

< Teach  
Me  
Skills />

# Занятие 33.

The background is a vibrant yellow field filled with a repeating pattern of various geometric shapes. These include circles, squares, triangles, and lines in different shades of yellow and orange. Some shapes are solid, while others are outlined or have a dotted texture. The overall effect is a modern, energetic, and abstract design.

1

**FastAPI**

# FastAPI

**FastAPI** - это быстрый веб-фреймворк для создания API с поддержкой Python 3.7+ на основе стандарта типов Python (PEP 484). Он позволяет разработчикам создавать эффективные веб-приложения и API с минимальным объемом кода благодаря использованию современных асинхронных технологий и автоматической генерации документации.

# Преимущества

- **Быстродействие:** FastAPI разработан с акцентом на производительность и эффективность. Благодаря асинхронной поддержке и использованию Python типов данных для автоматической валидации запросов и ответов, FastAPI достигает высокой скорости обработки запросов.
- **Простота использования:** FastAPI построен с акцентом на простоту использования и интуитивное API. Это делает процесс разработки API более приятным и быстрым для разработчиков.
- **Автоматическая документация:** FastAPI автоматически генерирует интерактивную документацию для вашего API, основанную на стандарте OpenAPI (ранее известном как Swagger). Это значительно упрощает отладку, тестирование и работу с API, поскольку разработчики могут легко просматривать доступные эндпоинты, параметры запросов и ожидаемые ответы.

# Преимущества

- **Асинхронная поддержка:** FastAPI полностью поддерживает асинхронное программирование, что позволяет эффективно обрабатывать большое количество одновременных запросов без блокировки потока выполнения.
- **Поддержка WebSocket:** FastAPI предоставляет встроенную поддержку WebSocket, что делает его отличным выбором для приложений, которые требуют двунаправленного взаимодействия между клиентами и сервером.



# Зачем нужен FastAPI

FastAPI предоставляет разработчикам удобный и эффективный инструментарий для создания веб-приложений и API на Python. Он особенно полезен, когда требуется быстрый и надежный фреймворк для создания API с асинхронной поддержкой и автоматической документацией.

В связи с его быстродействием и простотой использования, FastAPI подходит для различных проектов и приложений, где важны высокая производительность, быстрая разработка и отзывчивый интерфейс для разработчиков и пользователей API. Он позволяет сосредоточиться на бизнес-логике и функциональности приложения, не заботясь о многих аспектах, связанных с низкоуровневой реализацией веб-сервера.



2

**Pydantic**

# Pydantic

**Pydantic** - это библиотека Python для валидации данных и сериализации объектов на основе аннотаций типов Python (PEP 484). Он предоставляет простой и удобный способ описывать модели данных с помощью стандартных типов Python и автоматически проверять их соответствие ожидаемым типам данных.

```
class User(BaseModel):  
    id: int  
    username: str  
    email: str  
    is_active: bool
```



# Pydantic - зачем он нужен

- **Валидация данных:** Pydantic позволяет задавать ожидаемые типы данных для каждого поля в модели данных. Когда объект создается на основе такой модели, Pydantic автоматически проверяет соответствие переданных данных ожидаемым типам. Это позволяет легко и безопасно валидировать входные данные, предотвращая ошибки и исключая несоответствия типов.
- **Сериализация данных:** Pydantic обеспечивает простой способ преобразования объектов Python в словари JSON и обратно (сериализация и десериализация). Это особенно полезно при работе с API, когда требуется преобразование объектов данных в формат JSON для передачи по сети и обратно.

# Pydantic - зачем он нужен

- **Автоматическая документация:** Pydantic интегрируется хорошо с другими инструментами, такими как FastAPI, и позволяет автоматически генерировать схемы данных для API. Это делает документацию более точной и актуальной, так как она всегда соответствует фактической структуре данных в вашем приложении.
- **Упрощение работы с данными:** Pydantic предоставляет удобные методы для манипулирования данными внутри моделей, такие как валидация, приведение типов, группировка полей и дополнительные проверки. Это упрощает работу с данными и облегчает создание сложных структур данных.
- **Поддержка асинхронности:** Pydantic полностью совместим с асинхронными приложениями, что позволяет использовать его в среде, где активно используется асинхронное программирование (например, с FastAPI).

# Pydantic - зачем он нужен

В целом, Pydantic делает работу с данными в Python более удобной, безопасной и позволяет уменьшить объем кода, связанного с валидацией и сериализацией данных. Он становится особенно полезным в проектах, где важна точность и надежность обработки данных, например, в веб-приложениях, API, обработке данных и многих других сценариях разработки.

3

**Асинхронность**

# Асинхронность

**Асинхронность (асинхронное программирование)** - это стиль программирования, который позволяет выполнять несколько задач одновременно, без ожидания завершения каждой задачи перед началом выполнения следующей. Это особенно полезно, когда выполнение задач может занимать много времени, и блокировка ожидания результата может замедлить работу программы.

Традиционный (синхронный) подход к программированию выполняет задачи последовательно. Когда задача выполняется, программа останавливается и ждет ее завершения, прежде чем переходить к следующей задаче. Это может быть неэффективно, особенно если задачи требуют много времени, например, запросы к внешнему серверу или операции ввода-вывода (I/O).



# Асинхронность

В асинхронном программировании, с помощью механизмов асинхронных операций (например, асинхронные функции, генераторы, колбэки), задачи выполняются параллельно или переключаются во время операций ввода-вывода. Таким образом, одна задача может быть приостановлена, чтобы другая задача могла выполняться, пока первая ждет завершения некоторой операции.

```
async def async_task(name, seconds):  
    print(f"{name} начал выполнение.")  
    await sleep(seconds)  
    print(f"{name} закончил выполнение.")
```

4

# Dependency Injection

# Dependency Injection

**Dependency Injection (DI)** - это паттерн проектирования программного обеспечения, который способствует разделению ответственности в коде и делает его более модульным, тестируемым и поддерживаемым. FastAPI предоставляет встроенную поддержку Dependency Injection.

В FastAPI можно использовать функцию `Depends`, чтобы внедрять зависимости в ваши endpoint-ы API. Зависимости могут быть любыми объектами или сервисами Python, которые требуются вашим функциям конечных точек для выполнения своих задач. Общие случаи использования зависимостей включают подключения к базе данных, аутентификацию, настройки и внешние сервисы.

4

# Домашнее задание

# Домашнее задание

## Написать API:

Нужно написать RESTful приложение на fastapi для реализации логики из предыдущего домашнего задания:

- эндпоинт для отображения всех пользователей
- эндпоинт для отображения идей пользователя
- эндпоинт для создания идеи

Миграции в этом приложении создавать не будем, будем подключаться к уже существующей БД, для того чтобы получить строку подключения можно использовать `load_dotenv`, также и для других переменных, которые могут быть вынесены из приложения



The background is a vibrant yellow color, covered with a dense, repeating pattern of various geometric shapes. These shapes include circles, squares, triangles, and lines, some of which are filled with a fine grid pattern. The shapes are scattered across the entire surface, creating a dynamic and modern aesthetic.

**Спасибо за внимание!**