Департамент образования и науки Костромской области

Областное государственное бюджетное профессиональное образовательное учреждение «Костромской энергетический техникум имени Ф.В. Чижова»

(ОГБПОУ «КЭТ имени Ф.В. Чижова»)

Областное государственное бюджетное профессиональное образовательное учреждение «Костромской энергетический техникум имени Ф.В. Чижова»

(ОГБПОУ «КЭТ имени Ф.В. Чижова»)

КУРСОВАЯ РАБОТА

Разработка информационной системы для

**курьерской службы доставки**

ПМ.06 Сопровождение информационных систем

МДК.06.03 Устройство и функционирование информационной системы

**специальность:** 09.02.07 «Информационные системы и программирование»

Студент **Солодяшкин А.Ю.***\_\_\_\_\_\_\_\_\_* \_\_\_\_\_\_\_\_\_\_\_\_\_\_.\_\_.20\_\_ г.

*ФИО Подпись*

Оценка выполнения и защиты курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Бессараб С.К. **\_\_\_\_\_\_\_\_\_\_\_\_** \_\_.\_\_.20\_\_ г.

*Подпись*

Кострома 2022

**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ КОСТРОМСКОЙ ОБЛАСТИ**

**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«КОСТРОМСКОЙ ЭНЕРГЕТИЧЕСКИЙ ТЕХНИКУМ ИМ. Ф.В. ЧИЖОВА»**

**специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
|  |  | **УТВЕРЖДАЮ:** |
|  |  | Зам. директора по учебной работе  \_\_\_\_\_\_\_\_\_\_\_(Е.И.Тюрина)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20 г. |
|  |  |  |
|  |  |  |

**КУРСОВОЙ ПРОЕКТ**

**ЗАДАНИЕ**

**Студента 1 группы 3 курса**

**Солодяшкина Андрея Юрьевича**

**2022 год**

**Тема курсового проекта:**

Разработка информационной системы для курьерской службы доставки

1. Содержание пояснительной записки

ТИТУЛЬНЫЙ ЛИСТ

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ

ВВЕДЕНИЕ

ГЛАВА 1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ

* 1. Краткие сведения об информационной системе
  2. Общие требования к информационной системе
  3. Разработка технического задания на разработку информационной системы
  4. Обоснование необходимости внедрения и использования информационной системы

ГЛАВА 2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1. Физическая структура базы данных

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Разработка кода информационной системы

ГЛАВА 4. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

4.1. План тестирования

4.2. Написание тест-кейсов

4.3. Юнит-тестирование

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

СПИСОК ИЛЛЮСТРАЦИЙ

ПРИЛОЖЕНИЯ

Приложение №1 Инструкции для использования системы (руководство пользователя);

Приложение №2 Техническое задание;

Приложение №3 ER-ДИАГРАММА;

Приложение №4 ЛИСТИНГ КОДА;

Приложение №5 Ссылка на облачно хранилище и репозиторий с курсовой работой.

1. Графическая часть
2. ER-ДИАГРАММА;
3. ЛИСТИНГ КОДА;
4. Электронная часть

Содержимое папки с курсовым проектом в облачном хранилище, на репозитории и на электронном носителе:

1. Папка - Практическая часть (информационная система, исходный код заархивированный; скрипт базы данных);

2. Файл - Текст курсовой работы (в формате .docx) и распечатать. Объем 30 страниц без приложения;

3. Файл - Приложения (техническое задание, диаграммы, скриншоты основных окон ИС, листинг кода);

4. Папка - Презентация и доклад (время на выступление максимум 5 -7 мин, 15 слайдов максимум);

5. Файл - Видео (с демонстрацией работы основного функционала не больше 1 мин (видео в формате mp4 объем до 10 Мб);

6. Файл - readme.txt (внутри: ФИО, тема курсовой работы, ссылка на облачное хранилище, ссылка на репозиторий);

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания: \_\_\_\_\_\_\_

Срок выполнения: \_\_\_\_\_\_\_\_\_\_

Руководитель курсового проекта: преподаватель \_\_\_\_\_\_\_\_\_ (Бессараб С.К.)

Рассмотрено и утверждено на заседании ПЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ПЦК \_\_\_\_\_\_\_\_\_(Ищук О.В.)

ОГЛАВЛЕНИЕ

[ГОЛОСАРИЙ 5](#_Toc105660771)

[ВВЕДЕНИЕ 6](#_Toc105660772)

[1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ 7](#_Toc105660773)

[1.1. Краткие сведения об информационной системе 7](#_Toc105660774)

[1.2. Общие требования к информационной системе 8](#_Toc105660775)

[1.3. Разработка технического задания на разработку информационной системы 10](#_Toc105660776)

[1.4. Обоснование необходимости внедрения и использования информационной системы 11](#_Toc105660777)

[2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ 13](#_Toc105660778)

[2.1. Физическая структура базы данных 14](#_Toc105660779)

[3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc105660780)

[Разработка кода информационной системы 17](#_Toc105660781)

[4. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 35](#_Toc105660782)

[Тест план 35](#_Toc105660783)

[Написание тест-кейсов 36](#_Toc105660784)

[Юнит-тестирование 37](#_Toc105660785)

[ЗАКЛЮЧЕНИЕ 39](#_Toc105660786)

[Список литературы 40](#_Toc105660787)

[**Список иллюстраций** 41](#_Toc105660788)

[ПРИЛОЖЕНИЯ 42](#_Toc105660789)

[Приложение №1 42](#_Toc105660790)

[Приложение №2 43](#_Toc105660791)

[Приложение №3 44](#_Toc105660792)

[Приложение №4 45](#_Toc105660793)

[Приложение №5 46](#_Toc105660794)

ГОЛОСАРИЙ

**Курьерская служба** — это профессиональная организация, специализирующаяся на оперативной доставке грузов, деловой корреспонденции и документов.

**ЗАКАЗ** - по праву РФ вид оферты - предложение заказчика (покупателя) изготовить, поставить (продать) ему продукцию (товары), выполнить работу, оказать услуги с определением сроков, объемов, количества, ассортимента, качества и других необходимых данных. По договору поставки может представляться покупателем в форме спецификации. Согласие подрядчика, поставщика (продавца) оформляется письмом, телеграммой либо подтверждается подписью на документе

**Товар** - как объект, созданный для обмена и несущий какую-то пользу для потребителя.

**Груз** - перемещаемый кем-либо или чем-либо в пространстве объект.

Документы - зафиксированная на материальном носителе информация в виде текста, звукозаписи или изображения с реквизитами, позволяющими её идентифицировать.

**Клиент** - лицо или учреждение, обслуживаемое какой-л. кредитной, торговой или промышленной организацией.

**Пользователь** - лицо, участвующее в функционировании автоматизированной системы или использующее результаты её функционирования.

ВВЕДЕНИЕ

Сегодня направление доставки товаров, документов и других грузов, в связи с развитием Интернет-торговли и стремлением бизнеса доставить своему потребителю ценность в кратчайшие сроки, стремительно набирает обороты. Направление доставки может быть как основным для компаний, оказывающих курьерские и транспортные услуги, так и вспомогательным для торговых предприятий, интернет-магазинов или фирм, предоставляющих услуги выездным способом, доставку в пункты выдачи.

Основными потребителями услуг курьерской службы являются компании разных сфер бизнеса. Наиболее важными факторами при заказе пересылки становятся: стоимость услуг, сроки доставки, опыт и репутация фирмы-исполнителя. Услугами курьерской службы пользуются в основном корпоративные клиенты. Ими становятся банки, страховые компании, автопроизводители, рекламные фирмы, производители электроники, издательский бизнес, фарминдустрия, общественное питание.

Таким образом, данная тема является **актуальной.**

**Объект исследования:** курьерская служба доставки.

**Предмет исследования:** процессы по сбору заказов.

**Цель:** Разработка информационной системы (ИС) для курьерской службы доставки.

**Задачи:**

1. Провести анализ предметной области;
2. Составить техническое задание;
3. Выделить основные требования к системе;
4. Спроектировать базу данных;
5. Создать программное обеспечение (ПО) для данной системы;
6. Провести тестирование системы.

На основании вышеперечисленных целей и задач была разработана информационная система, удовлетворяющая требованиям заказчика.

1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ

В данной главе описывается краткая информация об информационной системе, общие требования, разработка технического задания, обоснование необходимости использования информационной системы.

Информационная система (ИС) — система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы, которые обеспечивают и распространяют информацию.

Данная информационная система предназначения для учёта вызовов скорой помощи.

* 1. Краткие сведения об информационной системе

В данном разделе даётся краткая информация об информационной системе. Она представлена в таблице 1.

Таблица 1 Краткие сведения об информационной системе

**«Краткая информация об информационной системе»**

|  |  |
| --- | --- |
| Название | Описание |
| Цель | Создание десктопного приложения для информационной системы для курьерской службы доставки |
| Предметная область | «Курьерская служба доставки» |
| Основные объекты рассматриваемой предметной области | Заказы, клиент, груз; |
| Какой процесс будет автоматизирован | оформление заказа |
| Платформа | Windows |
| Язык | C# |
| Технология | WPF |
| База данных | MS SQL |

Таким образом, сведения, представленные в таблице, дают нам возможность быстро войти в курс разработки информационной системы.

1.2. Общие требования к информационной системе

В данном разделе описываются функциональные и нефункциональные требования к информационной системе.

При разработке программного обеспечения **функциональные требования** определяют функции, которые должно выполнять все приложение или только один из его компонентов. Функция состоит из трех шагов: ввод данных — поведение системы — вывод данных. Он может вычислять, манипулировать данными, выполнять бизнес-процессы, устанавливать взаимодействие с пользователем или выполнять любые другие задачи.

Другими словами, функциональное требование — это то, ЧТО приложение должно или не должно делать после ввода некоторых данных.

Функциональные требования важны, поскольку они показывают разработчикам программного обеспечения, как должна вести себя система. Если система не соответствует функциональным требованиям, значит, она не работает должным образом.

**Нефункциональные требования** определяют стандарты производительности и атрибуты качества программного обеспечения, например удобство использования системы, эффективность, безопасность, масштабируемость и т.д.

В то время как функциональные требования определяют, что делает система, нефункциональные требования описывают, КАК система это делает. Например, веб-приложение должно обрабатывать более 15 миллионов пользователей без какого-либо снижения производительности, или веб-сайт не должен загружаться более 3 секунд.

Если приложение не соответствует нефункциональным требованиям, оно продолжает выполнять свои основные функции, однако не сможет обеспечить удобство для пользователя.

Нефункциональные требования важны, поскольку они помогают разработчикам программного обеспечения определять возможности и ограничения системы, которые необходимы для разработки высококачественного программного обеспечения. Следовательно, нефункциональные требования так же важны, как и функциональные требования для успешного внедрения продукта [1].

В соответствии с требованиями к ИС, были выделены следующие функциональные и нефункциональные требования к ИС:

**Функциональные требования к информационно системе**

* При в ходе нового пользователя приложение должно ему предложить:
  + - Регистрацию
    - Войти
* При регистрации пользователь должен ввести:
  + - Фамилию
    - Имя
    - E-mail
    - Пароль
* После регистрации пользователя должно перекинуть на начальную страницу
* После входа пользователя как клиента его должно перекинуть на страницу, где просматриваются оставленные заказы и есть кнопка для оформления заказа
* При нажатии на кнопку «оформить заказ» клиента должно перекинуть на страницу оформления заказа, где ему необходимо указать:
  + - Куда
    - Откуда
    - Что отправляете
    - Тип оплаты
      * При отправке
      * При получении
    - Тип доставки
      * Обычная
      * Экспресс
* После нажатия на кнопку «Оформить заказ» клиента должно перекинуть на страницу, где сообщается об успешном оформлении заказа.
* На странице «успешного оформления заказа» должна присутствовать кнопка для возвращения на страницу с оформленными заказами, где должна присутствовать карточка с оформленным заказом.
* Карточка с оформленным заказом должна содержать следующую информацию:
  + - Откуда
    - Куда
    - Что отправляется
    - Тип оплаты
    - Тип доставки
    - Стоимость

**Нефункциональные требования к информационно системе**

* Отказоустойчивость
* Возможность одновременной работы нескольких пользователей
* Интуитивность

Таким образом, сведения, данные в этом разделе, дают нам понять функциональные и нефункциональные требования к информационной системе.

1.3. Разработка технического задания на разработку информационной системы

**Техническое задание (ТЗ):** Исходный технический документ для проведения работы, устанавливающий требования к создаваемому изделию (его СЧ или КИМП) и технической документации на него, а также требования к объему, срокам проведения работы и форме представления результатов.

Документирование требований регламентировано российскими ГОСТ 19.201-78 "Техническое задание, требования к содержанию и оформлению" и ГОСТ 34.602-89 "Техническое задание на создание автоматизированной системы" (ТЗ на АС).

Второй документ, по сути, является более проработанной версией первого, адаптированной к созданию автоматизированных информационных систем, поэтому далее рассмотрена структура ТЗ в соответствии с ГОСТ 34.602-89 [2] [3].

На основании документации было составлено ТЗ, с которым можно ознакомиться в приложении №2.

1.4. Обоснование необходимости внедрения и использования информационной системы

Данная система является настольным приложением, так как такой подход обеспечивает следующие преимущества:

* **Функциональность**. Заказывая персональную разработку, вы получаете уникальные функции для бизнеса, которые помогут вашей компании. Главная особенность – широкие возможности для функционала. Можно реализовать практически любую идею. При этом интерфейс будет удобным, привычным и интуитивно понятным для пользователей.
* **Безопасность и надежность**. Браузерные приложения уязвимы. Дело в том, что разработчики не могут повлиять на безопасность браузера, а веб-приложение напрямую зависит от его работы. Десктоп приложение – это гарантия безопасности и сохранности данных. Поэтому многие компании выбирают такое цифровое решение для своего предприятия.
* **Мобильность**. Десктоп приложения можно разрабатывать под различные операционные системы и даже использовать с мобильного устройства, сделав нативную версию. Но для синхронизации данных необходимо подключаться к интернету. Тем не менее, мобильность подразумевает использование приложения в любых условиях и дает возможность управлять бизнес-процессами компании когда угодно.
* **Быстродействие**. Принцип работы Desktop приложения прост: программа использует ресурсы компьютера, его память и возможности. С одной стороны, это может стать недостатком, если оборудование недостаточно хорошее. Но с другой, заказывая индивидуальную разработку, учитываются особенности системы. С должной оптимизацией такое приложение будет «летать» и выполнять задачи эффективно.

Таким образом, стало ясно, что выбор настольного приложения для системы курьерской службы доставки оказался правильным, потому что использование моего программного продукта предполагается корпоративными клиентами, для доставки важных бумаг, документов и конвертов с неизвестным содержимым.

1. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ

**База данных (БД)** — это программа, которая позволяет хранить и обрабатывать информацию в структурированном виде. БД это отдельная независимая программа, которая не входит в состав языка программирования. В базе данных можно сохранять любую информацию, чтобы позже получать к ней доступ [4].

**Система управления базами данных (СУБД**) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. Иными словами, СУБД — это набор программ, позволяющий организовывать, контролировать и администрировать базы данных. Большинство сайтов не могут функционировать без базы данных, поэтому СУБД используется практически повсеместно [5].

**Этапы проектирования** и создания базы данных определяются следующей последовательностью:

* построение информационно-логической модели данных предметной области;
* определение логической структуры реляционной базы данных;
* конструирование таблиц базы данных;
* создание схемы данных;
* ввод данных в таблицы (создание записей);
* разработка необходимых форм, запросов, макросов, модулей, отчетов;
* разработка пользовательского интерфейса [6].

**Таблицы** – Являются основным объектом любой БД, который хранит все имеющиеся в базе данные, а также саму структуру базы (поля, их типы и свойства)

**Запросы** – Используются при извлечении данных из таблиц и предоставлении их в удобном для пользователя виде. С их помощью отбираются данные, сортируются и фильтруются. Кроме того, можно преобразовывать данные по заданному алгоритму, создавать новые таблицы, автоматически заполнять таблицы данными из других источников, выполнять простейшие вычисления в таблицах и многое другое [7].

С ER диаграммой данной ИС Вы можете ознакомится в приложении №3.

* 1. Физическая структура базы данных

**Физическая структура базы данных**. Физическая организация БД - это совокупность методов и средств размещения данных во внешней памяти и созданная на их основе внутренняя (физическая) модель данных.

При работе с БД необходимо взять за основу базовые запросы, которые называют CRUD.

**CRUD** (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») — акроним, обозначающий четыре базовые функции, используемые при работе с персистентными хранилищами данных.

Запросы к базе данных ИС.

Для демонстрации работы запросов мы из представили на скриншотах ниже (См. Рис 2-9).

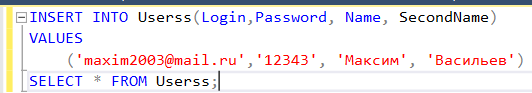


Рисунок 1 Регистрация нового пользователя

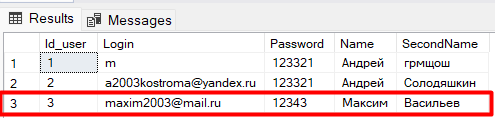


Рисунок 2 Результат. Новый пользователь успешно зарегистрирован

Данный запрос к БД формирует заказ для пользователя, которого мы регистрировали ранее.

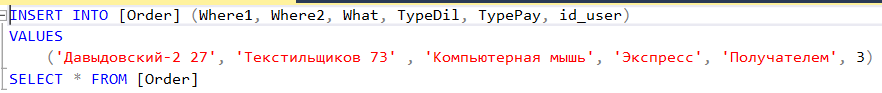


Рисунок 3 Запрос. Создание заказа

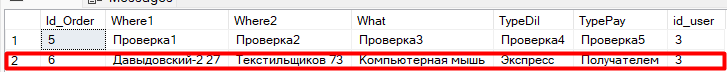


Рисунок 4 Результат. Заказа в списке заказов

Данный запрос к БД показывает какие заказы делал наш созданный пользователь.

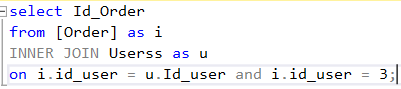


Рисунок 5 Запрос. Ввывод информации о заказах, сделанных пользователем

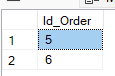


Рисунок 6 Результат. Ввывод информации по заказам сделанным пользователем

Таким образом, при помощи запросов к БД, можно уже определить, как будет работать функционал в данной ИС.

1. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

Разработка программного обеспечения (англ. software development) — деятельность по созданию нового программного обеспечения.

Общие этапы разработки

С# – это объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++.

Его особенности:

* статистическая типизация;
* поддерживается полиморфизм;
* поддерживается перегрузка операторов;
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android [8].

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#, VB.NET и F#, либо совмещать и то, и другое [9].

XAML (eXtensible Application Markup Language) - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF (а также к Silverlight) данный язык используется прежде всего для создания пользовательского интерфейса декларативным путем. Хотя функциональность XAML только графическими интерфейсами не ограничивается: данный язык также используется в технологиях WCF и WF, где он никак не связан с графическим интерфейсом [10].

Также для работы с базой данных через C# понадобилось использовать язык запросов LINQ.

LINQ (Language-Integrated Query) представляет простой и удобный язык запросов к источнику данных. В качестве источника данных может выступать объект, реализующий интерфейс IEnumerable (например, стандартные коллекции, массивы), набор данных DataSet, документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных.

Разработка кода информационной системы

Сперва было создано начальное окно (см. Рис 7), в котором у пользователя есть выбор:

* Авторизоваться в ранее зарегистрированный аккаунт, нажав кнопку «Войти»
* Зарегистрироваться, нажав кнопку «Зарегистрироваться»

Код xaml представлен ниже:

<Grid Background="#F5DD00">

<TextBlock HorizontalAlignment="Center" Foreground="Red" FontSize="30" Margin="0,50,0,0" VerticalAlignment="Top" >\_R-C-D\_</TextBlock>

<Button x:Name="btnLog"

Width="300"

Height="40"

Margin="0,100,0,0"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="Black"

Content="Войти"

FontSize="17"

Grid.Row="0"

Click="btnLog\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFFFFF" Offset="0" />

<GradientStop Color="#FFFFFF" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

<Button x:Name="btnReg"

Width="300"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="Black"

Content="Регистрация"

FontSize="17"

Grid.Row="0"

Click="btnReg\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFFFFF" Offset="0" />

<GradientStop Color="#FFFFFF" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

</Grid>

Код cs:

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void btnLog\_click(object sender, RoutedEventArgs e)

{

Login login = new Login();

login.Show();

Close();

}

private void btnReg\_click(object sender, RoutedEventArgs e)

{

Reg reg = new Reg();

reg.Show();

Close();

}

}



Рисунок 7Начальное окно

На странице «Регистрация» (рис.8) пользователь регистрирует новый аккаунт, введя следующие данные:

* Электронную почту (она же является логином для входа в аккаунт)
* Пароль
* Имя
* Фамилия

Страницы добавляются во Frame в окне, тем самым окно никуда не исчезает, а страницы меняются. Также страницы получают данные из БД.

Код xaml представлен ниже:

<Grid Background="#F5DD00">

<Grid.RowDefinitions>

<RowDefinition Height="100"></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition Height="100"></RowDefinition>

</Grid.RowDefinitions>

<TextBlock HorizontalAlignment="Center" Foreground="Red" FontSize="30" Margin="0,50,0,0" VerticalAlignment="Top" >\_R-C-D\_</TextBlock>

<Border Grid.Row="1" Width="300" Height="40" Margin="0,0,0,0" CornerRadius="10" VerticalAlignment="Top"

Background="White">

<Grid >

<Rectangle StrokeThickness="1"/>

<TextBox Margin="1"

BorderThickness="0"

Text="{Binding Login}"

Background="Transparent"

VerticalAlignment="Center"

Padding="5"

Foreground="Black"

x:Name="LoginTb"

>

</TextBox>

<TextBlock IsHitTestVisible="False"

Text="Введите Email"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="10,0,0,0"

FontSize="11"

Foreground="DarkGray"

Grid.Column="1"

>

<TextBlock.Style>

<Style TargetType="{x:Type TextBlock}">

<Style.Triggers>

<DataTrigger Binding="{Binding Text , ElementName=LoginTb}" Value="">

<Setter Property="Visibility" Value="Visible"></Setter>

</DataTrigger>

</Style.Triggers>

<Setter Property="Visibility" Value="Collapsed"></Setter>

</Style>

</TextBlock.Style>

</TextBlock>

</Grid>

</Border>

<!--Пароль-->

<Border Background="White"

CornerRadius="10"

Width="300"

Height="40"

Grid.Row="1"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,70,0,0"

>

<Grid >

<Rectangle StrokeThickness="1"/>

<TextBox Margin="1"

BorderThickness="0"

Background="Transparent"

VerticalAlignment="Center"

Padding="5"

Text="{Binding Password}"

Foreground="Black"

x:Name="PasswordTb"

>

</TextBox>

<TextBlock IsHitTestVisible="False"

Text="Введите пароль"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="10,0,0,0"

FontSize="11"

Foreground="DarkGray"

Grid.Column="1"

>

<TextBlock.Style>

<Style TargetType="{x:Type TextBlock}">

<Style.Triggers>

<DataTrigger Binding="{Binding Text , ElementName=PasswordTb}" Value="">

<Setter Property="Visibility" Value="Visible"></Setter>

</DataTrigger>

</Style.Triggers>

<Setter Property="Visibility" Value="Collapsed"></Setter>

</Style>

</TextBlock.Style>

</TextBlock>

</Grid>

</Border>

<!--Повторение пароля-->

<Border Background="White"

CornerRadius="10"

Width="300"

Height="40"

Grid.Row="1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Margin="0,70,0,0"

>

<Grid >

<Rectangle StrokeThickness="1"/>

<TextBox Margin="1"

BorderThickness="0"

Background="Transparent"

VerticalAlignment="Center"

Padding="5"

Foreground="Black"

Text="{Binding Name}"

x:Name="NameTb"

>

</TextBox>

<TextBlock IsHitTestVisible="False"

Text="Введие имя"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="10,0,0,0"

FontSize="11"

Foreground="DarkGray"

Grid.Column="1"

>

<TextBlock.Style>

<Style TargetType="{x:Type TextBlock}">

<Style.Triggers>

<DataTrigger Binding="{Binding Text , ElementName=NameTb}" Value="">

<Setter Property="Visibility" Value="Visible"></Setter>

</DataTrigger>

</Style.Triggers>

<Setter Property="Visibility" Value="Collapsed"></Setter>

</Style>

</TextBlock.Style>

</TextBlock>

</Grid>

</Border>

<!--Повторение пароля-->

<Border Background="White"

CornerRadius="10"

Width="300"

Height="40"

Grid.Row="1"

HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,0,0"

>

<Grid >

<Rectangle StrokeThickness="1"/>

<TextBox Margin="1"

BorderThickness="0"

Background="Transparent"

VerticalAlignment="Center"

Padding="5"

Foreground="Black"

x:Name="SecondTb"

Text="{Binding SecondName}"

>

</TextBox>

<TextBlock IsHitTestVisible="False"

Text="Введите Фамилию"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="10,0,0,0"

FontSize="11"

Foreground="DarkGray"

Grid.Column="1"

>

<TextBlock.Style>

<Style TargetType="{x:Type TextBlock}">

<Style.Triggers>

<DataTrigger Binding="{Binding Text , ElementName=SecondTb}" Value="">

<Setter Property="Visibility" Value="Visible"></Setter>

</DataTrigger>

</Style.Triggers>

<Setter Property="Visibility" Value="Collapsed"></Setter>

</Style>

</TextBlock.Style>

</TextBlock>

</Grid>

</Border>

<Button x:Name="btnLog"

Width="300"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="Black"

Content="Войти"

FontSize="17"

Grid.Row="2"

Click="btnLog\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFFFFF" Offset="0" />

<GradientStop Color="#FFFFFF" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

</Grid>

Код cs:

public partial class Reg : Window

{

private Userss \_currentUser = new Userss();

public Reg()

{

DataContext = \_currentUser;

InitializeComponent();

}

private void btnLog\_click(object sender, RoutedEventArgs e)

{

bool canRegister = Entities1.GetContext().Usersses.All(p => p.Login != LoginTb.Text.Trim());

if (canRegister)

{

Entities1.GetContext().Usersses.Add(\_currentUser);

Entities1.GetContext().SaveChanges();

MessageBox.Show("Вы прошли регистрацию");

Login login = new Login();

login.Show();

Close();

}

else

{

MessageBox.Show("Вы не прошли регистрацию, такой Email существует");

}

}

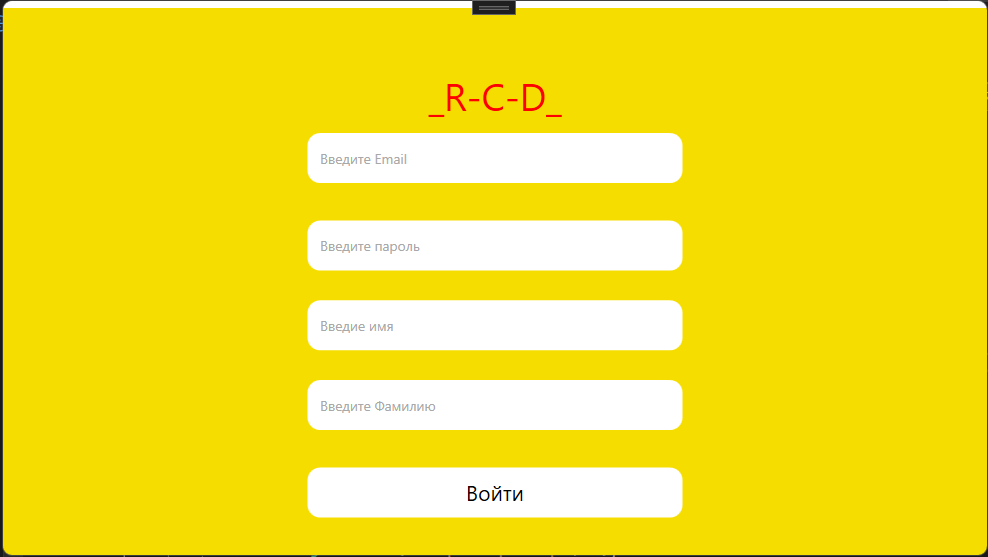


Рисунок 8 Страница "Регистрация"

Главная страница. В зависимости от пользователя показывает все его оформленные заказы или их отсутствие (см. Рис 9, 10).

Код xaml представлен ниже:

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="100"></RowDefinition>

</Grid.RowDefinitions>

<ListView x:Name="ListOrd" Background="Gray">

<ListView.ItemTemplate>

<DataTemplate>

<Border HorizontalAlignment="Center" Background="White" Width="700" Height="250" CornerRadius="20">

<StackPanel Orientation="Horizontal">

<StackPanel Orientation="Vertical" Width="150" HorizontalAlignment="Left">

<TextBlock Text="Откуда" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock Text="Куда" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock Text="Что отправляеться" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock Text="Тип оплаты" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock Text="Тип доставки" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

</StackPanel>

<StackPanel Width="400">

<TextBlock x:Name="where1" Text="{Binding Where1}" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock x:Name="where2" Text="{Binding Where2}" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock x:Name="WhatTb" Text="{Binding What}" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock x:Name="PayTb" Text="{Binding TypePay}" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

<TextBlock x:Name="Dilivery" Text="{Binding TypeDil}" FontWeight="Bold" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"></TextBlock>

</StackPanel>

<TextBlock x:Name="Cash" Grid.Column="2" FontSize="30" HorizontalAlignment="Center" VerticalAlignment="Center">500p</TextBlock>

<StackPanel Grid.Column="2" Width="70" Height="250" HorizontalAlignment="Center" >

<Button x:Name="Red" Height="40" VerticalAlignment="Top" HorizontalAlignment="Right" Background="Transparent" BorderThickness="0" BorderBrush="Transparent" FontSize="30" Margin="0,0,0,150" Click="Red\_Click">/..</Button>

<Button x:Name="close" Height="40" VerticalAlignment="Bottom" HorizontalAlignment="Right" Background="Transparent" BorderThickness="0" BorderBrush="Transparent" FontSize="30" Margin="0,0,0,0" Click="close\_Click">X</Button>

</StackPanel>

</StackPanel>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<StackPanel Grid.Row="1" Background="Gray"></StackPanel>

<Button x:Name="btnLog"

Width="300"

Height="40"

VerticalAlignment="Bottom"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="Black"

Content="Оформить заказ"

FontSize="17"

Grid.Row="1"

Click="btnLog\_click" Margin="246,0,246,10">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FAFF00" Offset="0" />

<GradientStop Color="#FAFF00" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

</Grid>

Код cs представлен ниже:

public partial class User : Page

{

private int iduse;

public User(int id)

{

InitializeComponent();

iduse = id;

if (Entities1.GetContext().Orders.FirstOrDefault(p => p.id\_user == iduse) != null)

{

ListOrd.ItemsSource = Entities1.GetContext().Orders.Where(p => p.id\_user == iduse).ToList();

}

}

private void btnLog\_click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Orders(null , iduse));

}

private void close\_Click(object sender, RoutedEventArgs e)

{

Order order = Entities1.GetContext().Orders.Where(p => p.id\_user == iduse).FirstOrDefault();

Entities1.GetContext().Orders.Remove(order);

Entities1.GetContext().SaveChanges();

MessageBox.Show("Вы удалили");

ListOrd.ItemsSource = Entities1.GetContext().Orders.Where(p => p.id\_user == iduse).ToList();

}

private void Red\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Orders((sender as Button).DataContext as Order, iduse));

}

}

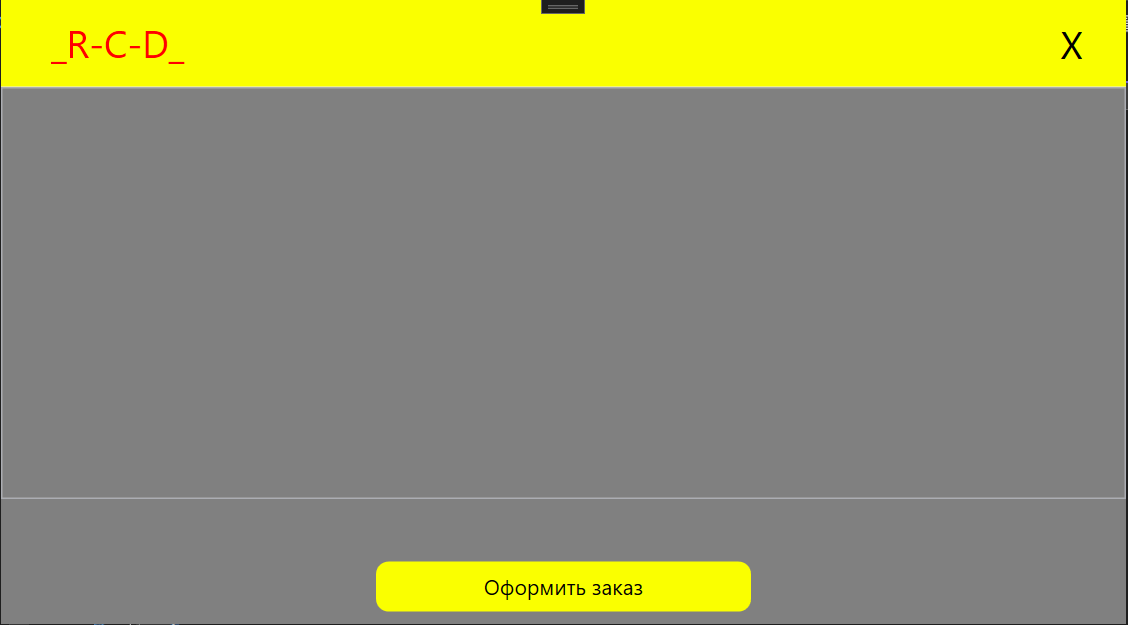


Рисунок 9 Страница «Главная (без заказа)»

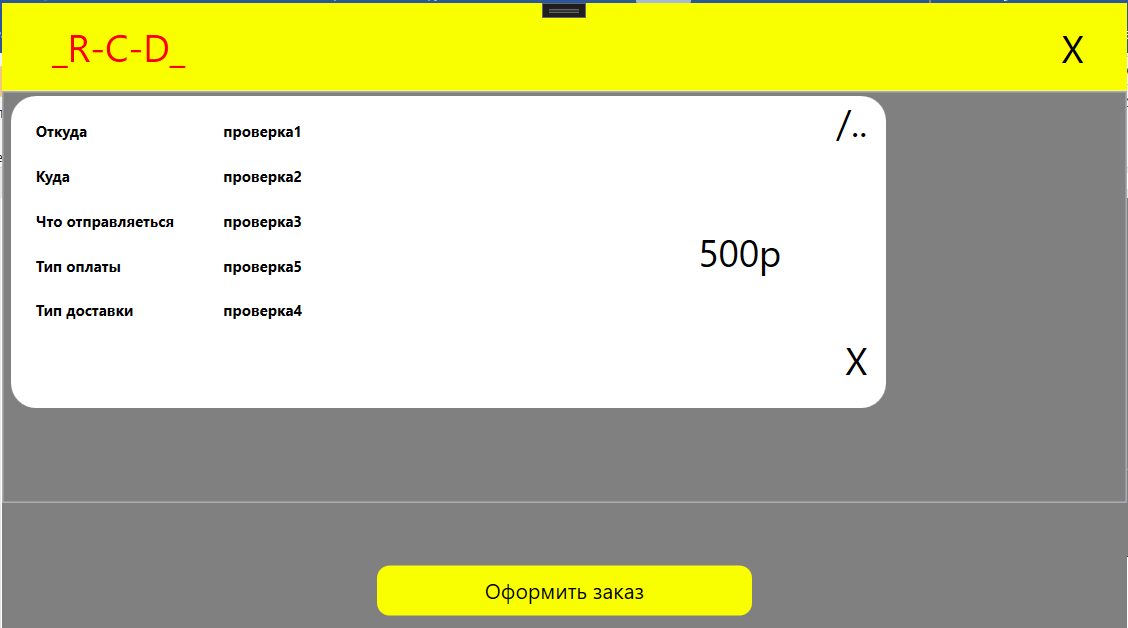


Рисунок 10 Страница «Главная(с заказом)»

Окно оформления заказа (рис.11). Здесь пользователь для оформления заказа должен ввести:

* Куда
* Откуда
* Что Отправляете
* Тип доставки (экспресс, обычная)
* Тип оплаты (Отправителем, получателем)

Код xaml представлен ниже:

<Grid Background="#ECECEC">

<Grid.RowDefinitions>

<RowDefinition Height="0"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<TextBox x:Name="Where1Tb" Text="{Binding Where1}" FontSize="25" Grid.Row="1" Width="300" Height="50" Margin="0,30,0,0" VerticalAlignment="Top"></TextBox>

<TextBox x:Name="Where2Tb" Text="{Binding Where2}" FontSize="25" Grid.Row="1" Width="300" Height="50" Margin="0,100,0,0" VerticalAlignment="Top"></TextBox>

<TextBox x:Name="WhatSendTb" Text="{Binding What}" FontSize="25" Grid.Row="1" Width="300" Height="50" Margin="0,00,0,60" VerticalAlignment="Center"/>

<TextBox x:Name="TdeliveryTb" Text="{Binding TypeDil}" FontSize="25" Grid.Row="1" Width="300" Height="50" Margin="0,80,0,0" VerticalAlignment="Center"/>

<Button x:Name="btnLog"

Width="300"

Height="40"

VerticalAlignment="Bottom"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="Black"

Content="Оформить заказ"

FontSize="17"

Grid.Row="1"

Click="btnLog\_click" Margin="246,0,246,10">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FAFF00" Offset="0" />

<GradientStop Color="#FAFF00" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

<TextBox x:Name="TpaymentTb" Text="{Binding TypePay}" FontSize="25" Grid.Row="1" Width="300" Height="50" Margin="0,220,0,0" VerticalAlignment="Center"/>

<TextBlock HorizontalAlignment="Left" Margin="177,46,0,0" Grid.RowSpan="2" TextWrapping="Wrap" Text="Куда" VerticalAlignment="Top"/>

<TextBlock HorizontalAlignment="Left" Margin="177,119,0,0" Grid.RowSpan="2" TextWrapping="Wrap" Text="Откуда" VerticalAlignment="Top"/>

<TextBlock HorizontalAlignment="Left" Margin="157,186,0,0" Grid.RowSpan="2" TextWrapping="Wrap" Text="Что Отправляете" VerticalAlignment="Top"/>

<TextBlock HorizontalAlignment="Left" Margin="177,256,0,0" Grid.RowSpan="2" TextWrapping="Wrap" Text="Тип доставки" VerticalAlignment="Top"/>

<TextBlock HorizontalAlignment="Left" Margin="177,329,0,0" Grid.RowSpan="2" TextWrapping="Wrap" Text="Тип оплаты" VerticalAlignment="Top"/>

</Grid>

Код cs представлен ниже:

public partial class Orders : Page

{

private int idus;

private Order \_currentorder = new Order();

public Orders(Order selectedOrder, int id)

{

idus = id;

InitializeComponent();

if (selectedOrder != null)

{

\_currentorder = selectedOrder;

}

DataContext = \_currentorder;

}

private void btnLog\_click(object sender, RoutedEventArgs e)

{

if (\_currentorder.Id\_Order == 0)

{

Entities1.GetContext().Orders.Add(new Order {Where1 = Where1Tb.Text , Where2 = Where2Tb.Text , id\_user = idus ,TypeDil =TdeliveryTb.Text, TypePay = TpaymentTb.Text, What = WhatSendTb.Text});

}

Entities1.GetContext().SaveChanges();

MessageBox.Show("Спасибо, что выбрали нашу доставку!!!");

Manager.MainFrame.Navigate(new User(idus));

}

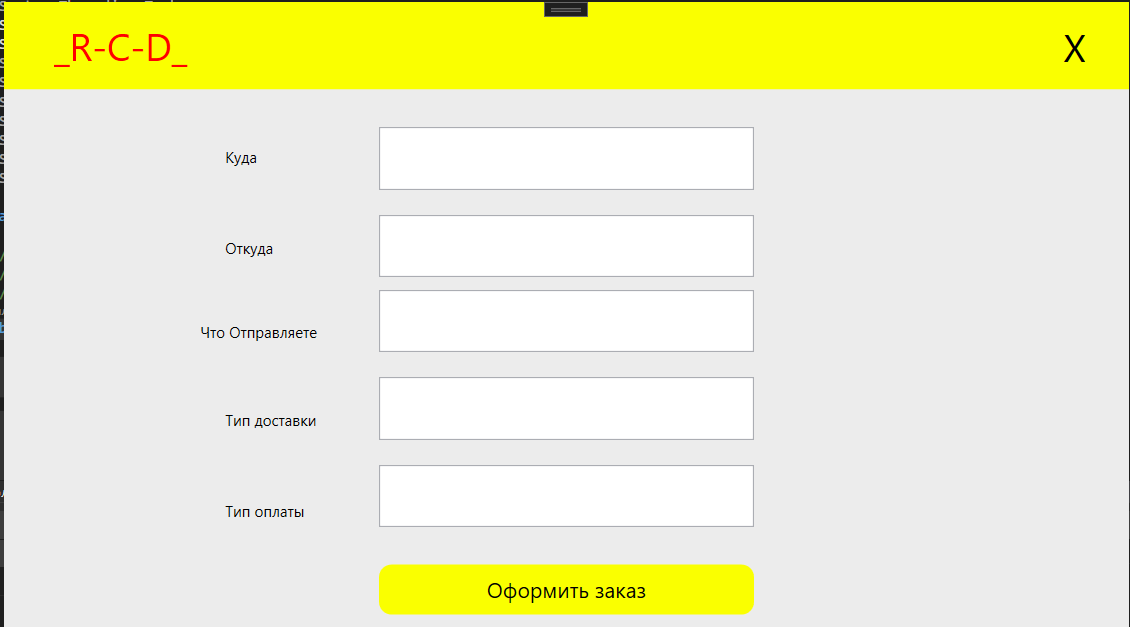
} 

Рисунок 11 Страница «Оформление заказа»

1. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

**Тестирование программного обеспечения** (Software Testing) — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

Этапы тестирования:

1. Анализ продукта
2. Работа с требованиями
3. Разработка стратегии тестирования и планирование процедур контроля качества
4. Создание тестовой документации
5. Тестирование прототипа
6. Основное тестирование
7. Стабилизация
8. Эксплуатация [11]

Тестовая документация включает в себя:

1. тест план;
2. тестовая стратегия;
3. чек-лист;
4. тестовый сценарий;
5. тестовый комплект;
6. пользовательская история (User Story);
7. отчет о дефекте [12].

Тест план

Тест план — это артефакт тестирования, описывающий действия, которые будут происходить в процессе тестирования — от разработки стратегии до критериев поиска ошибок. Он также описывает логику завершения задач и оценку рисков со сценариями их разрешения

Будет протестирована следующая функциональность ИС:

• Модуль авторизации – приоритет высокий;

• Модуль оформления заказа – приоритет высокий;

• Модуль регистрации – приоритет высокий;

Нагрузочное тестирование и тестирование безопасности не будет проведено в виду отсутствия необходимых ресурсов.

Таблицу сделать!!!

Написание тест-кейсов

**Тест-кейс** — это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату.

Атрибуты тест-кейса

Любой тест-кейс обязательно включает в себя:

* Уникальный идентификатор тест-кейса — необходим для удобной организации хранения и навигации по нашим тест-наборам.
* Название — основная тема, или идея тест-кейса. Кратное описание его сути.
* Предусловия — описание условий, которые не имеют прямого отношения к проверяемому функционалу, но должны быть выполнены. Например, оставить комментарий на вашем портале может только зарегистрированный пользователь. Значит для тест-кейса «Создание комментария» будет необходимо выполнение предусловия «пользователь зарегистрирован», и «пользователь авторизован»
* Шаги — описание последовательности действий, которая должна привести нас к ожидаемому результату
* Ожидаемый результат — результат: что мы ожидаем увидеть после выполнения шагов [13] [14].

Таблица 2 Тест-кейсы

Тест-кейсы.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название | Модуль | Предусловия | Шаги | Ожидаемый результат |
| 1 | Регистрация нового пользователя | Регистрация |  | 1. Написать адрес електронной почты, пароль, имя, фамилия 2. Нажать копку «Зарегистрироваться» | 1. Новый пользователь будет зарегистрирован |
| 2 | Вход в аккаунт с не верными данными | Авторизация | Произвести попытку входа в аккаунт с неверным паролем | 1. Вести логин (электронную почту) 2. Ввести неверный пароль 3. Нажать кнопку «Войти» | Будет выдано предупреждение о том, что нужно зарегистрироваться |
| 3 | Вход в аккаунт с верными данными | Авторизация | Произвести попытку входа в аккаунт с валидными данными от аккаунта | 1. Вести логин (электронную почту) 2. Ввести верный пароль 3. Нажать кнопку «Войти» | 1. Успешный вход в аккаунт |
| 4 | Оформление заказа | Оформление заказа |  | 1. Войти в аккаунт 2. Нажать кнопку «Оформить заказ» 3. Заполнить все поля | 1. Заказ будет оформлен 2. Пользователя вернёт на начальную страницу, но уже с карточкой оформленного заказа |

Юнит-тестирование

Юнит-тестирование (Модульное тестирование) — это метод тестирования программного обеспечения, при котором создаются модули, то есть небольшие части приложения, поведение каждого из которых проверяется отдельно. Модульное тестирование выполняется на этапе разработки приложения. Модулем может быть что угодно, например, процедура или функция.

Модульное тестирование состоит из трех этапов.

1. Во-первых, инициализация небольшого фрагмента приложения, которое вы хотите протестировать.
2. Для этого вызывается метод, применяемый в качестве стимула к тестируемой системе.
3. Последний этап: наблюдение за поведением проверяемого модуля. Если наблюдаемое поведение соответствует ожиданиям пользователя, то модульный тест проходит. Этот пошаговый процесс также называется AAA (Arrange, Act, Assert).

Для данной ИС были составлен 1 юнит-тест на проверку входа пользователя с валидными логином и паролем, но тест не работает, потому что при выполнении его выдаётся ошибка:

The type 'System.Data.Entity.DbContext' is defined in an assembly that is not referenced. You must add a reference to assembly 'EntityFramework, Version=5.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'.

public void LoginingTest()

{

string password = "12343";

string email = "maxim2003@mail.ru";

bool expected =true;

bool actual = password.Equals(expected);

Assert.IsTrue(actual);

Entities1.GetContext().Usersses.Any(p => p.Login == email && p.Password == password);

ЗАКЛЮЧЕНИЕ

В итоге работы была разработана информационная система курьерской службы доставки

Выполнены следующие задачи:

1. Проведён анализ предметной области;
2. Составлено техническое задание;
3. Выделены основные требования к системе;
4. Спроектирована база данных;
5. Создано программное обеспечение (ПО) для данной системы;
6. Проведено тестирование системы.

Был получен информационный продукт, который соответствует требованиям заказчика.

В результате работы были закреплены знания в разработке приложений на языке программирования C# на основе WPF, языке запросов LINQ, работа с базой данных.

# **Список литературы**

|  |  |
| --- | --- |
| [1] | «Функциональные и нефункциональные требования: полное руководство,» [В Интернете]. Available: https://bestprogrammer.ru/izuchenie/funktsionalnye-i-nefunktsionalnye-trebovaniya-polnoe-rukovodstvo. |
| [2] | «ТЕХНИЧЕСКОЕ ЗАДАНИЕ,» [В Интернете]. Available: https://docs.cntd.ru/document/1200144624. |
| [3] | «Лекция 11: Документирование требований,» [В Интернете]. Available: https://intuit.ru/studies/curriculums/15720/courses/174/lecture/4732. |
| [4] | «База данных. Реляционная база данных,» [В Интернете]. Available: https://htmlacademy.ru/tutorial/php/databases. |
| [5] | «Что такое СУБД,» [В Интернете]. Available: https://www.nic.ru/help/chto-takoe-subd\_8580.html.. |
| [6] | «Этапы проектирования баз данных,» [В Интернете]. Available: https://studopedia.ru/2\_10121\_etapi-proektirovaniya-baz-dannih.html. |
| [7] | А. Н. Королёва, «Основные объекты СУБД: таблицы, формы, запросы, отчеты,» 1 06 2017. [В Интернете]. Available: https://spravochnick.ru/bazy\_dannyh/sistema\_upravleniya\_bazami\_dannyh\_-\_subd/osnovnye\_obekty\_subd\_tablicy\_formy\_zaprosy\_otchety/. |
| [8] | «Язык программирования C#: краткая история, возможности и перспективы,» Space Police, 29 03 2021. [В Интернете]. Available: https://timeweb.com/ru/community/articles/chto-takoe-csharp. |
| [9] | «Введение в WPF,» [В Интернете]. Available: https://metanit.com/sharp/wpf/1.php. |
| [10] | «XAML,» [В Интернете]. Available: https://metanit.com/sharp/wpf/2.php. |
| [11] | «Фундаментальная теория тестирования,» [В Интернете]. Available: https://habr.com/ru/post/549054/. |
| [12] | «Виды тестовой документации,» [В Интернете]. Available: https://bugza.info/vidy-testovoj-dokumentacii/.. |
| [13] | «Как писать тест-кейсы: полное руководство,» [В Интернете]. Available: https://testengineer.ru/kak-pisat-test-kejsy-polnoe-rukovodstvo/. |
| [14] | «Пишем максимально эффективный тест-кейс,» [В Интернете]. Available: https://habr.com/ru/post/246463/. |

**Список иллюстраций**

[Рисунок 1 Регистрация нового пользователя 14](#_Toc105399760)

[Рисунок 2 Результат. Новый пользователь успешно зарегистрирован 14](#_Toc105399761)

[Рисунок 3 Запрос. Создание заказа 15](#_Toc105399762)

[Рисунок 4 Результат. Заказа в списке заказов 15](#_Toc105399763)

[Рисунок 5 Запрос. Ввывод информации о заказах, сделанных пользователем 15](#_Toc105399764)

[Рисунок 6 Результат. Ввывод информации по заказам сделанным пользователем 15](#_Toc105399765)

[Рисунок 7Начальное окно 20](#_Toc105399766)

[Рисунок 8 Страница "Регистрация" 27](#_Toc105399767)

[Рисунок 9 Страница «Главная (без заказа)» 31](#_Toc105399768)

[Рисунок 10 Страница «Главная(с заказом)» 31](#_Toc105399769)

[Рисунок 11 Страница «Оформление заказа» 34](#_Toc105399770)

[Таблица 1 Краткие сведения об информационной системе 7](#_Toc105399771)

[Таблица 2 Тест-кейсы 37](#_Toc105399772)

ПРИЛОЖЕНИЯ

### Приложение №1

Инструкции для использования системы (руководство пользователя);

### Приложение №2

Техническое задание

### Приложение №3

ER-ДИАГРАММА

### Приложение №4

ЛИСТИНГ КОДА

private Userss \_currentUser = new Userss();

public Reg()

{

DataContext = \_currentUser;

InitializeComponent();

}

private void btnLog\_click(object sender, RoutedEventArgs e)

{

bool canRegister = Entities1.GetContext().Usersses.All(p => p.Login != LoginTb.Text.Trim());

if (canRegister)

{

Entities1.GetContext().Usersses.Add(\_currentUser);

Entities1.GetContext().SaveChanges();

MessageBox.Show("Вы прошли регистрацию");

Login login = new Login();

login.Show();

Close();

}

else

{

MessageBox.Show("Вы не прошли регистрацию, такой Email существует");

}

}

### Приложение №5

Ссылка на облачно хранилище и репозиторий с курсовой работой

Яндекс Диск: <https://disk.yandex.ru/d/KaHpUpoTs20EYw>

GitHub: <https://github.com/XoLoDiLniC/Solodyashkin-Andrey-Yurievich-3-1-2022/tree/main/%D0%9A%D1%83%D1%80%D1%81%D0%BE%D0%B2%D0%BE%D0%B9%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82>