### Частное учреждение образования

### «Колледж бизнеса и права»

### СОЗДАНИЕ WEB-ПРИЛОЖЕНИЯ ПО АВТОМАТИЗАЦИИ РАБОТЫ МОТОСАЛОНА «Kawasaki»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### к курсовому проектированию по дисциплине

### «Конструирование программ и языки программирования»

КП Т.196005.401 ПЗ

Руководитель проекта (А.В.Кривошеина)

Учащийся (И.М.Володьков )

Минск, 2024

**СОДЕРЖАНИЕ**

Изм.

Изм.

Лист

Лист

№ докум.

№ докум.

Подпись

Подпись

Дата

Дата

Лист

Лист

3

3

КП Т.196005.401 ПЗ

ПЗ

ДП Т.517074.401 ПЗ

Разраб.

Разраб.

ВолодьковИ.М..

Мовчан П.И.

Провер.

Провер.

Кривошеина А.В.

Багласова Е.В.

Т. контр.

Т. контр.

*.*

*Якимович К.О.*

Н. контр.

Н. контр.

Багласова Е.В.

Утверд.

Утверд.

*Багласова Т.Г.*

### *СОЗДАНИЕ WEB-ПРИЛОЖЕНИЯ ПО* *АВТОМАТИЗАЦИИ РАБОТЫ МОТОСАЛОНА «Kawasaki»*

*Программное средство для автоматизации учета заявок жителей Фрунзенского района г.Минска*

Лит.

Лит.

Листов

Листов

54

КБП

КБП

[Введение 3](#_Toc169941869)

[1. Описание задачи 4](#_Toc169941870)

[1.1. Анализ предметной области 4](#_Toc169941871)

[1.2. Постановка задачи 6](#_Toc169941872)

[2. Проектирование системы 7](#_Toc169941873)

[2.1. Требования к приложению 7](#_Toc169941874)

[2.2. Проектирование модели 7](#_Toc169941875)

[2.3. Организация данных 10](#_Toc169941876)

[2.4. Концептуальный прототип 12](#_Toc169941877)

[3. Описание реализации программного средства 18](#_Toc169941878)

[3.1. Инструменты разработки и применения технологии 18](#_Toc169941879)

[3.2. Порядок авторизации пользователей 18](#_Toc169941880)

[3.3. Организация данных 19](#_Toc169941881)

[3.4. Функции: логическая и физическая организация 22](#_Toc169941882)

[3.5. Входные и выходные данные 24](#_Toc169941883)

[3.6. Функциональное тестирование 25](#_Toc169941884)

[3.7. Описание справочной системы 30](#_Toc169941885)

[4. Применение 31](#_Toc169941886)

[4.1. Назначение программного средства 31](#_Toc169941887)

[4.2. Условия применения 31](#_Toc169941888)

[Заключение 33](#_Toc169941889)

[Список использованных источников 34](#_Toc169941890)

[Приложение А 35](#_Toc169941891)

[Приложение Б 41](#_Toc169941892)

[Приложение В 42](#_Toc169941893)

# **Введение**

В современном мире, где технологии проникают во все сферы жизни, отрасль продаж и обслуживания мототехники не является исключением. Автоматизация процессов становится все более актуальной и важной задачей, позволяющей минимизировать потери рабочего времени персонала, экономить на его численности и повышать уровень сервиса.

Целью курсового проекта «WEB-приложение для автоматизации работы мотосалона» является создание программного средства, которое позволит автоматизировать работу менеджера и сотрудников мотосалона, улучшая взаимодействие с клиентами и оптимизируя внутренние процессы.

Пояснительная записка к курсовому проекту содержит пять разделов:

Описание задачи: В этом разделе раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть решены. Описываются существующие аналоги.

Проектирование системы: В этом разделе перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются требования к приложению, строится концептуальный прототип, описывается логическая и физическая организация данных, проектируется справочная система.

Описание реализации программного средства: В этом разделе представлены общие сведения о программном средстве и его функциональном назначении, входные и выходные данные.

Применение предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы.

Заключение будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст программы.

Графическая часть содержит в себе все необходимые диаграммы для проектирования данного программного средства, такие как:

* диаграмма вариантов использования;
* диаграмма развертывания;
* диаграмма классов;
* структура сайта.

# **Описание задачи**

# **Анализ предметной области**

Мотосалоны, как и многие другие бизнесы, все больше полагаются на современные технологии для повышения эффективности своей работы. Автоматизация процессов в мотосалоне становится критически важной задачей, позволяющей минимизировать потери рабочего времени персонала, сократить численность штата и улучшить качество обслуживания клиентов.

Целью проекта «Веб-приложение для автоматизации работы мотосалона» является создание программного средства, которое позволит автоматизировать работу мотосалона, улучшая взаимодействие с клиентами и оптимизируя внутренние процессы.

Основные элементы автоматизации работы мотосалона:

-Прогнозирование спроса на различные модели мотоциклов и аксессуары.

-Контроль продаж мотоциклов и сопутствующих товаров.

-Ведение базы данных клиентов.

-Обработка заказов.

-Добавление товаров в каталог.

Автоматизация процессов с помощью веб-приложения позволяет значительно увеличить эффективность работы, минимизировать вероятность ошибок, связанных с человеческим фактором, и сократить время на выполнение рутинных задач. Компьютерная обработка информации обеспечивает быстрый доступ к актуальным данным, возможность их анализа и принятия обоснованных решений на основе полученной информации. Это особенно важно в сфере продаж и обслуживания мототехники, где требуется понимание того, какой товар выбрать.

Потенциальная аудитория

Потенциальной аудиторией данного приложения являются мотосалоны, и клиенты мотосалонов, которым необходима оперативная и точная информация о наличии мотоциклов и сопутствующих товаров к ним.

# **Постановка задачи**

Исходя из анализа предметной области, можно выделить следующие задачи, подлежащие автоматизации:

* автоматизировать добавление товаров;
* автоматизировать управление запасами;
* автоматизировать поиск товаров по различным категориям;
* автоматизировать добавление клиентов;
* автоматизировать генерацию документов;

В сравнении с другими web-приложениями для автоматизации работы мотосалона, такими как www.major-kawasaki.ru, motorrika.ru, auto.ru, и другими, web-приложение «Kawasaki» имеет следующие преимущества:

* гибкость и масштабируемость: Kawasaki спроектирован так, чтобы быть гибким и масштабируемым, позволяя легко добавлять новые функции и поддерживать большое количество пользователей;
* пользовательский макет: «Kawasaki» имеет интуитивно понятный и простой в использовании пользовательский макет, что облегчит работу с ним для всех пользователей, независимо от их уровня технической грамотности.

Таким образом, «Kawasaki» предлагает уникальный набор функций, которые помогут улучшить эффективность и производительность в области автоматизации работы мотосалона, делая его важным инструментом для любого бизнеса, который хочет оставаться конкурентоспособным в современном быстро меняющемся бизнес-окружении.

# **Проектирование системы**

# **Требования к приложению**

В приложении Kawasaki будет реализована система аутентификации для обеспечения безопасности данных пользователя. Это означает, что доступ к функционалу приложения будет возможен только после ввода уникальных учетных данных (имя пользователя и пароль). Это помогает предотвратить несанкционированный доступ и защитить конфиденциальность данных пользователя.

Макет приложения Kawasaki будет разработан таким образом, чтобы обеспечить простоту и удобство использования. Однако, некоторые функции могут быть доступны только авторизованным пользователям.

Макет приложения Kawasaki будет иметь четкую и последовательную структуру. Все элементы управления будут иметь стандартный размер, форму и расположение, чтобы облегчить навигацию пользователям. Шрифт будет выбран таким образом, чтобы обеспечить хорошую читаемость на всех устройствах. Кроме того, цветовая схема и стиль дизайна будут выбраны так, чтобы обеспечить приятное визуальное восприятие и удобство использования.

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор intel core i3 6006u;
* оперативная память 8 Гбайт и более;
* свободное место на диске 100 Мбайт;
* монитор;
* мышь, клавиатура;

# **Проектирование модели**

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями.

Наиболее распространенные средства моделирования являются: диаграмма вариантов использования, диаграмма классов, диаграмма развертывания, структура сайта.

Диаграмма вариантов использования – это тип диаграммы, используемой в области разработки программного обеспечения для визуализации системы с точки зрения её актёров и вариантов использования.

Основные элементы диаграммы вариантов использования:

* актеры: ­ актеры представляют собой людей или системы, которые взаимодействуют с системой. Они находятся вне системы и могут быть как людьми (например, пользователи или администраторы), так и другими системами или устройствами;
* варианты использования: варианты использования – это функции или действия, которые система может выполнять в ответ на запросы от актеров. Они представляют собой цели, которые ­ актеры хотят достичь с помощью системы;
* отношения: отношения между актёрами и вариантами использования показывают, как ­ актеры и варианты использования взаимодействуют друг с другом. Они могут быть обозначены стрелками или линиями.

Диаграмма вариантов использования помогает разработчикам лучше понять, как система будет работать и какие функции она будет выполнять. Она также помогает определить требования к системе и спланировать её разработку. Это важный инструмент в процессе проектирования системы.

Диаграмма вариантов использования представлена на листе 1 графической части.

На этой диаграмме можно выделить трёх пользователей программным продуктом. Клиент, который может обратиться к логисту с просьбой о перевозке товара. Логист, который добавляет клиента в систему и может его редактировать, а также создавать заявки на перевозку посылок. Сотрудник транспортного отдела, который создаёт новых водителе и транспорт.

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их атрибуты (полей), методы, Макеты и взаимосвязи (отношений) между ними. Она используется для визуализации структуры классов в системе и их взаимосвязей.

Вот основные элементы диаграммы классов:

* Классы: Классы представляют собой ключевые элементы в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента: имя класса, поля (атрибуты) класса и методы класса.
* Отношения: Отношения между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализация.

Диаграмма классов соответствует принципам объектно-ориентированного программирования (ООП) и является одним из базовых инструментов проектирования ООП-систем. Она помогает лучше понимать структуру системы и ее компоненты.

В данном приложении присутствуют следующие классы:

* пользователь;
* заказ;
* товар;
* тип товара;

Диаграмма классов представлена на листе 2 графической части.

Диаграмма развертывания – это графическое представление физических компонентов системы, их взаимосвязей и размещения в сети. Эта диаграмма используется для планирования и анализа инфраструктуры, а также для оптимизации развертывания приложений.

Основные элементы диаграммы развертывания:

-Узел (Node): Узел представляет собой физический элемент, такой как сервер или устройство, на котором размещаются компоненты системы.

-Компонент (Component): Компоненты представляют собой программные модули или приложения, размещенные на узлах.

-Взаимосвязь (Communication Path): Взаимосвязи показывают связи между узлами, такие как сети или коммуникационные каналы.

-Артефакт (Artifact): Артефакты представляют собой физические файлы или наборы данных, которые размещаются и обрабатываются на узлах.

Диаграмма развертывания позволяет визуализировать физическую архитектуру системы, что является важным для анализа, планирования и управления инфраструктурой. Она помогает идентифицировать возможные узкие места, планировать ресурсы и оптимизировать развертывание приложений.

Диаграмма развертывания представлена на листе 3 графической части.

Диаграмма структуры сайта – это графическое представление организации и взаимосвязей страниц и разделов веб-сайта. Эта диаграмма используется для планирования навигации, оптимизации пользовательского опыта и управления содержимым.

Основные элементы диаграммы структуры сайта:

-Начальная страница (Home Page): Начальная страница является основной точкой входа на сайт и отправной точкой для навигации.

-Разделы (Sections): Разделы представляют собой основные категории или подразделы сайта, которые группируют связанные страницы.

-Страницы (Pages): Страницы представляют собой отдельные веб-документы, содержащие контент, такой как текст, изображения, видео и другие элементы.

-Ссылки (Links): Ссылки показывают навигационные пути между страницами и разделами, обеспечивая переходы пользователей по сайту.

-Меню навигации (Navigation Menu): Меню навигации представляет собой структурированный список ссылок, помогающий пользователям ориентироваться на сайте.

-Диаграмма структуры сайта позволяет визуализировать организацию контента и навигацию, что является важным для проектирования, планирования и оптимизации веб-сайта. Она помогает улучшить пользовательский опыт, обеспечивая логическую и интуитивно понятную структуру сайта.

Диаграмма структуры сайта представлена на листе 4 графической части.

# **Организация данных**

Наиболее распространенным средством моделирования таких типов документации являются схемы-сети (ERD), которые применяются для графического представления многих элементов разработанных программ систем и представляют некоторые габаритные размеры обозначений для определения данных и отношений между ними. С помощью такой диаграммы можно описать отдельные компоненты концептуальной модели или выразить свою точку зрения взаимосвязей между ними, иногда какое значение или реализацию системы. Основными потенциалом данной нотации является понятная сущность в каждом случае каждый рассматриваемый объект может явиться экземпляром одного и только одного сущности, должен иметь уникальное имя или идентификатор, а также отличается от других экземпляров данной сущности.

Графическая модель данных строится таким образом, чтобы связь между отделениями сущности органа не только естественный характер соответствующего отношения, но и дополнительные аспекты обязательности связей.

Информационная модель представлена на диаграмме «Сущность-связь» на рисунке 2.2

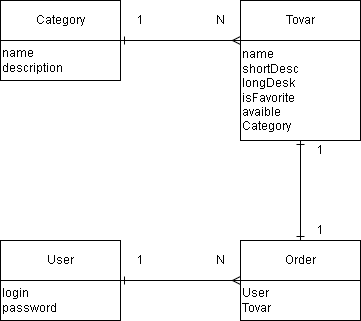


Рисунок 2.2 – Диаграмма «Сущность-связь»

Исследовав предметную область, можно выделить следующие сущности: «Пользователь», «Категория», «Товар», «Заказ».

Для сущности «Пользователь» атрибутами будут являться:

* логин
* пароль;

Для сущности «Товар» атрибутами будут являться:

* название;
* короткое описание;
* полное описание;
* является ли товар популярным;
* наличие;
* категория

Для сущности «Категория» атрибутами будут являться:

* название;
* описание;

Для сущности «Заказ» атрибутами будут являться:

* пользователь;
* товар;

Для сущности «Пользователь» атрибутами будут являться:

Типы связей между сущностями:

* пользователь-заказ: один ко многим;
* товар-заказ: один к одному;
* категория-товар: один многим;

# **Концептуальный прототип**

Концептуальный прототип состоит из описания внешнего вида пользовательского Макета, а именно, элементов управления.

Разрабатываемое приложение будет содержать шесть основных форм: «Главная», «Авторизация», «Регистрация», «Список товаров», «О нас» и «Корзина».

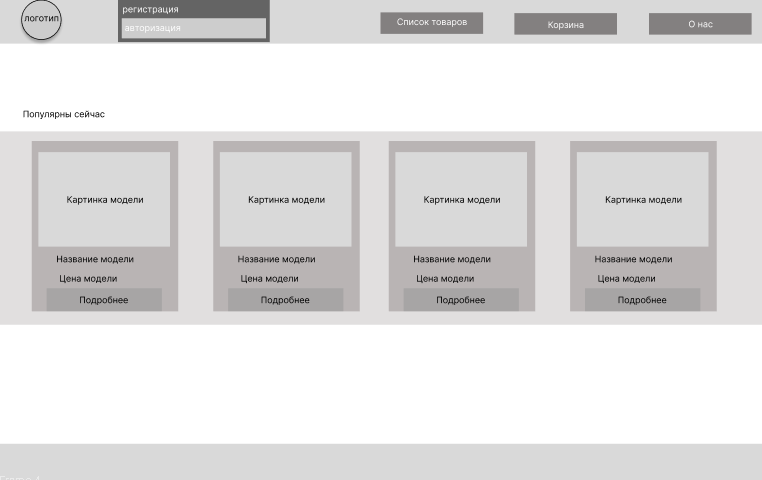
Главная страница:

На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

Дальше на странице располагается блок, в котором находятся другие блоки, в которых отображается такая информация о товаре как: картинка в тэге img, название, цена, кнопка «подробнее»

Внизу страницы отображается footer с контактной информацией

Макет главной страницы представлен на рисунке 2.3

Русинок 2.3 Макет главной страницы

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

-при нажатии на кнопку «Подробнее», вас переведет на страницу товара

Страница «каталог товаров»:

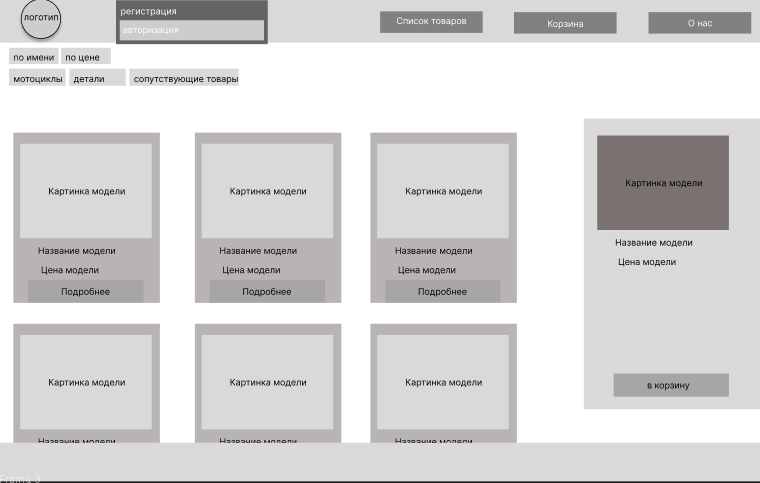
На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

В тэге body отображается : 5 кнопок, из них две для сортировки по названию и по цене, остальные три для сортировки по категориям товаров

Дальше на странице располагается блок, в котором находятся другие блоки, в которых отображается такая информация о товаре как: картинка в тэге img, название, цена, кнопка «подробнее»

Внизу страницы отображается footer с контактной информацией

Макет страницы с каталогом товаровпредставлен на рисунке 2.4

 Русинок 2.4 Макет страницы с каталогом товаров

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

-при нажатии на кнопку «по имени», осуществляется сортировка всех товаров в каталоге по имени

-при нажатии на кнопку «по цене», осуществляется сортировка всех товаров в каталоге по цене

-при нажатии на кнопку «мотоциклы», в каталоге будут отображаться только товары из категории «мотоциклы»

-при нажатии на кнопку «детали», в каталоге будут отображаться только товары из категории «детали»

-при нажатии на кнопку «сопутствующие товары», в каталоге будут отображаться только товары из категории «сопутствующие товары»

-при нажатии на кнопку «в корзину» выбранный товар помещается в корзину

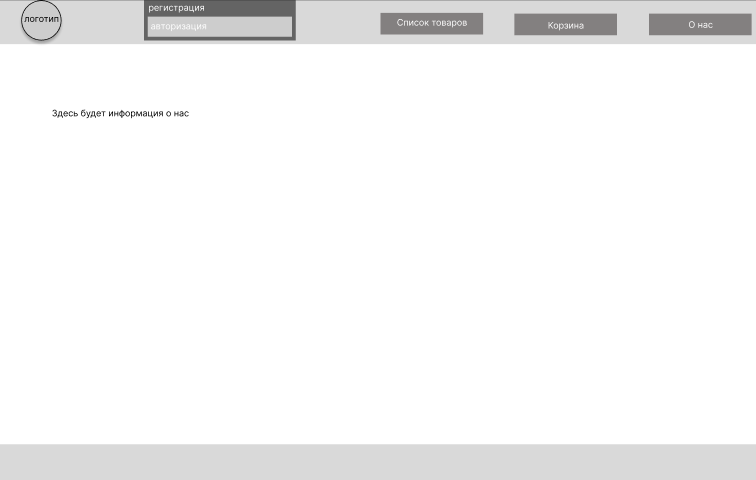
Страница «О нас»

На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

В тэге body отображается кртакая информация о нас

Внизу страницы отображается footer с контактной информацией

Макет страницы «О нас» представлен на рисунке 2.5

 Русинок 2.5 Макет страницы «О нас»

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

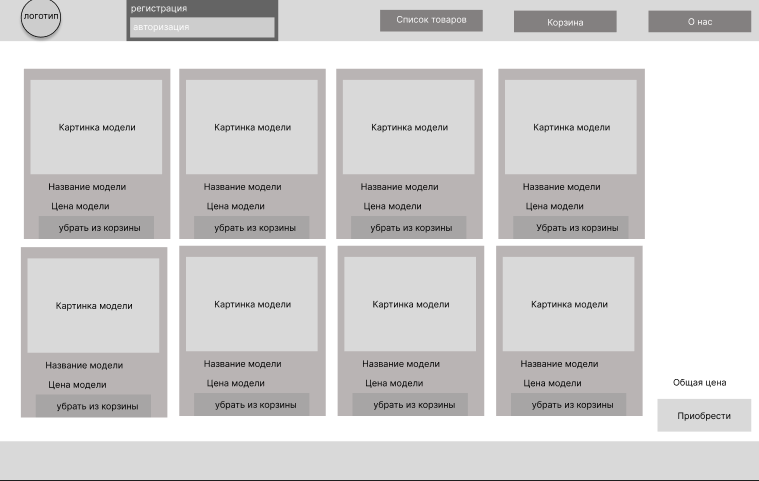
-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

Страница «Корзина»

На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

Дальше на странице находятся блоки, в которых отображается такая информация о товаре как: картинка в тэге img, название, цена, кнопка «убрать из корзины», справа от них отображаеся общая цена за все товары в корзине и кнопка приобрести

Макет страницы «Корзина» представлен на рисунке 2.6

 Русинок 2.6 Макет страницы «Корзина»

Внизу страницы отображается footer с контактной информацией

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

-при нажатии на кнопку «убрать из корзины» выбранный товар удаляется из корзины

-при нажатии на кнопку приобрести, заказ добавляется в базу данных, а пользователю отправляется электронная версия документа для покупки

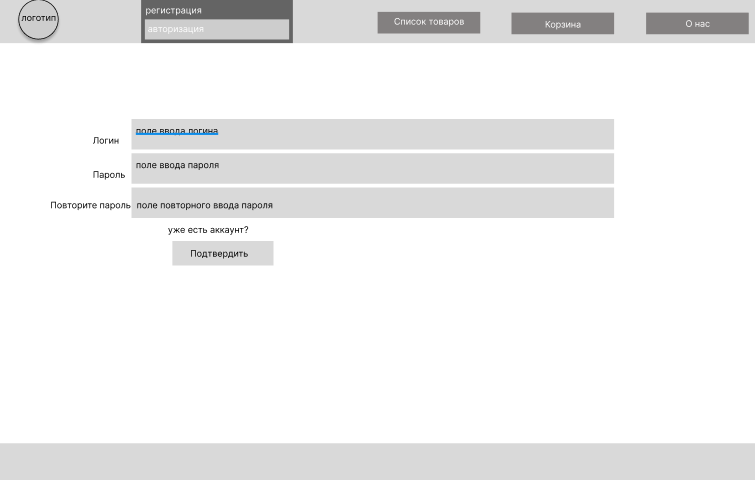
Страница регистрации

На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

Дальше на странице отображается 3 поля для ввода, слева от них указывается что в эти поля надо вводить, а так же две кнопки «уже есть аккаунт?» и «подтвердить»

Внизу страницы отображается footer с контактной информацией

Макет страницы «регистрация» представлен на рисунке 2.7

 Русинок 2.7 Макет страницы «Регистрация»

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

-при нажатии на кнопку «уже есть аккаунт?» осуществляется перевод на страницу авторизации

-при нажатии кнопки «подтвердить» данные которые были введены в поля проверяются на корректность, и если все правильно, то пользователь добавляется в базу данных, иначе всплывает окно с ошибкой

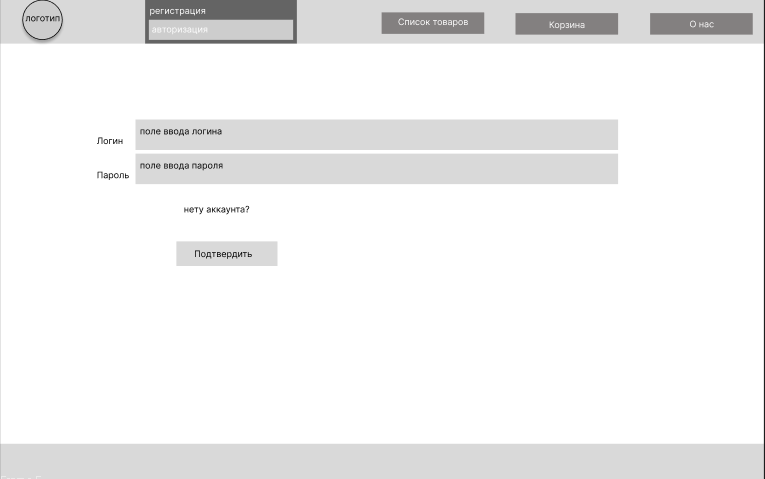
Страница авторизации

На ней располагается Header, в котором находится img с логотипоми и такие кнопки как «авторизация», «регистрация», «список товаров», «корзина», «о нас»

Дальше на странице отображается 2 поля для ввода, слева от них указывается что в эти поля надо вводить, а так же две кнопки «нету аккаунт?» и «подтвердить»

Внизу страницы отображается footer с контактной информацией

Макет страницы «Авторизация» представлен на рисунке 2.8

 Русинок 2.8 Макет страницы «Авторизация»

-при нажатии на логотип осуществляется перевод на главную страницу

-при нажатии на кнопку «регистрация», осуществляется перевод на страницу регистрации

-при нажатии кнопки «авторизация», осуществляется перевод на страницу авторизации

-при нажатии на кнопку «список товаров», осуществляется перевод в каталог товаров

-при нажатии кнопки «о нас», осуществляется перевод на страницу с информацией о нас

-при нажатии на кнопку «нету аккаунта?» осуществляется перевод на страницу авторизации

-при нажатии кнопки «подтвердить» данные которые были введены в поля проверяются на корректность, и если все правильно, то пользователь авторизуется на сайте, иначе всплывает окно с ошибкой

# **Описание реализации программного средства**

# **Инструменты разработки и применения технологии**

Инструментами разработки для будущего программного приложения будут являться:

* операционная система MS Widows 11;
* программная среда разработки Microsoft Visual Studio 2022;
* язык программирования C#;
* система управления реляционными базами данных (СУБД) Microsoft SQL Server 2024;
* программа для построения блок-схем Microsoft Visio
* язык определения данных DDL;
* Framework Razor Pages.

MS Windows 11 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства MS Windows. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами.

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского Макета, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию.

MS Visual Studio 2022 включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня.

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

DDL состоит из набора SQL-команд, используемых для создания, изменения и удаления структур баз данных, но не данных. Он просто имеет дело с описаниями схемы базы данных и используется для создания и изменения структуры объектов, присутствующих в базе данных. Все DDL-команды автоматически фиксируются, что означает, что все изменения навсегда сохраняются в базе данных. DDL подразделяется на пять команд, которые широко используются в SQL-запросах.

Microsoft Visio – векторный графический редактор, редактор диаграмм и блок-схем для Windows. Выпускается в трёх редакциях: Standard, Professional и Pro for Office 365.Аналогично с Adobe Reader, в стандартный набор программ Microsoft Office входит только средство для просмотра и печати диаграмм Microsoft Visio Viewer. Полнофункциональная версия Microsoft Visio Professional для создания и редактирования монограмм и диаграмм. Первоначально Visio разрабатывался и выпускался компанией Shapeware, затем переименованной в Visio Corporation. Microsoft приобрела компанию в 2000 году, тогда продукт назывался Visio 2000.

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Razor Pages - это фреймворк, входящий в состав ASP.NET Core, предназначенный для упрощения создания веб-приложений, которые преимущественно ориентированы на страницы. В отличие от традиционного шаблона MVC (Model-View-Controller), Razor Pages позволяет сосредоточиться на отдельных страницах, каждая из которых объединяет свою логику и представление.

# **Порядок авторизации пользователей**

В приложении можно выделить следующие функции:

Сотрудник транспортного отдела:

* Аутентификация: пользователь регистрируется в системе, предоставляя необходимые учетные данные (логин и пароль);
* Авторизация: после успешной аутентификации пользователь получает доступ к некоторым функциям;
* Функции: создание заказа.

# **Организация данных**

Организация данных подразумевает создание модели данных, главными элементами, которые являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает четыре таблицы.

Таблица «Category» содержит информацию категориях товаров, структура таблицы представлена в таблице 3.1

Таблица 3.1 – Category

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| Id | int | 4 | Номер категории |
| categoryName | nvarchar(MAX) | 1000 | Название категории |
| desk | nvarchar(MAX) | 1000 | описание |

Таблица «Tovar» содержит информацию о товарах организации, структура таблицы представлена в таблице 3.2

Таблица 3.2 – Tovar

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| Id | Int | 4 | Номер товара |
| Name | nvarchar(MAX) | 1000 | Названиетовара |
| shortDesc | nvarchar(MAX) | 1000 | Короткое описание товара |
| longDesc | nvarchar(MAX) | 1000 | Длинное описание товара |
| Price | Int | 4 | Цена товара |
| isFavorite | Bit | 1 | Есть ли в списке популярных товаров |
| Avaible | Bit | 1 | Есть ли в наличии |
| caegoryId | Int | 4 | Номер категории |

Таблица «User» содержит информацию о пользователях, структура таблицы представлена в таблице 3.3

Таблица 3.3 – User

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| Id | int | 4 | Номер пользователя |
| login | nvarchar(MAX) | 1000 | логин пользователя |
| password | nvarchar(MAX) | 1000 | пароль |

Таблица «Order» содержит информацию о заказах, структура таблицы представлена в таблице 3.4

Таблица 3.4 – Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| Id | int | 4 | Номер заказа |
| userId | Int | 4 | Номер пользователя |
| tovarId | Int | 4 | Номер Товара |

Все таблицы представлены на рисунке 3.1

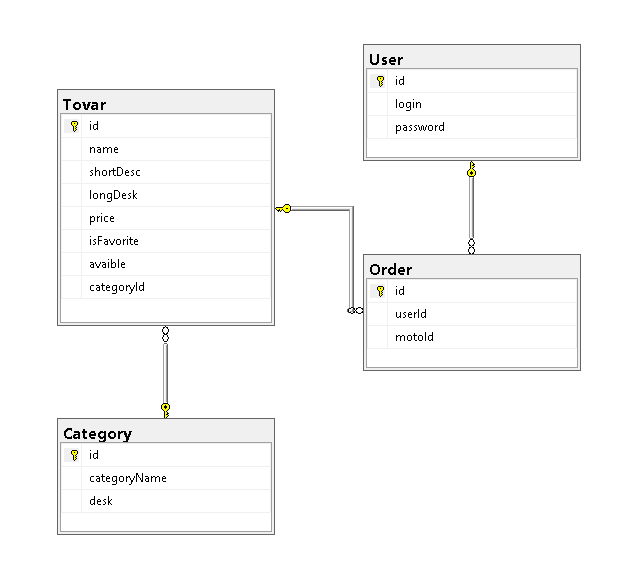


Рисунок 3.1 – Схема данных

# **Функции: логическая и физическая организация**

В данном проекте реализованы функции добавления, редактирования и удаления информации о пользователях, заказах и товарах; формирование договора на покупку и информация об посылках заказчика; формирование поиска.

Функция добавления информации о пользователях находиться на форме «Регистрация». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события OnPost. Все данные, вводимые в поля формы, будут сохранены в таблице «User». Код функции добавления пользователя:

public IActionResult OnPost()

{

if (!ModelState.IsValid)

{

return Page();

}

if (User.password != ConfirmPassword)

{

ModelState.AddModelError("ConfirmPassword", "Passwords do not match.");

return Page();

}

\_context.User.Add(User);

\_context.SaveChanges();

return RedirectToPage("/Login");

}

Функция добавления информации о товарах находиться на форме «Добавление товара». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Tovar». Код функции добавления товара:

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

{

try

{

string name = NameTextBox.Text;

string shortDesc = ShortDescTextBox.Text;

string longDesk = LongDescTextBox.Text;

string img = name + ".jpg";

int price = int.Parse(PriceTextBox.Text);

bool isFavorite = IsFavoriteCheckBox.IsChecked.Value;

bool avaible = AvaibleCheckBox.IsChecked.Value;

int categoryId = (int)comboBox1.SelectedValue;

string sql = "INSERT INTO Moto (name, shortDesc, longDesk, img, price, isFavorite, avaible, categoryId) VALUES (@name, @shortDesc, @longDesk, @img, @price, @isFavorite, @avaible, @categoryId)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sql, connection))

{

command.Parameters.AddWithValue("@name", name);

command.Parameters.AddWithValue("@shortDesc", shortDesc);

command.Parameters.AddWithValue("@longDesk", longDesk);

command.Parameters.AddWithValue("@img", img);

command.Parameters.AddWithValue("@price", price);

command.Parameters.AddWithValue("@isFavorite", isFavorite);

command.Parameters.AddWithValue("@avaible", avaible);

command.Parameters.AddWithValue("@categoryId", categoryId);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Данные успешно добавлены");

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

Функция формирования созданного договора происходит при нажатии на кнопку «Заказать» на странице «Корзина». Происходит обработка события CreateOrderFile, в процессе которого будет создан документ pdf со всеми данными в договоре. Код функции создания документа pdf:

public IActionResult CreateOrderFile()

{

var model = GetStashViewModel();

if (model.CurUser == null || model.Tovars == null)

{

return BadRequest("Invalid user or products");

}

var fullString = new StringBuilder();

foreach (var moto in model.Tovars)

{

fullString.AppendLine($"{moto.name}: {moto.price} руб.");

var order = new Order

{

userId = model.CurUser.id,

motoId = moto.id

};

\_context.Order.Add(order);

}

\_context.SaveChanges();

var fileName = "Order.pdf";

var fileDirectory = Path.Combine(\_hostingEnvironment.WebRootPath, "App\_Data");

var filePath = Path.Combine(fileDirectory, fileName);

if (!Directory.Exists(fileDirectory))

{

Directory.CreateDirectory(fileDirectory);

}

var fontPath = Path.Combine(\_hostingEnvironment.WebRootPath, "fonts", "arial.ttf");

try

{

using (var writer = new PdfWriter(filePath))

{

using (var pdf = new PdfDocument(writer))

{

var document = new Document(pdf);

var font = PdfFontFactory.CreateFont(fontPath, PdfEncodings.IDENTITY\_H);

document.Add(new Paragraph("Заказ")

.SetFont(font)

.SetFontSize(20)

.SetTextAlignment(TextAlignment.CENTER));

document.Add(new Paragraph($"Заказчик: {model.CurUser.login}")

.SetFont(font)

.SetFontSize(12)

.SetMarginTop(20));

document.Add(new Paragraph("Заказывает:")

.SetFont(font)

.SetFontSize(12)

.SetMarginTop(10));

foreach (var moto in model.Tovars)

{

document.Add(new Paragraph($"{moto.name}: {moto.price} руб.")

.SetFont(font)

.SetFontSize(12));

}

}

}

var fileBytes = System.IO.File.ReadAllBytes(filePath);

return File(fileBytes, "application/pdf", fileName);

}

catch (Exception ex)

{

// Handle exceptions (logging, rethrowing, etc.)

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}Полный код программных модулей представлен в приложении А.

# **Входные и выходные данные**

Входными данными являются данные, заносимые программой в таблицы: «Tovar», «User», «Order».

Для добавления данных в таблицу «Tovar» необходимо ввести следующие данные:

­ название;

­ короткое описание;

­ полное описание;

­ является ли товар популярным;

­ наличие;

­ категория

Для добавления данных в таблицу «User» необходимо ввести следующие данные:

* логин
* пароль;

Для добавления данных в таблицу «Order» необходимо ввести следующие данные:

* пользователь;
* товар;

Выходные данными являются данные, которые экспортируются в файл формата txt, а именно договор покупки. А также отчёт:

Договор генерируется по нажатию на кнопку «Заказать». Выходные документы представлен в приложении В.

# **Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствия требованиям. Оценка производится в соответствии с ожидаемым и фактическим результатом.

Тестирование программы будет производится последовательно, переходя из одной программы в другую. Во время теста будут проверятся все действия с программой, навигация пунктов меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

В таблице 3.7 представлен набор тест-кейсов, подготовленных для проведения функционального тестирования.

Таблица 3.7 Набор тест-кейсов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название теста | Peconditions | Шаг выполнения | Ожидаемый результат | Фактический результат |
| 1 | Переход на страницу со списком товаров |  | 1.зайти на сайт «Kawasaki»  2.Нажать кнопку «Список товаров» | 1.Сайт корректно отображается в браузере  2.Переход на страницу «Список товаров» | 1.Сайт корректно отображается в браузере  2.Переход на страницу «Список товаров» |
| 2 | Тестирование кнопки сортировка по названию | 1.зайти на сайт «Kawasaki» | 1. Нажать кнопку «Список товаров»  2. Нажать на кнопку «По названию» | 1.Переход на страницу «Список товаров»  2.Отображаемые товары стали отображаться в отсортированном по названию виде | 1.Переход на страницу «Список товаров»  2.Отображаемые товары стали отображаться в отсортированном по названию виде |
| 3 | Тестирование кнопки сортировки по цене | 1.зайти на сайт «Kawasaki» | 1. Нажать на кнопку «Список товаров»  2.Нажать на кнопку «По цене» | 1.Переход на страницу «Список товаров»  2.Отображаемые товары стали отображаться в отсортированном по цене виде | 1.Переход на страницу «Список товаров»  2.Отображаемые товары стали отображаться в отсортированном по цене виде |
| 4 | Тестирование кнопки сортировки по категории «Мотоцикл» | 1.зайти на сайт «Kawasaki» | 1. Нажать на кнопку «Список товаров»  2.Нажать на кнопку «Мотоциклы» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Мотоцикл» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Мотоцикл» |
| 5 | Тестирование кнопки сортировки по категории «Деталь» | 1.зайти на сайт «Kawasaki» | 1. Нажать на кнопку «Список товаров»  2.Нажать на кнопку «Детали» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Деталь» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Деталь» |
| 6 | Тестирование кнопки сортировки по категории «Сопутствующий товар» | 1.зайти на сайт «Kawasaki» | 1. Нажать на кнопку «Список товаров»  2.Нажать на кнопку «Сопутствующие товары» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Сопутствующий товар» | 1.Переход на страницу «Список товаров»  2.Стали отображаться товары только с категорией «Сопутствующий товар» |
| 7 | Тестирование добавления товара в корзину | 1.Зайти на сайт «Kawasaki» | 1.Нажать кнопку «Список товаров»  2.Нажать кнопку «Подробнее» под товаров Kawasaki Ninja z900  3.Нажать кнопку «Добавить в корзину» | 1.Переход на страницу «Список товаров»  2.Информация о товаре отображается в правой части страницы  3.Товар успешно добавлен в корзину | 1.Переход на страницу «Список товаров»  2.Информация о товаре отображается в правой части страницы  3.Товар успешно добавлен в корзину |
| 8 | Тестирование добавления нескольких товаров корзину и отображение их в корзине | 1.Зайти на сайт «Kawasaki» | 1.Нажать кнопку «Список товаров»  2.Нажать кнопку «Подробнее» под товаров Kawasaki Ninja z900  3.Нажать кнопку «Добавить в корзину»  4.Нажать кнопку «подробнее» под товаром KAWASAKI Z900  5. Нажать кнопку «Добавить в корзину»  6.Нажать кнопку «Корзина» | 1.Переход на страницу «Список товаров»  2.Информация о товаре отображается в правой части страницы  3.Товар успешно добавлен в корзину  4. Информация о товаре отображается в правой части страницы  5.Товар успешно добавлен в корзину  6.Открылась страница корзины и на ней отображается 2 ранее выбранных товара | 1.Переход на страницу «Список товаров»  2.Информация о товаре отображается в правой части страницы  3.Товар успешно добавлен в корзину  4. Информация о товаре отображается в правой части страницы  5.Товар успешно добавлен в корзину  6.Открылась страница корзины и на ней отображается 2 ранее выбранных товара |
| 9 | Тестирование приобретения товара | 1.Зайти на сайт «Kawasaki»  2.Добавить товар в корзину | 1.Нажать кнопку «Корзина»  2.Нажать «Приобрести» | 1.Переход на страницу корзины  2.Загрузка договора на приобретение каждого товара из корзины | 1.Переход на страницу корзины  2.Загрузка договора на приобретение каждого товара из корзины |
| 10 | Авторизация | 1.Зайти на сайт «Kawasaki» | 1.Нажать кнопку «Войти»  2.Нажать Кнопку «Уже есть аккаунт»  3.Ввести данные:  Логин: «Login»  Пароль: «Pass»  4. Нажать кнопку «Далее» | 1.Открытие страницы регистрации  2.Открылась страница авторизации  3.Данные корректно отображаются в строках для ввода  4.Успешно авторизованы | 1.Открытие страницы регистрации  2.Открылась страница авторизации  3.Данные корректно отображаются в строках для ввода  4.Успешно авторизованы |
| 11 | Регистрация | 1.Зайти на сайт «Kawasaki» | 1.Нажать кнопку «Войти»  2.Ввести данные:  Логин: «Login»  Пароль: «Pass»  Повторить пароль: «Pass»  3. Нажать кнопку «Далее» | 1.Открытие страницы регистрации  2.Данные корректно отображаются в строках для ввода  3.Успешно зарегистрированы | 1.Открытие страницы регистрации  2.Данные корректно отображаются в строках для ввода  3.Успешно зарегистрированы |

# **Описание справочной системы**

Справочная система была разработана с использованием программного обеспечения Dr.Explain.

Справочная система включает в себя несколько ключевых разделов, которые были определены в предоставленном изображении:

- Общие элементы управления

- Главная страница

- Страница «Список товаров»

- Страница «Корзина»

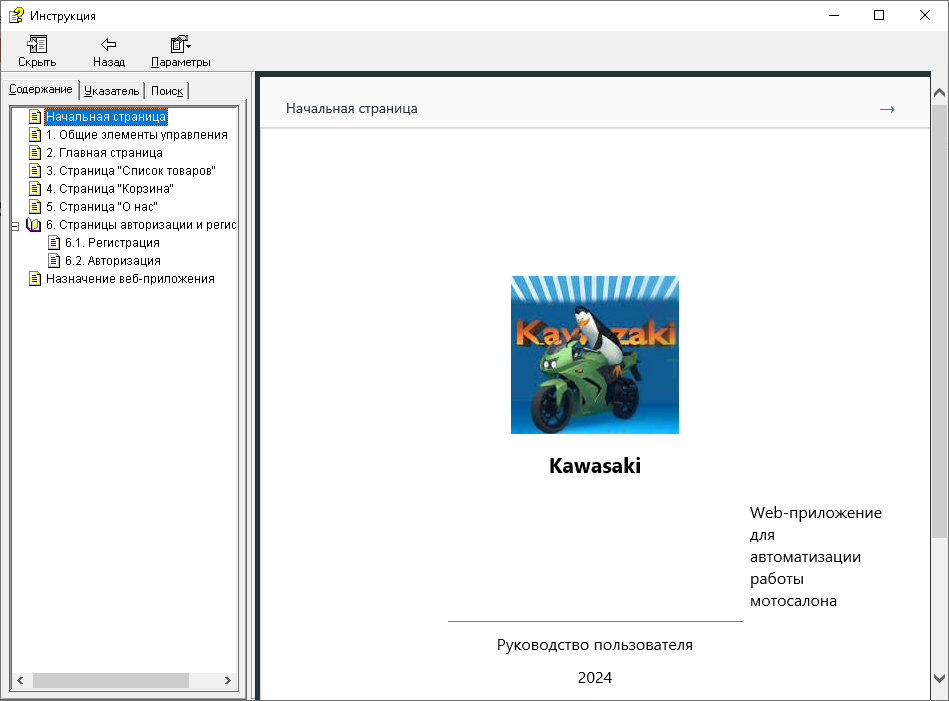
- Страница «О нас»

- Страницы авторизации и регистрации

- Назначение веб-приложения

Справочная система может быть вызвана из главной формы при нажатии на кнопку «Справка». Справка представлена на рисунке 3.3

Рисунок 3.3 – Справка



# **Применение**

# **Назначение программного средства**

Приложение «Kawasaki» предназначено для автоматизации работы мотосалона. Программное средство представляет пользователю возможности для добавления и отображения имеющихся в базе объектов: товаров, пользователей, категорий товаров, заказов, уменьшает затраты времени используемого при составлении договора на покупку мотоцикла, ведет учет посылок клиентов. Под эту роль был создан специальный макет, который позволит логисту быстро и удобно заключать договоры.

Основные разработанные функции:

* Автоматизация добавления товаров;
* Автоматизация управления запасами;
* Автоматизация поиска товаров по различным категориям;
* Автоматизация добавления клиентов;
* Автоматизация генерации документов;

Приложение «Kawasaki» разработано специально для мотосалонов операций и не предназначено для использования в других отраслях. Оно оптимизировано для работы с данными, связанными с продажей мотоциклов, и может не подходить для управления данными, не относящимися к мотоциклам и сопутствующим им товаров.

# **Условия применения**

Для применения данного программного средства необходимы следующие технические требования:

* процессор intel core i3 6006u;
* оперативная память 8 Гбайт и более;
* свободное место на диске 100 Мбайт;
* монитор;
* мышь, клавиатура;
* платформа .Net Framework v4.8;

# **Заключение**

Проект “Kawasaki” представляет собой программное средство, разработанное для автоматизации работы мотосалона. Организационная сущность задачи заключается в создании инструмента, который облегчит работу продавцов и консультантов мотосалонов.

Технические требования для программы включают процессор Intel Core i3 6006u или выше, минимальный объем оперативной памяти 8 Гбайт, операционную систему Windows 10, платформу .Net Framework v4.8. Логическая организация данных включает описание структуры данных и связей между объектами (товар, категория, заказ, пользователь). Физическая организация данных включает выбор СУБД и оптимизацию запросов. Справочная система описывает функциональность и предоставляет помощь пользователям.

Программное средство Kawasaki предоставляет возможности добавления и отображения объектов в базе данных. Входные данные включают информацию о товарах, категориях, пользователях. Выходные данные могут быть документами.

Среди преимуществ разработанного ПС можно выделить также интуитивный пользовательский Макет.

Kawasaki представляет собой важный инструмент для работы мотосалонов, помогая им оставаться конкурентоспособными в современном бизнес-окружении.

# **Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение технологий» / С.В. Банцевич, Т.Г. Багласова. – Минск: КБП, 2022. – 49 c.
2. Михалевич, В.Ю. Методические указания к курсовому проектированию для обучающихся специальности 2-40 01 01 «Программное обеспечение информационных технологий – Минск: КБП, 2023. – 3 c.
3. ГОСТ Р 54694-2011. Логистика. Термины и определения. – Введ. 2013-01-01. – М.: Стандартинформ, 2012.;
4. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
5. 9 лучших программ для автоматизации рабочих процессов [2024 г.] [Электронный ресурс] – Режим доступа: https://nauchniestati.ru/spravka;
6. Автоматизация рабочего места – RPA ROBIN [Электронный ресурс] – Режим доступа: https://www.atlassian.com/ru/agile;
7. Автоматизированное рабочее место (АРМ): что это? [Электронный ресурс] – Режим доступа: https://www.liga-commersantov.ru/articles;
8. Обязанности менеджера по работе с клиентами – чем занимается и как [Электронный ресурс] – Режим доступа: https://sales-generator.ru/blog;
9. Разработка автоматизированного рабочего места менеджера [Электронный ресурс] – Режим доступа: https://spravochnick.ru/informatika;
10. Топ лучших приложений для управления персоналом для автоматизации [Электронный ресурс] – Режим доступа: https://www.guru99.com/ru/best;
11. ТОП-26 программ мониторинга и контроля работы сотрудников за [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles;
12. Удобство и эффективность: разработка автоматизированного рабочего места [Электронный ресурс] – Режим доступа: <https://appmaster.io/ru/blog>;
13. Что значит АРМ (Автоматизированное рабочее место)? [Электронный ресурс] – Режим доступа: https://alexrovich.ru/info;
14. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
15. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
16. C Sharp – Википедия [2024 г.] [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp;
17. [Microsoft Office – Wikipedia](https://en.wikipedia.org/wiki/Microsoft_Office) [2024 г.] [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Microsoft\_Office;
18. [Microsoft Visio – Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visio) 2024 г.] [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visio>;
19. Windows 11 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа: https://support.microsoft.com/ru-ru/meetwindows11. – Дата доступа: 16.03.2024;

# **Приложение А**

**(обязательное)**

**Текст программных модулей**

MotosController.cs

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Data;

using WebApplication1.Data.interfaces;

using WebApplication1.Data.mocks;

using WebApplication1.Data.Models;

using WebApplication1.Pages;

using WebApplication1.ViewModels;

namespace WebApplication1.Controllers

{

public class MotosController : Controller

{

private readonly IMockMotos \_allMotos;

private readonly IMotosCategory \_allCategories;

public MotosController(IMockMotos iAllMotos, IMotosCategory iMotosCat)

{

\_allMotos = iAllMotos;

\_allCategories = iMotosCat;

}

public Tovar GetMotoById(int id)

{

Tovar moto =DBObjects.Motos.FirstOrDefault(m => m.id == id);

return moto;

}

public ViewResult Motorcycle()

{

MotoListViewModel obj= new MotoListViewModel();

obj.AllMoto = \_allMotos.Motos;

obj.currCategory = "Моторкукле";

return View(obj);

}

}

}

StashController.cs

using System;

using System.IO;

using System.Text;

using iText.IO.Font.Constants;

using iText.Kernel.Font;

using iText.Kernel.Pdf;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Data;

using WebApplication1.Data.Models;

using WebApplication1.Services;

using iText.Kernel.Pdf;

using iText.Layout;

using iText.Layout.Element;

using iText.Layout.Properties;

using iText.IO.Font.Constants;

using iText.Kernel.Font;

using System.IO;

using iText.IO.Font;

namespace WebApplication1.Controllers

{

public class StashController : Controller

{

private readonly GlobalStashService \_globalStashService;

private readonly GlobalUserService \_globalUserService;

private readonly IWebHostEnvironment \_hostingEnvironment;

private readonly AppDBContent \_context;

public StashController(GlobalStashService globalStashService, GlobalUserService globalUserService, IWebHostEnvironment hostingEnvironment, AppDBContent context)

{

\_globalStashService = globalStashService;

\_globalUserService = globalUserService;

\_hostingEnvironment = hostingEnvironment;

\_context = context;

}

public IActionResult Index()

{

var stash = \_globalStashService.Stash;

var curUser = \_globalUserService.CurUser;

var viewModel = new StashViewModel

{

Tovars = stash,

CurUser = curUser

};

return View(viewModel);

}

[HttpPost]

public IActionResult CreateOrderFile()

{

var model = GetStashViewModel();

if (model.CurUser == null || model.Tovars == null)

{

return BadRequest("Invalid user or products");

}

var fullString = new StringBuilder();

foreach (var moto in model.Tovars)

{

fullString.AppendLine($"{moto.name}: {moto.price} руб.");

var order = new Order

{

userId = model.CurUser.id,

motoId = moto.id

};

\_context.Order.Add(order);

}

\_context.SaveChanges();

var fileName = "Order.pdf";

var fileDirectory = Path.Combine(\_hostingEnvironment.WebRootPath, "App\_Data");

var filePath = Path.Combine(fileDirectory, fileName);

if (!Directory.Exists(fileDirectory))

{

Directory.CreateDirectory(fileDirectory);

}

var fontPath = Path.Combine(\_hostingEnvironment.WebRootPath, "fonts", "arial.ttf");

try

{

using (var writer = new PdfWriter(filePath))

{

using (var pdf = new PdfDocument(writer))

{

var document = new Document(pdf);

var font = PdfFontFactory.CreateFont(fontPath, PdfEncodings.IDENTITY\_H);

document.Add(new Paragraph("Заказ")

.SetFont(font)

.SetFontSize(20)

.SetTextAlignment(TextAlignment.CENTER));

document.Add(new Paragraph($"Заказчик: {model.CurUser.login}")

.SetFont(font)

.SetFontSize(12)

.SetMarginTop(20));

document.Add(new Paragraph("Заказывает:")

.SetFont(font)

.SetFontSize(12)

.SetMarginTop(10));

foreach (var moto in model.Tovars)

{

document.Add(new Paragraph($"{moto.name}: {moto.price} руб.")

.SetFont(font)

.SetFontSize(12));

}

}

}

var fileBytes = System.IO.File.ReadAllBytes(filePath);

return File(fileBytes, "application/pdf", fileName);

}

catch (Exception ex)

{

// Handle exceptions (logging, rethrowing, etc.)

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

private StashViewModel GetStashViewModel()

{

return new StashViewModel

{

CurUser = \_globalUserService.CurUser,

Tovars = \_globalStashService.Stash

};

}

}

}

UserController.cs

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Services;

namespace WebApplication1.Controllers

{

public class UserController : Controller

{

public IActionResult Index()

{

return View();

}

private readonly GlobalUserService \_globalUserService;

public UserController(GlobalUserService globaluserService)

{

\_globalUserService = globaluserService;

}

}

}

IAllMotos.cs

using WebApplication1.Data.Models;

namespace WebApplication1.Data.interfaces

{

public interface IAllMotos

{

IEnumerable<Tovar> Motos { get; }

IEnumerable<Tovar> GetFavMotorcycles { get; }

Tovar GetMotorcycle(int motoId);

}

}

IMockMotos.cs

using WebApplication1.Data.Models;

namespace WebApplication1.Data.interfaces

{

public class IMockMotos

{

public IEnumerable<Tovar> Motos { get; }

}

}

IMotosCategory.cs

using WebApplication1.Data.Models;

namespace WebApplication1.Data.interfaces

{

public interface IMotosCategory

{

IEnumerable<Category> AllCategories { get; }

}

}

MockCategory.cs

using WebApplication1.Data.interfaces;

using WebApplication1.Data.Models;

namespace WebApplication1.Data.mocks

{

public class MockCategory : IMotosCategory

{

public IEnumerable<Category> AllCategories

{

get; set;

}

}

}

MockMotos.cs

using System.Collections.Generic;

using WebApplication1.Data.interfaces;

using WebApplication1.Data.Models;

namespace WebApplication1.Data.mocks

{

public class MockMotos : IMockMotos

{

private readonly IMotosCategory \_CategoryMoto = new MockCategory();

public IEnumerable<Tovar> Motos {

get;

}

public IEnumerable<Tovar> GetFavMotorcycles { get ; set; }

public Tovar GetMotorcycle(int motoId)

{

throw new NotImplementedException();

}

}

}

Category.cs

namespace WebApplication1.Data.Models

{

public class Category

{

public int id { get; set; }

public string categoryName { get; set; }

public string desk { get; set; }

}

}

Order.cs

namespace WebApplication1.Data.Models

{

public class Order

{

public int id { get; set; }

public int userId { get; set; }

public int motoId { get; set; }

public User User { get; set; }

public Tovar Moto { get; set; }

}

}

Tovar.cs

namespace WebApplication1.Data.Models

{

public class Tovar

{

public int id { get; set; }

public string name { get; set; }

public string shortDesc { get; set; }

public string longDesk { get; set; }

public ushort price { get; set; }

public bool isFavorite { get; set; }

public bool avaible { get; set; }

public int categoryId { get; set; }

public Category Category { get; set; }

}

}

User.cs

namespace WebApplication1.Data.Models

{

public class User

{

public int id { get; set; }

public string login { get; set; }

public string password { get; set; }

}

}

AllMotosRepository.cs

using Microsoft.EntityFrameworkCore;

using WebApplication1.Data.interfaces;

using WebApplication1.Data.Models;

namespace WebApplication1.Data.Repository

{

public class AllMotosRepository : IAllMotos

{

private readonly AppDBContent \_appDBContent;

public AllMotosRepository(AppDBContent appDBContent)

{

\_appDBContent = appDBContent;

}

public IEnumerable<Tovar> Motos => \_appDBContent.Tovar.Include(c=> c.Category);

public IEnumerable<Tovar> GetFavMotorcycles => \_appDBContent.Tovar.Where(p => p.isFavorite).Include(c => c.Category);

public Tovar GetMotorcycle(int motoId) => \_appDBContent.Tovar.FirstOrDefault(p => p.id == motoId);

}

}

CatrgoryRepository.cs

using WebApplication1.Data.interfaces;

using WebApplication1.Data.Models;

namespace WebApplication1.Data.Repository

{

public class CategoryRepository : IMotosCategory

{

private readonly AppDBContent appDBContent;

public CategoryRepository(AppDBContent appDBContent) { this.appDBContent = appDBContent; }

public IEnumerable<Category> AllCategories =>appDBContent.Category;

}

}

AppDBContent.cs

using Microsoft.EntityFrameworkCore;

using WebApplication1.Data.Models;

namespace WebApplication1.Data

{

public class AppDBContent:DbContext

{

public DbSet<Tovar> Tovar { get; set; }

public DbSet<Category> Category { get; set; }

public DbSet<User> User { get; set; }

public DbSet<Order> Order { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Order>()

.HasOne(o => o.User)

.WithMany()

.HasForeignKey(o => o.userId);

modelBuilder.Entity<Order>()

.HasOne(o => o.Moto)

.WithMany()

.HasForeignKey(o => o.motoId);

}

public AppDBContent(DbContextOptions<AppDBContent> options):base(options) {

Database.EnsureCreated();

}

}

}

DBMoto.cs

namespace WebApplication1.Data

{

public class DBMoto

{

public string name { get; set; }

public string shortDesc { get; set; }

public string longDesk { get; set; }

public string img { get; set; }

public ushort price { get; set; }

public bool isFavorite { get; set; }

public bool avaible { get; set; }

public int categoryId { get; set; }

}

}

DBObjects.cs

using Microsoft.EntityFrameworkCore;

using WebApplication1.Data.Models;

namespace WebApplication1.Data

{

public class DBObjects

{

private AppDBContent \_content;

public static void Initial(AppDBContent content) {

}

private static Dictionary<string, Category> category;

public static Dictionary<string, Category> Categories

{

get

{

return category;

}

}

public static List<Tovar> Motos

{

get

{

var options = new DbContextOptionsBuilder<AppDBContent>().UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Database=MotoShop;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;Encrypt=True").Options;

using (var context = new AppDBContent(options))

{

return context.Tovar.ToList();

}

}

}

public static List<Category> Categorys

{

get

{

var options = new DbContextOptionsBuilder<AppDBContent>().UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Database=MotoShop;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;Encrypt=True").Options;

using (var context = new AppDBContent(options))

{

return context.Category.ToList();

}

}

}

}

}

DesignTimeDbContextFactory.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

using Microsoft.Extensions.Configuration;

using System.IO;

namespace WebApplication1.Data

{

public class DesignTimeDbContextFactory : IDesignTimeDbContextFactory<AppDBContent>

{

public AppDBContent CreateDbContext(string[] args)

{

IConfigurationRoot configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

var builder = new DbContextOptionsBuilder<AppDBContent>();

var connectionString = configuration.GetConnectionString("DefaultConnection");

builder.UseSqlServer(connectionString);

return new AppDBContent(builder.Options);

}

}

}

Program.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.EntityFrameworkCore;

using WebApplication1.Services;

using WebApplication1.Data;

using WebApplication1.Data.interfaces;

using WebApplication1.Data.mocks;

using WebApplication1.Data.Repository;

using Microsoft.Extensions.Configuration;

using WebApplication1.Data.Models;

namespace WebApplication1

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

var connection = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<AppDBContent>(options => options.UseSqlServer(connection));

builder.Services.AddRazorPages();

builder.Services.AddControllers(options => options.EnableEndpointRouting = false);

builder.Services.AddScoped<IAllMotos, AllMotosRepository>();

builder.Services.AddScoped<IMockMotos, MockMotos>();

builder.Services.AddTransient<IMotosCategory, CategoryRepository>();

builder.Services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

builder.Services.AddMvc();

builder.Services.AddMemoryCache();

builder.Services.AddSession();

builder.Services.AddSingleton<GlobalStashService>();

builder.Services.AddSingleton<GlobalUserService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

endpoints.MapControllerRoute(

name: "stash",

pattern: "stash",

defaults: new { controller = "Stash", action = "Index" });

endpoints.MapRazorPages();

});

using (var scope = app.Services.CreateScope())

{

DBObjects.Initial(scope.ServiceProvider.GetRequiredService<AppDBContent>());

}

app.Run();

}

}

}

GlobalStashService.cs

using System.Collections.Generic;

using System.Linq;

using WebApplication1.Data.Models;

namespace WebApplication1.Services

{

public class GlobalStashService

{

public List<Tovar> Stash { get; private set; }

public GlobalStashService()

{

Stash = new List<Tovar>();

}

public void AddToStash(Tovar moto)

{

if (moto != null && !Stash.Contains(moto))

{

Stash.Add(moto);

}

}

public List<string> GetStashNames()

{

return Stash.Select(m => m.name).ToList();

}

}

}

GlobalUserService.cs

using WebApplication1.Data.Models;

namespace WebApplication1

{

public class GlobalUserService

{

public User CurUser;

public void SetUser(User user)

{

CurUser = user;

}

public User GetCurUser()

{

return CurUser;

}

}

}

dbsettings.json

{

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\MSSQLLocalDB;Database=MotoShop;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;Encrypt=True"

}

}

appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=MotoShop;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;Encrypt=True"

}

}

MotoListViewModel.cs

using WebApplication1.Data.Models;

namespace WebApplication1.ViewModels

{

public class MotoListViewModel

{

public IEnumerable<Tovar> AllMoto { get; set; }

public string currCategory { get; set; }

}

}

StashViewModel.cs

namespace WebApplication1.Data.Models

{

public class StashViewModel

{

public List<Tovar> Tovars { get; set; }

public User CurUser { get; set; }

}

}

Layout.cshtml

<!doctype html>

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Kawasaki</title>

<link rel="shortcut icon" href="~/Image/logo.ico" type="image/x-icon" />

<link href="~/css/site.css" rel="stylesheet" type="text/css" />

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet">

<style>

body {

font-family: 'Roboto', sans-serif;

color: #333;

background-color: #f9f9f9;

margin: 0;

padding: 0;

}

header {

background: linear-gradient(to right, #ff4e50, #f9d423);

width: 100%;

height: 70px;

color: #fff;

border-bottom: 2px solid #ff4e50;

display: flex;

align-items: center;

justify-content: space-between;

padding: 0 20px;

box-sizing: border-box;

position: fixed;

top: 0;

left: 0;

z-index: 1000;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.header-container {

display: flex;

justify-content: space-between;

align-items: center;

width: 100%;

}

.logo-container {

display: flex;

align-items: center;

}

.nav-container {

display: flex;

align-items: center;

}

.round {

border-radius: 30px;

border: 3px solid #ff4e50;

box-shadow: 0 0 15px #666;

}

ul.hr {

margin: 0;

padding: 0;

list-style: none;

display: flex;

align-items: center;

}

ul.hr li {

margin: 0 20px;

font-size: 14pt;

color: #fff;

transition: transform .3s ease, color .3s ease;

}

ul.hr li:hover {

padding: 3px;

color: #ffe29a;

transform: scale(1.1);

}

a {

color: inherit;

text-decoration: none;

}

.login-button {

background-color: #ffd700;

color: #333;

padding: 10px 20px;

border-radius: 20px;

text-decoration: none;

font-size: 14pt;

margin-left: 20px;

transition: background-color 0.3s, transform 0.3s;

}

.login-button:hover {

background-color: #e6ac00;

transform: scale(1.1);

}

main {

padding: 80px 20px 20px;

box-sizing: border-box;

}

footer {

text-align: center;

padding: 20px;

background-color: #ff4e50;

color: #fff;

position: fixed;

bottom: 0;

width: 100%;

}

</style>

</head>

<body>

<header>

<div class="header-container">

<div class="logo-container">

<a class="link" href="/Index">

<img src="/Image/logo.jpg" alt="Logo" class="round" width="50" height="50" />

</a>

</div>

<nav class="nav-container">

<ul class="hr">

<li><a class="link" style="text-decoration:none; color: #191817" href="/Motorcycle">Список товаров</a></li>

<li><a class="link" style="text-decoration:none; color: #191817" href="/Stash">Корзина</a></li>

<li><a class="link" style="text-decoration:none; color: #191817" href="/About">О нас</a></li>

</ul>

<a class="login-button" href="/Login">Войти</a>

</nav>

</div>

</header>

<main>

<div class="main">

@RenderBody()

</div>

</main>

<footer>

<div id="motoDetailsContainer"></div>

<p>&copy; 2024 Kawasaki. All rights reserved.</p>

</footer>

</body>

</html>

Index.cshtml

@model WebApplication1.Data.Models.StashViewModel

@using Microsoft.Web;

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stash</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f8f9fa;

margin: 0;

padding: 20px;

}

.container {

max-width: 1200px;

margin: 0 auto;

}

.moto-block {

background: #fff;

border: 1px solid #ddd;

border-radius: 8px;

padding: 16px;

margin: 16px;

text-align: center;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

display: inline-block;

width: 300px;

transition: transform 0.2s;

}

.moto-block:hover {

transform: scale(1.05);

}

.moto-block img {

max-width: 100%;

border-radius: 8px;

margin-bottom: 15px;

}

.moto-block h2 {

font-size: 1.5em;

margin: 0.5em 0;

color: #333;

}

.moto-block p {

margin: 0.5em 0;

color: #666;

}

.total-sum {

text-align: right;

font-size: 1.5em;

margin: 20px;

color: #333;

}

.download-button {

display: flex;

justify-content: flex-end;

margin: 20px;

}

.btn-download {

background-color: #007bff;

border-color: #007bff;

color: #fff;

padding: 10px 20px;

font-size: 1.2em;

border-radius: 5px;

text-decoration: none;

transition: background-color 0.3s, border-color 0.3s;

}

.btn-download:hover {

background-color: #0056b3;

border-color: #004085;

}

.user-greeting {

margin: 20px 0;

text-align: center;

font-size: 1.5em;

color: #333;

}

</style>

</head>

<body>

<div class="container">

<h1 class="text-center">Stash</h1>

<div class="user-greeting">

@if (Model.CurUser != null)

{

<h1>Купити эти товары @Model.CurUser.login</h1>

}

</div>

<div class="d-flex flex-wrap justify-content-center">

@foreach (var moto in Model.Tovars)

{

String q = Convert.ToString(moto.name) + ".webp";

<div class="moto-block">

<**img** class="img-thumbnail" src="~/Image/@q" alt="@moto.name" />

<h2>@moto.name</h2>

<p>Цена: @moto.price руб.</p>

</div>

}

</div>

<div class="total-sum">

Общая сумма: @Model.Tovars.Sum(m => m.price) руб.

</div>

<div class="download-button">

<**form** method="post" action="/Stash/CreateOrderFile">

<button type="submit" class="btn-download">Скачать заказ</button>

</**form**>

</div>

</div>

</body>

</html>

Index.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using System.Collections.Generic;

using System.IO;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Wordprocessing;

using WebApplication1.Data.Models;

using WebApplication1.Services;

namespace WebApplication1.Pages.Shared

{

public class IndexModel : PageModel

{

private readonly GlobalUserService \_globalUserService;

private readonly GlobalStashService \_globalStashService;

public User DocCurUser { get; set; }

public List<Tovar> tovars { get; set; }

public IndexModel(GlobalUserService globalUserService, GlobalStashService globalStashService)

{

\_globalUserService = globalUserService;

\_globalStashService = globalStashService;

tovars = new List<Tovar>();

}

public void OnGet()

{

DocCurUser = \_globalUserService.CurUser;

tovars = \_globalStashService.Stash;

}

public IActionResult OnPostCreateWordDocument()

{

var documentStream = new MemoryStream();

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(documentStream, DocumentFormat.OpenXml.WordprocessingDocumentType.Document, true))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = new Body();

if (DocCurUser != null)

{

Paragraph para1 = new Paragraph(new Run(new Text($"Заказчик {DocCurUser.login} подтверждает покупку таких товаров как:")));

body.Append(para1);

foreach (var tovar in tovars)

{

Paragraph para = new Paragraph(new Run(new Text(tovar.name)));

body.Append(para);

}

}

else

{

Paragraph para1 = new Paragraph(new Run(new Text("авторизуйтесь для покупки")));

body.Append(para1);

}

mainPart.Document.Append(body);

mainPart.Document.Save();

}

documentStream.Seek(0, SeekOrigin.Begin);

return File(documentStream, "application/vnd.openxmlformats-officedocument.wordprocessingml.document", "PurchaseConfirmation.docx");

}

}

}

About.cshtml

@page

@model WebApplication1.Pages.AboutModel

@{

ViewData["Title"] = "Kawasaki";

}

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewData["Title"]</title>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet">

<style>

body {

font-family: 'Roboto', sans-serif;

margin: 0;

padding: 0;

background-color: #f9f9f9;

color: #333;

}

.content {

padding: 20px;

box-sizing: border-box;

}

.map-container {

width: 100%;

height: 400px;

margin-bottom: 20px;

position: relative;

}

.map-container iframe {

width: 100%;

height: 100%;

border: none;

}

.contact-info {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

max-width: 600px;

margin: 0 auto;

}

.contact-info p {

margin: 10px 0;

}

footer {

text-align: center;

padding: 20px;

background-color: #ff4e50;

color: #fff;

position: fixed;

bottom: 0;

width: 100%;

}

</style>

</head>

<body>

<div class="content">

<a href="https://yandex.by/maps/org/kolledzh\_biznesa\_i\_prava/1050754189/?utm\_medium=mapframe&utm\_source=maps" style="color:#eee;font-size:12px;position:absolute;top:0;">Колледж бизнеса и права</a>

<a href="https://yandex.by/maps/157/minsk/category/college/184106236/?utm\_medium=mapframe&utm\_source=maps" style="color:#eee;font-size:12px;position:absolute;top:14px;">Колледж в Минске</a>

<div class="map-container">

<iframe src="https://yandex.by/map-widget/v1/?l=sat&ll=27.435670%2C53.917988&mode=search&oid=1050754189&ol=biz&sctx=ZAAAAAgBEAAaKAoSCewVFtwPcDtAEVtdTgmI9UpAEhIJnigJibSNjz8RvVZCd0mcdT8iBgABAgMEBSgKOABA%2FZ4GSAFiQXJlbGV2X3JhbmtpbmdfaGVhdnlfY2xpY2tfbWFwc19mb3JtdWxhPTAuMzU6bDNfY2xpY2tfZGM0OTAzMDhfZXhwYkJyZWxldl9yYW5raW5nX2hlYXZ5X2NsaWNrX3dvcmxkX2Zvcm11bGE9MC4zNTpsM19jbGlja19kYzQ5MDMwOF9leHBiQXJlbGV2X3JhbmtpbmdfaGVhdnlfY2xpY2tfc2VycF9mb3JtdWxhPTAuMzU6bDNfY2xpY2tfZGM0OTAzMDhfZXhwagJ1YZ0BzcxMPaABAKgBAL0BWXlkc2VjbmVfZmFjdW5fZGV2aWNlPXplcm9fZGl2aWNlYlRpbWU9MTY0Nzg4MzA5NTA0OQ%3D%3D&sll=27.435670%2C53.917988&source=wizbiz\_new\_map\_single&sspn=0.054580%2C0.018018&tab=overview&text=Колледж%20бизнеса%20и%20права&z=15.2" allowfullscreen="true"></iframe>

</div>

<div class="contact-info">

<p>Мы мото-салон по продаже мотоциклов Kawasaki, у нас вы можете приобрести и оценить лучшие мотоциклы на рынке по демократичным ценам! Работаем круглосутошно, ждём вас!</p>

<p>Наши контакты:</p>

<p>Телефон: +375 (25) 955-49-97</p>

<p>Email: <a href="mailto:4ilobek.4@gmail.com">4ilobek.4@gmail.com</a></p>

</div>

</div>

<footer>

<p>&copy; 2024 Kawasaki. All rights reserved.</p>

</footer>

</body>

</html>

Index.cshtml

@page

@using System.Globalization

@using WebApplication1.Data.Models

@model Motoshtml

@{

ViewData["Title"] = "Kawasaki - Главная";

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>@ViewData["Title"]</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

body {

margin: 0;

padding: 0;

background-color: #000;

color: #fff;

font-family: Arial, sans-serif;

}

.hero-image {

width: 100%;

height: auto;

object-fit: cover;

}

.scrollmenu {

background-color: #333;

overflow-x: auto;

white-space: nowrap;

padding: 20px 0;

}

.scrollmenu .item {

display: inline-block;

color: white;

text-align: center;

margin: 0 10px;

background-color: darkgray;

border-radius: 10px;

overflow: hidden;

width: 300px;

}

.scrollmenu .item img {

width: 100%;

height: 200px;

object-fit: cover;

}

.scrollmenu .item h3, .scrollmenu .item p {

margin: 10px;

}

</style>

</head>

<body>

<**img** src="~/Image/Display.png" alt="Hero Image" class="hero-image">

<div class="container text-center my-5">

@if (@Model.CurUser != null)

{

<h1>@Model.CurUser.login, привет!</h1>

}

else

{

<h1>Зорегистрируйся</h1>

}

<h4 class="mt-4">Популярны сейчас</h4>

</div>

<div class="scrollmenu">

@foreach (var moto in Model.favoriteMotos)

{

String q = Convert.ToString(moto.name) + ".webp";

<div class="item">

<**img** src="~/Image/@q" alt="@moto.name" />

<h3>Модель: @moto.name </h3>

<p>Цена: @moto.price.ToString("c", new CultureInfo("en-GB"))</p>

</div>

}

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-VkpZl0cTAD4aS0M9tk0uGpHAGZT+J0z08n1poaB8ua5e+9HbuqC/d1R3jkh9Uqz8" crossorigin="anonymous"></script>

</body>

</html>

Index.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using WebApplication1.Data;

using WebApplication1.Data.Models;

using WebApplication1.Data.Repository;

namespace WebApplication1.Pages

{

public class Motoshtml : PageModel

{

private readonly AppDBContent \_context;

private readonly GlobalUserService \_globalUserService;

public IEnumerable<Tovar> favoriteMotos { get; set; }

public User CurUser;

public Motoshtml(AppDBContent db, GlobalUserService globalUserService)

{

\_context = db;

\_globalUserService = globalUserService;

}

public void OnGet()

{

AllMotosRepository motorcyces = new AllMotosRepository(\_context);

favoriteMotos = motorcyces.GetFavMotorcycles;

CurUser = \_globalUserService.CurUser;

}

}

}

Login.cshtml

@page

@model LoginModel

@using WebApplication1.Data.Models

@inject WebApplication1.Data.AppDBContent dbContext

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

body {

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #f8f9fa;

margin: 0;

}

.login-container {

background: #ffffff;

padding: 40px;

border-radius: 10px;

width:500px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.login-container h1 {

margin-bottom: 30px;

}

.login-container .form-label {

margin-bottom: 10px;

font-weight: bold;

}

.login-container .btn-primary {

background-color: #007bff;

border-color: #007bff;

}

.login-container .btn-primary:hover {

background-color: #0056b3;

border-color: #004085;

}

.login-container a {

color: #007bff;

text-decoration: none;

}

.login-container a:hover {

text-decoration: underline;

}

</style>

</head>

<body>

<div class="container">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="login-container">

<h1 class="text-center">Login</h1>

<**form** method="post">

<div class="mb-3">

<label for="login" class="form-label">Login:</label>

<input type="text" id="login" name="login" class="form-control" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Password:</label>

<input type="password" id="password" name="password" class="form-control" required>

</div>

<div class="text-center">

<p><a href="/Register">Already haven't an account? Register here.</a></p>

</div>

<div class="mb-3 text-center">

<button type="submit" class="btn btn-primary">Login</button>

</div>

</**form**>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-VkpZl0cTAD4aS0M9tk0uGpHAGZT+J0z08n1poaB8ua5e+9HbuqC/d1R3jkh9Uqz8" crossorigin="anonymous"></script>

</body>

</html>

Login.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using WebApplication1.Data;

using WebApplication1.Data.Models;

using WebApplication1.Services;

namespace WebApplication1.Pages

{

public class LoginModel : PageModel

{

private readonly AppDBContent \_context;

[BindProperty]

public string Login { get; set; }

[BindProperty]

public string Password { get; set; }

private readonly GlobalUserService \_globalUserService;

public LoginModel(AppDBContent context, GlobalUserService globalUserService)

{

\_context = context;

\_globalUserService = globalUserService;

}

public void OnGet()

{

}

public IActionResult OnPost()

{

if (!ModelState.IsValid)

{

return Page();

}

var user = \_context.User.FirstOrDefault(u => u.login == Login && u.password == Password);

if (user == null)

{

ModelState.AddModelError(string.Empty, "Invalid login attempt.");

return Page();

}

\_globalUserService.SetUser(user);

return RedirectToPage("/Index");

}

}

}

MotoDetails.cshtml

@page

@using WebApplication1.Data

@using WebApplication1.Data.Models

@model WebApplication1.Pages.MotosModel

@{

ViewData["Title"] = "Motos";

}

@{

const string defaultId = "1"; // Default value for id

string id = Request.Query["id"].ToString(); // Get id from URL

if (string.IsNullOrEmpty(id))

{

id = defaultId; // Use default value if id is not provided

}

Tovar moto = DBObjects.Motos.FirstOrDefault(c => c.id == Convert.ToInt32(id));

if (moto == null)

{

<p>Moto not found.</p>

return;

}

String q = Convert.ToString(moto.name) + ".webp";

Category category = DBObjects.Categorys.FirstOrDefault(c => c.id == moto.categoryId);

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewData["Title"]</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

#sticky-container {

position: fixed;

width: 400px;

top: 50%;

right: 0;

transform: translateY(-50%);

background-color: white;

padding: 20px;

border: 1px solid #ccc;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

z-index: 1000;

border-radius: 10px;

}

.img-thumbnail {

border-radius: 10px;

}

.btn-custom {

background-color: #ffcc00;

border: none;

color: black;

}

.btn-custom:hover {

background-color: #e6b800;

}

</style>

</head>

<body>

<div id="sticky-container">

<**form** method="post">

<**img** class="img-thumbnail" src="~/Image/@q" alt="@moto.name" width="100%" height="auto" />

<div class="mt-3">

<h3 style="color:black;">Модель: @moto.name </h3>

<p style ="color:black;">Цена: @moto.price.ToString("c")</p>

<p style="color:black;">Категория: @category?.categoryName </p>

<p style="color:black;">@moto.longDesk</p>

<**form** method="post" **asp-page-handler**="AddToStash">

<input type="hidden" name="id" value="@moto.id" />

<button type="submit" formaction="Motorcycle?handler=AddToStash&id=@moto.id" class="btn btn-custom btn-block">Add to Stash</button>

</**form**>

</div>

</**form**>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-VkpZl0cTAD4aS0M9tk0uGpHAGZT+J0z08n1poaB8ua5e+9HbuqC/d1R3jkh9Uqz8" crossorigin="anonymous"></script>

</body>

</html>

MotoDetails.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

namespace WebApplication1.Pages

{

public class MotoDetailsModel : PageModel

{

public void OnGet()

{

}

}

}

Motorcycle.cshtml

@page

@using WebApplication1.ViewModels

@model WebApplication1.Pages.MotosModel;

@{

Layout = "\_Layout";

ViewData["Title"] = "Kawasaki - Главная";

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>@ViewData["Title"]</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

.moto-img {

width: 100%;

height: 250px;

object-fit: cover;

}

.moto-container {

padding: 20px;

}

.btn-custom {

margin-right: 10px;

margin-bottom: 10px;

}

</style>

</head>

<body>

<div class="container mt-5">

<h1 class="text-center">Все мотоциклы</h1>

<div class="mt-3 mb-3 text-center">

<**form** method="post" class="d-inline">

<button type="submit" name="filter" value="1" class="btn btn-primary btn-custom">Мотоциклы</button>

<button type="submit" name="filter" value="2" class="btn btn-primary btn-custom">Детали</button>

<button type="submit" name="filter" value="3" class="btn btn-primary btn-custom">Сопутствующие товары</button>

<button type="submit" name="sort" value="price" class="btn btn-secondary btn-custom">Сортировать по цене</button>

<button type="submit" name="sort" value="name" class="btn btn-secondary btn-custom">Сортировать по названию</button>

</**form**>

</div>

<div class="row mt-5 moto-container">

@if (Model.DisplayedMoto != null)

{

foreach (var moto in Model.DisplayedMoto)

{

String q = Convert.ToString(moto.name) + ".webp";

<div class="col-lg-4 mb-4">

<div class="card">

<**img** class="card-img-top moto-img" src="~/Image/@q" alt="@moto.name" />

<div class="card-body">

<h5 class="card-title">Модель: @moto.name</h5>

<p class="card-text">Цена: @moto.price.ToString("c")</p>

<a href="javascript:void(0)" onclick="loadMotoDetails(@moto.id)" class="btn btn-outline-warning">Подробнее</a>

</div>

</div>

</div>

}

}

</div>

<div id="motoDetailsContainer" class="mt-5"></div>

</div>

<script>

function loadMotoDetails(id) {

fetch(`/MotoDetails?id=${id}`)

.then(response => response.text())

.then(data => {

document.getElementById('motoDetailsContainer').innerHTML = data;

});

}

</script>

</body>

</html>

Motorcycle.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using System.Collections.Generic;

using System.Linq;

using WebApplication1.Data;

using WebApplication1.Data.Models;

using WebApplication1.Services;

namespace WebApplication1.Pages

{

public class MotosModel : PageModel

{

private readonly GlobalStashService \_globalStashService;

private readonly List<Tovar> \_motoss;

public List<Tovar> DisplayedMoto { get; set; }

public List<string> DisplayedMoton { get; set; }

public MotosModel(GlobalStashService globalStashService)

{

\_globalStashService = globalStashService;

\_motoss = DBObjects.Motos;

}

public void OnGet()

{

DisplayedMoto = \_motoss;

DisplayedMoton = \_globalStashService.GetStashNames();

}

public void OnPost(string filter, string sort)

{

IEnumerable<Tovar> filteredMotos = \_motoss;

if (!string.IsNullOrEmpty(filter))

{

int categoryId = int.Parse(filter);

filteredMotos = filteredMotos.Where(m => m.categoryId == categoryId);

}

if (!string.IsNullOrEmpty(sort))

{

if (sort == "price")

{

filteredMotos = filteredMotos.OrderBy(m => m.price);

}

else if (sort == "name")

{

filteredMotos = filteredMotos.OrderBy(m => m.name);

}

}

DisplayedMoto = filteredMotos.ToList();

DisplayedMoton = \_globalStashService.GetStashNames();

}

public IActionResult OnPostAddToStash(int id)

{

var selectedMoto = \_motoss.FirstOrDefault(m => m.id == id);

if (selectedMoto != null)

{

\_globalStashService.AddToStash(selectedMoto);

}

return RedirectToPage("/Motorcycle");

}

}

}

Register.cshtml

@page

@model RegisterModel

@using WebApplication1.Data.Models

@inject WebApplication1.Data.AppDBContent dbContext

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Register User</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<style>

body {

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #f8f9fa;

margin: 0;

}

.register-container {

background: #ffffff;

padding: 40px;

border-radius: 10px;

width: 500px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.register-container h1 {

margin-bottom: 30px;

}

.register-container .form-label {

margin-bottom: 10px;

font-weight: bold;

}

.register-container .btn-primary {

background-color: #007bff;

border-color: #007bff;

}

.register-container .btn-primary:hover {

background-color: #0056b3;

border-color: #004085;

}

.register-container a {

color: #007bff;

text-decoration: none;

}

.register-container a:hover {

text-decoration: underline;

}

</style>

</head>

<body>

<div class="container">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="register-container">

<h1 class="text-center">Register User</h1>

<**form** method="post">

<div class="mb-3">

<label for="login" class="form-label">Login:</label>

<input type="text" id="login" name="login" class="form-control" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Password:</label>

<input type="password" id="password" name="password" class="form-control" required>

</div>

<div class="mb-3">

<label for="confirmPassword" class="form-label">Confirm Password:</label>

<input type="password" id="confirmPassword" name="confirmPassword" class="form-control" required>

</div>

<div class="text-center">

<p><a href="/Login">Already have an account? Login here.</a></p>

</div>

<div class="mb-3 text-center">

<button type="submit" class="btn btn-primary">Register</button>

</div>

</**form**>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-VkpZl0cTAD4aS0M9tk0uGpHAGZT+J0z08n1poaB8ua5e+9HbuqC/d1R3jkh9Uqz8" crossorigin="anonymous"></script>

</body>

</html>

Register.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using WebApplication1.Data;

using WebApplication1.Data.Models;

namespace WebApplication1.Pages

{

public class RegisterModel : PageModel

{

private readonly AppDBContent \_context;

[BindProperty]

public User User { get; set; }

[BindProperty]

public string ConfirmPassword { get; set; }

public RegisterModel(AppDBContent context)

{

\_context = context;

}

public void OnGet()

{

// Метод вызывается при загрузке страницы (HTTP GET)

}

public IActionResult OnPost()

{

if (!ModelState.IsValid)

{

return Page();

}

if (User.password != ConfirmPassword)

{

ModelState.AddModelError("ConfirmPassword", "Passwords do not match.");

return Page();

}

\_context.User.Add(User);

\_context.SaveChanges();

return RedirectToPage("/Login");

}

}

}

# **Приложение Б**

**(справочное)**

**Формы выходных документов**

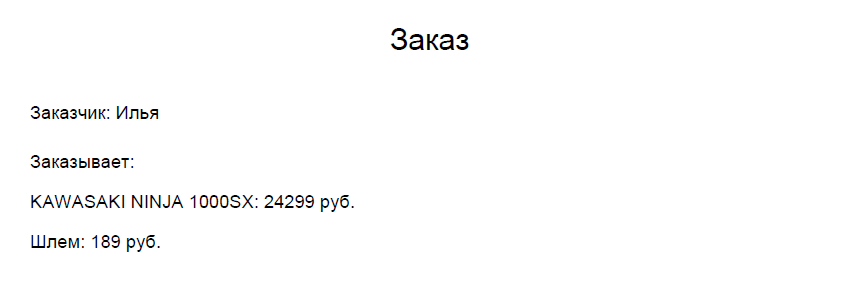


Рисунок Б.1 – Договор заказа