуры на экологию окружающей среды / [Электронный ресурс]. - URL: http://www.pnzstroi.ru/obzory/112307/vliyanie-pererabotki-makulatury-na-ekologiyu-okruzhayuschej-sredy (дата обращения 2.03.2025).

4. Глотова Т.В. Объектно-ориентированная методология разработки сложных систем / Глотова Татьяна Владимировна – Пенза: Издательство ПГУ, 2016 – 40 с.

5. Дейтел П., Дейтел Х., Дейтел Э., Моргано М. Android для программистов: создаём приложения. — СПб.: Питер, 2013. — 560 с.

6. Локтева Д.А. Клиент-серверное приложение на базе JavaFX - МГТУ им. Н.Э. Баумана, 2023, 36 с.

7. Макулатура / [Электронный ресурс] // Википедия. - URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%83%D0%BB%D0%B0%D1%82%D1%83%D1%80%D0%B0 (дата обращения 2.03.2025).

8. Орлов С. А. Технологии разработки программного обеспечения: учебник / С. А. Орлов. - СПб., 2024. - 527 с.

9. Роджерс Р., Ломбардо Д., Медникс З., Мейк Б. Android. Разработка приложений – М.: ЭКОМ Паблишерз, 2010 – 400 с.

10. Руководство по JavaFX – metanit.com URL: https://metanit.com/java/javafx/ (дата обращения 2.03.2025).

11. Шилдт Г. Java. Полное руководство, (8-е изд.) - М.: «И.Д.Вильямс», 2012, 1104 с.

12. Язык программирования Java – metanit.com URL: https://metanit.com/java/tutorial/ (дата обращения 2.03.2025).

13. Java / [Электронный ресурс] // metanit.com. - URL: https://metanit.com/java/tutorial/ (дата обращения 25.03.2025).

14. Elmasri, R., & Navathe, S. B. (2020). Fundamentals of Database Systems (7th ed.). – M.: Pearson, 2020. – 1200 с..

15. Jakob Freund, Bernd Rucker Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company – CreateSpace – 2016

16. Joseph M. Hellerstein, Michael Stonebraker, James Hamilton Architecture of a Database System – Hanover, USA 2020.

17. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts (7th ed.). McGraw-Hill, 2019. – 1376 с.

18. Delivery Club добавил в приложение заказ «такси» для мусора / [Электронный ресурс]. - https://shoppers.media/news/1279_delivery-club-dobavil-v-prilozenie-zakaz-taksi-dlya-musora (дата обращения 2.03.2025).

19. EcoTaxi – сервис вывоза отходов / [Электронный ресурс]. - URL: https://ecotaxi.io/ (дата обращения 2.03.2025).

20. IDEF0 / [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/IDEF0 (дата обращения 25.03.2025).

21. Ubirator, сервис по вывозу вторсырья / [Электронный ресурс]. - URL: https://ubirator.com/b2b/upro?ysclid=lvcdfet6pk678069286 (дата обращения 2.03.2025).

Приложение А

Класс ConfigurationManager

```
// Импорт необходимых классов
package com.sorting.httpserver.server_sorting_app.config;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.sorting.httpserver.server_sorting_app.util.Json;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

// Класс отвечает за получения информации о конфигурации сервера
public class ConfigurationManager
{
   private static ConfigurationManager myConfigurationManager;
   private static Configuration myCurrentConfiguration;

   private ConfigurationManager()
   {
   }

   public static ConfigurationManager getInstance()
   {
      if (myConfigurationManager == null)
         myConfigurationManager = new ConfigurationManager();
      return myConfigurationManager;
   }

   /*
   Используется для загрузки файла конфигурации по указанному пути.
    */
   public void loadConfigurationFile(String filePath)  {
      FileReader fileReader = null;
```

```
            try
            {
               fileReader = new FileReader(filePath);
            }
            catch (FileNotFoundException e)
            {
               throw new HttpConfigurationException(e);
            }
            StringBuffer sb = new StringBuffer();
            int i;
            try
            {
               while ( (i = fileReader.read()) != -1)
                  sb.append((char) i);
            }
            catch (IOException e)
            {
               throw new HttpConfigurationException(e);
            }
            JsonNode conf = null;
            try
            {
               conf = Json.parse(sb.toString());
            }
            catch (IOException e)
            {
               throw new HttpConfigurationException("Error parsing the Configuration File",
e);
            }

            myCurrentConfiguration = new Configuration(conf.get("port").asInt(),
conf.get("webroot").asText());
         }
```

```
    /*
    Возврат к текущей загруженной конфигурации
     */
    public Configuration getCurrentConfiguration()
    {
        if (myCurrentConfiguration == null)
        {
            throw new HttpConfigurationException("No Current Configuration Set");
        }
        return myCurrentConfiguration;
    }
}
```

Приложение Б

Класс HttpConnectionWorkerThread

```
// Импорт необходимых классов
package com.sorting.httpserver.server_sorting_app.core;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

// Класс описывает процесс подключения и дальнейшую обработку http соединения
public class HttpConnectionWorkerThread extends Thread
{
   private final static Logger LOGGER = LoggerFactory.getLogger(HttpConnectionWorkerThread.class);
   private Socket socket;
   public HttpConnectionWorkerThread(Socket socket)
   {
      this.socket = socket;
   }

// Обработка поступающих запросов на сервер и их обработка
   @Override
   public void run()
   {
      InputStream inputStream = null;
      OutputStream outputStream = null;
      try
      {
         inputStream = socket.getInputStream();
         outputStream = socket.getOutputStream();
```

```
            String html = "<html><head><title>Simple Java Http Server<html></title></head><body><h1>Server starting</h1></body></html>";

            final String CRLF = "\n\r"; //13,10

            String response =
                "HTTP/1.1 200 OK" + CRLF + //Status Line : HTTP VERSION RESPONSE_CODE RESPONSE_MESSAGE
                    "Content-Length: " + html.getBytes().length + CRLF + //HEADER
                    CRLF +
                    html +
                    CRLF + CRLF;

            outputStream.write(response.getBytes());

            /*try {
               sleep(5000);
            } catch (InterruptedException e) {
               e.printStackTrace();
               //throw new RuntimeException(e);
            }*/
            LOGGER.info("Connection Processing Fineshed...");
         }
         catch (IOException e)
         {
            LOGGER.error("Problem with communication", e);
         }
         finally
         {
            if (inputStream!= null)
            {
               try
               {
```

```
                inputStream.close();
            }
            catch (IOException e) {}
        }
        if (outputStream!= null)
        {
            try
            {
                outputStream.close();
            }
            catch (IOException e) {}
        }
        if (socket!=null)
        {
            try
            {
                socket.close();
            }
            catch (IOException e) {}
        }
    }
  }
}
```

Приложение В

Класс ServerListenerThread

```
// Импорт необходимых классов
package com.sorting.httpserver.server_sorting_app.core;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

// Основной прослушивающий поток сервера
public class ServerListenerThread extends Thread {
   private final static Logger LOGGER = LoggerFactory.getLogger(ServerListenerThread.class);
   private int port;
   private String webroot;
   private ServerSocket serverSocket;

   public ServerListenerThread(int port, String webroot) throws IOException {
      this.port = port;
      this.webroot = webroot;
      this.serverSocket = new ServerSocket(this.port);
   }

// Функция ожидает подключения от клиентов и создает для этих подключений отдельный поток обработки
   @Override
   public void run() {

      try
      {
         while (serverSocket.isBound() && !serverSocket.isClosed())
```

```
                {
                    Socket socket = serverSocket.accept();

                    LOGGER.info(" * Connection accepted: " + socket.getInetAddress());

                    HttpConnectionWorkerThread workerThread = new
HttpConnectionWorkerThread(socket);
                    workerThread.start();
                }
            }
            catch(IOException e)
                {
                    LOGGER.error("Problem with setting socket", e);
                }
            finally
            {
                if (serverSocket!=null)
                {
                    try {
                        serverSocket.close();
                    }
                    catch (IOException e)
                    {
                        throw new RuntimeException(e);
                    }
                }

            }
        }
    }
```

Приложение Г

Класс DbHandler

```
// Импорт необходимых классов
package com.sorting.httpserver.server_sorting_app.database;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.*;

// Класс взаимодействия с базой данных
public class DbHandler extends DatabaseConfig
{
    static Connection dbConnection;



        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        }
        catch(Exception e)
        {
            // Todo: add handler.
            // нужно добавить логирование при ошибке подгрузки.
        }
        dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);

        return dbConnection;
    }

// Функция посылает запрос к базе данных о регистрации нового пользователя
    public void signUpUser(String full_name, String login, String password)
    {
```

```
            String insert = "INSERT INTO " + Const.WORKERS_TABLE +
                    "(" + Const.WORKERS_FULL_NAME + "," + Const.WORKERS_PASSWORD +
                    "," + Const.WORKERS_LOGIN + ")" +
                    "VALUES(?,?,?)";

            try {
                PreparedStatement prSt = getDbConnection().prepareStatement(insert);
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                //throw new RuntimeException(e);
                e.printStackTrace();
            }
        }

    // Функция возвращает таблицу работников
        public ObservableList<Workers> getDataworkers()
        {
            ObservableList<Workers> list = FXCollections.observableArrayList();
            try
            {
             PreparedStatement ps = getDbConnection().prepareStatement("select * from workers");
             ResultSet rs = ps.executeQuery();
             while(rs.next())
             {
                list.add(new Workers(rs.getInt(Const.WORKERS_ID),
                        rs.getString(Const.WORKERS_FULL_NAME),
                        rs.getString(Const.WORKERS_LOGIN),
                        rs.getString(Const.WORKERS_PASSWORD)));
             }
            }
            catch (SQLException e)
```

```
            {
                throw new RuntimeException(e);
            } catch (ClassNotFoundException e) {
                throw new RuntimeException(e);
            }

            return list;
        }

    // Функция возвращает таблицу контейнеров
        public ObservableList<Container> getDatacontainer()
        {
            ObservableList<Container> list = FXCollections.observableArrayList();
            try
            {
                PreparedStatement ps = getDbConnection().prepareStatement("select * from container");
                ResultSet rs = ps.executeQuery();

                while(rs.next())
                {
                    list.add(new Container(rs.getInt(Const.CONTAINER_ID),
                            rs.getString(Const.CONTAINER_ADDRESS)));
                }
            }
            catch (SQLException e)
            {
                throw new RuntimeException(e);
            } catch (ClassNotFoundException e) {
                throw new RuntimeException(e);
            }

            return list;
```

```
    }
// Функция возвращает таблицу состояния контейнера
public ObservableList<ContainerAccounting> getDataContainerAccounting()
    {
        ObservableList<ContainerAccounting> list = FXCollections.observableArrayList();
        try
        {
            PreparedStatement ps =  getDbConnection().prepareStatement("select full_name,
date, address, status, fullness, comment from container_status ");
            ResultSet rs = ps.executeQuery();

            while(rs.next())
            {
                list.add(new
ContainerAccounting(rs.getString(Const.WORKERS_FULL_NAME),
                        rs.getString(Const.STATUS_DATA),
                        rs.getString(Const.CONTAINER_ADDRESS),
                        rs.getString(Const.STATUS_CON),
                        rs.getString(Const.STATUS_FULL_NESS),
                        rs.getString(Const.STATUS_COMMENT)));
            }
        }
        catch (SQLException e)
        {
            throw new RuntimeException(e);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        return list;
    }
// Функция возврщает таблицу состояния заявок
    public ObservableList<ApplicationsAccounting> getDataApplicationsaccounting()
```

```
        {
            ObservableList<ApplicationsAccounting> list = FXCollections.observableArrayList();
            try
            {
                PreparedStatement ps =  getDbConnection().prepareStatement("select full_name, status_request, comment, date  from status_request ");
                ResultSet rs = ps.executeQuery();

                while(rs.next())
                {
                    list.add(new ApplicationsAccounting(rs.getString(Const.WORKERS_FULL_NAME),
                            (rs.getString(Const.STATUS_REQUEST)),
                            (rs.getString(Const.STATUS_COMMENT)),
                            rs.getString(Const.STATUS_DATA)));
                }
            }
            catch (SQLException e)
            {
                throw new RuntimeException(e);
            } catch (ClassNotFoundException e) {

                throw new RuntimeException(e);
            }

            return list;
        }

    // Перечисления знаков сравнения
        public enum MatchingSymbol
        {
            bigger,
```

```
    little,
        equal,
        little_equal,
        bigger_equal
    }

    private String ToString(MatchingSymbol ms)
    {
        return switch (ms) {
            case bigger -> ">";
            case little -> "<";
            case bigger_equal -> ">=";
            default -> null;
        };
    }

    // Функция возвращает таблицу контейнеров, но применяя фильтрацию
    public ObservableList<Container> getDatacontainerMatch(int containerId, MatchingSymbol ms)
    {
        ObservableList<Container> list = FXCollections.observableArrayList();
        try
        {
            PreparedStatement ps = getDbConnection().prepareStatement("select * from container where " + Const.CONTAINER_ID + " " +
                                                                      ToString(ms) + " " + containerId);
            ResultSet rs = ps.executeQuery();

            while(rs.next())
            {
                list.add(new Container(rs.getInt(Const.CONTAINER_ID),
                        rs.getString(Const.CONTAINER_ADDRESS)));
            }
```

```
        }
        catch (SQLException e)
        {
            throw new RuntimeException(e);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        return list;
    }
}
```