

Инженерная школа Информационных технологий и робототехники

Направление подготовки 09.04.01 Информатика и вычислительная техника

Профиль Разработка интернет-приложений

Отделение Информационных технологий

**Курсовая работа по дисциплине**

**«Технологии разработки программного обеспечения»**

|  |
| --- |
| **Тема работы** |
| **ТЕМА РАБОТЫ** |

Студент

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа** | **ФИО** | **Подпись** | **Дата** |
| Группа | ФИО |  |  |

Проверил

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Должность** | **ФИО** | **Ученая степень, звание** | **Подпись** | **Дата** |
| Доцент | Поляков А.Н. | к.т.н. |  |  |

Томск – 2022 г.

Оглавление

[Введение 4](#_Toc93256763)

[1 ТРЕБОВАНИЯ К ПРОГРАММЕ 7](#_Toc93256764)

[1.1 Назначение программы 7](#_Toc93256765)

[1.2 Область применения 7](#_Toc93256766)

[1.3 Задачи, решаемые программой 7](#_Toc93256767)

[1.4 Функциональные требования 7](#_Toc93256768)

[1.5 Требования к надежности 8](#_Toc93256769)

[1.6 Требования к эргономике и технической эстетике 9](#_Toc93256770)

[1.7 Требования к программной документации 9](#_Toc93256771)

[1.8 Стадии и этапы разработки 9](#_Toc93256772)

[1.9 Диаграмма вариантов использования 10](#_Toc93256773)

[1.10 Детализация вариантов использования 10](#_Toc93256774)

[1.10.1 Вариант использования «Получить список сданных в аренду автомобилей» 11](#_Toc93256775)

[1.10.2 Вариант использования «Оставить отзыв» 12](#_Toc93256776)

[1.10.3 Вариант использования «Поиск автомобиля» 13](#_Toc93256777)

[1.10.4 Вариант использования «Фильтрация автомобиля» 14](#_Toc93256778)

[1.10.5 Вариант использования «Добавление автомобиля» 15](#_Toc93256779)

[1.10.6 Вариант использования «Посмотреть арендованные» 16](#_Toc93256780)

[1.10.7 Вариант использования «Оставить заявку на желаемое авто» 17](#_Toc93256781)

[1.10.8 Вариант использования «Арендовать машину» 18](#_Toc93256782)

[2 АНАЛИЗ 19](#_Toc93256783)

[2.1 Диаграмма классов анализа 19](#_Toc93256784)

[2.2 Диаграмма состояний 21](#_Toc93256787)

[3 ПРОЕКТИРОВАНИЕ 24](#_Toc93256788)

[3.1 Проектные классы 24](#_Toc93256789)

[3.2 Диаграмма пакетов системы 25](#_Toc93256790)

[3.3 Диаграммы последовательностей для операций проектных классов 26](#_Toc93256791)

[4 РЕАЛИЗАЦИЯ 33](#_Toc93256792)

[4.1 Разработка приложения 33](#_Toc93256793)

[4.2 Тестирование 34](#_Toc93256794)

[4.2.1 Модульные тесты 35](#_Toc93256795)

[4.2.2 Интеграционные тесты 38](#_Toc93256796)

[4.2.3 Построение и выполнение тестов 38](#_Toc93256797)

[4.2.4 Покрытие кода 39](#_Toc93256798)

[4.2.5 Запуск приложения для тестирования 40](#_Toc93256799)

[5 ДОКУМЕНТАЦИЯ 41](#_Toc93256800)

[5.1 Назначение программы 41](#_Toc93256801)

[5.2 Условия запуска программы 41](#_Toc93256802)

[5.3 Выполнение программы 41](#_Toc93256803)

[5.3.2 Авторизация пользователя 42](#_Toc93256804)

[5.3.3 Редактирование профиля 42](#_Toc93256805)

[5.3.5 Добавления автомобиля 44](#_Toc93256806)

[5.3.6 Поиск автомобиля 45](#_Toc93256807)

[5.3.7 Аренда автомобиля 46](#_Toc93256808)

[5.3.8 Отмена аренда 46](#_Toc93256809)

[5.3.9 Завершение аренды 47](#_Toc93256810)

[5.4 Сообщения программы 47](#_Toc93256811)

[ЗАКЛЮЧЕНИЕ 51](#_Toc93256812)

[Список использованных источников 52](#_Toc93256813)

# Введение

Мы живем в 21 веке, веке высоких технологий, в веке, в котором прогресс это непрерывный процесс. Прогресс идет таким темпом, что человек сам уже не способен справляться со всеми проблемами. В этом ему помогают им же созданные технические достижения. Компьютерная техника так прочно вошла в нашу жизнь, что мы уже не можем представить свое существование без неё. Все технические новации и достижения созданы человеком, чтобы облегчить его жизнь. Мы живем в городах, население которых насчитывается десятками тысяч, а порой и миллионами. Современный мир настолько огромен и разнообразен, что найти нужную тебе вещь достаточно сложно. Как говорил Кейнс – «Спрос рождает предложение». А в современном мире у любого спроса всегда есть выбор, какой из предложенных товаров выбрать. И победит тот товар, который будет наиболее качественен, популярен, а также доступен. Зачем кому-то ехать в другой конец города, полдня стоять в пробках, разглядывать множество товаров в магазине, если можно сделать всё это удаленно. Просто необходимо зайти на сайт нужной вам организации или магазина, и просмотрев спокойно весь ассортимент, выбрать нужный товар.

Интернет довольно сильно вошел в повседневную жизнь. Люди часами сидят во всемирной паутине. Они общаются, знакомятся, обсуждают новости, развлекаются, делятся фотографиями. И делать покупки им тоже проще осуществлять через интернет. Есть те, кому некогда или они не могут поехать за покупками, к примеру, из-за маленьких детей. Другим просто не охота делать это хотя бы, потому что через интернет это сделать проще. Также нельзя исключать людей с ограниченными возможностями, ведь для них это может оказаться настоящим испытанием. Всё больше и больше людей начинают осуществлять свои покупки через интернет, и было бы не разумно не развивать такой вид бизнеса. Ведь он довольно выгоден, зачастую ассортимент интернет магазина намного разнообразнее простого супермаркета. На его содержание не нужно таких огромных площадей для торговых рядов и прилавков, они работают 24 часа в сутки и 7 дней в неделю, стоимость товара в интернет-магазинах зачастую ниже, из-за меньших издержек.

Не все вещи нам нужны на постоянное пользование. А могут нам понабиться лишь на время, стоимость может быть большой, а их покупка может себя не оправдать. Для решения этой проблемы еще в 12 веке была придумана аренда. Эта услуга удачно перекочевала в интернет. Всё по той же причине – это позволяет экономить время. А также даёт возможность ознакомиться со всем списком и его характеристиками товара. Для поиска нужных нам товаров есть множество фильтров, с помощью которых поиск облегчается.

Целью курсового проекта является разработка интернет-сервиса ООО «Tomsk-Car-Rent».

Объектом данного курсового проекта является магазин ООО " Tomsk-Car-Rent", который занимается услугами аренды автомобилей.

Данный проект был разработан на языке Python. Python – интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Выбор пал на Python благодаря его многочисленным достоинствам:

* интерпретатор Python адаптирован практически для всех платформ и операционных систем;
* расширяемость языка – имеется возможность совершенствования языка всеми заинтересованными программистами;
* наличие широкого набора подключаемых к программе модулей, которые обеспечивают различные дополнительные возможности;
* стандартная библиотека позволяет использовать сторонние решения;
* функционал сайтов на языке Python ограничен лишь фантазией клиента, при этом, как правило, разработка даже сложных решений занимает меньше времени, чем на других языках. Основанием для выбора языка программирования Python явились следующие его преимущества: повышенная безопасность, надежность исходных кодов, что, снижает вероятность взлома и уязвимости от различного рода атак. Для написания кода приложения используется среда разработки PyCharm.

PyCharm – среда разработки для языка Python, которая имеет бесплатную Community Edition лицензию. Среда подключается к интерпретатору и готова к использованию. PyCharm имеет встроенный дебаггер, удобный редактор кода, инструменты для рефакторинга.

Для создания приложения использовалась платформа Django. С помощью простой и надежной платформы Django, основанной на языке Python, можно создавать мощные веб-решения, написав всего несколько строк кода. Ее использование позволяет обеспечить высокую скорость и гибкость разработки, а также решать широкий спектр прикладных задач.

# 1 ТРЕБОВАНИЯ К ПРОГРАММЕ

## 1.1 Назначение программы

Сайт фирмы ООО «Tomsk-Car-Rent» предназначен для сдачи в аренду автомобилей. На сайте можно найти машину, удовлетворяющую вашим требованиям и предпочтениям.

## 1.2 Область применения

Разрабатываемый программный продукт предназначен для частных лиц и организаций желающих сдать в аренду свое авто либо арендовать чужой автомобиль.

## 1.3 Задачи, решаемые программой

Для обеспечения возможности решать требуемые задачи с помощью данного приложения, необходимо соблюдать следующие требования:

1. Язык реализации – Python с интерпретатором версии не ниже 3.4;
2. Фреймворк Django версии не ниже 1.8;
3. Основной язык программы – русский;
4. Система должна производить валидацию полей не более, чем в течение трех секунд;
5. Система должна производить авторизацию пользователя не более, чем в течение трех секунд;
6. Данная система должна разрабатываться и функционировать в рамках существующего законодательства Российской Федерации.
7. Функционал системы должен отвечать соответствующим заявленным требованиям.

## 1.4 Функциональные требования

1. Программа «Найди Автомобиль» (далее система) должна предоставлять полный список автомобилей, которые добавлены в данную систему, с отображением их изображений и описания;
2. Система должна отображать список автомобилей отфильтрованных по выбранной марке;
3. Система должна предоставить полный список марок автомобилей;
4. Система должна подсчитывать общую сумму стоимости аренды;
5. Система должна предоставлять возможность отмены аренды до ее начала;
6. Система должна предоставлять пользователю возможность выбора даты аренды;
7. Система должна предоставлять форму для ввода необходимых для оформления аренды данных;
8. Система должна сохранять данные об аренде после ее оформления;
9. Система должна предоставлять возможность аутентификации пользователя;
10. Система должна предоставлять возможность авторизации пользователя;
11. Система должна предоставлять возможность просмотра авторизованному пользователю аренд, сделанных в данной программной системе;

## 1.5 Требования к надежности

Программа должна корректно реагировать на действия пользователя в соответствии на задаваемые им запросы.

Программа должна контролировать корректность вводимых данных перед записью в базу данных, т. е. в случае попытки записи в базу данных с не заполненными обязательными полями игнорировать данную операцию (запись не производить) и требовать заполнения всей необходимой информации.

Интерфейс должен быть интуитивно понятен и приветлив. Он не должен изобиловать большой палитрой цветов, отвлекать пользователя.

## 1.6 Требования к эргономике и технической эстетике

Взаимодействие пользователя с программой должно осуществляться посредством графического интерфейса программной системы, с помощью браузера. Интерфейс программы должен быть выполнен в едином стиле. На главном экране иметь удобное меню навигации, при перемещении по страницам основные элементы навигации должен иметь одинаковое расположение.

При вводе некорректных данных, должны выдаваться сообщение о том, что данные введены не корректно, с просьбой ввести данные корректно.

## 1.7 Требования к программной документации

Документация представлена в виде отчета по курсовой работе, который состоит из следующих частей:

1. Титульный лист, номинальный объем – 1страница;
2. Содержание, номинальный объем – 1 страница;
3. Введение, номинальный объем – 1 страница;
4. Требования к программе, номинальный объем – 8 страниц;
5. Анализ, номинальный объем – 8 страниц;
6. Проектирование, номинальный объем – 8 страниц;
7. Реализация, номинальный объем – 8 страниц;
8. Документация, номинальный объем – 8страниц;
9. Заключение, номинальный объем – 1 страница;
10. Список использованных источников, номинальный объем – 1 страница.

## 1.8 Стадии и этапы разработки

Сроки разработки: с 1 сентября по 22 декабря 2015 г.

Основные этапы разработки программы:

1. Описание требований к системе;
2. Выявление классов. Построение и описание диаграммы классов анализа;
3. Построение и описание диаграммы состояний;
4. Построение и описание проектных классов;
5. Построение и описание диаграмм последовательности для операций проектных классов;
6. Построение и описание диаграммы пакетов;
7. Разработка программы;
8. Модульное тестирование;
9. Настройка системы непрерывной интеграции.

## 1.9 Диаграмма вариантов использования

Диаграммой вариантов использованияназывается диаграмма, на которой показана совокупность вариантов использования (по-другому, прецедентов) и актеров, а также отношения между ними.

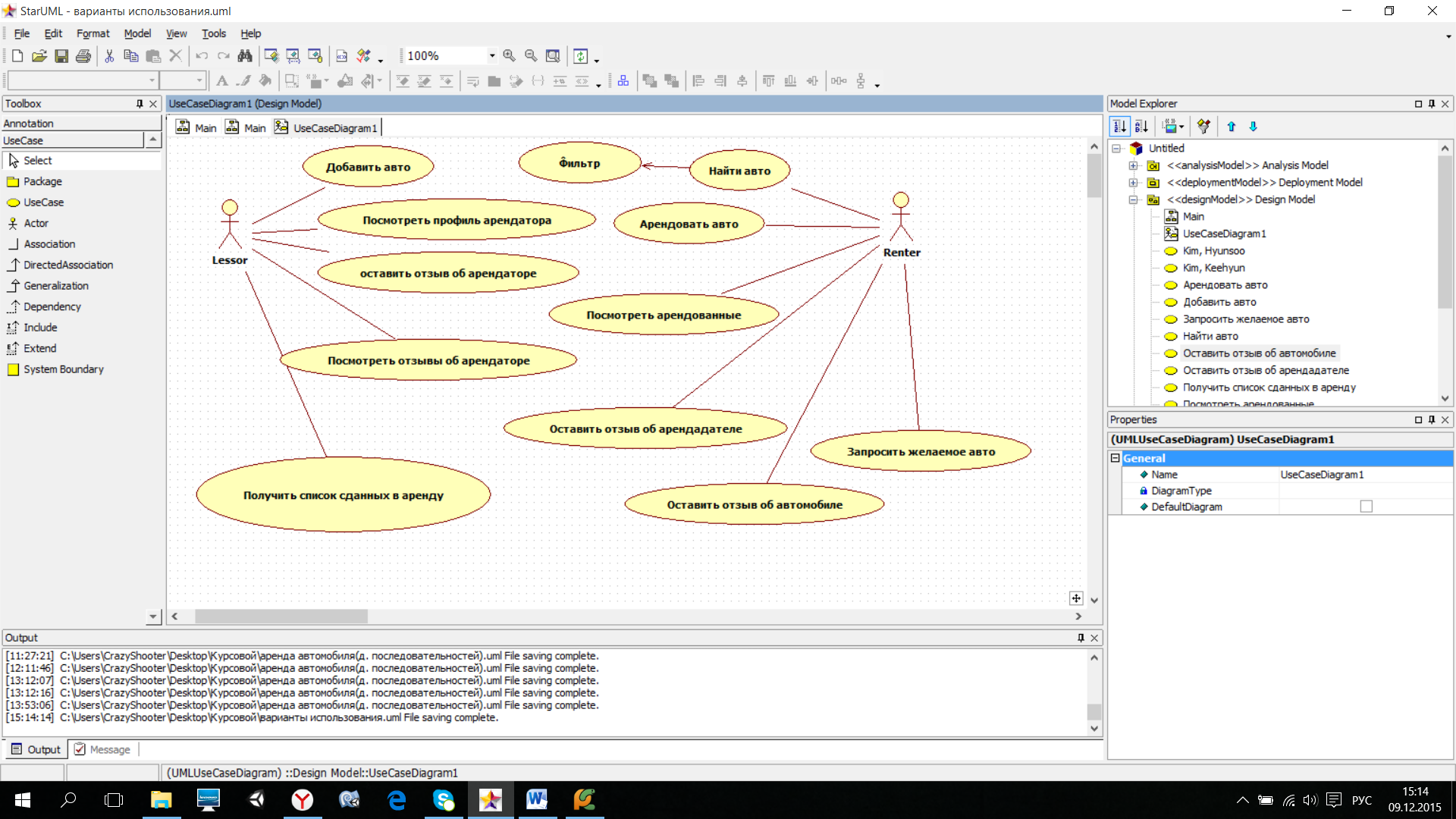


Рис. .1 Диаграмма вариантов использования программной системы

## 1.10 Детализация вариантов использования

Ниже представлены спецификации некоторых вариантов использования, изображенных на рис. 1.1 [2].

### 1.10.1 Вариант использования «Получить список сданных в аренду автомобилей»

|  |
| --- |
| Вариант использования: Получить список сданных в аренду автомобилей |
| Краткое описание:  Просмотр своих автомобилей Арендодателем, находящихся в аренде |
| Главные актеры:  Арендатор |
| Второстепенные актеры:  Арендодатель |
| Предусловие:   1. Вход в систему 2. Наличие хотя бы 1 авто в аренде |
| Основное поток:   1. Прецедент начинается после запроса у системы Арендодателем списка его автомобилей, находящихся в аренде. 2. Система проверяет существование таких авто Арендодателя, которые на данный момент находятся в аренде. 3. Система выводит Арендодателю информацию об аренде(-ах):    1. Автомобиль, находящийся в аренде    2. Дата начала аренды    3. Дата окончания аренды    4. Арендатор    5. Ссылка на страницу аренды |
| Постусловие:  Отображение системой списка автомобилей |
| Альтернативные потоки:  нет |

### 1.10.2 Вариант использования «Оставить отзыв»

|  |
| --- |
| Вариант использования: Оставить отзыв |
| Краткое описание:  Оставить отзыв об автомобиле |
| Главные актеры:  Арендодатель |
| Второстепенные актеры:  Арендатор |
| Предусловие:   1. Наличие у арендатора, заказанного в прошлом автомобиля, на который арендатор не оставил отзыв 2. Открыта информация об истёкшей аренде |
| Основное поток:   1. Арендатор выбирает опцию «оставить отзыв» 2. Система отображает диалог отзыва 3. Арендатор вводит текст отзыва 4. Арендатор выбирает оценку автомобиля по пятибалльной шкале 5. Арендатор подтверждает отзыв 6. Система сохраняет отзыв 7. Система уведомляет модератора о новом отзыве |
| Постусловие:  Отзыв сохранен |
| Альтернативные потоки:  нет |

### 1.10.3 Вариант использования «Поиск автомобиля»

|  |
| --- |
| Вариант использования: Поиск автомобилей |
| Краткое описание:  Вывод всех доступных автомобилей |
| Главные актеры:  Арендатор |
| Второстепенные актеры:  нет |
| Предусловие:  нет |
| Основное поток:   1. Вариант использования начинается, когда Арендатор запрашивает список автомобилей 2. Система принимает запрос и выводит список 3. Для каждого найденного автомобиля система выводит    1. Миниатюрное представление автомобиля    2. Краткое описание автомобиля       1. статус занятости       2. год выпуска       3. марка       4. модель       5. цвет   Точка расширения: Фильтрация автомобилей |
| Постусловие:  Выведен список доступных автомобилей |
| Альтернативные потоки:  нет |

### 1.10.4 Вариант использования «Фильтрация автомобиля»

|  |
| --- |
| Расширяющий вариант использования: Фильтрация автомобилей |
| Краткое описание:  Система отфильтровывает список автомобилей в соответствии с заданными параметрами и выводит их на экран |
| Главные актеры:  Арендатор |
| Второстепенные актеры:  Нет |
| Предусловие:  Открыт диалог поиска автомобилей |
| Основное поток:   1. Арендатор выбирает опцию «Фильтрация автомобилей» 2. Система запрашивает у Арендатора критерий поиска (год выпуска, модель) 3. Арендатор вводит запрашиваемый критерий 4. Система ищет авто, соответствующие критерию Арендатора 5. Если система находит соответствующие авто, тогда    1. Для каждого найденного авто система выводит       1. миниатюрное представление авто       2. краткое описание продукта (статус занятости, год выпуска, марка, модель)   6. Иначе  6.1. Система сообщает Арендатору о том, что соответствующие авто не найдены |
| Постусловие:  Выведен список отфильтрованных автомобилей |
| Альтернативные потоки:  Нет |

### 1.10.5 Вариант использования «Добавление автомобиля»

|  |
| --- |
| Вариант использования «Добавление автомобиля» |
| Краткое описание:  Система позволяет автовладельцам добавить автомобиль для аренды. |
| Главные актеры:  Арендодатель |
| Второстепенные актеры:  Арендатор |
| Предусловия:  1. Клиент должен иметь при себе действующие на момент выдачи автомобиля водительское удостоверение и паспорт. Водительский стаж должен составлять не менее двух лет.  2. Залоговая сумма и цена проката за автомобиль в течение срока проката зависит от класса выбранного автомобиля.  3. В случае повреждения или кражи автомобиля ответственность клиента ограничивается Франшизой. |
| Основной поток:   1. Прецедент начинается, когда Арендодатель выбирает опцию «добавить авто». 2. Если система запрашивает данные об автомобиле, тогда 3. Для каждого автомобиля 4. Система показывает марку, модель, год выпуска, сроки аренды. 5. Система добавляет данные об авто |
| Постусловия:  Нет. |
| Альтернативные потоки:  Нет. |

### 1.10.6 Вариант использования «Посмотреть арендованные»

|  |
| --- |
| Вариант использования «Посмотреть арендованные» |
| Краткое описание:  Система позволяет арендатором посмотреть арендованные автомобили. |
| Главные актеры:  Арендатор |
| Второстепенные актеры:  Нет |
| Предусловия:  Нет |
| Основной поток:   1. Прецедент начинается, когда Арендатор выбирает опцию «посмотреть арендованные». 2. Если Арендатор хочет посмотреть информацию об автомобиле, тогда для каждого арендованного автомобиля выводится:    1. марку,    2. модель,    3. год выпуска,    4. сроки аренды. |
| Постусловия:  Нет. |
| Альтернативные потоки:  Нет. |

### 1.10.7 Вариант использования «Оставить заявку на желаемое авто»

|  |
| --- |
| Прецедент: Оставить заявку на желаемое авто. |
| Краткое описание:  Система позволяет пользователю оставить заявку на добавление необходимого автомобиля в систему проката, в случае его отсутствия. |
| Главные актеры:  Пользователь. |
| Второстепенные актеры:  Нет. |
| Предусловия:   1. Пользователь вошел в систему. |
| Основной поток:   1. Прецедент начинается, когда пользователь не находит нужного автомобиля. 2. Система запрашивает у пользователя: год выпуска, марку, модель, цвет и соглашение на обработку персональных данных. 3. Пользователь вводит запрашиваемые данные |
| Постусловия:  Заявка создана. |
| Альтернативные потоки:   1. Данные введены корректно, система выводит на экран сообщение о том, что все прошло успешно, заявка принята, заявка сохраняется. 2. Система требует исправить или дополнить введенные данные, далее повторяются пункты 3 (основного потока) и 1 (альтернативного) пока все данные не будут введены верно, или пользователь не отменит действие. |

### Вариант использования «Арендовать машину»

|  |
| --- |
| Вариант использования: Арендовать машину |
| Краткое описание:  Система арендует автомобиль в соответствии с заданными параметрами и выводит их на экран. |
| Главные актеры:  Арендатор |
| Второстепенные актеры:  Арендодатель |
| Предусловие:  Выбор машины, переход на страницу автомобиля |
| Основное поток:   1. Арендатор выбирает даты начала и конца аренды 2. Система проверяет статус машины на эти даты 3. Если система подтверждает корректность выбранных параметров, то   3.1. Система отображает общую стоимость сделки  3.2. Система арендует автомобиль на выбранные даты  4. Иначе  4.1 Система сообщает Арендатору о том, что данные введены некорректно. |
| Постусловие:  Система резервирует машину на эти даты. Оплата ренты |
| Альтернативные потоки:  Нет |

# 2 АНАЛИЗ

При разработке программного обеспечения одной из основных методологий является анализ. На данном этапе создается аналитическая модель предметной области. Создается она путем выявления классов и отношений между ними. Важным моментом является то, что необходимо ограничиваться лишь теми классами, которые являются частью предметной области. Если мы будем придерживаться данной стратегии, то это поможет нам выполнить каждый вариант использования простым и понятным.

## 2.1 Диаграмма классов анализа

Классы анализа представляют собой абстракции предметной области, которые ни коем образом не относятся к области решения.

На рис. 2.1 представлена диаграмма классов анализа для разрабатываемого приложения.

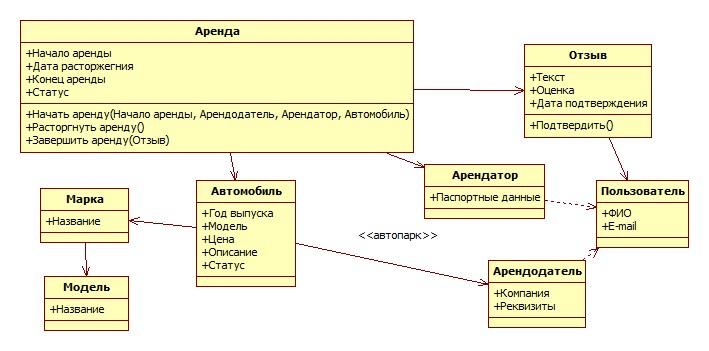


Рис. .1 Диаграмма классов анализа

Из рисунка видно, что классы анализа содержат в себе только базисные классы, так называемые классы сущностей, их атрибуты, а также отношения между самими классами.

Основным классом является «Пользователь», который содержит 2 поля, необходимых для идентификации: ФИО и E-mail. «Пользователь» может принимать 2 роли. Это реализуется в классах «Арендодатель» и «Арендатор». Класс «Арендодатель» содержит поля, необходимые для привязки пользователя к определённой компании: Компания и Реквизиты. Класс «Арендатор» содержит единственное поле «Паспортные данные».

У каждого «Арендодателя» есть автопарк, который описывается классом «Автомобиль». Класс содержит поля, которые необходимы для описания автомобиля: Год выпуска, Наименование, Цена, Описание, Статус. Поле «Наименование» составляется путём получения названий из классов «Марка» и «Модель». Поле «Статус» определяет, возможна ли аренда автомобиля, или же он уже находится в аренде.

Класс «Аренда» описывает основную функцию системы – аренда автомобиля. Класс обладает атрибутами: Начало аренды, Дата расторжения, Конец аренды, Статус. Все атрибуты, кроме «Статус» принадлежат к типу Дата. «Статус» определяет, в каком состоянии находится аренда: в режиме ожидания, началась или уже закончилась. Класс «Аренда» обладает процедурами: Начать аренду, Расторгнуть аренду, Завершить аренду. Метод «Начать аренду» принимает аргументы, необходимые для создания самого факта аренды: Начало аренды, Арендодатель, Арендатор, Автомобиль. Метод «Завершить аренду» принимает аргумент «Отзыв».

Класс «Отзыв» необходим для того чтобы охарактеризовать «Пользователя» другим «Пользователем». После завершения аренды «Арендодатель» может оставить отзыв об «Арендаторе», где приводит его удовлетворение или недовольство от выполнения обязательствами аренды «Арендатором». «Арендатор» может в своём отзыве дать оценку аренде, поведению «Арендодателя». Класс «Отзыв» содержит поля: Текст, Оценка, Дата подтверждения. Так же, класс обладает методом «Подтвердить»: модератор или другой привилегированный пользователь подтверждает отзыв. «Дата подтверждения» – дата, когда привилегированный пользователь подтвердил корректность «Отзыва».



## Диаграмма состояний

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Очевидно, что диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и, как мы увидим далее, диаграммы деятельности). Часто можно услышать, что диаграмма состояний показывает автомат, но об этом мы поговорим подробнее чуть позже. Диаграмма состояний полезна при моделировании жизненного цикла объекта.

От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса – одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события. Понятие жизненного цикла применимо как раз к реактивным объектам, настоящее состояние (и поведение) которых обусловлено их прошлым состоянием. Но диаграммы состояний важны не только для описания динамики отдельного объекта. Они могут использоваться для конструирования исполняемых систем путем прямого и обратного проектирования [4].

Классом, изменяющимся динамически, в данной системе являются классы «Аренда» и «Автомобиль», экземпляры которых изменяют свое состояние в процессе своего существования в данной системе.

На рис. 2.2 представлена диаграмма конечных автоматов для класса «Аренда» данной системы.

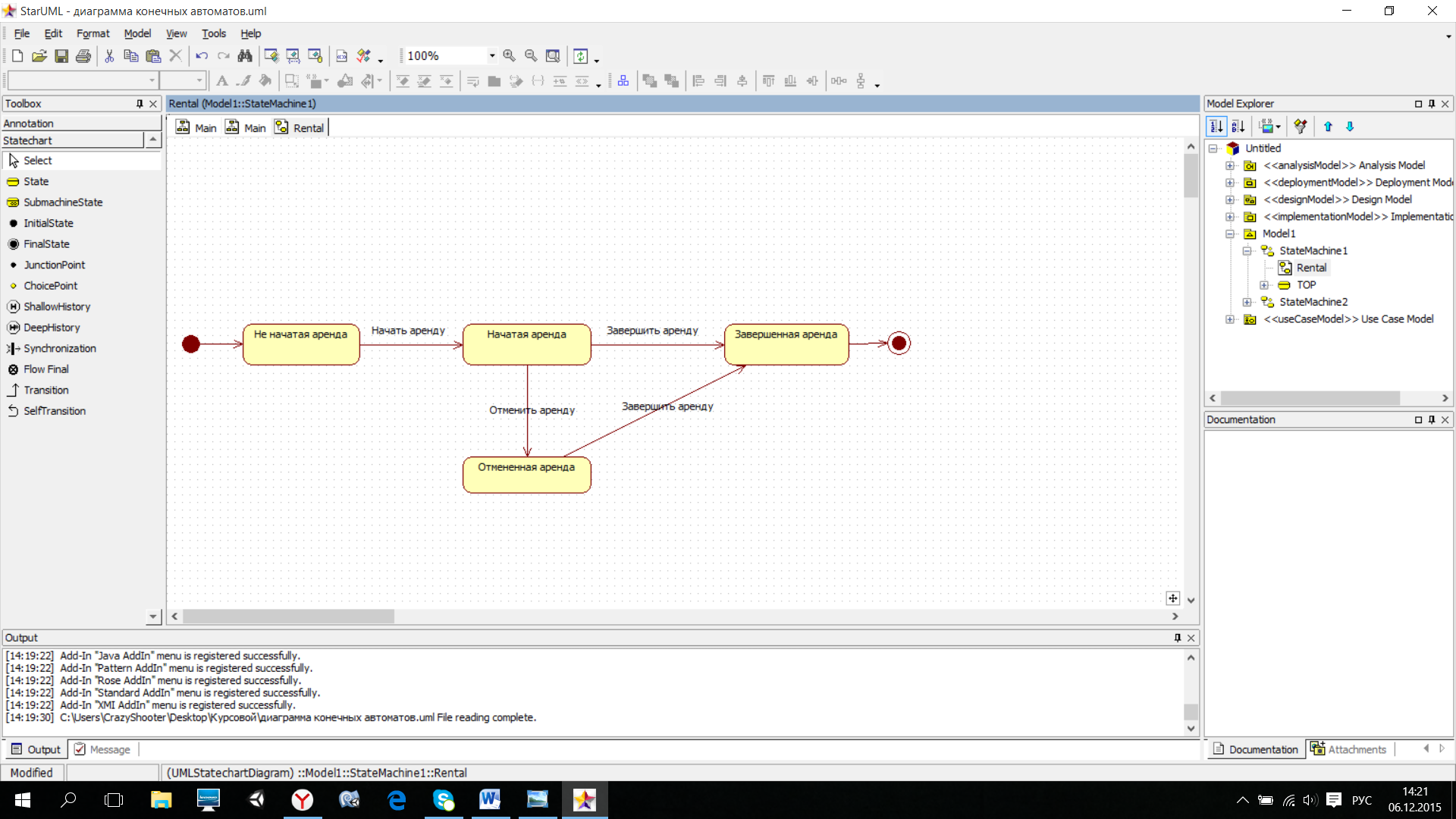


Рис. 2.2 Диаграмма конечных автоматов класса «Аренда»

Как видно на рис. 2.2, класс «Аренда» может находиться в 4-х состояниях: не начатая, начатая, завершенная и отмененная.

При оформлении аренды происходит создание экземпляра класса «Аренда», вызывается функция оформления аренды, после оформления которой «Аренда» автоматически переходит в состояние не начатой.

После наступления даты начала аренды она переходит в состояние «Начатая аренда».

Из состояния «Начатая» аренду может перевести в состояние «Отмененная» пользователь, который её начал, до её начала.

Из состояния «Начатая» заказ может перевести в состояние «Отмененная» переводит арендодатель после окончания аренды и возвращения автомобиля.

На рис. 2.3 представлена диаграмма конечных автоматов для класса «Автомобиль» данной системы.

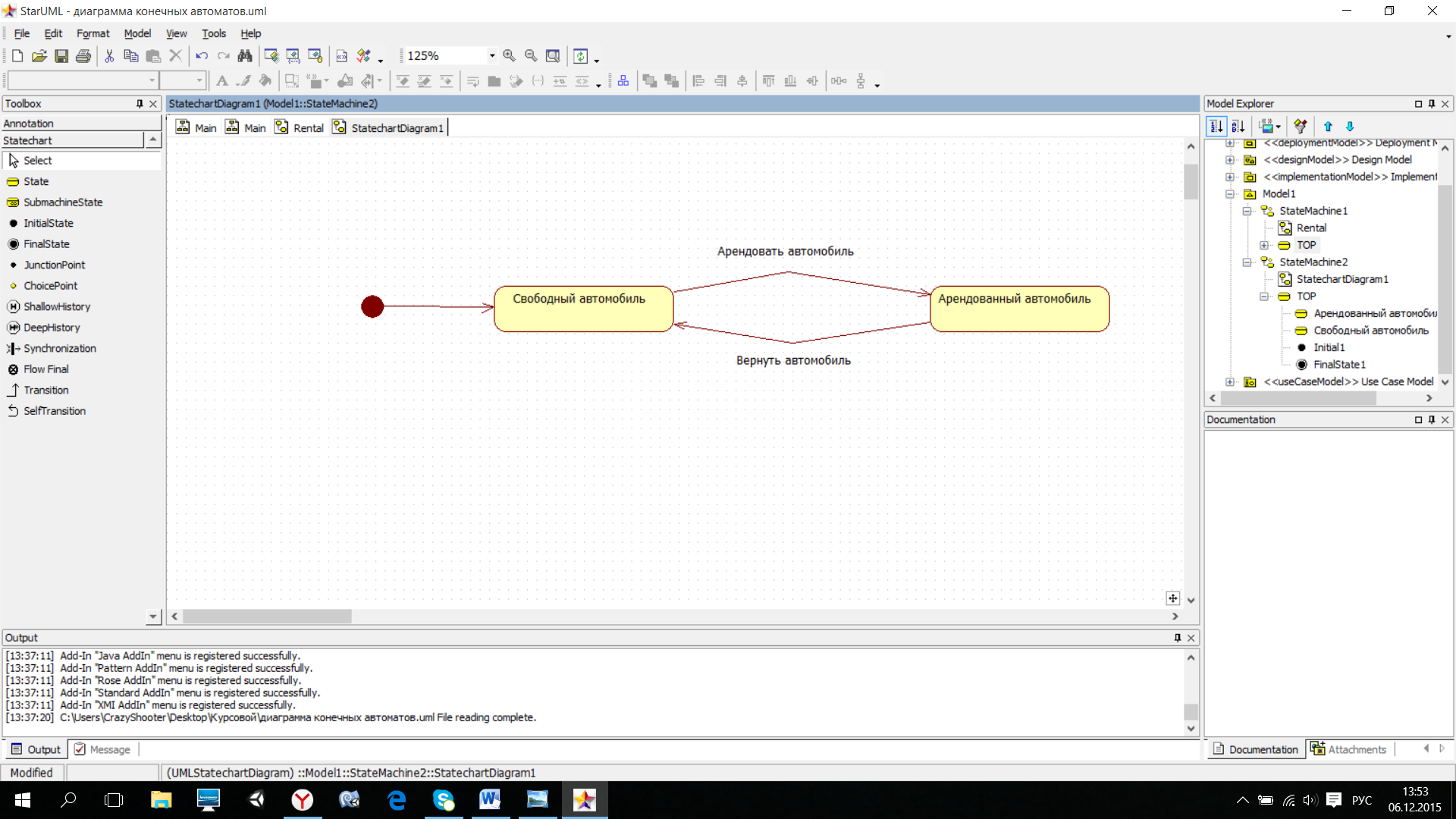


Рис. 2.3 Диаграмма конечных автоматов класса «Автомобиль»

Как видно на рис. 2.3, класс «Автомобиль» может находиться в 2-х состояниях: свободный и арендованный.

При добавлении экземпляра класса «Автомобиль», автомобиль автоматически переходит в состояние «Свободный автомобиль».

Из состояния «Свободный автомобиль» автомобиль переходит в состояние «Арендованный автомобиль» автоматически после начала аренды данного авто.

После завершения аренды автомобиль автоматически переходит из состояния «Арендованный автомобиль» в «Свободный автомобиль».

Созданные на данном этапе диаграммы состояний подготовили основу для проектирования системы, ориентируясь на выбранные языки программирования, платформы и средства разработки. Данный этап является одним из самых важных.

# 3 ПРОЕКТИРОВАНИЕ

На этапе проектирования, опираясь на ранее написанные аналитическую и логическую модели, создается новая модель, которая будет охватывать последующую реализацию. Проектная модель является более детализированной и должна учитывать конкретные технические решения.

## 3.1 Проектные классы

Проектные классы – это классы, описание которых настолько полно, что они могут быть реализованы [3, с. 372]. В отличие от классов анализа, описание содержит детали реализации. Диаграмма проектных классов представлена на рис. 3.1.

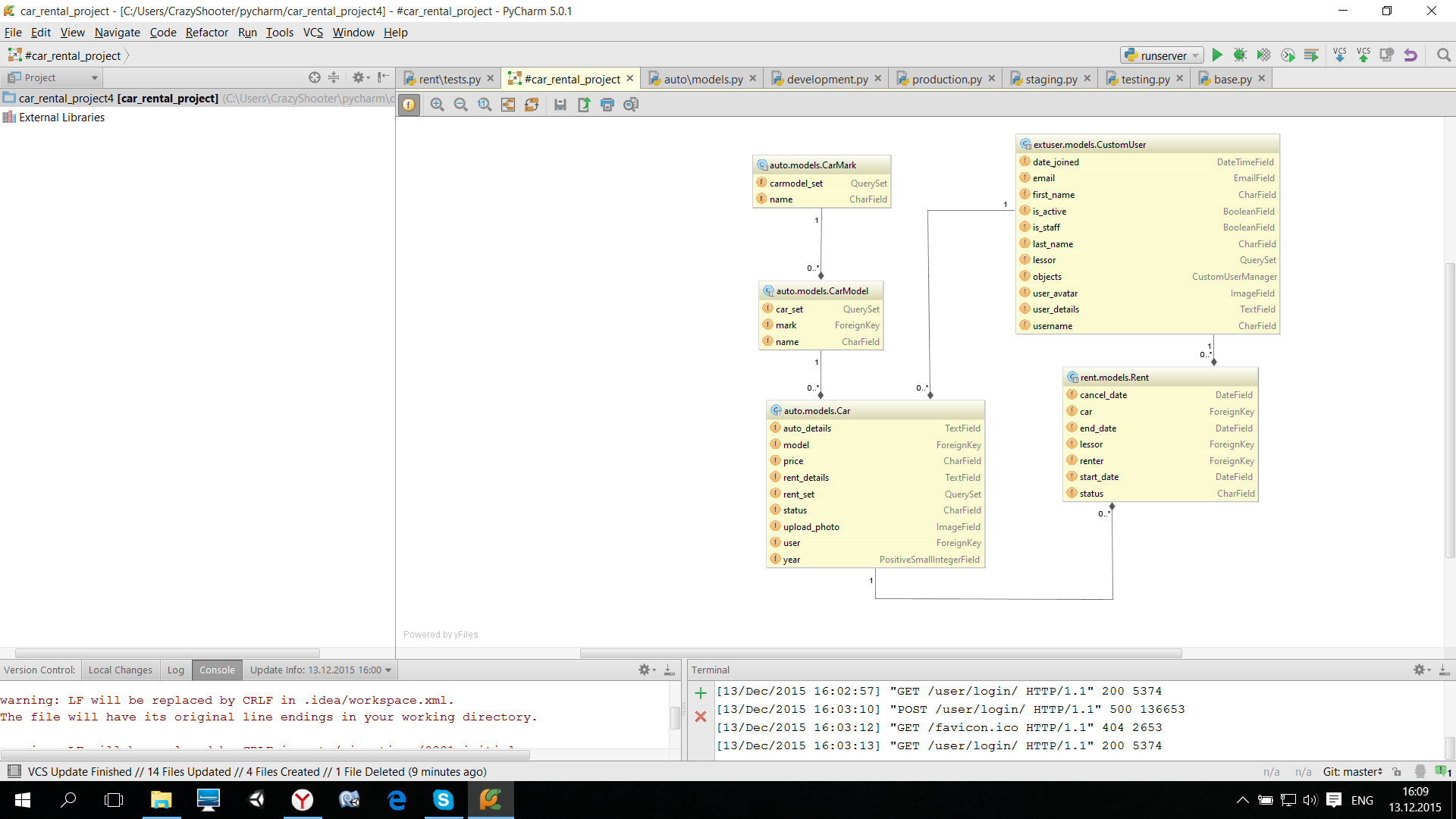


Рис. 3.1 Диаграмма проектных классов

По сравнению с диаграммой классов анализа произошли следующие изменения: класс «Отзыв» был опущен до следующей модификации проекта, классы «Арендатор» и «Арендодатель» были объединены в один класс «Пользователь». В класс «Автомобиль» были добавлены атрибуты: «Цена», «Подробности о машине», «Подробности аренды», «Изображение машины», «Статус». В класс «Аренда» был добавлен атрибут «Статус».

## 3.2 Диаграмма пакетов системы

Пакет – общецелевой механизм для организации различных элементов модели во множество, реализующий системный принцип декомпозиции модели сложной системы и допускающий вложенность пакетов друг в друга.

Пакет – основной способ организации элементов модели в языке UML. Каждый пакет владеет всеми своими элементами, т.е. теми элементами, которые включены в него. Про соответствующие элементы пакета говорят, что они принадлежат пакету или входят в него. При этом каждый элемент может принадлежать только одному пакету. В свою очередь, одни пакеты могут быть вложены в другие.

Подпакет – пакет, который является составной частью другого пакета.

По определению все элементы подпакета принадлежат и более общему пакету. Тем самым для элементов модели задается отношение вложенности пакетов, которое представляет собой иерархию.

Для графического изображения пакетов на диаграммах применяется специальный графический символ – большой прямоугольник с небольшим прямоугольником, присоединенным к левой части верхней стороны первого. Можно сказать, что визуально символ пакета напоминает пиктограмму папки в популярном графическом интерфейсе. Внутри большого прямоугольника может записываться информация, относящаяся к данному пакету. Если такой информации нет, то внутри большого прямоугольника записывается имя пакета, которое должно быть уникальным в пределах рассматриваемой модели. Если же такая информация имеется, то имя пакета записывается в верхнем маленьком прямоугольнике [4].

Диаграмма пакетов представлена на рис. 3.2. Она состоит из пакетов, содержащих в себе классы для формирования интерфейса пользователя.

Проект «Car\_rental» содержит в себе следующие пакеты: «User», «Auto», «Rent» и «Feedback». Пакет «User» состоит из класса «User» и от этого пакета зависят «Rent» , «Auto» и «Feedback». Пакет «Rent» состоит только из класса «Rent» и зависит только от пакета «Feedback», состоящего из одноименного класса «Feedback». Пакет «Auto» состоит из классов «Car», «CarModel» и «CarMark» и от этого пакета зависит пакет «Rent».

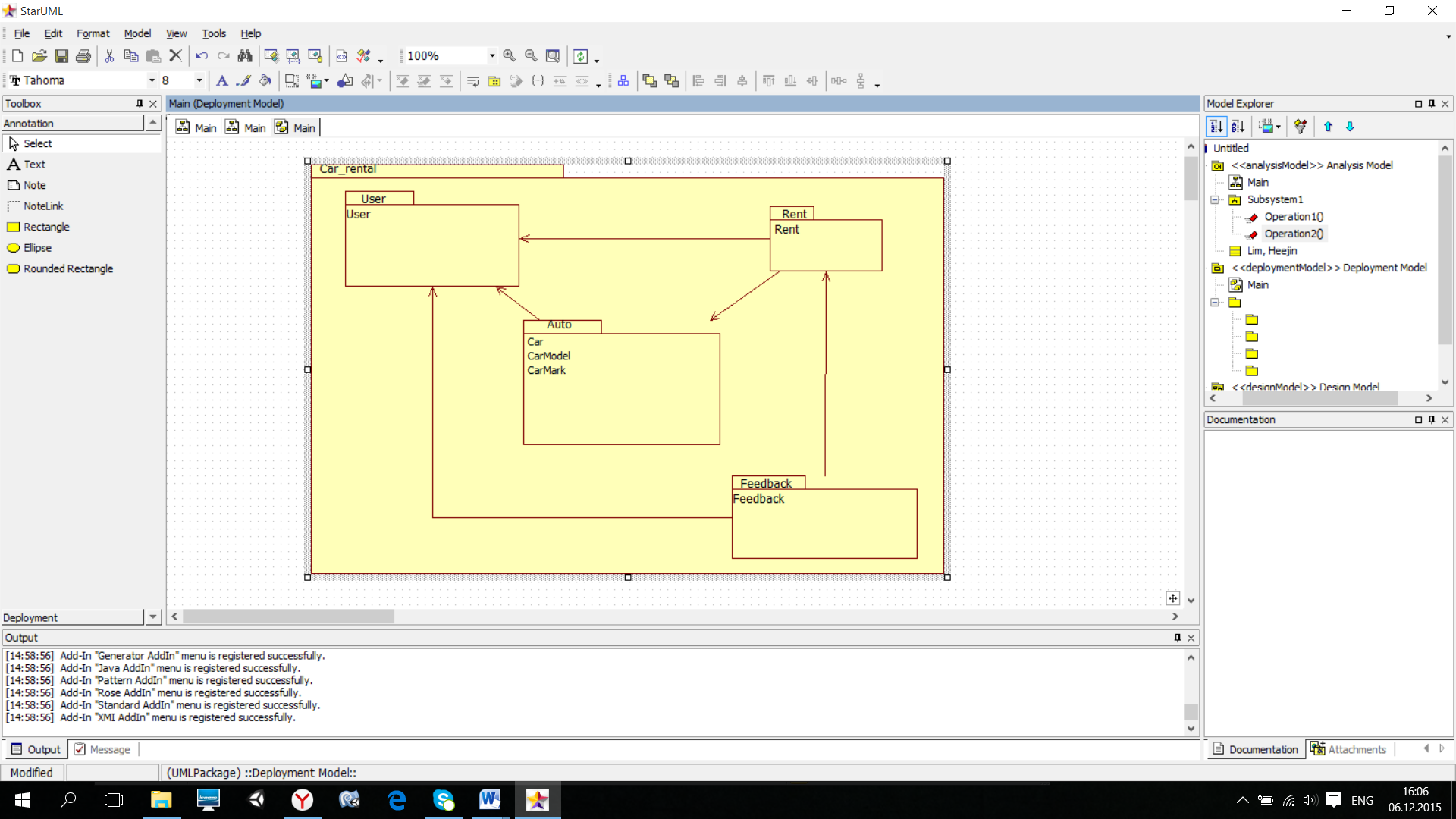


Рис. 3.2 Диаграмма пакетов программной системы

Пакет «User» содержит класс «User», который может просматривать, создавать или отменять аренды, так же он может добавлять авто. Пакет «Auto» содержит списки автомобилей с подробной информацией о них. Пакет «Rent» содержит информацию об арендах. Пакет «Feedback» содержит отзывы пользователей об аренде.

## 3.3 Диаграммы последовательностей для операций проектных классов

Диаграмма последовательностей отображает взаимодействие объектов в динамике.

В UML взаимодействие объектов понимается как обмен информацией между ними. При этом информация принимает вид сообщений. Кроме того, что сообщение несет какую-то информацию, оно некоторым образом также влияет на получателя. В этом плане UML полностью соответствует основным принципам ООП, в соответствии с которыми информационное взаимодействие между объектами сводится к отправке и приему сообщений.

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами.

Диаграммы последовательностей можно использовать для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования! Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста [4].

Были спроектированы диаграммы последовательностей для нескольких вариантов использования. На рис. 3.3 представлена диаграмма последовательности для операции проектного класса «Добавить аренду».

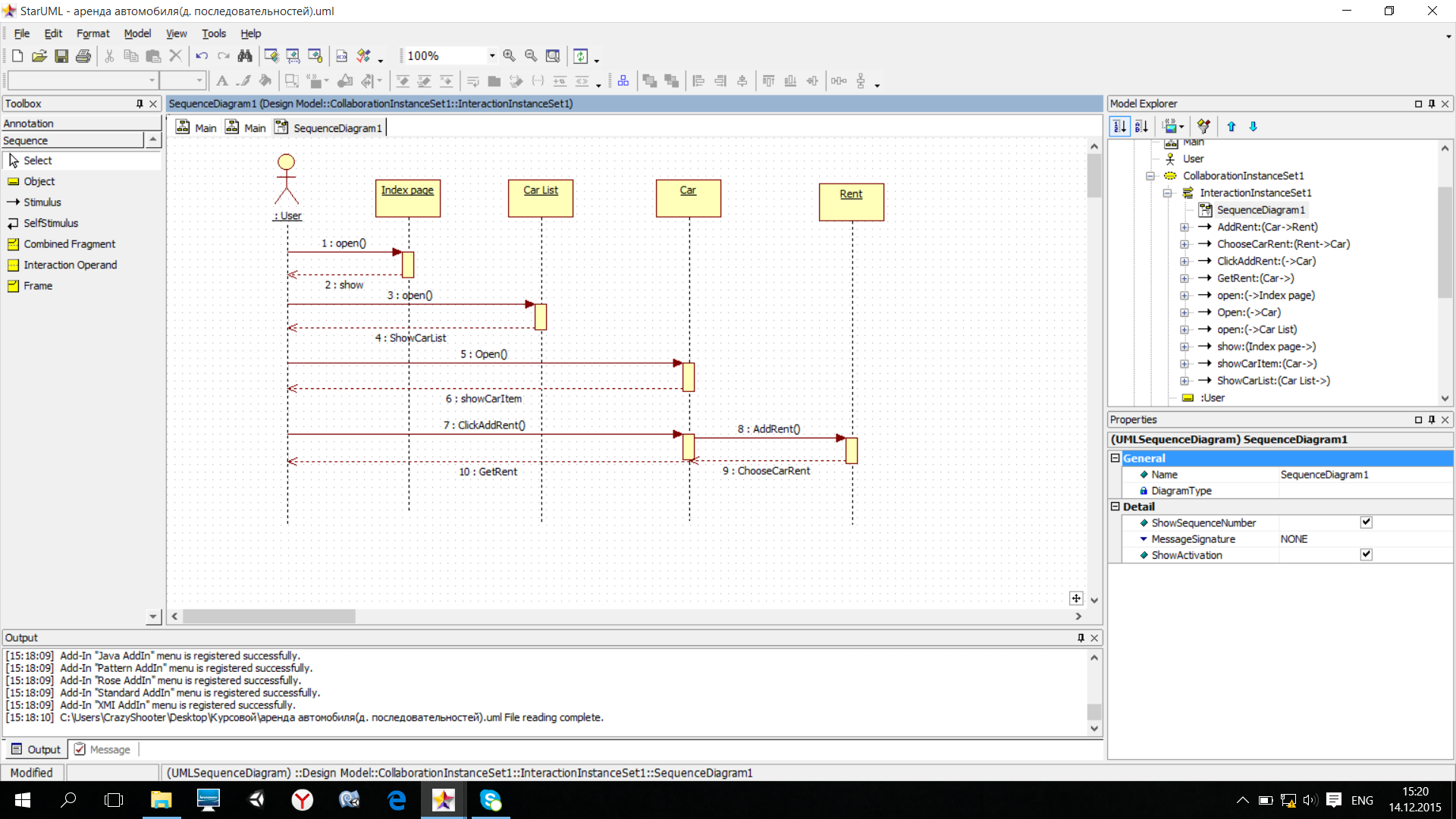


Рис. 3.3 Диаграмма последовательностей для операции проектного класса «Добавить аренду»

Для операции проектного класса «Добавить аренду» пользователю необходимо открыть главную страницу веб-приложения, с которой мы переходим на страницу каталога, выбрать понравившуюся машину, которая перенаправит его на страничку, непосредственно отображающую информацию об авто, где можно нажать кнопку арендовать авто. После нажатия на кнопку выведется сообщение об успешно начатой аренде. Кнопка «Арендовать», располагается под описанием машины на странице.

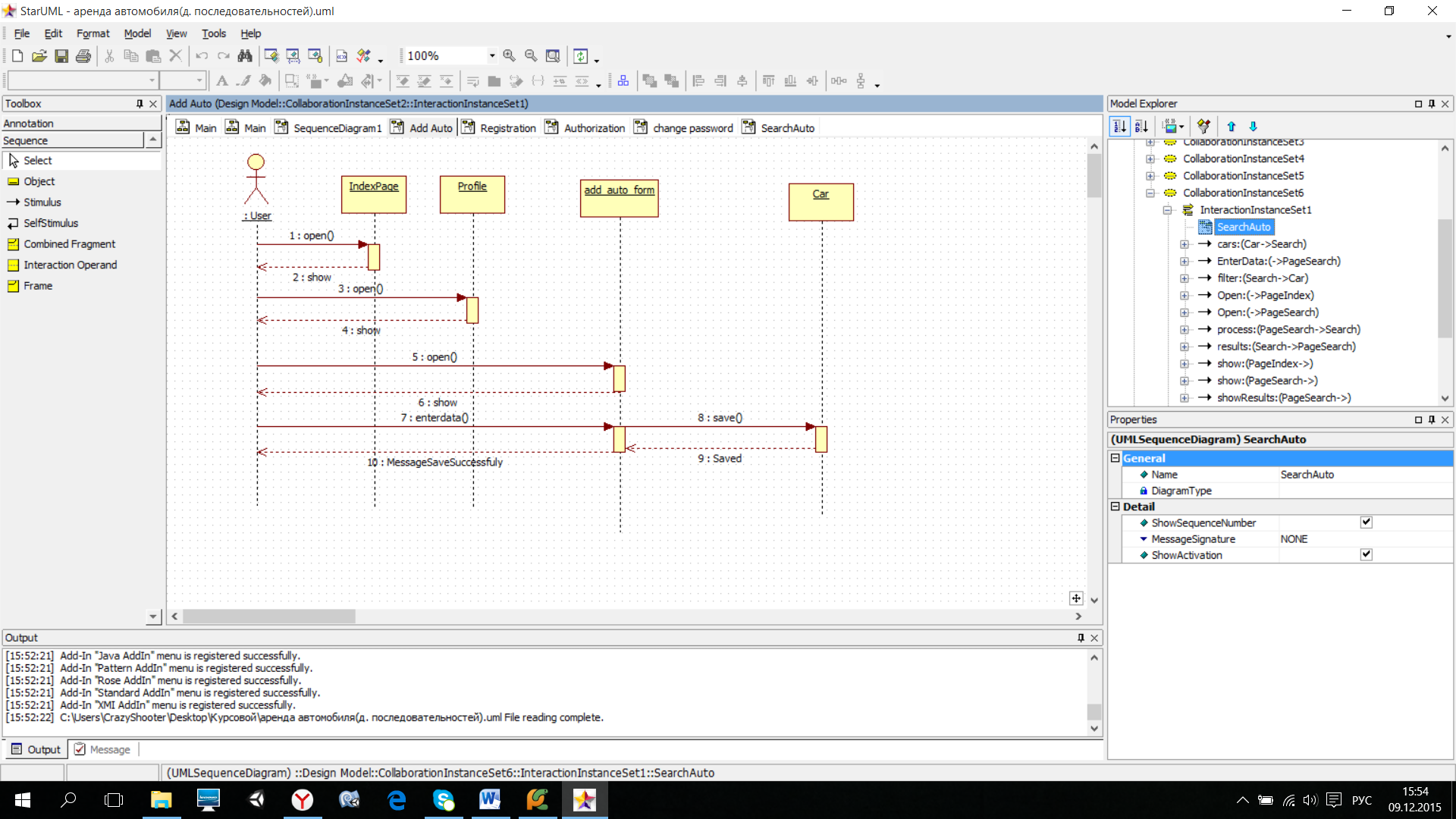


Рис. 3.4 Диаграмма последовательностей для операции проектного класса «Добавление автомобиля»

Сценарий описываемой последовательности сообщений между объектами программной системы начинается с того момента, когда авторизированный пользователь зашел на главную страницу веб-приложения, после которой он переходит в свой профиль, там он выбирает пункт «Добавить авто», вводит соответствующие данные и сохраняет их. После чего выводится сообщение об успешном добавлении нового автомобиля.

На рис. 3.5 изображена диаграмма последовательности для операции проектного класса «Авторизация пользователя».

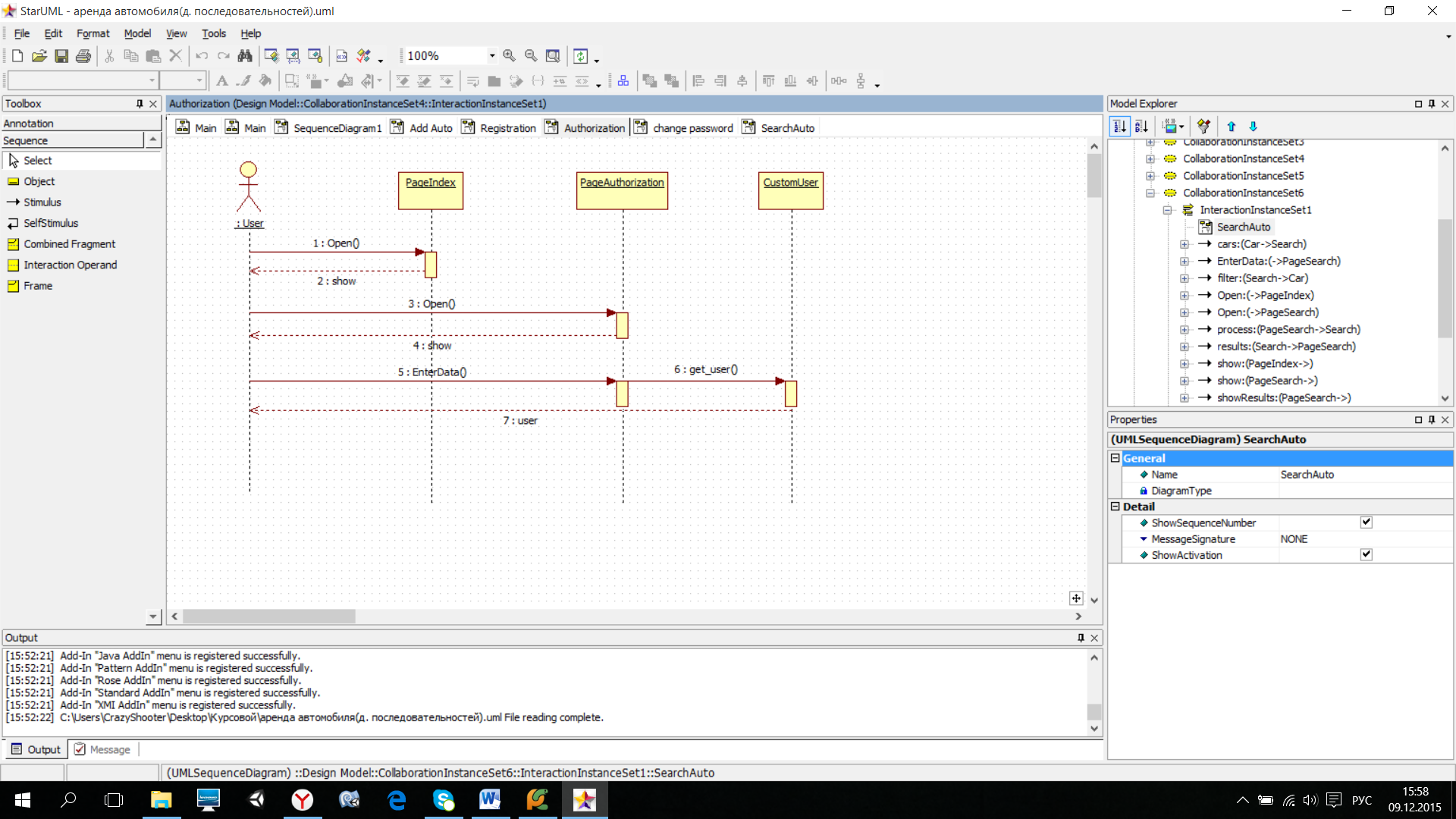


Рис. 3.5 Диаграмма последовательностей для операции проектного класса «Авторизация пользователя»

Для операции проектного класса «Авторизация пользователя» пользователю необходимо открыть главную страницу веб-приложения, на которой нужно нажать на кнопку авторизации. После вывода формы авторизации, нам необходимо заполнить необходимые поля. Если данные были введены корректно, вы войдем в систему. После авторизации имя нашего аккаунта отобразится в нижнем левом углу и появится кнопка выхода из учетной записи.

На рис. 3.6 изображена диаграмма последовательности для варианта использования «Смена пароля».



Рис. 3.6 Диаграмма последовательностей для операции проектного класса «Смена пароля»

Для операции проектного класса «Смена пароля» пользователю необходимо открыть страницу своего профиля, на которой есть кнопка смены пароля. Для смены пароля необходимо ввести старый пароль и два раза новый. После удачной смены пароля выведется сообщение об успешной операции.

На рис. 3.7 изображена диаграмма последовательности для варианта использования «Поиск авто».

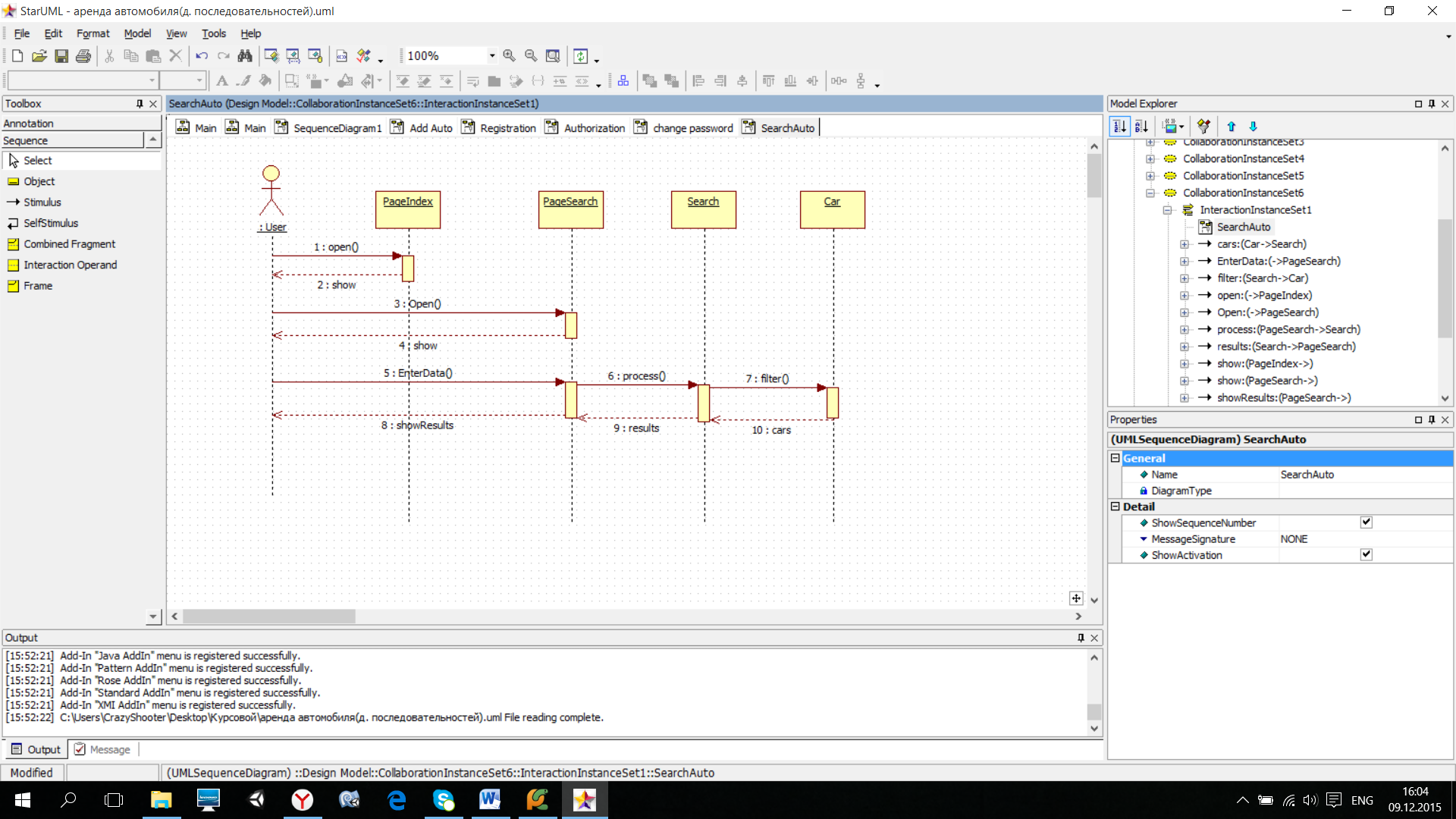


Рис. 3.7 Диаграмма последовательностей для операции проектного класса «Поиск авто»

Для операции проектного класса «Поиск авто» необходимо пользователю открыть страницу поиска авто с главной страницы. После ввода параметров поиска и нажатия на соответствующую кнопку, нам выведется список в соответствии с заданными параметрами.

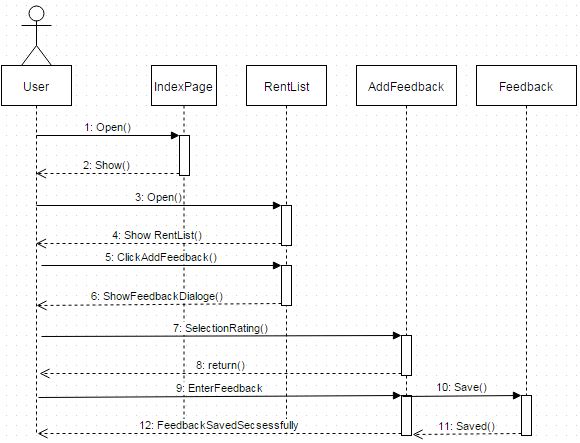


Рис. 3.8 Диаграмма последовательностей для варианта использования «Оставить отзыв»

Для варианта использования «Оставить отзыв» необходимо пользователю открыть страницу арендованных авто с главной страницы. После ввода оценки, текстового отзыва и нажатия на соответствующую кнопку, отзыв сохраняется и становится доступен для просмотра другим пользователям.

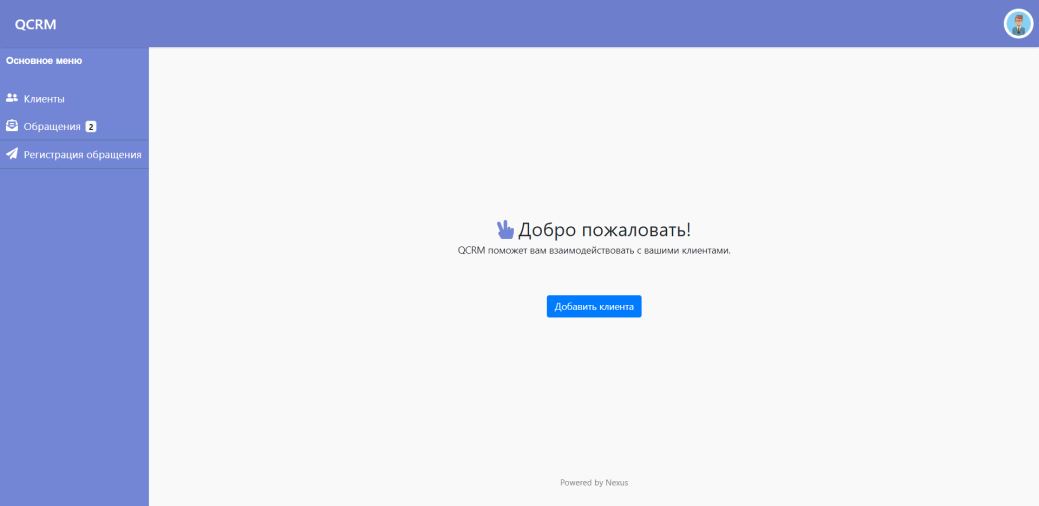
# 4 РЕАЛИЗАЦИЯ

## 4.1 Разработка приложения

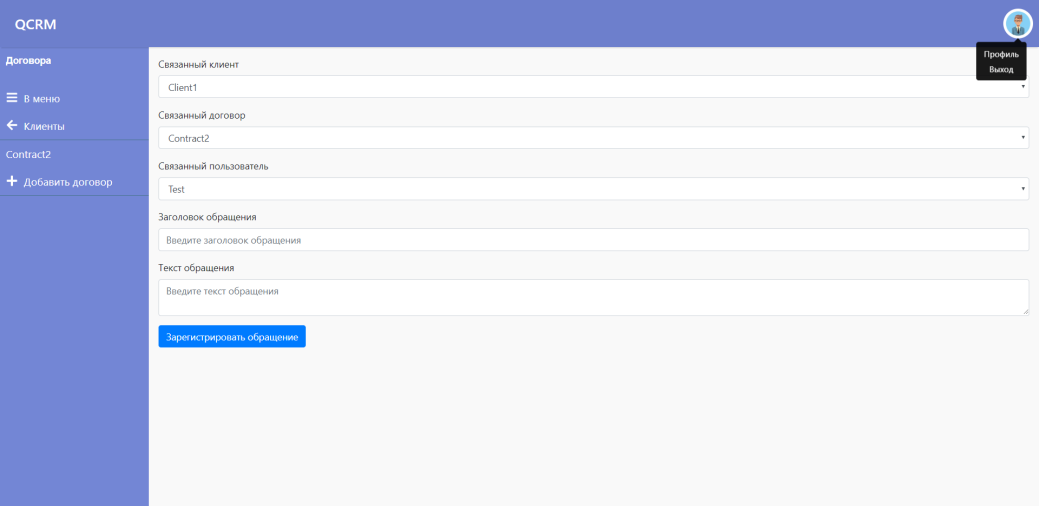
Некоторые части интерфейса приложения представлены на рисунках:



Страница аутентификации



Стартовая страница приложения



Форма регистрации обращения

## 4.2 Тестирование

Тестирование проводится с целью обеспечить качество разрабатываемого программного продукта. Стандарт ISO-8402, посвященный описанию систем обеспечения качества программного обеспечения, под качеством понимает "совокупность характеристик программного продукта, относящихся к его способности удовлетворять установленные и предполагаемые потребности клиента". Основным параметром качества программы является надёжность. Надёжность определяется как вероятность его работы без отказов в течении определённого периода времени, рассчитанная с учётом стоимости для пользователя каждого отказа. Отказ программного обеспечения – это проявление ошибки в нём. Отсюда тестирование ПО – это процесс выполнения программы с целью обнаружения в ней ошибок. "Удачным" тестом является такой, на котором выполнение программы завершилось с ошибкой. Напротив, "неудачным" называется тест, не позволивший выявить ошибку в программе. Основные принципы организации тестирования:

1. Необходимой частью каждого теста должно являться описание ожидаемых результатов работы программы;
2. Программе не должна тестироваться её автором;
3. Организация – разработчик программного обеспечения не должна "единолично " его тестировать;
4. Необходимо подбирать тесты не только для правильных (предусмотренных) входных данных, но и для неправильных (непредусмотренных);
5. При анализе результатов каждого теста необходимо проверять, не делает ли программа того, что она не должна делать;
6. "Принцип скопления ошибок" – вероятность наличия не обнаруженных ошибок в некоторой части программы прямо пропорциональна числу ошибок, уже обнаруженных в этой части;

Процесс тестирования состоит из трёх этапов:

1. Проектирование тестов.

1. Исполнение тестов.
2. Анализ полученных результатов. [5]

### 4.2.1 Модульные тесты

Каждая сложная программная система состоит из отдельных частей – модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе системы в целом, необходимо вначале протестировать каждый модуль системы в отдельности. В случае возникновения проблем это позволит проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них. Такое тестирование модулей по отдельности получило называние модульного тестирования (unit testing).

Для каждого модуля, подвергаемого тестированию, разрабатывается тестовое окружение, включающее в себя драйвер и заглушки, готовятся тест-требования и тест-планы, описывающие конкретные тестовые примеры.

Основная цель модульного тестирования – удостовериться в соответствии требованиям каждого отдельного модуля системы перед тем, как будет произведена его интеграция в состав системы. Для создания unit-теста используется класс TestCase, от которого наследуются все созданные разработчиком классы с тестами.

В силу того, что модули, подвергаемые тестированию, обычно невелики по размеру, модульное тестирование считается наиболее простым (хотя и достаточно трудоемким) этапом тестирования системы. Однако, несмотря на внешнюю простоту, с модульным тестированием связано две проблемы. [6]

1. Проверка создания объекта аренды

Class RentTest(TestCase):  
  
 def test\_rent\_created(self):  
 user1 = CustomUser.objects.create(  
 password='123',  
 email='blabla@mail.ru',  
 )  
 mark1 = CarMark.objects.create(name='Toyota')  
 model = CarModel.objects.create(name='testmodel', mark=mark1)  
 car = Car.objects.create(  
 price=23.00,  
 model=model,  
 year=1990,  
 user=user1,  
 )  
 rent = Rent.objects.create(  
 renter=user1,  
 car=car,  
 end\_date=date(2015, 12, 30),  
 )  
 self.assertEqual(rent, Rent.objects.get(renter=user1, car=car))

Данный тест проверяет возможность создания аренды. В качестве обязательных аргументов аренда принимает, арендатора и машину.

2. Проверка свойства, задающего дату истечения заказа:

**def** test\_car\_belongs\_to\_user(self):  
 user1 = CustomUser.objects.create(  
 password=**'123'**,  
 email=**'blabla@mail.ru'**,  
 )  
 mark1 = CarMark.objects.create(name=**'Toyota'**)  
 model = CarModel.objects.create(name=**'testmodel'**, mark=mark1)  
 car = Car.objects.create(  
 price=23.00,  
 model=model,  
 year=1990,  
 user=user1,  
 )  
 self.assertEqual(car, Car.objects.get(user=user1))

Данный тест проверяет, принадлежит ли машина объекту юзер.

3.Проверка статуса аренды:

**def** test\_rent\_status(self):  
 user1 = CustomUser.objects.create(  
 password=**'123'**,  
 email=**'blabla@mail.ru'**,  
 )  
 mark1 = CarMark.objects.create(name=**'Toyota'**)  
 model = CarModel.objects.create(name=**'testmodel'**, mark=mark1)  
 car = Car.objects.create(  
 price=23.00,  
 model=model,  
 year=1990,  
 user=user1,  
 )  
 rent = Rent.objects.create(  
 renter=user1,  
 car=car,  
 end\_date=date(2015, 12, 30),  
 )  
 self.assertEqual(rent.status, Rent.STATUS\_UNSET)

В ходе выполнения данного теста проверяется поле status аренды. Если оно равно Rent.STATUS\_UNSET, то метод работает правильно.

1. Проверка получения названия модели:

**def** test\_model\_str(self):  
 mark = CarMark.objects.create(name=**'supermark'**)  
 model = CarModel.objects.create(name=**'supermodel'**, mark=mark)  
 self.assertEqual(  
 model.\_\_str\_\_(),  
 **'supermodel'**, **"Incorrect name of model"** )

Если названия марки и модели совпадают с указанными, то метод работает правильно, иначе выводится сообщение об ошибке.

### 4.2.2 Интеграционные тесты

Интеграционное тестирование – одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию.

Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования.

Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц – объединения, множества или группы модулей – выполняется через их интерфейс, с использованием тестирования «чёрного ящика» [7].

### 4.2.3 Построение и выполнение тестов

Выполнение всех тестов для курсового проекта необходимо использовать утилиту manage.py test. Чтобы обратиться к какому – то определенному приложению или проверки теста, необходимо использовать python manage.py test <application\_name>.

Хочется отметить, что в тестах Django версии 1.7 и выше – это то, что они не используют настоящую базу данных. Для тестов Django создается в оперативной памяти база данных специально для тестов, а поля в ней заполняются теми данными, которые были использованы в тестах [8].

Запустим тесты во всем приложении:

C:\Car\_rental\_project>python manage.py test

Creating test database for alias 'default'...

..............

----------------------------------------------------------------------

Ran 12 tests in 25.084s

OK

Destroying test database for alias 'default'..Затраченное время: 00:00:29.935

Из результатов запуска тестов видно, что все тесты, которых было запущено 12, прошли успешно.

### 4.2.4 Покрытие кода

Покрытие кода является важной метрикой для обеспечения качества тестируемого приложения, особенно если речь о проектах со сложной логикой и большим объемом кода. Анализ покрытия кода выполняется с помощью специального инструментария, который позволяет проследить в какие строки, ветви и т.д. кода, были вхождения во время работы автотестов. Наиболее известные инструменты для проведения измерения покрытия кода: AQTime, Bounds Checker, Bullseye Coverage, Coverage Meter, Clover, NCover, IBM Rational PurifyPlus, Intel Compiler, Intel Code Coverage Tool Prototype, JetBrains. С помощью анализа покрытия кода можно оценить плотность покрытия авто-тестами исполняемого кода тестируемого приложения (можно ответить на вопрос «какой объем тестирования мы (наши автотесты) выполняем?»). При детальном анализе результатов покрытия кода автотестами можно оценить покрытие отдельных компонентов системы (т.е. можно ответить на вопросы: что и в каком объеме мы тестируем?, в каких местах нужно оптимизировать покрытие?, какие места системы не проверяются тестами? и т.д.). Таким образом, зная данную метрику, станет ясно для каких тестовых случаев нужно создать новые тесты, или убрать дублирующие тесты. Данные мероприятия помогут увеличить значение метрики Code Coverage, что в свою очередь должно повысить качество кода и качество тестируемого приложения в целом. Естественно, чем выше показатель данной метрики – тем лучше, однако уже хорошо если у вас покрыты тестами наиболее сложные и важные фрагменты кода.

Различают несколько способов измерения покрытия кода. В Википедии представлены следующие определения:

* **Покрытие операторов** – каждая ли строка исходного кода была выполнена и протестирована?;
* **Покрытие условий** – каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована?;
* **Покрытие путей** – все ли возможные пути через заданную часть кода были выполнены и протестированы?;
* **Покрытие функций** – каждая ли функция программы была выполнена?;
* **Покрытие вход/выход** – все ли вызовы функций и возвраты из них были выполнены? [9].

Jenkins CI позволяет оценить степень покрытия кода тестами.

### 4.2.5 Запуск приложения для тестирования

Приложение запускается на сервере разработки (IP 127.0.0.1:8000). Cервер этот предназначен только для непосредственно для разработки нашего приложения. Для него не требуется много ресурсов.

Сервер запускается, как раньше сказано на 127.0.0.1:8000. Запуск и отладка проекта производится на локальном тестовом сервере, который предназначен для разработки и выявления правильности работы приложения, отладки кода программы, тестирования пользовательского интерфейса.

Успешность тестирования также позволяет оценить вышеупомянутый сервис Jenkins CI.

# 5 ДОКУМЕНТАЦИЯ

## 5.1 Назначение программы

Разработанный проект ООО «Tomsk-Car-Rent» предназначен для оформления аренды автомобиля для своих нужд.

В частности, данный сайт позволяет:

* Добавить автомобиль в свою персональную коллекцию;
* Просматривать информацию различных автомобилей;
* Найти необходимый автомобиль;
* Взять в аренду интересующий автомобиль;
* Просмотреть список своих аренд.
* Отменить аренду, если она ещё не началась;

Сайт фирмы ООО «Tomsk-Car-Rent» рассчитан на людей, имеющих доступ в интернет и желающих арендовать автомобиль быстро и удобно.

## 5.2 Условия запуска программы

Для пользования сайтом пользователю необходимо иметь персональный компьютер с возможностью выхода в интернет со скоростью не менее 256 КБ/с. Также необходим установленный браузер и устройство ввода.

## 5.3 Выполнение программы

Сервис выполняет следующие функции:

**5.3.1 Регистрация пользователя**

Для начала работы с сервисом необходимо зарегистрироваться. Для этого достаточно нажать на кнопку зарегистрироваться в нижней левой части меню. Для регистрации необходимо заполнить поля Email и пароля (рис. 5.1).

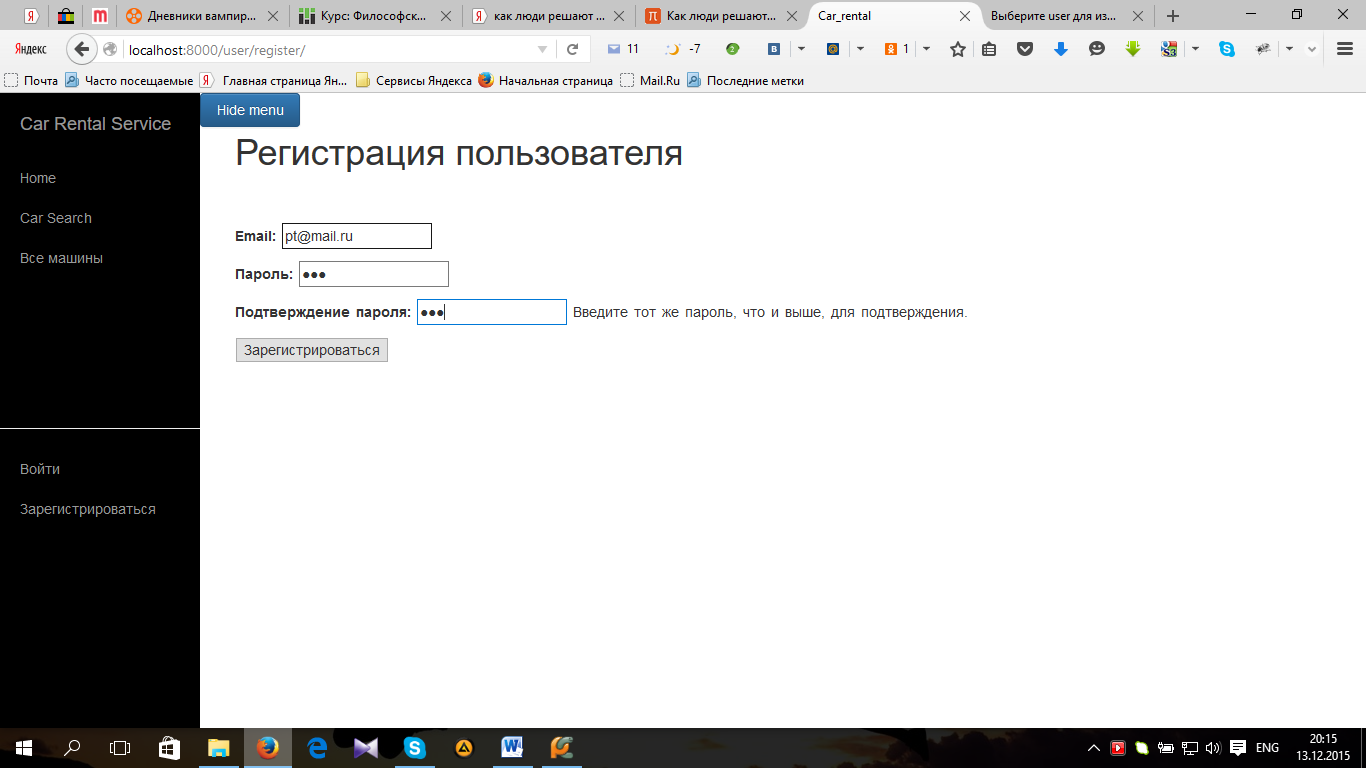


Рис. 5.1 Форма регистрации пользователя

### 5.3.2 Авторизация пользователя

Для авторизации необходимо кликнуть по кнопке «Войти» в нижней левой части меню на любой страницы. Откроется страница с полями: Email и Пароль, который необходимо заполнить. Общий вид страницы авторизации представлен на рис. 5.2.

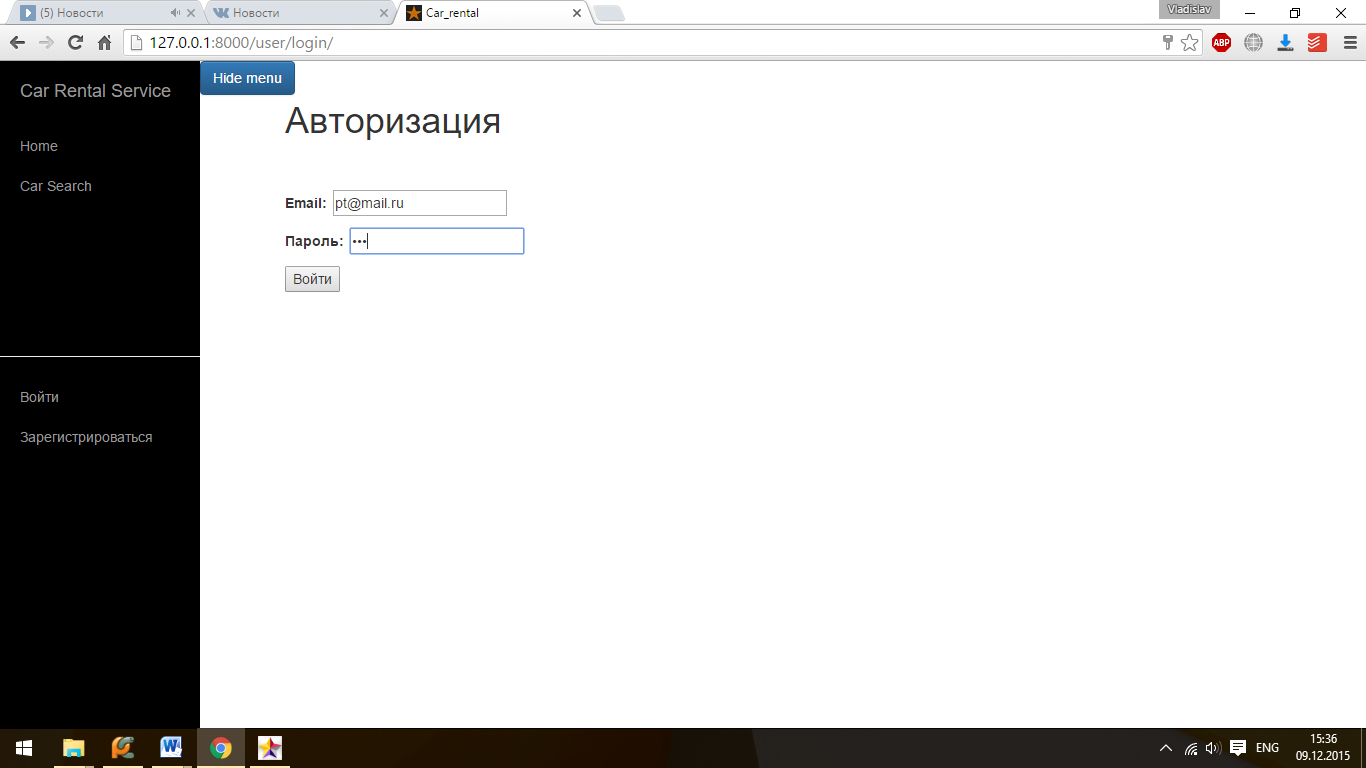


Рис. 5.2 Форма авторизации пользователя

Если введённые данные о пользователе корректны, то авторизация завершается и пользователь возвращается на главную страницу.

### Редактирование профиля

На страницу профиля можно перейти через панель пользователя. Страница пользователя содержит информацию, введённую пользователем при регистрации. Её можно отредактировать, нажав на кнопку «Редактировать» (рис. 5.3).



Рис 5.3 Профиль пользователя

После редактирования профиля вся изменённая информация будет сохранена. Форма редактирования профиля представлена на рис. 5.4.

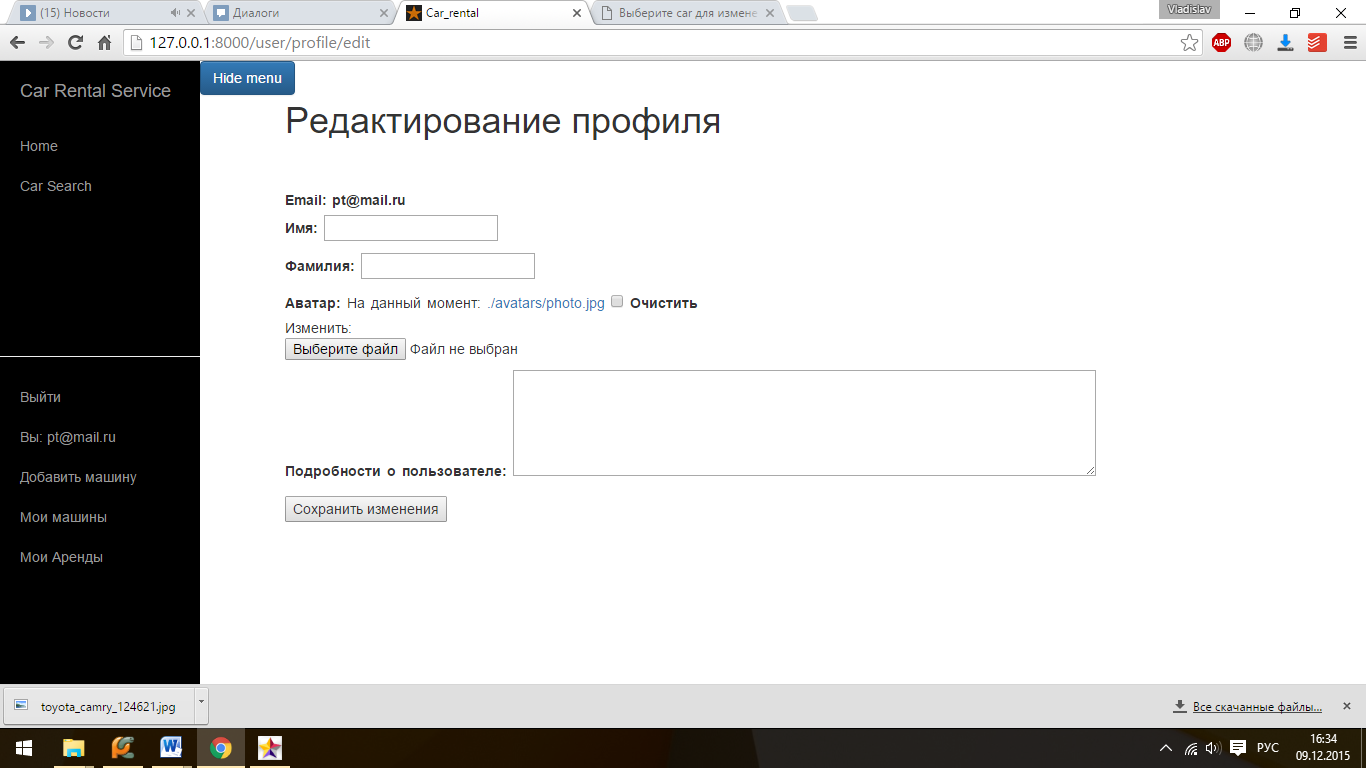


Рис. 5.4 Форма редактирования профиля

* + 1. **Смена пароля**

Также в профиле можно сменить пароль. Это происходит при нажатии на кнопку «Сменить пароль». В форме необходимо ввести старый пароль и новый (рис. 5.5).

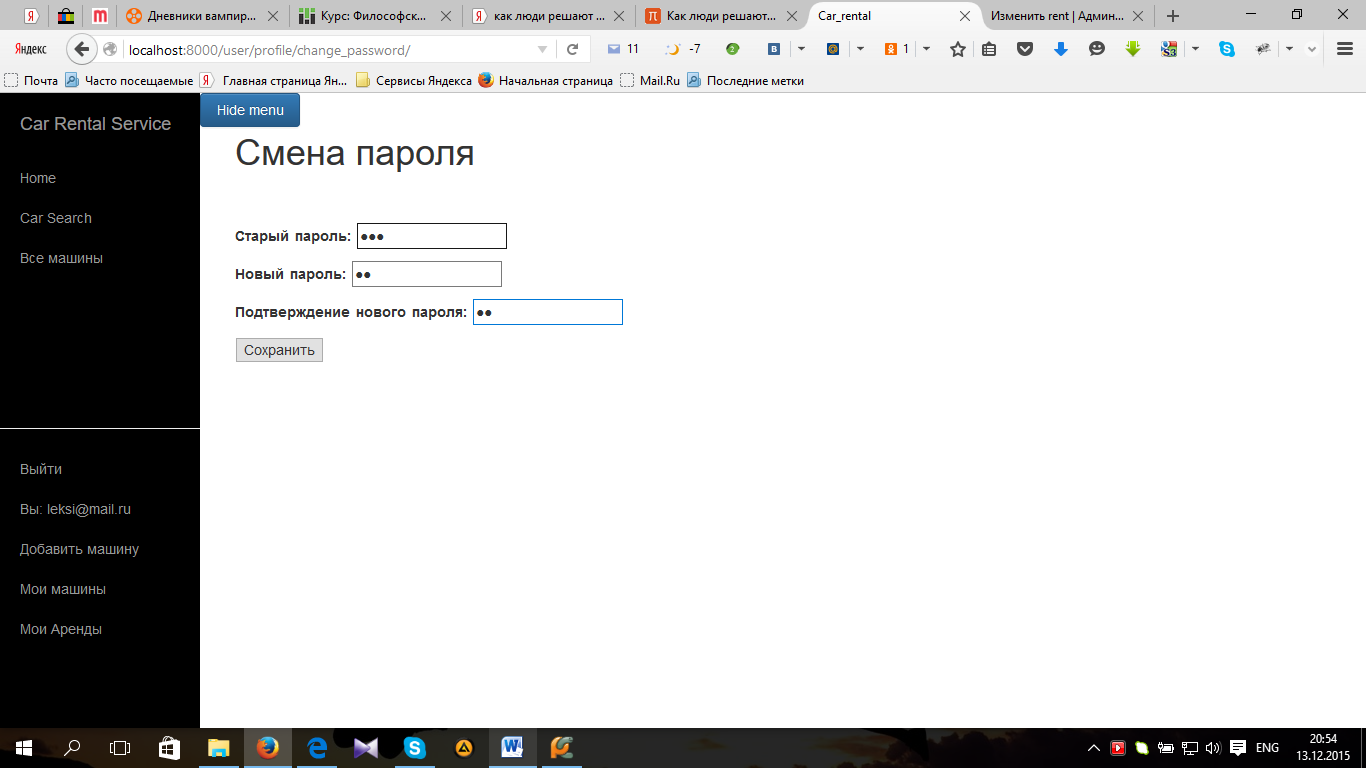


Рис. 5.5 Форма смены пароля

### 5.3.5 Добавления автомобиля

Добавить автомобиль в свою коллекцию можно через кнопку «Добавить машину» на панели пользователя. Форма добавления автомобиля представлена на рис. 5.6.

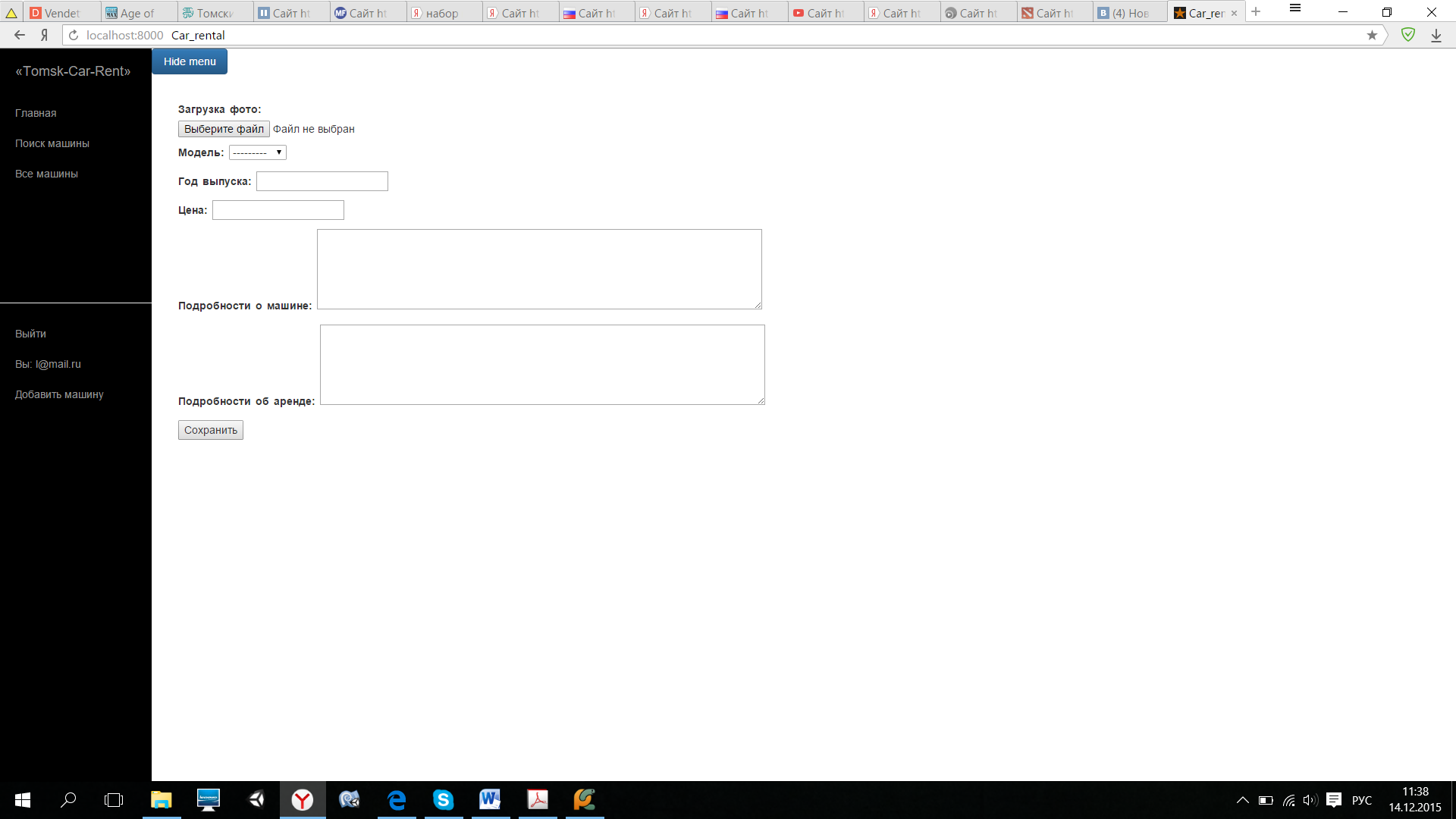


Рис. 5.6 Форма добавления автомобиля

После заполнения всех полей автомобиль будет добавлен в базу данных сайта.

### 5.3.6 Поиск автомобиля

Форма поиска автомобиля представлена на рис. 5.7.

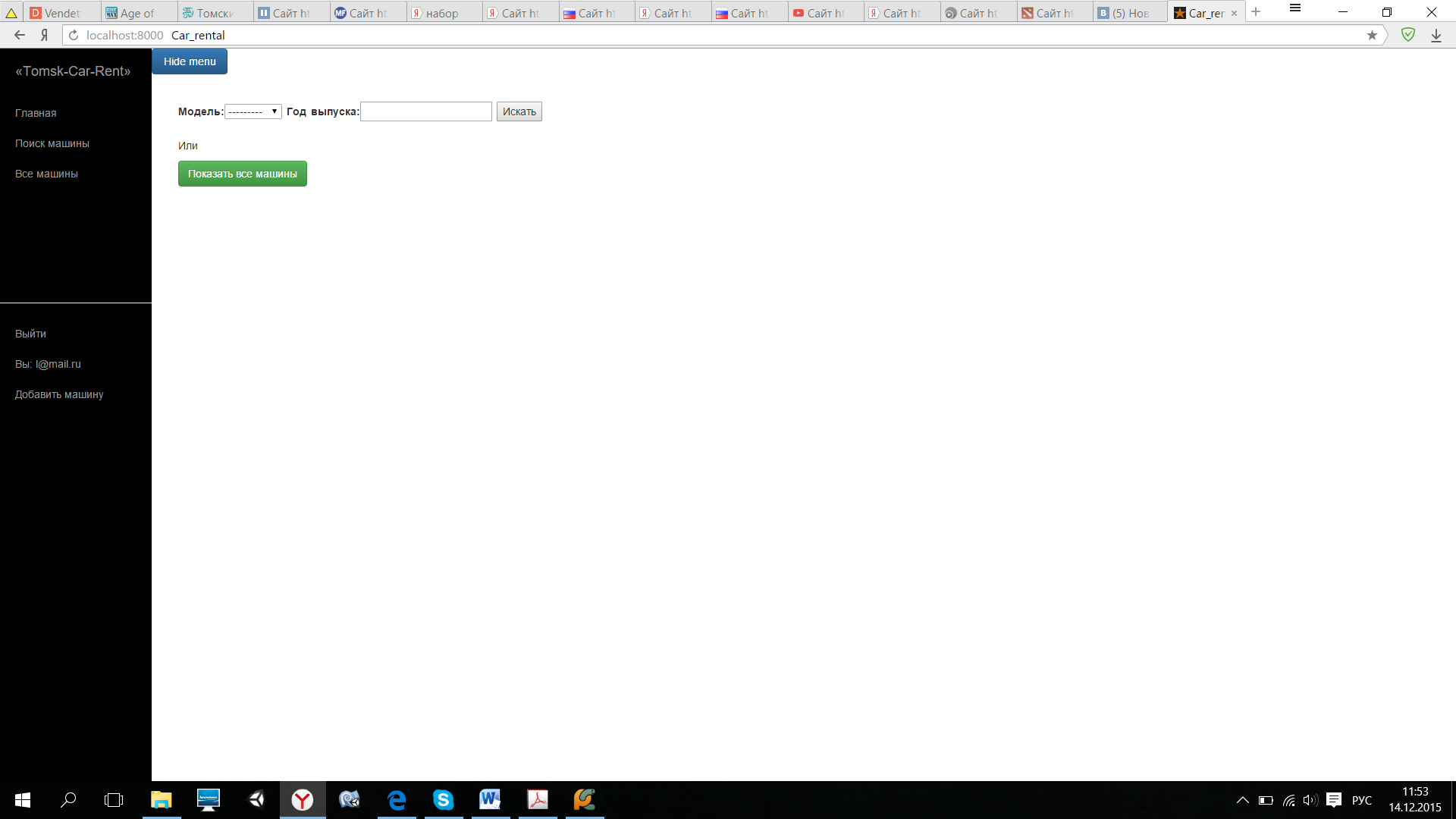


Рис. 5.7 Форма поиска автомобиля

Клиент может выбрать модель автомобиля и указать год, чтобы получить список подходящих автомобилей. Так же пользователь может нажать на кнопку «Показать все машины», чтобы увидеть все автомобили, зарегистрированные на сайте (рис. 5.8).

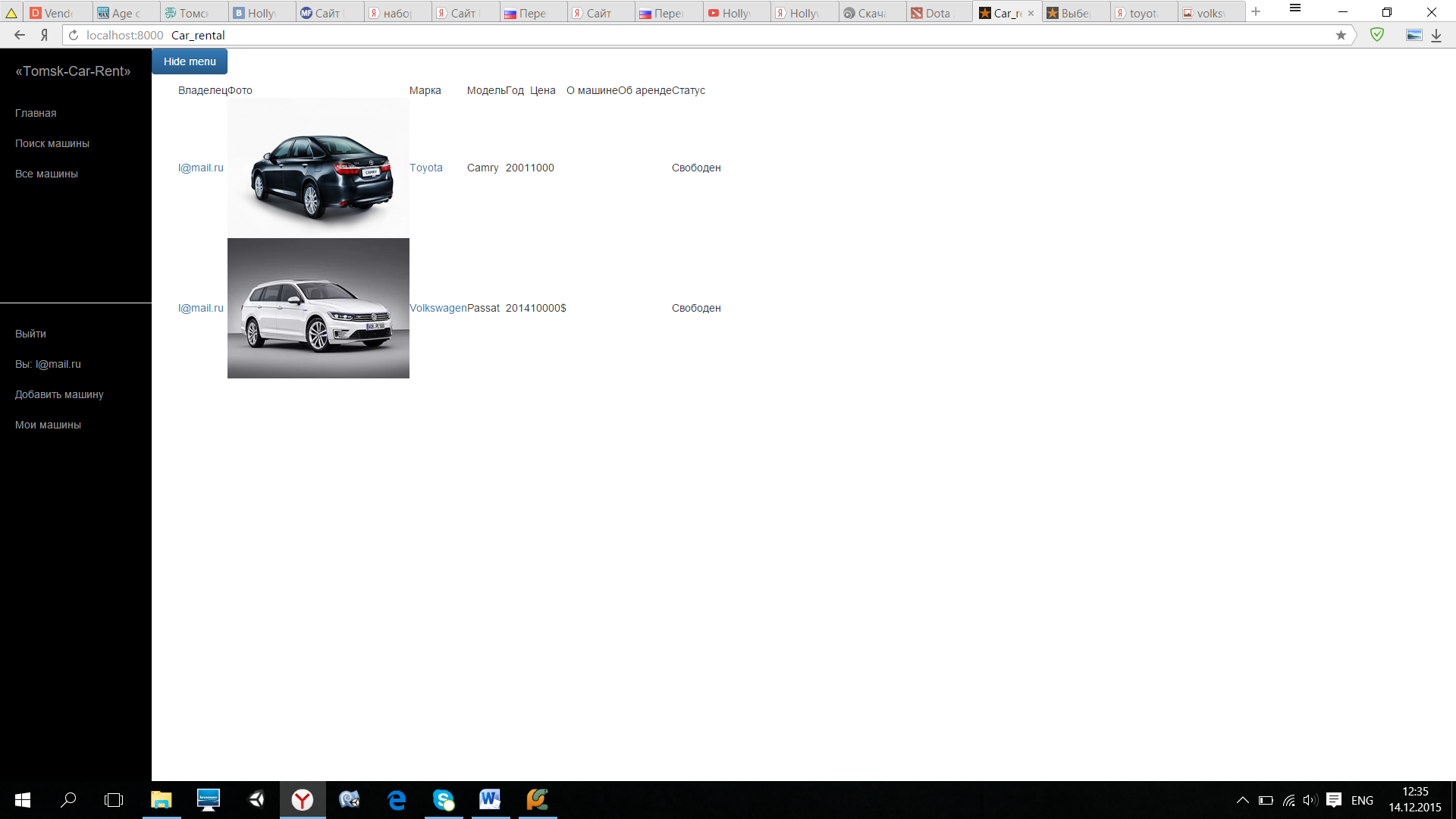


Рис. 5.8 Форма представления всех автомобилей на сайте

### 5.3.7 Аренда автомобиля

Для того чтобы арендовать автомобиль, необходимо сначала выбрать автомобиль, нажав на его марку (рис. 5.9), которая приведет к подробному описанию автомобиля.

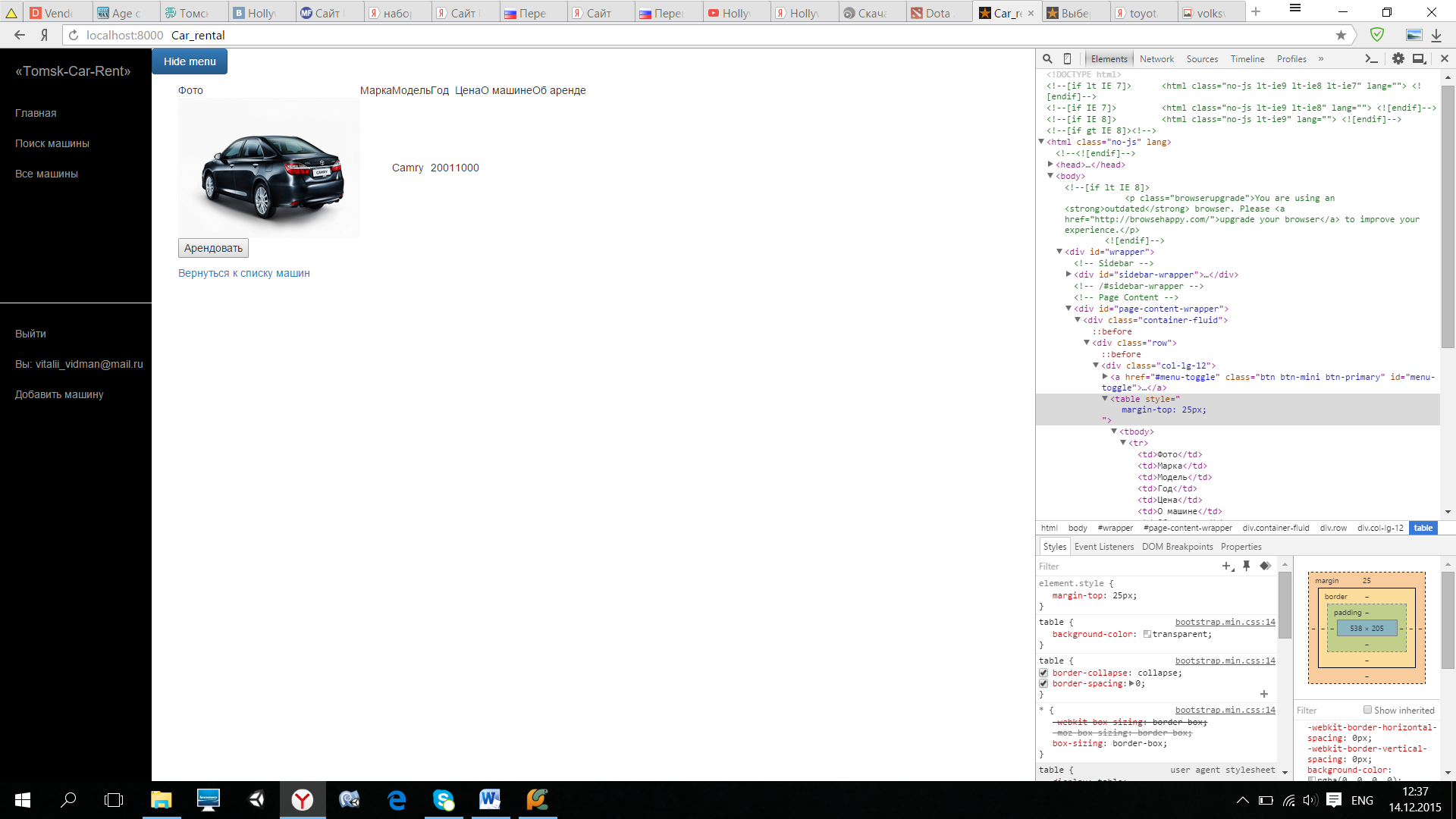


Рис. 5.9 Подробное описание автомобиля

На форме описания автомобиля необходимо нажать на кнопку «Арендовать», чтобы перейти на форму аренды (рис. 5.10).

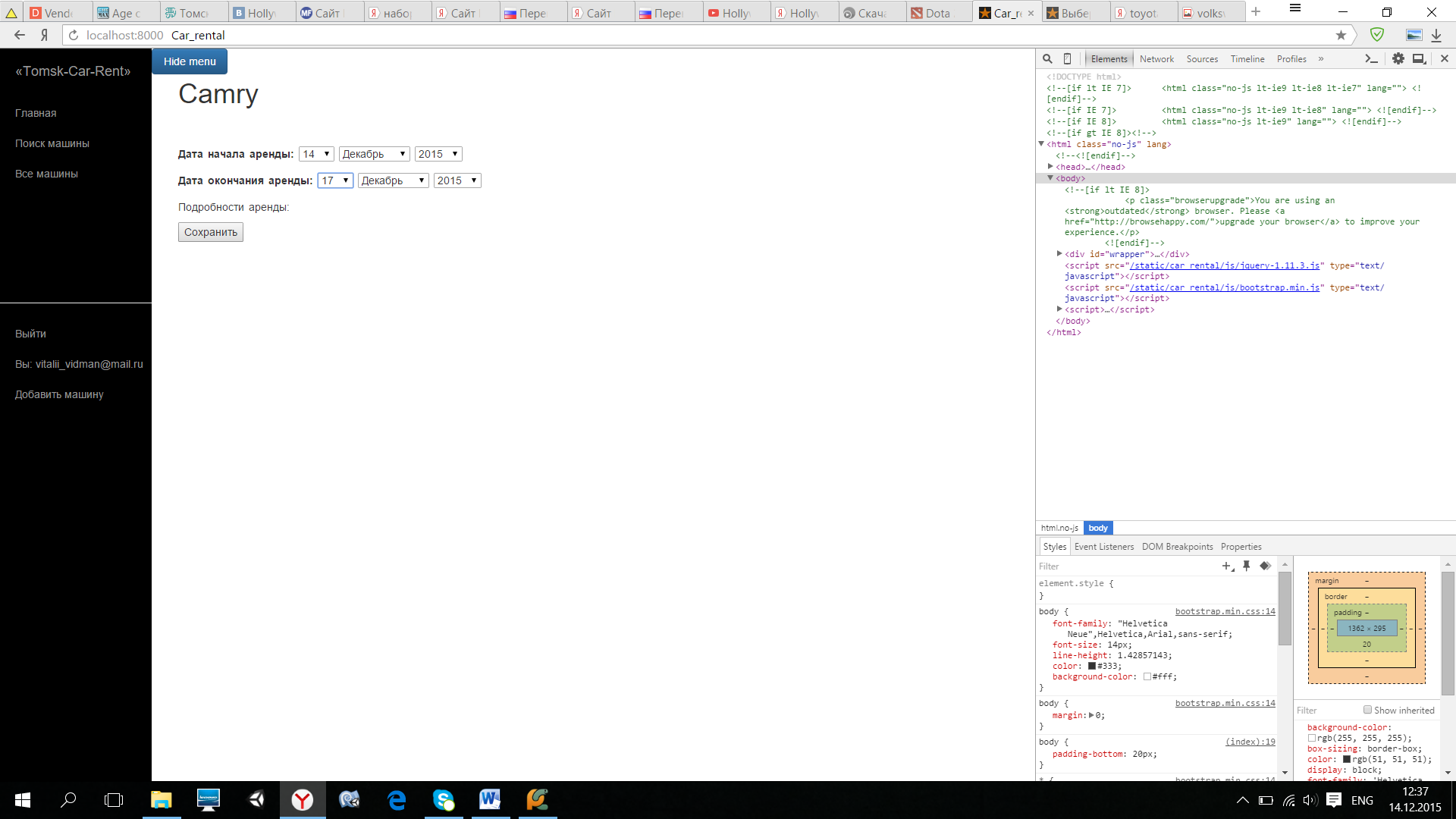


Рис. 5.10 Аренда автомобиля

После заполнения полей аренда будет зарегистрирована на сайте.

### Отмена аренда

Пользователь вправе отменить аренду, если она ещё не была начата. Для этого необходимо перейти в «Мои аренды» через кнопку на панели пользователя. Перед пользователем будет представлен список всех его аренд, и если аренда ещё не начата, то пользователь может нажать «Отменить» и аренда будет считаться несостоявшейся (рис. 5.11).



Рис. 5.11 Список аренд

### Завершение аренды

Завершить начатую аренду может только владелец машины во избежание недоразумений. Для этого нужно в меню выбрать «Аренды моих машин» (рис. 5.12).

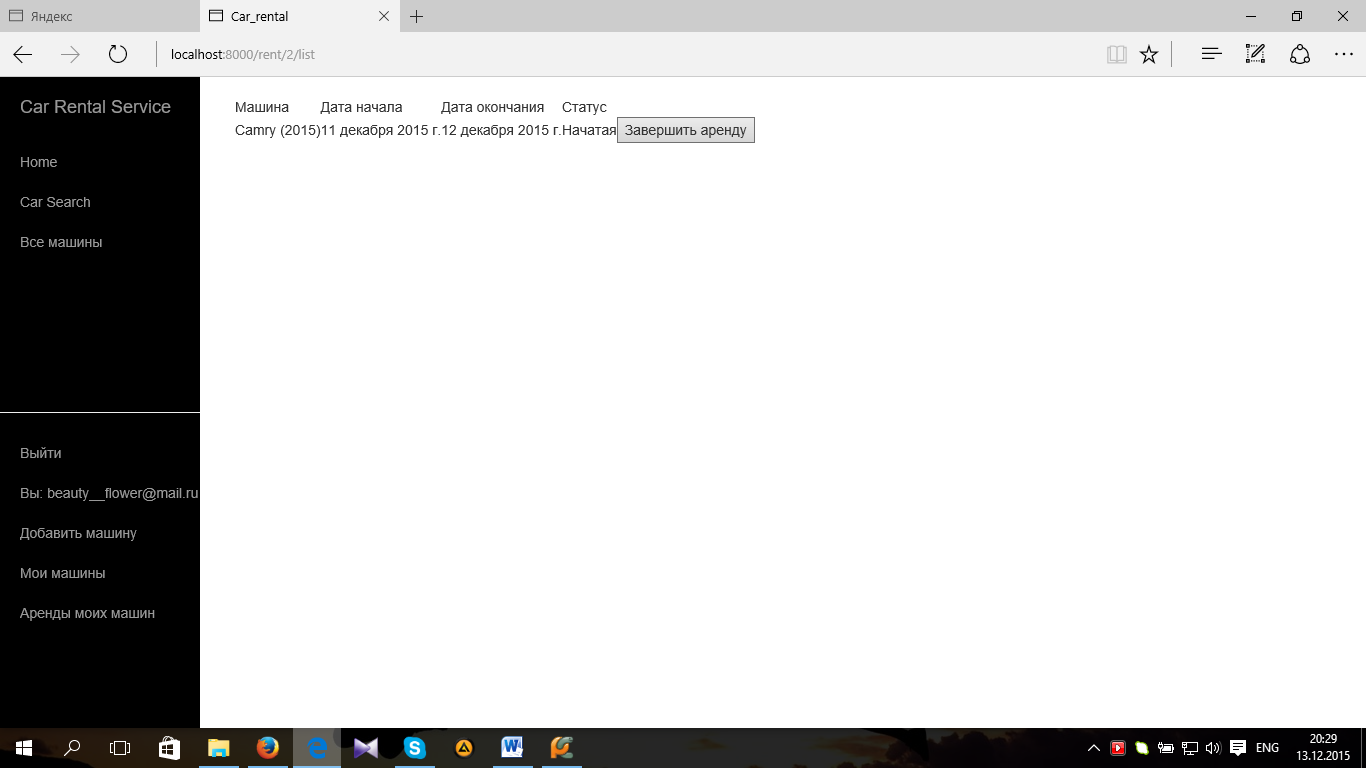


Рис. 5.12 Список аренд

## 5.4 Сообщения программы

При заполнении полей форм пользователем нередко могут появляться сообщения о некорректности введенных данных. Это происходит в том случае, если на поля наложены ограничения, то есть при их заполнении неверными данными или оставлении пустыми пользователь получает сообщение-уведомление об ошибке.

**5.4.1 Ошибки при регистрации**

Если при регистрации пользователь указал разные пароли он будет об этом уведомлен (рис. 5.13). Также сообщение об ошибке может появиться при не заполнении полей формы.

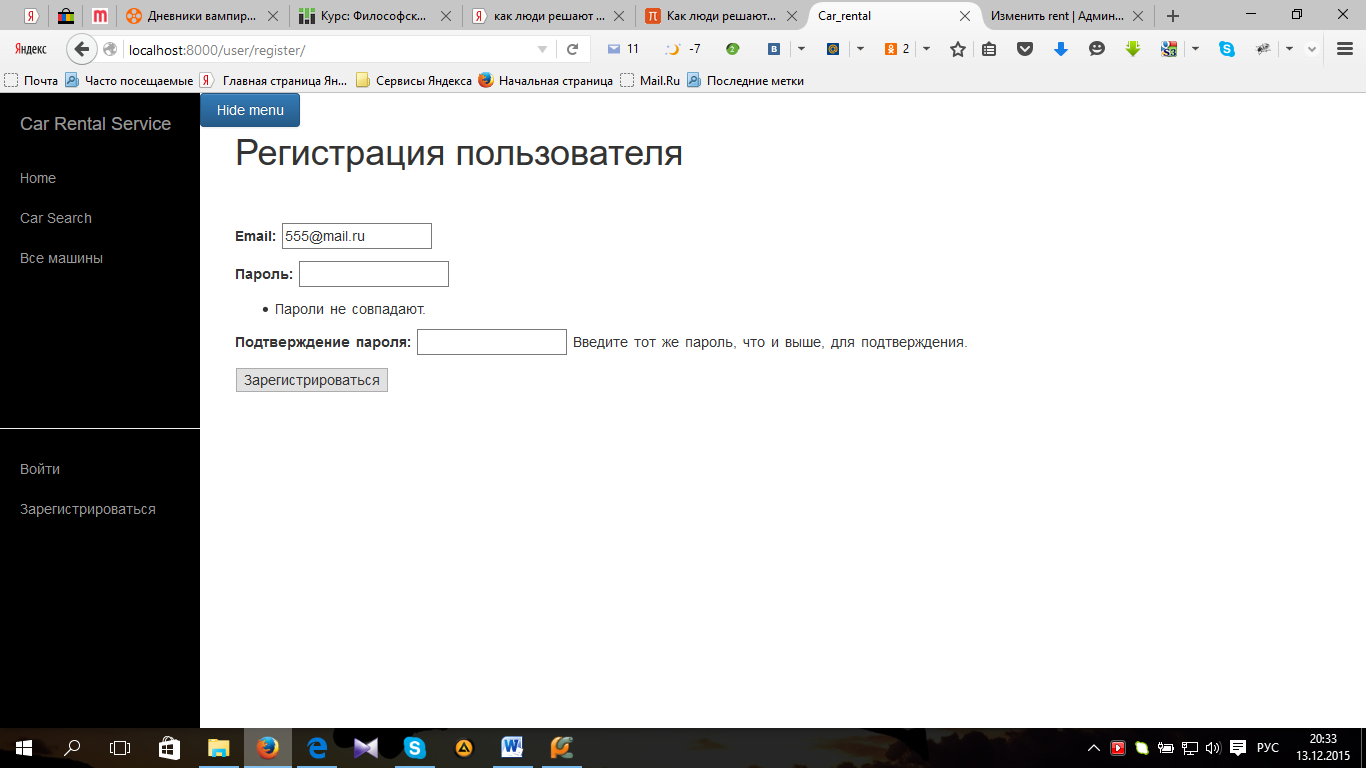


Рис. 5.13 Сообщение о некорректной регистрации

**5.4.2 Ошибки при авторизации**

Если комбинация Email-Пароль была указана неверно, на странице появится оповещение об этом (рис. 5.14).

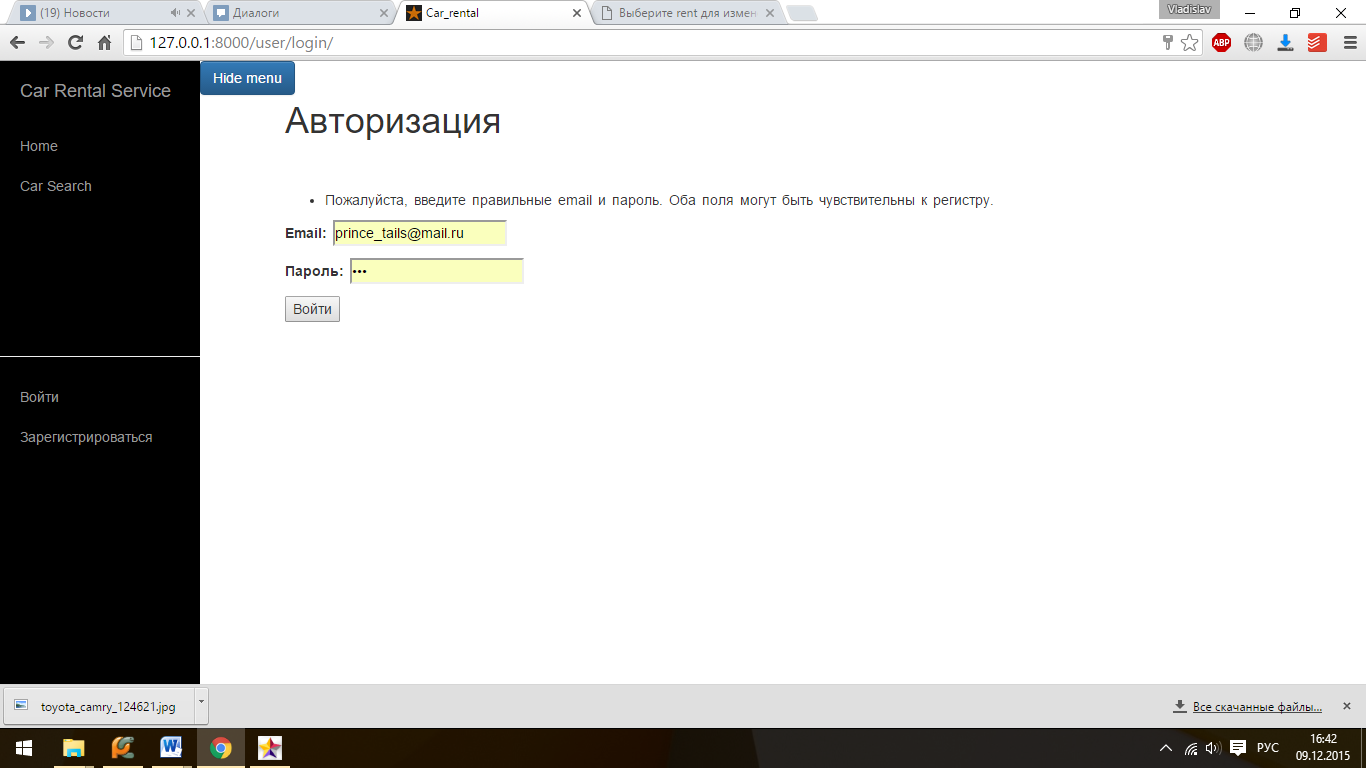


Рис. 5.14 Сообщение о некорректной авторизации

В качестве реализации сообщений пользователю была выполнена только проверка Email-Пароля. Как известно, сайт, программа или веб-приложение, в принципе, могут существовать и без них. Но как показала практика, пользователь не всегда может сам понять, что нужно изменить. Поэтому существуют такие своеобразные сообщения-подсказки.

**5.4.3 Ошибки при смене пароля**

При смене пароля сообщение о некорректном вводе данных может всплыть, если введен неправильно старый пароль (рис. 5.15) либо новый пароль не совпадает.

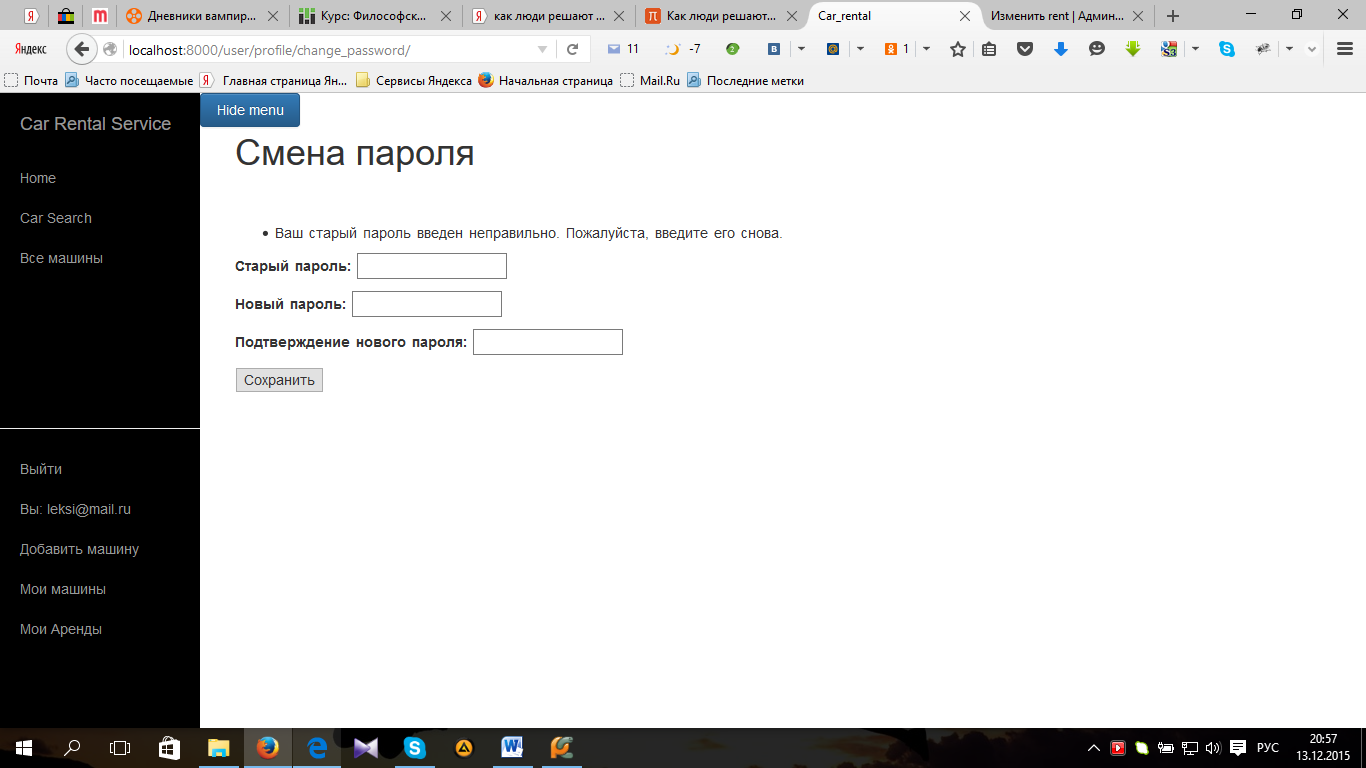


Рис. 5.15 Сообщение о некорректном вводе пароля

**5.4.3 Ошибки при добавлении аренды**

При добавлении аренды осуществлена проверка на вводимые даты. Дата начала аренды, также как и дата окончания, не могут быть меньше текущей даты. В свою очередь дата окончания аренды не может быть меньше даты начала (рис. 5.16).

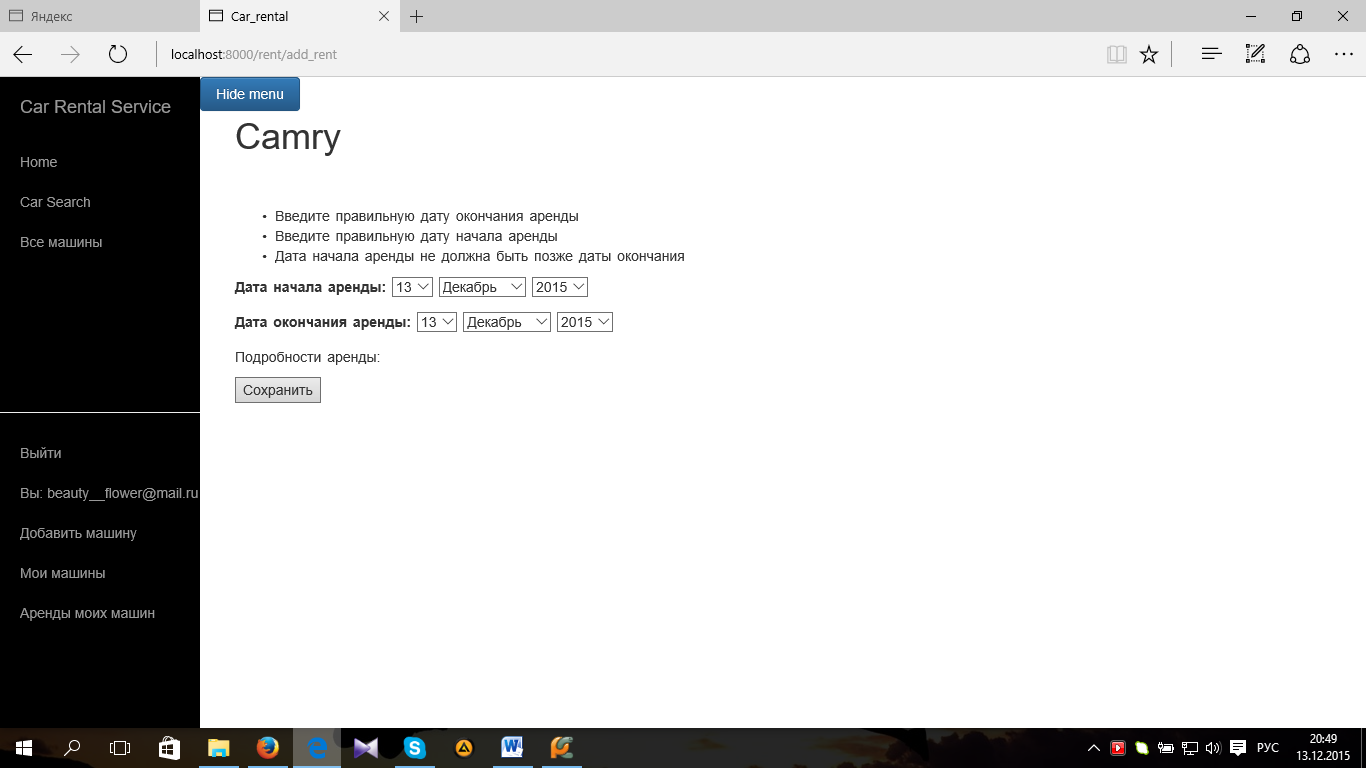


Рис. 5.16 Сообщение о некорректном вводе дат при добавлении аренды

**5.4.4 Ошибки при добавлении новой машины**

При добавлении новой машины необходимо внимательно заполнять поля, но иногда даже внимательно следя за вводимыми данными можно совершить ошибку. Для этого в нашей системе происходит проверка года выпуска, он не может быть больше текущего года (рис. 5.17).

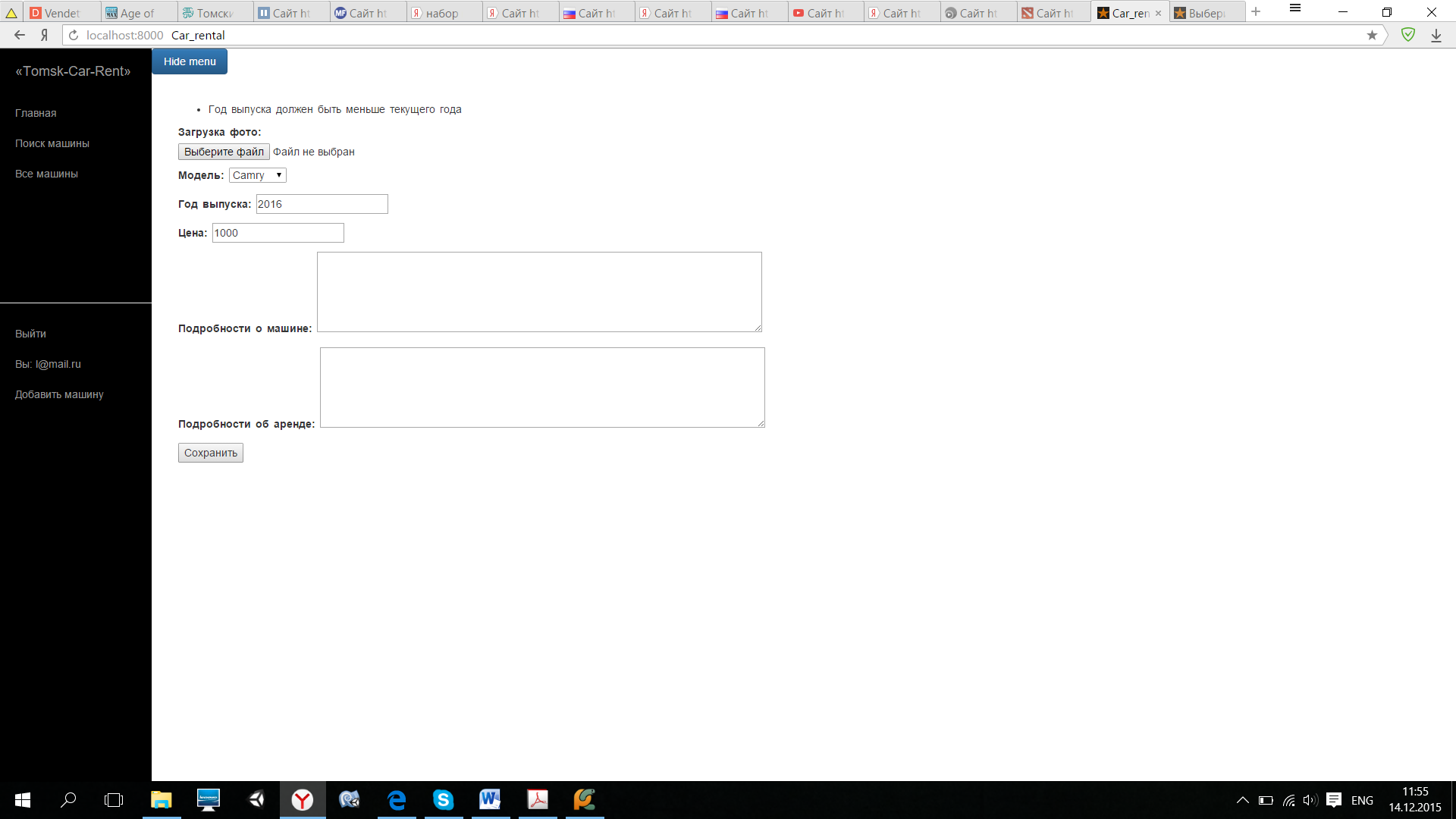


Рис. 5.17 Сообщение о некорректном вводе года выпуска машины при добавлении машины

# ЗАКЛЮЧЕНИЕ

Результатом данной курсовой работы является созданный сайт аренды автомобилей ООО «Tomsk-Car-Rent», разработанный с использованием языка программирования Python и платформы для создания веб-приложений Django. При разработке проекта была использована система контроля версий GitHub и одноименный хостинг.

Создание данного приложения включало анализ и проектирование, что облегчило его разработку и помогло соблюдать требования. Поэтому данный подход был оправдан.

В будущем планируется дальнейшая модификация приложения, так как есть еще множество идей, которые в связи с отсутствием необходимого времени, не удалось реализовать. При доработке и модификации программного продукта, возможно, его внедрения для практического применения, так как рассматриваемая область имеет большой спрос в современно мире. Кроме того, преимущество использования Django, которое заключается в возможности создания API для работы с сервисом с помощью любого устройства без изменений в исходном коде программы.

# Список использованных источников

1. ГОСТ 19.201-78. Техническое задание. Требования к содержанию и оформлению;
2. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 624 с., ил.
3. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя: Пер.с англ.-М: ДМК, 2000. – 432 с.,ил.
4. НОУ ИНТУИТ. Лекция. Виды диаграмм UML [Электронный ресурс] – URL: http://www.intuit.ru/studies/courses/1007/229/lecture/5954?page=4, свободный (дата обращения 05.12.2015).
5. Теория тестирования программного обеспечения [Электронный ресурс] – URL: http://alexproger.narod.ru/Interv/test1.html, свободный (дата обращения 29.11.2015).
6. НОУ ИНТУИТ. Лекция. Модульное тестирование [Электронный ресурс] – URL: http://www.intuit.ru/studies/courses/1040/209/lecture/5394, свободный (дата обращения 29.11.2015).
7. Интеграционное тестирование – Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5\_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5, свободный (дата обращения 29.11.2015).
8. Создаём своё первое приложение с Django, часть 5 – Документация Django 1.8 [Электронный ресурс] – URL: http://djbook.ru/rel1.8/intro/tutorial05.html, свободный (дата обращения 29.11.2015).
9. Как мы Code Coverage измеряли [Электронный ресурс] – URL: http://bugscatcher.net/archives/1103, свободный (дата обращения 29.11.2015).
10. Непрерывная интеграция – Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F\_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F, свободный (дата обращения 30.11.2015).
11. Настройка Jenkins для django проекта с нуля / Хабрахабр [Электронный ресурс] – URL: http://habrahabr.ru/post/132521/, свободный (дата обращения 30.11.2015).