Содержание

[Введение 4](#_Toc184397907)

[1 Анализ предметной области и постановка задачи 6](#_Toc184397908)

[1.1 Анализ предметной области 6](#_Toc184397909)

[1.2 Постановка задачи 6](#_Toc184397910)

[2 Разработка структуры страниц 8](#_Toc184397911)

[2.1 Выбор и обоснование метода построение структуры страницы 8](#_Toc184397912)

[2.2 Выбор и обоснование доступа к содержимому страницы 9](#_Toc184397913)

[2.3 Разработка логической структуры главной страницы 9](#_Toc184397914)

[3 Разработка страниц 11](#_Toc184397915)

[3.1 Разработка шапки страницы 11](#_Toc184397916)

[3.2 Разработка меню навигации по сайту 11](#_Toc184397917)

[3.3 Настройка компонентов взаимодействия с базой данных 12](#_Toc184397918)

[4 Разработка структуры приложения и динамического содержимого страниц 14](#_Toc184397919)

[4.1 Разработка каскадных таблиц стилей CSS 14](#_Toc184397920)

[4.2 Разработка бизнес-логики работы с базой данных 16](#_Toc184397921)

[5 Тестирование сайта 20](#_Toc184397922)

[5.1 Отчёт о тестировании 20](#_Toc184397923)

[5.2 Руководство пользователя 21](#_Toc184397924)

[Заключение 31](#_Toc184397925)

[Список литературы 32](#_Toc184397926)

# Введение

Современные интернет-сервисы для автосалонов становятся важным инструментом в мире автомобильной торговли, обеспечивая удобство и доступность для покупателей и продавцов. В условиях быстро развивающегося рынка автомобилей, где конкуренция возрастает, а потребительские ожидания становятся все выше, такие платформы предлагают инновационные решения для упрощения процесса покупки и продажи автомобилей.

Наш интернет-сервис автосалона предоставляет пользователям возможность легко находить и сравнивать различные модели автомобилей, получать актуальную информацию о ценах и характеристиках, а также осуществлять онлайн-записи на тест-драйвы и консультации. Благодаря интуитивно понятному интерфейсу и мощным фильтрам поиска, пользователи могут быстро находить именно тот автомобиль, который соответствует их потребностям и бюджету.

Одним из таких решений является веб-приложение, построенное с использованием архитектуры MVC (Model-View-Controller). Эта архитектура эффективно разделяет бизнес-логику, данные и пользовательский интерфейс, что способствует упрощению разработки, тестирования и поддерживаемости приложения. Важнейшим компонентом в контексте этой архитектуры является frontend, который отвечает за взаимодействие пользователя с приложением и его визуальное представление. Использование frontend-технологий, таких как HTML, CSS и JavaScript, позволяет создавать интуитивно понятные, динамичные и визуально привлекательные интерфейсы.

HTML — это язык разметки, который используется для структурирования содержания веб-страниц. Он определяет элементы, такие как заголовки, абзацы, списки, ссылки и изображения, с помощью тегов. HTML создает основную структуру страницы и позволяет браузерам отображать контент.CSS — это язык стилей, который отвечает за визуальное оформление веб-страниц. С помощью CSS можно изменять цвет, шрифт, размеры, отступы и расположение элементов на странице. CSS позволяет создавать адаптивные и отзывчивые дизайны, которые выглядят хорошо на различных устройствах. JavaScript — это язык программирования, который добавляет интерактивность и динамичность на веб-страницы. С его помощью можно управлять элементами на странице, обрабатывать события (например, клики и ввод данных), а также взаимодействовать с сервером для загрузки данных без перезагрузки страницы. JavaScript позволяет создавать сложные веб-приложения, такие как игры, формы и интерактивные интерфейсы. HTML, CSS и JavaScript работают вместе, чтобы создать полноценный пользовательский опыт. Эти три технологии являются основой веб-разработки и позволяют создавать современные, функциональные и привлекательные веб-приложения.

Сочетание этих технологий с архитектурой MVC позволяет создать мощное и гибкое приложение, которое будет легко масштабировать и модифицировать. В результате пользователи получат не только эффективный инструмент для анализа авто на рынке, но и удобный интерфейс, который будет интуитивно понятен и доступен для пользователей с разным уровнем технической подготовки.

**1** **Анализ предметной области и постановка задачи**

**1.1** **Анализ предметной области**

Анализ предметной области для интернет-сервиса автосалона включает в себя изучение ключевых аспектов автомобильной торговли, потребностей пользователей и особенностей рынка. Это позволяет определить цели и функциональные требования к сервису, а также выявить конкурентные преимущества.

В рамках интернет-сервиса автосалона выделяются несколько ключевых участников. Во-первых, покупатели, которые могут быть как конечными пользователями, так и корпоративными клиентами, ищущими автомобили для личного использования или бизнеса. Эти пользователи заинтересованы в удобном поиске, сравнении и приобретении автомобилей. Во-вторых, продавцы — это автосалоны и дилеры, предлагающие автомобили на продажу. Они нуждаются в эффективных инструментах для управления своим ассортиментом, продвижения товаров и взаимодействия с клиентами.

Для успешного функционирования интернет-сервиса необходимо учитывать потребности и ожидания пользователей. Одной из основных потребностей является удобство поиска, так как пользователи хотят быстро находить автомобили по различным параметрам, таким как марка, модель, цена и год выпуска. Также важен доступ к актуальной информации о характеристиках автомобилей, ценах, акциях и специальных предложениях. Пользователи ожидают возможности онлайн-взаимодействия, включая запись на тест-драйвы и получение онлайн консультаций.

**1.2** **Постановка задачи**

Сайт интернет-сервиса автосалона должен быть разработан с учетом потребностей пользователей и лучших практик веб-дизайна. Он должен обеспечивать удобный и интуитивно понятный интерфейс, который позволит пользователям легко находить информацию и совершать покупки.

Во-первых, дизайн сайта должен быть современным и привлекательным. Эстетика играет важную роль в восприятии бренда, и визуально привлекательный интерфейс может значительно повысить доверие пользователей. Цветовая палитра должна быть гармоничной и соответствовать теме автомобильной торговли, а шрифты — читабельными и легкими для восприятия.

Во-вторых, навигация на сайте должна быть простой и логичной. Пользователи должны иметь возможность легко перемещаться по различным разделам, таким как каталог автомобилей, информация о текущих акциях, услуги автосалона и контактная информация. Меню должно быть доступным на каждой странице.

При разработке frontend веб-приложения интернет-сервиса автосалона в рамках MVC ASP.NET следует учитывать множество технических требований. Приложение должно быть высокопроизводительным и безопасным, что требует тщательной проработки архитектуры и выбора инструментов.

Для реализации пользовательского интерфейса рекомендуется использовать Razor Views, которые позволяют динамически генерировать HTML-код на основе данных модели. Это упрощает интеграцию между фронтендом и бэкендом, обеспечивая плавное взаимодействие между компонентами приложения.

Кроме того, необходимо реализовать функционал для визуализации данных. Для этого можно использовать JavaScript-библиотеки, такие как Chart.js.

Для наглядного представления была создана диаграмма вариантов использования (рисунок 1).

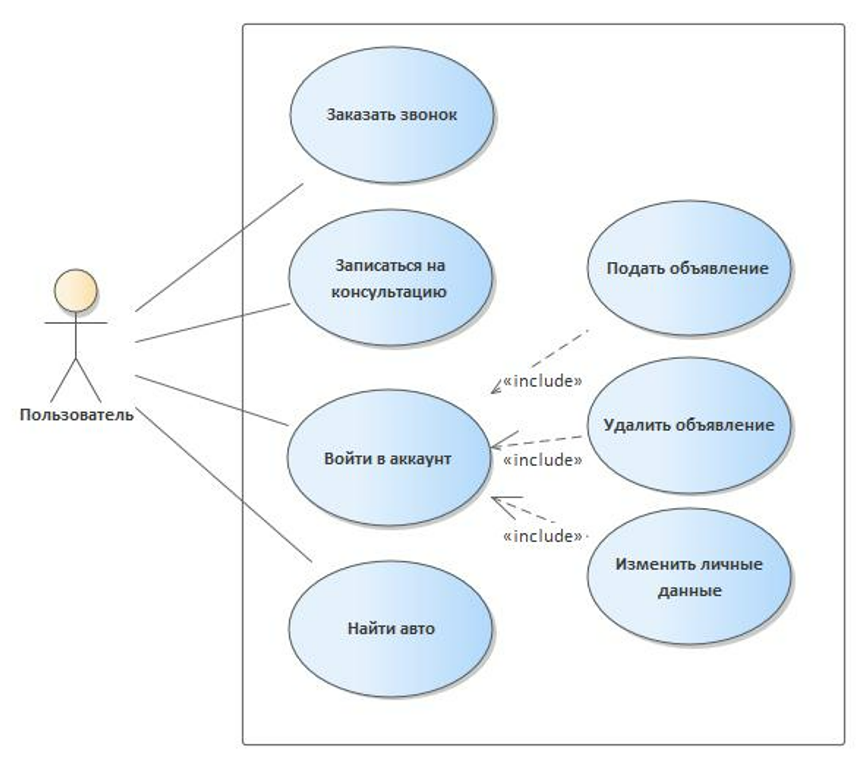


Рисунок 1 – Диаграмма вариантов использования

Таким образом, сайт интернет-сервиса автосалона должен сочетать в себе привлекательный дизайн, удобную навигацию, функциональность, что обеспечит пользователям положительный опыт и поможет успешно конкурировать на рынке.

**2** **Разработка структуры страниц**

**2.1 Выбор и обоснование метода построение структуры страницы**

В процессе создания веб-приложений интернет-сервиса автосалона одним из самых действенных способов организации структуры страниц является применение Razor Pages в ASP.NET. Этот метод предоставляет разработчикам возможность создавать динамичные и интерактивные веб-страницы с помощью кода C#, что существенно упрощает задачу отображения сложных объектов и управления данными.

Основные преимущества Razor Pages:

1. Интуитивно понятная структура;

2. Интеграция C# и HTML: Razor позволяет использовать C# код непосредственно в HTML-разметке. Это позволяет разработчикам легко добавлять динамическое содержимое, например, выводить список транзакций, используя циклы, такие как for или foreach, для перебора коллекций объектов.

Для передачи данных между сервером и клиентом Razor Pages предлагает несколько подходов, среди которых наиболее распространены ViewModel, ViewBag и ViewData.

ViewModel - это наиболее предпочтительный способ передачи данных, так как он обеспечивает строгую типизацию и ясность. ViewModel представляет собой класс, который содержит все необходимые данные, которые должны быть переданы на страницу.

ViewBag - это динамическое свойство, которое позволяет передавать данные с сервера на клиент без необходимости создания отдельного класса.

ViewData - это еще один способ передачи данных, который использует словарь для хранения информации в формате ключ-значение.

Таким образом, использование Razor Pages в ASP.NET для построения структуры страниц предоставляет разработчикам мощный инструмент для создания динамичных веб-приложений. Возможность интеграции C# кода и HTML разметки делает разработку более гибкой и удобной. Применение ViewModel, ViewBag и ViewData позволяет эффективно управлять передачей данных между сервером и клиентом, обеспечивая четкую и структурированную работу с информацией. Это сочетание методов позволяет создать интуитивно понятный и функциональный интерфейс, что особенно важно для приложений, связанных с веб-приложениями интернет-сервиса автосалона.

**2.2 Выбор и обоснование доступа к содержимому страницы**

В процессе создания веб-приложений интернет-сервиса автосалона необходимо учитывать важный аспект — метод доступа к содержимому страниц, основанный на ролях пользователей. Такой подход позволяет эффективно управлять правами доступа в зависимости от уровня полномочий каждого пользователя.

На начальном этапе следует установить различные роли пользователей, такие как клиент и обычный пользователь. Каждая из этих ролей будет иметь свои специфические права доступа к определенным частям приложения.

Внедрение механизма контроля доступа позволяет ограничивать доступ к определенному контенту в зависимости от роли пользователя. В рамках ASP.NET MVC это можно реализовать с помощью атрибутов авторизации, таких как [Authorize] или же используя json файл для хранения состояния пользователя, которые применяются к контроллерам или отдельным методам. Это помогает защищать конфиденциальные данные и предотвращает несанкционированный доступ.

Применение ролей также упрощает управление правами доступа. При необходимости добавления новых функций или ролей достаточно внести изменения в настройки, не затрагивая основной код приложения. Это обеспечивает высокую гибкость и масштабируемость, позволяя легко адаптировать приложение к изменяющимся требованиям бизнеса.

Для обычного пользователя будет предоставлен доступ к просмотру и поиску авто. Обычный пользователь сможет просматривать характеристики авто, а также связываться с менеджерами автосалона. Клиент — это пользователь зарегистрировавшейся в данном веб-приложении. Ему будет представлен свой личный кабинет и возможность выставлять свой авто на продажу.

Таким образом, определён доступ и взаимодействие с содержимым страницы для клиента и обычного пользователя.

**2.3 Разработка логической структуры главной страницы**

При создании главной страницы веб-приложения интернет-сервиса автосалона необходимо тщательно продумать ее логическую структуру. Это обеспечит пользователю интуитивно понятный интерфейс и легкий доступ ко всем необходимым функциям. Основные компоненты структуры включают следующие элементы:

1. Заголовок. Заголовок страницы должен содержать название страницы, которое отображает, что будет на ней происходить.

2. Подзаголовки. Они будут отображать элементы для перехода и просмотра информации о интересующих авто.

3. Карточки элементов. Будут отображать всю информацию о авто.

4. Кнопки управления. Кнопки для управления содержимым, то есть кнопки для удаления, создания, а также кнопки поиска поиска авто.

Таким образом, определена логическая структура страницы пользователя и клиента.

**3** **Разработка страниц**

## 3.1 Разработка шапки страницы

Шапка сайта – это титульная верхняя часть страницы, в которой располагаются элементы навигации (меню) и элементы, содержащие информацию об интернет-приложении.

Рекомендации по разработке шапки:

1. Структура и элементы: шапка обычно включает в себя название приложения, логотип и навигационное меню. Эти элементы помогают пользователю быстро ориентироваться в приложении и переходить к основным разделам.

2. Доступность: важно, чтобы элементы шапки были доступны пользователям на всех страницах приложения. Это может быть достигнуто с помощью макетов или частичных представлений в ASP.NET, которые обеспечивают единый стиль и функциональность.

При разработке приложения было решено добавить шапку на страницы для работы клиента и обычного пользователя. В шапку было решено добавить навигационное меню и логотип приложения. В результате разработки была создана данная шапка (рисунок 1).

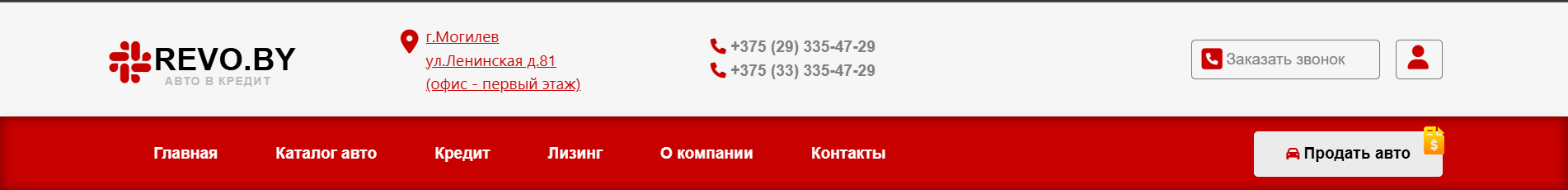


Рисунок 2 – Шапка веб-приложения интернет-сервиса автосалона

Таким образом, была создана шапка для страниц всех пользователей, которая включает в себя логотип приложения и навигационные элементы.

## 3.2 Разработка меню навигации по сайту

Меню навигации является одним из ключевых элементов пользовательского интерфейса интернет-приложения, так как оно обеспечивает пользователям легкий доступ ко всем основным разделам и функциям. Меню навигации должно быть логично структурировано и интуитивно понятным. Оно должно быть доступным на всех страницах сайта. Меню навигации должно быть визуально привлекательным и соответствовать общему стилю приложения. Использование четких шрифтов, контрастных цветов и интуитивных иконок поможет сделать меню более понятным и удобным для пользователей.

В рамках веб-приложения интернет-сервиса автосалона было создано меню навигации, которое находится в шапке страниц.

1.Главная– переход на главную страницу пользователей;

2.Каталог авто– переход на страницу для поиска и просмотра авто;

3.Кредит– переход на страницу с информацией о кредитовании.

4.Лизинг– – переход на страницу с информацией о возможности приобретения авто в лизинг.

5.О компании – переход на страницу с информацией о компании автосалона;

6.Контакты– переход на страницу с информацией о контактах автосалона;

7.Продать авто– переход на страницу для клиентов продажи авто.

Переход между страницами отображён на диаграмме состояний (рисунок 3).



Рисунок 3 – Диаграмма состояний

Таким образом, пользователи приложения будут иметь возможность передвигаться по сайту с помощью навигационного меню, что упрощает взаимодействие с приложением и делает переход между страницами более удобным.

## 3.3 Настройка компонентов взаимодействия с базой данных

При разработке веб-приложения интернет-сервиса автосалона важным аспектом является настройка компонентов, которые обеспечивают взаимодействие с базой данных. Это включает реализацию кнопок для выполнения операций и отображение данных на страницах.

Кнопки в пользовательском интерфейсе играют ключевую роль в обеспечении интерактивности приложения. Основные действия, которые пользователи могут выполнять с данными, включают создание, редактирование и удаление записей.

В рамках клиентской части интернет-приложения автосалона будут добавлены такие элементы управления, как кнопка для продажи авто, а также снятия авто с продажи на данном веб-сервисе.

Для отображения данных из БД на странице используется Razor Pages в ASP.NET. Данные могут быть получены из базы данных и переданы в представление, где они будут динамически отображаться.

Для просмотра данных, которые имеются в БД будут отображены записи с информацией о доходе или расходе. Запись будет содержать нужную информацию: изображение, название и так далее.

**4 Разработка структуры приложения и динамического содержимого страниц**

**4.1 Разработка каскадных таблиц стилей CSS**

Каскадные таблицы стилей (CSS) являются важным инструментом для оформления и стилизации веб-приложений. Они позволяют разработчикам эффективно управлять внешним видом элементов на страницах, обеспечивая при этом гибкость и согласованность дизайна. В этом разделе рассмотрим основные аспекты разработки каскадных таблиц стилей, которые помогут создать визуально привлекательное и удобное приложение для интернет-сервиса автосалона

Для оформления страниц приложения были созданы таблицы стилей. При разработке любой страницы были использованы индивидуальные таблицы стилей, но некоторые из них являются ключевыми и были использованы в каждой из созданных страниц. Такими являются: ColumConteinerMain, flex-item, shadow-text, background-image, footer.

.background-image {

background-size: cover;

background-position: center;

position: absolute;

object-fit: cover;

margin: 0%;

padding: 0%;

width: 100%;

height: 100%;

overflow: hidden;

}

.shadow-text {

text-shadow: 2px 2px 20px rgba(0, 0, 0, 0.8);

}

.flex-item {

font-weight: 600;

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

margin: 0%;

}

.ColumConteinerMain {

display: flex;

flex-direction: column;

background-color: rgb(246, 246, 246);

width: 100%;

height: 100%;

margin: 0%;

padding: 0%;

}

.footer {

display: flex;

justify-content: center;

align-content: center;

align-items: center;

background-color: rgb(23, 23, 23);

width: 100%;

height: 10%;

white-space: nowrap;

}

При разработке так же были использованы модальные окно, для создания данного эффекта была разработана своя таблица стилей где:

Modal – основной скрытый контейнер.

Modak-content – содержимое в скрытом контейнере.

Close – стиль для кнопки закрытия.

Tab – стиль скрытия содержимого.

Active – показывает активный tab.

.modal {

display: none; /\* Скрыто по умолчанию \*/

position: fixed;

z-index: 1000;

left: 0;

top: 0;

width: 100%;

height: 100vh;

background-color: rgba(0, 0, 0, 0.2); /\* Полупрозрачный фон \*/

}

.modal-content {

background-color: rgba(0, 0, 0, 0.5);

height: 100%;

width: 100%;

padding: 20px;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

box-sizing: border-box;

}

.close {

color: #aaa;

font-size: 28px;

font-weight: bold;

position: absolute;

right: 20px;

top: 20px;

cursor: pointer;

}

.close:hover,

.close:focus {

color: black;

text-decoration: none;

}

.close2 {

color: #aaa;

font-size: 28px;

font-weight: bold;

position: absolute;

right: 20px;

top: 20px;

cursor: pointer;

}

.close2:hover,

.close2:focus {

color: black;

text-decoration: none;

}

.tab {

display: none; /\* Скрыто по умолчанию \*/

}

.active {

display: block; /\* Показать активный таб \*/

}

4**.2 Разработка бизнес-логики работы с базой данных**

При создании веб-приложения интернет-сервиса автосалона важным этапом является разработка бизнес-логики, которая управляет взаимодействием с базой данных. Бизнес-логика определяет, как данные будут обрабатываться, хранятся и извлекаться, а также какие правила и условия будут применяться к этому процессу.

Реализация удаления объявления авто для клиента:

public async Task<IActionResult> DeleteAuto(int IdAuto)

{

var auto = await dataBase.Autos.FindAsync(IdAuto); // Найти автомобиль по идентификатору

if (auto == null)

{

return NotFound(); // Если не найден, вернуть 404

}

dataBase.Autos.Remove(auto); // Удалить автомобиль

await dataBase.SaveChangesAsync(); // Сохранить изменения в базе данных

return RedirectToAction("Announcements");

}

Реализация добавления клиента в БД:

[HttpPost]

public async Task<IActionResult> Registration(string Name, string Surname, string Patronymic, string Login, string Password)

{

Users users = new Users();

users.Name = Name;

users.Surname = Surname;

users.Patronymic = Patronymic;

users.License = false;

users.Login = Login;

users.Password = Password;

if (ModelState.IsValid)

{

dataBase.Users.Add(users); // Добавляем пользователя в контекст

await dataBase.SaveChangesAsync(); // Сохраняем изменения

return RedirectToAction("Index"); // Перенаправление на индекс или другую страницу

}

return RedirectToAction("Index"); // Если модель не валидна, возвращаем представление с ошибками

}

Реализация поиска клиента в БД:

public async Task<IActionResult> SubmitAds()

{

int idUser = AppSettings.GetIdUser();

var user = await dataBase.FoundUserSever(idUser);

ViewBag.IdUser = idUser;

ViewBag.Name = user.Name;

ViewBag.Surname = user.Surname;

ViewBag.Patronymic = user.Patronymic;

ViewBag.Login = user.Login;

ViewBag.Password = user.Password;

return View();

}

public async Task<Users> FoundUserSever(int IdUser)

{

var user = await Users.SingleOrDefaultAsync(u => u.IdUser == IdUser);

return user;

}

Реализация обновления данных пользователя в личном кабинете:

[HttpPost]

public async Task<IActionResult> UpdateUser(string Name, string Surname, string Patronymic, string Login, string Password, string Password2)

{

int idUser = AppSettings.GetIdUser();

var user = await dataBase.FoundUserSever(idUser);

if (user != null && Password == Password2)

{

user.Name = Name;

user.Surname = Surname;

user.Patronymic = Patronymic;

user.Login = Login;

user.Password = Password;

dataBase.Users.Update(user);

dataBase.SaveChanges();

return RedirectToAction("Cabinet");

}

return RedirectToAction("Cabinet");

}

Реализация чтения таблицы авто в БД:

public async Task<List<Auto>> GetAllAutoAsync()

{

List<Auto> AllAuto = new List<Auto>();

// Получаем данные из всех таблиц

var autoRecord = await Autos.ToListAsync();

// Заполняем список таблиц

foreach (var auto in autoRecord)

{

AllAuto.Add(auto);

}

return AllAuto;

}

Реализация чтения таблицы пользователь в БД:

public async Task<List<Users>> GetAllUserAsync()

{

List<Users> AllUser = new List<Users>();

// Получаем данные из всех таблиц

var userRecord = await Users.ToListAsync();

// Заполняем список таблиц

foreach (var user in userRecord)

{

AllUser.Add(user);

}

return AllUser;

}

# 5 Тестирование сайта

# 5.1 Отчёт о тестировании

После разработки программного модуля было проведено его тестирование. Результаты тестирования представлены в таблице 1.

Таблица 1 – Тестирование разработанной программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Описание теста | Ожидаемый результат | Результат тестирования | |
| 1 | 2 | 3 | 4 | |
| 1. Модуль входа в систему | | | | |
| 1.1 | Запуск приложения | Отображается главная страница | | + |
| 1.1 | В шапке страницы нажать на кнопку "продать авто" |  | | + |
| 1.2 | В шапке страницы нажать на иконку авторизации | Отображается страница входа | |  |
| 1.3 | В форме поля логин или пароль пользователя пустые | Появилась подсказка внизу незаполненного поля:    Загрузка главной страницы не произошла | | + |
| 1.4 | Введены неверные логин или пароль и  нажатие кнопки «Войти». | Загрузка главной страницы не происходит. Пароль сброшен | | + |
| 1.5 | В форме авторизации пользователь ввёл верные данные и  нажал кнопку «Войти». | Загрузится Главная страница приложения. | | + |
| 2. Личный кабинет пользователя | | | | |
| 2.1 | Изменение пароля в личном кабинете без повторного его введения | Появилась подсказка о повторе пароля | | + |
| 2.2 | На странице «мои объявления» нажата кнопка удалить | Обновляеется страница и из списка поданных объявлений исчезает удаленное | | + |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 2.3 | Нажатие на кнопку добавить авто на странице подачи объявления | Страница обновляется, в БД добавляется новое авто | + |
| 3. Работа с модальными окнами | | | |
| 3.1 | Нажатие на кнопку заказать звонок | Открывается модальное окно с двумя полями и кнопкой «записаться» | + |
| 3.2 | При нажатии на кнопку записаться | Модальное окно закрывается на почту менеджера приходит письмо | + |

Таким образом, были протестированы основные функции приложения.

**5.2 Руководство пользователя**

Начало работы с веб-приложением осуществляется с главной страницы сайта (рисунок 4). На данной страницы представлено основное навигационное меню. Отличие клиента от обычного пользователя заключается в наличии личного кабинета и возможностью продажи авто.



Рисунок 4 – Главная страница сайта

Данная страница является навигационной, некоторые активные элементы введут на указывают одинаковый путь назначения на сайте. Данная особенность связанна с облегчением работы пользователя с сайтом. На главной странице представлены такие навигационные элементы как: переход на главную страницу, переход на страницу каталога авто, переход на страницу кредитования, переход на страницу условия взятия авто в лизинг, переход на информационную страницу о кампании, а также контакты и еще некоторых функции. В таблице 2 указаны элементы присущие на странице, а также их назначение.

Таблица 2 – Элементы главной страницы

|  |  |
| --- | --- |
| Назначение | Элемент |
| 1 | 2 |
| Переход на главную страницу |  |
|  |
| Переход на страницу каталога авто |  |
|  |
|  |
| Переход на страницу кредитования |  |
|  |
| Переход на страницу условия взятия авто в лизинг |  |
|  |
| Переход на информационную страницу о кампании |  |
|  |
|  |

Продолжение таблицы 2

|  |  |
| --- | --- |
| 1 | 2 |
| Переход на страницу о контактах |  |
|  |
| Переход к авторизации пользователя |  |
| Вызов модального окна заказа звонка |  |
|  |
| Переход к просмотру информации о авто |  |

Страница кредитования является информационной и описывает способы приобретения авто в кредит (рисунок 5).

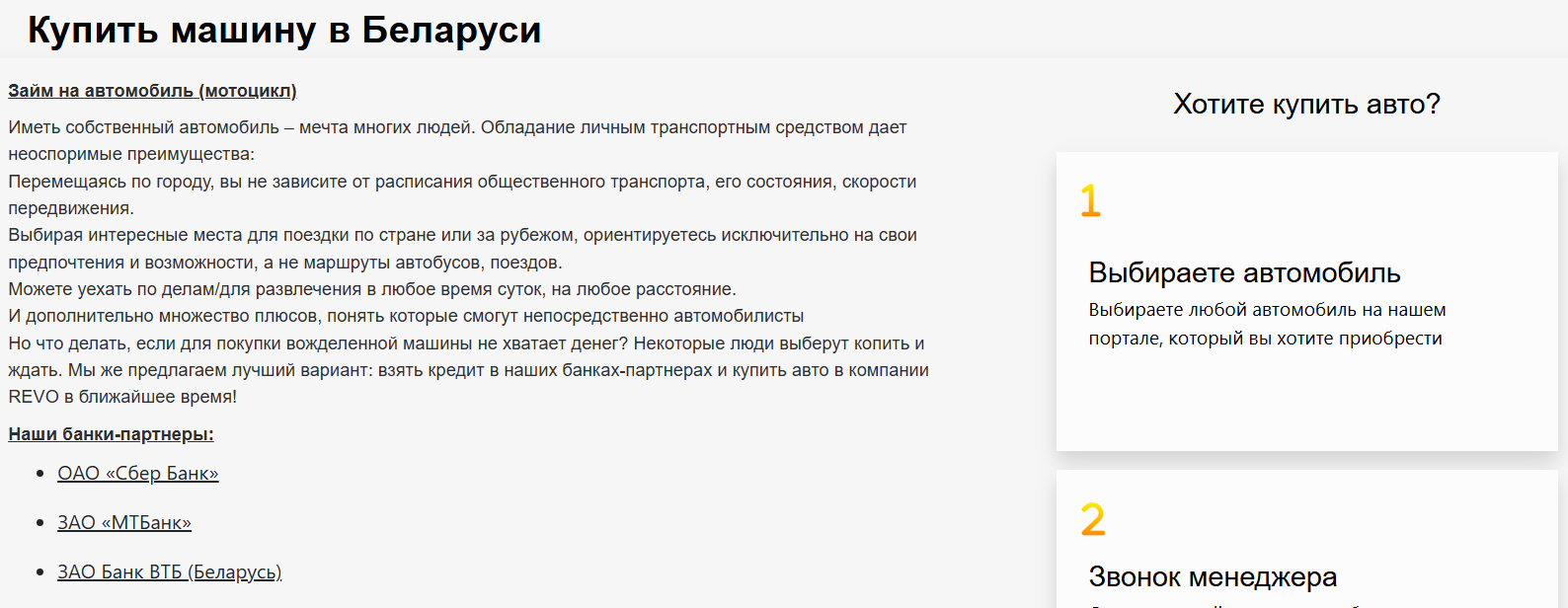


Рисунок 5 – Страница кредитования

Страница приобретения авто в лизинг является информационной и описывает способы приобретения авто в лизинг (рисунок 6).

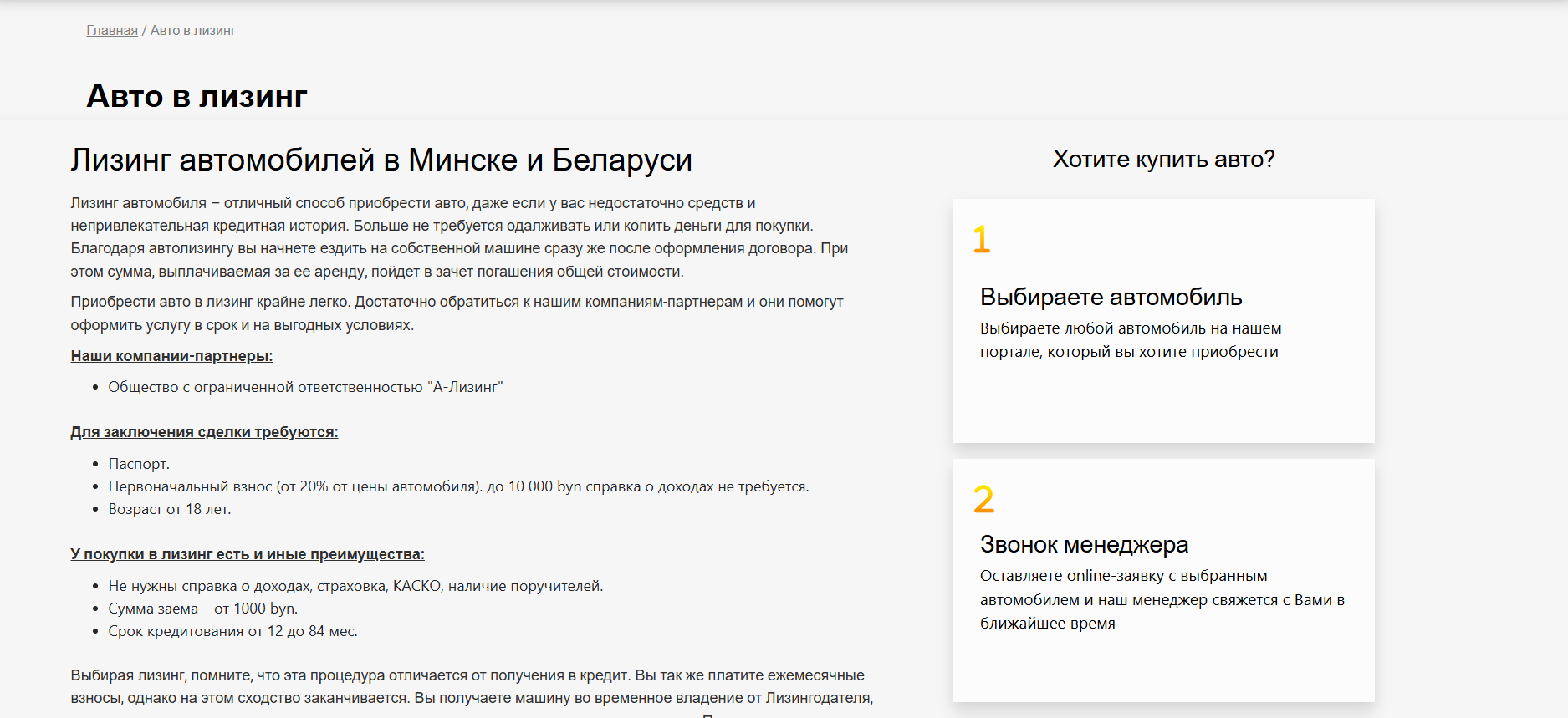


Рисунок 6 – Страница приобретения авто в лизинг

Страница «О компании» является информационной и описывает особенности компании автосалона (рисунок 7).

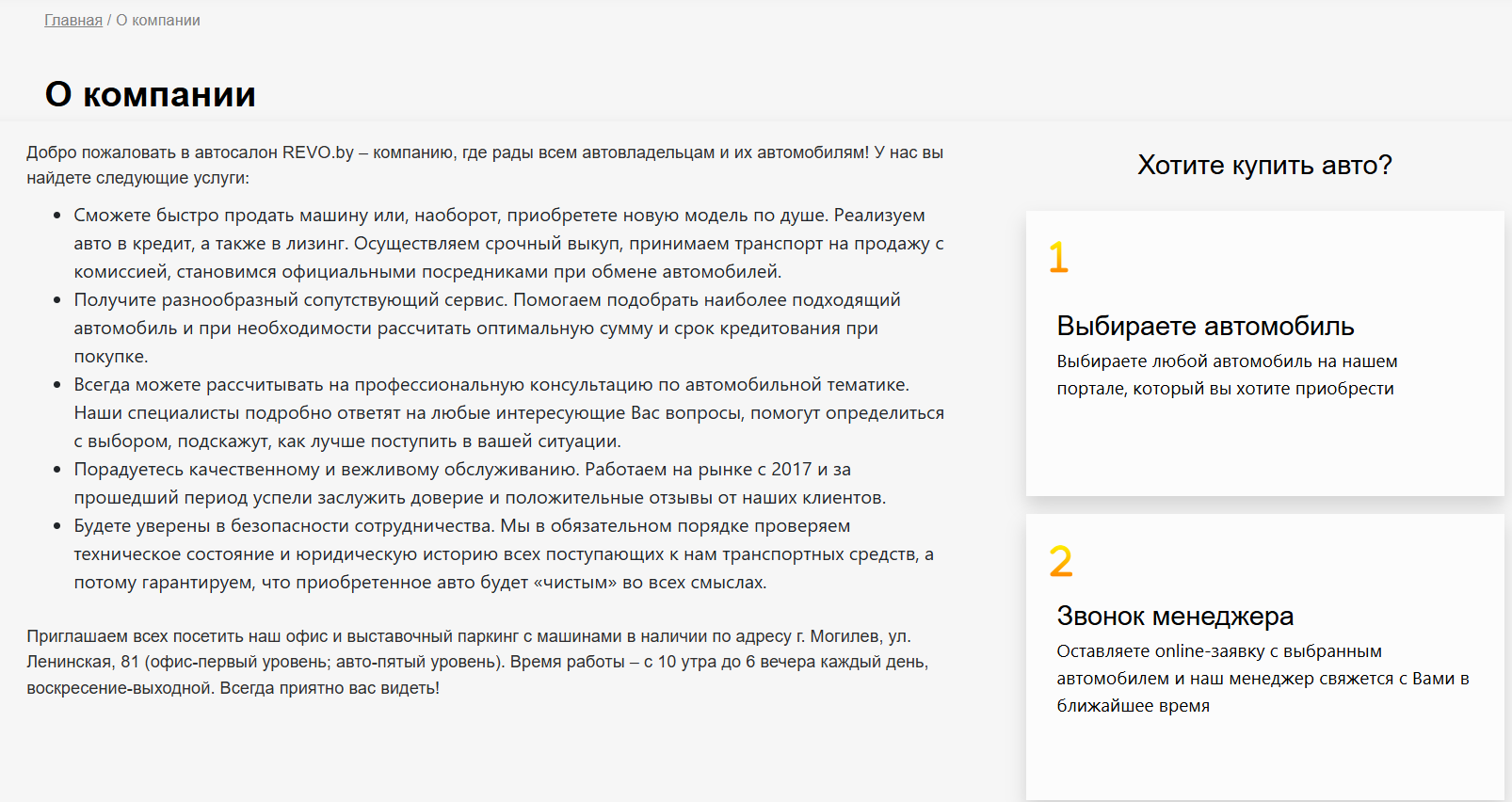


Рисунок 7 – Страница «О компании»

Страница «Контакты» является информационной и описывает методы связи с компанией автосалона (рисунок 8).



Рисунок 8 – Страница «Контакты»

Страница каталог авто предоставляет удобный доступ к информации о интересующих автомобилях (рисунок 9). Поиск авто осуществляется с помощью фильтрации авто по марке, модели, цены и стоимости (рисунок 10). После выбора интересующих параметров авто, страница обновляется и выводит соответствующие автомобили (рисунок 9).

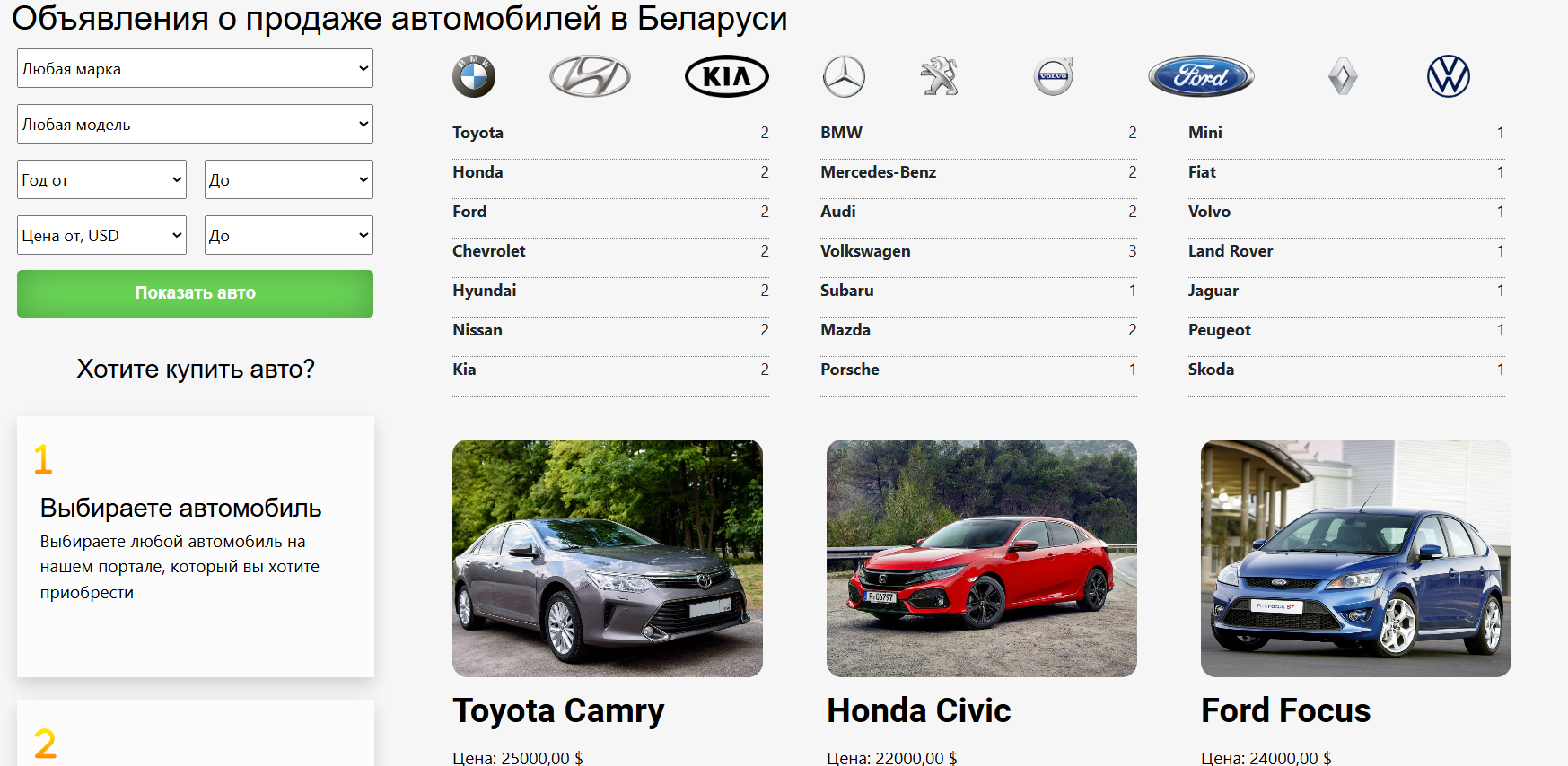


Рисунок 9 – Страница «Каталог авто»

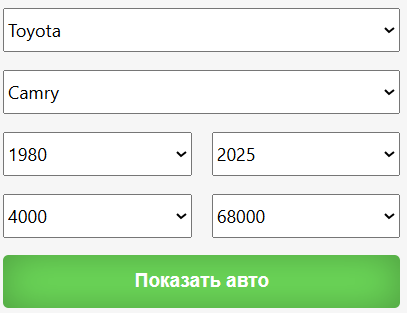


Рисунок 10 – Страница «Фильтрация авто»



Рисунок 11 – Страница «Фильтрация авто»

При нажатии на автомобиль открывается страница описания выборного автомобиля (рисунок 11). В ней представлены такие параметры как год выпуска, тип кузова, тип двигателя и так далее (рисунок 13). При желании приобрести авто или задать интересующие вопросы, имеется возможность оставить заявку (рисунок 14). После отправки заявки, с вами свяжется менеджер.

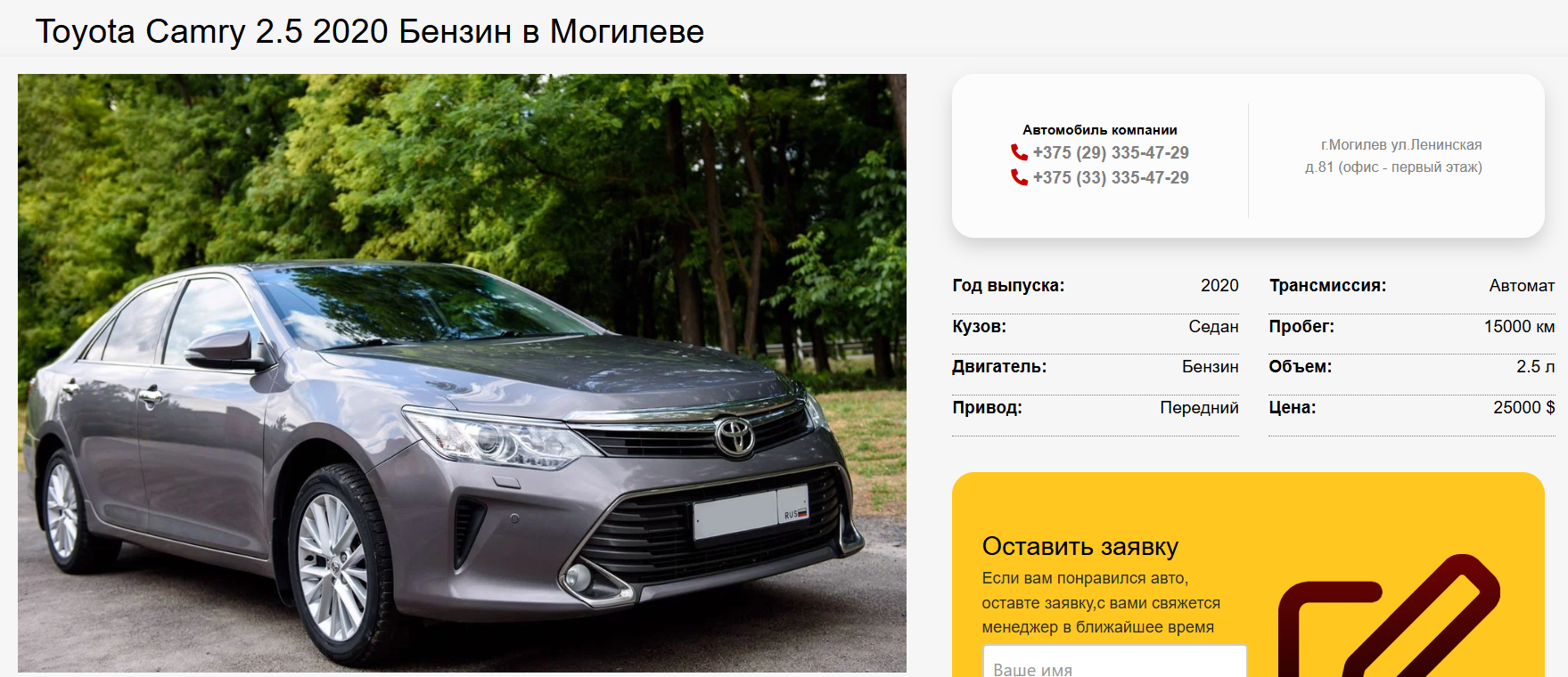


Рисунок 12 – Страница описания авто

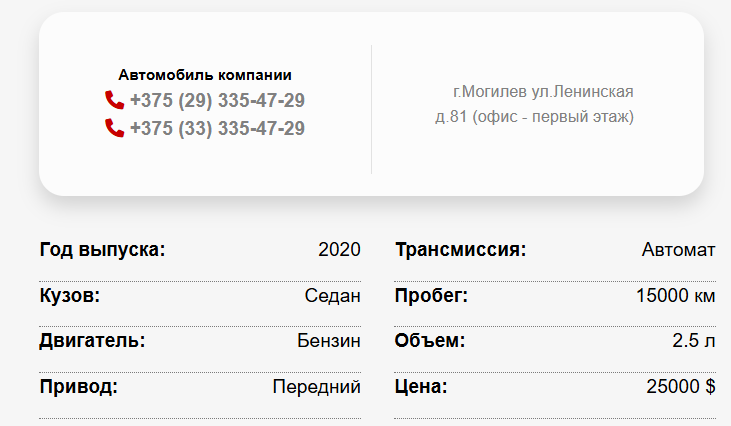


Рисунок 13 – Характеристики авто

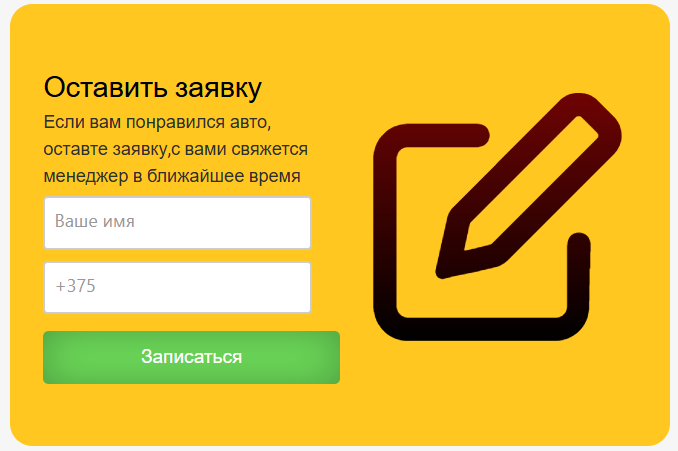


Рисунок 14 – Подача заявки

Если же у вас остались те или вопросы вы можете заказать звонок, нажав на кнопку «Заказать звонок» в шапке на странице. После нажатия появится модальное окно (рисунок 15)

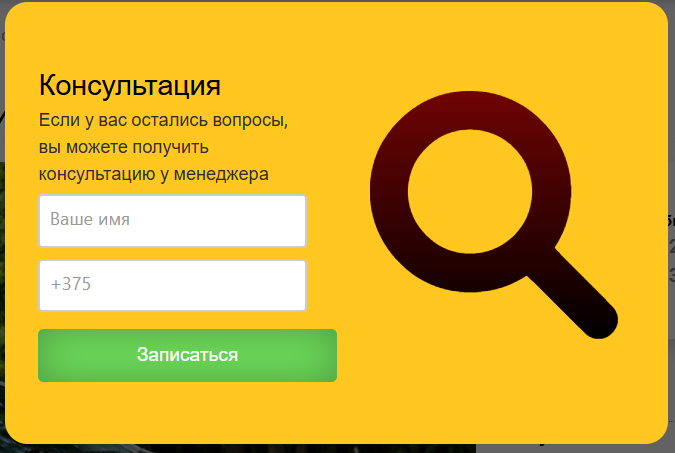


Рисунок 15– Запись на консультацию

При нажатии нажатии на кнопку «Продать авто», сайт попросит вас зайти в аккаунт (рисунок 16). Если вы уже зарегистрированы, то перейдете на страницу продажи авто.

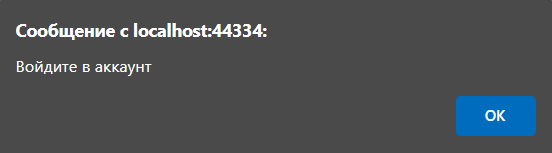


Рисунок 16– Запись на консультацию

Что бы войти в аккаунт требуется ввести логин и пароль (рисунок 17), если же у вас нет аккаунта, то требуется зарегистрироваться (рисунок 18).

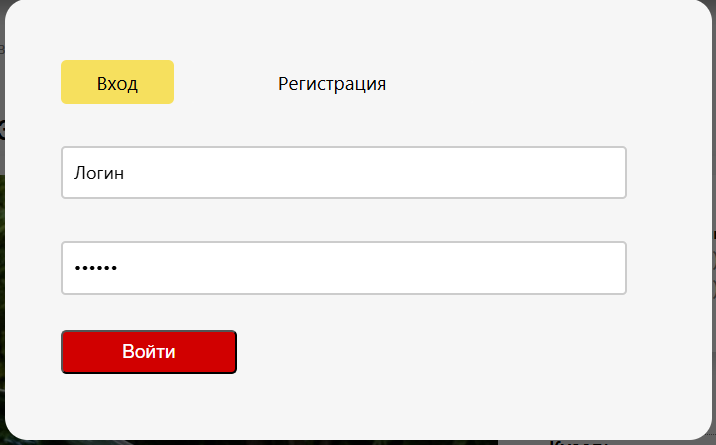


Рисунок 17– Вход в аккаунт

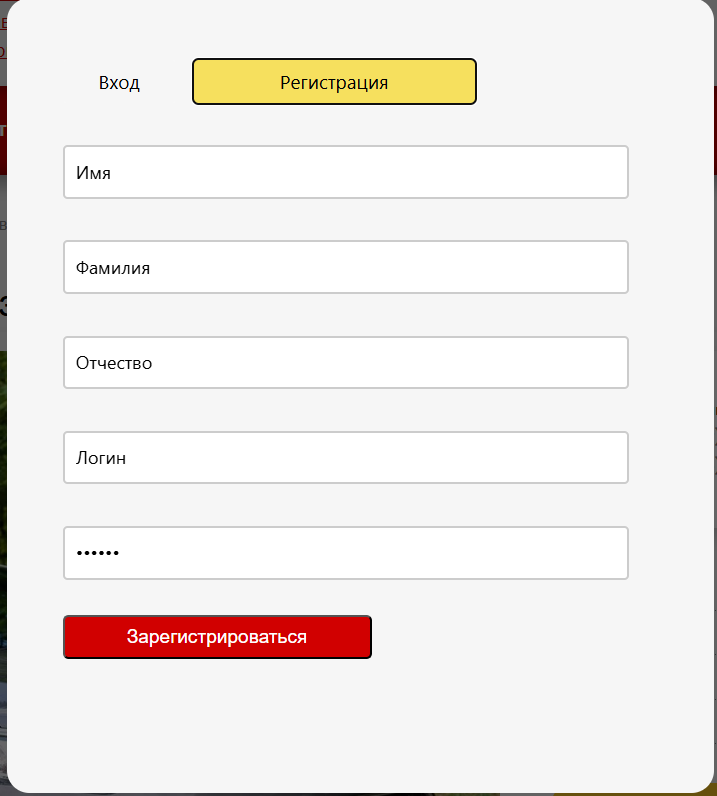


Рисунок 18– Регистрация

После входа у вас появится возможность зайти в свой личный кабинет (рисунок 19). В личном кабинете вы можете подать свое объявления, просмотреть уже поданные, а также поменять свои личные данные.

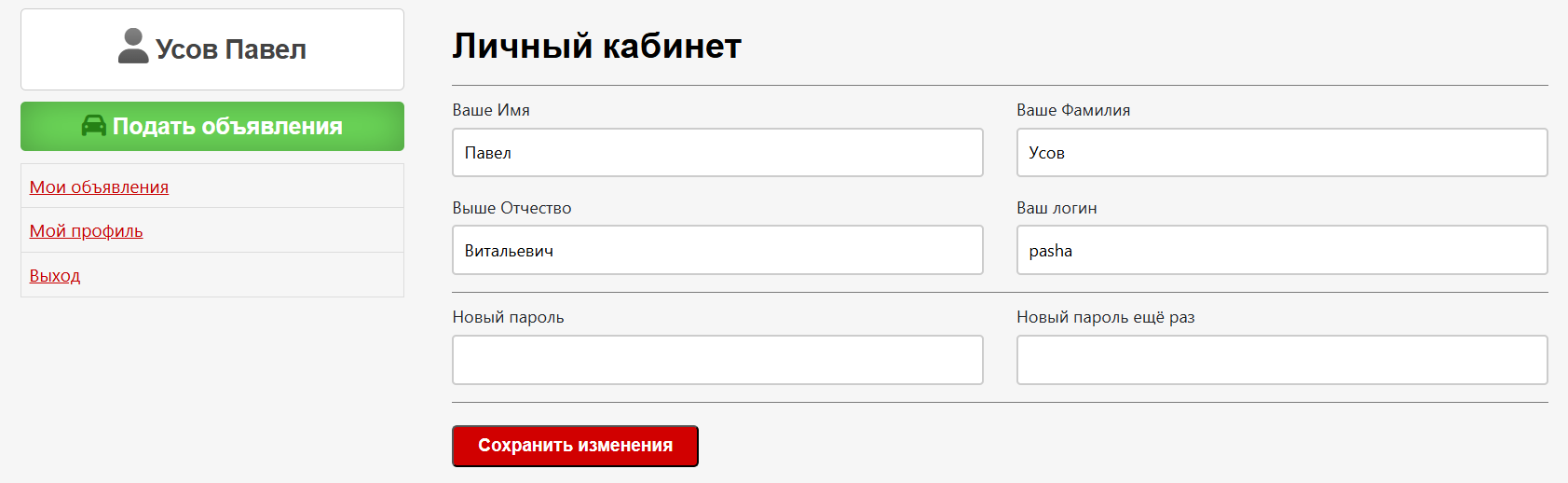


Рисунок 19– Личный кабинет

Свои объявления можно просмотреть, а также удалить, нажав на кнопку «Удалить» (рисунок 20).

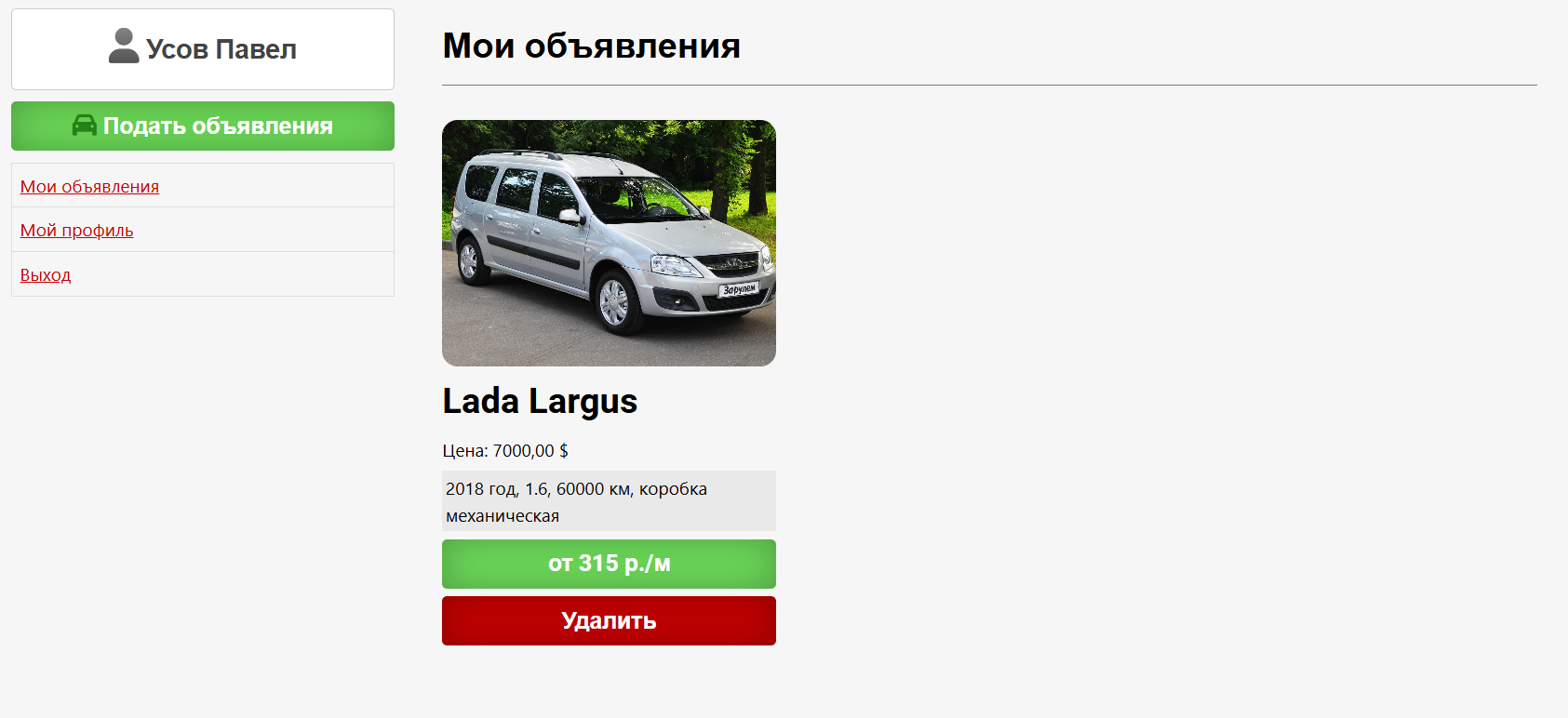


Рисунок 20– Мои объявления

Что бы подать объявления требуется нажать на кнопку «Подать объявления». Далее требуется заполнить характеристики авто и нажать на кнопку «Подать объявления» (рисунок 21).

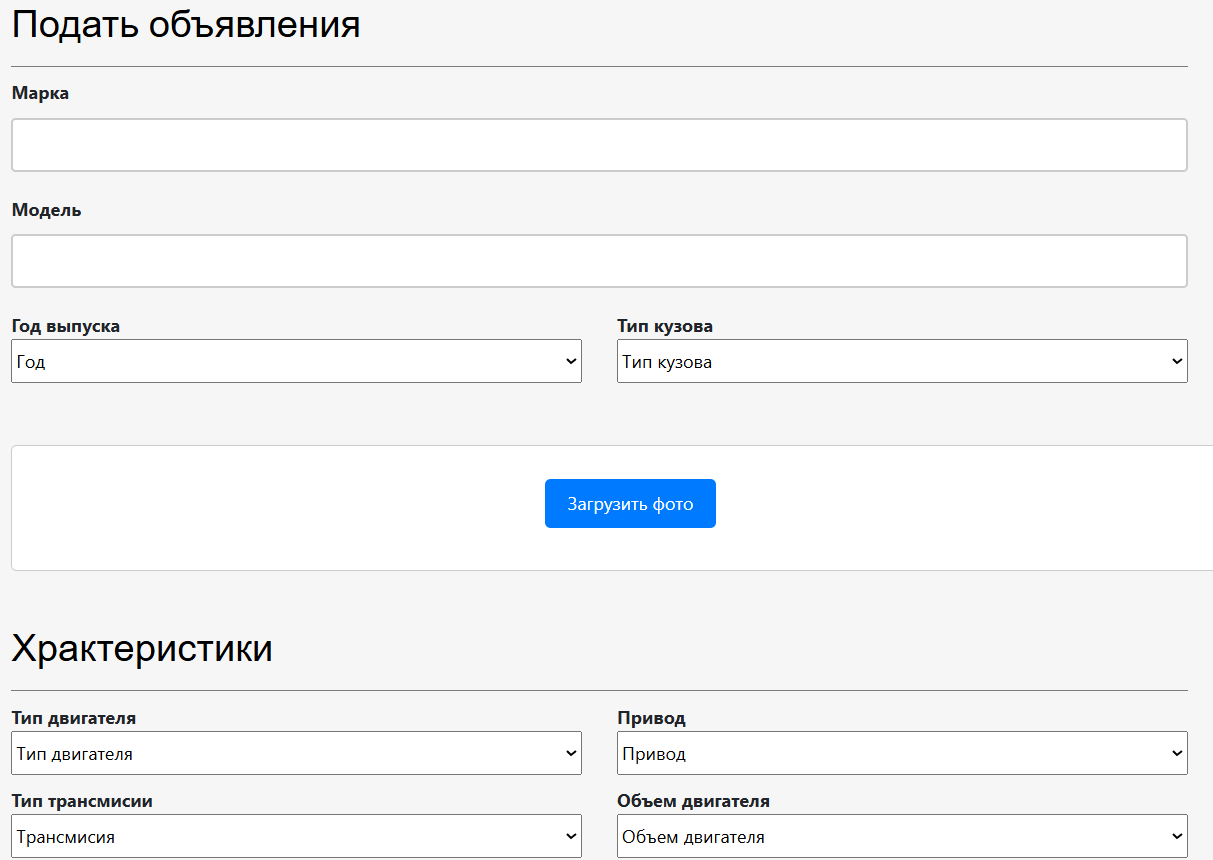


Рисунок 21– Подача объявления

# Заключение

Создание интернет-сервиса автосалона — это важный шаг к улучшению взаимодействия с клиентами и оптимизации бизнес-процессов. Такой сайт должен быть не только визуально привлекательным, но и функциональным, отвечая потребностям пользователей в удобстве и доступности информации о предлагаемых автомобилях.

Сайт должен предоставлять пользователям возможность легко находить, а также получать актуальную информацию о ценах, характеристиках и акциях. Интуитивно понятный интерфейс, мощные инструменты поиска и фильтрации, а также возможность онлайн-взаимодействия, такие как запись на тест-драйвы и консультации, сделают процесс покупки более удобным и эффективным.

Использование архитектуры MVC (Model-View-Controller) в разработке таких приложений позволяет эффективно разделять бизнес-логику, данные и визуальное представление, что делает приложение более масштабируемым и поддерживаемым. Frontend, как неотъемлемая часть архитектуры, отвечает за взаимодействие с пользователем, обеспечивая интуитивно понятный и динамичный интерфейс, реализованный с использованием HTML, CSS и JavaScript. Эти технологии обеспечивают как структуру страницы, так и её визуальное оформление и интерактивность, что делает веб-приложение функциональным и удобным в использовании.

В результате сочетания MVC и передовых frontend-технологий, пользователи получают мощный и гибкий инструмент для управления объявлениями авто в веб-приложении автосалона. Такой подход обеспечивает не только эффективный учет, но и адаптивный интерфейс, подходящий для людей с различным уровнем технической подготовки. Таким образом, создание веб-приложения интернет-сервиса автосалона становится важной составляющей современных технологий, ориентированных на улучшение качества жизни пользователей.

# Список литературы

1. **Хеник, Б.** - HTML и CSS. Путь к совершенству (HTML и CSS: The Good Parts), 2017 – 284 с.

2. **Флэнаган, Д.** - "JavaScript. Подробное руководство (JavaScript. The Definitive Guide)", 2018. – 992 с.

3. **Полуэктова, Н. Р.** - Разработка веб-приложений. — М.: Юрайт, 2024. — 205 c.

4. **Кангин,** В. В. Интернет. Языки HTML и JavaScript. — М.: ТНТ, 2021. — 488 c.

5. **Нагаева, И. А.; Фролов, А. Б.; Кузнецов, И. А.** Основы web-дизайна. Методика проектирования. — М.: Директ-Медиа, 2021. — 184 c.

6. **Бэнкс, А.** – React и Redux. Функциональная веб-разработка. Руководство / Бэнкс Алекс. - М.: Питер, 2018. - 458 c.

7. **Хоган, Б. П.** – HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Брайан П. Хоган. - М.: Питер, 2013. — 128 c.