

Министерство образования и науки

донецкой народной республики

Государственное образовательное учреждение

высшего профессионального образования

«донецкий национальный университет»

Физико-технический факультет

Кафедра компьютерных технологий

Направление подготовки 09.03.01 Информатика и вычислительная техника

**К защите допустить:**

Зав. кафедрой компьютерных технологий

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Ермоленко

(подпись)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**ДИПЛОМНАЯ работа**

на тему: **«Разработка информационной подсистемы учета клиентов салона красоты»**

Студентка: Круподер Александра Сергеевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель: ст. преп. **Бодряга В.В.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Работа представлена на кафедру «\_\_\_» \_\_\_\_ 2017 г. рег. № \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись принявшего)

Донецк 2017

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ, ТЕРМИНОВ 3](#_Toc484761988)

[ВВЕДЕНИЕ 4](#_Toc484761989)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc484761990)

[1.1 Состояние вопроса 5](#_Toc484761991)

[1.2 Актуальность и цель работы 8](#_Toc484761992)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 9](#_Toc484761993)

[2.1 Общие требования к продукту 9](#_Toc484761994)

[2.2 Требование к пользовательским интерфейсам 10](#_Toc484761995)

[2.3 Требования к программным интерфейсам 14](#_Toc484761996)

[2.4 Пользователи продукта и требования к ним 14](#_Toc484761997)

[2.5 Функции продукта 14](#_Toc484761998)

[2.6 Сценарии использования продукта 15](#_Toc484761999)

[2.7 Ограничения 16](#_Toc484762000)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 17](#_Toc484762001)

[3.1 HTML5 17](#_Toc484762002)

[3.2 CSS3 17](#_Toc484762003)

[3.3 JavaScript 18](#_Toc484762004)

[3.5 Django framework 19](#_Toc484762005)

[3.6 PyCharm IDE 19](#_Toc484762006)

[3.7 SQLite 20](#_Toc484762007)

[4 РАЗРАБОТКА ПО 21](#_Toc484762008)

[4.1 Создание проекта веб-приложения 21](#_Toc484762009)

[4.2 Проектирование БД 22](#_Toc484762010)

[4.3 Настройка URL-менеджера 28](#_Toc484762011)

[4.4 Создание шаблонов 29](#_Toc484762012)

[4.5 Создание представлений 34](#_Toc484762013)

[4.6 Интерфейс администратора Django 36](#_Toc484762014)

[5 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА 39](#_Toc484762015)

[5.1 Структура проекта 39](#_Toc484762016)

[5.2 Описание объектов и их взаимодействия 41](#_Toc484762017)

[6. Тестирование и внедрение 43](#_Toc484762018)

[6.1 Тестирование разработанного ПО 43](#_Toc484762019)

[6.2 Установка программы 45](#_Toc484762020)

[ЗАКЛЮЧЕНИЕ 47](#_Toc484762021)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc484762022)

[ПРИЛОЖЕНИЕ А 49](#_Toc484762023)

[ПРИЛОЖЕНИЕ Б 53](#_Toc484762024)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ, ТЕРМИНОВ

MVC (Model-View-Controller) — логика доступа к данным, бизнес-логика и логика отображения — составляют концепцию, которую называют шаблоном Модель-Представление-Управление архитектуры программного обеспечения. В этой концепции термин «Модель» относится к логике доступа к данным; термин «Представление» относится к той части системы, которая определяет, что показать и как; а термин «Управление» относится к той части системы, которая определяет какое представление надо использовать, в зависимости от пользовательского ввода, по необходимости получая доступ к модели.

Сервер – любая система, процесс, компьютер, которые обладают произвольным вычислительным ресурсом (памятью, временем и т.д.).

Framework – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

# ВВЕДЕНИЕ

Построение и развитие информационного общества признается ведущей мировой тенденцией XXI в., которая определяет необходимость формирования глобальных информационного и экономического пространств. Новая технологическая парадигма информационного общества базируется на электронике и генной инженерии, а ее основой является информационные системы и технологии. Эта парадигма заменяет парадигму индустриального общества.

В современных условиях эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Следовательно, повышение эффективности управленческой деятельности становится одним из направлений совершенствования деятельности предприятия в целом.

В настоящее время в городах очень популярны сети салонов красоты. Все эти салоны объединяет одна проблема - отсутствие какой-либо автоматизации процессов. На данный момент большинство операции по записи, услугам, заказу продукции происходят вручную. Учет обычно происходит при помощи журнала, что очень сильно сказывается на времени обслуживания, поэтому целью ВКР стало решение данных проблем на примере создания информационной подсистемы учета клиентов салона красоты.

Исходя из указанной цели, можно выделить частные задачи, поставленные в ВКР: проанализировать предметную область, существующие аналоги и их реализацию, составить ТЗ, разработать структуру и функционал, соответствующий техническому заданию, протестировать разработанное приложение. Для выполнения поставленных задач следует изучить технологии: HTML, CSS, Java Script, Pyhton, Django, PyCharm, SQLite.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Состояние вопроса

Современные салоны красоты предлагают довольно широкий спектр услуг, стремясь, чтобы клиент получил все желаемые процедуры в одном месте. Ассортимент предлагаемых услуг постоянно расширяется, ведь теперь люди не только делают стрижку, ухаживают за ногтями и волосами, но и проводят процедуры по омоложению и коррекции организма, борются со стрессами и негативными воздействиями окружающей среды. Клиенты пришли к пониманию, что домашний уход по эффективности ни в какое сравнение не идет с салонными процедурами. К тому же в салоне экономится время и силы, добавляются положительные эмоции от общения.

Для успешной работы и занятия уверенных позиций на рынке индустрии красоты необходимы не только качественные процедуры, но и постоянное управление процессами, создание условий для привлечения клиентов, их удержания, четкий учет материалов, расходуемого времени, отслеживание каждого клиента, чтобы избежать его ухода. Именно автоматизация позволяет построить весь процесс качественно, чтобы клиентам вновь и вновь захотелось бы приходить в салон красоты, а квалифицированный персонал не думал о смене места работы.

Преимущества информатизации медицинских бизнес-процессов уже оценили ведущие игроки индустрии красоты крупнейших городов СНГ. Соответствовать курсу на всеобщую автоматизацию бизнеса стремятся также и небольшие частные салоны, и фитнес-центры в городах Сибири.

Информатизация подразумевает под собой перенос некоторых процессов в информационную сферу. Например, многие крупные и не очень салоны красоты создают собственные сайты, в которых можно ознакомится со всеми услугами предприятия, что избавляет от необходимости, например, создавать и распространять брошюры.

С целью выявления недостатков и достоинств, уточнения терминологии, был проанализированные один из существующих аналогов: «1С-Рарус: Комильфо: Салон красоты» [1]. Решение «1С-Рарус: Комильфо: Салон красоты» это не просто бухгалтерский учет деятельности - но и активный инструмент маркетинга, контроля персонала, удержания постоянных и привлечения новых клиентов.

Программный продукт «1С-Рарус: Комильфо: Салон красоты» разработан на платформе “1С: Предприятие 8” и предназначен для автоматизации оперативного и финансового учета в салонах красоты, SPA - салонах, парикмахерских, массажных кабинетах, соляриях и других заведениях индустрии красоты и здоровья. Решение объединяет в себе функционал Фронт и Бэк - офиса. Программный продукт может применяться для автоматизации как одиночных, так и сетей предприятий.

Основная решаемая задача программного комплекса - ведение базы клиентов, назначение для них визитов, оформление оказания услуг, продажи товаров, расчет заработной платы персонала. Ведется полный цикл взаиморасчетов с клиентом от момента записи до расчета за оказанные услуги. Ведется учет движения товаров и материалов по складам и ответственным лицам. Основные возможности:

* Учет клиентов (ведение базы с контактной информацией и историей взаимоотношений, назначение визита, оформление продажи)
* Учет материалов (оформление операций по поступлению, перемещению, списанию, в том числе и автоматическому в ходе оказания услуг)
* Учет работы сотрудников (формирование графиков работы, контроль занятости, автоматическое переназначение мастеров, в случае их невыхода на работу)
* Выписка авансовых отчетов
* Управление движением денежных средств: банк и касса
* Оформление розничных продаж
* Управление правами пользователей

Кроме этого, система Комильфо имеет дополнительные возможности в реализации:

* Возможность автоматического запуска интерфейса «Рабочее место администратора салона» при начале работы пользователя в системе
* Быстрое начисление заработной платы по любой схеме учета
* Учет скидок (разовые, накопительные и периодические с возможностью отбора по полу, возрасту, семье и определенным услугам или товарам)
* Отчеты по деятельности салона (по материалам, по клиентам, по мастерам, по скидкам, по прибыли и многие, многие другие).
* Автоматическое формирование прайс-листов собственного салона и прейскуранта цен поставщиков
* Учет выдачи подарочных сертификатов, выдаваемых салоном на суммы и услуги
* Контроль взаиморасчетов с клиентами и сотрудниками
* Возможность работы сотрудников на нескольких должностях одновременно
* Создание распределенной информационной базы по нескольким салонам

Решение «Комильфо: Салон красоты» интересно для всех организаций оказывающих косметические, косметологические, парикмахерские и другие услуги в области красоты и здоровья. Ведется полный цикл взаиморасчетов с клиентом от момента записи до расчета за оказанные услуги. Ведется учет движения товаров и материалов по складам и ответственным лицам.

Сканеры штрих-кодов, считыватели магнитных карт позволяют наладить систему автоматических дисконтных скидок для постоянных клиентов, причем система допускает гибкий механизм различных скидок и привилегий, в том числе и накопительных. Поддерживается более десятка видов торгового оборудования, и список его постоянно расширяется.

В процессе анализа было выявлено, что данная система предоставляет большее количество услуг, многие из которых не нужны небольшим предприятиям, а также является платной [2].

## 1.2 Актуальность и цель работы

Одним из наиболее прогрессирующих направлений в области информационных технологий является разработка и внедрение информационных систем с использованием современных средств управления базами данных. Развитие компьютерной техники и, что более важно, развитие коммуникационных технологий, в последнее время является мощным двигателем использования информационных систем.

С внедрением информационной системы процессы работы салона красоты очень сильно упростятся. Многие действия, которые требовали внимания и отнимали время персонала будут автоматизированы. Все это упростит процесс и избавит менеджеров от ненужной работы. Авторизация позволит предотвратить несанкционированный доступ из вне.

Целью курсовой работы является разработка информационной подсистемы учета клиентов салона красоты. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Ознакомится с предметной областью.

2. Проанализировать существующие аналоги и их реализацию.

3. В качестве результата анализа составить ТЗ для приложения.

4. Изучить технологии: HTML, CSS, Java Script, Pyhton, Django, PyCharm, SQLite.

5. Разработать структуру и функционал веб-приложения, соответствующий техническому заданию.

6. Разработать клиентскую и серверную части программы.

7. Проанализировать разработанное веб-приложение.

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 2.1 Общие требования к продукту

В рамках ВКР требуется разработать информационную подсистему учета клиентов салона красоты. Для примера, следует выбрать салон спа-услуг. Подсистема должна быть реализована в качестве веб-сайта, данный выбор позволит проводить учет клиентов салона красоты из любого места где есть интернет. Среди функций подсистемы должны быть:

* Добавление записи о посещении
* Удаление записи
* Просмотр записей в виде таблицы
* Сортировка по отдельным полям
* Поиск по ключевым словам

Все данные приложения должны храниться в базе данных в виде таблиц. Записи о посещениях в базе данных должны учитывать следующие параметры:

* ФИО посетителя
* Дату и время посещения
* Услугу, которой воспользовался посетитель
* Работника, который принял посетителя
* Заметки о посещениях
* Идентификатор записи

База данных должны хранить данные о работниках и услугах сап-салона, о новостях. Большинство данных будут информативны и для клиентов салона, поэтому следует реализовать интерес не только для администраторов (работников салона) но и для клиентов, что позволит потенциальным клиентам ознакомиться с услугами и персоналом заведения.

## 2.2 Требование к пользовательским интерфейсам

Интерфейсом веб-приложения является веб-страницы сайта. При их создании в дизайне следует придерживаться простоты и понятности, а также текущих стандартов интерфейса веб-страниц.

В целом, веб страницы сайта придерживаться одного шаблона, который изображён на рисунке 1 Данный шаблон состоит из двух частей веб-страницы, который не меняют своего содержания от перехода по веб-страницам: шапки и подвала. Шапка сайта содержит панель навигации (1.1) для перехода по страницам и логотип сайта (1.2). Панель навигации должна вкладку, которая отвечает за текущую страницу. Подвал сайта должен содержать ссылки на группы в социальных сетях и ссылку на панель администратора с левой стороны (3.1) и телефонный номер справочной – справа (3.2). Основная часть (2) должна содержать основный контент.

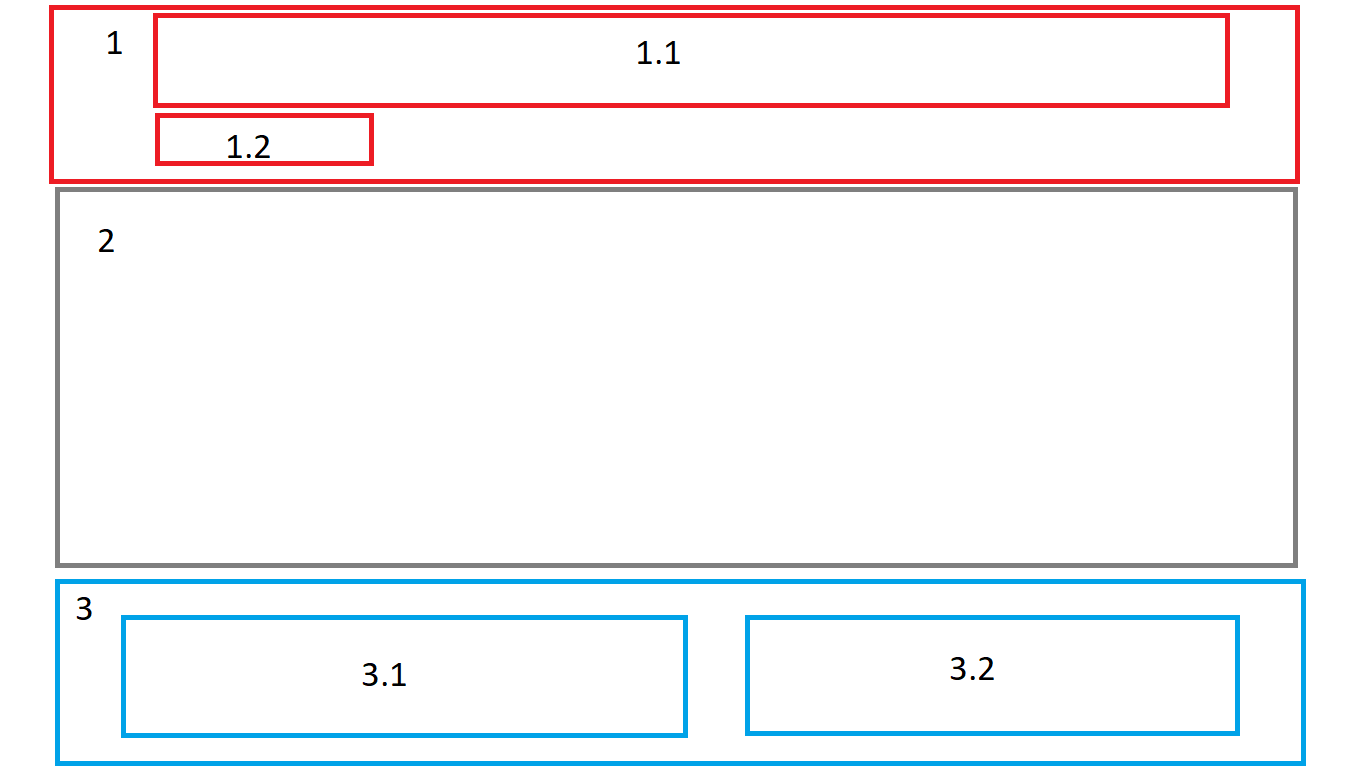


Рисунок – Базовый шаблон веб-страниц

Основной контент сайта можно разбить на четыре части: Стартовая страница, страница услуг, страница с информацией о сотрудниках и страница с контактами для связи.

Схема стартовой страницы сайта изображена на рисунке 2

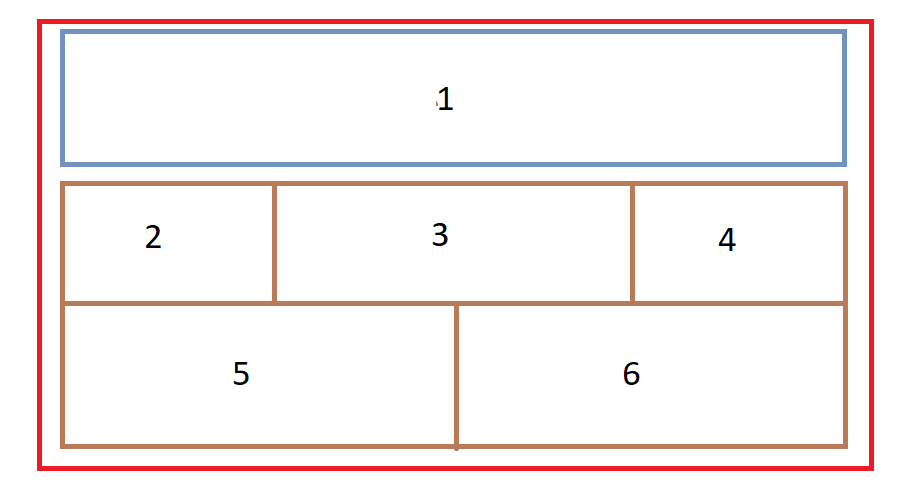
 На рисунке 2 можно увидеть следующие схематические компоненты интерфейса стартовой страницы сайта:

Рисунок – Схема стартовой страницы сайта

1. Рисунок, который отображал бы тематику сайта и самого салона красоты.
2. Блок с последними новостями и акциями сайта.
3. Блок с популярными услугами.
4. Блок с инструкций того как записаться на прием.
5. Блок, в котором перечислены преимущества данного спа-салона.
6. Блок с новыми услугами.

Первый блог должен быстро вводить пользователя в тематику сайта поэтому на нём должна быть изображена одна из процедур салона. во втором блоке можно отображать в текущие акции и новости, которые заинтересуют клиента. Остальные блоки должны располагать клиента на то чтобы он воспользовался услугами салона.

Вторая страница сайта наиболее информативна для пользователя она содержит полный список услуг спа-салона. Отобразить данные об услуге можно таблицы, в которой будет два столбца, а строк в зависимости от количества услуг что позволит компактно разместить данные на странице. Каждая ячейка таблицы должна содержать следующий информацию об услуге, как это изображено на рисунке 3.

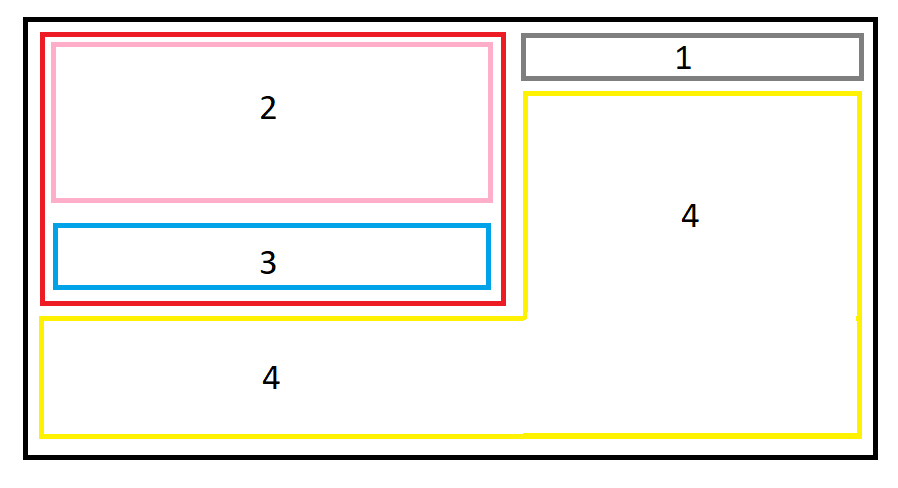


Рисунок - Структура отобржения данных об услуге

На рисунке 3 изображены следующие данные об услугах:

1. Название услуги.
2. Изображение процесса.
3. Цена, длительность.
4. Описание услуги.

Название услуги должно выделяться среди остального текста, а описание должно занимать всё свободное место в ячейке.

Третья страница сайта спа-салона должна содержать список всех сотрудников. Пользователь должен быть проинформирован о том, как выглядит сотрудник, его имя, его специальность, то чем он занимается, его номер ДЛЯ связи и ссылку на его аккаунт в социальной сети. Блок схема информации о сотруднике должна отображаться, изображена на рисунке 4.

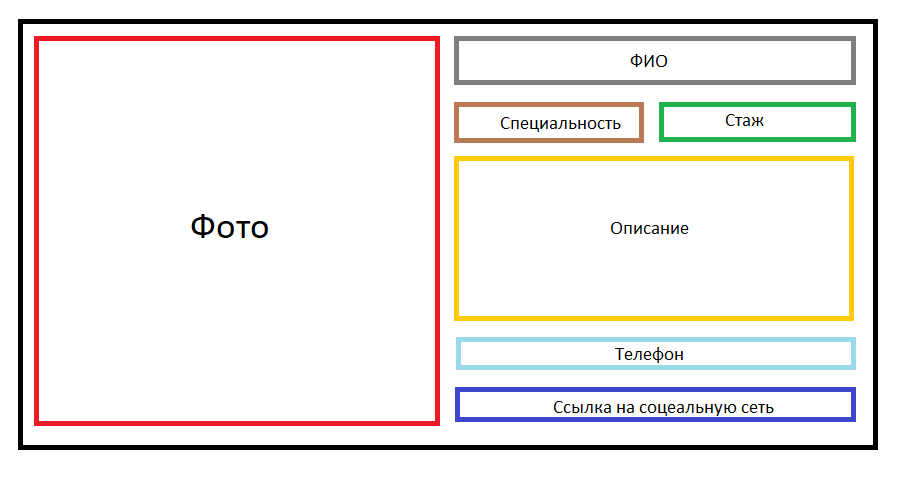


Рисунок – Схема отображения информации о сотрудниках

ФИО, специальность и телефон должны выделяться среди остального текста, а ссылка должна отображаться в виде кнопки. Порядок сортировки специалистов должен зависеть от стажа работы, то есть специалист с наибольшим опытом отображается самый первый, а с наименьшим – последний.

На странице с контактами должна отображаться основная информация для связи и форма отправки сообщения на e-mail фирмы. Также следует расположить интерактивный элемент в виде карты с отображением местонахождения фирмы.

## 2.3 Требования к программным интерфейсам

Для установки и работы программы необходимо иметь вычислительную систему следующей минимальной конфигурации:

* процессор: 1.8 ГГц
* оперативная память: от 512 мб
* видеокарта от 256 мб памяти
* более 50 мб памяти на диске для установки

Необходимо обеспечить программное взаимодействие системы с операционными системами Windows XP/7/8/8.1/10.

## 2.4 Пользователи продукта и требования к ним

Разрабатываемое веб-приложение может понадобится для ведения учета клиентов салона красоты и информирования о нем, поэтому пользователей продукта можно разделить на две категории:

* Администраторы.
* Клиенты.

Опираясь на функционал веб-приложения, можно выдвинуть следующий ряд требований для вышеупомянутых пользователей:

* Подключение к сети Интернет.
* Установленный веб-браузер.

## 2.5 Функции продукта

Функционал сайта можно разбить на два типа: обычный функционал и расширенный. Обычно функционал доступен для всех. В него входит просмотр страниц сайта. Расширенный функционал входит возможность редактирование данных внутри сайта и доступен лишь администратору если тот ведёт логин и пароль. Основной функционал администратора состоит в том, чтобы редактировать добавлять, удалять записи о посещениях клиентами салона, просмотр данных в виде таблицы, которую можно сортировать, а также производить поиск по ней.

## 2.6 Сценарии использования продукта

Начальный сценарий: пользователь открывает веб-сайт в браузере. Пользователь использует веб-браузер для просмотра веб-страниц сайта, ссылки которого могу перенаправлять как на страницы внутри сайта, так и на внешние веб-страницы.

Сценарий авторизации: пользователь нажимает на кнопку для администратора, которая находится в подвале сайта после чего открывает страницу с авторизацией.

Сценарий 1: Пользователь вводит неверно логин и пароль. Выводится сообщение "Пожалуйста, введите корректные имя пользователя и пароль учётной записи. Оба поля могут быть чувствительны к регистру.".

Сценарий 2: Пользователь вводит верный логин и пароль нажимает кнопку войти после чего открывается страница администрирования.

Сценарии использования страницы администрирования:

Сценарий 1: Пользователь нажимает кнопку открыть сайт. Открывается стартовая страница сайта.

Сценарий 1.1: Пользователь нажимает кнопку для администратора, после чего открывается в панель администрирования без авторизации.

Сценарий 2: Пользователь нажимать на кнопку выйти, после чего пользователя перенаправляет на страницу надписью: "Благодарим вас за время, проведенное на этом сайте."

Сценарий 3: Пользователь нажимает на кнопку изменить пароль. Открывается новая страница который три поля: Поле для ввода текущего пароля, поле для ввода нового пароля и поле для подтверждения нового пароля.

Сценарий 4: Пользователь выбирает из списка данные, которые хочет изменить. Открывается страница с этими данными.

Сценарий 4.1: Пользователь выбрал записи о посещениях, открылась страница в который выведены данные о посещениях в виде таблицы.

Сценарий 4.1.1: Пользователь вводит данные в поле поиска и нажимает кнопку поиска. Страница перезагружается, после чего отображается отсортированные данные.

Сценарий 4.1.2: Пользователь нажимает на название одного из столбцов таблицы, после чего, без перезагрузки, данные в таблице сортируются.

Сценарий 4.2: Пользователь выбрал любую запись открылась страница в который выведены данные в виде списка. Страница позволяет производить выделение элементов и их множественное удаление, добавление данных.

## 2.7 Ограничения

* Продукт будет поддерживать только русский язык пользовательского интерфейса.
* Продукт не содержит подсказок к элементам интерфейса.
* Продукт не предусматривает автоматического перехода на платформы, не перечисленные в данном документе.
* Скорость работы приложения будет зависеть от: производительности машины и степени нагрузки ЦП на сервере и у клиента, интернет-соединения.

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

## 3.1 HTML5

HTML5 (англ. HyperText Markup Language, version 5) — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), ещё с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (англ. HTML Living Standard). Цель разработки HTML5-улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие новые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был удалён. Некоторые элементы, например, <a>, <menu> и <cite>, были изменены, переопределены или стандартизированы.

## 3.2 CSS3

CSS3 (англ. Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JS, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

CSS3 преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо. CSS3 основан на CSS2.1, дополняет существующие свойства и значения и добавляет новые нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных.

## 3.3 JavaScript

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

## 3.5 Django framework

Django (Джанго) — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

## 3.6 PyCharm IDE

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Основным критерием выбора данной IDE является то, что она поддерживает веб-разработку на Django. Среди возможностей PyCharm, помогающих в разработке проекта можно выделить следующие:

* Создание каркаса проекта Django
* Запуск тестового сервера Django, нажатием одной кнопки
* Отображение лога сервера в интегрированной в IDE консоли
* Быстрый запуск командной строки
* Подсветка синтаксиса, в том числе и синтаксиса шаблонов Django

## 3.7 SQLite

SQLite – это реляционная база данных, запросы к которой можно осуществлять при помощи языка запросов SQL. База данных не поддерживает все особенности SQL и уступает в функциональности другим развитым СУБД, но вполне подходит для хранения и извлечения информации.

Классические СУБД, такие как MySQL (а также MS SQL, Oracle, PostgreeSQL) состоят из отдельного сервера, поддерживающего работу базы данных и прослушивающих определённый порт, на предмет обращения клиентов. В качестве клиента может выступать в том числе и расширение PHP, реализующего интерфейс, с помощью которого осуществляются запросы к базе. Движок SQLite и интерфейс к ней реализованы в одной библиотеке, что увеличивает скорость выполнения запросов. Такой сервер часто называют встроенным.

Встроенный сервер имеется и у других баз данных, например, у MySQL, но его использование требует лицензионных отчислений, поэтому не получило широкое распространение в мире открытых исходных кодов.

SQLite является бес типовой базой данных. Точнее, есть только два типа – целочисленный "integer" и текстовый "text". Причём "integer" используется преимущественно для первичного ключа таблицы, а для остальных данных пойдёт "text". Длина строки, записываемой в текстовое поле, может быть любой.

Все базы данных хранятся в файлах, по одному файлу на базу. Количество баз данных, а также таблиц в них, ограниченно только свободным местом, имеющимся на сайте. А максимально возможный объём одной базы данных составляет 2 Тб.

Так как все данные хранятся в файлах, проблем с переносом базы данных с одного хостинга на другой не существует – достаточно лишь скопировать соответствующие файлы.

# 4 РАЗРАБОТКА ПО

## 4.1 Создание проекта веб-приложения

Для разработки веб-приложения был выбран фреймворк Django, поэтому для создания веб-приложения необходимо начать с создания проекта Django. Использование PyCharm IDE при разработке ПО значительно сократить выполнение анной задачи. Для этого нужно вызвать окно диалога для создания нового проекта, затем выбрать тип проекта (Django). Далее следует выполнить следующие шаги:

* Выбрать каталог для создания проекта
* Выбрать интерпретатор Python (если на нем не будет установлен Django, то он автоматически установится)
* Выбрать название общей папки для хранения шаблонов
* Выбрать название приложения (если не будет введено, то проект будет создан без приложения)

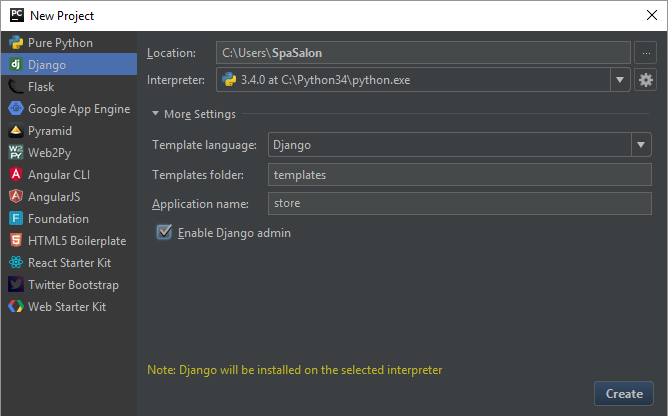
После заполнения необходимых полей для разрабатываемого проекта окно диалога будет выглядеть, как на рисунке 5.

Рисунок - Диалог создание проекта Django

После выполнения вышеуказанных действий будет создан проект Django, содержащий папку с приложением, в которой будут содержатся основные файлы проекта.

Django фреймворк использует свой аналог концепции MVC, под названием MVT (model-template-view). Это значит, что, при отсутствии базы данных в проекте, следует создать еще шаблоны (templates) и представления (views).

Первое, что нужно сделать после создания проекта – это настроить файлы urls.py. Этот модуль содержит код Python, который отображает URL-шаблоны (регулярные выражения) и связанные функции Python (представления). Этот модуль может быть короткий или длинный настолько, насколько это нужно. Он может ссылаться на другие такие модули. И, так как это код Python, может создаваться динамически.

## 4.2 Проектирование БД

4.2.1 Концептуальное проектирование БД

Назначение концептуального представления БД состоит в том, чтобы представить формализованную информацию о предметной области таким образом, чтобы она было достаточно емкой для оценки глубины и корректности проработки проекта базы данных.

Были выделены основные сущности, необходимые для решения задачи:

* Услуга
* Работник
* Новость
* Заявка на посещение
* Пользователь

Основными сущностями БД, как можно заметить, является запись об посещении. Данный обьект должен иметь поля: дату и время посещения, заметки. Данный обьект должен быть связан с услугой, которую прошел клиент, с сотрудником, который обслужил клиента и с клиентом, который решил воспользоваться данной услугой.

Новости имеют минимум полей (заголовок, дата и текст), в отличие от обьект услуг, который должен иметь данные: название, стоимость, продолжительность, описание, а также должен быть со всеми сотрудниками, которые могут эту услугу предоставить.

Обьект сотрудника имеет 6 полей: ФИО, должность, дата устройства на данную работу, описание, телефон и ссылку на какую-то из социальных сетей. Данный обьект может иметь связь сразу с несколькими услугами.

4.2.2 Логическое проектирование БД

В Django большинство взаимодействий с базой данных осуществляется посредством механизма объектно-ориентированного отображения (Object-Relational Mapper или ORM) – функциональности, имеющейся, помимо Django, и в других современных инфраструктурах Web-разработки, таких как Rails. Системы ORM обретают все большую популярность среди разработчиков, так как они автоматизируют множество типичных взаимодействий с базой данных и используют знакомый объектно-ориентированный подход вместо инструкций SQL.

Самая важная часть модели – и единственная обязательная – это список полей таблицы базы данных которые она представляет. Поля определены атрибутами класса. Нельзя использовать имена, конфликтующие с API моделей, такие как clean, save или delete.

Каждое поле в модели должно быть экземпляром соответствующего Field класса. Django использует классы полей для определения такой информации:

* Типа колонки в базе данных (например, INTEGER, VARCHAR).
* Виджет используемый при создании поля формы.
* Минимальные правила проверки данных, используемые в интерфейсе администратора и для автоматического создания формы.

В Django есть большое количество полей; полный список можно посмотреть на странице списка полей. Можно легко добавить собственное поле.

Для каждого поля есть набор предопределенных аргументов. Например, CharField (и унаследованные от него) имеют обязательный аргумент max\_length, который определяет размер поля VARCHAR для хранения данных этого поля. Также есть список стандартных аргументов для всех полей. Все они не обязательны. Все они описаны в разделе про аргументы полей модели, список самых используемых из них описан в таблице 1.

Таблица – Аргументы для создания полей моделей в Django

|  |  |
| --- | --- |
| Имя | Описание |
| null | Если True, Django сохранит пустое значение как NULL в базе данных. По умолчанию - False. |
| blank | Если True, поле не обязательно и может быть пустым. По умолчанию - False.  Это не то же что и null. null относится к базе данных, blank - к проверке данных. Если поле содержит blank=True, форма позволит передать пустое значение. При blank=False - поле обязательно. |
| choices | Итератор (например, список или кортеж) 2-х элементных кортежей, определяющих варианты значений для поля. При определении, виджет формы использует select вместо стандартного текстового поля и ограничит значение поля указанными значениями. |
| default | Значение по умолчанию для этого поля. Это может быть значение или функция. Если это функция - она будет вызвана при каждом создании объекта. |
| help\_text | Подсказка, отображаемая в поле формы. Это полезно для описания поля, даже если поле не используется в формах. |
| primary\_key | При True поле будет первичным ключом.  Если primary\_key=True не указан ни для одного поля, Django самостоятельно добавит поле типа IntegerField для хранения первичного ключа, по-этому вам не обязательно указывать primary\_key=True для каждой модели. Подробнее Первичный ключ по умолчанию. |
| unique | При True поле будет уникальным. |

Основное преимущество реляционных баз данных - возможность добавлять связи для таблиц. Django предоставляет возможность использовать три самых используемых типа связей: многое-к-одному, многое-ко-многому и один-к-одному.

Для определения связи используется django.db.models.ForeignKey,  [ManyToManyField](https://djbook.ru/rel1.7/ref/models/fields.html#django.db.models.ManyToManyField) или OneToOneField. Можно использовать их так же, как и другие типы Field: добавляя, как атрибут в модель. Для данных типов необходимо указать обязательный позиционный аргумент: класс связанной модели.

Помимо основных атрибутов обьектов описанных ранее, следует расширить объекты приложения для улучшения функционала. Например, для реализации алгоритма записи в салон, следует добавить к объекту записи два булевых поля – одно для клиента, а другое для сотрудника, в дальнейшем эти поля будут использоваться для проверки состояния записи.

С учётом вышеупомянутого были созданы классы, часть которых можно рассмотреть на рисунке 6.

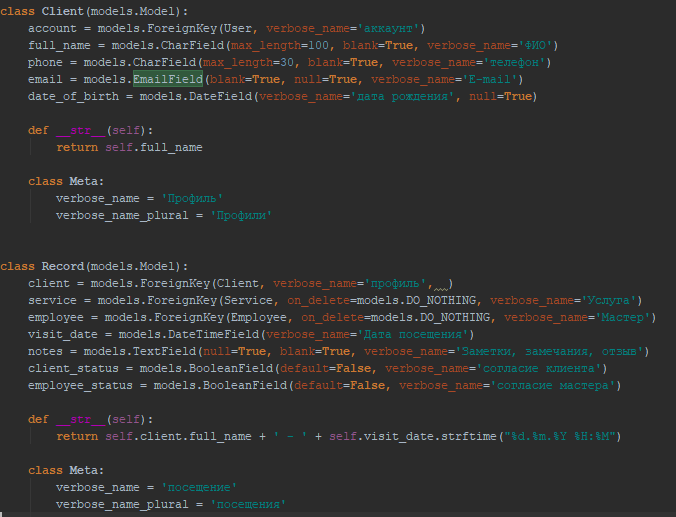


Рисунок – Классы, представляющие таблицы в БД

На рисунке 6, помимо атрибутов классов, можно еще увидеть созданный класс Meta и переопределенный метод \_\_str\_\_. Meta класс содержит настройки обьектов, т.е. он работает не только для моделей, для форм и не только.

Основные параметры класса Meta, используемые в данном проекте описаны в таблице 2:

Таблица 2 – Основные параметры класса Meta

|  |  |
| --- | --- |
| Имя | Описание |
| abstract | При abstract = True, эта модель будет абстрактной моделью. |
| app\_label | Если модель находится не в models.py (например, модели находятся в модулях пакета myapp.models), модель должна определять к какому приложению принадлежит модель. |
| db\_table | Название таблицы в базе данных для этой модели. |
| ordering | Сортировка по-умолчанию используемая при получении объектов. Это кортеж или список строк. Каждая строка — это название поля с необязательным префиксом “-”, который указывает на нисходящую сортировку. Поля без “-” будут отсортированы по возрастанию. Используется”?” для случайной сортировки. |
| permissions | Дополнительные разрешения (permissions) будут добавлены в таблицу разрешений при создании модели. Разрешения на добавление, удаление и изменение автоматически создаются для каждой модели. |
| verbose\_name,  verbose\_name\_plural | Читабельное название модели, в единственном и множественном числе соответственно. |

4.2.3 Физическое проектирование БД

Физическое проектирование базы данных подразумевает процесс подготовки переноса реализации базы данных на определенную СУБД. На изображении ниже представлена физическая схема базы данных для ВКР, которая получилась в результате работы Django-OR. Для построения схемы (рисунок 7) было выбрано веб-приложение DbDesigner.

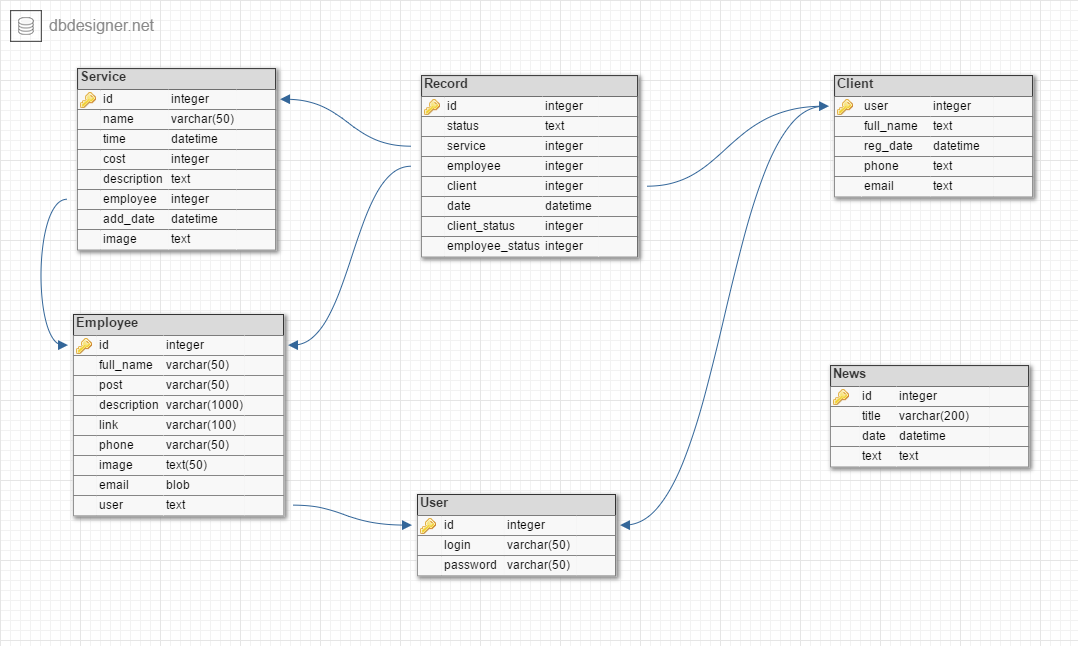


Рисунок – Логическая схема базы данных

## 4.3 Настройка URL-менеджера

Одно из первых, что нужно сделать после создания проекта – это настроить файлы urls.py. Этот модуль содержит код Python, который отображает URL-шаблоны (регулярные выражения) и связанные функции Python (ваши представления). Этот модуль может быть короткий или длинный настолько, насколько это нужно. Он может ссылаться на другие такие модули. И, так как это код Python, может создаваться динамически.

В нашем случае код менеджеров URL будут достаточно просты. Основной модуль, который находится в корне проекта предназначен, в основном, для переадресации URL-ов на другие модули приложений для дальнейшей обработки. Это сделано для того, чтобы не загромождать данный файл и не выносить логику обработки URL-ов приложений в файл корня проекта. Внутренний менеджер URL-адресов приложения связывает URL и контроллер, который его обрабатывает.

Для создания полной функциональности сайту следует разбить все URL-адреса на три типа: административной панели, где будет храниться функционал; на адреса визуальной части сайта, которая будет красиво отображать данные сайта для потенциальных клиентов и дополнительные адреса. На рисунке 8 изображено содержание менеджеров URL-адресов.

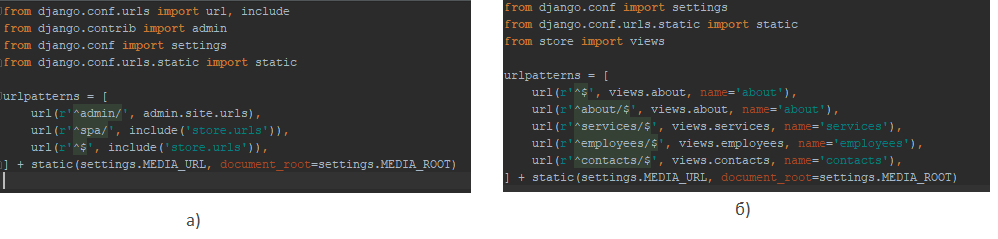


Рисунок – Содержимое URL-менеджеров:

a – менеджер проекта; б – менеджер приложения

## 4.4 Создание шаблонов

Будучи веб фреймверком, Django позволяет динамически генерировать HTML. Самый распространенный подход - использование шаблонов. Шаблоны содержат статический HTML и динамические данные, рендеринг которых описан специальным синтаксисом.

Django предоставляет стандартный API для загрузки и рендеринга шаблонов, незавимисо от используемого бэкенда. Загрузка включает в себя поиск шаблона по названию и предварительную обработку, обычно выполняется загрузка шаблона в память. Рендеринг означает передачу данных контекста в шаблон и возвращение строки с результатом.

Шаблон Django – это просто текстовый файл, или строка Python, которые следуют языку шаблонов Django. Определенные конструкции распознаются и интерпретируются шаблонизатором. Основные – это переменные и теги.

Шаблон рендерится с контекстом. Рендеринг заменяет переменные на их значения, которые ищутся в контексте, и выполняет теги. Все остальное выводится как есть. Синтаксис языка шаблонов Django использует четыре конструкции: переменные, теги, фильтры, комментарии.

Переменные выводят значения из контекста, который является словарем. Переменные выделяются в двойные фигурные скобки. Обращение к ключам словаря, атрибутам объектов и элементам списка выполняется через точку, а если значение переменной является вызываемый объект, шаблонизатор вызовет его без аргументов и подставит результат.

Теги позволяют добавлять произвольную логику в шаблон. Например, теги могут выводить текст, добавлять логические операторы, такие как “if” или “for”, получать содержимое из базы данных, или предоставлять доступ к другим тегам. Теги выделяются {% имя\_тега %}. Большинство тегов принимают аргументы, а некоторые требуют закрывающий тег (например, if или for).

Фильтры преобразуют переменные и аргументы тегов, а комментарии позволяют нужны для комментариев и могут быть однострочными (например, {# это будет комментарий #}) и многострочными (тег {% comment %}).

Также, шаблоны поддерживают вложенность. Данный механизм позволяет не повторять одни и те же куски кода в шаблонах, а использовать их как переменные. Следуя ТЗ в ВКР было создано 4 шаблона для отображения контента и один базовый шаблон, который содержит шапку и подвал сайта. Данный механизм работает за счет двух шаблонных тегов: extends и block.

Тег extends указывает что данный шаблон наследуется от родительского, а block определяет блок разметки, который может быть переопределен в дочернем шаблоне. Extends использоваться двумя способами:

* {% extends "base.html" %} (с кавычками) использует буквальное значение "base.html" в качестве названия родительского шаблона.
* {% extends variable %} использует значение variable. Если значение строка, Django использует ее как название родительского шаблона. Если значение переменной объект Template, Django использует этот объект как родительский шаблон.

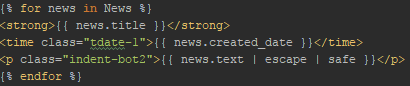
 Для отображения списка обьектов из базы был использован тег for, который перебирает все элементы в массиве, что пришел в шаблон с контроллера. Для отображения новостей выглядит это следующим образом, как на рисунке 9.

Рисунок – Пример использования тега for

Также, внутри шаблонов были использованы и другие теги, такие как:

* Load – загрузка библиотеки тегов
* Static - создает ссылку на файл в директории со статическими файлами
* Сycle – подстановка переменных при повторном вызове.

Первым, используется тег load с параметром static, что позволит использовать последний в шаблоне. Шаблонный тег static используется для указания пути к статическим файлам в тегах link и script. Таким образом, подключение файлов в шапке сайта будет выглядеть следящим образом (рисунок 10).

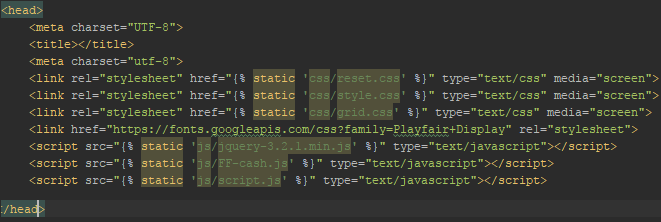


Рисунок – Пример подключения статических файлов

Также, кроме интерфейса сайта, который будет выступать для отображения данных потенциальным клиентам, следует позаботиться и об интерфейсе, который бы мог выступать в качестве интерфейса пользователей сайта. Для этих целей в Django существует готовое приложение admin (автоматический интерфейс администратора) и это одна из сильных сторон. Он использует мета-данные модели чтобы предоставить многофункциональный, готовый к использованию интерфейс для работы с содержимым сайта. Установка интерфейса администратора состоит из семи шагов:

* Добавить 'django.contrib.admin' в настройку INSTALLED\_APPS.
* Интерфейс администратора имеет зависимости - django.contrib.auth, django.contrib.contenttypes, django.contrib.messages and django.contrib.sessions. Если эти приложения не входят в список INSTALLED\_APPS, то нужно добавить их.
* Добавить django.contrib.messages.context\_processors.messages в TEMPLATE\_CONTEXT\_PROCESSORS и MessageMiddleware в MIDDLEWARE\_CLASSES. (Эти параметры включены по умолчанию, необходимо добавить их только, если изменялись настройки.)
* Определить какие модели будут редактироваться через интерфейс администратора.
* Для каждой модели можно создать класс ModelAdmin, который инкапсулирует настройки интерфейса администратора для конкретной модели.
* Создть экземпляр AdminSite и добавить в него модели с соответствующими классами ModelAdmin.
* Добавьте AdminSite в URLconf.

Будучи частью фреймворка, приложение admin имеет свои шаблоны, представления и так далее, однако данное приложение вместе с его файлами находятся в папке самого фреймворка. Для того, чтобы редактировать эти данные можно скопировать нужные файлы приложения в соответствующие папки (статические файлы внутрь любого каталога static, а шаблоны в любую папку, указанную в TEMPLATE\_DIRS). Поскольку приоритет файлов, находящихся в проекте выше, чем у файлов сторонних библиотек, то, после выполнения данных действий, наши файлы будут подхватываться с проекта, а нужные сторонни файлы – с приложения admin, которое находится внутри Django.

Для изменения были выбраны три файла с приложения:

* Base.html – файл с шапкой панели администрирования. В этом файле изменены надписи и ссылки на основную страницу сайта. Был добавлен параметр для отображения текущей страницы.
* Base\_site.html – основной шаблон страницы, в котором произвелись незначительные изменения (в основном в шапке страницы).
* Login.html – страница авторизации, изменено позиционирование блоков.

Из файла login был создан файл registration. Стиль элементов данной страницы схож со стилем страницы авторизации, но данный шаблон использует другую форму для отображения.

## 4.5 Создание представлений

Функция представления, или коротко представление – это функция Python, которая принимает Web-запрос и возвращает Web-ответ. Ответом может быть HTML-содержимое страницы, или перенаправление, или 404 ошибка, или XML-документ, или изображение, или что угодно. Представление содержит всю необходимую логику для создания ответа. Этот код может находиться где угодно, главное, чтобы он находился в PYTHON\_PATH. Никаких других требований нет – никакой “магии”. Несмотря на возможность расположить код представлений где угодно, принято держать его в файле views.py, который находиться в каталоге проекта или приложения. Для данной задачи были составлены 4 представления, изображенные на рисунке 11.



Рисунок – Файл с представлениями проекта

Представление contacts просто компилирует шаблон и возвращает его, а остальные представления используют QuerySet для получения, сортировки и фильтрации данных.

QuerySet в Django является представлением некоторого числа строк в базе данных, опционально отфильтрованных посредством запроса. Например, следующий код является представлением всех людей в базе данных по имени ‘Dave’: «person\_set = Person. objects. Filter (first\_name = "Dave») ». Приведенный выше код не запускает какие-либо запросы к базе данных. Можно взять person\_set и применить дополнительные фильтры, или передать его в функцию, и ничего не будет отправлено в базу данных. Это хорошо, потому что запросы к базе данных являются одной из вещей, которые существенно замедляют веб-приложения. Как только начинается осуществляться перебор по QuerySet, все строки, соответствующие QuerySet извлекутся из базы данных и будут преобразованы в объекты моделей Django. Это называется вычисление (evaluation). Затем эти модели сохранятся во встроенный кэш QuerySet, так что, если осуществите перебор по QuerySet снова, то не выполните тот же запрос повторно.

У данного обьекта есть метод, которые был использован в ВКР, а именно - order\_by. Данный метод управляет сортировкой обьекто. Например код «Entry.objects.filter(pub\_date\_\_year=2005).order\_by('-pub\_date', 'headline')» в результате отсортирует данные в обратном порядке по полю pub\_date, далее по полю headline. Знак “минус” в "-pub\_date" указывает на “нисходящую” сортировку. Сортировка по возрастанию подразумевается по-умолчанию. Что бы отсортировать случайно используйте "?", однако следует помнить, что данный метод может работать достаточно медленно. Для сортировки по полю из другой модели, используйте синтаксис аналогичный тому, который используется при фильтрации по полям связанной модели. То есть, название поля, далее два нижних подчеркивания (\_\_), и имя поля в новой модели, и так далее. Например «Entry.objects.order\_by('blog\_\_name', 'headline')».

## 4.6 Интерфейс администратора Django

Одна из сильных сторон Django – это автоматический интерфейс администратора. Он использует мета-данные модели чтобы предоставить многофункциональный, готовый к использованию интерфейс для работы с содержимым сайта, которым обычно являются модели. По умолчанию Django имеет свой набор прав доступа. Всеми правами обладает суперпользователь, но их можно создавать, добавлять, привязывать к группам и отдельным пользователям, что делает процесс администрирования учетными записями очень гибким.

Однако, базовых возможностей Django не хватает, когда дело доходит до настоящего, среднего или крупного проекта. В таком случае следует использовать класс ModelAdmin – это отображение модели в интерфейсе администратора. Его код добавляют в файл admin.py приложения. Рассмотрим простой пример ModelAdmin на рисунке 12. В этом примере, класс ModelAdmin не определяет никаких настроек (пока). В результате мы получим интерфейс, предоставляемый по умолчанию.

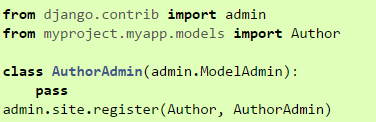


Рисунок - Простой пример ModelAdmin

ModelAdmin очень гибкий. Он содержит ряд параметров для настройки интерфейса администратора. Все настройки определяются в подклассе ModelAdmin как значения специальных атрибутов и функций. Настройки ModelAdmin, использованые в ВКР представлены в таблице 3:

Таблица 3 – Некоторые из настроект ModelAdmin

|  |  |
| --- | --- |
| Имя | Описание |
| fields | Если необходимо внести небольшие изменения форму на странице редактирования и добавления, например, изменить список отображаемых полей, их порядок или сгруппировать их, можно использовать настройку fields (сложные изменения можно выполнить используя настройку fieldsets описанную ниже). |
| form | По умолчанию ModelForm создается динамически для модели. Этот атрибут используется для определения формы на страницах добавления и редактирования. Можно указать собственный подкласс ModelForm для переопределения этих страниц. |
| list\_display | list\_display указывает какие поля отображать на странице списка объектов. Можно указать четыре варианта значений в list\_display:   * Поле модели. * Функция, которая принимает один аргумент - объект модели. * Название атрибута ModelAdmin. Работает так же как и функция. * Название атрибута модели. Работает так же как и функция, но self в этом контексте объект модели. |
| list\_filter¶ | Укажите list\_filter что бы определить фильтры данных в правой панели страницы списка объектов. list\_filter - это список элементов, которые могу быть:   * Название поля следующего типа: BooleanField, CharField, DateField, DateTimeField, IntegerField, ForeignKey или ManyToManyField. * Подкласс django.contrib.admin.SimpleListFilter, в котором необходимо определить атрибуты title и parameter\_name и переопределить методы lookups и queryset. * Кортеж, где первый элемент название поля, а второй - подкласс django. contrib. admin. FieldListFilter * Можно указать собствен SimpleListFilter ный шаблон для фильтра, унаследовав |
| ordering | Оrdering позволяет определить сортировку на странице списка объектов. Это должен быть список или кортеж в формате аналогичном параметру ordering. Если атрибут не указана, Django будет использовать сортировку по умолчанию модели. |

Каждый из данных свойств может быть получен из функции с аналогичным названием. Для организации атрибутов в виде функций требуется больше кода, но данный подход неизбежен, если необходимо работать с динамическими данными.

# 5 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 5.1 Структура проекта

Файловая структура является структурой проекта, а также структурой модулей Python, что является важной составляющей Django-проекта. Файловую структуру проекта можно рассмотреть на рисунке 13.

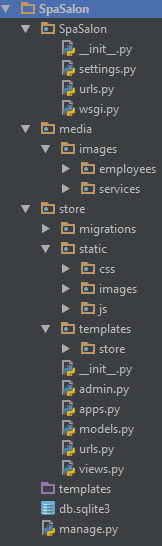


Рисунок - Файловая структура проекта

Данная структура проекта (рисунок 13) является стандартной структурой Django-проектов и достигается с помощью выполнения команд startproject и startapp, однако данные команды не пришлось выполнять вручную, так как среда разработки PyCharm выполнила их автоматически. На первом шаге, с помощью пакета Django была создана следующая структура:

* Корневая директория SpaSalon/: – это корневая директория проекта. Его название никак не используется Django, можно переименовать его как угодно
* manage.py: Скрипт, который позволяет взаимодействовать с проектом Django
* Внутренний каталог SpaSalon/ - это пакет Python проекта. Его название – это название пакета Python, которое используется для импорта чего-либо из проекта (например, Site.urls)
* SpaSalon /\_\_init\_\_.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
* SpaSalon /settings.py: Настройки/конфигурация проекта
* Site/urls.py: Конфигурация URL-ов для проекта Django. Это “содержание” всех Django-сайтов
* SpaSalon /wsgi.py: Точка входа проекта для WSGI-совместимых веб-серверов

На следующем шаге, командой startapp, был создан каталог внутри корневой папки с файлами проекта со следующим содержимым:

* \_\_init\_\_.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
* migrations/: Директория, в которой сохраняются миграции
* static/: Папка со статическими файлами приложения (скриптами, файлами стилей и т.д.)
* templates/: Папка с шаблонами
* apps.py: Модуль, который добавляет в проект текущее приложение

Помимо автоматически созданных файлов, в директорию с приложением была добавлена директория static для статических фалов стилей и Java Script файлов, которые разделены по разным поддиректориям.

Также, в папке с приложением была создана папка templates для шаблонов. Данная директория должна иметь специальную структуру, в которой шаблоны, относящиеся к приложению должны содержаться в папке, которая имеет такое же название, как и приложение.

## 5.2 Описание объектов и их взаимодействия

Архитектура приложения зачастую зависит от архитектуры фреймворка, с помощью которого он был создан. Django-приложение состоит из четырех основных компонентов: модель данных, представление, шаблоны, URL.

Любая модель является стандартным python-классом. Объектно-ориентированный ORM обеспечивает таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL, однако в данном проекте не было необходимости создавать свои модели.

Представление - объекты типа view в django выполняют разнообразные функции, в том числе контролируют запросы пользователя, выдают контекст в зависимости от его роли. View – это обычная функция, которая вызывается в ответ на запрос какого-то адреса (url) и возвращает контекст. В данном проекте существует только одно представление index.

Шаблоны являются формой представления данных. Шаблоны имеют свой собственный простой метаязык и являются одним из основных средств вывода на экран.

URL - это всего лишь механизм внешнего доступа к представлениям (view). Встроенные в url-файлы регулярные выражения делают механизм достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким url-файлами, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: url-файлы становятся как бы самодостаточными и начинают жить независимо от представления.

Общая схема взаимодействия компонентов любого Django-проекта показана на рисунке 14.



Рисунок – Архитектура Django-приложения

# 6. Тестирование и внедрение

## 6.1 Тестирование разработанного ПО

Назначение, область применения и требования к надежности программного продукта показаны в таблице 4, а результаты анализа надежности в таблице 5.

Таблица 4 – Назначение, область применения и требования к надежности

|  |  |  |
| --- | --- | --- |
| Назначение | Область применения | Требования к надежности |
| Учет клиентов салона красоты | Маркетинг | Скорость обработки, функциональность |

Таблица 5 - Анализ надежности программного продукта

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № Теста | | Данные для теста  (вводимая информация) | | Время функционирования программы | | Результат (успешно\отказ) | | Результат | |
|  | | Открытие сайта | | 1.3 секунды | | успешно | | Отображение стартовой веб-страницы сайта | |
|  | | Добавление записи | | Менее секунды | | успешно | | Открытие страницы со всеми записями | |
|  | | Добавление записи с некорректными параметрами | | Мгновенно | | Успешно | | Обведение полей с неверными данными | |
|  | | Сортировка по полю в таблице записей | | 0.8 секунд | | успешно | | Перезагрузка страницы с последующей сортировкой полей | |
|  | | Поиск по полю в таблице записей | | 0.8 секунд | | успешно | | Перезагрузка страницы с отображением только тех полей, которые соответствуют поиску. | |

Данные о скорости загрузки веб-страниц получены с помощью инструментов Google Chrome для тестирования веб-страниц. Пример откладки веб-страницы с помощью Google Chrome изображен на рисунке 15.

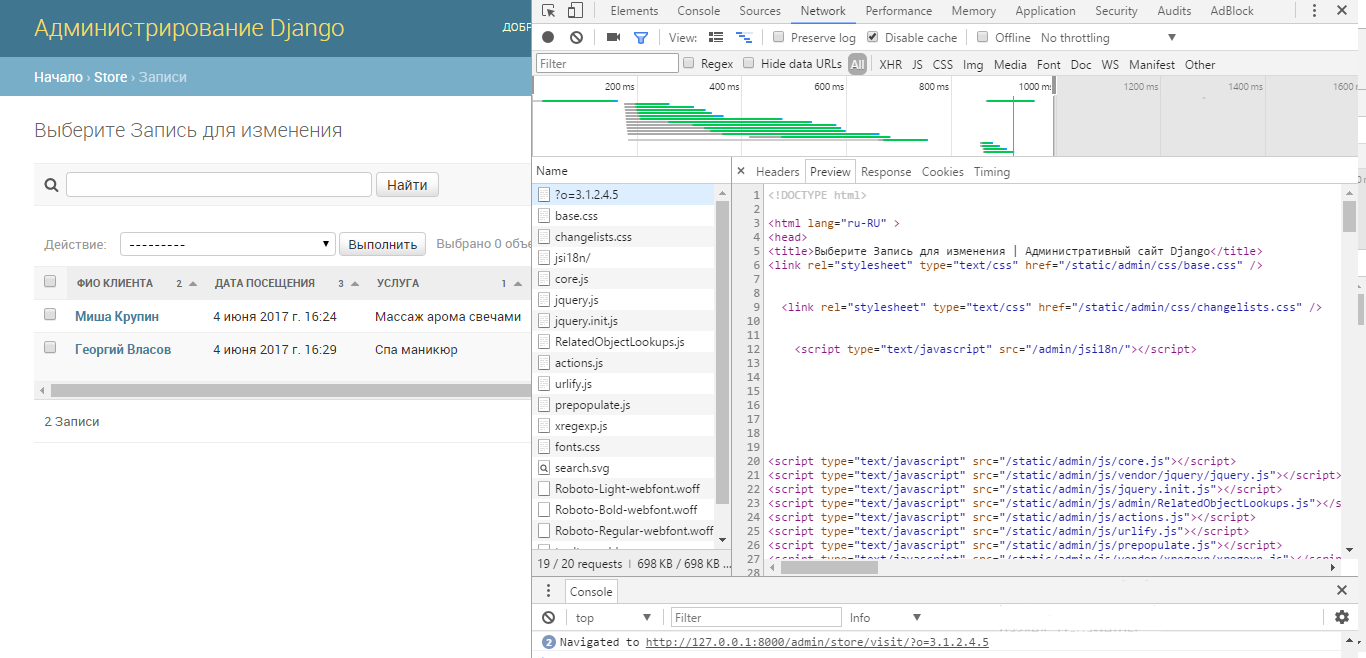


Рисунок – Использование инструментов Google Chrome

для тестирования веб-страницы

## 6.2 Установка программы

6.2.1 Необходимые аппаратные и программные средства

Для установки ПО действуют следующие системные требования:

* + Не менее 500 mb памяти на жестком диске
  + Процессор – Intel Pentium CPU 2117U@ 1.80GHz (2 ядра)
  + Видеокарта от 256 mb памяти
  + Оперативная память от 1 GB и выше
  + Доступ к сети Internet

Поскольку, Python мультиплатформенный, то данное веб-приложение может работать практически на любой системе, на которой будет работать Python и будет доступ к интернету.

Для функционирования ПО действуют следующие минимальное программные требования:

* + Django v 1.10
  + Python v 3.4
  + Pillow v 3.4.2
  + Django-tables2 v 1.7

Для полноценной работы веб-приложения необходимо произвести развертывание проекта на полноценный веб-сервер (Apache, например).

6.2.2 Описание процесса развертывания

Основной платформой для развертывания Django является WSGI, это фактически стандарт для веб-серверов и приложений на Python. Команда startproject установит простую WSGI конфигурацию по умолчанию, которую можно впоследствии изменить под нужды проекта и использовать с любым WSGI-совместимым веб-сервером.

Одна из ключевых концепций развертывания с WSGI заключается в указании функции, или вызываемого объекта, application, который использует веб-сервер для взаимодействия с кодом. Обычно это объект application модуля Python доступного для сервера.

Команда startproject создаст файл <project\_name>/wsgi.py, который содержит вызываемый объект application. Этот файл использует как встроенным сервером для разработки, так и на боевом WSGI сервере.

WSGI сервер получает путь к объекту application из своих настроек. Встроенный сервер Django использует настройку WSGI\_APPLICATION. По умолчанию она равна <project\_name>.wsgi.application, и указывает на объект application в <project\_name>/wsgi.py.

Когда WSGI сервер загружает ваше приложение, Django необходимо импортировать модуль с настройками. Django использует переменную окружения DJANGO\_SETTINGS\_MODULE для определения расположения модуля настроек. Она должна содержать путь для импорта этого модуля. Если переменная не определена, wsgi.py использует значение mysite.settings, где mysite название вашего проекта.

В данной курсовой работе будет рассмотрен самый популярный вариант развёртывания - Apache и mod\_wsgi. Mod\_wsgi является модулем веб-сервера Apache, который может взаимодействовать с любым приложением Python, в том числе Django. Django работает с любой версией Apache, поддерживающей mod\_wsgi.

После установки и активации mod\_wsgi нужно отредактировать файл httpd.conf веб-сервера Apache, изменив его определенным образом. Значение WSGIScriptAlias указывает местоположение приложений, (/ обозначает корневую директорию), вторым значением указывается расположение файла “WSGI (как правило, в корне проекта). Эти настройки позволят Apache обрабатывать любой запрос из директории, указанной как базовая с помощью WSGI-приложения, хранящегося в ней. WSGIPythonPath гарантирует, что проект доступен для импорта; иначе говоря, что команда import Site сработает. Значение <Directory> просто предоставляет Apache доступ к файлу wsgi.py.

# ЗАКЛЮЧЕНИЕ

В результате работы над ВКР выполнены все поставленные задачи. Разработан работоспособное веб-приложение, которое позволяет отображать нужные данные, а также производить учет клиентов салона красоты. Проанализированы существующие аналоги, выделены базовые критерии оценки. Был изучен ряд технологий для создания проекта: HTML, CSS, Java Script, Pyhton, Django, PyCharm, SQLite, JSON. Также, было написано обоснование выбора данных технологий в курсовом проекте. Функционал веб-приложения был утвержден согласно разработанной структуре программы. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной и серверной частью.

На данный момент при работе с веб-приложением доступны следующие просмотр информации о салоне красоты SpaSalon, просматривать и редактировать данные слона при наличии данных для авторизации, производить полный учет посещений салона

Дальнейшее развитие программы может быть связано с добавлением обьектов и их связей в БД, улучшением информированием пользователя, повышением безопасности данных сайта. Тестирование программы показало ее работоспособность.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Описание «1С-Рарус: Комильфо: Салон красоты» // [rarus]

URL: https://rarus.ru/1c-beauty/1c-predpriyatie-8-salon-krasoty/

(дата обращения 02.03.17).

2. Список цен на комплексы автоматизации 1С – Битрикс // [rarus]

URL: https://rarus.ru/1c-beauty/ (дата обращения 2.03.2017)

3. Dbdesigner.net – Онлайн калькулятор таблиц // [Beauty]

URL: <https://dbdesigner.net/designer/>

(дата обращения 03.03.17).

4. Wikipedia – информационный ресурс:

URL: https://ru.wikipedia.org/wiki/HTML5

(дата обращения: 13.02.2017)

URL: https://ru.wikipedia.org/wiki/JavaScript

(дата обращения: 14.02.2017)

URL: https://en.wikipedia.org/wiki/Brackets\_(text\_editor)

(дата обращения: 14.02.2017)

5. JavaScript. Подробное руководство / Дэвид Флэнаган / Год: 2013

6. w3school: информационный ресурс

URL: https://www.w3schools.com/html/default.asp  
(дата обращения: 22.02.2017)

URL: https://www.w3schools.com/css/default.asp  
(дата обращения: 24.02.2017)

URL: https://www.w3schools.com/js/default.asp  
(дата обращения: 26. 02.2017)

# ПРИЛОЖЕНИЕ А

**Экранные формы**

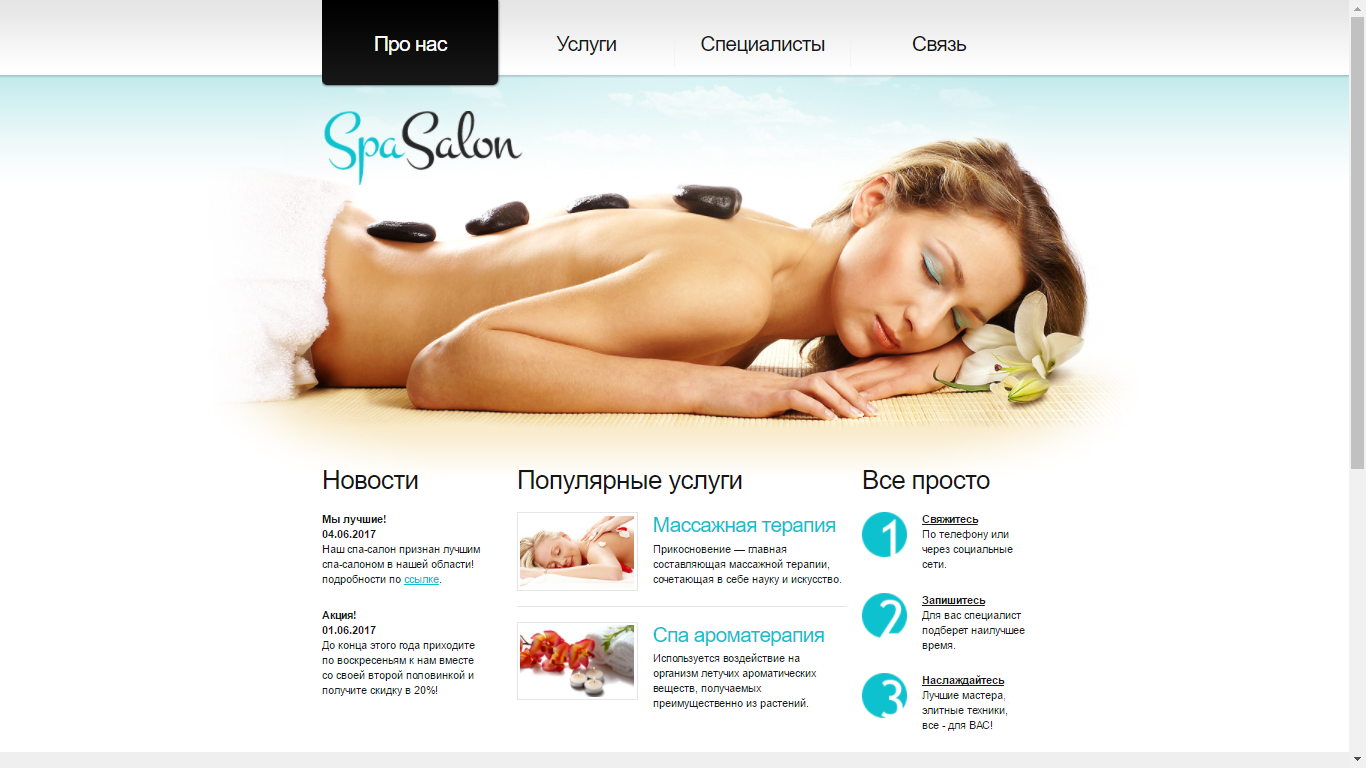


Рисунок А.1 - Вид шапки сайта, логотипа и первой части

стартовой страницы сайта

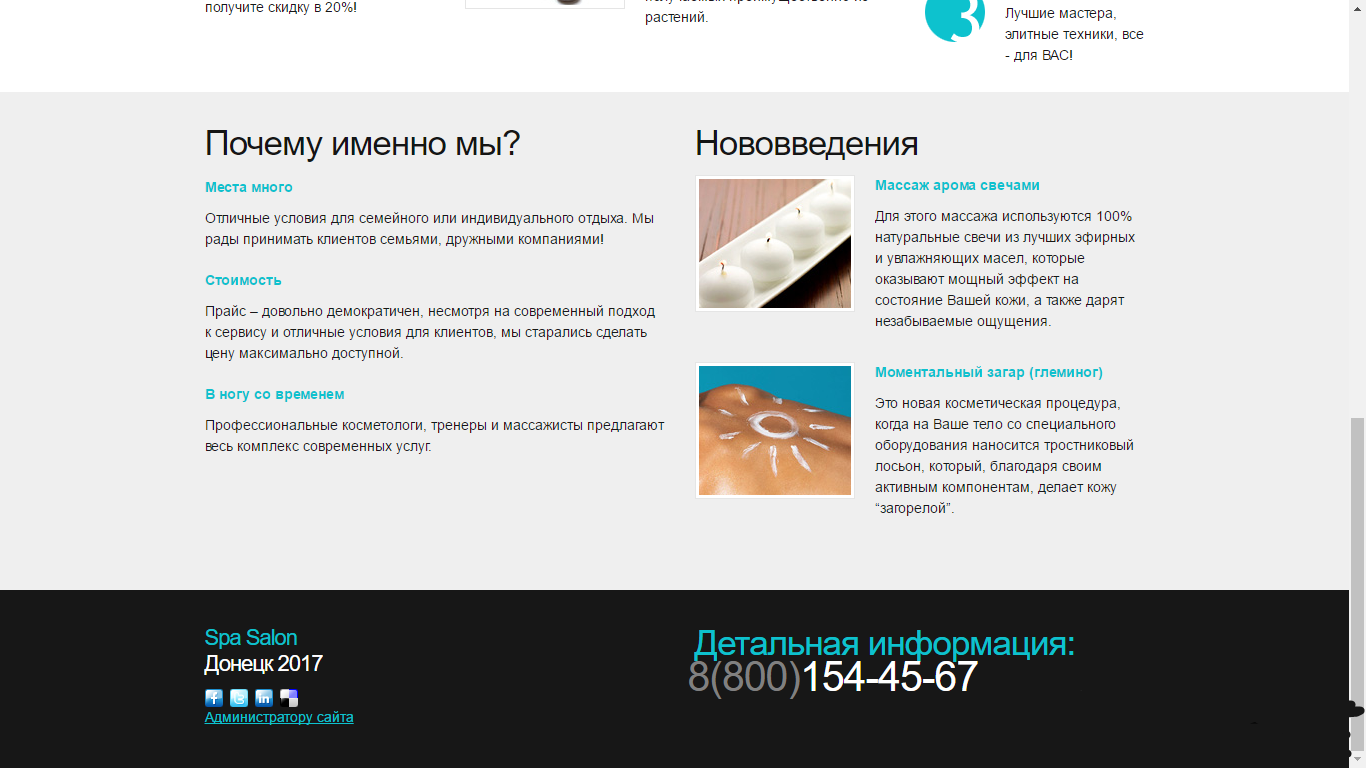


Рисунок А.2 – Вид второй части стартовой страницы сайта и подвала сайта



Рисунок А.3 – Вид контента страницы с услугами

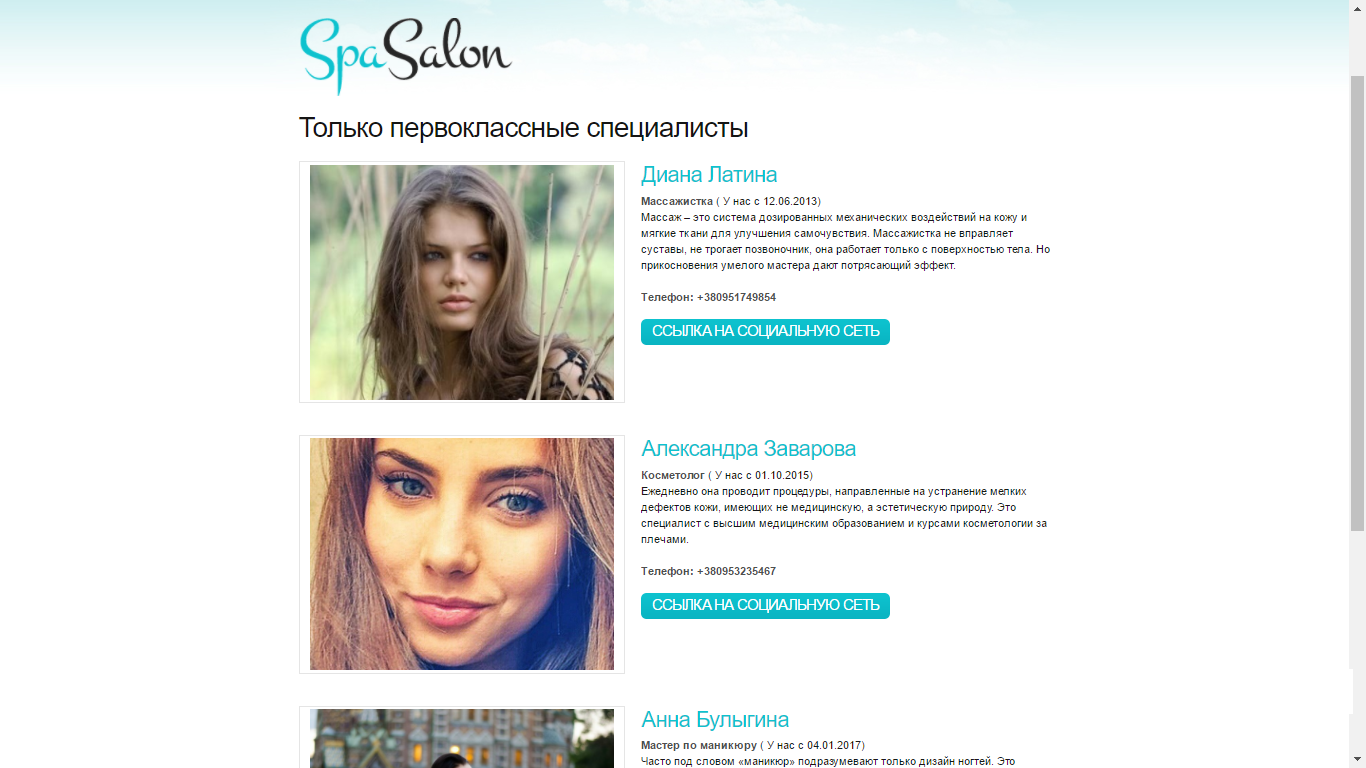


Рисунок А.4 – Вид контента страницы с сотрудниками

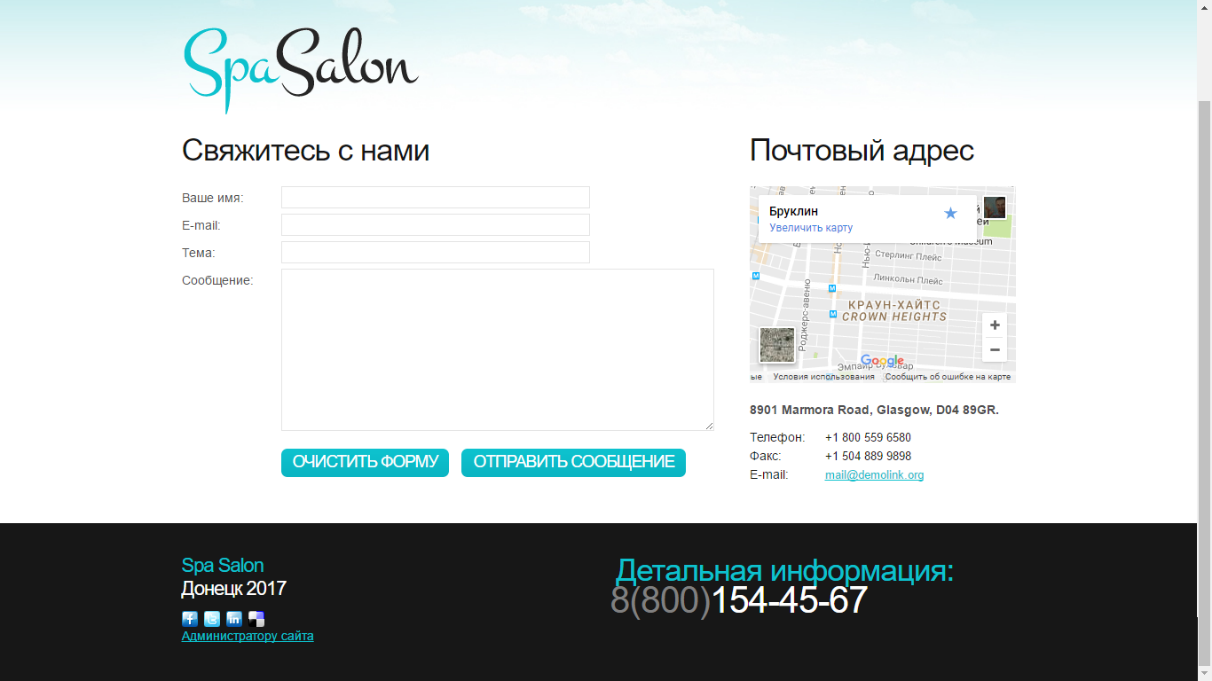


Рисунок А.5 - Вид контента и подвала страницы для связи

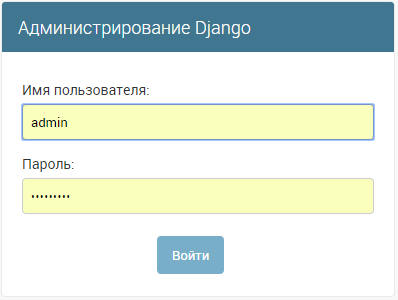


Рисунок А.6 – Вид страницы авторизации

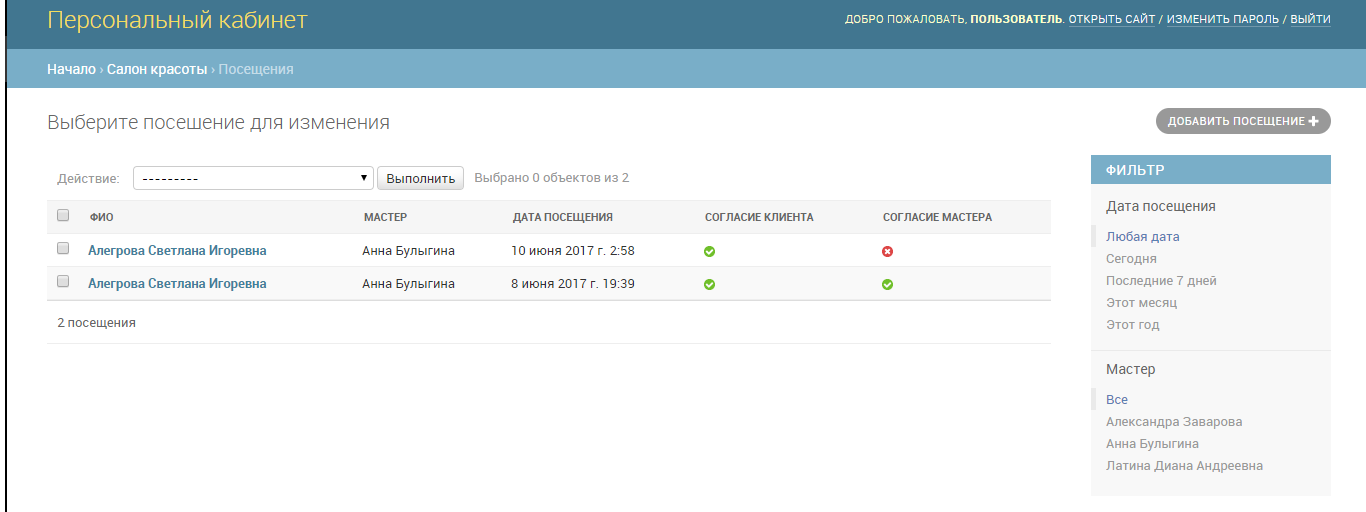


Рисунок А.7 – Вид панели управления записями о посещении

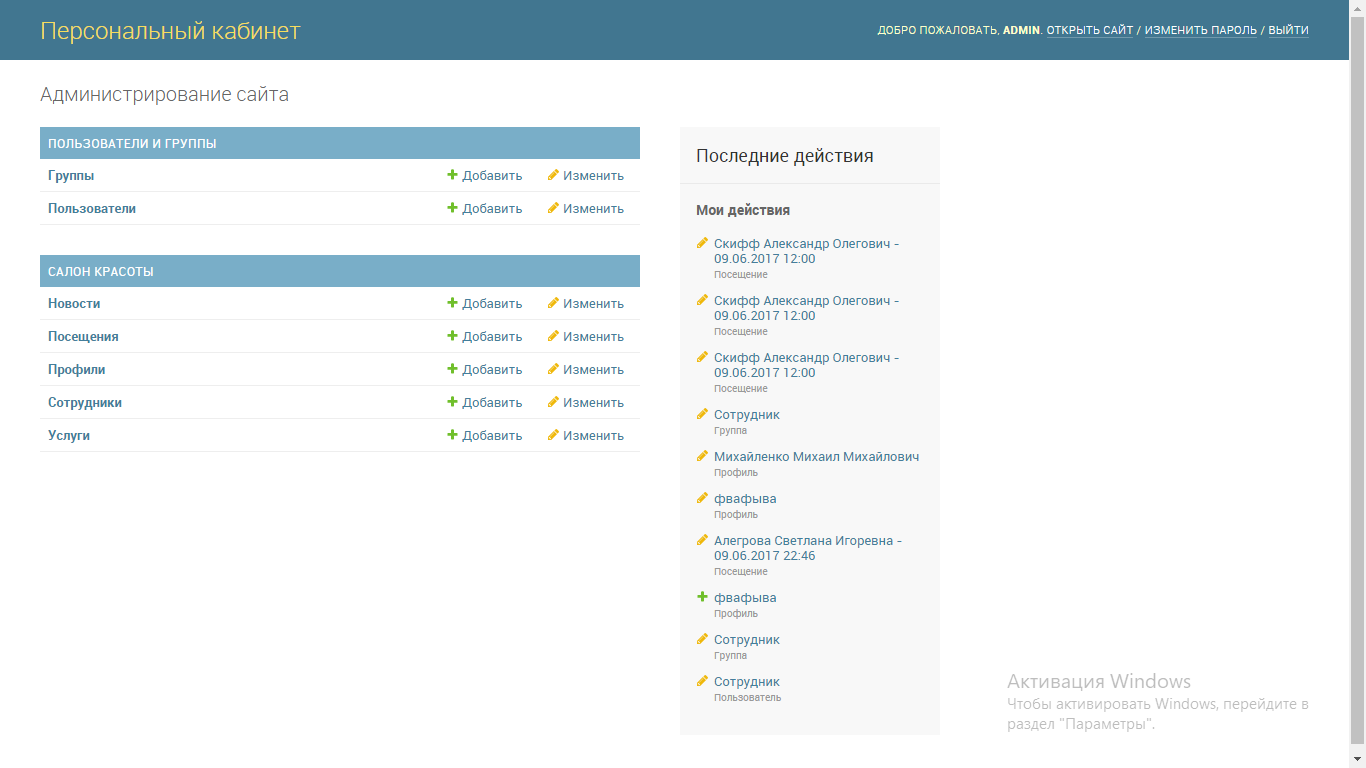


Рисунок А.8 – Главная страница панели администрирования

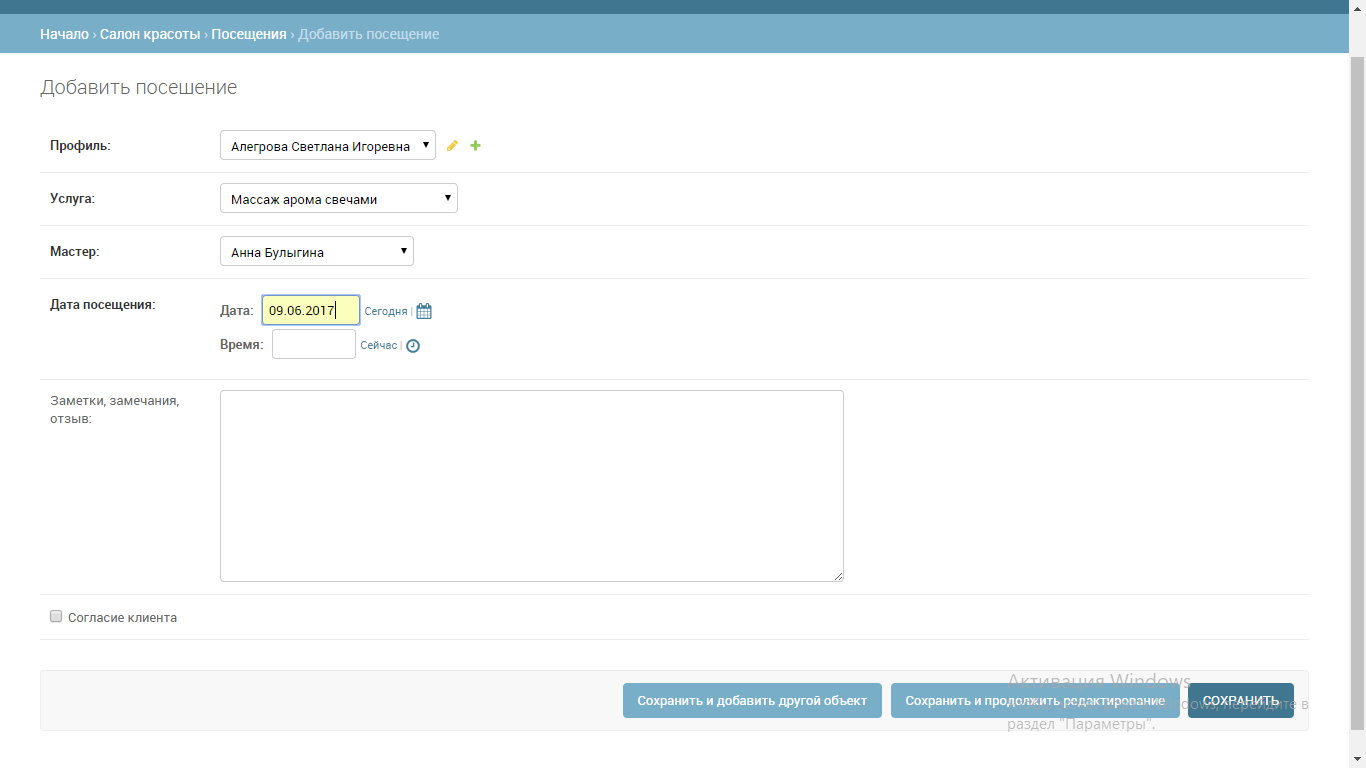


Рисунок А.9 – Вид страницы редактирования

новой или существующей записи

# ПРИЛОЖЕНИЕ Б

**Фрагменты листинга**

Листинг Б.1 — Базовый шаблон сайта.

{% load staticfiles %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title></title>

<meta charset="utf-8">

<link rel="stylesheet" href="{% static 'css/reset.css' %}" type="text/css" media="screen">

<link rel="stylesheet" href="{% static 'css/style.css' %}" type="text/css" media="screen">

<link rel="stylesheet" href="{% static 'css/grid.css' %}" type="text/css" media="screen">

<link href="https://fonts.googleapis.com/css?family=Playfair+Display" rel="stylesheet">

<script src="{% static 'js/jquery-3.2.1.min.js' %}" type="text/javascript"></script>

<script src="{% static 'js/FF-cash.js' %}" type="text/javascript"></script>

<script src="{% static 'js/script.js' %}" type="text/javascript"></script>

{% block css %}{% endblock %}

</head>

<body class="{% block page\_name %}{% endblock %}">

<div class=" bg">

<heade <div class="menu-row">

<div class="main">

<div class="container\_12">

<div class="wrapper">

<div class="grid\_12">

<nav class="wrapper">

<ul class="menu">

<li><a class="about" href="{% url 'about' %}">Про нас</a></li>

<li><a class="services" href="{% url 'services' %}">Услуги</a></li>

<li><a class="employees" href="{% url 'employees' %}">Специалисты</a></li>

<li class="last-item"><a class="contacts" href="{% url 'contacts' %}">Связь</a></li>

</ul>

</nav>

<h1><span>SpaSalon</span></h1>

</div>

</div>

</div> <div class="ic">More <a href="http://www.templatemonster.com/">Website Templates</a> @

TemplateMonster.com - August29th 2011!

</div>

</div>

</div>

</header>

<!-- content -->

{% block content %}

{% endblock %}

</div>

<!-- footer -->

<footer>

<div class="main">

<div class="container\_12">

<div class="wrapper">

<div class="grid\_6">

<div class="footer-text">

<b><strong>Spa Salon</strong> <br> Донецк 2017</b> </div>

<ul class="list-services">

<li>

<a class="item-1" href="#"></a>

</li>

<li>

<a class="item-2" href="#"></a>

</li>

<li>

<a class="item-3" href="#"></a>

</li>

<li>

<a class="item-4" href="#"></a>

</li>

</ul>

<div class="">

<a href="/admin">Администратору сайта</a>

</div>

</div>

<div class="grid\_6">

<div class="support-phone">

<strong>Детальная информация:</strong>

<b>8(800)</b>154-45-67

</div>

</div>

</div>

</div>

</div></footer>

<script> {% block js %}{% endblock %}

</script>

</body></html>

Листинг Б.2 — Шаблон страницы «Услуги»

{% extends 'store/base.html' %}

{% load staticfiles %}

{% block css %}{% endblock %}

{% block page\_name %}services{% endblock %}

{% block content %}

<section id="content">

<div class="main">

<div class="container\_12">

<div class="wrapper">

<h3 class="p2">Все, что вы пожелаете</h3>

</div>

{% for service in services %} {% cycle '<div class="wrapper border-top">' '' %}

<article class="grid\_6" style="padding-bottom: 20px">

<figure class=" border grid\_3">

<img width="100%" src="{{ service.image.url }}" alt=""/>

<p></p>

<p>Стоимость - {{ service.cost }} руб.</p> <p>Продолжительность - {{ service.time }}(ч:м).</p>

</figure>

<h6 class="prev-indent-bot"><a class="link" href="#">{{ service.name }}</a></h6>

<p class="indent-bot">{{ service.description }}</p>

</article>

{% cycle '' '</div>' %}

{% endfor %}

</div>

</div>

</section>

{% endblock %}

{% block js %}{% endblock %}

Листинг Б.3 — Файл admin.py

@admin.register(Client)

class ClientAdmin(ModelAdmin):

# функция получения полей для формы создания и редактирования

def get\_fields(self, request, obj=None):

list\_display = ['full\_name', 'phone', 'account', 'email']

if not request.user.is\_superuser:

list\_display.remove('account')

return list\_display

# ограничение пользователя на просмотр чужих профилей

def get\_queryset(self, request):

qs = super(ClientAdmin, self).get\_queryset(request)

if request.user.is\_superuser:

return qs

return qs.filter(account=request.user)

# Функция сохранения, добавляет модель и прикрепляет к создателю, если это не админ

def save\_model(self, request, obj, form, change):

if not change and not request.user.is\_superuser:

obj.account = request.user

obj.save()

@admin.register(Employee)

class EmployeeAdmin(ModelAdmin):

# функция получения полей для формы создания и редактирования

def get\_fields(self, request, obj=None):

list\_display = super(EmployeeAdmin, self).get\_fields(request, obj)

if not request.user.is\_superuser:

list\_display.remove('account')

return list\_display

# ограничение сотрудника на просмотр чужих профилей

def get\_queryset(self, request):

qs = super(EmployeeAdmin, self).get\_queryset(request)

if request.user.is\_superuser:

return qs

return qs.filter(account=request.user)

# ограничение сотрудника на редактирование информации

def get\_readonly\_fields(self, request, obj=None):

if not request.user.is\_superuser:

readonly\_fields = ['full\_name', 'post', 'date\_of\_employment']

if not request.user.groups.filter(name='Сотрудник').exists():

readonly\_fields.remove('notes')

return readonly\_fields

else:

return []

class MyModelChoiceField(ModelChoiceField):

def label\_from\_instance(self, obj):

return obj.client

# форма с методом для валидации записи на посещение

class RecordForm(forms.ModelForm):

def clean(self):

cleaned\_data = super(RecordForm, self).clean()

employee = cleaned\_data.get('employee')

service = cleaned\_data.get('service')

employee\_services = []

try:

employee\_services = ', '.join([i.name.lower() for i in Service.objects.filter(employees=employee)])

services = Service.objects.get(Q(employees=employee), Q(id=service.id))

except Service.DoesNotExist:

self.add\_error('employee',

'Сотрудник не предоставляет выбранную услугу. Услуги мастера: {0}.'.format(

employee\_services))

# регистрация своей модели в админку

@admin.register(Record)

class RecordAdmin(ModelAdmin):

form = RecordForm

list\_display = ['client\_full\_name', 'employee\_full\_name', 'visit\_date', 'client\_status', 'employee\_status']

def client\_full\_name(self, obj):

return obj.client.full\_name

client\_full\_name.allow\_tags = True

client\_full\_name.short\_description = 'ФИО'

def employee\_full\_name(self, obj):

return obj.employee.full\_name

employee\_full\_name.allow\_tags = True

employee\_full\_name.short\_description = 'Мастер'

def formfield\_for\_foreignkey(self, db\_field, request, \*\*kwargs):

if db\_field.name == "client":

kwargs["queryset"] = Client.objects.filter(account=request.user)

return super().formfield\_for\_foreignkey(db\_field, request, \*\*kwargs)

def get\_list\_filter(self, request):

list\_filter = ['visit\_date']

if not request.user.groups.filter(name='Пользователь').exists():

list\_filter.append('client')

if not request.user.groups.filter(name='Сотрудник').exists():

list\_filter.append('employee')

return list\_filter

'''

функция получения полей для формы создания и редактирования,

собирает все поля для отображения, но вырезает некоторые

в зависимости от группы текущего пользователя

'''

def get\_fields(self, request, obj=None):

self.obj\_copy = deepcopy(obj)

list\_display = [f.name for f in Record.\_meta.get\_fields()]

list\_display.remove('id')

if request.user.groups.filter(name='Пользователь').exists():

list\_display.remove('employee\_status')

if request.user.groups.filter(name='Сотрудник').exists():

list\_display.remove('client\_status')

return list\_display

'''

Функция установки всех полей в режим "только для чтения",

если прием уже начался или закончился.

Доступно только поле "Заметки" всем, кроме сотрудников.

Так пользователи могут оставить комментарий об посещении.

'''

def get\_readonly\_fields(self, request, obj=None):

if obj:

if obj.visit\_date < timezone.now():

readonly\_fields = [f.name for f in Record.\_meta.get\_fields()]

if not request.user.groups.filter(name='Сотрудник').exists():

readonly\_fields.remove('notes')

return readonly\_fields

return []

'''

функция удаляет кнопку удаления записи для пользователя,

если прием уже начался или закончился

'''

def has\_delete\_permission(self, request, obj=None):

if obj:

if obj.client\_status and obj.employee\_status and obj.visit\_date < timezone.now() and not request.user.is\_superuser and not request.user.groups.filter(

name='Сотрудник').exists():

return False

return True