**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Факультет информационных технологий и прикладной математики**

**Реферат**

на тему “Язык программирования Python.

Веб-фреймворк Django”

**по курсу**

**«Фундаментальная информатика»**

**I семестр**

|  |  |
| --- | --- |
| Студент:  Группа:  Руководитель:  Оценка:  Дата: | Рябов С.М.  М8О-104Б-21  Безлуцкая Е.Н. |

**Москва**

**2022г.**

**Содержание**

1. **Введение**
2. **Краткая история Python**
3. **Веб-фреймворк Django**

* Что такое Django?
* Веб-сервер и Django
* Возможности фреймворка
* Сравнение с фреймворком Flask, плюсы и минусы

1. **Создание своего локального сервера на Django**

* Структура кода

1. **Заключение**

**Введение**

Python – высокоуровневый язык программирования общего назначения. Стал одним из самых популярных языков. Его чаще всего используют в анализе данных, машинном обучении, в веб-разработке, а так же и в других сферах, включая разработку игр. Django – это свободный фреймворк для веб-приложений, как раз на языке Python, который используется в таких популярных сайтах как Intstagram, Mozilla, Pinterest, Youtube, Google. Из этого можно сделать вывод, что этот фреймворк используется на практике.

**Python – краткая история**

Программист Гвидо ван Россум
История языка программирования Python началась в конце 1980-х. Гвидо ван Россум приступил к созданию этого языка в в декабре 1989 года в центре математики и информатики в Нидерландах. Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Амёба.

*Гвидо ван Россум* Ван Россум является основным автором Python и продолжал выполнять центральную роль в принятии решений относительно развития языка вплоть до 12 июля 2018 года.

Первый прототип состоял из простой виртуальной машины, парсера и среды выполнения. В нём также присутствовал базовый синтаксис, операторы, словари, списки, строки и небольшое количество других типов данных. Главная фишка заключалась в том, что будущий Python предлагал гибкую модель расширяемости — то есть, помимо стандартных возможностей, каждый программист мог самостоятельно добавить в систему нужные типы объектов.

Прототип Python понравился разработчикам CWI, и многие сразу включились в процесс: начали использовать язык для внутренних проектов и помогли доработать код.

Как появилось название питон? Гвидо ван Россум считал нейминг пустой тратой времени и поэтому назвал свой язык в честь комедийного сериала «Летающий циру Монти Пайтона» - это одно из его любимых шоу и первая ассоциация, которая пришла ему в голову.

Со временем этот язык набирал популярность, благодаря своей простоте и удобству, и на сегодняшний день является одним из самых популярных языков программирования.

**Веб-фреймворк Django**

**Что такое Django?**

Django - бесплатный и свободный фреймворк для веб-приложений, написанный на Python. Фреймворк — это набор компонентов, которые помогают разрабатывать веб-сайты быстро и просто. Назван в честь музыканта Джанго Рейнхардт.

Каждый раз при разработке веб-сайтов требуются похожие компоненты: способ аутентифицировать пользователей (вход, выход, регистрация), панель управления сайтом, формы, инструменты для загрузки файлов и т. д.

Со временем люди обратили внимание на возникновение однотипных проблем при веб-разработке, так что они объединились и создали фреймворки, такие как Django и другие. Они предлагают готовые шаблоны для использования. В целом фреймворки и существуют, чтобы облегчить процесс разработки и позволить «не изобретать велосипед».

**Веб-сервер и Django**

Веб-сервер – это птростая компьютерная программа, основная задача которой является приём HTTP-запросов, обработка их и выдача HTTP-ответа. Django как раз и отвечает за создание контета, который будет отправлен в ответе.

Когда на сервер приходит запрос, он переадресуется Django, который пытается понять, что конкретно от него требуют. В начале распознается адрес веб-страницы, эту часть процесса выполняет urlresolver. Берет список шаблонов и пытается сопоставить их с URL. Если что-то совпадает, то Django перенаправляет запрос соответсвтвующей функции (которая называется view). В функции view мы уже можем обращаться, например, к базе данных за определенной информации. По запросу функция view генерирует ответ, а Django отправляет его веб-браузеру пользователя.

**Возможности фреймворка**

Dозможности Django:

• ORM, API доступа к БД с поддержкой транзакций (последовательных операций над БД)

• встроенный интерфейс администратора, с уже имеющимися переводами на многие языки

• диспетчер URL на основе регулярных выражений

• расширяемая система шаблонов с тегами и наследованием

• система кеширования

• интернационализация

• подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты

• «generic views» — шаблоны функций контроллеров

• авторизация и аутентификация, подключение внешних модулей аутентификации: LDAP, OpenID и проч.

• система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий

• библиотека для работы с формами (наследование, построение форм по существующей модели БД)

• встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение

**Сравнение с фреймворком Flask**

**Flask**и**Django** — оба взрослые, расширяемые веб фреймворка, которые, в корне своем, предоставляют аналогичный функционал в обработке запросов, поддержке документов, но различаются в масштабе ответственности.

Большая часть отличий между двумя фреймворками проистекают из разных подходов, остальные — из отличных основных проектных решений. Вот небольшой список ключевых различий, которые могут повлиять на ваше решение:

* Объект Request — Flask использует локальные потоки, а Django передает запрос там, где это нужно.
* Формы — Django доступен со встроенными формами, которые интегрируются с ORM и админкой сайта. Flask не поддерживает формы по умолчанию, но вы можете использовать WTForms, чтобы заполнить этот пробел.
* Базы данных — Django доступен со встроенной ORM и системой миграции, которая может управлять базами данных. Flask не может этим похвастаться, однако есть инструменты, такие как SQLAlchemy, которые предоставляют аналогичный функционал (или даже больше).
* Аутентификация и привилегии пользователям — Django предоставляет приложение аутентификации, которое предоставляет реализацию по умолчанию для пользовательского управления и привилегий. Flask предоставляет безопасные куки в качестве инструмента вашей собственной реализации.
* Панель администратора — Django включает в себя полностью интегрированный админ-интерфейс для управления данными приложения. Flask не имеет таких функций, но Flask-Admin — очень популярное расширение, которое можно использовать для создания аналогичного административного инструмента.

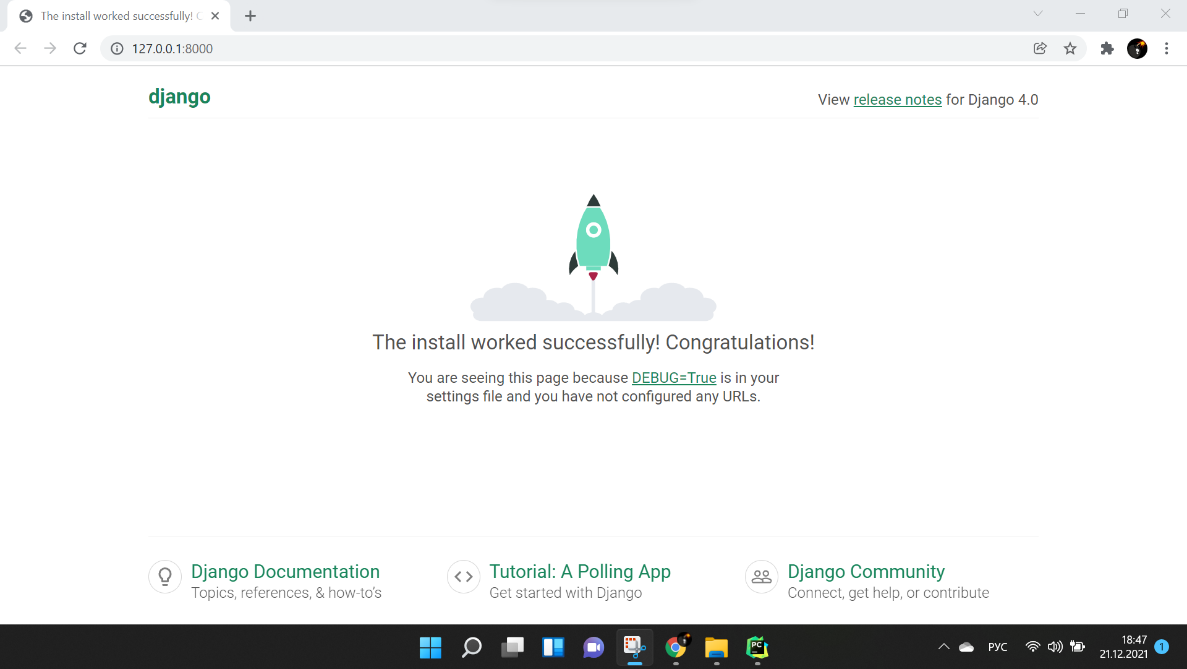
Если же использовать определения, которые дали сами разработчики у себя на официальных сайтах, то Django позволяет использовать меньше кода и делать это быстрее, чем тот же Flask. Но преимуществом второго фреймворка является то, что он позволяет более тонко контролировать весь процесс создания веб-сайта. То есть Django подойдет больше для быстрой и непыльной работы, когда Flask требует больших знаний и усидчивости, что влияет на более широкий функционал этой программы.

**Создание своего локального сервера**

**Создание проекта**

Я решил попробовать создать свой локальный сервер с возможностью генерации чисел.

Среда разработки PyCharm. При создании проекта я выбрал версию питона Python 3.10. После того как создал проект, сразу решил запустить сервер со стандартным портом 8000 и вот, что я увидел

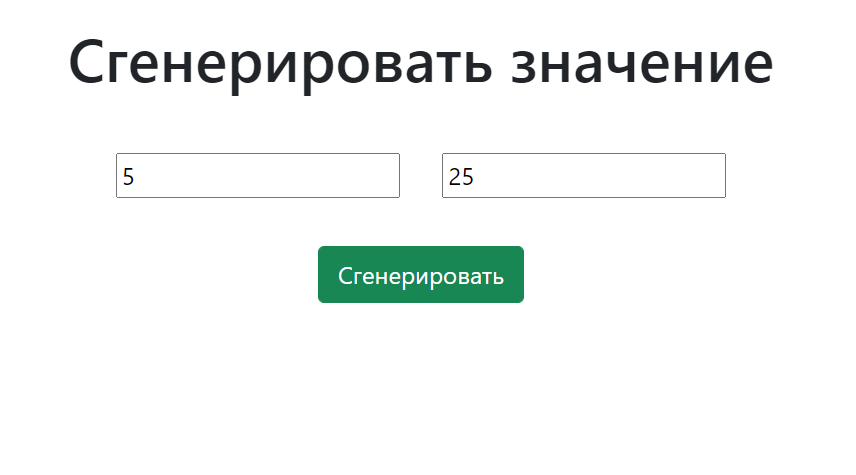


Это значит, что все установилось правильно и без ошибок.

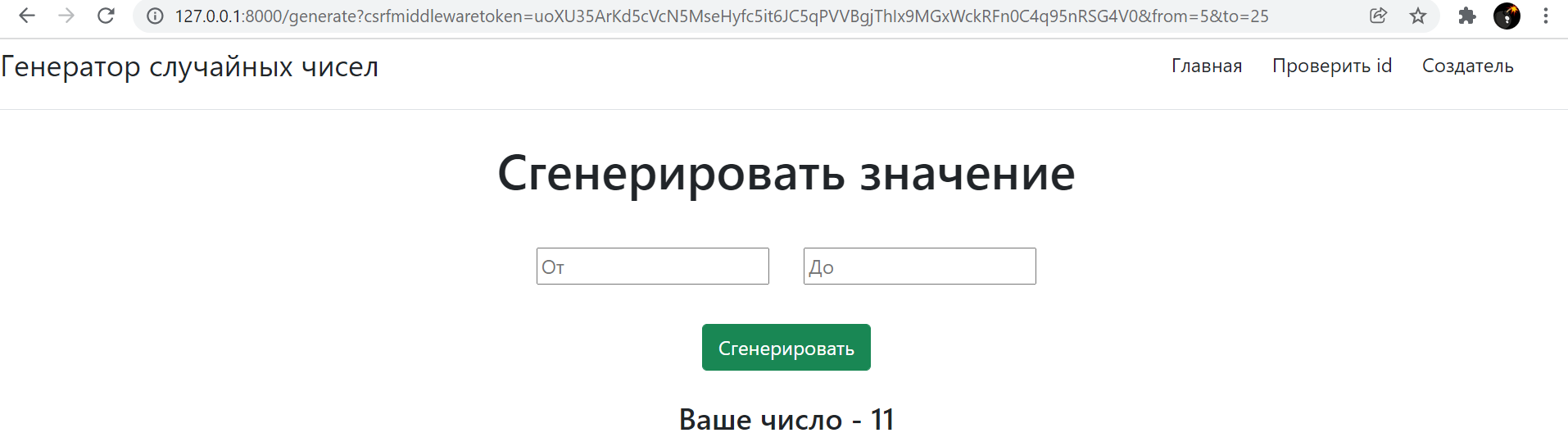
Далее передо мной стояла цель создать страницу, у которой будет возможность генерировать число, в пределах, которые задает пользователь. Для более менее приятного визуала, я решил скопировать готовый код с сайта ru.wix.com с шаблонами сайтов.

Следующим пунктом создания сайта с генерацией случайных чисел – это создание форм, в которые можно записывать границы генераций числа и кнопки, после нажатия на которую сайт запишет границы и выдаст значение. Формы принимают значения целых чисел, но при этом, то, что находится в форме записывается в переменную как строка. Из-за этого возникает некое неудобство в том, что нужно перевести из string в int, так же проблемы возникают в том, чтобы проверить корректность вводимых данных

Вот что у меня получилось.



После нажатия на кнопку «сгенерировать», пользователя перебрасывает на домен /generate после чего выводится сгенерированное значение



Можно заметить, что в строке с адресом локального сайта появились значения, from=5 to=25, что означает, мы запустили генерацию от 5 до 25. Если ввести какое нибудь неверное значение, например от 5 до 0, то выдаст ошибку, этот случай как раз был рассмотрен в коде программы, в файле views.py

Хотел еще реализовать возможность возвращения сгенерированного числа через id генерации, но возникла проблема с записью сгенерированного числа в БД.

Так вот, я самостоятельно попробовал написать локальный сервер с генерацией чисел.

**Структура кода.**

**Структура проекта такая**

1. DjangoMango/ (название проекта)
   * + - 1. .idea/
         2. \_\_pycache\_\_/
         3. djangoMango/
         4. main/

\_\_pycache\_\_/

migrations/

templates/

\_\_init\_\_.py

admin.py

apps.py

models.py

urls.py

views.py

* + - * 1. templates/
        2. db.sqlite3
        3. manage.py

Основной папкой является, как не удивительно, папка main. Она содержит в себе все основные файлы, требующиеся для стабильной работы программы.

Внутри main из основных находятся urls.py, views.py и models.py. Первый файл хранит в себе шаблоны URL, с которыми сопоставляет запрос HTTP. Далее, если что-то совпадает, то Django перенаправляет на функцию view. В файле views.py уже содержиться информация, то как надо обработать запрос на веб-сервер. То есть в этом файле содержиться код, который вызывается в той или иной ситуации, например сгенирировать какое-нибудь число, посчитать заданны значения или что-то в этом роде, даже обращение к базе данных может быть вызвано в этой функции. Если говорить про models.py, то в этом файле содержаться модели вывода информации, такие как шрифт, размер, цвет текста. Также там можеть быть информация о каких-нибудь фотография, которые должны изображаться на странице. Получается, что этот файл содержит в себе то, как будет внешне выглядеть файл, после обработки запроса.

Admin.py содержит информацию про админа локального сервера, то есть кто может использовать панель админа. Перед этим нужно зарегестрировать профиль через консоль, а дальше добавить профиль в этот файл.

Вне папки main есть еще такой каталог как djangoMango. Совпадает с названием проекта, это не просто так. В нем содержатся настройки самого проекта, их не стоит менять без необходимости, вот я и не трогал эту папку.

**Заключение.**

В заключение хочется сказать, что фреймворк Django отлично подходит для создания своих сайтов, довольно таки прост в освоении, благодаря тому, что много информации на различный иностранных форумах. Требует не таких завышенных навыков, например как у фреймворка Flask, тк представляет больше готовых решений для той или иной проблемы, с которыми встречаются большинство веб-разработчиков. Простоту в освоении я проверил на личном опыте, написав простенький локальный сервер. Этот фреймворк очень удобен и предоставляет большое количество возможностей, ведь не зря его используют крупные компании для создания и поддержания своих сайтов.