**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт: Институт Кибернетики

Направление подготовки (специальность): Информатика и вычислительная техника

Кафедра: Оптимизации Систем Управления

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине «Технологии разработки программного обеспечения»

на тему «Автоматизированная система регистрации заказов в интернет-магазине “ООО "Авто-импорт"”»

Выполнили студенты гр. 8ВМ43 \_\_\_\_\_\_\_\_\_\_\_\_ А.В. Быков

\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Рогова

\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Холманский

\_\_\_\_\_\_\_\_\_\_\_\_ Е.Н. Чеботарева

Дата сдачи пояснительной записки преподавателю \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Руководитель : ассистент каф. ОСУ И.А. Заикин

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Оценка руководителя) (Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

(Дата проверки)

Курсовой проект студенты А.В. Быков, Е.В. Рогова, М.В. Холманский, Е.Н. Чеботарева выполнили и защитили

(Ф.И.О.)

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

(дата защиты )

Томск 2014 г.

Оглавление

[Введение 4](#_Toc409631520)

[1 Требования к программе 7](#_Toc409631521)

[1.1 Назначение программы 7](#_Toc409631522)

[1.2 Область применения 7](#_Toc409631523)

[1.3 Задачи, решаемые программой 7](#_Toc409631524)

[1.4 Функциональные требования 7](#_Toc409631525)

[1.5 Требования к надежности 8](#_Toc409631526)

[1.6 Требования к эргономике и технической эстетике 9](#_Toc409631527)

[1.7 Требования к программной документации 9](#_Toc409631528)

[1.8 Стадии и этапы разработки 10](#_Toc409631529)

[1.9 Диаграмма вариантов использования 10](#_Toc409631530)

[1.10 Детализация вариантов использования 11](#_Toc409631531)

[1.10.1 Вариант использования «Записаться на техническое обслуживание online» 11](#_Toc409631532)

[1.10.2 Вариант использования «Выбрать авто» 12](#_Toc409631533)

[1.10.3 Вариант использования «Записаться на тест драйв online» 13](#_Toc409631534)

[1.10.4 Вариант использования «Оформить заявку на покупку автомобиля» 13](#_Toc409631535)

[2 Анализ 15](#_Toc409631536)

[2.1 Диаграмма классов анализа 15](#_Toc409631537)

[2.2 Диаграмма состояний 17](#_Toc409631538)

[3 Проектирование 20](#_Toc409631539)

[3.1 Проектные классы 20](#_Toc409631540)

[3.2 Диаграмма пакетов системы 21](#_Toc409631541)

[3.2 Диаграммы последовательностей для операций проектных классов 22](#_Toc409631542)

[4 Реализация 26](#_Toc409631543)

[4.1 Тестирование 26](#_Toc409631544)

[4.1.1 Модульные тесты 26](#_Toc409631545)

[4.1.2 Интеграционные тесты 28](#_Toc409631546)

[4.1.3 Построение и выполнение тестов 29](#_Toc409631547)

[4.1.4 Покрытие кода 30](#_Toc409631548)

[4.1.5 Запуск приложения для тестирования 30](#_Toc409631549)

[4.2 Непрерывная интеграция 31](#_Toc409631550)

[5 Документация 34](#_Toc409631551)

[5.1 Назначение программы 34](#_Toc409631552)

[5.2 Условия запуска программы 34](#_Toc409631553)

[5.3 Выполнение программы 35](#_Toc409631554)

[5.3.1 Авторизация работника заведения 35](#_Toc409631555)

[5.3.2 Просмотр списка заказов работником заведения 35](#_Toc409631556)

[5.3.3 Выбор автомобилей по маркам 36](#_Toc409631557)

[5.3.4 Добавление автомобиля в корзину 37](#_Toc409631558)

[5.3.5 Работа с корзиной 37](#_Toc409631559)

[5.3.6 Формирование заказов 38](#_Toc409631560)

[5.3.7 Оформление заказа 38](#_Toc409631561)

[5.4 Сообщения программы 40](#_Toc409631562)

[Заключение 42](#_Toc409631563)

[Список использованных источников 43](#_Toc409631564)

[Приложение А 45](#_Toc409631565)

[Приложение Б 47](#_Toc409631566)

[Приложение В 48](#_Toc409631567)

[Приложение Г 49](#_Toc409631568)

# Введение

Использование интернет-технологий в коммерческой деятельности – одна из наиболее актуальных задач в современном мире. ХХI век уже невозможно представить без интернета и интернет технологий. Всемирная паутина прочно вошла в нашу жизнь. Количество пользователей Интернета в настоящее время составляет свыше 25% от числа жителей Земли, а по информации Международного телекоммуникационного союза (ITU) Общемировая аудитория Всемирной паутины составляет почти два миллиарда человек. При этом с 2000 года она увеличилась более чем в пять раз. Показатель проникновения интернета по всему миру равняется 28,7 процента. Таким образом, выход в Сеть есть у каждого четвертого жителя Земли.

Причем, как правило, наиболее активными пользователями сети интернет являются молодежь, жители крупных городов, а также люди с высоким уровнем дохода. Следовательно, можно сказать, что было бы крайне неразумно оставлять без внимания потребительские запросы этой аудитории, не попытавшись удовлетворить хотя бы часть таких запросов прямо в Интернете.

Реализовать эту идею на практике призваны Интернет-магазины. Интернет-магазин — высокоэффективный и малозатратный инструмент для успешного ведения бизнеса. В отличие от обычного магазина, интернет-магазин открыт 24 часа в сутки, 7 дней в неделю, доступен любому пользователю в любой точке планеты и не требует финансовых затрат на аренду помещения, значительно снижает издержки на персонал. Интернет-магазин сочетает невысокую стоимость и высокую эффективность: создание интернет-магазина помогает компании завоевать или укрепить свои позиции на рынке товаров и услуг, увеличить клиентскую базу и повысить популярность бренда. Иными словами, создание интернет-магазина приносит владельцам реальную прибыль. Именно поэтому в настоящее время все большее количество компаний приходит к решению о создании интернет-магазина.

 Из всего вышеуказанного, можно заключить, что популярность Интернет-магазинов растёт с каждым днём.  Хорошо раскрученный Интернет-магазин приносит прибыли не меньше, чем его обычный аналог, причём содержать и изготовить Интернет-магазин на порядок выгодней - нет затрат на дорогую аренду и других расходов.  
   Однако не все четко представляют себе, каким он должен быть интернет магазин, как верно рассчитать обеспечить его прибыльность и как правильно вести учет расходов, связанных с его созданием и обслуживанием.  
  Ведь качественный интернет-магазин - это магазин с хорошо структурированным и интересно описанным ассортиментом, с высоким качеством обслуживания его клиентов, с удобной навигацией и поисковой системой, обеспечивающие быстрый и удобный поиск товара, его демонстрацию и понятный заказ. Немаловажен фактор привлекательности внешнего оформления сайта магазина, а также множество других особенностей и мелочей, которые помогают клиенту сделать покупку. Все это – может стать гарантией того, что клиент обязательно посетит этот интернет-магазин снова.

Целью курсового проекта является разработка интернет-магазина ООО "Авто- Импорт".

Объектом данного курсового проекта является магазин ООО "Авто- импорт", который занимается продажей автомобилей и оказанием различных услуг сервиса.

Данный проект был разработан на языке Python. 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.

При создании данного web-приложения используется открытая платформа Django на основе языка программирования Python. Выбор данной платформы обоснован тем, что она позволяет создавать web ресурсы достаточно быстро по сравнению с другими платформами, (например, на основе php) и дает возможность не создавать лишних элементов, не подстраиваться под заданные рамки.

# 1 Требования к программе

## 1.1 Назначение программы

Web приложение ООО "Авто - импорт" предназначено для обеспечения пользователей полной, оперативной и актуальной информацией, касающейся системы обслуживания клиентов в интернет магазине, также продажи автомобилей и оказанием различных услуг сервиса магазина.

## 1.2 Область применения

Данный программный продукт является сервисом, предназначенным для организаций, предоставляющих актуальную информацию о различных услугах автомобильного сервиса.

## 1.3 Задачи, решаемые программой

1. Язык реализации – Python с интерпретатором версии не ниже 3.4;
2. Фреймворк Django версии не ниже 1.7;
3. Основной язык программы – русский;
4. Система должна производить валидацию полей не более, чем в течении трех секунд;
5. Система должна производить авторизацию пользователя не более, чем в течении трех секунд;
6. Функциональная система должна разрабатываться и функционировать в рамках существующего законодательства Российской Федерации.

## 1.4 Функциональные требования

1. Программа ООО "Авто - импорт" (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 предоставить возможность оставить комментарий к автомобилю;
5. Система должна подсчитывать общую суму стоимости автомобилей, добавленных в корзину заказа;
6. Система должна предоставлять возможность просмотра корзины заказов;
7. Система должна предоставлять возможность удаления всех автомобилей из корзины заказов;
8. Система должна предоставлять пользователю возможность выбора вида заказа;
9. Система должна предоставлять форму для ввода необходимых для оформления заказа данных;
10. Система должна сохранять заказ после его оформления;
11. Система должна предоставлять возможность аутентификации пользователя;
12. Система должна предоставлять возможность авторизации пользователя;
13. Система должна предоставлять возможность просмотра авторизованному пользователю заказов, сделанных в данной программной системы;
14. Система должна предоставлять возможность принятия заказа на исполнение и отмены заказа авторизованному пользователю.

## 1.5 Требования к надежности

Работа системы должна гарантировать средний аптайм – 99 % в год. Для этого система должна быть развернута на облачном сервисе, предоставляющем платформу как услугу, либо на сервере организации-клиента под управлением последней стабильной версии Windows Server или серверной версии операционной системы Linux.

При вводе некорректной информации программная система должна выдать предупреждающее сообщение о некорректности введенных данных и предложить повторить ввод.

Восстановление системы после критических ошибок, приводящих к неработоспособности системы должно происходить в течение суток после момента возникновения ошибки.

## 1.6 Требования к эргономике и технической эстетике

Взаимодействие пользователя и программы осуществляется посредством графического интерфейса клиентской программной системы – браузера. Программа должна иметь элементы навигации. Программа должна быть выполнена в едином стиле, иметь одинаковое расположение основных элементов навигации. Программа должна поддерживать горячие клавиши подтверждения ввода – Enter и закрытия подсказок и всплывающих окон – Escape.

При вводе некорректных данных, должны выдаваться сообщения на основном языке программной системы о неправильном вводе данных и предложение ввести данные заново.

## 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 2014 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

Диаграммы вариантов использования – это начальный этап проектирования программного обеспечения. Они отображают функциональность будущего приложения со стороны пользователя и представляют собой самый простой способ отображения функциональности приложения.

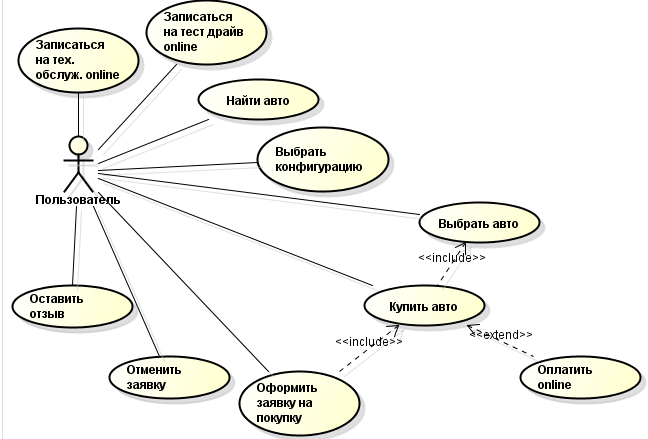


Рис. 1. Диаграмма вариантов использования программной системы

## 1.10 Детализация вариантов использования

Ниже представлены спецификации некоторых вариантов использования, изображенных на рис. 1[3, с. 101-103].

### 1.10.1 Вариант использования «Записаться на техническое обслуживание online»

|  |
| --- |
| Вариант использования: **Записаться на техническое обслуживание online** |
| Краткое описание:  Клиент записывается на техническое обслуживание. |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия:   1. Пользователь вошел в систему. |
| Основной поток:   1. Пользователь выбирает опцию «Записаться на техническое обслуживание online»; 2. Система запрашивает персональные данные клиента (город, дилера, марку, модель, регистрационный номер, пробег, год выпуска, тип обслуживания, желаемую дату, желаемое время, дополнительные пожелания, код проверки, соглашение на обработку персональных данных.) 3. Клиент вводит персональную информацию; 4. Клиент подтверждает введенные данные; 5. Система создает обращение; 6. Система возвращает Клиента к странице просмотра атомобилей. |
| Постусловия:   1. Заказ на техническое обслуживание оформлен. |
| Альтернативные потоки:   1. Нет. |

### 1.10.2 Вариант использования «Выбрать авто»

|  |
| --- |
| Вариант использования: **Выбрать авто** |
| Краткое описание:  Найти автомобиль. |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия:   1. Пользователь выбирает автомобиль. |
| Основной поток:   1. Пользователь выбирает опцию «Найти авто». 2. Система запрашивает у пользователя название марки автомобиля. 3. Пользователь выбирает из справочника нужную марку машины 4. Система запрашивает у пользователя название модели выбранной марки. 5. Пользователь выбирает модель автомобиля. 6. Система запрашивает комплектация автомобиля: базовая, средняя, максимальная. 7. Пользователь выбирает нужную комплектацию. 8. Система сохраняет полученные данные. 9. Система выполняет поиск автомобиля. 10. Система формирует всю информацию по автомобилю: описание, комплектацию, марку, модель, фотографию, дилера. |
| Постусловия: Автомобиль найден |
| Альтернативные потоки:   1. Данные введены корректно, система выводит на экран сообщение о том, что запрос выполнен успешно и данные выведены на экран. 2. Система выводит сообщение об отсутствии автомобиля, соответствующего введённым параметрам (критериям). |

### 1.10.3 Вариант использования «Записаться на тест драйв online»

|  |
| --- |
| Вариант использования: **Записаться на тест драйв** **online** |
| Краткое описание:  Записаться на тест драйв online |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия: Пользователь вошел в систему. |
| Основной поток:   1. Пользователь выбирает опцию «Записаться на тест драйв. Online». 2. Система запрашивает у пользователя: марку, модель, год выпуска, желаемую дату, желаемое время, дополнительные пожелания, код проверки, 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. |

### 1.10.4 Вариант использования «Оформить заявку на покупку автомобиля»

|  |
| --- |
| Вариант использования: Оформить заявку на покупку автомобиля |
| Краткое описание:  Клиент оформляет заявку на покупку автомобиля. |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия:   1. Клиент вошел в систему; 2. Система отображает корзину заказов. |
| Основной поток:   1. Вариант использования начинается, когда Клиент выбирает опцию оформить заявку на покупку автомобиля; 2. Клиент выбирает модель автомобиля; 3. Клиент выбирает конфигурацию автомобиля; 4. Клиент выбирает возможную дату доставки автомобиля; 5. Клиент вводит персональную информацию; 6. Клиент подтверждает введенные данные; 7. Система оформляет заказ покупки; 8. Система возвращает Клиента к странице просмотра корзины заказов. |
| Постусловия:   1. Заказ покупки оформлен. |
| Альтернативные потоки:   1. Отмена заказа покупки; 2. Неудачное оформление заказа покупки. |

|  |
| --- |
| Альтернативный поток: Оформить заявку на покупку автомобиля: Отмена заказа покупки; |
| Краткое описание:  Отмена заказа покупки в процессе оформления заказа. |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия:   1. Выбрана опция отмены заказа. |
| Основной поток:   1. Альтернативный поток начинается на любом шаге (до 7-го) основного потока; 2. Система возвращает клиента на предыдущую страницу. |
| Постусловия:   1. Заказ на покупку не оформлен. |
| Альтернативный поток: Оформить заявку на покупку автомобиля: Неудачное оформление заказа покупки |
| Краткое описание:  Невозможность заказа покупки по причинам ошибок передачи данных. |
| Главные актеры: Клиент. |
| Второстепенные актеры: нет. |
| Предусловия:   1. Введены все данные, необходимые для оформления заказа покупки; 2. Клиент подтвердил введенные данные. |
| Основной поток:   1. Альтернативный поток начинается на шаге 7 основного потока; 2. Система сообщает Клиенту о неудачном оформлении заказа покупки; |
| Постусловия:   1. Заказ самовывоза не оформлен. |

# 2 Анализ

Этап анализа при разработке программного обеспечения является основным этапом при использовании тяжеловесных методологий разработки. На данном этапе полученные на шаге выявления требований функциональные требования к программной системе представляются в виде отношений между классами предметной области. Создается аналитическая модель предметной области путем выявления классов и отношений между ними. При создании аналитической модели крайне важно ограничиться лишь теми классами, которые являются частью словаря предметной области. Это поможет сделать её кратким, простым и понятным описанием поведения системы [3, c. 144-145].

Классы анализа должны представлять собой четкую абстракцию предметной области и должны проецироваться на реальные бизнес-понятия предметной области.

Данная программная система состоит из четырех логических модулей, каждый из которых содержит классы, соответствующие определенным логическим аспектам поведения системы: это классы, содержащие в себе информацию о филиалах города, необходимые для оформления заказов; классы, содержащие информацию о автомобилях, предлагаемых в списках данных филиалов; классы, описывающие заказ, который клиенты могут оформить в существующих заведениях; Общая схема диаграммы классов анализа представлена в приложении В.

## 2.1 Диаграмма классов анализа

На рис. 2 представлена диаграммы классов, описывающая классы, хранящие в себе информацию системы.

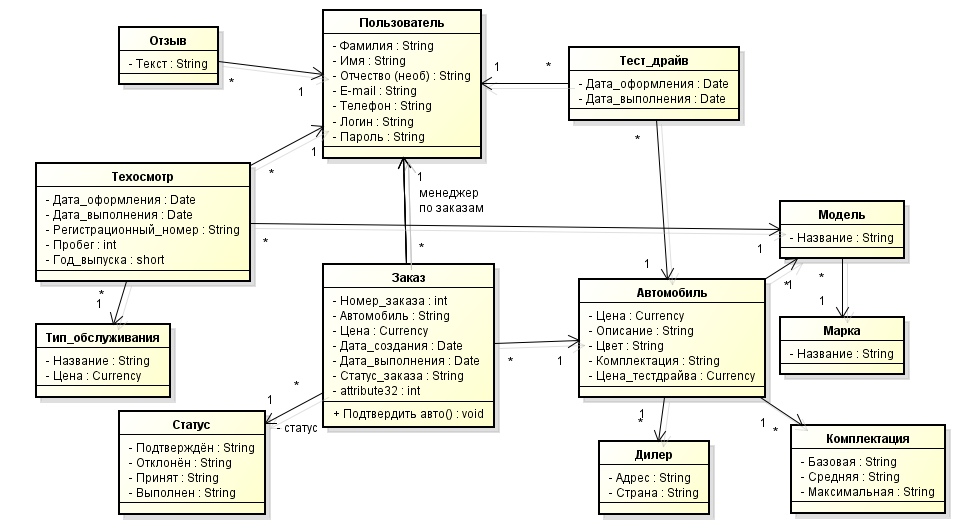


Рис. 2. Диаграмма классов анализа

Как видно на рис. 2 представлены, классы анализа, которые содержит в себе только базисные классы, так называемые классы сущностей, а также отношения между ними.

Основным классом является «Пользователь», содержащая в себе атрибуты:

* Фамилия;
* Имя;
* Электронную почту;
* Логин;
* Пароль;

Под «Пользователем» система понимает, как заказчики(клиенты), так и персонал (в том числе и менеджер по заказам), отвечающий за работу данного автосервиса. «Пользователь» может оставить отзыв, записаться на техническое обслуживание и тест-драйв, а также оформить заказ на покупку автомобиля. Исходя из всего выше сказанного, мы имеем, такие сущности, как «Отзыв», «Техосмотр», «Тест-драйв», «Заказ».

У «Отзыва» есть атрибут «Текст», что в реализации будет представлять мнение клиента.

«Техосмотр» содержит атрибуты – «Дата оформления», «Дата выполнения», «Регистрационный номер», «Пробег», «Год выпуска».

«Тест-драйв» же имеет два свойства: «Дата оформления» и «Дата выполнения».

Эти четыре класса связывает отношение 1: М (включая класс «Заказ»).

В «Техосмотр» входит класс «Тип обслуживания», который является справочником с постоянными значениями и имеет атрибуты, такие как «Номер заказа», «Автомобиль», «Цена», «Дата создания», «Дата выполнения», «Статус заказа». За подтверждение заказов отвечает «Пользователь», т.е. «Менеджер по заказам». Также присутствует метод «Подтвердить авто».

Классы «Статус» и «Автомобиль» соединены внешним ключом с классом «Заказ». «Статус» содержит в себе конкретные значения: «Подтверждён», «Отклонён», «Принят», «Выполнен».

«Автомобиль» имеет атрибуты «Цена», «Описание», «Цвет», «Комплектация». Присутствуют внешние связи с классами «Дилер», состоящего из «Адреса» и «Страны», и «Комплектация», имеющий значения-константы: «Базовая», «Средняя», «Максимальная», а также на справочник «Модель», указывающий модели автомобилей, которые есть в наличии.

«Модель», в свою очередь, соединена со справочником «Марка», где перечислены все марки автомобилей. Определённую (то есть выбранную) «Модель» можно отправить на «Техосмотр», о чём говорит соответствующая связь.

## 2.2 Диаграмма состояний

Классом, изменяющимся динамически, в данной системе является только класс «Заказ», экземпляр которого изменят свое состояние в процессе своего существования в данной системе.

Динамическое изменение состояния класса описывается с помощью диаграммы конечных автоматов. Данные диаграммы могут применяться как для описания жизненного цикла класса анализа, так и для описания поведения проектного класса. Построение диаграммы конечных автоматов применяется обычно на конечных этапах анализа или проектирования, что позволяет максимально подробно уточнить поведение того или иного класса [3, с. 474-475].

На рис. 3 представлена диаграмма конечных автоматов для класса «Заказ» данной системы.



Рис. 3. Диаграмма конечных автоматов класса «Заказ»

Как видно на рис. 3, класс «Заказ» может находиться в 4-х состояниях: на рассмотрении, выполняется, отменен и выполнен.

При оформлении заказа происходит создание экземпляра класса «Заказ», вызывается функция оформления заказа, которая автоматически переводит его в состояние на рассмотрении.

Из состояния заказа «на рассмотрении» перевести в любое другое состояние класс может экземпляр класса «РаботникФилиала», который переводит его либо в состояние «Выполняется», либо в состояние «Отменен». Это зависит от того, смог ли клиент подтвердить то, что он сделал заказ в данном заведении. Это может быть осуществлено с помощью оставленного клиентом номера мобильного телефона, однако это никак не регламентируется в разрабатываемой системе.

Из состояния «Выполняется» заказ может быть переведен в состояние «Выполнен» автоматически, после того, как текущая дата будет больше даты исполнения заказа.

В данной системе присутствуют жесткие ограничения по изменению состояния заказа. Из состояния заказа «Отменен», если он переведен в него сотрудником заведения, заказ может быть только удален, но не переведен в состояние «Выполняется». Для этого клиенту необходимо оформить новый заказ. Так же и из состояния «Выполняется» заказ уже не может быть отменен. Для этого необходимо связаться с клиентом по оставленному им при оформлении заказа номеру.

Из состояний «Отменен» и «Выполнен», в которые каждый из сформированных заказов рано или поздно придет, заказ может быть только удален. Это определяется датой истечения заказа, которая определяется автоматически, исходя из даты исполнения заказа, прибавляя 30 дней к значению этой даты.

Проделанные на данном этапе разработки программной системы действия подготовили основу для перехода к проектированию программной системы, ориентируясь на конкретные языки программирования, платформы и средства разработки. Находясь практически в самом начале цикла разработки программной системы, данный этап является одним из самых важных для тяжеловесных методологий разработки, потому что детальная проработка данного этапа позволяет снизить повышающуюся на дальнейших этапах жизненного цикла продукта стоимость вносимых изменений в готовую программную систему [4, с. 26-28].

# 3 Проектирование

Проектирование определяет, как будет реализована логическая аналитическая модель, которая была создана ранее.

## 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 рисунке 4.

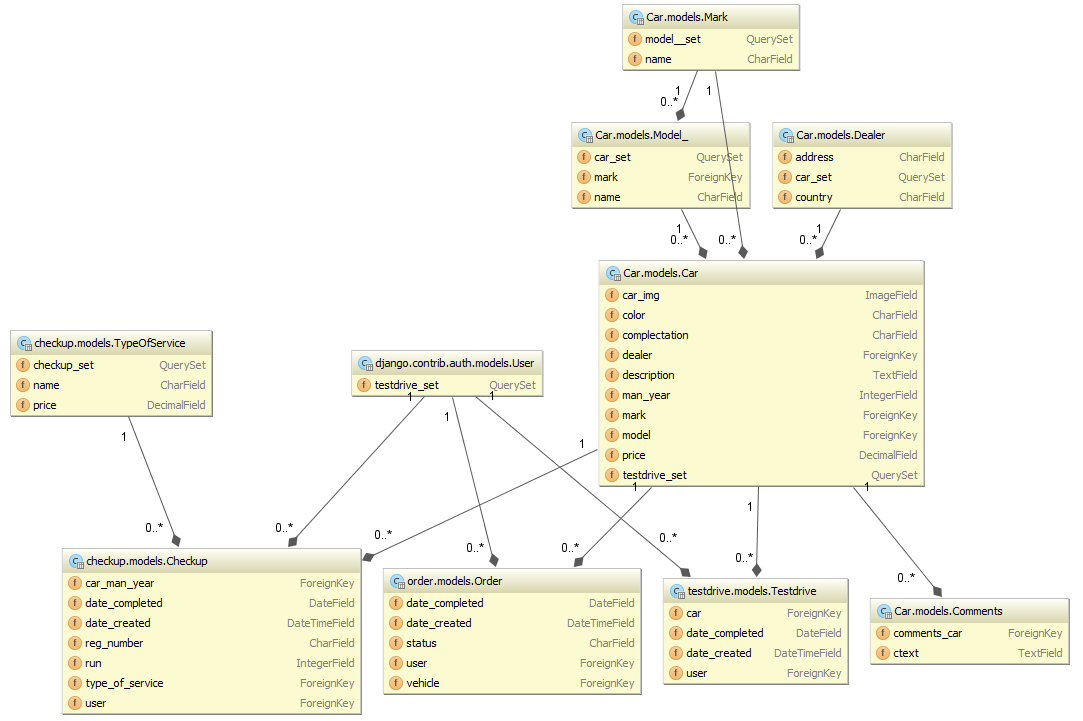


Рис. 4. Диаграмма проектных классов (пересечения убрать)

По сравнению с диаграммой классов анализа произошли следующие изменения: класс «Комплектация» был внесён в класс «Авто», а класс «Статус» – в класс «Заказ». В класс «Авто» был добавлен атрибут «Изображение машины». Появилась возможность получить название модели автомобиля, марки, данные дилера. В целом, диаграмма классов анализа не отличается от диаграммы проектных классов.

## 3.2 Диаграмма пакетов системы

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 пакеты. В этом случае первые называются подпакетами, поскольку все элементы подпакета будут принадлежать более общему пакету. Тем самым для элементов модели задается отношение вложенности пакетов, которое представляет собой иерархию. [5].

Диаграмма пакетов представлена на рис. 5. Она состоит из пакетов, содержащих в себе классы, шаблоны и представления для формирования интерфейса пользователя.

Пакет «Отзыв» состоит только из класса «Отзыв» и зависит только от пакета «Пользователь» (user), состоящего из одноименного класса. Пакет «Авто» состоит из классов «Марка», «Модель», «Дилер», «Автомобиль», от него и от пакета «Пользователь» зависят пакеты «Техосмотр» (checkup) (классы «Техосмотр» и «Тип обслуживания»), «Заказ» (order) (класс «Заказ») и «Тест драйв» (класс «Тестдрайв»).

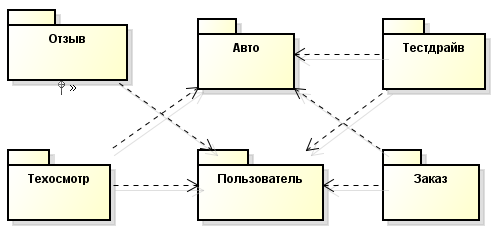


Рис. 5. Диаграмма пакетов программной системы

Пакет «Пользователь» содержит класс, который может просматривать, подтверждать или отменять заказы. Пакеты «Заказ», «Техосмотр», «Тестдрайв» содержат классы, служащие для создания и хранения информации по заявкам

## 3.2 Диаграммы последовательностей для операций проектных классов

Диаграмма последовательности (англ. sequence diagram) — диаграмма, на которой показано взаимодействие объектов (обмен между ними сигналами и сообщениями), упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления [6]. Неотъемлемой частью объекта на диаграмме последовательности является линия жизни объекта. Линия жизни показывает время, в течение которого объект существует в Системе. Периоды активности объекта в момент взаимодействия показываются с помощью фокуса управления. Временная шкала на диаграмме направлена сверху вниз [7].

Были спроектированы диаграммы последовательностей для нескольких вариантов использования. На рис. 6 представлена диаграмма последовательности для варианта использования «Добавить комментарий»

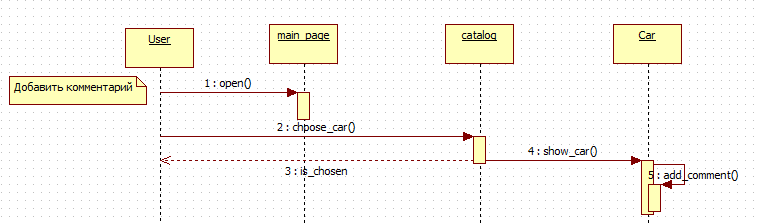


Рис. 6. Диаграмма последовательностей для варианта использования «Добавить комментарий»

Для варианта использования «Добавить комментарий» необходимо пользователю 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 написать свою точку зрения. В текстовом окне пользователь пишет своё мнение и нажимает кнопку «Добавить комментарий», который отображается под описанием машины на странице.



Рис. 7. Диаграмма последовательности для варианта использования «Покупка автомобиля»(ошибки с возвращением)

Сценарий описываемой последовательности сообщений между объектами программной системы начинается с того момента, когда пользователь, авторизированный зашел на главную страницу веб-приложения, после которой переходит в каталог доступных машин, которая его перенаправляет на страницу того автомобиля, который выбрал пользователь. После чего пользователь выбирает купить автомобиль или вернуться к началу каталога. Если выбирает покупку переходит на страницу корзины, на которой осуществляется покупка выбранного автотранспорта. После чего действия завершается, и пользователь переходит на главную страницу.

На рис. 8 изображена диаграмма последовательности для варианта использования «Записаться на техническое обслуживание»



Рис. 8. Диаграмма последовательности для варианта использования «Записаться на тестдрайв»

Для варианта использования «Записаться на тестдрайв» необходимо пользователю открыть главную страницу веб-приложения, с которой мы переходим на страницу услуги, выбрать функцию записаться на тестдрайв и оставить контактную информацию для выполнения заявки сотрудниками автосервиса. После чего действия завершается, и пользователь переходит на главную страницу.

На рис. 9 изображена диаграмма последовательности для варианта использования «Записаться на техническое обслуживание».



Рис. 9. Диаграмма последовательности для варианта использования «Записаться на техническое обслуживание»

Для варианта использования «Записаться на техническое обслуживание» необходимо пользователю открыть главную страницу веб-приложения, с которой мы переходим на страницу услуги, выбрать функцию записаться на техническое обслуживание. Система запрашивает персональные данные, пользователь должен ввести персональные данные. Система сохраняет обращение, после чего пользователь переходит на начальную страницу.

# 4 Реализация

## 4.1 Тестирование

Тестирование веб-приложений представляет собой сложную задачу, так как веб-приложение состоит из нескольких слоев логики – от HTTP запросов на уровне обработки запросов до форм проверки и рендеринга страницы. С помощью тестового фреймворка, встроенного в Django, и нескольких утилит вы можете эмулировать запросы пользователя и проверять какая страница показалась в ответ.

Тестирование – это проверка корректности работоспособности программы, через нахождение соответствия между реальными и ожидаемыми поведением программы, осуществляемые на наборе тестов. Оно помогает доработать продукт.

### 4.1.1 Модульные тесты

Вид тестирования, при котором проверяются модули программы. Идея разбить код программы на отдельные модули. Также модульное тестирование можно считать, как «живое» документирование. Это значит, что клиенты, которые не знают, как работает данный класс, могут разобраться с помощью модульного теста [8].

Файл test.py каждого приложения, который содержит тесты, создается в Django в ходе создания приложения в проекте. В данном файле можно создавать unit- и doc- тесты. В данной курсовой работе были выбраны unit-тесты для тестирования атрибутов и методов, которые их изменяют. Проще говоря, unit-тесты служат для непосредственной проверки заполненных полей, а не поэтапной работы программы.

Для создания unit-теста используется класс TestCase, от которого наследуются все созданные разработчиком классы с тестами.

1. Проверка свойства добавления комментариев к автомобилю

class CommentsTest(TestCase):

def test\_add\_comment\_view(self):

mark1 = Mark.objects.create(name='mark1')

model = Model\_.objects.create(name='model1', mark=mark1)

dealer = Dealer.objects.create(address='address', country='country1')

car = Car.objects.create(price=23.00, description='the best car', color='black', mark=mark1, model=model,

dealer=dealer, man\_year=1990, complectation=Car.COMPLECTATION\_BASE, car\_img='D://win.png')

data = dict(ctext='test comment')

self.client.post(reverse('add\_comment', kwargs=dict(car\_id=car.id)), data=data)

comment = Comments.objects.get()

self.assertEquals(comment.ctext, 'test comment')

self.assertEquals(comment.comments\_car, car)

Данный тест проверяет возможность добавления комментариев. Для их добавления необходимо выбрать машину.

1. Проверка метода, подтверждающего заказ:

class OrderTest(TestCase):

def test\_order\_confirm(self):

user1 = User.objects.create(

username='ooo',

password='999',

email='r@mail.ru',

first\_name='ff',

last\_name='ll'

)

mark1 = Mark.objects.create(name='mark11')

model = Model\_.objects.create(name='model11', mark=mark1)

dealer = Dealer.objects.create(address='address', country='country1')

car = Car.objects.create(

price=23.00,

description='the best car',

color='black',

mark=mark1,

model=model,

dealer=dealer,

man\_year=1990,

complectation=Car.COMPLECTATION\_BASE,

car\_img='win.png'

)

order1 = Order.objects.create(

user=user1,

vehicle=car,

date\_created=datetime(2014, 11, 12).date(),

date\_completed=date(2014, 11, 12),

status=Order.STATUS\_ACCEPTED,

)

order1.confirm()

self.assertEqual(order1.status, Order.STATUS\_CONFIRMED)

В ходе выполнения данного теста проверяется поле status заказа после его подтверждения. Если оно равно Order.STATUS\_CONFIRMED, то метод работает правильно.

1. Проверка получения названия модели:

class ModelTest(TestCase):

def test\_model\_name(self):

mark1 = Mark.objects.create(name='mark1')

model = Model\_.objects.create(name='model1', mark=mark1)

self.assertEqual(

model.\_\_str\_\_(),

'mark1 model1', "Incorrect name of model"

)

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

### 4.1.2 Интеграционные тесты

Интеграционные тесты в Django создаются с помощью класса WebTest, который является наследником TestCase, который нужен для модульного тестирования. Основное отличие TestCase и WebTest это переменная self.testapp из DjangoTestApp. Она позволяет получить доступ к API WebTest.

Важно понимать, что интеграционные тесты — это не замена юнит-тестам, а только дополнение к ним, и что 100% покрытие никак не гарантирует отсутствия ошибок. Юнит-тесты — точные, они говорят, что именно поломалось, они крайне полезны при рефакторинге и в сложных местах проекта.

Чтобы показать различие: в unit-тесте для формы регистрации мы бы создали объект для регистрации любого пользователя по e-mail, передавали бы в него разные словари с данными и смотрели бы, какие вызываются исключения. Юнит-тесты максимально приближены к листингу, тестируют отдельный его метод или атрибут, и позволяют проверить, что все части системы по отдельности работаю правильно. Интеграционные тесты помогают проверять, что и вместе они работают тоже правильно [9].

### 4.1.3 Построение и выполнение тестов

Для одновременного выполнения всех тестов используется утилита manage.py test. Для обращения к какому-то определенному приложению и проверки его тестов, можно использовать python manage.py test <application\_name>.

Самое интересное в тестах Django версии 1.7 и выше – это то, что они не используют настоящую базу данных. Для тестов Django создает в оперативной памяти базу данных специально для тестов. И поля в ней заполняются теми данными, которые были использованы в тестах [10].

Запустим тесты во всем приложении:

C:\Car\_Shop>python manage.py test

Creating test database for alias 'default'...

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

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

Ran 4 tests in 0.084s

OK

Destroying test database for alias 'default'..Затраченное время: 00:00:8.935

Из результатов запуска тестов видно, что все тесты, которых было запущено 4, прошли успешно.

### 4.1.4 Покрытие кода

Это инструмент для проверки покрытия кода тестами. Оно выражается в процентном соотношении тестированных полей ко всем полям приложения. Покрытие кода используется для получения набора тестов при регрессивном тестировании, потому что в ходе тестирования выявляются недостатки, а также закрываются непокрытые участки кода.

В python для создания покрытия тестами используется пакет coverage. С её помощью можно получить отчет о покрытие в виде html-файла, xml-файла, вывести в консоль или текстовый файл.

Некоторые тестировщики считают, что делать 100% покрытие кода unit-тестами не нужно, так как это бесполезно и тратит время. Но, чтобы добиться защищенного и качественно работающего приложения, потому что приложения в современном мире постоянно обновляются, соответственно, после каждого обновления поля, которые были созданы в более старых версиях, могут работать некорректно. А при полном покрытии кода все возможные ошибки будут найдены.

### 4.1.5 Запуск приложения для тестирования

Запуск приложения осуществляется на сервере разработки. Этот сервер служит непосредственно для разработки приложения. Ему не нужно много ресурсов, в отличие от сервера для непосредственной работы приложения Apache Tomcat, например.

Сервер запускается на 127.0.0.1:8000. Запуск и отладка проекта на локальном тестовом сервере для разработки необходим для выявления правильности работы приложения, тестирования пользовательского интерфейса и отладки кода программы.

## 4.2 Непрерывная интеграция

Непрерывная интеграция (Continuous Integration, CI) – это автоматическая сборка проекта из исходного кода, при этом дополнительно возможен запуск автоматических тестов, проверка качества кода, развёртывание приложения на тестовом сервере (для серверных приложений).

Сборка проекта происходит при изменении исходного кода проекта или/и по расписанию.

В результате достигается:

* раннее обнаружение ошибок,
* автоматизация рутинных операций,
* как результат – экономия времени и повышение качества кода.

Особенно актуально использование непрерывной интеграции если над проектом работает несколько человек - сразу становится понятно кто, что и когда сломал. Организовать процесс автоматической сборки приложения не так уж сложно - для этого существует множество готовых продуктов [11].

В качестве системы непрерывной интеграции выбран сервис Travis CI, который поддерживается системой контроля версий на бесплатном (для публичных репозиториев) хостинге GitHub. Travis CI – это распределенный веб сервис для сборки и тестирования проектов по разработке программного обеспечения на более чем 20-ти языках программирования, использующих GitHub в качестве хостинга исходного кода [12].

Для того, чтобы осуществить привязку своего репозитория на GitHub необходимо быть хозяином этого репозитория.

Прежде всего необходимо войти в учетную запись Travis CI через аккаунт на GitHub. Для этого нужно с сайта сервиса (<https://travis-ci.org/>) нажать на кнопку “Sign in with GitHub”.

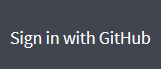


Рис. 10. Кнопка входа в учетную запись Travis CI

Следующим шагом является активация синхронизации репозитория проекта на GitHub с сервисом Travis CI. Для этого необходимо зайти в настройки репозиториев профиля на Travis CI и включить работу сервиса с выбранным репозиторием с помощью переключателя.

Следующим шагом, указывающим сервису Travis то, как работать с репозиторием, является добавление в корневую директорию репозитория файла .travis.yml. Данный файл содержит скрипт, указывающий сервису то, как настроен репозиторий проекта: на каком языке написан исходный код, что нужно выполнить перед сборкой проекта, какие скрипты нужно выполнить после сборки проекта и другие опции, о которых можно узнать на сайте Travis CI [13].

Содержание файла .travis.yml для данного курсового проекта представлено ниже.

language: python

python:

- "3.4"

install: "pip install -r requirements.txt"

script:

- "flake8"

- "coverage run manage.py test"

- "coverage report"

notifications:

email:

on\_failure:

- khmv1@tpu.com

Как видно из содержания файла, в нем указывается, что проект написан на языке python. Это необходимо для того, чтобы Travis смог правильно интерпретировать проект и создать необходимое виртуальное окружение на сервере для сборки проекта. Далее указывается версия интерпретатора python, необходимая для выполнения установок и скриптов на сервере.

После этого указываются секции, которые выполняет Travis CI в проекте. В секции install он выполняет скрипт по установке необходимых зависимостей для корректной работы программной системы. В секции scripts указываются скрипты, которые также необходимо выполнить после сборки проекта и установки всех зависимостей. В данном случае выполняется скрипт flake8, проверяющий стиль написания кода на python (настройки для выполнения данного скрипта находятся в файле проекта tox.ini), затем запускаются тесты и ведется подсчет процента покрытия кода тестами с помощью команд coverage.

При возникновении ошибок сборки Travis CI высылает на почту пользователя, который сделал коммит, на котором возникла ошибка, уведомление о том, что его коммит привел к нарушению работы проекта. В файле Travis CI данного проекта также указывается e-mail, на который, в случае возникновении ошибок, всегда будет высылаться уведомление.

После настройки файла .travis.yml для проверки того, работает ли Travis CI с репозиторием необходимо зайти в настройки репозитория, во вкладку «Webhooks and Services» и выбрать сервис Travis CI, на открывшейся странице нужно нажать на кнопку «Test service» (Рис. 11), чтобы убедится в работоспособности сервиса с вашим репозиторием.



Рис. 12. Тестирование сервиса Travis CI на GitHub

Сама по себе такая проверка не инициирует сборку проекта. Для запуска построения проекта необходимо сделать коммит и отправить его в репозиторий на сервер GitHub.

Увидеть то, как прошла сборка и логи сборки можно на сайте Travis CI, зайдя под своей учетной записью. Если все команды завершились с кодом 0, то сборка проекта считается выполненной успешно и её присваивается статус «Passed», в противном случае ей присваивается статус «Failed».

# 5 Документация

## 5.1 Назначение программы

Разработанный проект «ООО "Авто-импорт"» предназначен для оформления заказов на покупку, техосмотр и тестдрайв автомобиля.

В частности данный сайт позволяет:

* Просматривать информацию различных автомобилей;
* Добавлять понравившейся автомобиль в корзину, изменять конфигурацию автомобиля, удалять выбранный автомобиль из корзины;
* Оформлять заказы на выбранный список услуг;
* Отслеживать состояние выполнения заказов;
* Просматривать и обрабатывать списки заказов.

Сайт «ООО "Авто-импорт"» рассчитан на людей, имеющих доступ в интернет и желающих получать детальную информацию о понравившемся автомобиле и осуществлять заказы на покупку и различных видов услуг, не выходя из дома.

## 5.2 Условия запуска программы

Для работы на сайте пользователю необходимо иметь персональный компьютер с возможностью выхода в интернет со скоростью не менее 256 КБ/с. Также необходим установленный браузер и возможность ввода.

В ходе самой эксплуатации есть ряд ограничений, такие как:

* При оформлении заказа обязательными являются все поля;
* При оформлении заказа нельзя указывать дату и время, позже, чем текущее время + 2 часа – это время, необходимое для обработки заказа сотрудником заведения;
* При оформлении заказа номер телефона не может состоять из букв, а длина номера должна быть равна 10 цифрам.

## 5.3 Выполнение программы

Сервис выполняет следующие функции:

### 5.3.1 Авторизация и регистрация пользователей

Для прохождения авторизации необходимо кликнуть по кнопке «Войти» в правом нижнем углу любой страницы (Рис. 12).



Рис. 12. Кнопка «Войти»

Затем открывается новая страница с полями: Логин и Пароль. Эта страница показана на Рис. 13.

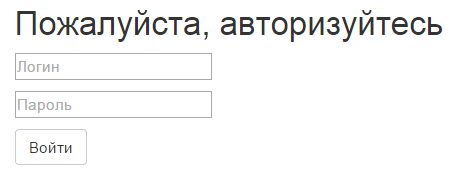


Рис. 13. Страница авторизации для работника заведения

Каждое поле является обязательным. После заполнения каждого поля, Работник заведения нажимает кнопку «Войти», после чего он получает доступ к другим функциям сайта.

### 5.3.2 Просмотр списка заказов зарегистрированным пользователем

После авторизации пользователь имеет доступ к странице заказов. Здесь же заказ можно принять или отменить (Рис. 14).

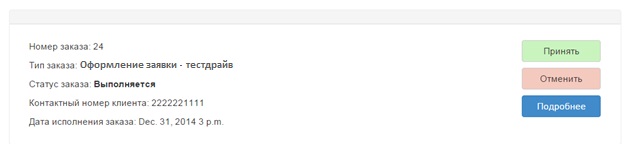


Рис. 14. Страница списка заказов

### 5.3.3 Выбор автомобилей по маркам

Все модели относятся к определенной марки авто, список марок представлен на Рис. 15.

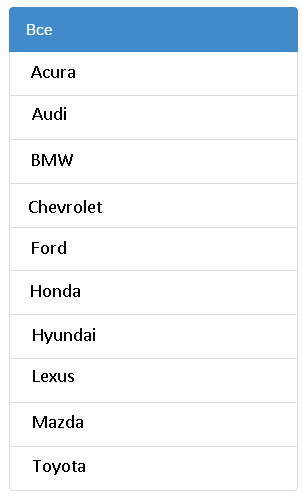


Рис. 15. Список марок автомобилей

Клиент может выбрать марку (по умолчанию «Все»). Список моделей автомобилей фильтруется согласно выбранной марки.

### 5.3.4 Добавление автомобиля в корзину

Для добавления автомобиля в корзину необходимо кликнуть по кнопке «Добавить в корзину» (Рис. 16).

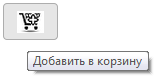


Рис. 16. Кнопка «Добавить в корзину»

При этом один автомобиль выбранной модели будет добавлен в корзину.

### 5.3.5 Работа с корзиной

Для того чтобы перейти в корзину заказов, необходимо кликнуть по иконке корзина, или ссылке с соответствующей надписью (Рис. 27).

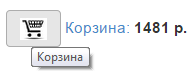


Рис. 17. Кнопка и текстовая ссылка «Корзина»

Сама корзина содержит в себе список заказов, (Рис. 28).

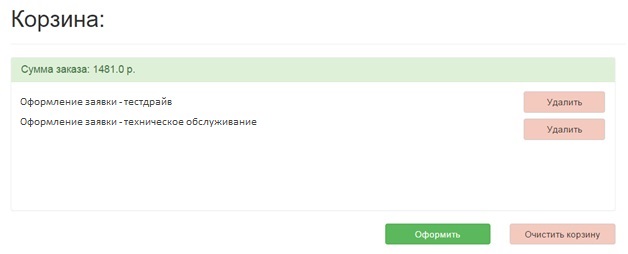


Рис. 18. Корзина заказов

В корзине клиенту предоставляется возможность удаления выбранных услуг.

Также возможна полная очистка корзины. Когда выбор сделан, клиенту необходимо кликнуть по кнопке «Оформить».

### 5.3.6 Формирование заказов

После переход по ссылке «Оформить» корзины заказов система на основе данных корзины формирует один или несколько заказов для клиента. Клиент может посмотреть детальную информацию по каждому заказу (Рис. 29, кнопка «Подробнее»).

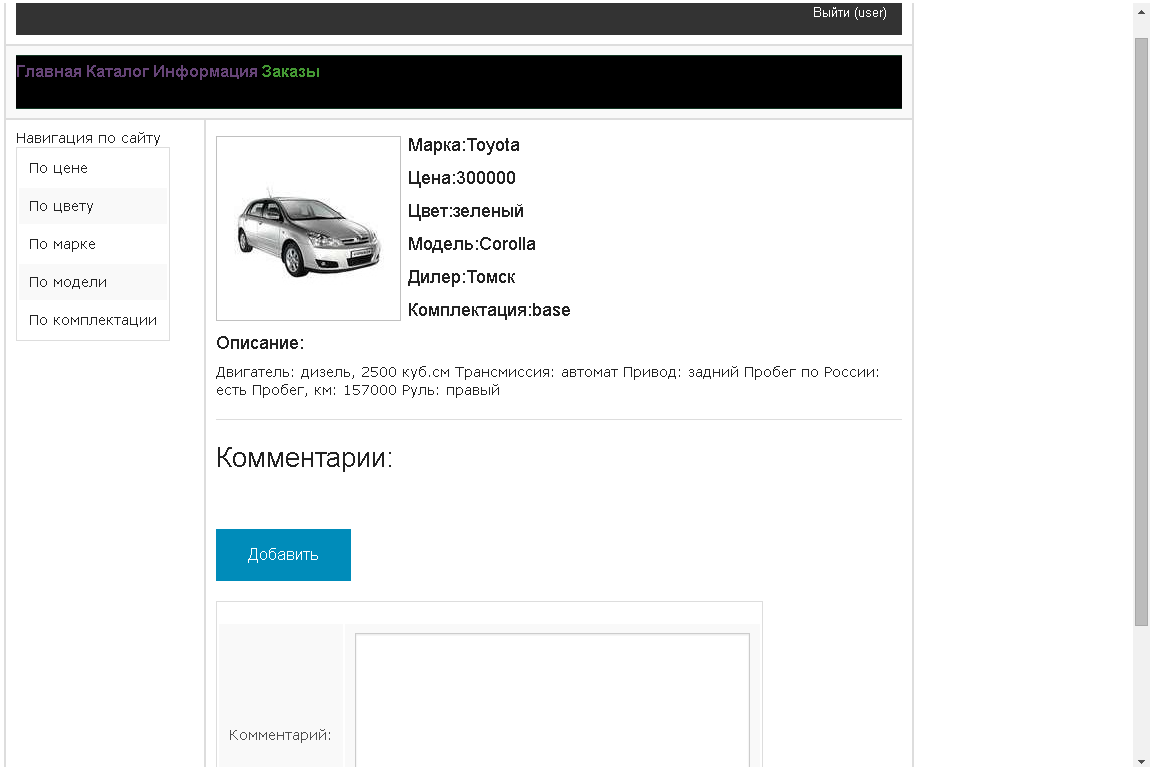


Рис. 19. Детальная информация о заказе

### 5.3.7 Оформление заказа

После перехода по ссылке оформления заказа клиент переадресовывается на страницу «Оформление заказа». Здесь необходимо выбрать тип заказа: «Тест драйв», «Техническое обслуживание» или. Если клиент выбрал «Тест драйв», то ему необходимо заполнить ряд полей формы (Рис. 20). Часть полей имеют значения по умолчанию, но дату бронирования, время и контактный телефон пользователю необходимо заполнить в любом случае.

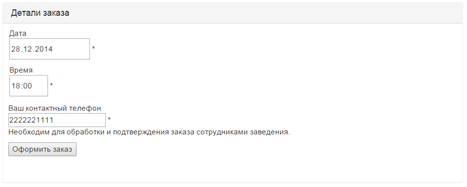


Рис. 20. Форма заказа «Тест драйв»

Если клиент выбрал «Заказ технического обслуживания» (Рис. 21), то ему также необходимо выбрать дату, время и контактный телефон.

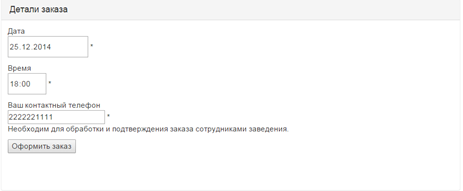


Рис. 21. Форма заказа «Заказ технического обслуживания»

Если оформление заказа прошло успешно, пользователь получит уведомление (Рис. 22).

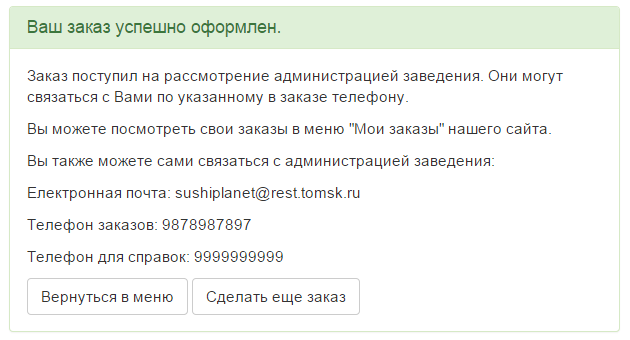


Рис. 22. Сообщение об успешном оформлении заказа

## 5.4 Сообщения программы

При выполнении различных операций на сайте, могут возникать сообщения обозначающие неверные действия пользователя. Некоторые поля являются обязательными или на них стоят иные ограничения, и, при не заполнении этих полей или вводе некорректных данных, пользователь получает соответствующие уведомление.

Так, например, если поля Логин или Пароль пусты, а работник заведения пытается авторизоваться, то система выдаст предупреждение (Рис. 36).

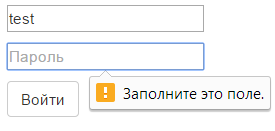


Рис. 23. Предупреждение о пустом поле

Если комбинация Логин-Пароль была указана неверно, на странице появится оповещение об этом (Рис. 37).

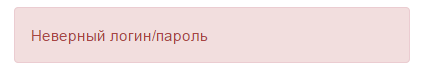


Рис. 24. Оповещение о неверной комбинации Логин-Пароль

Если при оформлении заказа одно или несколько обязательных полей были не заполнены, клиент получит уведомление об этом напротив некорректно заполненного поля (Рис. 25).



Рис. 25. Пример сообщения о некорректном вводе данных на формах оформления заказов

# Заключение

Итогом выполнения данной курсовой работы стало приложение автомобильного интернет-магазина, разработанное с использованием языка программирования Python и фреймворка 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.

При соответствующих доработке и улучшении данный программный продукт может быть внедрен для практического применения, так как рассматривалась реальная предметная область. 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. Продажа авто в Томске [Электронный ресурс] / auto.ru– Электрон. Дан. – URL: http://tomsk.auto.ru/, свободный (дата обращения 11.12.2014).
2. Почему Django [Электронный ресурс] / OLDMIN TEAM – создание, поддержка и обслуживание сайтов – Электрон. Дан. – URL: <http://oldmin.org/ru/article/pochemu-django/>, свободный (дата обращения 16.12.2014).
3. 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.
4. Макконнелл С. Совершенный код – Пер. с англ. – СПб: Питер, 2007. – 896 с.
5. Пакеты в языке UML [Электронный ресурс] / Портал знаний – Электрон. Дан. – URL: http://www.znannya.org/?view=Packages\_language\_UML, свободный (дата обращения 16.12.2014).
6. Диаграмма последовательности [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. Дан. – URL: <http://ru.wikipedia.org/wiki/Диаграмма_последовательности>, свободный (дата обращения 10.12.2014).
7. Теория и практика UML. Диаграмма последовательности [Электронный ресурс] / IT-GOST.RU – электронная библиотека стандартов оформления проектной документации – Электрон. Дан. – URL: <http://it-gost.ru/articles/view_articles/94>, свободный (дата обращения 10.12.2014).
8. Модульное тестирование [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. Дан. – URL: <http://ru.wikipedia.org/wiki/Модульное_тестирование>, свободный (дата обращения 10.12.2014).
9. Пишем функциональные/интеграционные тесты для проекта на django [Электронный ресурс] / Хабрахабр – самое крупное в Рунете сообщество людей, занятых в индустрии высоких технологий – Электрон. Дан. – URL: <http://habrahabr.ru/post/91471/>, свободный (дата обращения 19.12.2014).
10. Пишем функциональные/интеграционные тесты для проекта на django [Электронный ресурс] / Документация Django 1.4 (читать 1.7) – Электрон. Дан. – URL: <http://djbook.ru/rel1.4/topics/testing.html#the-test-database>, свободный (дата обращения 20.12.2014).
11. Что такое Continious Integration [Электронный ресурс] / Apache maven project – Электрон. Дан. – URL: http://www.apache-maven.ru/tools/continuous-integration.html, свободный (дата обращения 20.12.2014).
12. Travis CI [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. дан. – URL: <https://ru.wikipedia.org/wiki/Travis_CI/>, свободный (дата обращения: 10.12.2014).
13. Building a Python Project [Электронный ресурс] / Travis CI: Travis CI Documentation– Электрон. дан. – URL: <http://docs.travis-ci.com/user/languages/python/>, свободный (дата обращения: 10.12.2014).

# Приложение А

Расчет КТУ

Заполнил А.В. Быков Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Быков | Рогова | Холманский | Чеботарева |
| Быков | 1 | 1 | 0 | 1 |
| Рогова | 1 | 1 | 0 | 1 |
| Холманский | 2 | 2 | 1 | 2 |
| Чеботарева | 1 | 1 | 0 | 1 |

Заполнила Е.В.Рогова Таблица 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Быков | Рогова | Холманский | Чеботарева |
| Быков | 1 | 1 | 0 | 1 |
| Рогова | 1 | 1 | 0 | 1 |
| Холманский | 2 | 2 | 1 | 2 |
| Чеботарева | 1 | 1 | 0 | 1 |

Заполнил М.В. Холманский Таблица 3

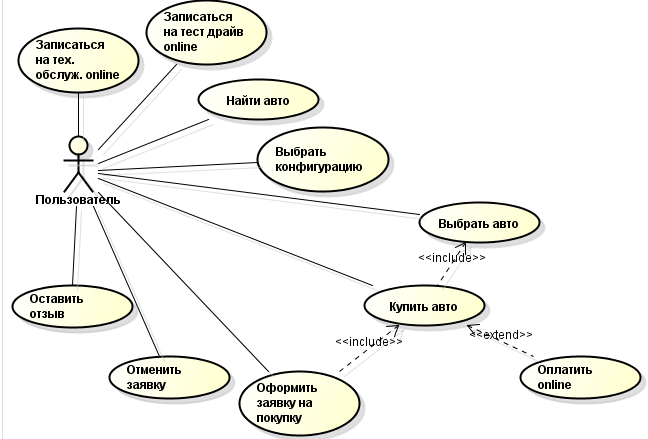
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Быков | Рогова | Холманский | Чеботарева |
| Быков | 1 | 1 | 0 | 1 |
| Рогова | 1 | 1 | 0 | 1 |
| Холманский | 2 | 2 | 1 | 2 |
| Чеботарева | 1 | 1 | 0 | 1 |

Заполнила Е.Н. Чеботарева Таблица 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Быков | Рогова | Холманский | Чеботарева |
| Быков | 1 | 1 | 0 | 1 |
| Рогова | 1 | 1 | 0 | 1 |
| Холманский | 2 | 2 | 1 | 2 |
| Чеботарева | 1 | 1 | 0 | 1 |

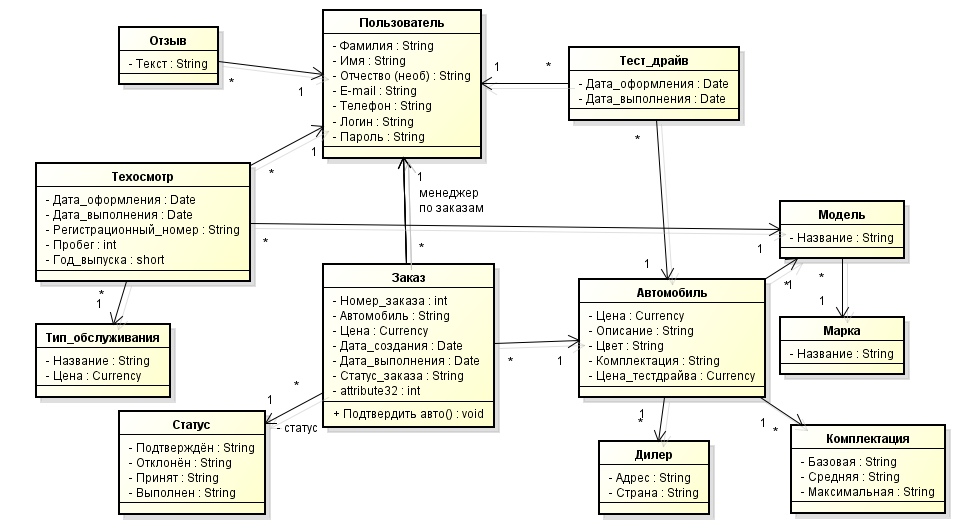
# Приложение Б

Диаграмма вариантов использования



# Приложение В

Диаграмма классов анализа



# Приложение Г

Диаграмма проектных классов

