

< Teach
Me
Skills />

Занятие 26.

Документирование API

Подготовка к деплойменту

`nginx, gunicorn, uwsgi`

1

Документирование API

Зачем документировать API

- Коммуникация, понимание и согласованность между разработчиками
- Инструкция по использованию (для конечных пользователей API)
- Упрощение интеграции с другими приложениями и сервисами
- Улучшение опыта разработчиков (онбоардинг, разработка, использование, воспроизведение)
- Уменьшение количества необходимых ресурсов и времени (избегаем частых ошибок)

Документирование DRF

- автоматическая генерация интерактивной документации на основе кода
- использование аннотаций - описание полей (тип, description)
- предоставление примеров и схем данных
- поддержка разных форматов документации - Swagger, OpenAPI
- расширяемость и настройка (много кастомных опций)

Swagger

Swagger - это набор инструментов для разработки, описания и визуализации RESTful API. Он предоставляет средства автоматической генерации интерактивной документации для API на основе спецификации OpenAPI (ранее известной как Swagger Specification)

- 1. OpenAPI Specification (ранее Swagger Specification):** Это спецификация в формате JSON или YAML, которая описывает структуру API
- 2. Swagger Editor:** создание, редактирование и проверка спецификации OpenAPI. Он обеспечивает подсказки, валидацию и другие полезные функции для облегчения процесса создания спецификации.
- 3. Swagger UI:** Веб-интерфейс, который автоматически генерирует интерактивную документацию API на основе спецификации OpenAPI
- 4. Swagger Codegen:** Инструмент, который автоматически генерирует клиентский код для различных языков программирования на основе спецификации OpenAPI

Postman

Postman – это HTTP-клиент для тестирования API. HTTP-клиенты тестируют отправку запросов с клиента на сервер и получение ответа от сервера.

1. составлять и отправлять HTTP-запросы к API;
2. создавать коллекции (набор последовательных запросов) и папки запросов для сокращения времени тестирования;
3. менять параметры запросов (например ключи авторизации и URL);
4. менять окружения для запросов (например на тестовом стенде, локально или на сервере);
5. добавлять при вызове API контрольные точки (фиксацию момента передачи данных);
6. проводить автоматизированное тестирование API по коллекции запросов с помощью Collection Runner

Правила хорошего тона

2

Переменные окружения

- SECRET_KEY
- DEBUG
- Ключи для базы данных (пароль, хост, юзер, бд)
- любые API-ключи, токены, адреса электронной почты

все, что может быть изменено (и должно быть изменено без переписывания кода), а также что небезопасно хранить напрямую

.env

.env.template

load_dotenv()

requirements.txt

список необходимых для запуска приложения библиотек

может быть разделен на dev.txt, prod.txt и тд

для управления зависимостями также могут использоваться **poetry** и **pipenv**

в базовых настройках этих менеджеров уже осуществлено управление в зависимости от среды

settings.py

файл settings.py

или папка settings с файлами base.py, dev.py, prod.py

в таком случае нужно будет явно указывать файл с настройками для запуска (wsgi.py)

python manage.py runserver --settings=todos.settings.dev

- all settings files need to be version-controlled
- don't repeat yourself
- keep secret keys safe

.gitignore

```
1 .idea  
2 *__pycache__  
3 .env  
4 db.sqlite3  
5 venv  
6 notes  
7 /static
```

3

nginx, gunicorn, uicorn

Где можно развернуть django приложение

- на виртуальной машине - GCP VM, AWS EC2, etc
- в докере
- на heroku
- в kubernetes
- и многое другое

Окружение развертывания

Окружение развертывания - это среда, которое предоставляет сервер, на котором вы будете размещать свой веб-сайт для публичного запуска и доступа. Данное окружение включает в себя:

- Железо на котором будет запускаться сайт.
- Операционную систему (Linux, Windows).
- Языки программирования времени выполнения (скриптовые) и библиотеки, которые использует ваш сайт.
- Веб-сервер, используемый для обслуживания страниц и другого контента (Nginx, Apache, uwsgi, gunicorn).
- Сервер приложений, который передает "динамические" запросы между сайтом Django и веб-сервером.
- Базу данных, от которой зависит ваш сайт.

Зачем нужны asgi.py и wsgi.py

Файлы asgi.py и wsgi.py являются частью структуры проекта Django и необходимы для запуска веб-приложения на сервере.

- wsgi.py: Файл wsgi.py используется для запуска Django приложения на сервере, который поддерживает WSGI (Web Server Gateway Interface). WSGI является стандартным интерфейсом между веб-сервером и веб-приложением в Python. Файл wsgi.py содержит код, который загружает объект приложения Django и обрабатывает веб-запросы. Когда веб-сервер получает запрос от клиента, он передает его в wsgi.py, который затем передает запрос в соответствующее Django приложение для обработки.
- asgi.py: Файл asgi.py используется для запуска Django приложения на сервере, который поддерживает ASGI (Asynchronous Server Gateway Interface). ASGI предоставляет асинхронную архитектуру для обработки веб-запросов в Python. Файл asgi.py выполняет аналогичные задачи, что и wsgi.py, но для серверов, поддерживающих ASGI. В некоторых случаях, например, когда необходима асинхронная обработка запросов или взаимодействие с WebSockets, требуется использование ASGI сервера вместо WSGI.

Почему нельзя просто запустить manage.py в проде

Он удобен в использовании, но не рекомендуется для использования в продуктовой среде по следующим причинам:

1. Производительность: Встроенный сервер разработки в Django не предназначен для высоконагруженных производственных сред. Он медленнее и менее масштабируем, чем специализированные серверы, такие как Gunicorn или uWSGI.
2. Надежность: Встроенный сервер разработки не имеет механизмов автоматической перезагрузки или восстановления после сбоев, которые обеспечиваются веб-серверами, специально предназначенными для использования в продуктовой среде.
3. Безопасность: Встроенный сервер разработки не предназначен для использования в публичной сети и не обеспечивает необходимых механизмов безопасности, таких как SSL-шифрование или защита от атак.

Gunicorn

Gunicorn (сокращение от "Green Unicorn") - это сервер HTTP для обслуживания приложений Python, включая веб-приложения Django. Он является одним из наиболее популярных веб-серверов, используемых для развертывания приложений Django в производственной среде.

Gunicorn создан с учетом производительности и масштабируемости и может обрабатывать большое количество одновременных запросов. Он предоставляет поддержку для WSGI (Web Server Gateway Interface), что позволяет взаимодействовать с приложениями, реализующими этот стандарт.

Gunicorn

Преимущества использования Gunicorn:

- Производительность:** Gunicorn способен эффективно обрабатывать большое количество одновременных запросов благодаря многопроцессной архитектуре. Он может запускать несколько процессов-работников, обслуживающих запросы, и эффективно распределять нагрузку между ними.
- Масштабируемость:** Gunicorn обеспечивает горизонтальное масштабирование, позволяя запускать несколько экземпляров приложения и распределять нагрузку между ними. Это позволяет увеличить пропускную способность и обрабатывать больше запросов.
- Надежность:** Gunicorn имеет встроенный механизм автоматической перезагрузки, который обеспечивает восстановление работы в случае сбоев. Он также поддерживает механизмы управления процессами и мониторинга состояния работы сервера.
- Простота использования:** Gunicorn легко настраивается и интегрируется с приложениями Django. Его использование сводится к простым командам командной строки, что упрощает развертывание приложений.

Gunicorn

Обычно Gunicorn используется в сочетании с веб-сервером, таким как Nginx или Apache, который выполняет функции прокси-сервера и обрабатывает статические файлы, а Gunicorn обрабатывает динамические запросы и передает их в Django приложение.

В целом, Gunicorn является надежным и эффективным выбором для развертывания веб-приложений Django в производственной среде, обеспечивая высокую производительность и масштабируемость.

Nginx

Nginx (произносится "энджин-экс") - это высокопроизводительный веб-сервер, прокси-сервер и сервер обратного прокси. Он был создан с целью обеспечить эффективную и масштабируемую доставку веб-содержимого.

Некоторые ключевые особенности Nginx:

1. Веб-сервер: Nginx может служить веб-сервером, обрабатывая HTTP-запросы и доставляя статические файлы (HTML, CSS, JavaScript, изображения и т. д.) напрямую клиенту. Он имеет эффективную архитектуру, позволяющую обслуживать тысячи одновременных подключений с низким потреблением ресурсов.
2. Прокси-сервер: Nginx может выступать в роли прокси-сервера, принимая запросы от клиента и перенаправляя их на другой сервер. Это позволяет разделить нагрузку между несколькими серверами или балансировать нагрузку для обеспечения высокой доступности и производительности.

Nginx

1. Сервер обратного прокси: Nginx может работать в качестве сервера обратного прокси, принимая запросы от клиента и передавая их на задние (backend) сервера, где находятся динамические компоненты веб-приложения. Это позволяет разделить статические и динамические контенты, улучшая производительность и обеспечивая более гибкую конфигурацию.
2. Балансировка нагрузки: Nginx обеспечивает возможность распределения нагрузки между несколькими серверами. Он может использовать различные алгоритмы балансировки, такие как round-robin, least connections и IP hash, чтобы эффективно распределять запросы между серверами и обеспечивать отказоустойчивость.
3. Поддержка протоколов: Nginx поддерживает не только протокол HTTP, но и другие протоколы, такие как HTTPS (защищенная передача данных по протоколу HTTP с использованием шифрования SSL/TLS), WebSocket, HTTP/2 и многие другие.

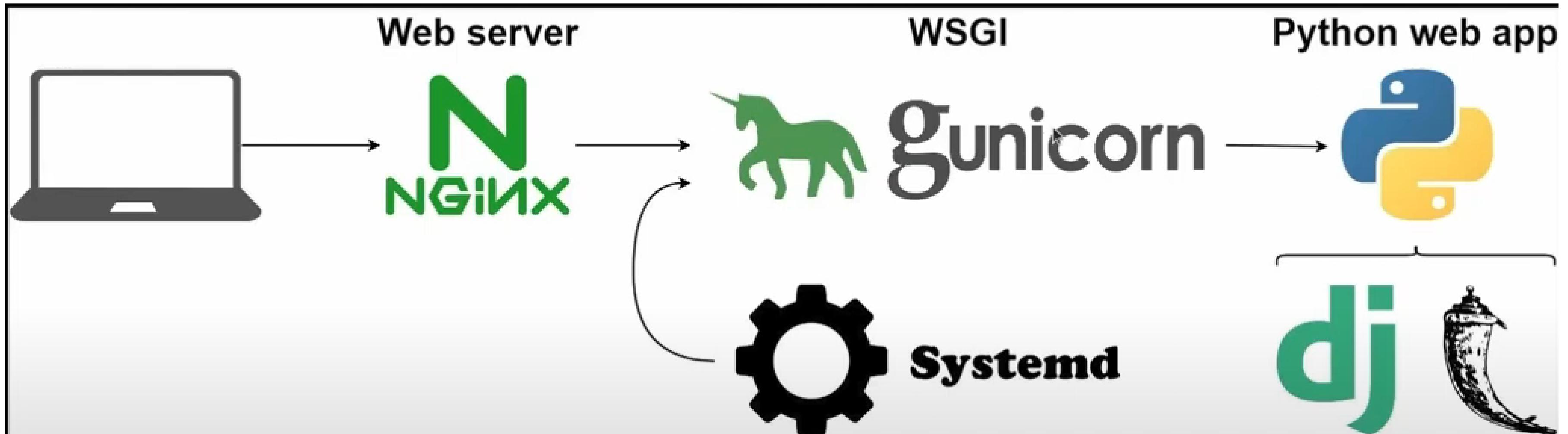
Supervisor

Supervisor (называемый также `Supervisord`) - это система управления процессами для UNIX-подобных операционных систем. Он обеспечивает надежный способ запуска, остановки и перезапуска процессов на сервере. Supervisor часто используется для управления и обеспечения непрерывной работы демонов, включая веб-серверы, фоновые задачи, очереди сообщений и другие компоненты приложений.

systemctl - это командный интерфейс управления службами (`services`) в системах, использующих `systemd`. `Systemd` является системой инициализации и управления процессами в современных дистрибутивах Linux, таких как `Ubuntu`, `CentOS`, `Fedora` и других.

`systemctl start redis-server`

Схема работы приложения



Пару слов в пользу linux

- 1. Поддержка инструментов и фреймворков:** Linux обычно имеет широкую поддержку для инструментов и фреймворков Python. Многие популярные веб-фреймворки, такие как Django, Flask и Pyramid, разрабатываются и тестируются с учетом Linux-среды. Это облегчает установку, настройку и развертывание веб-приложений на Linux.
- 2. Удобство использования:** Linux является свободно распространяемой операционной системой с открытым исходным кодом, что делает его привлекательным для разработчиков и системных администраторов. Он предлагает широкий набор инструментов и пакетов, необходимых для разработки и развертывания приложений на Python.
- 3. Совместимость с серверными инструментами:** Многие серверные инструменты и технологии, такие как Nginx, Gunicorn, Supervisor и базы данных, имеют хорошую поддержку на Linux. Они обеспечивают высокую производительность, масштабируемость и надежность, что важно для веб-приложений в продакшн-среде.

Пару слов не в пользу windows

- 1. Различия в окружении:** Windows и Linux имеют разные окружения выполнения и набор инструментов. Некоторые инструменты и библиотеки могут быть специфичны для определенной платформы, и это может потребовать дополнительных настроек или привести к различиям в поведении приложения между операционными системами.
- 2. Развёртывание:** Если вы разрабатываете веб-приложение для развертывания в прод среде, которая работает на Linux-серверах, то разработка на Windows может усложнить процесс развертывания и поддержки приложения. В таких случаях рекомендуется использовать операционную систему, близкую к целевой среде развертывания.
- 3. Совместимость и тестирование:** Возможны некоторые несовместимости и различия в поведении между Windows и Linux, особенно при работе с файловыми путями, символами новой строки и другими операционно-зависимыми функциями. Поэтому важно тестировать приложение на целевой платформе, чтобы обнаружить и исправить возможные проблемы.

4

Домашнее задание

Домашнее задание

- Добавить в проект swagger документацию
- Обновить приложение в соответствии с best practice с урока

Спасибо за внимание!