|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**Ознакомительная практика**

приказ Университета о направлении на практику от «09» февраля 2022 г. № 1103-С

|  |  |  |
| --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студент группы ИКБО-02-21 | «\_\_» июня 2022 | (подпись и расшифровка подписи) |
| Отчет утвержден.  Допущен к защите: |  |  |
| Руководитель практики  от кафедры | «\_\_» июня 2022 | (подпись и расшифровка подписи) |

Москва 2022 г.

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**Ознакомительная практика**

**Студенту 1 курса учебной группы ИКБО-02-21**

**Никитиной Валерии Александровне**

**Место и время практики:** РТУ МИРЭА кафедра ИиППО, с 09 февраля 2022 г. по 31 мая 2022 г.

**Должность на практике:**  студент

**1. СОДЕРЖАНИЕ ПРАКТИКИ:**

1.1. Изучить: технологии разработки веб-приложений

1.2. Проанализировать: методологию разработки клиент-серверного приложения, в частности веб-приложений

1.3. Ознакомиться: с фреймворками языков высокого уровня для разработки веб-приложений

**2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** подготовить доклад на научно-техническую конференцию студентов и аспирантов РТУ МИРЭА, подготовить презентационный материал

**3. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** В процессе практики рекомендуется использовать периодические издания и отраслевую литературу годом издания не старше 5 лет от даты начала прохождения практики

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель практики от кафедры  «09» февраля 2022 г. | | | *Подпись* | | (Литвинов В.В.) |
| Задание получил | |  |  | |  |
| «09» февраля 2022 г. | |  | *Подпись* | | (Никитина В.В.) |
|  | |  |  | |  |
|  | |  |  | |  |
|  | |  |  | |  |
|  | |  |  | |  |
|  | |  |  | |  |
| **СОГЛАСОВАНО:** | |  |  | |  |
| Заведующий кафедрой: | | | | | |
| «09» февраля 2022 г. | | | *Подпись* | | (Болбаков Р.Г.) |
| **Проведенные инструктажи:** |  | | |  | |
| Охрана труда: |  | | | «09» февраля 2022 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. каф. ИиППО | |
| Инструктируемый | *Подпись* | | | Никитина В.А. | |
|  |  | | |  | |
| Техника безопасности: |  | | | «09» февраля 2022 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. каф. ИиППО | |
| Инструктируемый | *Подпись* | | | Никитина В.А. | |
|  |  | | |  | |
| Пожарная безопасность: |  | | | «09» февраля 2022 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. каф. ИиППО | |
| Инструктируемый | *Подпись* | | | Никитина В.А. | |
|  | | | |  | |
| С правилами внутреннего распорядка ознакомлен: | | | | «09» февраля 2022 г. | |
|  | *Подпись* | | | Никитина В.А. | |

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ  
ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКИ**

студента Никитиной В.А. 1 курса группы ИКБО-02-21 очной формы обучения, обучающегося по направлению подготовки 09.03.04 Программная инженерия

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки**  **выполнения** | **Этап** | **Отметка о выполнении** |
| 1 | 09.02.2022 | Подготовительный этап, включающий в себя организационное собрание (Вводная лекция о порядке организации и прохождения учебной практики, инструктаж по технике безопасности, получение задания на практику) |  |
| 2 | 14.02.2022 | Подготовительный этап (Участие в круглом столе на тему «Проблема достоверности информации в современном мире») |  |
| 3 | 21.02.2022 | Подготовительный этап (Участие в круглом столе на тему «Информационно-коммуникационные технологии для организации информационного процесса») |  |
| 4 | 28.02.2022 | Подготовительный этап (Участие в круглом столе на тему «Информационная и библиографическая культура») |  |
| 5 | 07.03.2022 | Исследовательский этап (Поиск, отбор и анализ материалов для выполнения задания по практике) |  |
| 8 | 28.03.2022 | Представление руководителю (лично) структурированного материала: аналитический обзор предметной области |  |
| 10 | 11.04.2022 | Технологический этап (исследование по теме) |  |
| 14 | 09.05.2022 | Представление руководителю (лично) результаты исследования |  |
| 14 | 09.05.2022 | Согласование с руководителем доклада на научно-техническую конференцию студентов и аспирантов РТУ МИРЭА |  |
| 16 | 31.05.2022 | Подготовка окончательной версии отчета (Оформление материалов отчета в полном соответствии с требованиями на оформление письменных учебных работ студентов) |  |

Руководитель практики от  
кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Литвинов В.В., ассистент/

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Никитина В.А./

**Согласовано:**

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Болбаков Р.Г., к.т.н., доцент/

ОГЛАВЛЕНИЕ

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 7](#_Toc103784699)

[ВВЕДЕНИЕ 9](#_Toc103784700)

[1. Аналитический обзор технологии разработки веб-приложений 10](#_Toc103784701)

[1.1. Типы web-приложений 10](#_Toc103784702)

[1.2. Серверные web-приложения 10](#_Toc103784703)

[1.3. Клиентские web-приложения 11](#_Toc103784704)

[1.4. SPA приложения 11](#_Toc103784705)

[1.5. MPA приложения 11](#_Toc103784706)

[1.6. PWA приложения 12](#_Toc103784707)

[1.7. Выводы к главе 1 13](#_Toc103784708)

[2. Принцип работы web-приложений 13](#_Toc103784709)

[2.1. Клиентская часть (Frontend) 13](#_Toc103784710)

[2.2. Серверная часть (Backend) 14](#_Toc103784711)

[2.3. HTTP 16](#_Toc103784712)

[2.4. HTTP-запросы 18](#_Toc103784713)

[2.5. Выводы к главе 2 18](#_Toc103784714)

[3. Фреймворки для разработки веб-приложений (на примере Django) 19](#_Toc103784715)

[3.1. Почему Django? 19](#_Toc103784716)

[3.2. Как выглядит код Django? 20](#_Toc103784717)

[3.3. Выводы к главе 3 22](#_Toc103784718)

[ЗАКЛЮЧЕНИЕ 23](#_Toc103784719)

[СПИСОК ЛИТЕРАТУРЫ 24](#_Toc103784720)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**Ethernet - семейство технологий пакетной передачи данных между устройствами для компьютерных и промышленных сетей.**

**MAC-адрес - уникальный идентификатор, присваиваемый каждой единице активного оборудования или некоторым их интерфейсам в компьютерных сетях Ethernet.**

**IP-адрес** - уникальный числовой идентификатор устройства в компьютерной сети, работающей по протоколу IP.

**TCP - Transmission Control Protocol (один из основных протоколов передачи данных интернета. Предназначен для управления передачей данных интернета).**

**UDP - User Datagram Protocol (простой, ориентированный на дейтаграммы протокол без организации соединения, предоставляющий быстрое, но необязательно надежное транспортное обслуживание).**

**HTTP - HyperText Transfer Protocol (протокол прикладного уровня передачи данных).**

**SMTP - Simple Mail Transfer Protocol (широко используемый сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP).**

**FTP - File Transfer Protocol (это стандартный протокол связи, используемый для передачи компьютерных файлов с сервера клиенту в компьютерной сети).**

**SSH - Secure Shell Protocol (сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой)**

**SPA** - Single Page Application (одностраничные веб-приложения).

**MPA** - **Multi Page Application** (многостраничные веб-приложения).

**PWA** - **Progressive Web Application** (прогрессивные веб-приложения).

**AJAX - Asynchronous JavaScript and XML (подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.).**

# ВВЕДЕНИЕ

**“Web-приложение** – программа с определенным набором функционала, использующая в качестве клиента браузер. То есть, приложение можно отнести к веб-приложению, если приложению для осуществления бизнес-логики требуется сетевое соединение и наличие на стороне пользователя браузера.[1]”

В настоящее время можно выделить 3 типа приложений:

* десктопные;
* мобильные;
* Web.

Десктопные приложения предполагают установку клиента на стороне пользователя. В зависимости от типа операционной системы, процессора, видеокарты и других параметров могут потребоваться разные версии программы. Это создает определенные неудобства как разработчикам (им потребуется постоянно искать ошибки в разных средах), так и пользователям (необходимость скачивать постоянные обновления).

Мобильные приложения предназначены исключительно для смартфонов и планшетов с учетом установленной там системы (Android, iOS и др.). Это также является проблемой для разработчиков. Важно отметить, что многие мобильные приложения фактически являются web-приложениями.

На сегодняшний день наиболее динамично развиваются web-приложения, так как это полноценная программа, доступ к которой пользователь получает через интернет, то есть она не требует установки на устройство.

# Аналитический обзор технологии разработки веб-приложений

# 1.1. Типы web-приложений

Хоть первые web-приложения и появились в конце 20 века, до сих пор нет единой классификации их видов. Это связано с тем, что последние 5-10 лет их развитие совершило революционный скачок, породив новые разновидности.

В общем виде все web-приложения можно разбить на 5 типов. Деление в некоторой степени условное, так как возможно сочетание нескольких типов в одном приложении:

* Серверные web-приложения
* Клиентские web-приложения
* SPA приложения
* MPA приложения
* PWA приложения

# 1.2. Серверные web-приложения

Серверные web-приложения работают на удаленных компьютерах. Для их написания используют такие языки программирования: Python, Java, Ruby, PHP, C# и др. Они практически не требуют пользовательского вмешательства. Переход между страницами вызывает генерацию нового контента, который отображается у клиента.

Пример чисто серверного приложения – **push-уведомления** (от почтовых сервисов, мессенджеров, операторов связи) в смартфонах. Клиент получает информацию, что появилось новое сообщение, письмо, изменения в тарифе, не предпринимая для этого никаких действий.

# 1.3. Клиентские web-приложения

Клиентские приложения в чистом виде не требуют серверной части и обходятся возможностями JavaScript, используя в качестве оболочки браузер пользователя. Они не сохраняют результат своей работы дольше одной сессии.

Типичные примеры таких приложений: простые игры, браузерный фоторедактор.

# 1.4. SPA приложения

**“Single page application** (**SPA**, одностраничные веб-приложения) реализуют сложный функционал в рамках одного окна браузера без перезагрузки. Динамическое обновление содержимого страницы достигается технологией **AJAX**(**Asynchronous JavaScript and XML**, асинхронный **JavaScript** и **XML**). В ответ на действия пользователя (прокрутка страницы, нажатие кнопок, заполнение формы, движение ползунка и т.п.) содержание страницы будет меняться.[5]”

Неопытный пользователь может даже посчитать, что столкнулся с десктопным приложением, так как все изменения происходят практически моментально. К слову, многие мобильные приложения используют такой подход.

В сочетании с фреймворками **JavaScript**(**Angular**, **React**, **Vue**) работа таких программ становится максимально плавной.

Практически все почтовые сервисы являются **SPA**.

# 1.5. MPA приложения

**“Multi Page Application** (**MPA**, многостраничные web-приложения) применяются для построения сложных систем. В данном случае любые изменения в данных приводят к полной перезагрузке страницы.[5]” Когда имеется большой массив данных и контента, разнообразие представляемых сведений, **MPA**подходят лучше всего.

Несмотря на то, что они требуют больших объемов ресурсов для реализации и существенно дороже, другие виды web-приложений их заменить не могут. Однако тенденции показывают, что общая доля **MPA**постепенно снижается.  
Стандартный пример такого приложения – интернет-магазины с большим массивом товаров.

# 1.6. PWA приложения

**“Progressive Web Application** (**PWA**, прогрессивные web-приложения) – новый способ «подачи» web-сервисов, который максимально сближает их с обычным, привычным десктопным приложением, но на качественно более высоком уровне.[5]”

Представим ситуацию: человек посещает некоторый сайт, который предлагает ему установить его на другие устройства. Теперь и на ПК, и в телефоне вы сможете получать уведомления, работать оффлайн, независимо от модели устройств и их мощностей.

Главная область применения таких приложений – мобильные устройства. Пользователю больше не нужно входить на AppStore или PlayMarket, чтобы скачать программу – все сделает браузер автоматически (а еще создаст ярлык на рабочем столе, позволит работать с собой без доступа к сети и т.д.). Фактически, мы получаем аналог обычного приложения с тем же функционалом и множеством плюсов (для **PWA**не требуется лишняя память смартфона, они весят менее 1 Мб).

При появлении нового контента **PWA**отправляет пользователю push-уведомление. Следует признать, что в скором времени эти приложения смогут заменить практически все мобильные аналоги. PWA-версии приложений встречаются у многих компаний (тот же Aliexpress почти в 2 раза повысил конверсию от новых посетителей благодаря этому).

1.7. Выводы к главе 1

Таблица 1 - Основные преимущества и недостатки SPA, MPA и PWA

приложений:

|  |  |  |
| --- | --- | --- |
| Приложение | Преимущества | Недостатки |
| SPA | Высокая скорость работы, кеширование данных, быстрая разработка | Нагрузка на браузер, возможны утечки памяти и уязвимости в безопасности |
| MPA | Привычны посетителям, легкая seo-оптимизация | Сложная разработка и поддержка, существенные финансовые траты |
| PWA | Быстрая разработка, высокая скорость загрузки, оффлайн-работа | Поддерживается не всеми браузерами и платформами |

# Принцип работы web-приложений

Для понимания функционирования web-приложения следует рассмотреть его основные структурные компоненты.

В сети устройства, имеющие выход в Интернет, принято называть узлами. Ими могут быть:

* компьютеры;
* мобильные аппараты;
* роутеры.

Чтобы описать работу сетевых приложений следует разделить узлы на серверы и клиенты. За серверную часть отвечает **backend-разработка**, а за клиентскую – **frontend**.

# 2.1. Клиентская часть (Frontend)

**Frontend**подразумевает создание визуальной части приложения, которая выполняет функции на стороне пользователя. Это все то, что посетитель сайта видит своими глазами, с чем он может напрямую взаимодействовать (от дизайна до отдельных элементов, на которые можно нажать, ввести текст, подвигать и т.д.).

# 2.2. Серверная часть (Backend)

**Backend**связан с тем, что пользователь визуально никак оценить не может и к чему не имеет доступа. Это логика работы приложения, осуществляемая на удаленном сервере.

Когда вы открываете страницу приложения, то от вас поступает запрос на сервер. Там он обрабатывается, а пользователю возвращается сгенерированная web-страница. Взаимодействия со страницей также ведут к формированию запросов на сервер (заполнение формы, просмотр увеличенной фотографии, поисковый запрос и т.д.).

“Разработка серверной части приложений связана с рядом языков программирования (**Java**, **Python**, **C#**, **C++**), а упрощение такой разработки достигается использованием бэкэнд-фреймворков (**Django**) и веб-серверов (**Nginx**, **Node.js** и др.).[3]”

Немаловажной частью бэкэнда являются базы данных. Они нужны для хранения массивов данных, которые при запросе пользователя извлекаются и отображаются в веб-приложении. На практике могут использоваться разные базы данных, наиболее популярные из которых: PostgreSQL, MySQL, MongoDB.

Следовательно, используя **web-приложение** мы имеем дело с клиентской и серверной частями, упрощенное взаимодействие которых показано на рисунке.

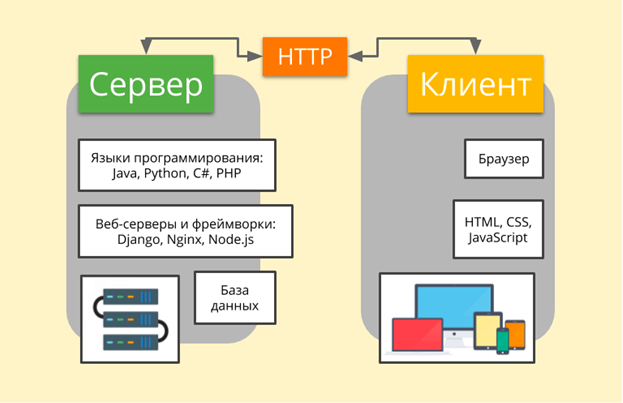


Рисунок 1 – Взаимодействие серверной и клиентской

частей веб-приложения

“От клиента через браузер поступает запрос на сервер (для этого используется **HTTP-протокол**). Сервер получает этот запрос, обрабатывает его, достает из базы данных соответствующие сведения и отправляет обратно клиенту.

На стороне клиента загружается страница, которая представляется в необходимом виде при помощи **HTML**, **JS**и **CSS**.[7]”

Такая связь между клиентом и сервером способна устанавливаться многократно по мере изменения запросов, данных на сайте, обновлении серверной части и т.д. При этом может осуществляться перезагрузка страниц, происходить дозагрузка данных на лету без перезагрузки приложения. В некоторых случаях данные кешируются для ускорения обработки повторных запросов.

Чтобы сервер и клиент понимали друг друга, применяется **HTTP-протокол**. В последнее время в целях безопасности используется расширение **HTTPS**, которое при помощи криптографических протоколов **TLS**шифрует данные. Это необходимо для исключения возможности перехвата сведений сторонними лицами или устройствами, которым они не адресованы.

# 2.3. HTTP

“Уровень приложения является самым верхним в модели **TCP/IP**. Работа web-приложений связана с протоколом **HTTP**данного уровня. Для лучшего понимания функционирования сетевых приложений необходимо детально рассмотреть **HTTP-протокол**, его специфику, используемые запросы.[4]”

**“HTTP** (**HyperText Transfer Protocol**, протокол передачи гипертекста) изначально предназначался для передачи данных в виде **HTML-документов**, а сегодня отвечает за передачу любых данных в клиент-серверном приложении.[2]”

Программное обеспечение для работы с **HTTP-протоколом** делят на 3 категории:

* серверы (обрабатывают запросы);
* клиенты (отправляют запросы, потребляют информацию);
* прокси (транспортные службы).

**“HTTP-протокол** имеет несколько версий. Наиболее ходовой в нынешнее время считается **HTTP/1.1**. **HTTP/2** введен в обиход в 2015 году и позволяет уплотнять канал (передавать большее количество запросов при имеющихся мощностях). В 2019 году появилась третья версия протокола, предполагающая замену **TCP**на **UDP**в роли основного транспортного протокола.[2]”

Между клиентом и сервером передаются **HTTP-сообщения**(**HTTP-запросы** и **HTTP-ответы**). Их структура всегда одинакова (содержание может отличаться) и представлена в указанном на рисунке порядке:



Рисунок 2 – Структура HTTP-сообщения

1. В стартовой строке указывается тип запроса, адрес ресурса и версия протокола, код состояния и др.
2. В заголовках содержатся сведения о теле сообщения, применяемые параметры передачи.
3. В теле сообщения передаются сами данные, отделенные от заголовков пустой строкой.

Заголовок и строка обязательно должны присутствовать в сообщении, тела может не быть.

**Коды состояния** – часть ответов сервера, содержащая трехзначное число и поясняющую фразу. Их существует 5 классов:

* информационные (1хх)
* говорящие об успехе передачи (2хх)
* перенаправления (3хх)
* ошибки на стороне клиента (4хх)
* ошибки на стороне сервера (5хх)

Наиболее известные: 404 (нет такой страницы), 200 (все хорошо, ошибок нет), 503 (не удалось получить доступ к сервису).

# 2.4. HTTP-запросы

Чтобы сервер понял, чего именно хочет клиент, используются HTTP-запросы (методы). Изначально применялся только один метод – GET, но сегодня их список намного шире.

Таблица 2 – HTTP-запросы

|  |  |
| --- | --- |
| Метод | Характеристика |
| OPTIONS | Определяет возможности сервера |
| GET | Получение необходимых данных с сервера |
| HEAD | Ответ сервера не содержит ссылок, нужно для валидации ссылок |
| POST | Передача сведений от пользователя |
| PUT | Отвечает за перезапись имеющегося ресурса |
| PATCH | Перезаписывает только часть ресурса |
| DELETE | Удаление данных |
| TRACE | Позволяет отслеживать информацию, изменяемую промежуточными серверами в запросах |
| CONNECT | Создание зашифрованного туннеля между клиентом и сервером с использованием прокси |

# 2.5. Выводы к главе 2

Веб-приложения состоят из серверной части (back-end, бэкенд) и клиентской части (front-end, фронтенд). Пользователи взаимодействуют с клиентской частью через интерфейс, который отображается в браузере (Chrome, Firefox, Safari, Edge и др.). По команде пользователя запрос отправляется на сервер через интернет. На сервере его обрабатывает серверный код и возвращает клиенту ответ.

# Фреймворки для разработки веб-приложений (на примере Django)

“При создании сложных веб-приложений прибегают к помощи программных веб-каркасов (веб-фреймворки), которые позволяют значительно сократить затраты на разработку полноценного веб-приложения, так как реализует множество стандартных функций приложения.[7]”

Веб-фреймворк — это набор модульных инструментов, которые абстрагируют большую часть трудностей и повторений, присущих веб-разработке. Например, большинству веб-сайтов нужен такой же базовый функционал: возможность подключения к базе данных, установка URL маршрутов, отображение контента на странице, корректная защита и так далее. Вместо того, чтобы воссоздавать все это с нуля, программисты на протяжении многих лет создавали веб-фреймворки на всех основных языках программирования: Django и Flask в Python, Rails в Ruby и Express в JavaScript среди многих, многих других.

# 3.1. Почему Django?

Django представляет собой высокоуровневую платформу, которая позволяет создавать веб-приложения, написав всего несколько строк программного кода. Эта платформа отличается простотой и гибкостью, позволяя без труда создавать собственные решения. “Платформа Django написана на языке Python, объектно-ориентированном языке программирования, который соединяет в себе мощь таких языков системного программирования, как C/C++ и Java, с непринужденностью и скоростью разработки языков сценариев, таких как Ruby и Visual Basic.[8]” Это дает пользователям возможность создавать приложения, способные решать самые разнообразные задачи.

Django унаследовал подход Python "batteries-included" и включает в себя поддержку из коробки для общих задач в веб-разработке:

• аутентификация пользователя;

• шаблоны, маршруты и представления;

• интерфейс администратора;

• надежная безопасность;

• поддержка нескольких серверных баз данных;

• и многое другое.

Такой подход значительно упрощает работу веб-разработчиков. Можно сосредоточиться на том, что делает веб-приложение уникальным, а не изобретать колесо, когда дело доходит до стандартной функциональности веб-приложения. Напротив, несколько популярных фреймворков, в первую очередь Flask на Python и Express на JavaScript, используют подход "микропрограммирования". Они предоставляют только минимум, необходимый для простой веб-страницы, и оставляют на усмотрение разработчика установку и настройку сторонних пакетов для репликации базовой функциональности веб-сайта. Такой подход обеспечивает большую гибкость для разработчика, но также дает больше возможностей для ошибок.

По состоянию на 2018 Год Django находится в активной разработке более 13 лет. Миллионы программистов уже использовали Django для создания своих сайтов. И это, несомненно, хорошо. Веб-разработка трудна. Не имеет смысла повторять один и тот же код и ошибки, когда большое сообщество блестящих разработчиков уже решило эти проблемы для нас.

В то же время, Django остается в стадии активной разработки и имеет годовой график выпуска. Сообщество Django постоянно добавляет новые функции и улучшения безопасности. Если вы создаете сайт с нуля Джанго это отличный выбор.

# 3.2. Как выглядит код Django?

“На традиционном информационном веб-сайте веб-приложение ожидает HTTP-запросы от веб-браузера (или другого клиента). Когда запрос получен, приложение разрабатывает то, что необходимо на основе URL-адреса и данных в POST или GET запросах. В зависимости от того, что требуется, далее он может читать или записывать информацию из базы данных или выполнять другие задачи, необходимые для удовлетворения запроса. Затем приложение вернёт ответ веб-браузеру, часто динамически создавая HTML-страницу для отображения в браузере, вставляя полученные данные в HTML-шаблон.[8]”

Веб-приложения, написанные на Django, обычно группируют код, который обрабатывает каждый из этих шагов, в отдельные файлы:

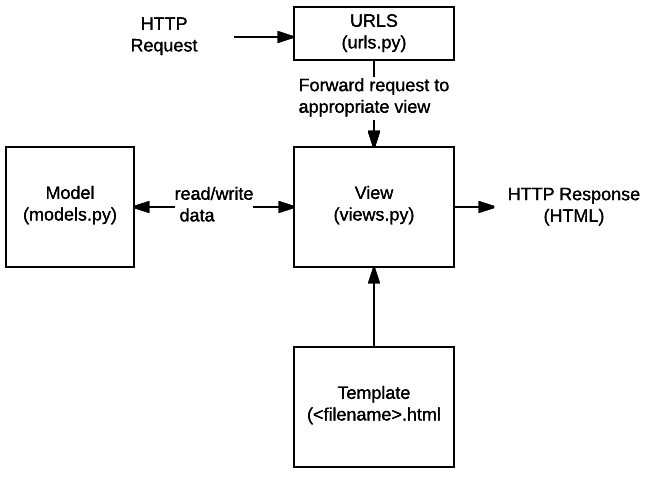


Рисунок 3 – Группировка кода на Django

**URLs**: хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения в виде аргументов.

**View:** view — это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.

**Models**: модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

**Templates**: template — это текстовый файл, определяющий структуру или разметку страницы, с полями для подстановки, которые используются для вывода актуального содержимого. View может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели.

# 3.3. Выводы к главе 3

Django - отличный выбор для любого разработчика, который хочет создавать современные, надежные веб-приложения с минимальным количеством кода. Он популярен, находится в стадии активной разработки и тщательно протестирован крупнейшими веб-сайтами в мире.

# ЗАКЛЮЧЕНИЕ

Веб-приложения уверенно занимают свое место в интернете и продолжают эволюционировать. Это обусловлено удобством их применения, а также готовностью к использованию на мобильных устройствах. Web-приложения представляют собой особый тип программ, основанных на архитектуре "клиент-сервер". Web-приложение располагается и исполняется на сервере, получая от клиента исходные данные для работы, а также передавая ему результаты работы в виде HTML-кода, отображаемого в браузере. Создание сайта представляет собой маркетинговый шаг, направленный на создание информационного ресурса, который предоставит возможность для компании как удержать старых клиентов, так и привлечь новых.

# СПИСОК ЛИТЕРАТУРЫ

1. Лоран Б. Создание web-приложений в Silverlight / Б. Лоран. - Москва: Мир, 2012. - 79 c.
2. Дейв М. Разработка приложений для iPhone, iPad и iPod touch с использованием iOS SDK / М. Дейв - Уильямс: Диалектика, 2012. - 77 c.
3. Дейтел Х. М. Как программировать для Internet & WWW / Х.М. Дейтел, П.Дж. Дейтел, Т.Р. Нието. - Москва: Бином, 2010. - 98 c.
4. Мол Д. Создание облачных, мобильных и веб-приложений на F# / Д. Мол. - Москва: ДМК Пресс, 2012. - 79 c.
5. Панфилов К. Создание веб-сайта от замысла до реализации / К. Панфилов. – Москва: ДМК Пресс, 2013. - 56 c.
6. Русеев, С. WAP. Технология и приложения. Наиболее полное руководство / С. Русеев. - Москва: БХВ-Петербург, 2017. - 15 c.
7. Шапошников, И. Web-сайт своими руками / И. Шапошников. - Москва: Книга по Требованию, 2013. - 38 c.
8. Форсье Д. Django. Разработка веб-приложений на Python / Д. Форсье, П. Биссекс, У. Чан. – Москва: Символ-Плюс, 2017. – 454 с.