**Департамент образования И НАУКИ города Москвы**

**ГБПОУ г. МОСКВЫ МОСКОВСКИЙ АВТОМОБИЛЬНО – ДОРОЖНЫЙ**

**КОЛЛЕДЖ им. А. А. НИКОЛАЕВА**

**КУРСОВОЙ ПРОЕКТ**

**Тема: Разработка программного обеспечения «Компания по продаже и обслуживанию аккумуляторов»**

Студент\_\_\_\_Кузьмин Д.Д.

Преподаватель\_\_\_\_Яковлев М.С.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Москва

2023

**Департамент образования И НАУКИ города Москвы**

**ГБПОУ г. МОСКВЫ МОСКОВСКИЙ АВТОМОБИЛЬНО – ДОРОЖНЫЙ**

**КОЛЛЕДЖ им. А. А. НИКОЛАЕВА**

**З А Д А Н И Е**

к курсовому проекту

студент дневного отделения, группы 4ИП

специальности 09.02.07 «Информационные системы и программирование»

Фамилия, имя, отчество Кузьмин Даниил Денисович

Тема курсового проекта Разработка программного обеспечения «Компания по продаже и обслуживанию аккумуляторов»

1. Постановка задачи:

Основной целью курсового проекта является Разработка программного обеспечения «Компания по продаже и обслуживанию аккумуляторов», которое будет оптимизировать учет продаж и обслуживания аккумуляторов для автомобилей.

Функционал программного обеспечения должен обладать возможностью уникального доступа в зависимости от роли пользователя, осуществлять продажу аккумулятора и обслуживание, а также информация о пользователях.

2. Исходные данные:

Информация о характеристиках аккумулятора, обслуживании, состоянии, методах оплаты, типах доставки, должностях, типах пользователей, статусах продажи.

3. Перечень вопросов, подлежащих разработке в курсовом проекте:

1. Постановка задачи и исследование предметной области;
2. Обоснование выбора ОС, СУБД и языка программирования;
3. Анализ разработанных программных обеспечений
4. Проектирование Use-case диаграммы;
5. Проектирование концептуальной модели данных;
6. Проектирование информационно-логической модели данных;
7. Проектирование физической модели данных;
8. Проектирование интерфейса пользователя. ­
9. Тестирование и обработка ПО.

Задание принял к исполнению « » \_\_\_\_\_\_ 2023 г.

Студент Д.Д. Кузьмин / И.О. Фамилия подпись

Срок сдачи « » 2023 г.

Руководитель курсового проекта М.С. Яковлев / И.О. Фамилия подпись

Задание рассмотрено на заседании цикловой комиссии « » 2023 г.

Председатель ПЦК / Т.Г. Сянина\_\_\_\_ подпись И.О. Фамилия,

**СОДЕРЖАНИЕ**

[СПИСОК СОКРАЩЕНИЙ 4](#_Toc127530563)

[ВВЕДЕНИЕ 6](#_Toc127530564)

[1.ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ 7](#_Toc127530565)

[1.1 Основные аспекты разработки ПО 7](#_Toc127530566)

[1.2 Анализ разработанных ПО 14](#_Toc127530567)

[2.РАЗДЕЛ ПРОЕКТИРОВАНИЯ 17](#_Toc127530568)

[2.1 Постановка задач 17](#_Toc127530569)

[2.2 Описание предметной области 17](#_Toc127530570)

[2.3 Анализ обоснования выбора СУБД 17](#_Toc127530571)

[2.4 Технические требования к разрабатываемому ПО 24](#_Toc127530572)

[2.5 Разработка интерфейса программного продукта 29](#_Toc127530573)

[3.ЭКСПЕРЕМЕНТАЛЬНЫЙ РАЗДЕЛ 44](#_Toc127530574)

[3.1 Тестирование ПО 44](#_Toc127530575)

[3.2 Отладка ПО 47](#_Toc127530576)

[ПРИЛОЖЕНИЕ 1 54](#_Toc127530577)

[Листинг программного продукта 54](#_Toc127530578)

[ПРИЛОЖЕНИЕ 2 60](#_Toc127530579)

[Проектные модели 61](#_Toc127530580)

**СПИСОК СОКРАЩЕНИЙ**

API - Application-programming interface;

DB - Data Bank;

DES - Data Encryption Standard;

ERD – Entity Relationship Diagram;

FK – Foreign Key;

GNU - GNU’s Not Unix;

HTML - Hyper Text Mark Language;

KDE - K Desktop Environment;

LTSC - Long-Term Servicing Channel.

LXDE - Lightweight X11 Desktop Environment;

MATE - Modular Automatic Test Equipment;

OS – Operating System;

PK – Primary Key;

SQL - Structured Query Language;

WDDM - Windows Display Driver Mode;

WF– WINDOWS FORMS;

XAML - eXtensible Application Markup Language;

XFCE - Ecks Eff See Eee;

БД - база данных;

ГБ – гигабайт;

ГГц – гигагерц;

ГСМ - горюче-смазочные материалы;

ЖЦ - жизненный цикл;

ЖЦПО - жизненный цикл программного обеспечения;

ИТ – информационные технологии;

ОС - операционная система;

ПК - персональный компьютер;

ПО - программное обеспечение;

ПП - программный продукт;

СУБД - система управления базами данных;

ТРК - топливораздаточные колонки;

ФИО – Фамилия Имя Отчество;

ФМД - физическая модель данных;

ЯП - язык программирования.

ВВЕДЕНИЕ

Компании по продажи и обслуживанию аккумуляторов в последние годы очень стремительно развивают свою деятельность. В автомобильной отрасли является одной из важных в экономике, для поддержания этой отрасли одним из элементов являются аккумулятор.

Актуальность разработки программного обеспечения «Компания по продаже и обслуживанию аккумуляторов» заключается в том, что модули учета и редактирования данных позволят значительно облегчить выполняемые сотрудниками задачи, а также в повышении уровня менеджмента.

Объектом исследования является компания, предоставляющая услуги по продаже и обслуживанию аккумуляторов транспортных средств «Компания по продаже и обслуживанию аккумуляторов».

Предметом исследования является автоматизация учета деятельности компании по продаже и обслуживанию аккумуляторов транспортных средств «Компания по продаже и обслуживанию аккумуляторов».

Основной целью курсового проекта является разработка программного обеспечения, позволяющей автоматизировать деятельность организации предоставляющей услуги по продаже и обслуживанию аккумуляторов транспортных средств.

Для достижения постановленной цели необходимо решить следующие задачи:

* Постановка задачи и исследование предметной области;
* Обоснование выбора ОС, СУБД и языка программирования;
* Анализ разработанных программных обеспечений
* Проектирование Use-case диаграммы;
* Проектирование концептуальной модели данных;
* Проектирование информационно-логической модели данных;
* Проектирование физической модели данных;
* Проектирование интерфейса пользователя. ­
* Тестирование и обработка ПО.

# **1.ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ**

## **1.1 Основные аспекты разработки ПО**

Жизненный цикл программного обеспечения – это условная схема, включающая отдельные этапы, которые представляют стадии процесса создания ПО. При этом на каждом этапе выполняются разные действия.

Этапы создания программных продуктов:

* Составление требований заказчика. На данном этапе производится работа с заказчиком и документирование его видения и его требований к программе;
* Проектирование программного продукта. На данном этапе создания программного продукта разрабатывается архитектура компонентов ПО, выбираются нужные шаблоны проектирования и составляется схема информационной базы данных системы;
* Разработка. После формулирования требований, и готовности архитектуры команда начинает разработку ПП. На этапе разработки также выполняется документирование системы;
* Тестирование. После разработки необходимо произвести тестирование системы, тем самым подтвердить её соответствие требованиям;
* Сопровождение ПП. После выпуска производится поддержка программного продукта и выпуск новых версий, которые исправляют ошибки и привносят новый функционал. Также необходимо осуществлять поддержку пользователей разработанного ПО.

Рассмотрим модели и методологии разработки ПО.

Основные модели разработки ПО:

* Waterfall Model — каскадная модель;
* V-model — V-образная модель, разработка через тестирование;
* Incremental Model — инкрементная модель;
* Iterative Model — итеративная модель;
* Spiral Model — спиральная модель.

Рассмотрим Waterfall.

В этой модели разработка осуществляется поэтапно: каждая следующая стадия начинается только после того, как заканчивается предыдущая.

При работе с каскадной моделью основная задача — написать подробные требования к разработке. На этапе тестирования не должно выясниться, что в них есть ошибка, которая влияет на весь продукт.

Плюсы:

* Наличие инструкций и правил по всему процессу. Работа начинается с подробного анализа требований и того, как будет реализован проект. Планы, этапы и процессы утверждаются заранее, фиксируются в документах;
* Определенность в сроках и бюджете. Стоимость продукта и сроки сдачи проекта рассчитаны и утверждены в самом начале и не меняются в процессе;

Минусы:

* Отсутствие гибкости. Невозможно предусмотреть все проблемы в проекте заранее. Из-за жесткой последовательности этапов недочеты станут известны только в конце проекта, придется делать дополнительные итерации и начинать работу заново, а это новые затраты и лишние рабочие часы;
* Заказчик не допускается до разработки и тестирования. Он не может комментировать макеты или прототипы и видит результат только в конце проекта. Если изменились требования или условия, заранее учесть это невозможно;
* Проблемы проявляются только при тестировании.

Первой моделью, получившей широкую известность и действительно структурирующей процесс разработки, является каскадная (водопадная) модель. Каждая стадия каскадной модели заканчивается получением некоторых результатов, которые служат в качестве исходных данных для следующей стадии. На рисунке 1.1 представлена каскадная модель.



**Рисунок 1.1** - Каскадная модель

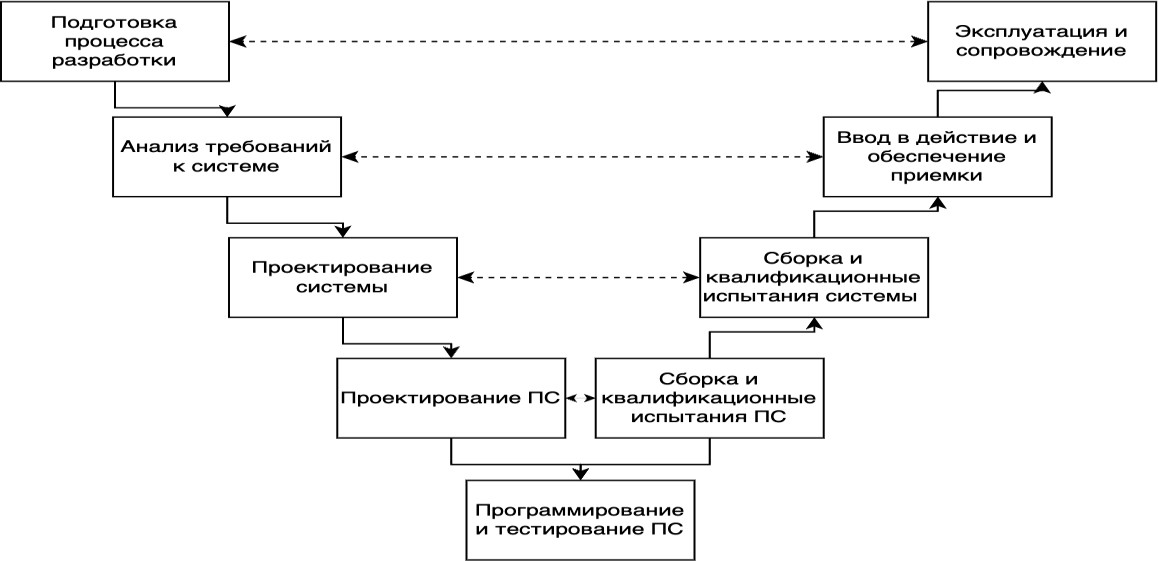
Преимущества применения каскадной модели заключаются в следующем:

* на каждой стадии формируется законченныӗ набор проектной документации, отвечающий критериям полноты и согласованности;
* выполняемые в логичной последователь стадии работ позволяют планировать сроки завершения всех работ и соответствующие затраты. Недостатки каскадной модели:
* в основе модели лежит последовательная линейная структура, в результате чего каждая попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике;
* пользователи не могут убедиться в качестве разработанного продукта до окончания всего процесса разработки.

Основной принцип V-образной модели заключается в том, что детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять.

V-модель – вариация каскадной модели, в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования – вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из стадий разработки влияют на развитие системы тестирования на каждой из стадий тестирования. На рисунке

* 1. представлена V модель.



**Рисунок 1.2 -** V-образная модель

При использовании V-образной модели при разработке проекта, для которого она в достаточной мере подходит, обеспечивается несколько преимуществ:

* + - в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта;
    - в V-образной модели определение требований выполняется перед разработкой проекта системы, а проектирование ПО — перед разработкой компонентов;
    - модель определяет продукты, которые должны быть получены в результате процесса разработки, причем каждые полученные данные должны подвергаться тестированию;
    - благодаря модели менеджеры проекта может отслеживать ход процесса разработки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы является контрольной точкой;
    - модель проста в применении.

Недостатки:

* + - с ее помощью непросто справиться с параллельными событиями;
    - в модели не предусмотрено внесение требования динамических изменений на разных этапах жизненного цикла;
    - тестирование требований в жизненном цикле происходит слишком поздно, вследствие чего невозможно внести изменения, не повлияв при этом на график выполнения проекта;
    - в модель не входят действия, направленные на анализ рисков.

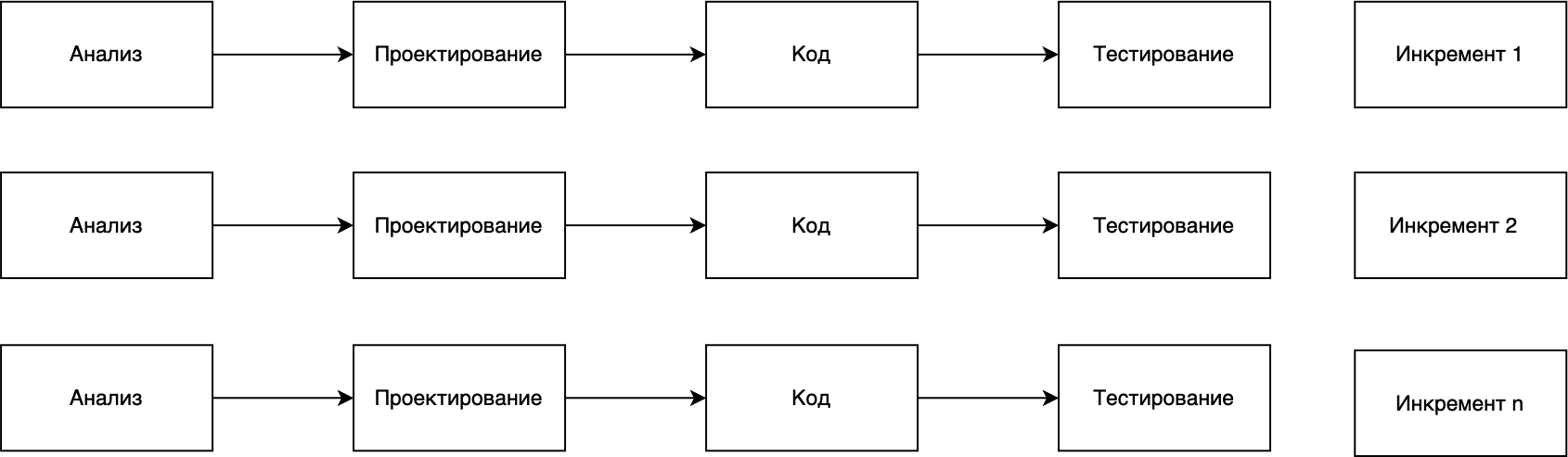
При Особенностью данной модели является разбиение стадий на три логических этапа: проектирование (детализация требований), реализация, тестирование.

Прототипная модель — это модель, в которой прототип разрабатывается ранее самого приложения.

Модели-прототипы обладают ограниченными функциональными возможностями и неэффективной производительностью по сравнению с реальным программным обеспечением. Фиктивные функции используются для создания прототипов. Это ценный механизм для понимания потребностей клиентов.

Прототип ПО разрабатывается ранее самого ПО для получения значимой обратной связи от пользователя. Обратная связь учитывается разработчиками, дорабатывается прототип и снова обсуждается, рассматривается клиентом на предмет изменений и доработок. Данный процесс идет до тех пор, пока модель не будет принята пользователем.

На рисунке 1.3 представлена прототипная модель.



**Рисунок 1.3** - Прототипная модель

Как только завершили процесс сбора требований, в скором времени создается первая версия прототипа, которая предоставляется клиенту для оценки.

Обратная связь клиента и новые, уточненные требования, замечания используются для усовершенствования прототипа и снова представляются заказчику для оценки. Как только клиент подтверждает прототип, он используется как набор требований для создания приложения. Далее ПО разрабатывается по подходам Водопадной модели.

Преимуществом прототипной модели является сокращением затрат и времени на разработку так как баги выявляются на ранних этапах, отсутствие свойств и функционала или изменение в требованиях могут быть выявлены на стадии оценки клиентом и могут быть дополнены на стадии внесения дополнений и вовлечение клиента с начальных этапов сокращает риск ошибок, двусмысленности в требованиях или понимания функционала. Но с тех пор, как клиент задействован на каждом этапе, им могут быть инициированы изменения в требованиях конечного продукта, который повышают сложность работ и могут увеличить сроки разработки продукта.

Недостатки:

* + - Путаница пользователя в отношении прототипа и результирующей системы;
    - Недостаточный анализ;
    - Непонимание разработчиками потребностей пользователей;
    - Разработчики цепляются за прототип;
    - Избыточное время разработки прототип;
    - Затраты на реализацию прототипирования.

При использовании спиральной модели прикладное ПО создается в несколько итераций (витков спирали) методом прототипирования. Создание прототипов осуществляется в несколько итераций, или витков спирали. Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации. На каждой итерации производится тщательная оценка риска превышения сроков и стоимости проекта, чтобы определить необходимость выполнения еще одной итерации, степень полноты и точности понимания требований к системе, а также целесообразность прекращения проекта

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла. Боэм формулирует 10 наиболее распространенных (по приоритетам) рисков:

* + - дефицит специалистов;
    - нереалистичные сроки и бюджет;
    - реализация несоответствующей функциональности;
    - разработка неправильного пользовательского интерфейса;
    - «золотая сервировка», перфекционизм, ненужная оптимизация и
    - оттачивание деталей;
    - непрекращающийся поток изменений;

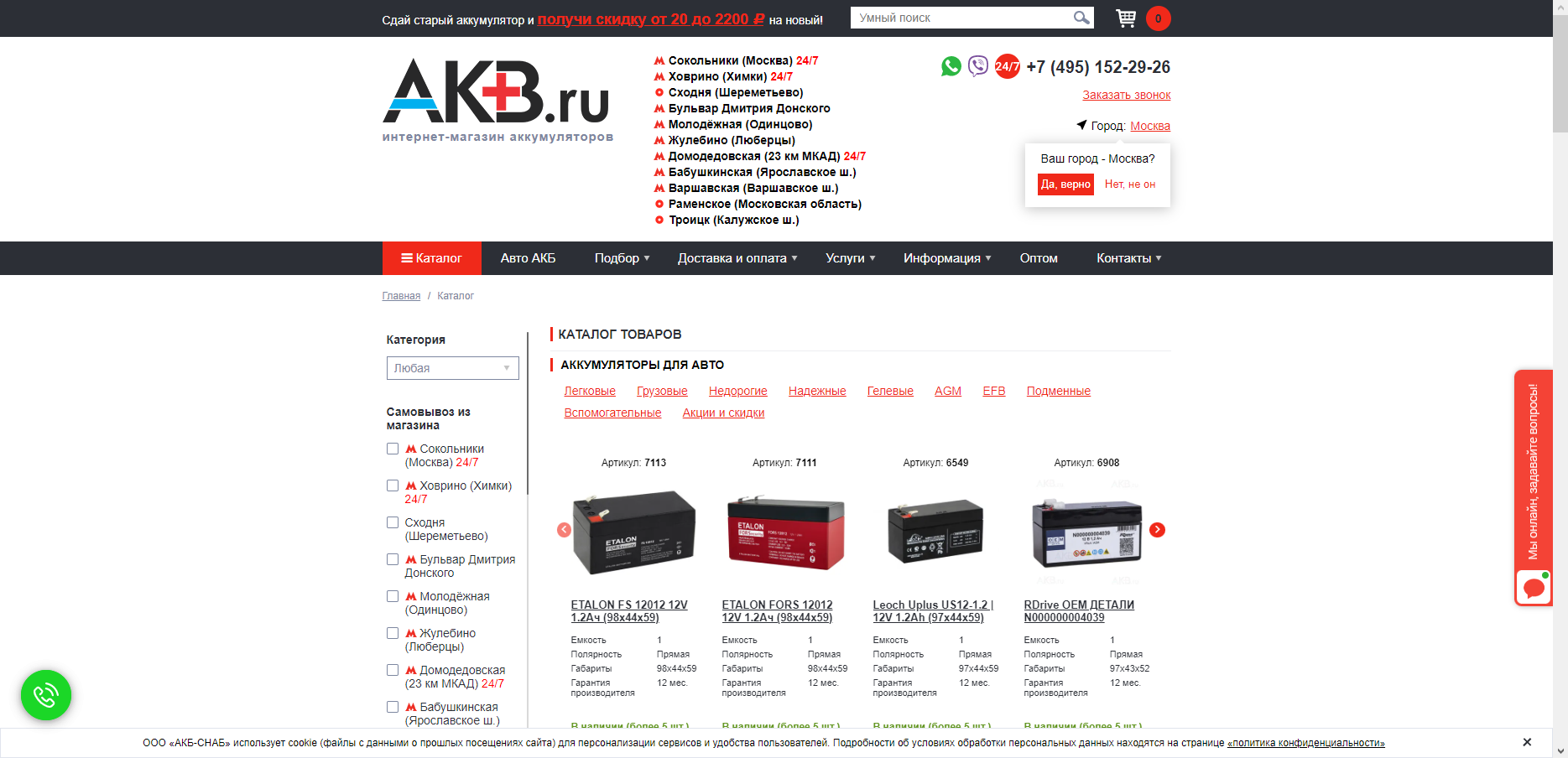
недостаточная производительность получаемой системы.

## **1.2 Анализ разработанных ПО**

На рынке программных обеспечений для осуществления продаже и обслуживанию аккумуляторов транспортных средств существуют решения, которые способны удовлетворить требованиям заказчика, например, сайт: <https://akb.ru/catalog/> по продаже и обслуживанию аккумуляторов транспортных средств. Основными критериями оценки являются:

* информативность (объем представленных сведений и детальность данных);
* навигация (удобная в использовании, все расписано по порядку);
* технические возможности;
* регулярность обновления информации.

На Рисунке 1.5 изображен сайт:



**Рисунок 1.5** – Сайт по продаже и обслуживанию аккумуляторов

Плюсы данного сайта:

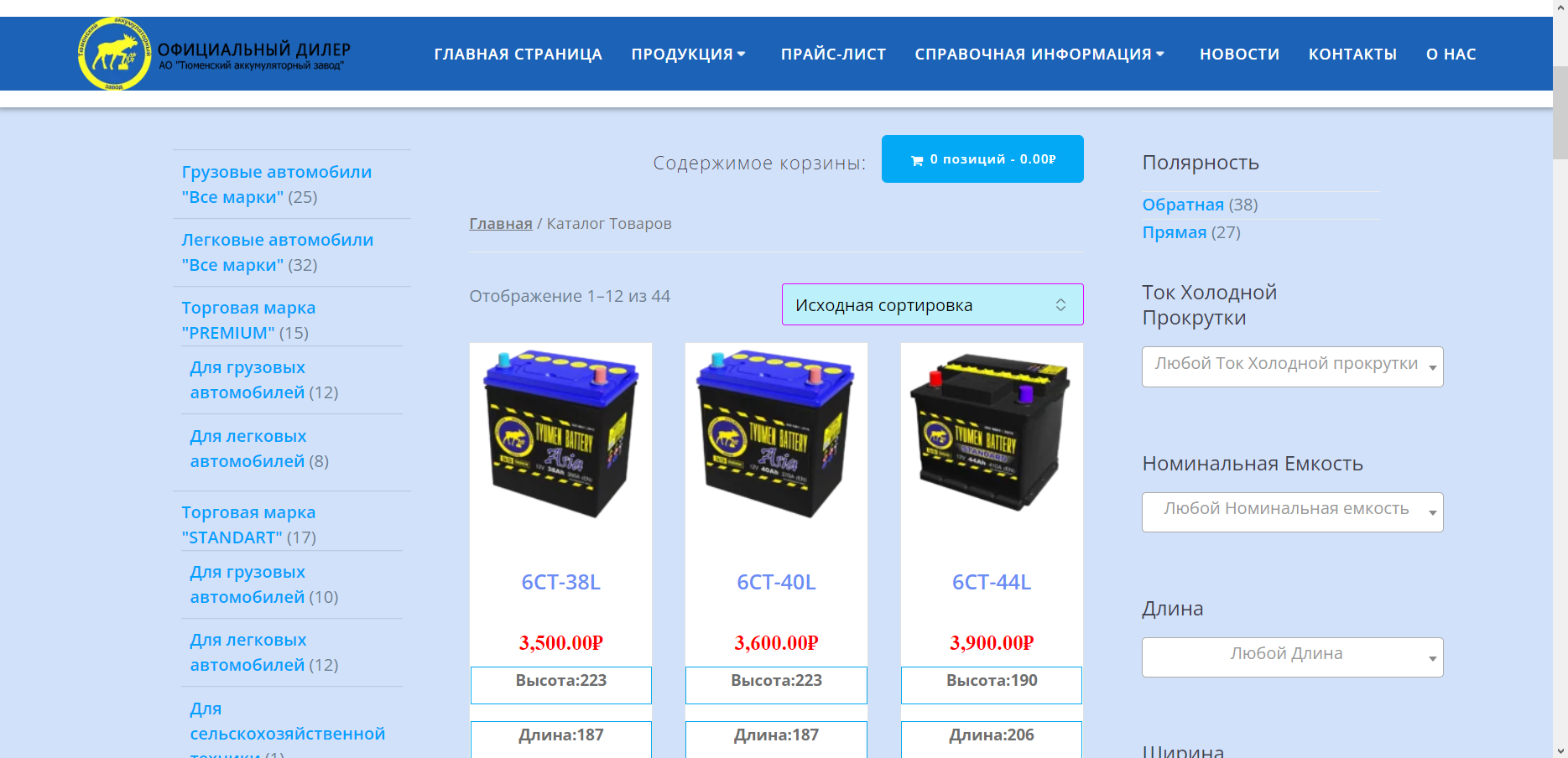
* Широкий выбор аккумуляторов, так и дополнений;
* Прост в понимании;
* Можно сравнить товары.

Минусы:

* Отсутствует личный кабинет все операции нужно проводить по звонку оператору;
* Чтобы узнать цену нужно заполнить форму;
* Отсутствует просмотр статус заказа.

Разберём еще один пример программного обеспечения по продаже и обслуживанию аккумуляторов транспортных средств такое как сайт: https://akb-best.ru

На рисунке 1.6 изображен сайт:



**Рисунок 1.6** – Сайт по продаже и обслуживанию аккумуляторов

Плюсы данного сайта:

* Обширный каталог товаров;
* Прост в понимании;
* Большое количество оказываемых услуг.

Минусы:

* Обязательная регистрация;
* Указана не конечная цена товара;
* Отсутствует просмотр статус заказа.

При анализе программных обеспечений в области автомобильной отрасли, были выявлены особенные черты, которые могут ускорить обслуживание клиентов в области продаже и обслуживанию аккумуляторов.

По итогам анализа было принято решение разработать программное обеспечение, для ПК которая смогла бы заметно упростить оформление заказа, а также для удобной работы с данными.

Таким образом в данной работе были изучены основные этапы разработки программного обеспечения, а именно основная цель проектирования организация данных и связи между ними. А также исследованы аналоги в сфере автоматизации продаж и обслуживания аккумуляторов.

# **2.РАЗДЕЛ ПРОЕКТИРОВАНИЯ**

## **2.1 Постановка задач**

Основной целью курсового проекта является создание программного обеспечения для продажи и обслуживанию аккумуляторов, которое позволит владельцам автомобилей покупать или обслуживать аккумуляторы.

База данных будет включать в себя информацию о клиентах, сотрудниках, роль профиля, вид аккумулятора, статус обслуживания, статус заказа, статус профиля, тип обслуживания.

Разрабатываемое ПО должно соответствовать следующим требованиям:

* Удобный интерфейс
* Масштабируемость
* Эффективное управление данными

## **2.2 Описание предметной области**

Предметная область для продажи и обслуживанию аккумуляторов содержит информацию о виде аккумулятора, сотрудниках и клиентов с их автомобилями.

Деятельность компании для продажи и обслуживанию аккумуляторов заключается в закупке аккумуляторов и запчастей у производителей и продаже физическим лицам в розницу. За счет разницы цены покупки и продажи компания аккумуляторов ведет свою деятельность. Задача заключается в автоматизации учета поступления продажи и с помощью программного продукта обеспечить принятие эффективных решений по закупу наиболее востребованных моделей и аккумуляторов.

В магазине работает следующий штат сотрудников: Администратор и клиент.

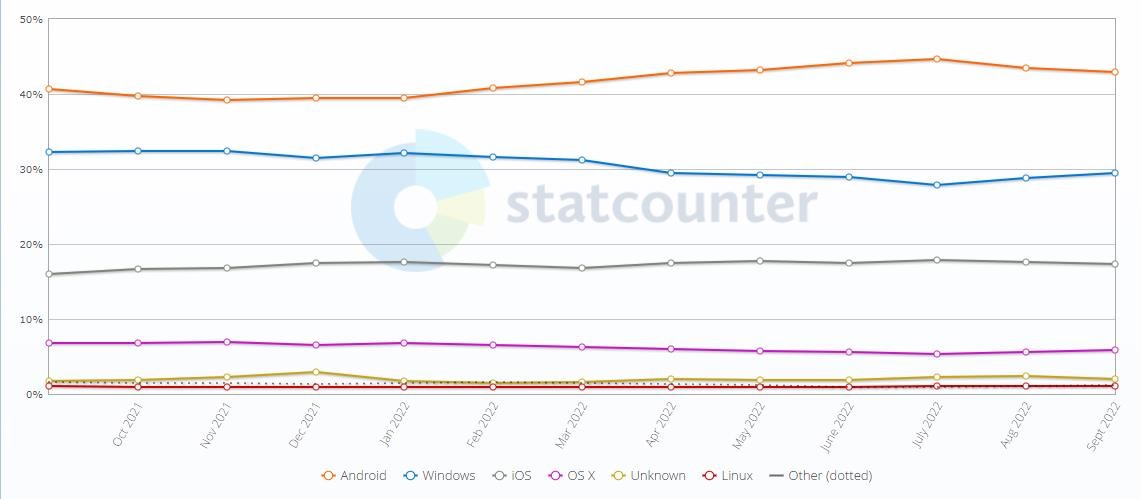
Администратор сможет иметь возможность управлять учетными записями пользователей системы, личными данными, заказами. В его функционал должны входить возможности по добавлению, редактированию или удалению записей.

Работа с клиентом будет происходить через приложение: потенциальный клиент будет делать заказ, при его оформлении формируется запрос в базу данных, где будут указываться данные о заказе.

## **2.3 Анализ обоснования выбора СУБД**

Операционная система — это программное обеспечение, которое управляет аппаратным обеспечением и другим программным обеспечением на компьютере или устройстве. Основная роль операционной системы заключается в предоставлении платформы для запуска приложений и служб, а также в управлении ресурсами устройства, такими как память, вычислительная мощность и хранилище.

Для выбора ОС для разработки ПО нужно воспользоваться статистикой, представленной на рисунке 2.1 по данным источника statcounter[[1]](#footnote-1).

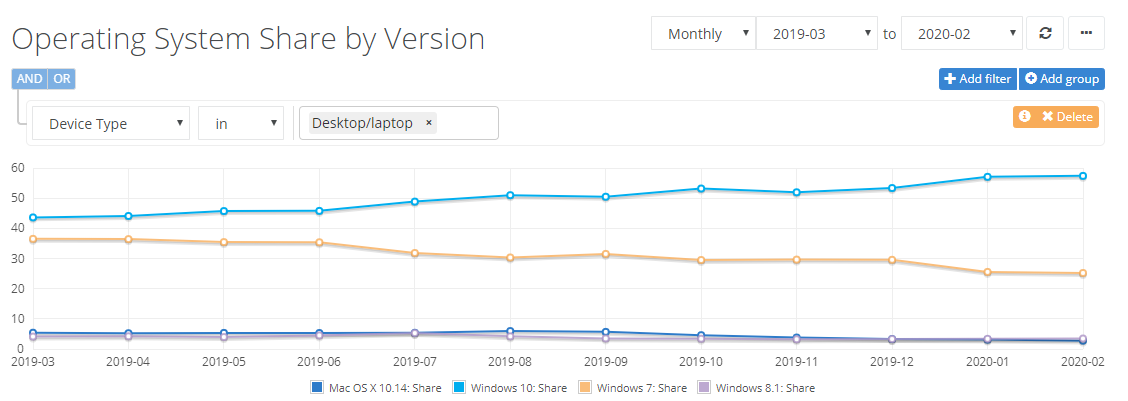


**Рисунок 2.1** – Популярность ОС на февраль 2023 года

Исходя из статистики Windows является самой популярной ОС в 2023 году и подходящею к разработке ПО по следующим причинам:

* Широкое внедрение: благодаря большой базе пользователей Windows обладает высоким уровнем проникновения на рынок, что делает ее подходящей платформой для разработки программного обеспечения, ориентированного на широкую аудиторию.
* Совместимость с широким спектром аппаратного обеспечения: Windows предназначена для работы на различных аппаратных платформах, от традиционных настольных компьютеров и ноутбуков до серверов, планшетов и даже встраиваемых систем, что делает ее гибкой и универсальной платформой для разработки программного обеспечения.
* Известная среда: Работникам не придется разбираться с ОС, что упрощает адаптацию к ПО.

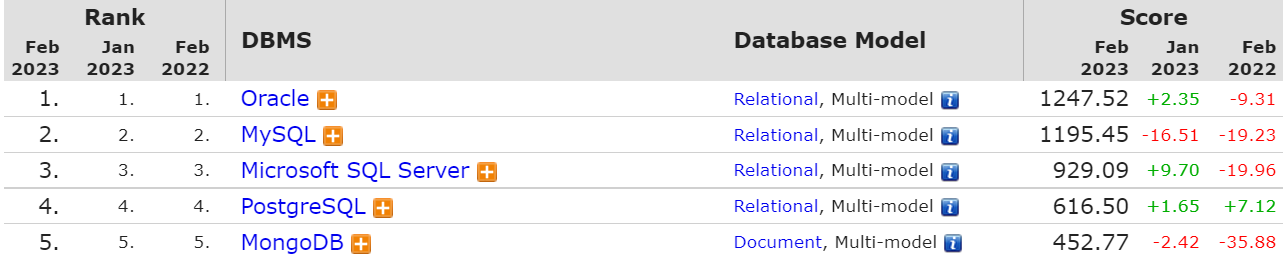
Далее исходя из выбранной ОС нужно выбрать версию Windows исходя из представленной статистике на рисунке 2.2 по данным ресурса statcounter[[2]](#footnote-2).



**Рисунок 2.2** – Популярность Windows на февраль 2023 года

Windows 10 является популярной и производительной операционной системой, которая широко используется для разработки программного обеспечения и соответственно исходя из своей популярности является знакомой многим, а значит вероятней всего будет знакомой для сотрудников что и облегчит адаптацию к ПО.

Далее для разработки ПО нужно выбрать СУБД. На рисунке 2.3 указана статистика популярности СУБД за февраль 2023 года из источника pypl.github.io[[3]](#footnote-3).



**Рисунок 2.3** – Популярность СУБД на февраль 2023 года.

Исходя из графики отсутствует явный отрыв какой-либо СУБД поэтому стоит рассмотреть каждую из первых 3 популярных.

Первой СУБД по популярности является Oracle. Oracle объявила о приостановке своей деятельности на территории России, а значит отсутствие обновлений что делает невозможным поддержку ПО.

Второй СУБД по популярности является MySQL. MySQL — это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность.

Эта СУБД позволяет выбирать различные средства для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства:

* Распространяется бесплатно;
* Скорость и надежность;
* Прекрасно документирована;
* Предлагает много функций, даже в бесплатной версии;
* Поддерживает набор пользовательских интерфейсов;
* MySQL является СУБД с открытым кодом, а значит присутствует возможность интеграции с различными языками программирования и платформами.

Недостатки:

* В MySQL сложно автоматизировать простейшие процессы;
* Для бесплатной версии доступна только платная поддержка.

Идеально подходит для организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

Ещё одной из популярных СУБД является программный продукт Microsoft SQL-сервер. Это система управления базами данных, ядро которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов одновременно.

Достоинства:

* Продукт очень прост в использовании;
* Текущая версия работает быстро и стабильно;
* Исходный код предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов;
* Есть возможность получить доступ к визуализации на мобильных устройствах;
* Хорошо взаимодействует с другими продуктами Microsoft.

Недостатки:

* Цена для юридических лиц оказывается неприемлемой для большей части организаций;
* Даже при тщательной настройке производительности корпорация SQL Server способен занять все доступные ресурсы;
* Сообщается о проблемах с использованием службы интеграции для импорта файлов.
* Низкая возможность интеграции с другими платформами и языками программирования.

Исходя из представленных сравнений для разработки ПО подходит MySQL по следующий причинам:

* Хорошая интеграция с другими языками программирования и платформами;
* MySQL является бесплатным:
* Отличная скорость и безопасность.

Далее для разработки нужно выбрать язык программирония.

Наиболее подходящими языками программирования для разработки настольного приложения являются c#, java и python.

C# - статически типизированный объектно-ориентированный язык.

Плюсы:

* Сильная система типов: C# имеет сильную систему типов, которая помогает обнаруживать ошибки на ранних стадиях процесса разработки и гарантирует, что код является надежным.
* Интеграция с .NET Framework: C# интегрирован с .NET framework, который предоставляет богатый набор библиотек;
* Высокая производительность: C# разработан таким образом, чтобы быть быстрым и эффективным.

Минусы:

* Ограниченная кроссплатформенная поддержка: C# широко используется для разработки под Windows, он имеет ограниченную поддержку для других платформ;
* Большая занимаемая площадь: приложения на C# могут быть относительно большими, а .NET framework требует значительного объема памяти и дискового пространства;
* Зависимость от Microsoft: C# и .NET framework сильно привязан к платформе Microsoft.

Python - типизированный интерпретируемый язык.

Плюсы:

* Простота. Его часто советуют в качестве первого «базового» языка, так как он очень прост в изучении и исполнении.
* Обширность применения.
* Простой синтаксис Python обеспечивает разработчикам отличную читаемость кода.
* Python это очень гибкий язык программирования.

Минусы:

* Низкая скорость. Интерпретируемость языка, которая позволяет запустить код на любой платформе, делает программы на нем не слишком быстрыми.
* Проблемы с потоками. Global Interpreter Lock (GIL) допускает выполнение только одного потока в каждый отдельный момент. Это создает заметные ограничения для использования языка Python. Ошибки исполнения. Скрипты компилируются каждый раз во время выполнения кода.

Java — это статически типизированный объектно-ориентированный язык.

Независимость от платформы: Java предназначена для запуска на любой платформе и устройстве, на которых установлена виртуальная машина Java (JVM), что упрощает разработку и развертывание приложений на различных платформах.

Плюсы:

* Надежность и безопасность: Java обладает рядом функций, которые делают ее надежной и защищенной, таких как проверка типов, автоматическое управление памятью и встроенные функции безопасности.
* Большое сообщество: Java имеет большое и активное сообщество разработчиков, которое предоставляет постоянную поддержку, учебные пособия и ресурсы для разработчиков Java.

Минусы:

* Проблемы безопасности: Java имеет историю уязвимостей в системе безопасности.

Исходя из сравнений C# лучше всего подходит для разработки ПО по следующим причинам:

* Производительность: C# — это статически типизированный язык, который упрощает оптимизацию кода для повышения производительности. Он также более производителен, чем Python и Java, когда речь заходит о задачах, специфичных для Windows;
* Встроенная поддержка Windows: C# был разработан Microsoft и является основным компонентом .NET framework, поэтому он имеет встроенную поддержку Windows и интеграцию с технологиями Windows.

С учетом выбора ОС Microsoft Windows и преимуществами языка C# перед другими языками для написания программного кода, в дальнейшей будет выбран именно он.

## **2.4 Технические требования к разрабатываемому ПО**

В таблице 2.1 представлены рекомендуемые аппаратные требования для ПК, на котором будет разрабатываться ПО.

**Таблица 2.1**

**Минимальные аппаратные требования для ПК разработчика**

|  |  |
| --- | --- |
| **Операционная система** | Windows 10 |
| **Оперативная память** | 4 ГБ |
| **Процессор** | 1.66 ГГц Intel i5 |
| **Место на диске** | 64 ГБ |

В результате уточнения технических требований для разработки ПО был начат этап проектирования программного продукта.

## **2.4.1 Диаграмма прецедентов**

Диаграмма вариантов использования[[4]](#footnote-4) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Каждая группа пользователей на диаграмме вариантов использования обозначается человечком, под которым записывается имя группы людей, которую он обозначает. В терминологии UML, этот человечек называется актёром. В общем случае, актёр обозначает любые сущности, использующие систему. Этими сущностями могут быть люди, технические устройства или даже другие системы.

Каждая группа пользователей использует определённые функции системы. На диаграмме вариантов использования функция системы изображается эллипсом, внутри которого записывается имя функции в форме глагола с пояснительными словами. В терминологии UML, этот эллипс называется вариантом использования. В общем случае, вариант использования – набор действий, который может быть использован актёром для взаимодействия с системой.

На диаграммах UML для связывания элементов используются различные соединительные линии, которые называются отношениями.

Каждое такое отношение имеет собственное название и используется для достижения определённой цели.

Диаграмма прецедентов пользователей сайта представлена на рисунке 1 в ПРИЛОЖЕНИЕ 2.

## **2.4.2 Проектирование концептуальной модели**

Концептуальная модель[[5]](#footnote-5) — это модель, представленная множеством понятий и связей между ними, определяющих смысловую структуру рассматриваемой предметной области или её конкретного объекта.

Концептуальная модель служит основой для проектирования базы данных и разработки системы. При этом каждый новый объект этой базы данных должен быть классифицирован в соответствии с понятиями этой модели. Это же справедливо и в отношении каждой операции. Сама концептуальная модель представлена на рисунке 2 в ПРИЛОЖЕНИИ 2.

## **2.4.3 Проектирование инфологической модели данных**

Инфологическая модель[[6]](#footnote-6) (информационно-логическая модель) —ориентированная на человека и не зависимая от типа СУБД модель предметной области, определяющая совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей.

Цель инфологической модели - обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты). Инфологическая модель представлена на рисунке 3 в ПРИЛОЖЕНИИ 2.

Словарь данных представлен ниже (смотреть Табл. 2.2 – 2.11).

**Таблица 2.2 -** order

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** | |
| PK | idOrder | int | 11 | да | Первичный ключ таблицы | |
| FK | idUser | int | 11 | да | Вторичный ключ таблицы «user» | |
| FK | idOrder | int | 11 | да | Вторичный ключ таблицы «Order» | |
| FK | status | nvarchar | 50 | да | Вторичный ключ таблицы «statuses» | |
|  | totalPrice | float | 11 | да |  |
|  | orderDate | date |  | да |  |
|  | orderTime | time |  | ла |  |

**Таблица 2.3 -** statuses

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | status | nvarchar | 50 | да | Первичный ключ таблицы |

**Таблица 2.4 -** Order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | idOrder | int | 11 | да | Первичный ключ таблицы |
| FK | idUser | int | 11 | да | Вторичный ключ таблицы «user» |
| FK | statusOrder | nvarchar | 50 | да | Вторичный ключ таблицы «statusOrder» |

**Таблица 2.5 -** statusOrder

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | status | nvarchar | 50 | да | Первичный ключ таблицы |

**Таблица 2.6 -** bucket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | idBucket | int | 11 | да | Первичный ключ таблицы |
| FK | idOrder | int | 11 | да | Вторичный ключ таблицы «Order» |
| FK | id | int | 11 | да | Вторичный ключ таблицы «menu» |

**Таблица 2.7 -** menu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | idName | int | 11 | да | Первичный ключ таблицы |
|  | name | nvarchar | 50 | да |  |

**Продолжение Таблица 2.7 -** menu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | price | float | 11 | да |  |
|  | photo | image |  | нет |  |
|  | discript | nvarchar | max | да |  |

**Таблица 2.8 -** user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | idUser | int | 11 | да | Первичный ключ таблицы |
|  | login | nvarchar | 50 | да |  |
|  | password | nvarchar | 50 | да |  |
| FK | gender | nvarchar | 50 | да | Вторичный ключ таблицы «userGenders» |
|  | surname | nvarchar | 50 | да |  |
|  | name | nvarchar | 50 | да |  |
|  | patronymic | nvarchar | 50 | нет |  |
|  | userPhoto | image |  | нет |  |
| FK | role | nvarchar | 50 | да | Вторичный ключ таблицы «userRoles» |
|  | birthday | nvarchar | 50 | да |  |
|  | phoneNumber | nvarchar | 50 | да |  |
| FK | status | nvarchar | 50 | да | Вторичный ключ таблицы «userStatus» |

**Таблица 2.9 –** userStatus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | status | nvarchar | 50 | да | Первичный ключ таблицы |
|  |  |  |  |  |  |

**Таблица 2.10 -** userRoles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | role | nvarchar | 50 | да | Первичный ключ таблицы |

**Таблица 2.11 -** userGenders

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключевое-поле** | **Название** | **Тип** | **Размер** | **Обяз/НеОбяз** | **Комментарий** |
| PK | userGender | nvarchar | 50 | да | Первичный ключ таблицы |

## **2.4.4 Проектирование физической модели данных**

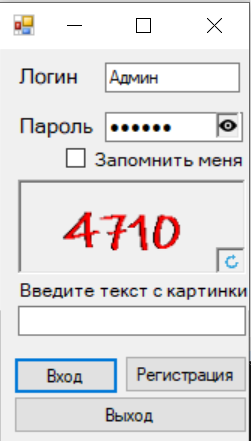
Физическая модель — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных.

В ПРИЛОЖЕНИИ 2 на Рисунке 4 изображена ERD диаграмм

## **2.5 Разработка интерфейса программного продукта**

При проектировании интерфейса были использованы поля ввода текста, выпадающие списки, а также функциональные кнопки. Некоторые из них генерируются с помощью кода. Простота дизайна была первоначальной целью, для того чтобы пользователю программы было удобно пользоваться ее функционалом.

На Рисунке 2.5 изображена страница авторизации. На форме присутствуют, 2 поля для ввода текста и кнопка. После успешного ввода данных сотрудник может войти в систему, в случае ошибки будет написано об этом.



**Рисунок 2.5** - Страница авторизации

Ниже представлен код аворизации, проверка данных на корректность, капча.

private void buttonEnter\_Click(object sender, EventArgs e)

{

foreach (var pb in this.Controls.OfType<TextBox>())

{

if (pb.Text == "" && pb.Name != "textBoxCaptcha")

{

MessageBox.Show("Одно или несколько полей пустые", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

using (ShopEntities db = new ShopEntities())

{

if (db.user.Any(p => p.login == textBoxLogin.Text))

{

var userData = db.user.Single(p => p.login == textBoxLogin.Text);

if (pictureBoxCaptcha.Visible == true)

{

if (textBoxCaptcha.Text != number.ToString())

{

MessageBox.Show("Попробуйте еще раз ввести Captcha", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

loadCaptchaImage();

}

else if (textBoxPassword.Text != userData.password)

{

MessageBox.Show("Неправильный пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

loadCaptchaImage();

}

else

{

if (userData.status == "Неактивен")

{

MessageBox.Show("Аккаунт неактивен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (userData.role == "Админ")

{

wrongPasswordCount = 0;

FormAdmin AdminForm = new FormAdmin(this, userData.login);

AdminForm.Show(this);

this.Hide();

if (checkBoxMemorize.Checked)

{

using (StreamWriter strem = new StreamWriter(filePath))

strem.WriteLine(textBoxLogin.Text + "\n" + RcaCryptDecrypt.Crypt(textBoxPassword.Text));

}

}

else

{

wrongPasswordCount = 0;

FormUser UserForm = new FormUser(this, userData.login);

UserForm.Show(this);

this.Hide();

if (checkBoxMemorize.Checked)

{

using (StreamWriter strem = new StreamWriter(filePath))

strem.WriteLine(textBoxLogin.Text + "\n" + RcaCryptDecrypt.Crypt(textBoxPassword.Text));

}

}

}

}

else if (textBoxPassword.Text != userData.password)

{

MessageBox.Show("Неправильный пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

wrongPasswordCount++;

if (wrongPasswordCount >= 3)

{

pictureBoxCaptcha.Visible = true;

labelCaptcha.Visible = true;

textBoxCaptcha.Visible = true;

pictureBoxRefreshCaptcha.Visible = true;

pictureBoxRefreshCaptcha.BringToFront();

this.Height = 330;

loadCaptchaImage();

}

}

else

{

if (userData.status == "Неактивен")

{

MessageBox.Show("Аккаунт неактивен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (userData.role == "Админ")

{

wrongPasswordCount = 0;

FormAdmin AdminForm = new FormAdmin(this, userData.login);

AdminForm.Show(this);

this.Hide();

if (checkBoxMemorize.Checked)

{

using (StreamWriter strem = new StreamWriter(filePath))

strem.WriteLine(textBoxLogin.Text + "\n" + RcaCryptDecrypt.Crypt(textBoxPassword.Text));

}

}

else

{

wrongPasswordCount = 0;

FormUser UserForm = new FormUser(this, userData.login);

UserForm.Show(this);

this.Hide();

if (checkBoxMemorize.Checked)

{

using (StreamWriter strem = new StreamWriter(filePath))

strem.WriteLine(textBoxLogin.Text + "\n" + RcaCryptDecrypt.Crypt(textBoxPassword.Text));

}

}

}

}

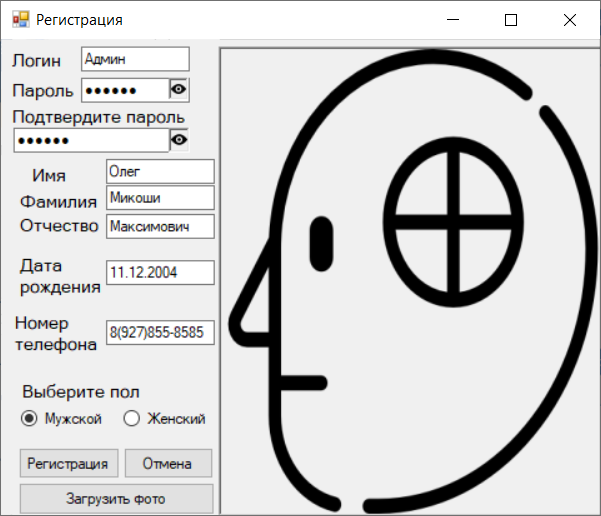
else

{

MessageBox.Show("Такого пользователя нету", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}



**Рисунок 2.6** - Страница регистрации

Ниже представлен код авторизации, проверка данных на корректность, добавления картинки.

private void buttonLoadPhoto\_Click(object sender, EventArgs e)

{

pictureBoxPhoto.Visible = true;

this.Width = 495;

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

pictureBoxPhoto.Image = Image.FromFile(openFileDialog.FileName);

}

catch

{

MessageBox.Show("Невозможно открыть выбранный файл", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void buttonRegister\_Click(object sender, EventArgs e)

{

foreach (var pb in this.Controls.OfType<TextBox>())

{

if (pb.Text == "" && pb.Name != "textBoxPatronymic")

{

MessageBox.Show("Одно или несколько полей пустые", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

foreach (var pb in this.Controls.OfType<MaskedTextBox>())

{

if (!pb.MaskCompleted)

{

MessageBox.Show("Одно или несколько полей пустые", "Ошибка");

return;

}

}

if (textBoxPassword.Text == textBoxAcceptPassword.Text)

{

if (checkUser())

{

MessageBox.Show("Такой пользователь уже существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (radioButtonFemale.Checked || radioButtonMale.Checked)

{

string rbText = "";

foreach (var pb in this.Controls.OfType<RadioButton>())

{

if (pb.Checked)

{

rbText = pb.Text;

}

}

if (pictureBoxPhoto.Image == null)

{

using (ShopEntities db = new ShopEntities())

{

db.user.Add(new user

{

login = textBoxLogin.Text.Trim(),

password = textBoxPassword.Text.Trim(),

gender = rbText,

surname = textBoxSurname.Text.Trim(),

name = textBoxName.Text.Trim(),

patronymic = textBoxPatronymic.Text.Trim(),

role = "Пользователь",

birthday = maskedTextBoxBirthday.Text,

phoneNumber = maskedTextBoxPhoneNumber.Text,

status = "Активен"

});

db.SaveChanges();

}

MessageBox.Show("Успешная регистрация", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

enterForm.Show();

}

else

{

using (ShopEntities db = new ShopEntities())

{

db.user.Add(new user

{

login = textBoxLogin.Text.Trim(),

password = textBoxPassword.Text.Trim(),

gender = rbText,

surname = textBoxSurname.Text.Trim(),

name = textBoxName.Text.Trim(),

patronymic = textBoxPatronymic.Text.Trim(),

userPhoto = convertImageToBytes(pictureBoxPhoto.Image),

role = "Пользователь",

birthday = maskedTextBoxBirthday.Text,

phoneNumber = maskedTextBoxPhoneNumber.Text,

status = "Активен"

});

db.SaveChanges();

}

MessageBox.Show("Успешная регистрация", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

enterForm.Show();

}

}

else

{

MessageBox.Show("Вы не выбрали пол", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

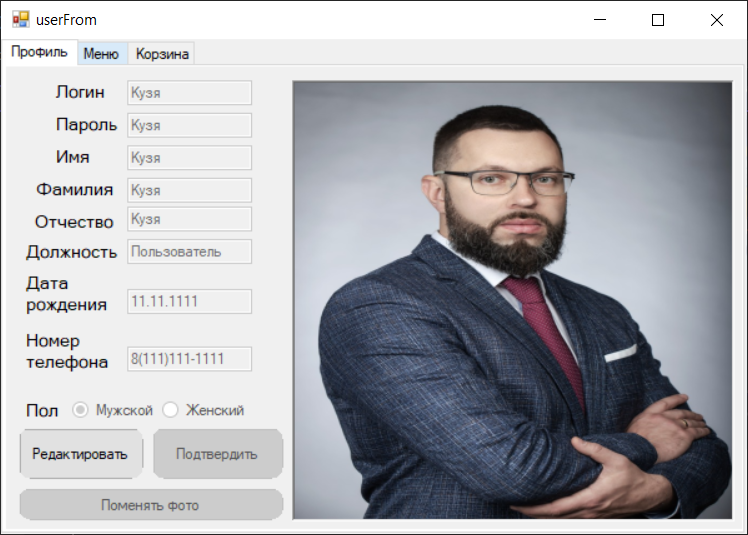
}

else

{

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}}



**Рисунок 2.7** – Страница редактирования профиля

Ниже представлен код редактирования профеля.

private void FormUser\_FormClosing(object sender, FormClosingEventArgs e)

{

enterForm.Show();

using (ShopEntities db = new ShopEntities())

{

if (db.order.Where(p => p.idUser == userData.idUser && p.status == "Активен").Count() > 0){

if (db.ShopOrder.Where(p => p.idUser == userData.idUser && p.statusOrder == "Активен").Count() > 0)

{

var order = db.ShopOrder.Single(p => p.idUser == userData.idUser && p.statusOrder == "Активен");

order.statusOrder = "Выполнен";

db.Entry(order).State = System.Data.Entity.EntityState.Modified;

}

var mainOrder = db.order.Single(p => p.idUser == userData.idUser && p.status == "Активен");

mainOrder.status = "Выполнен";

}

else

{

if (db.ShopOrder.Where(p => p.idUser == userData.idUser && p.statusOrder == "Активен").Count() > 0)

{

var order = db.ShopOrder.Single(p => p.idUser == userData.idUser && p.statusOrder == "Активен");

order.statusOrder = "Отменен";

db.Entry(order).State = System.Data.Entity.EntityState.Modified;

}

}

db.SaveChanges();

}

}

private void updateData()

{

string rbText = "";

foreach (var pb in tabPageProfile.Controls.OfType<RadioButton>())

{

if (pb.Checked)

{

rbText = pb.Text;

}

}

userData.login = textBoxLogin.Text;

userData.password = textBoxPassword.Text;

userData.name = textBoxName.Text;

userData.surname = textBoxSurname.Text;

userData.patronymic = textBoxPatronymic.Text;

userData.birthday = maskedTextBoxBirthday.Text;

userData.phoneNumber = maskedTextBoxPhoneNumber.Text;

userData.gender = rbText;

if (pictureBoxProfileImage.Image != null)

{

userData.userPhoto = convertImageToBytes(pictureBoxProfileImage.Image);

}

}

private void setDataFunc()

{

textBoxLogin.Text = userData.login;

textBoxPassword.Text = userData.password;

textBoxName.Text = userData.name;

textBoxSurname.Text = userData.surname;

textBoxPatronymic.Text = userData.patronymic;

textBoxRole.Text = userData.role;

maskedTextBoxBirthday.Text = userData.birthday;

maskedTextBoxPhoneNumber.Text = userData.phoneNumber;

if (userData.userPhoto != null)

{

pictureBoxProfileImage.Image = convertByteArrayToImage(userData.userPhoto);

}

if (userData.gender == "Мужской")

{

radioButtonMale.Checked = true;

}

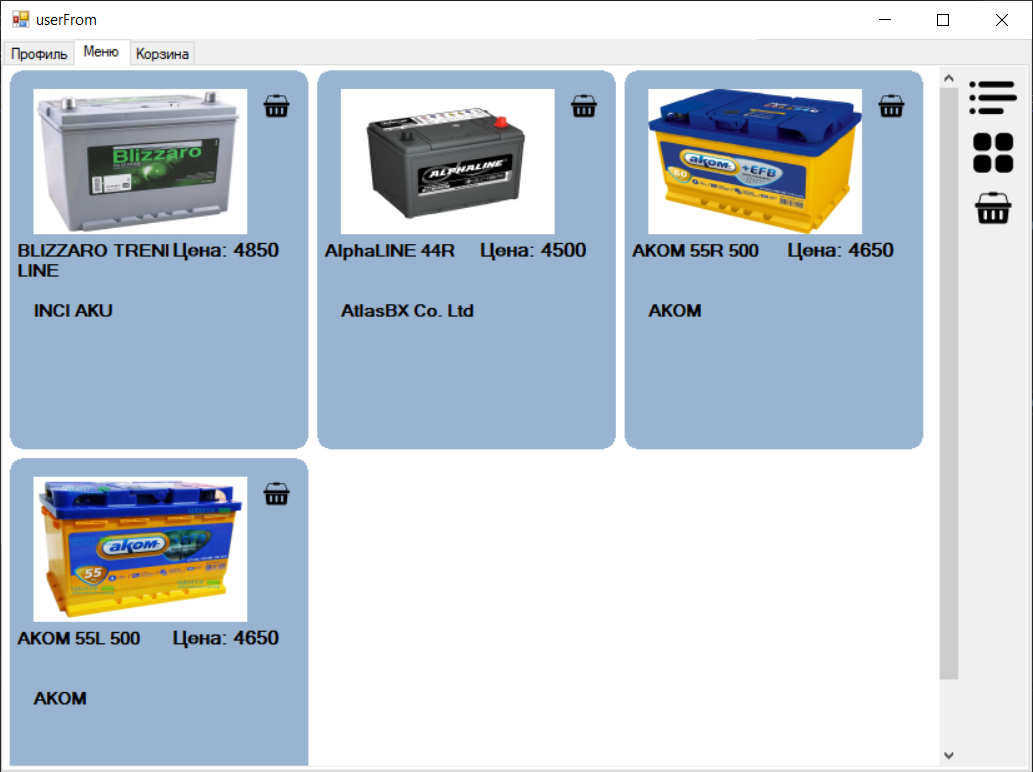
else

{

radioButtonFemale.Checked = true;

}

}



**Рисунок 2.8** – Страница товаров

Ниже изображен код добавление в корзину.

private void buttonAddToBucket\_Click(object sender, EventArgs e)

{

using (ShopEntities db = new ShopEntities())

{

if (db.ShopOrder.Where(p => p.idUser == userData.idUser && p.statusOrder == "Активен").Count() == 0)

{

newFoodOrder();

}

var order = db.ShopOrder.Single(p => p.idUser == userData.idUser && p.statusOrder == "Активен");

db.bucketShop.Add(new bucketShop

{

idShopOrder = order.idShopOrder,

idShop = id,

count = Convert.ToDouble(textBoxCountsRoads.Text)

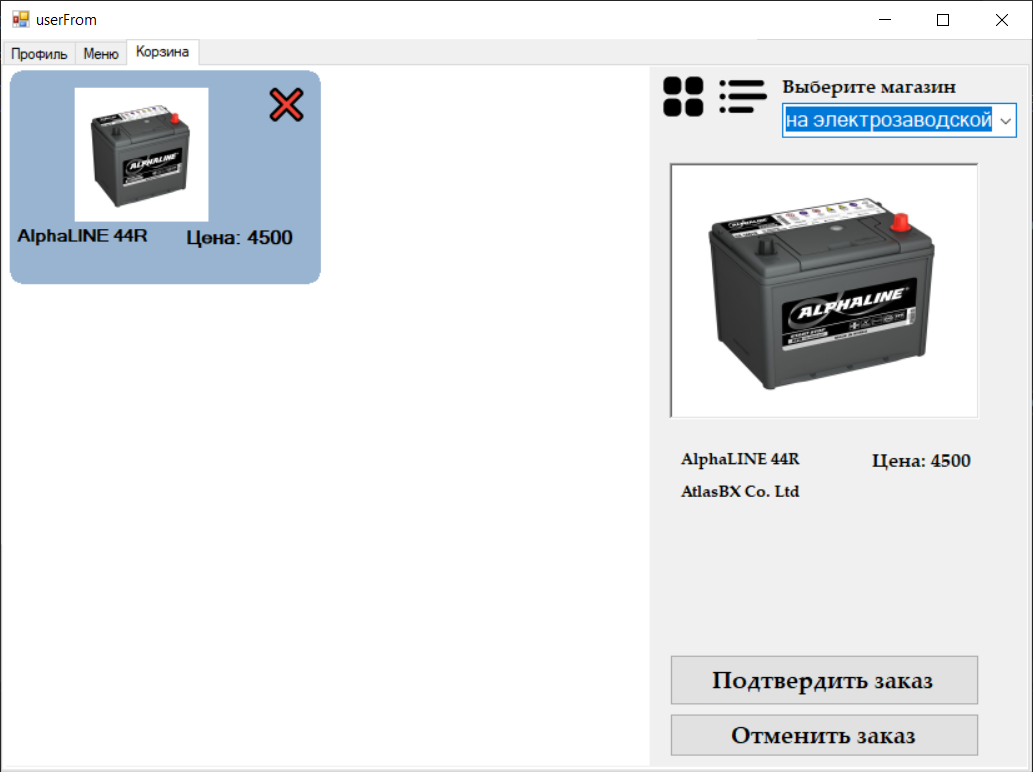
});

db.SaveChanges();

totalPrice += Convert.ToDouble(labelPrice.Text.Split(' ')[1])\* Convert.ToDouble(textBoxCountsRoads.Text);

}

}



**Рисунок 2.9** – Корзина

Ниже представлен код оформления заказа.

private void AddToOrder()

{

using (ShopEntities db = new ShopEntities())

{

if (db.ShopOrder.Where(p => p.idUser == userData.idUser && p.statusOrder == "Активен").Count() > 0)

{

var userOrder = db.ShopOrder.Single(p => p.idUser == userData.idUser && p.statusOrder == "Активен");

db.order.Add(new order

{

idShopOrder = userOrder.idShopOrder,

idUser = userData.idUser,

status = "Активен",

totalPrice = totalPrice,

orderDate = DateTime.Now,

orderTime = DateTime.Now.TimeOfDay

});

db.SaveChanges();

}

else

{

MessageBox.Show("Ваша корзину пуста или cамоката нет в наличие");

return;

}

var mainOrder = db.order.Single(p => p.idUser == userData.idUser && p.status == "Активен");

MessageBox.Show("Ваш заказ #" + mainOrder.idOrder + " оформлен");

this.Close();

}

}

# **3.ЭКСПЕРЕМЕНТАЛЬНЫЙ РАЗДЕЛ**

## **3.1 Тестирование ПО**

Тестирование программного обеспечения – это:

* процесс исследования ПО с целью получения информации о качестве продукта;
* процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом;
* оценка системы с тем, чтобы найти различия между тем, какой система должна быть и какой она есть.

В широком смысле, тестирование – это одна из техник контроля качества (Quality Control), которая включает планирование, составление тестов, непосредственно выполнение тестирования и анализ полученных результатов.

Тестирование Black-Box:

Методика тестирования без каких-либо знаний о внутренней работе приложения называется «черным ящиком». Тестер не обращает внимания на архитектуру системы и не имеет доступа к исходному коду. Как правило, при выполнении теста с «черным ящиком» тестер будет взаимодействовать с пользовательским интерфейсом системы, предоставляя входные данные и анализируя выходы, не зная, как и где обрабатываются входы.

Плюсы Black-Box:

* Хорошо подходит и эффективен для больших сегментов кода;
* Четкое разделение перспективы пользователя с точки зрения разработчика с помощью явно определенных ролей;
* Кодовый доступ не требуется.

Минусы Black-Box:

* Ограниченное покрытие;
* Неэффективное тестирование, из-за того, что тестер только имеет ограниченные знания о приложении.

Тестирование белым ящиком:

Проверка белого ящика — это подробное исследование внутренней логики и структуры кода. Тестирование с использованием белого ящика также называется тестированием стекла или открытым тестированием. Чтобы выполнить тестирование белого ящика в приложении, тестер должен знать внутреннюю работу кода.

Тестер должен заглянуть внутрь исходного кода и выяснить, какое устройство / блок кода ведет себя некорректно.

Плюсы белого ящика:

* Это помогает в оптимизации кода;
* Поскольку тестер знает исходный код, становится очень легко узнать, какой тип данных может помочь в эффективном тестировании приложения;
* Благодаря знаниям тестера о коде, максимальный охват достигается при написании сценария.

Минусы белого ящика:

* В связи с тем, что для тестирования белых ящиков требуется квалифицированный тестер, затраты увеличиваются;
* Иногда невозможно заглянуть в каждый уголок и угол, чтобы обнаружить скрытые ошибки, которые могут создавать проблемы, так как многие пути будут непроверенные.

Тестирование серым ящиком:

Тестирование на серой коробке — это метод тестирования приложения с ограниченным знанием внутренней работы приложения. При тестировании программного обеспечения фраза, чем больше вы знаете, тем лучше переносит массу при тестировании приложения.

Освоение домена системы всегда дает тестеру преимущество над кем-то с ограниченными знаниями домена. В отличие от тестирования черного ящика, где тестер тестирует только пользовательский интерфейс приложения; при тестировании в сером полете тестер имеет доступ к проектной документации и базе данных. Имея эти знания, тестер может подготовить лучшие тестовые данные и сценарии тестирования при составлении плана тестирования.

Плюсы серого ящика:

* Предлагает комбинированные преимущества тестирования черного ящика и белого ящика, где это возможно;
* Тест выполняется с точки зрения пользователя, а не дизайнера;
* Тестировщики серого ящика не полагаются на исходный код; вместо этого они полагаются на определение интерфейса и функциональные спецификации.

Минусы серого ящика:

* Поскольку доступ к исходному коду недоступен, возможность пройти через код и зону тестирования ограничена;
* Тесты могут быть излишними, если разработчик программного обеспечения уже выполнил тестовый пример.

После разбора нескольких способов тестирования. В конечном итоге было выбрано тестирование Black-Box. Так как он очень быстро позволяет найти ошибки в отличии от белого ящика, а также тестирование происходит с позиции пользователя.

В данном тестировании использовались такие методы как:

* Beta-тестирование;
* Регрессионное тестирование;
* Стресс тестирование.

Результатом теста по Beta-тестированию программа была полностью проверена на наличие багов в ПО. В конце тестирования они были полностью устранены.

По результатам Регрессивного тестирования было выполнено полное тестирование нового кода, а также исправление ошибок в новом коде.

Последнее что было выполнено это стресс тестирование. Было выполнено тестирование быстрой загрузки данных в выпадающие списки, а также таблицы.

## **3.2 Отладка ПО**

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Сперва cделаем отладку окна авторизации.

Была найдена ошибка при вводе данных несуществующего пользователя. Когда это происходит программа экстренно завершает работу. Для решения это проблемы был использован Try catch.

Исправленный код:

try

{

var userData = db.Users.Single(p => p.Login == textBoxLogin.Text);

id = userData.id;

if (passwordBox.Password != userData.password)

{

System.Windows.MessageBox.Show("Неправильный пароль", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

return;

}

Window window = null;

switch (userData.id\_job\_title)

{

case 1:

System.Windows.MessageBox.Show("Ваш аккаунт не актевиравн, обратитесь к админестратору", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

return;

case 2:

window = new BDEntitiesWindow();

break;

case 3:

window = new ShopTableWindow();

break;

case 4:

window = new OrdersWindow(id);

break;

case 5:

window = new DirectorWindow();

break;

}

classStandartMethods.ChangeWindow(window, (Window)this);

}

catch

{

System.Windows.MessageBox.Show("Не верные данные", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

}

Была найдена ошибка при входе. Программа может сообщить что пароль введён не верно даже если это не так. Ошибка связанна с тем, что данные пароля берутся из панели со скрытым паролем. Для решения ошибки надо в момент входа проверять активность функции.

Исправленный код:

if (showPaswordIsTrue)

passwordBox.Password = ShowPasswordBox.Text;

Отладка окна регистрации.

Была найдена ошибка при написании логина. Программа не запрещает написать логин, который уже есть. Для исправления этой ошибки, перед созданием пользователя был добавлен код который сверял логин с уже имеющимися, и в случаи совпадения прекращал операцию создания пользователя.

Исправленный код:

try

{

var loginCheck = db.Users.Single(p => p.Login == textBoxLogin.Text);

System.Windows.MessageBox.Show("Логин занят", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

}

catch

{

db.Users.Add(new Users

{

Login = textBoxLogin.Text,

password = passwordBox.Password,

id\_gender = gender,

surname = textBoxSurname.Text,

name = textBoxName.Text,

mail = textBoxMail.Text,

id\_role = 4,

date\_of\_birth = dateTime,

phone\_number = TextBoxPhoneNumber.Text,

id\_job\_title = 1

});

db.SaveChanges();

System.Windows.MessageBox.Show("Регистрация прошла успешна. Теперь обратитесь к админестратору для активации аккаунта", "Завершение опрерации", (MessageBoxButton)MessageBoxButtons.OK);

ButtonBack\_Click(null, null);

}

Была найдена ошибка. Программа не запрещает создавать слишком простой пароль, который не будет безопасным. Для исправления была добавлена функция, которая проверяет пароль на сложность.

Код функции:

private bool PassworHards(string passwor)

{

if (passwor.Length < 7)

{

System.Windows.MessageBox.Show("Пароль слишком короткий", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

return true;

}

if(!passwor.AsEnumerable().Any(ch => char.IsDigit(ch)))

{

System.Windows.MessageBox.Show("Пароль должен содежать цифры", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

return true;

}

if(!passwor.AsEnumerable().Any(ch => char.IsLetter(ch)))

{

System.Windows.MessageBox.Show("Пароль должен содержать буквы", "Ошибка", (MessageBoxButton)MessageBoxButtons.OK, (MessageBoxImage)MessageBoxIcon.Error);

return true;

}

return false;

}

Была найдена ошибка. Программа не проверяет заполнилось всех строк из-за чего можно оставить пустые строки и при попытке зарегистрироваться программа сломается. Для решения этой проблемы нужно проверять поля на заполнилось через отдельную функцию.

Код функции:

private bool ChackTextBox()

{

bool DataAreEmpty = false;

foreach(System.Windows.Controls.TextBox textBox in allTextBoxArray)

{

if (textBox.Text == "")

{

textBox.BorderBrush = Brushes.Red;

DataAreEmpty = true;

}

}

return DataAreEmpty;

}

# **ЗАКЛЮЧЕНИЕ**

В ходе разработки ПО учета продажи элетроаккумуляторов была получена полностью рабочая программа.

В результате выполнения курсового проекта было разработано ПО учета продажи элетроаккумуляторов, также выполнены все поставленные задачи, а именно:

* Постановка задачи и исследование предметной области;
* Обоснование выбора ОС, СУБД и языка программирования;
* Анализ разработанных программных обеспечений
* Проектирование Use-case диаграммы;
* Проектирование концептуальной модели данных;
* Проектирование информационно-логической модели данных;
* Проектирование физической модели данных;
* Проектирование интерфейса пользователя. ­
* Тестирование и обработка ПО.

Реализация ПО была осуществлена при помощи технологии WF с использованием языковых средств программирования С#. В качестве среды разработки была использована среда Microsoft Visual Studio 2022. Разработка базы данных выполнялась в СУБД MSSQL версия 19.0.1.

Таким образом цель курсового проекта была полностью достигнута и в соответствии с ней было полностью продемонстрированно рабочее программное обеспечение.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ Р 43.0.11-2014 - БАЗЫ ДАННЫХ В ТЕХНИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ - 2014
2. ГОСТ Р 51904-2002 - ОБЩИЕ ТРЕБОВАНИЯ К РАЗРАБОТКЕ И ДОКУМИНТИРОВАНИЮ – 2003
3. ГОСТ Р 57193-2016 - ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ - 2017
4. Гниденко Ирина Геннадиевна Технология разработки программного обеспечения - 978-5-534-05047-9 2021
5. Кэмпбелл Лейн, Мейджорс Черити Базы данных. Инжиниринг надежности – Питер 2020
6. Туманов В. SQL для хранения, обработки и анализа данных – Салон-Пресс 2021
7. Эдвард Сьорье, Проектирование и реализация систем управления базами данных, 2020 – 461 с.
8. DB-engines [Электронный ресурс] – Режим доступа: https://dbengines.com/en/ranking
9. MyFirstComp [Электронный ресурс] – Режим доступа: https://myfirstcomp.ru/operation\_systems/windows-10-minusy-i-plyusy-operacionnoj-sistemy/
10. Studbooks [Электронный ресурс] – Режим доступа: https://studbooks.net/2256941/informatika/infologicheskaya\_model\_dannyh
11. Systems education [Электронный ресурс] – Режим доступа: https://systems.education/use-case
12. Tproger [Электронный ресурс] – Режим доступа https://tproger.ru/articles/software-development-life-cycle/
13. drach.pro [Электронный ресурс] – Режим доступа: https://drach.pro/blog/hi-tech/item/145-db-comparison
14. habr [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/edison/blog/269789/
15. habr [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/111674/
16. lumpics [Электронный ресурс] – Режим доступа: https://lumpics.ru/pros-and-cons-of-mac-os/
17. ravesli [Электронный ресурс] – Режим доступа: https://ravesli.com/sravnenie-linux-i-windows-v-chem-raznitsa-i-chto-luchshe/
18. statcounter [Электронный ресурс] – Режим доступа https://gs.statcounter.com/windows-version-market-share/desktop/russian-federation
19. trends [Электронный ресурс] – Режим доступа: https://trends.rbc.ru/trends/education/601c1a6b9a79472c4806230a

# **ПРИЛОЖЕНИЕ 1**

**Листинг программного продукта**

Функция Load – генерирует автоматически поля ввода и выпадающие списки.

private void Load(string Value)

{

Colums.Clear();

DataTable DT = new DataTable();

LoadContainer.Children.Clear();

DT = ClassLogic.LoadData("SELECT \* FROM " + Value);

if (Value == "workers")

{

foreach (DataRow item in DT.Rows)

item["Password"] = DEncrypt.Decrypt(item["Password"].ToString());

}

Data.DataContext = DT;

foreach (DataColumn item in DT.Columns)

{

if (item.ColumnName.ToString().Contains("Combo"))

{

Colums.Add(item.ColumnName.ToString());

Label La = new Label();

La.Content = item.ColumnName + ":";

LoadContainer.Children.Add(La);

ComboBox Combo = new ComboBox();

Combo.Name = item.ColumnName + "Combo";

Combo.HorizontalAlignment = HorizontalAlignment.Left;

Combo.Width = 200;

Combo.Height = 25;

List<string> Coll = ClassLogic.LoadCombo("SELECT REFERENCED\_TABLE\_NAME FROM INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE WHERE REFERENCED\_TABLE\_SCHEMA = '" + ClassLogic.Conn.Database + "' AND TABLE\_NAME = '" + Value + "' AND COLUMN\_NAME = '" + item.ColumnName.ToString() + "';", 0, "");

List<string> GetColumns = ClassLogic.LoadCombo("SELECT DISTINCT COLUMN\_Name FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = N'" + Coll[0] + "'", 0, "");

try

{

bool Check = false;

foreach (var itemt in Data.Columns)

{

if (itemt.Header.ToString() == "NumRouteCombo")

{

Combo.ItemsSource = ClassLogic.LoadCombo("SELECT DISTINCT " + Coll[0] + "." + GetColumns[0] + " FROM " + Value + " CROSS JOIN " + Coll[0], 0, "");

Check = true;

}

}

if (Check != true)

{

Combo.ItemsSource = ClassLogic.LoadCombo("SELECT DISTINCT " + Coll[0] + "." + GetColumns[0] + "," + Coll[0] + "." + GetColumns[1] + " FROM " + Value + " CROSS JOIN " + Coll[0], 1, "");

List<string> Tags = ClassLogic.LoadCombo("SELECT DISTINCT " + Coll[0] + "." + GetColumns[0] + "," + Coll[0] + "." + GetColumns[1] + " FROM " + Value + " CROSS JOIN " + Coll[0], 0, "");

if (Int32.TryParse(Tags[0], out int Num) == true)

Combo.Tag = Tags;

}

}

catch (Exception)

{

Combo.ItemsSource = ClassLogic.LoadCombo("SELECT DISTINCT " + Coll[0] + "." + GetColumns[0] + " FROM " + Value + " CROSS JOIN " + Coll[0], 0, "");

}

LoadContainer.Children.Add(Combo);

}

else if (item.ColumnName.ToString().Contains("Password"))

{

Colums.Add(item.ColumnName.ToString());

Label La = new Label();

La.Content = item.ColumnName + ":";

LoadContainer.Children.Add(La);

PasswordBox Pass = new PasswordBox();

Pass.Name = item.ColumnName + "TB";

Pass.HorizontalAlignment = HorizontalAlignment.Left;

Pass.Width = 200;

Pass.Height = 25;

Pass.PasswordChar = '\*';

LoadContainer.Children.Add(Pass);

}

else if (item.ColumnName.Contains("Date"))

{

Colums.Add(item.ColumnName.ToString());

Label La = new Label();

La.Content = item.ColumnName + ":";

LoadContainer.Children.Add(La);

Xceed.WF.Toolkit.DateTimePicker Date = new Xceed.WF.Toolkit.DateTimePicker();

Date.Name = item.ColumnName + "TB";

Date.HorizontalAlignment = HorizontalAlignment.Left;

Date.Width = 200;

Date.Height = 25;

Date.Format = Xceed.WF.Toolkit.DateTimeFormat.Custom;

Date.FormatString = "dd.MM.yyyy hh:mm:ss";

LoadContainer.Children.Add(Date);

}

else if (!item.ColumnName.ToString().Contains("ID"))

{

Colums.Add(item.ColumnName.ToString());

Label La = new Label();

La.Content = item.ColumnName + ":";

LoadContainer.Children.Add(La);

TextBox TB = new TextBox();

TB.Name = item.ColumnName + "TB";

TB.HorizontalAlignment = HorizontalAlignment.Left;

TB.Width = 200;

TB.Height = 25;

LoadContainer.Children.Add(TB);

}

}

}

Функция DataMouseDoubleClick – отвечает за загрузку данных из таблицы в поля ввода.

private void DataMouseDoubleClick(object sender, MouseButtonEventArgs e)

{

int Count = 0;

string[] Mass = new string[Data.Columns.Count];

foreach (DataRowView Row in Data.SelectedItems)

{

for (int i = 0; i < Data.Columns.Count; i++)

Mass[i] = Row.Row.ItemArray[i].ToString();

}

IDS = Mass[0];

if (Int32.TryParse(IDS, out int S) == true & Data.Columns[0].Header.ToString().Contains("ID"))

Count++;

foreach (var item in LoadContainer.Children)

{

if (item.GetType() != typeof(Label))

{

if (item.GetType() == typeof(TextBox))

{

((TextBox)item).Text = Mass[Count];

Count++;

}

else if (item.GetType() == typeof(ComboBox))

{

var Tags = ((ComboBox)item).Tag;

if (Tags != null)

{

try

{

int Num = ((List<string>)Tags).IndexOf(Mass[Count]);

List<string> Items = (List<string>)((ComboBox)item).ItemsSource;

((ComboBox)item).SelectedItem = Items.ElementAt(Num);

}

catch (Exception) { }

}

else

((ComboBox)item).SelectedItem = Mass[Count];

Count++;

}

else if (item.GetType() == typeof(PasswordBox))

{

((PasswordBox)item).Password = Mass[Count];

Count++;

}

else if (item.GetType() == typeof(Xceed.WF.Toolkit.DateTimePicker))

{

((Xceed.WF.Toolkit.DateTimePicker)item).Text = Mass[Count];

Count++;

}

}

}

}

Функция AddValuesToList – добавляет значения в List.

public static void AddValuesToList(StackPanel SP)

{

Values.Clear();

foreach (var item in SP.Children)

{

if (item.GetType() == typeof(TextBox))

Values.Add(((TextBox)item).Text);

else if (item.GetType() == typeof(ComboBox))

{

if (((ComboBox)item).Tag != null)

{

int Num = ((ComboBox)item).Items.IndexOf(((ComboBox)item).Text);

List<string> Tag = (List<string>)((ComboBox)item).Tag;

Values.Add(Tag.ElementAt(Num));

}

else

Values.Add(((ComboBox)item).Text);

}

else if (item.GetType() == typeof(PasswordBox))

Values.Add(DEncrypt.Encrypt(((PasswordBox)item).Password));

else if (item.GetType() == typeof(Xceed.WF.Toolkit.DateTimePicker))

{

DateTime Date = DateTime.Parse(((Xceed.WF.Toolkit.DateTimePicker)item).Text);

string dates = Date.Year + "-" + Date.Month + "-" + Date.Day + " " + Date.Hour + ":" + Date.Minute + ":" + Date.Second;

Values.Add(dates);

}

}

}

Функция Delete – функция удаления записи.

public static void Delete(string EndColumn,string Table,string StartCombo, string Text)

{

try

{

Conn.Open();

MySqlCommand Cmd;

if (EndColumn != StartCombo & EndColumn.Contains("Status"))

Cmd = new MySqlCommand("UPDATE " + Table + " SET " + EndColumn + " = 'Удален' WHERE " + StartCombo + " = " + Text, Conn);

else

Cmd = new MySqlCommand("DELETE FROM "+ Table +" WHERE "+ StartCombo +" = '"+ Text +"'",Conn);

Cmd.ExecuteNonQuery();

MessageBox.Show("Данные были успешно удалены");

Conn.Close();

}

catch (Exception) { Conn.Close(); }

}

Функция Add – функция добавления записи

public static void Add(string NameTable, List<string> Column, List<string> Value)

{

try

{

string Start = "INSERT INTO `" + NameTable + "` (";

foreach (var item in Column)

Start += "`" + item + "`,";

Start = Start.Remove(Start.Length - 1) + ") VALUES (";

foreach (var item in Value)

Start += "'" + item + "',";

Start = Start.Remove(Start.Length - 1) + ");";

Conn.Open();

MySqlCommand Cmd = new MySqlCommand(Start, Conn);

Cmd.ExecuteNonQuery();

MessageBox.Show("Данные успешно добавлены");

Conn.Close();

}

catch (Exception) { Conn.Close(); }

}

Функция Update – функция изменения записи.

public static void Update(string NameTable, List<string> Column, List<string> Value, string WhereValue, string WhereColumn)

{

try

{

string Start = "UPDATE `" + NameTable + "` SET ";

for (int i = 0; i < Column.Count; i++)

Start += "`" + Column[i] + "` = '" + Value[i] + "',";

Start = Start.Remove(Start.Length - 1) + " WHERE `" + WhereColumn + "` = '" + WhereValue + "'";

Conn.Open();

MySqlCommand Cmd = new MySqlCommand(Start, Conn);

Cmd.ExecuteNonQuery();

MessageBox.Show("Данные успешно изменены");

Conn.Close();

}

catch (Exception) { Conn.Close(); }

}

Функция LoadCombo – функция загрузки данных в ComboBox.

public static List<string> LoadCombo(string Command, int SetColumn, string ExtraCommand)

{

List<string> Collection = new List<string>();

try

{

Conn.Open();

MySqlCommand Cmd = new MySqlCommand(Command + ExtraCommand, Conn);

MySqlDataReader DR = Cmd.ExecuteReader();

if (DR.HasRows)

{

while (DR.Read())

Collection.Add(DR[SetColumn].ToString());

}

Conn.Close();

}

catch (Exception) { Conn.Close(); }

return Collection;

}

Функция LoadData – загружает данные в таблицу.

public static DataTable LoadData(string Command)

{

DataTable DT = new DataTable();

try

{

Conn.Open();

MySqlCommand Cmd = new MySqlCommand(Command, Conn);

DT.Load(Cmd.ExecuteReader());

Conn.Close();

}

catch (Exception) { Conn.Close(); }

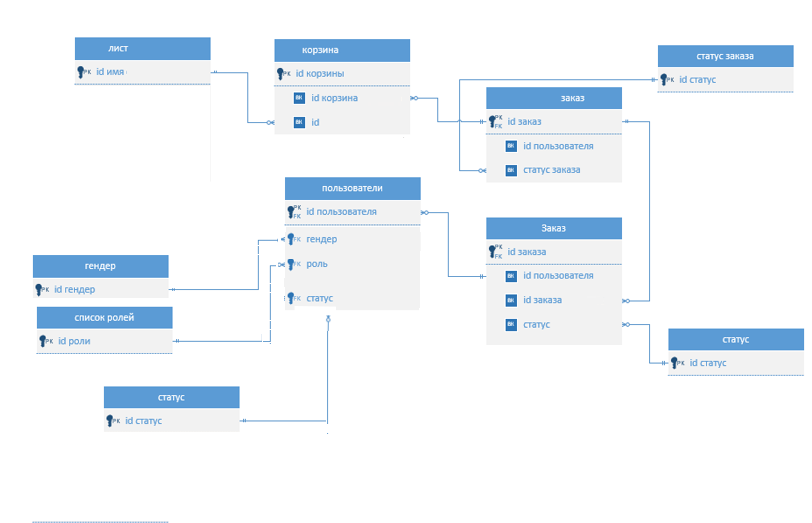
return DT;

# **ПРИЛОЖЕНИЕ 2**

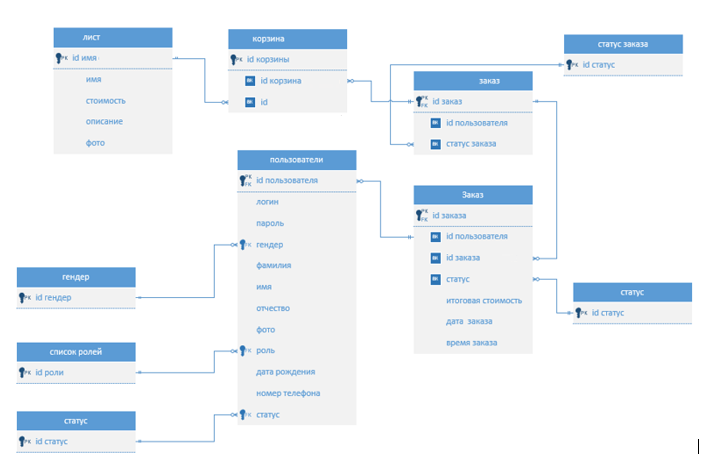
**Раздел проектирования**



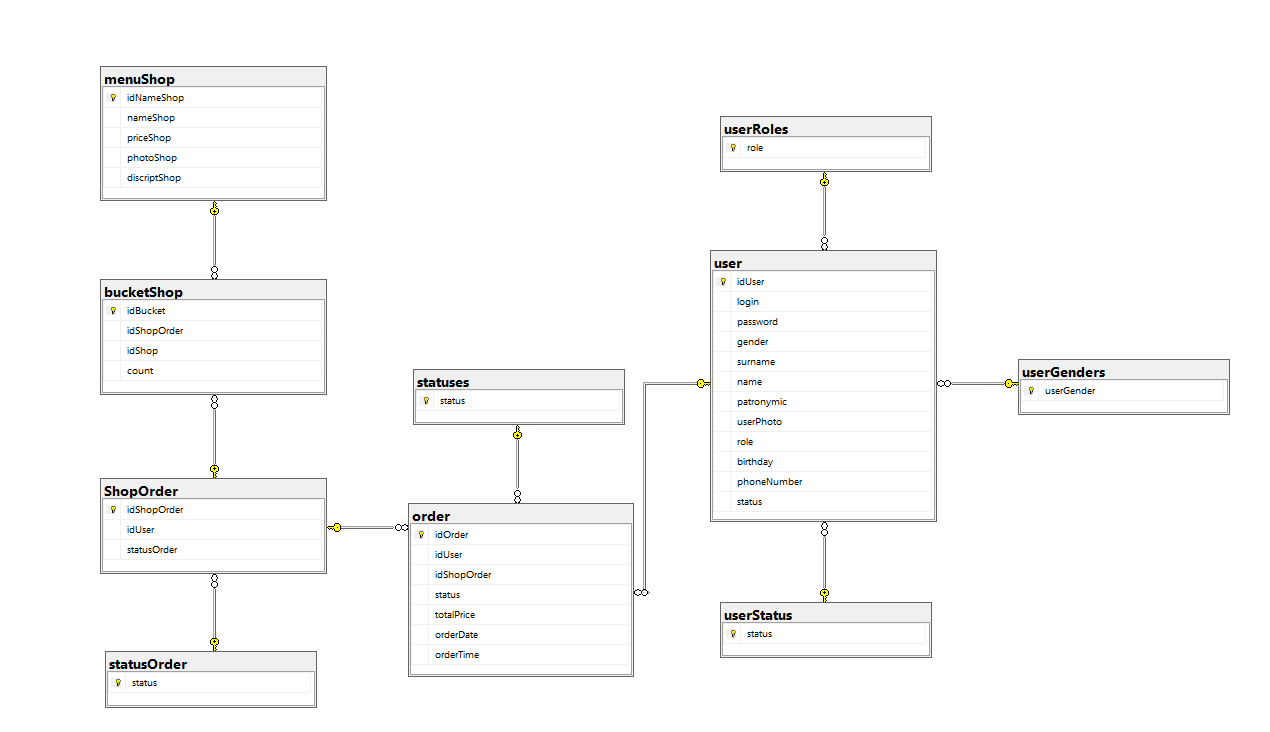
**Рисунок 1 -** Use-case диаграмма



**Рисунок 2** - Концептуальная модель



**Рисунок 3** - Инфологическая модель

****

**Рисунок 4-** ERD

1. statcounter [Электронный̆ ресурс] – Режим доступа: https://gs.statcounter.com/os-market-share/desktop/ [↑](#footnote-ref-1)
2. statcounter [Электронный̆ ресурс] – Режим доступа: https://gs.statcounter.com/windows-version-market-share/desktop/russian-federation [↑](#footnote-ref-2)
3. Github [Электронный ресурс] Режим доступа: https://pypl.github.io/DB.html [↑](#footnote-ref-3)
4. Хританков А. С., Полежаев В. А., Андрианов А. И. Проектирование на UML. Сборник задач по проектированию программных систем. 2-е. изд. – Екатеринбург.: Издательские решения, 2017. – 240 с.; ил. [↑](#footnote-ref-4)
5. Концептуальная модель - Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 c. [↑](#footnote-ref-5)
6. Инфологическая модель - Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 c. [↑](#footnote-ref-6)