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

«___»_____ 2024 г.

# Разработка Android-приложения по контролю технического обслуживания ратраков

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-376.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ Т.Ю. Маковецкая

Автор работы,
студент группы КЭ-404
_____ Д.А. Пашнин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 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
29.01.2024 г.

**ЗАДАНИЕ**
**на выполнение выпускной квалификационной работы бакалавра**
студенту группы КЭ-404
Пашнину Дмитрию Александровичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка Android-приложения по контролю технического обслуживания ратраков.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Kotlin Developer Documentation. [Электронный ресурс] URL: https://kotlinlang.org/docs/home.html (дата обращения: 10.02.2024 г.).

3.2. PostgreSQL Developer Documentation. [Электронный ресурс] URL: https://www.postgresql.org/docs/current/index.html (дата обращения: 10.02.2024 г.).

3.3. Android Studio Developer Documetation. [Электронный ресурс] URL: https://developer.android.com/studio/intro (дата обращения: 10.02.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Анализ предметной области.

4.2. Проектирование приложения.

4.3. Реализация приложения.

4.4. Тестирование функционала приложения.

**5. Дата выдачи задания:** 29.01.2024 г.

**Научный руководитель,**
доцент кафедры СП, к.ф.-м.н. Т.Ю. Маковецкая

**Задание принял к исполнению** Д.А. Пашнин

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ ..... 5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ ..... 7
1.1. Описание предметной области ..... 7
1.2. Анализ аналогичных проектов ..... 8
2. ПРОЕКТИРОВАНИЕ ..... 12
2.1. Проектирование приложения продавца ..... 12
2.2. Проектирование приложения покупателя ..... 15
2.2.1. Варианты использования приложения покупателя ..... 16
2.2.2. Архитектура системы ..... 17
2.2.3. Построение схемы данных ..... 18
2.2.4. Диаграмма классов приложения покупателя ..... 19
2.2.5. Проектирование интерфейса приложения покупателя ..... 19
3. РЕАЛИЗАЦИЯ ..... 22
3.1. Выбор программных средств реализации ..... 22
3.2. Реализация приложения продавца ..... 23
3.3. Реализация приложения покупателя ..... 25
4. ТЕСТИРОВАНИЕ ..... 28
ЗАКЛЮЧЕНИЕ ..... 30
ЛИТЕРАТУРА ..... 31
ПРИЛОЖЕНИЕ. Скриншоты приложения продавца ..... 33

# ВВЕДЕНИЕ

## Актуальность

Коммерческие компании постоянно стараются извлекать наибольшую выгоду из своих продуктов и услуг. Одно из многих возможных решений – это оптимизация различных задач. Обычно какая-либо оптимизация – это решение, на практическую реализацию которого необходимо потратить ресурсы (и, возможно, необходимо будет и в будущем), однако результат этого решения в долгосрочной перспективе принесет больше выгоды, чем было потрачено в сумме ресурсов. Также необходимо понимать, что чем меньше компания, тем быстрее и дешевле внедрение решения. Поэтому особенно стараются оптимизировать бизнес-процессы относительно новые (еще неинертные) компании.

Так, для достижения вышеперечисленных целей компанией ООО «КБ Ратрак-Урал» было принято решение перевести инструкцию по эксплуатации в электронный вид, и поставлять ее покупателям на планшете в приложении, которое также будет выполнять задачу контроля проведения технических обслуживаний. В соответствии с этим была поставлена цель данной работы.

## Постановка задачи

Целью выпускной квалификационной работы является разработка Android-приложения по контролю технического обслуживания ратраков. Для достижения поставленной цели необходимо решить следующие задачи:

1) анализ предметной области;
2) проектирование приложения;
3) реализация приложения;
4) тестирование функционала приложения.

## Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы. Объем работы составляет 34 страницы, объем списка литературы – 15 источников.

В первой главе описывается предметная область для формирования более глубокого понимания проблемы и затрагиваемых процессов. Также проводится обзор аналогов мобильных приложений из смежной области, выделены их сильные и слабые стороны с точки зрения пользователей, а также используемые решения, формирующие опыт взаимодействия пользователя (UX).

Вторая глава посвящена описанию требований и проектированию приложения. В ходе работы над данной главой была определена необходимость в разработке второго приложения, конечным пользователем которого является инженер компании-продавца. Также были сформированы функциональные и нефункциональные требования, построены диаграммы вариантов использования приложений, диаграммы классов, макеты экранов приложений. Спроектирована и отражена с помощью диаграммы компонентов архитектура системы. Построена схема базы данных.

В третьей главе анализируются программные средства разработки, производится их выбор для реализации приложений. Также приведены некоторые детали реализации приложения, проведено сравнение получившегося пользовательского интерфейса с макетами.

Четвертая глава содержит описание проведенных функционального и интеграционного тестирований, представлены их результаты.

В приложении содержатся макет и получившийся интерфейс одного из разработанных приложений.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Описание предметной области

Компания ООО «КБ Ратрак-Урал» является производителем и продавцом ратраков (снегоуплотнительной машины) в одном лице, вследствие чего именно инженеры данной компании определяют количество, сроки и этапы техобслуживания, но с точки зрения данной работы основной ролью компании, в противовес покупателю, является роль продавца.

При продаже ратрака компания назначает гарантийный срок работы. Для сохранения этого срока в течение эксплуатации ратрака при достижении «порогов» определенных показателей машины необходимо проводить техническое обслуживание согласно инструкции по эксплуатации.

Существуют различные виды работ, отличающиеся количеством обслуживаемых элементов и сложностью работ, и самые трудные работы в течение гарантийного срока проводит компания-продавец. Также в течение данного срока в обязанности компании-покупателя входит самостоятельное проведение более простых техобслуживаний.

Все техобслуживания имеют определенную периодичность, которая определяется временем, отработанным двигателем. Это время измеряется в «моточасах». Один моточас равен одному часу работы двигателя машины. Непроведение технического обслуживания компанией-покупателем влечет за собой снятие машины с гарантийного срока; и главной проблемой для компании-продавца в данном случае является проверка выполнения обязательств компанией-покупателем.

Способом контроля проведения технического обслуживания покупателем был выбран фотоотчет: на каждом шаге технического обслуживания, завязанном на определенном узле или элементе механику необходимо делать несколько фотографий. Например, при обслуживании съемного фильтра необходимо сфотографировать: установленный необслуженный фильтр, снятый фильтр, обслуженный установленный фильтр.

Для каждой модели контролируемые шаги определяются индивидуально инженерами компании-продавца, поскольку критически важные узлы машины могут меняться. Также могут меняться формы и детали техобслуживаний, и поскольку нет единого стандарта, в соответствии с которым можно было бы шаблонно обрабатывать документы и выделять те или иные техобслуживания, их конкретные шаги и элементы, которые необходимо контролировать, необходимо создать возможность их выделять для дальнейшей обработки.

### 1.2. Анализ аналогичных проектов

В данном случае невозможно проводить анализ аналогов из-за чрезвычайно узкой направленности приложения, прямые или хотя бы косвенные аналоги в свободном доступе в данной предметной области отсутствуют.

Однако есть большое количество приложений более бытового назначения. Примером данного класса приложений может служить приложение «Сервисная книжка автомобиля» [1].

Такие приложения позволяют вести сервисную книгу автомобиля: записывать различные проведенные виды работ, их стоимость, дату проведения и актуальный пробег (рисунок 1, слева), полученные штрафы и оформленные документы с указанием сроков действия и создания напоминаний об обновлении документов или проведении работ (рисунок 1, справа).

Стоит отметить высокую среднюю оценку данного приложения: 4,5 баллов из 5,0 возможных при 595 отзывах. При этом самые низкие отзывы ссылаются на технические проблемы, связанные с рекламой. В большинстве остальных отзывов интерфейс обычно описывается как «приятный», «интуитивный» и «красивый».

Большинство таких приложений специализируется на похожих вещах, однако некоторые из них предлагают возможность проведения диагностики различных компонентов при помощи подключения к электронному блоку управления автомобиля.

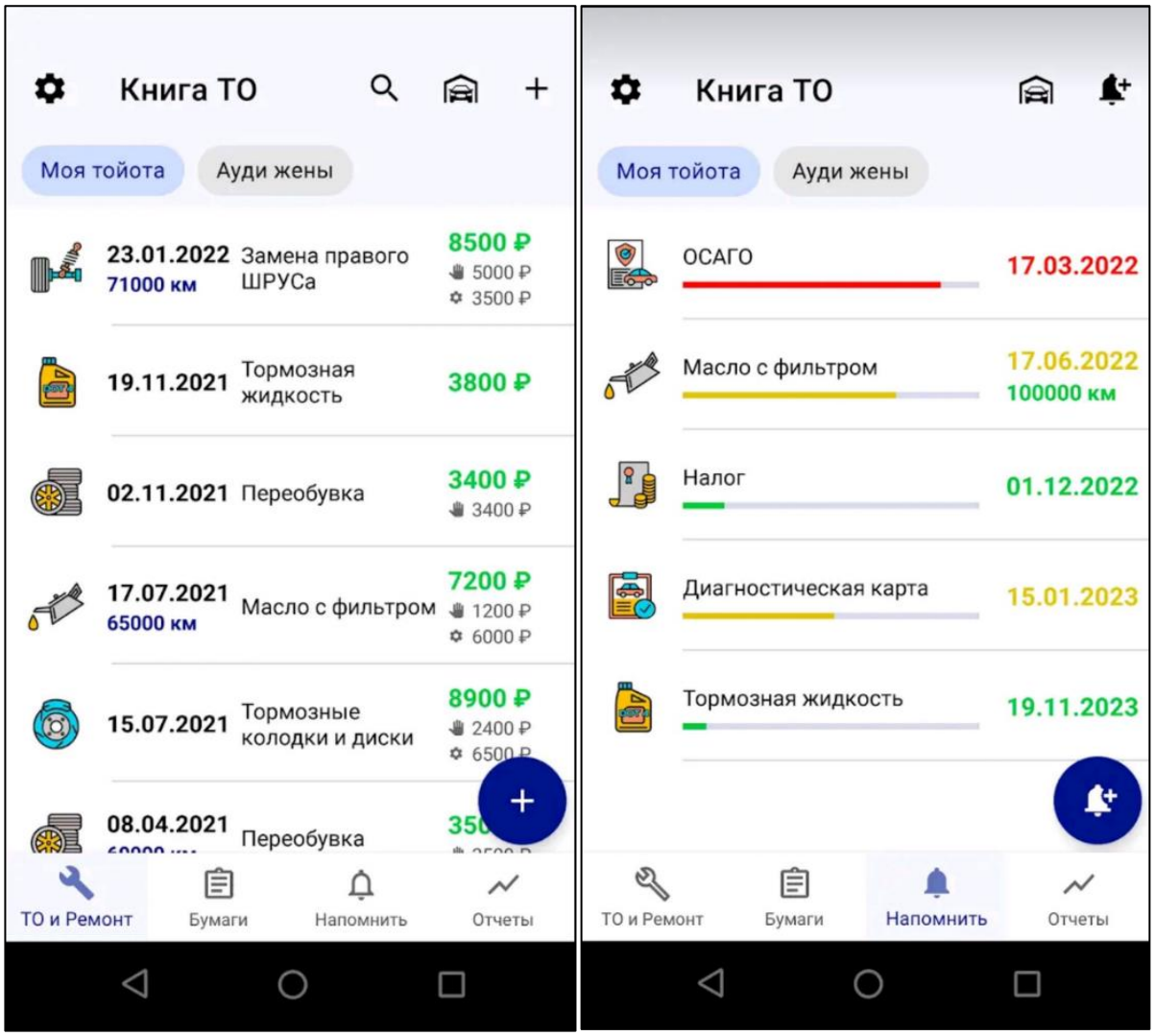


Рисунок 1 – Вкладки «ТО и Ремонт» (слева) и «Напомнить» (справа) приложения «Сервисная книжка автомобиля»

Наиболее близким к приложению, которое необходимо сделать, является приложение «DIY Car Maintenance» [2], в котором есть ряд технических работ, описанных шаг за шагом (рисунок 2).

Стоит также отметить среднюю оценку в 4,0 из 5,0 возможных при 25 тысячах отзывов. Это говорит о том, что функциональность в данном случае намного важнее интерфейса, который важно сделать понятным, пусть и простым.

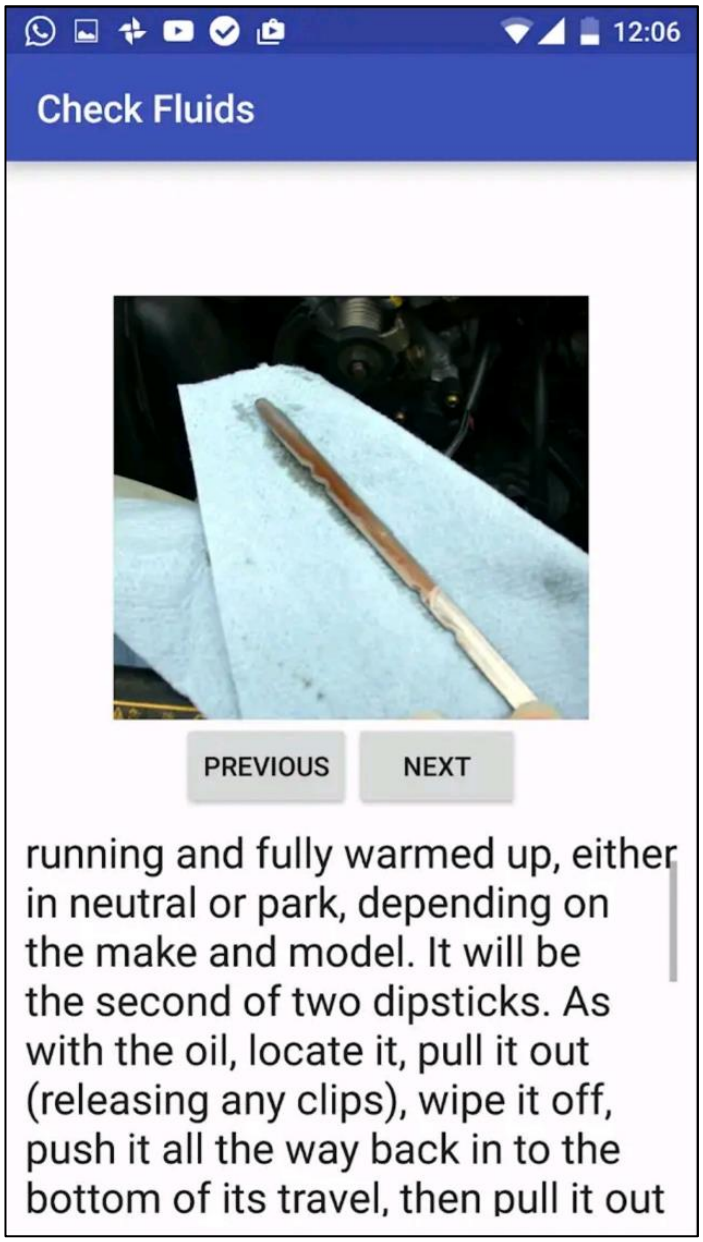


Рисунок 2 – Шаг проверки уровня масла в двигателе
в приложении «DIY Car Maintenance»

Иногда ключевые элементы на изображении выделяются обводкой для привлечения дополнительного внимания пользователя или используются визуальные подсказки, например, стрелки, показывающие направление установки (рисунок 3), движения или вращения (рисунок 4).

Рисунок 3 – Стоп-кадр анимированной иллюстрации установки тормозной колодки с визуальной подсказкой

Рисунок 4 – Стоп-кадр анимированной иллюстрации снятия болтов суппорта с визуальной подсказкой

Также в данном приложении некоторые шаги содержат не статичные изображения, а анимированные, что делает процесс еще более наглядным.

# 2. ПРОЕКТИРОВАНИЕ

**Требования к приложению**

В соответствии с задачей и проведенным анализом аналогов были составлены функциональные и нефункциональные требования к разрабатываемой системе [3]. Функциональные требования определяют функционал и поведение приложения с точки зрения пользователя. Нефункциональные определяют ряд свойств и ограничений, которое приложение должно иметь и соблюдать.

Поставленная задача с точки зрения функциональности делится на две отдельных последовательно выполняемых задачи:

1) инженер компании-продавца должен определить структуру каждого техобслуживания и выбрать подотчетные места для каждого этапа;

2) механик компании-покупателя должен проходить техобслуживания в соответствии с определенной инженером структурой, размещенными данными и требованиями.

Для решения этих задач будут разработаны два различных приложения. Приложение, используемое компанией-покупателем, дальше называется «приложение покупателя», компанией-продавцом – «приложение продавца».

## 2.1. Проектирование приложения продавца

Поскольку данное приложение рассчитано на использование только на рабочем месте инженера компании-продавца, была выбрана реализация в виде настольного приложения.

Были сформулированы требования к приложению. На рисунке 5 представлена диаграмма вариантов использования данного приложения [4]. Пользователем данного приложения является инженер компании-продавца.

**Функциональные требования**

Пользователь должен иметь возможность:

1) создать новое техобслуживание;

2) добавить этап в техобслуживание;

3) добавить изображение в этап;

4) добавить подотчетный элемент в этап;

5) удалить элемент;

6) сохранить данные.

**Нефункциональные требования**

Были установлены следующие нефункциональные требования:

1) данные должны сохраняться без потерь;

2) данные должны сохраняться так, чтобы затем они могли использоваться в виде ресурсов для приложения покупателя.

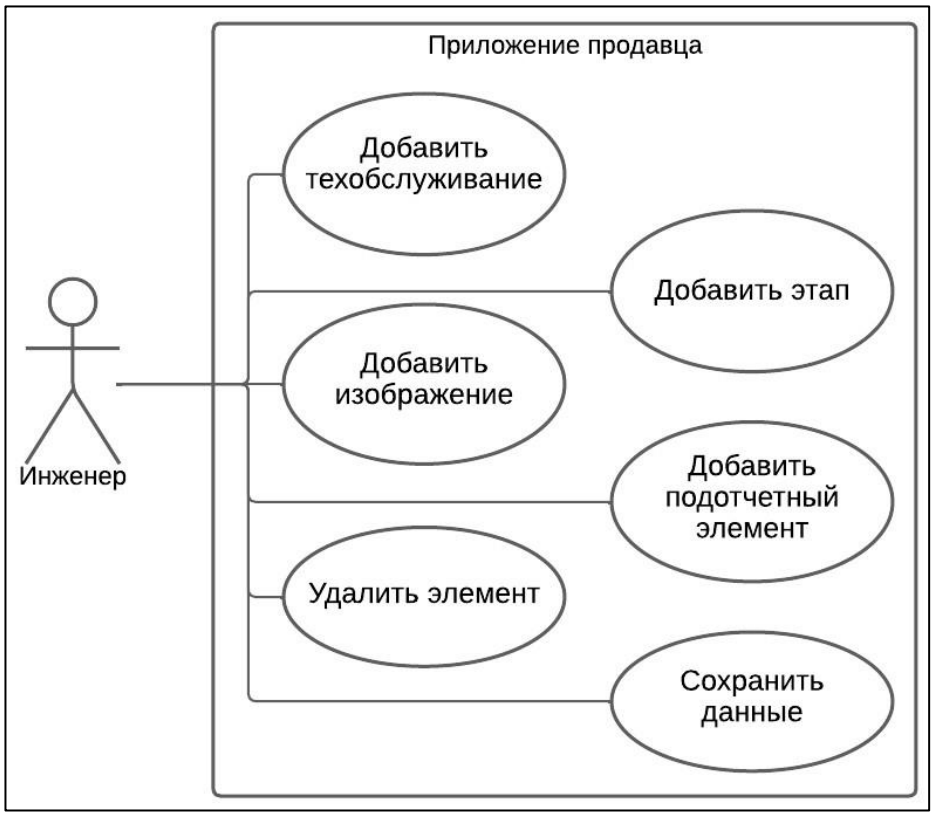


Рисунок 5 – Use Case диаграмма приложения продавца

С системой взаимодействует один актер – инженер.

Добавить техобслуживание: добавление техобслуживания в список. Техобслуживания хранят название и описание, а также шаги.

Добавить этап: добавление этапа в выбранное техобслуживание. Шаги хранят название, описание, иллюстрации, а также требования к элементам отчета.

Добавить изображение: добавление иллюстрации в выбранный шаг.

Добавить подотчетный элемент: добавление подотчетного элемента, в который на момент работы данного приложения можно внести только описание требования к элементу.

Удалить элемент: удаление выбранного элемента или группы.

Сохранить данные: сохранение данных для дальнейшего их использования в качестве ресурсов приложения покупателя.

Для реализации этих требований был спроектирован набор классов, изображенный на спецификационной диаграмме классов (рисунок 6). Данные классы задают структуру и минимальное необходимое наполнение классов приложения покупателя.

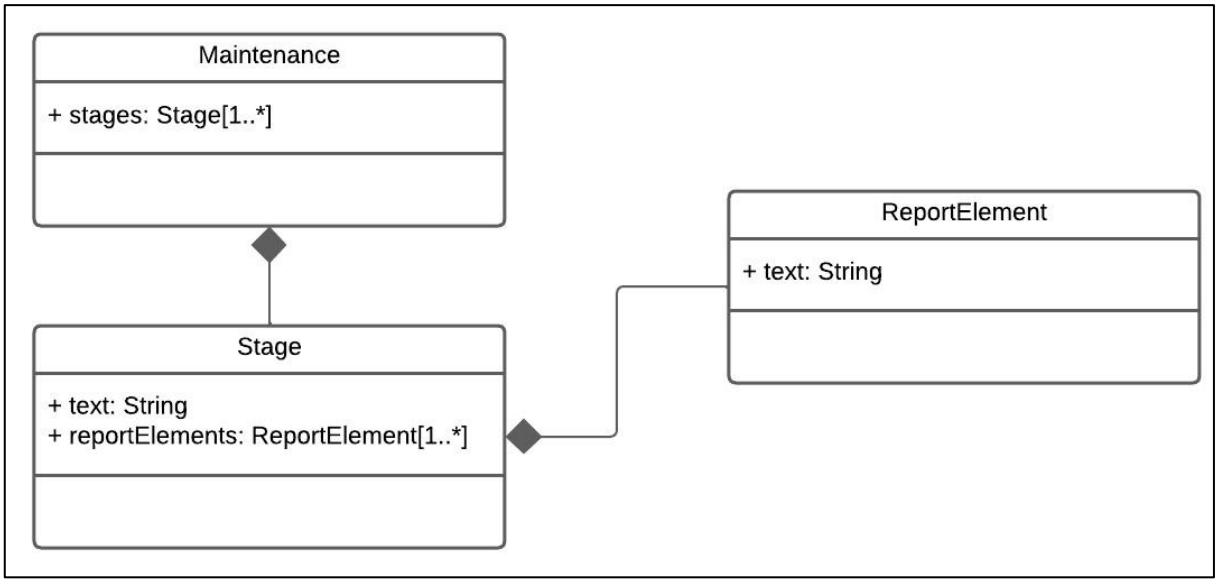


Рисунок 6 – Диаграмма классов приложения продавца

В соответствии с выявленными вариантами использования был разработан макет экрана приложения продавца. Разработка UX-дизайна приложения была проведена с помощью написания разметки XML, что обеспе-

чило максимальное сходство результата с макетом, поскольку эта же разметка используется непосредственно в программе. Макет представлен на рисунке 1 приложения А.

## 2.2. Проектирование приложения покупателя

В данном приложении пользователем является механик компании-покупателя. Совместно с компанией-продавцом были сформулированы функциональные и нефункциональные требования к приложению.

**Функциональные требования**

Пользователь должен иметь возможность:

1) просмотреть инструкцию по эксплуатации машины;
2) просмотреть список техобслуживаний, которые должна проводить компания-покупатель;
3) выбрать из списка техобслуживание для просмотра необходимых шагов и фотоотчета;
4) запустить прохождение техобслуживания;
5) завершить прохождение техобслуживания.

**Нефункциональные требования**

Были установлены следующие нефункциональные требования:

1) приложение должно работать на операционной системе Android 13 и выше;
2) в приложении должен быть простой интерфейс;
3) файл инструкции по эксплуатации должен храниться локально (на устройстве) после первого запуска;
4) при прохождении фотоотчета из-под приложения должна использоваться камера устройства;
5) приложение должно следить за моточасами машины посредством запроса данной информации от сервера;

6) вся подотчетная информация техобслуживания должна быть сохранена в базе данных под управлением СУБД PostgreSQL на сервере компании.

### 2.2.1. Варианты использования приложения покупателя

Согласно требованиям, была разработана диаграмма вариантов использования, которая изображена на рисунке 7.

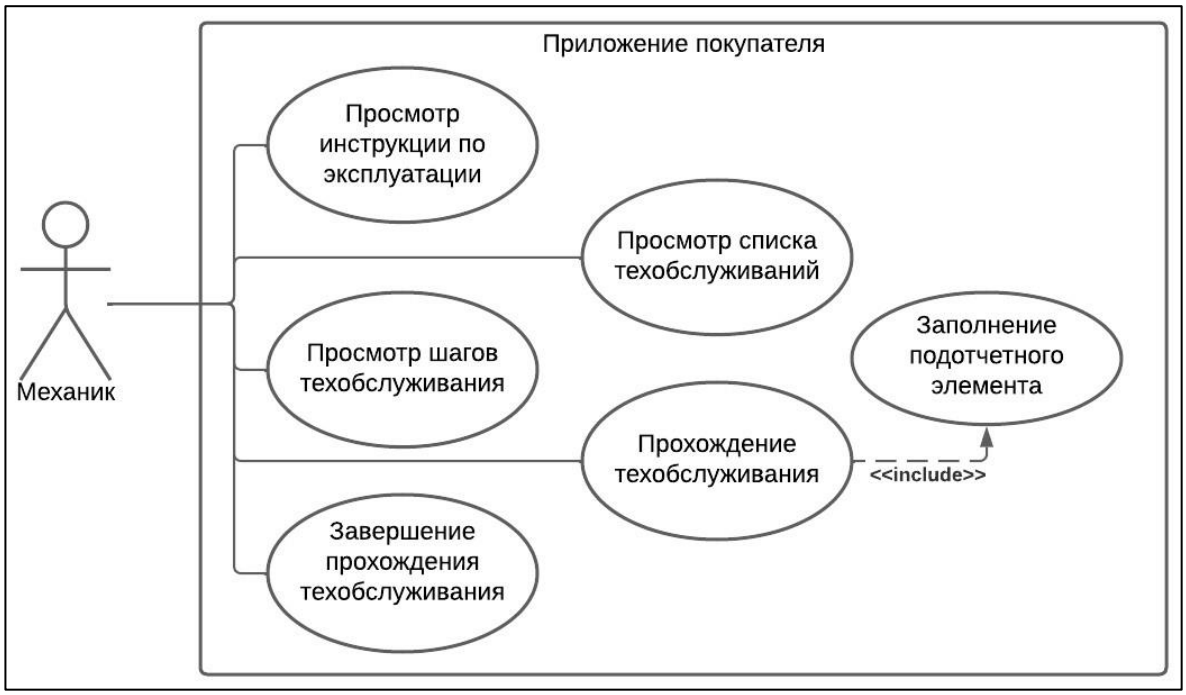


Рисунок 7 – Use Case диаграмма приложения покупателя

С системой взаимодействует только один актер – механик.

Прохождение техобслуживания: в зависимости от того, есть ли незавершенное прохождение техобслуживания либо запускает прохождение, либо позволяет продолжить прохождение техобслуживания. Это необходимо, так как техобслуживание может проводиться не за одну сессию. Данной функцией не получится злоупотреблять, поскольку неадекватное время прохождения техобслуживания будет заметно по данным на сервере.

Заполнение подотчетного элемента: поскольку в качестве способа контроля был выбран фотоотчет, позволяет добавить в элемент отчета фотографию, сделать которую необходимо с помощью камеры устройства, открытой через данное приложение.

Завершение прохождения техобслуживания: при наличии незавершенного техобслуживания система завершает его прохождение. При наличии пустых элементов отчета система должна вывести предупреждение.

### 2.2.2. Архитектура системы

Учитывая, что данные о текущем количестве моточасов хранятся в базе данных и обновляются посредством SQL-запросов к СУБД со стороны сервиса мониторинга ратрака, находящемся на том же сервере, архитектура системы представлена в виде UML диаграммы компонентов (рисунок 8) [5].

В архитектуре системы отсутствует приложение продавца, поскольку результат его работы – файлы, использующиеся в качестве ресурсов приложения покупателя – является фактически неотделимой частью приложения покупателя.

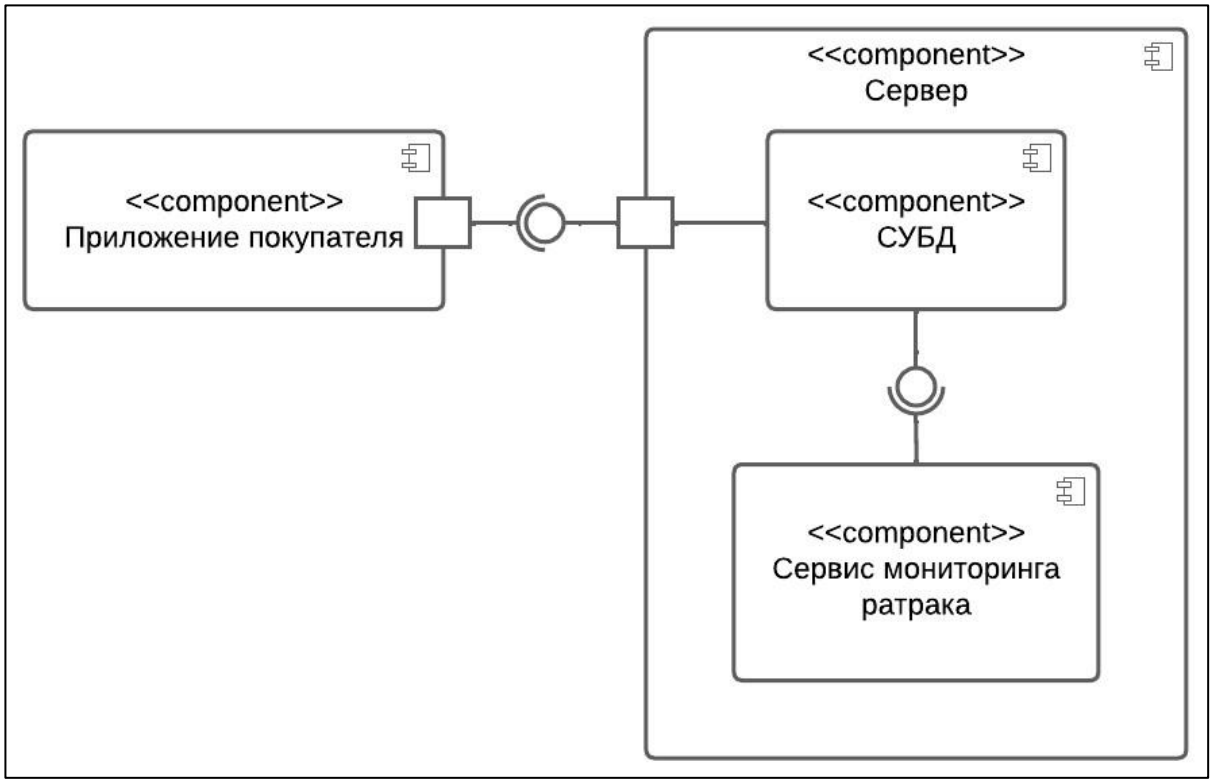


Рисунок 8 – Диаграмма компонентов

### 2.2.3. Построение схемы данных

Для хранения информации была построена схема базы данных, изображенная на рисунке 9. Данная схема построена с помощью применения модели «сущность-связь» в нотации Мартина («вороньи лапки») [6].

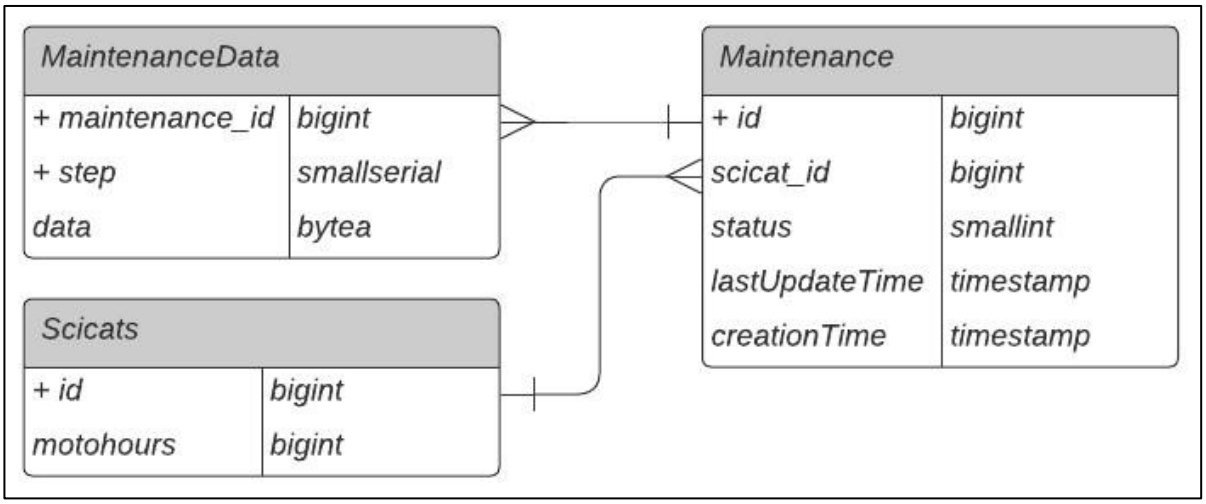


Рисунок 9 – Схема данных

Согласно данной схеме база данных состоит из следующих таблиц: Scicats, Maintenance, MaintenanceData.

Scicats: хранит данные об идентификаторе каждого ратрака (id) и его моточасы (motohours) на данный момент (исходя из предположения, что ратрак не заводится менее, чем на час).

Maintenance: хранит данные об идентификаторе техобслуживания (id); ратраке, над которым оно проводится (scicat_id); текущем статусе техобслуживания (status), последнем добавлении подотчетной информации в данные техобслуживания (lastUpdateTime) и времени создания техобслуживания (creationTime).

MaintenaceData: составной ключ из идентификатора техобслуживания (maintenance_id) и номера шага (step); сериализованные подотчетные элементы в виде массива байтов (data).

### 2.2.4. Диаграмма классов приложения покупателя

Для написания программы была выбрана объектно-ориентированная парадигма программирования [7]. Итоговая структура классов отображена на спецификационной диаграмме классов (рисунок 10). Данная структура классов является расширенным вариантом структуры классов приложения продавца.

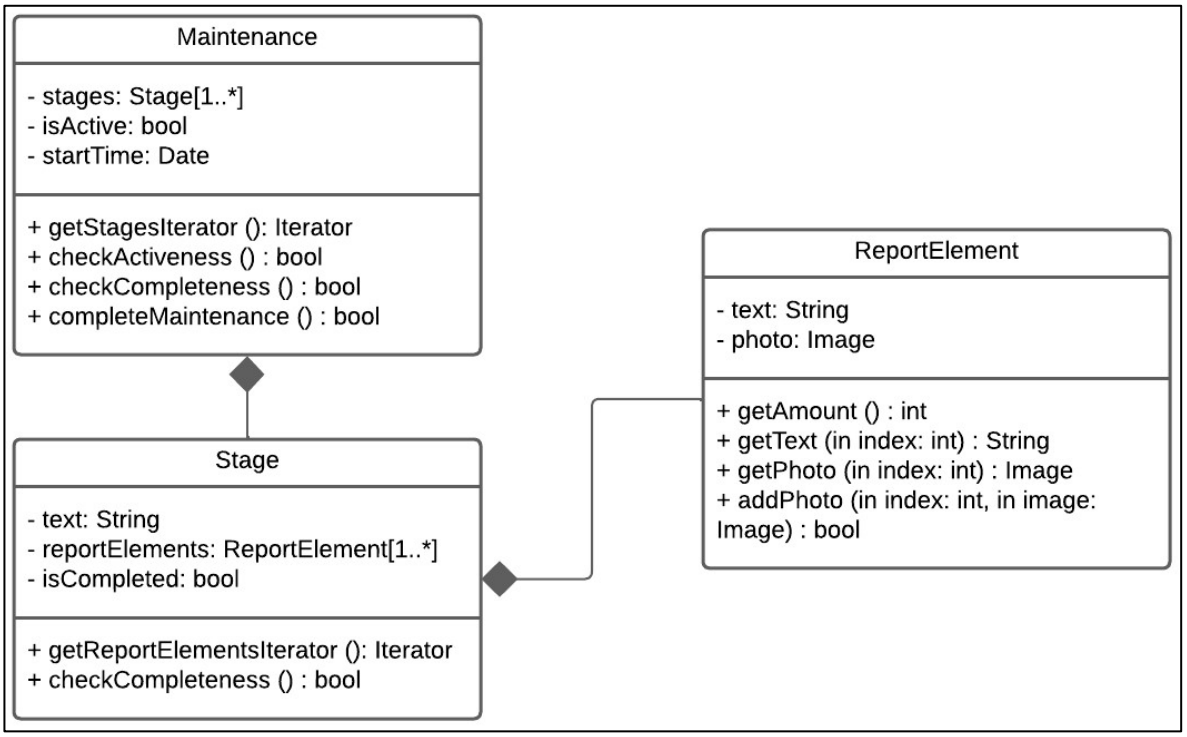


Рисунок 10 – Диаграмма классов приложения покупателя

Благодаря инкапсуляции при изменении требований к данным, хранящихся в элементах техобслуживания будет необходимо переписать только поля класса ReportElement, хранящие информацию и функции, связанные с присваиванием данным полям значений, а также те части программы, которые используют данный функционал.

### 2.2.5. Проектирование интерфейса приложения покупателя

В соответствии с выявленными требованиями и вариантами использования были разработаны макеты экранов приложения покупателя: экрана

списка техобслуживания (рисунок 11) и экрана шага техобслуживания (рисунок 12) [8]. Разработка UX-дизайна приложения была проведена в вебсервисе Figma.

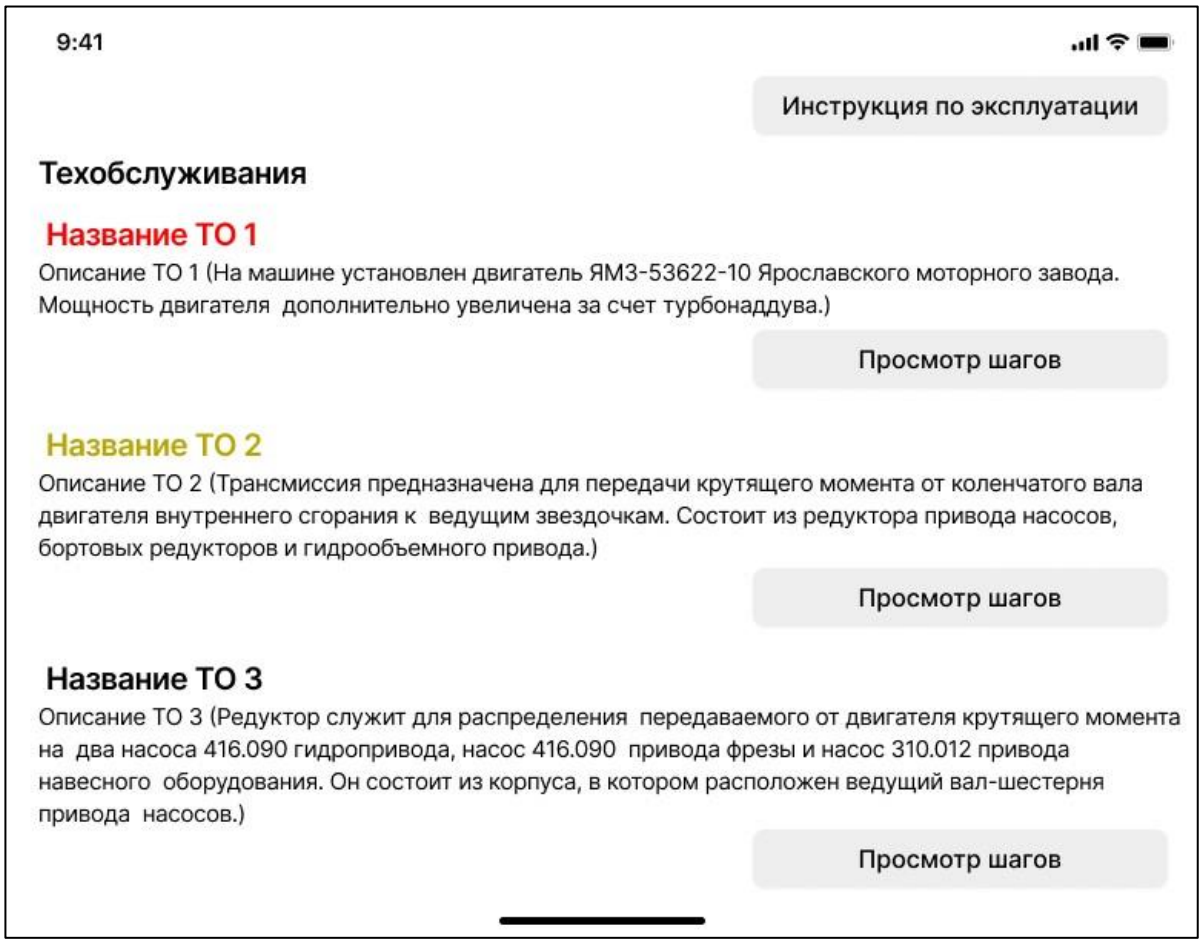


Рисунок 11 – Макет экрана списка техобслуживаний

На экране шага техобслуживания наполнение кнопки «Запуск / завершение ТО» и ее видимость должны зависеть от того, запущено ли уже данное техобслуживание (наполнение) и, если запущено, заполнены ли все подотчетные элементы (видимость). При нажатии на рамки требований к элементу фотоотчета должна открываться камера. После возврата на данный экран полученное фото должно отображаться вместо требования.

Видимости кнопок «Предыдущий шаг» и «Следующий шаг» зависят от того, существует ли соответствующий шаг относительно текущего. Иллюстрации и подотчетные элементы должна быть возможность пролистывать горизонтально, если одновременно они не помещаются на экран.


Рисунок 12 – Макет экрана шага техобслуживания

# 3. РЕАЛИЗАЦИЯ

## 3.1. Выбор программных средств реализации

Настольное приложение продавца будет реализовано на языке программирования C# с помощью Windows Presentation Foundation (WPF) [9].

Выбор инструментов для реализации приложения покупателя является более сложной задачей. В таблице 1 представлено сравнение современных языков программирования и инструментариев, используемых в комбинации для разработки приложений под операционную систему Android.

Таблица 1 – Сравнение языков программирования под ОС Android

| № | Наименование | Достоинства | Недостатки |
|---|---|---|---|
| 1 | Java | Строгая типизация, официальная поддержка Android Studio, кроссплатформенность, безопасность исполнения за счет архитектурных ограничений языка; простота (относительно C, C++); наличие большого количества open-source библиотек; «сильное» сообщество [10] | Система лицензирования; низкая производительность (по сравнению с C, C++); энергозатратность; большое количество кода (по сравнению с C++, Python) |
| 2 | Kotlin | Наследует большинство достоинств Java; совместимость с библиотеками Java; транслируемость в Java и обратно; официальная поддержка Android Studio; «корутины» (упрощает и оптимизирует асинхронное программирование) [11] | Небольшое сообщество; малое количество обучающих материалов; время компиляции; требует знания Java |
| 3 | C# (с использованием Xamarin) | Кроссплатформенность; удобство создания графического интерфейса; официальная поддержка Microsoft; дополнительная поддержка и возможности при покупке платных тарифов [12] | Стоимость при коммерческом использовании; часть возможностей официальной IDE доступна только с подпиской; низкая производительность (по сравнению с C, C++) |
| 4 | C/C++ (с использованием NDK) | Высокая производительность | Сложность написания программ; безопасность исполнения зависит только от разработчика; меньшее количество библиотек; сложность документации |
| 5 | Python | Разнообразие активных проектов, позволяющих использовать Python на Android [13] | Отсутствие инструментария, официально поддерживаемого разработчиками Python; низкая производительность |

С 2019 года (и на данный момент) рекомендуемым языком программирования для написания мобильных приложений является Kotlin [14]. Также поскольку требовательность задачи к производительности не является критическим аспектом, именно Kotlin и был выбран в качестве языка программирования для реализации приложения покупателя.

### 3.2. Реализация приложения продавца

В данном решении используются только стандартные библиотеки и элементы. Среди использованных стандартных элементов, таких как `Window`, `Grid`, `GridSplitter`, `Menu`, `Button` и `TreeView` особым образом стоит рассмотреть `TreeView` вследствие нетривиальности проблемы, возникшей при работе с данным элементом.

Для обработки сохранения предыдущего элемента и загрузки выбранного элемента дерева, которыми могут являться техобслуживание или шаг техобслуживания, к стандартным событиям `Unselected` и `Selected` были добавлены обработчики события. Обработчик события `Unselect` для элемента шага представлен в листинге 1, для элемента техобслуживания – в листинге 2.

Листинг 1 – Код обработчика события `Unselect` для элемента шага

```
private void SaveStageOnUnselect(object sender, RoutedEventArgs e)
{
    Stage stage = (Stage)((TreeViewItem)sender).Resources["source"];
    if (stage == null)
        return;
    stage.name = tb_curElementName.Text;
    stage.text = tb_curElementText.Text;
    stage.images.Clear();
    foreach (Image image in g_curStepImages.Children)
    {
        stage.images.Add(image);
    }
    stage.reportElements.Clear();
    foreach (TextBox reportElement in g_reportElements.Children)
    {
        stage.reportElements.Add(reportElement.Text);
    }
    RefreshDataTree();
}
```

Обработчики для элемента шага и элемента техобслуживания (листинг 2) отличаются. Главное их отличие, помимо работы с данными, состоит в том, что обработчик для элемента техобслуживания содержит в себе проверку, было ли вызвано данное событие именно элементом техобслуживания.

В стандартном элементе `TreeView` могут храниться только элементы `TreeViewItem`. При этом `TreeViewItem` также может хранить в себе `TreeViewItem`. В данной вложенной структуре при возникновении события `Selected` у одного из дочерних `TreeViewItem`, данное событие будет также поочередно вызвано у всех родительских элементов.

Если после обработки события элемента шага будет происходить обработка события элемента техобслуживания, то к данным шага пользователь не будет иметь доступа. Именно по этой причине в специальное поле Tag при создании элемента назначается его тип, выбираемый из пользовательского перечисления `TreeViewItemType`, который затем используется в этой проверке.

Листинг 2 – Код обработчика события `Unselect` для элемента ТО

```
private void SaveStageOnUnselect(object sender, RoutedEventArgs e)
{
    Stage stage = (Stage)((TreeViewItem)sender).Resources["source"];
    if (stage == null)
        return;
    stage.name = tb_curElementName.Text;
    stage.text = tb_curElementText.Text;
    stage.images.Clear();
    foreach (Image image in g_curStepImages.Children)
    {
        stage.images.Add(image);
    }
    stage.reportElements.Clear();
    foreach (TextBox reportElement in g_reportElements.Children)
    {
        stage.reportElements.Add(reportElement.Text);
    }
    RefreshDataTree();

}
```

Как и было указано в проектировании макета, при реализации использовалась та же разметка XML. Получившийся интерфейс представлен на рисунке 2 приложения А. Добавление изображения происходит посредством копирования изображения из буфера обмена.

### 3.3. Реализация приложения покупателя

При реализации были использованы нестандартные библиотеки. Ktorm: предоставляет функционал взаимодействия с базой данных. CameraX: предоставляет функционал взаимодействия с камерой устройства; является рекомендуемым решением при работе с камерой устройства.

В реализации приложения покупателя отдельно стоит рассмотреть необходимость отображения неопределенного количества элементов: техобслуживаний в списке; изображений и подотчетных элементов в шагах.

Для отображения необходимо использовать элемент `RecyclerView`. Главной проблемой в данном случае является написание адаптера, используемого `RecyclerView` для отображения элементов, содержащихся в привязанной структуре данных [15]. Для этого необходимо создать собственный класс, наследуемый от класса `RecyclerView.Adapter<VH extends RecyclerView.ViewHolder>`. Обязательными для переопределения при этом являются методы `onCreateViewHolder`, `getItemCount` и `onBindViewHolder`.

Адаптер для списка техобслуживаний представлен в листинге 3.

Листинг 3 – Код адаптера `RecyclerView` для списка техобслуживаний

```
class MaintenancesAdapter(private val maintenances: Array<Maintenance?>,
private val context: Context): RecyclerView.Adapter<Mainte-
nancesAdapter.MyViewHolder>() {
    class MyViewHolder(view: View): RecyclerView.ViewHolder(view) {
        val name: TextView = view.findViewById(R.id.mainte-
nance_in_list_name)
        val description: TextView = view.findViewById(R.id.mainte-
nance_in_list_description)
        val button: MaterialButton = view.findViewById(R.id.mainte-
nance_in_list_button)
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
```

```
        val view = LayoutInflater.from(parent.context).inflate(R.lay-
out.maintenance_in_list, parent, false)
        return MyViewHolder(view)
    }
    override fun getItemCount(): Int {
        return maintenances.size
    }
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        holder.name.text = maintenances[position]!!.name
        holder.description.text = maintenances[position]!!.description
        holder.button.setOnClickListener {
            val intent = Intent(context, StagesActivity::class.java)
            intent.putExtra("stages", maintenances[position]!!.stages)
            context.startActivity(intent)
        }
    }
}
```

Также при обновлении структуры данных, которая определена источником данных для адаптера необходимо оповещать адаптер об этом. Пример такого оповещения представлен в листинге 4 методом `notifyDataSetChange`.

Листинг 4 – Код функции обновления данных разметки `Activity` шага

```
private fun refresh(stageIndexShift: Int = +0) {
        stageIndex += stageIndexShift
        val curStage: Stage = stages[stageIndex]
        name.text = curStage.name
        scroll.fullScroll(View.FOCUS_UP)
        description.text = curStage.text
        illustrationsListSource.clear()
        illustrationsListSource.addAll(curStage.imagesResourcesID)
        illustrationsList.adapter?.notifyDataSetChanged()
        reportElementsListSource.clear()
        reportElementsListSource.addAll(curStage.reportElement!!)
        reportElementsList.adapter?.notifyDataSetChanged()
        if (stageIndex == 0 && prevStage.visibility == View.VISIBLE)
            prevStage.visibility = View.INVISIBLE
        if (stageIndex == stages.size - 1 && nextStage.visibility ==
View.VISIBLE)
            nextStage.visibility = View.INVISIBLE

}
```

Полученный интерфейс приложения покупателя (рисунок 13) отличается от макета главным образом вследствие того, что соотношение сторон отображаемого изображения может сильно различаться с соотношением сторон выделенного под него места.

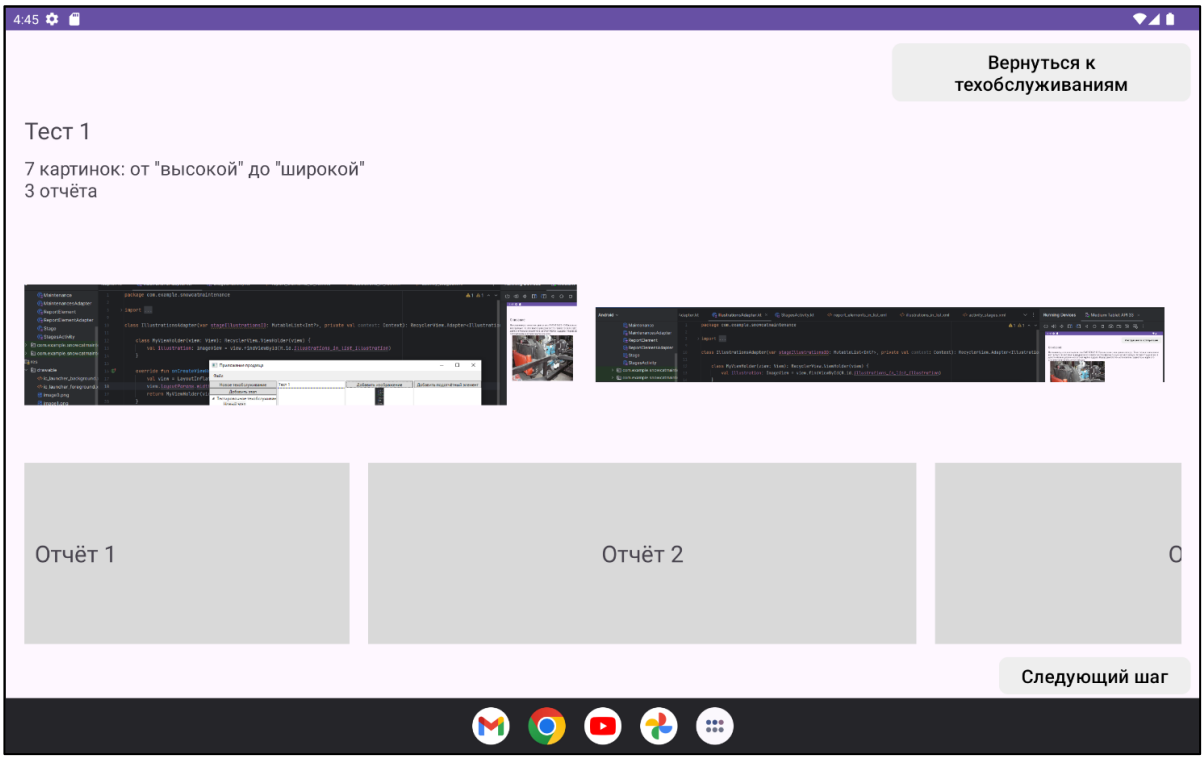


Рисунок 13 – Интерфейс пользователя приложения покупателя (шаг)

Для сравнения приведен вариант полученного интерфейса при подстановке более подходящих по формату изображений (рисунок 14).

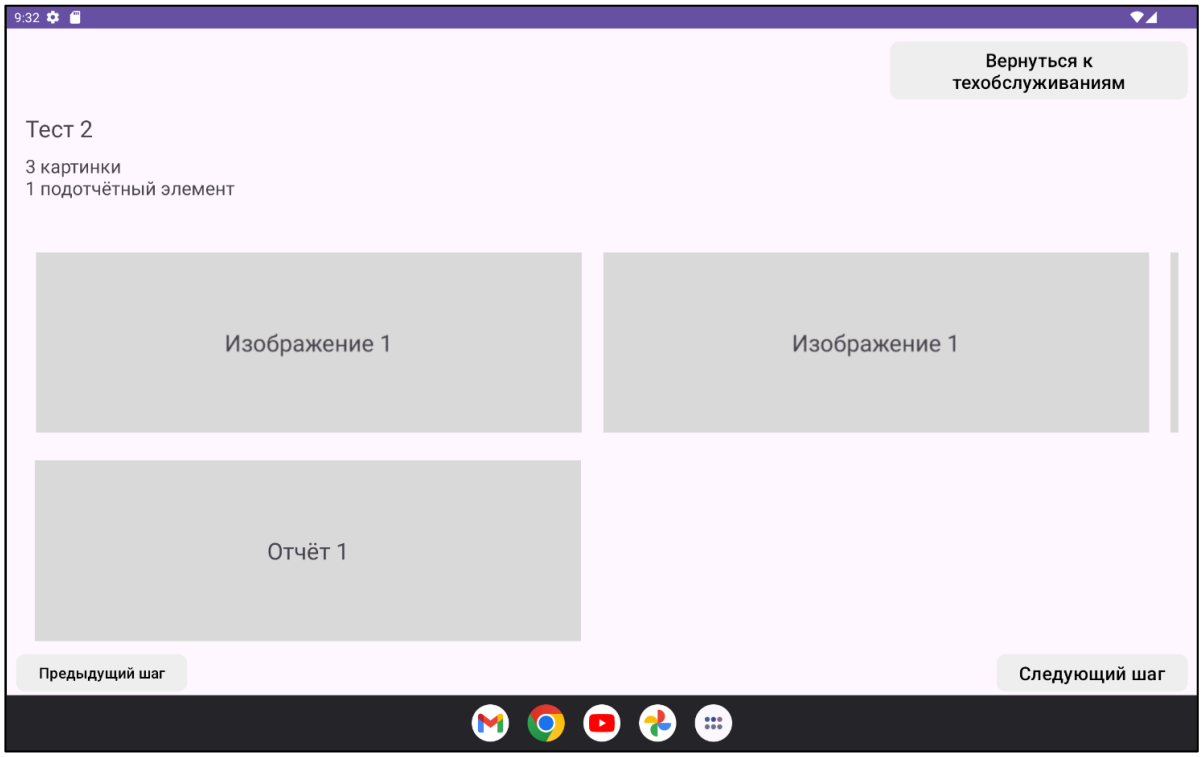


Рисунок 14 – Интерфейс пользователя приложения покупателя (шаг) при подстановке более подходящих по формату изображений

# 4. ТЕСТИРОВАНИЕ

## Функциональное тестирование

Для тестирования корректности работы программы с точки зрения пользователя на основе функциональных требований были проведены функциональные тестирования приложений покупателя (таблица 2) и продавца (таблица 3).

Таблица 2 – Функциональное тестирование приложения покупателя

| № | Название теста | Предусловия | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Добавление ТО | Пользователь нажал на кнопку «Новое техобслуживание» | Приложение создало новое техобслуживание с новым этапом | Да |
| 2 | Удаление ТО | Выбран элемент техобслуживания; пользователь нажал на кнопку «Удалить» | Выбранное техобслуживание удалено | Да |
| 3 | Добавление этапа | Выбран элемент техобслуживания; пользователь нажал на кнопку «Добавить этап» | В выбранном техобслуживании создан новый этап | Да |
| 4 | Удаление этапа | Выбран элемент этапа техобслуживания; пользователь нажал на кнопку «Удалить» | Выбранный этап удален | Да |
| 5 | Добавление изображения | Пользователь нажал на кнопку «Добавить изображение» | Добавлено изображение из буфера обмена | Да |
| 6 | Удаление всех изображений | Пользователь нажал на кнопку «Удалить все изображения» | Удалены все изображения | Да |
| 7 | Добавление подотчетного элемента | Пользователь нажал на кнопку «Добавить подотчетный элемент» | Добавлен новый подотчетный элемент | Да |
| 8 | Сохранение данных | Пользователь выбрал в меню «Сохранить» | Сформированные данные сохранены по относительному пути «CustomData/» | Да |

Таблица 3 – Функциональное тестирование приложения покупателя

| № | Название теста | Предусловия | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Просмотр инструкции | Пользователь нажал на кнопку «Инструкция по эксплуатации» | Приложение вывело инструкцию по эксплуатации на экран | Да |

Окончание таблицы 3

| № | Название теста | Предусловия | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 2 | Просмотр списка ТО | Пользователь запустил приложение | Приложение вывело список ТО на главном экране | Да |
| 3 | Просмотр шагов ТО | Пользователь нажал на кнопку «Просмотр шагов» | Приложение вывело экран шага техобслуживания с его элементами | Да |
| 4 | Прохождение ТО | Пользователь нажал на кнопку «Запуск ТО» | Приложение готово к заполнению пользователем подотчетных элементов | Да |
| 5 | Заполнение подотчетного элемента | Пользователь нажал на подотчетный элемент | Приложение открыло экран камеры | Да |
| 6 | Завершение ТО | Пользователь нажал на кнопку «Завершение ТО» | Приложение сформировало запрос на добавление данных в базу данных | Да |

**Интеграционное тестирование**

Данное тестирование необходимо для проверки сохранения целостности данных: верного сохранения приложением продавца и верного распознавания приложением покупателя, отсутствии потерь качества изображений. Таким образом, в данном тестировании необходим всего 1 тест (таблица 4).

Таблица 4 – Интеграционное тестирование

| Название теста | Предусловия | Ожидаемый результат | Тест пройден? |
|---|---|---|---|
| Проверка целостности данных | Сформирован и сохранен список техобслуживаний, их шаги и элементы шагов; сохраненные данные использованы в качестве ресурсов приложения | Приложение покупателя верно распознает все данные, данные не повреждены | Да |

## ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана система для контроля технического обслуживания ратраков. При этом были решены следующие задачи:

1) анализ аналогичных проектов;
2) формирование требований к приложению;
3) проектирование приложений;
4) проектирование архитектуры системы;
5) реализация приложений;
6) тестирование приложений.

В будущем планируется сбор обратной связи как для приложения продавца, так и для приложения покупателя, доработка приложений и архитектуры системы, доведение системы до внедрения в промышленную эксплуатацию. Главной проблемой в данном случае является сезонность использования получившегося решения: техобслуживания, привязанные к моточасам, не проводятся во время простоя техники, то есть, в среднем, полгода. В таком случае, способов получения отзывов представляется всего два. Первый – тестирование на контрольной группе, что в данном случае является чересчур затратным и неоправданным решением. Второй – запрос внеочередного проведения клиентом одного из техобслуживаний, Данный способ может негативно повлиять на деловые отношения с клиентом, поэтому его применение требует особой осторожности.

**ЛИТЕРАТУРА**

1. Сервисная книжка автомобиля. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.cars.android.carapps.carnotes&hl=ru&gl=US (дата обращения: 17.05.2024 г.).

2. DIY Car Maintenance. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.csw.engineer.Diycarmanuals&hl=en_US (дата обращения: 17.05.2024 г.).

3. Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. // М.: Вильямс, 2002. – 448 с.

4. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. // М.: ДМК Пресс, 2006. – 253 с.

5. Бондаренко М.Ф. Моделирование и проектирование бизнес-систем: методы, стандарты, технологии / М.Ф. Бондаренко, С.И. Маторин, Е.А. Соловьева // Харьков: Компания СМИТ, 2004. – 272 с.

6. Мюллер Р. Дж. Базы данных и UML проектирование. – М.: Изд-во Лори, 2002. – 420 с.

7. Бадд Т. Объектно-ориентированное программирование в действии. – СПб.: Питер, 1997. – 83 с.

8. Айрапетян Г. М. Дизайн мобильного приложения // Молодой ученый, 2018. – Т. 48. – С. 12–15.

9. Desktop Guide (WPF .NET). [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=net-desktop-8.0 (дата обращения: 17.05.2024 г.).

10. Overview of Java. [Электронный ресурс] URL: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/jjdev/Java-overview.html (дата обращения: 17.05.2024 г.).

11. Kotlin for Android. [Электронный ресурс] URL: https://kotlinlang.org/docs/android-overview.html (дата обращения: 17.05.2024 г.).

12. What is Xamarin? [Электронный ресурс] URL: https://learn.microsoft.com/en-us/previous-versions/xamarin/get-started/what-is-xamarin (дата обращения: 17.05.2024 г.).

13. Список активных проектов для работы Python на Android. [Электронный ресурс] URL: https://wiki.python.org/moin/Android (дата обращения: 17.05.2024 г.).

14. Android’s Kotlin-first approach. [Электронный ресурс] URL: https://developer.android.com/kotlin/first#:~:text=At%20Google%20I%2FO%202019,easily%20integrates%20into%20existing%20apps (дата обращения: 17.05.2024 г.).

15. Adapter for RecyclerView. [Электронный ресурс] URL: https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter (дата обращения: 17.05.2024 г.).

**ПРИЛОЖЕНИЕ. Скриншоты приложения продавца**

На рисунках 1–2 приведены скриншоты макета и получившегося интерфейса приложения продавца.

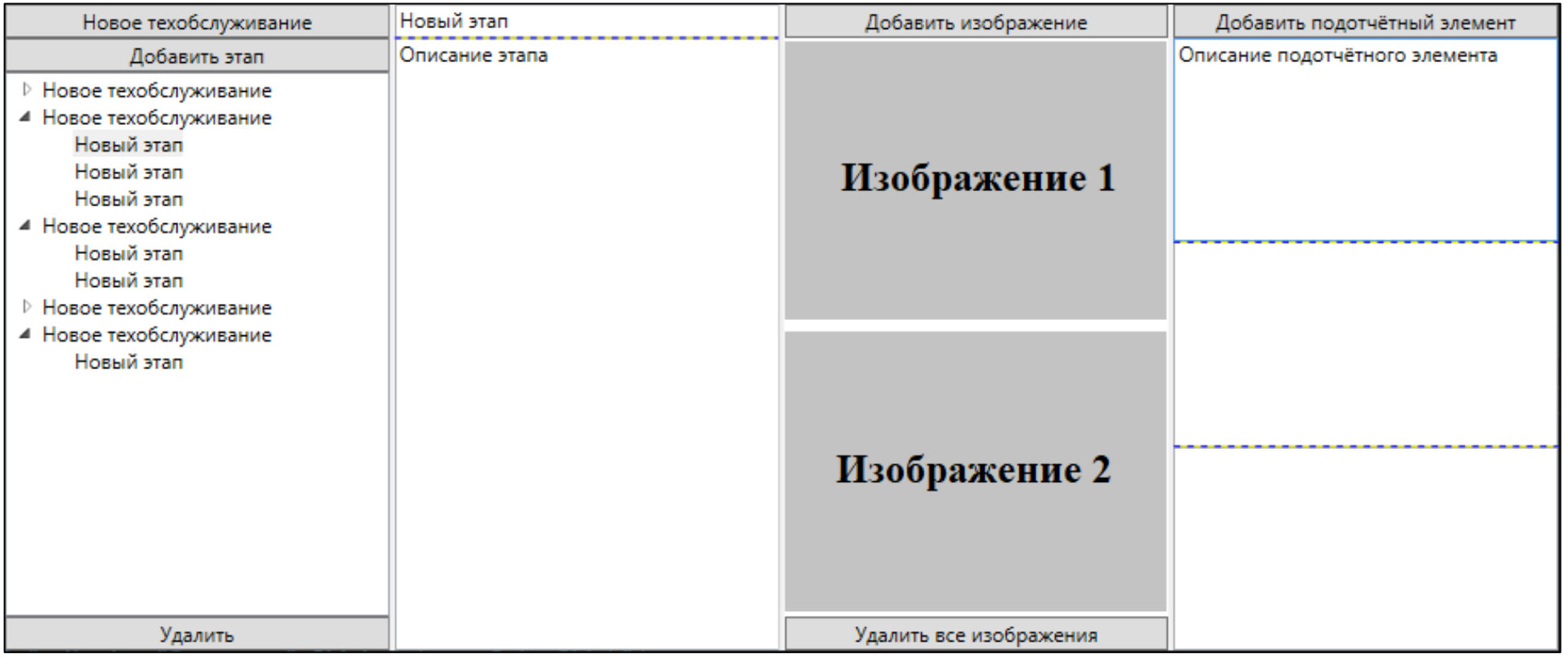


Рисунок 1 – Макет основного экрана приложения продавца

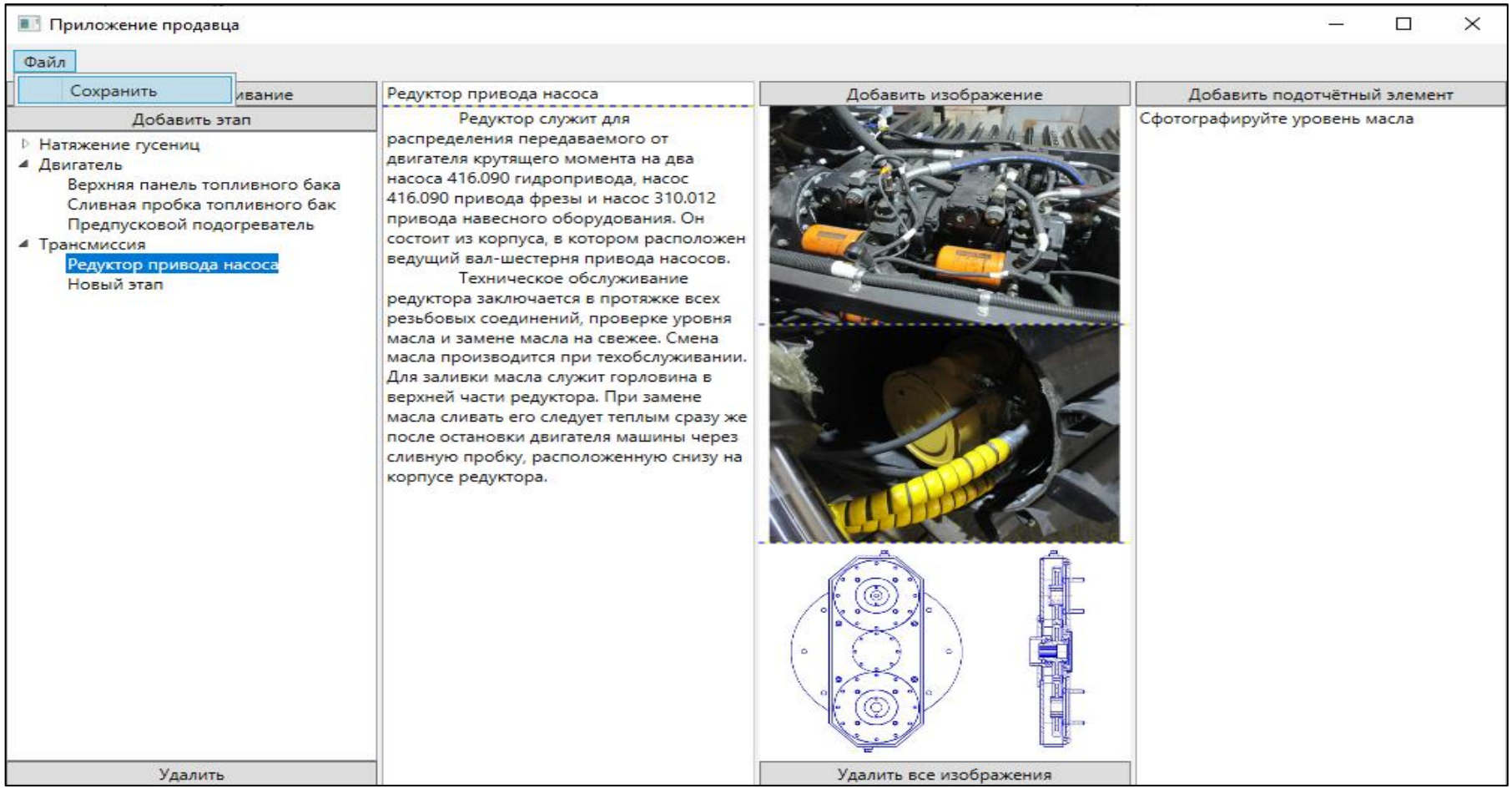


Рисунок 2 – Пользовательский интерфейс приложения продавца