







Знакомство с FastAPI

Семинар 5





Что будет сегодня на семинаре. Наши цели:

-  Узнать про FastAPI и его возможности
-  Разобраться в настройке среды разработки
-  Изучить создание базового приложения FastAPI
-  Узнать об обработке HTTP-запросов и ответов
-  Разобраться в создании конечных точек API
-  Изучить автоматическую документацию по API



Вопросы?

Вопросы?












Вопросы?



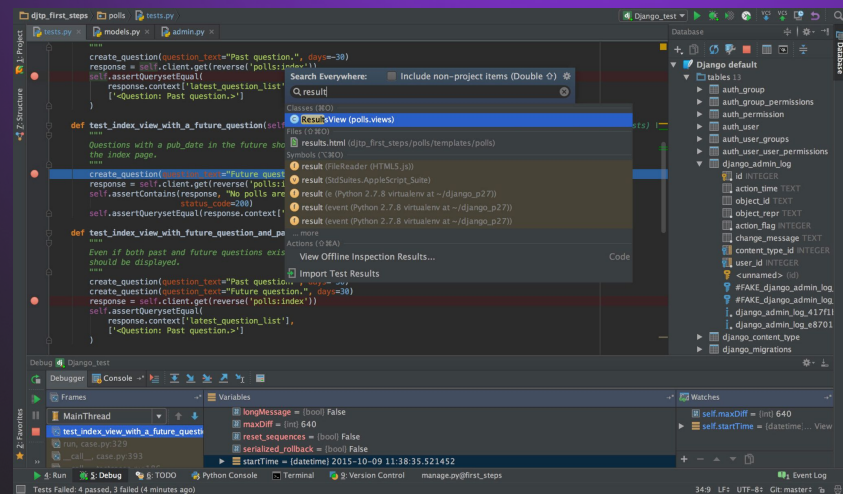


Задание №1

-  Создать API для управления списком задач. Приложение должно иметь возможность создавать, обновлять, удалять и получать список задач.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс Task с полями id, title, description и status.
-  Создайте список tasks для хранения задач.
-  Создайте маршрут для получения списка задач (метод GET).
-  Создайте маршрут для создания новой задачи (метод POST).
-  Создайте маршрут для обновления задачи (метод PUT).
-  Создайте маршрут для удаления задачи (метод DELETE).
-  Реализуйте валидацию данных запроса и ответа.









Решение в IDE



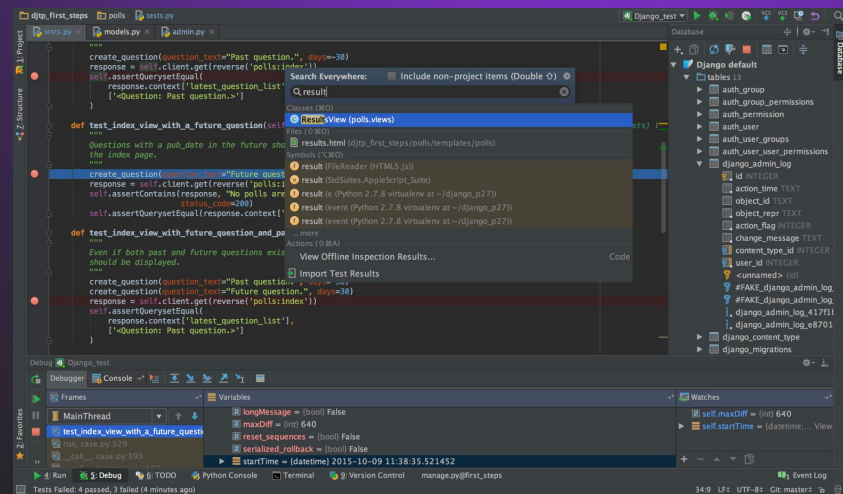


Задание №2

-  Создать API для получения списка фильмов по жанру. Приложение должно иметь возможность получать список фильмов по заданному жанру.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс Movie с полями id, title, description и genre.
-  Создайте список movies для хранения фильмов.
-  Создайте маршрут для получения списка фильмов по жанру (метод GET).
-  Реализуйте валидацию данных запроса и ответа.









Решение в IDE



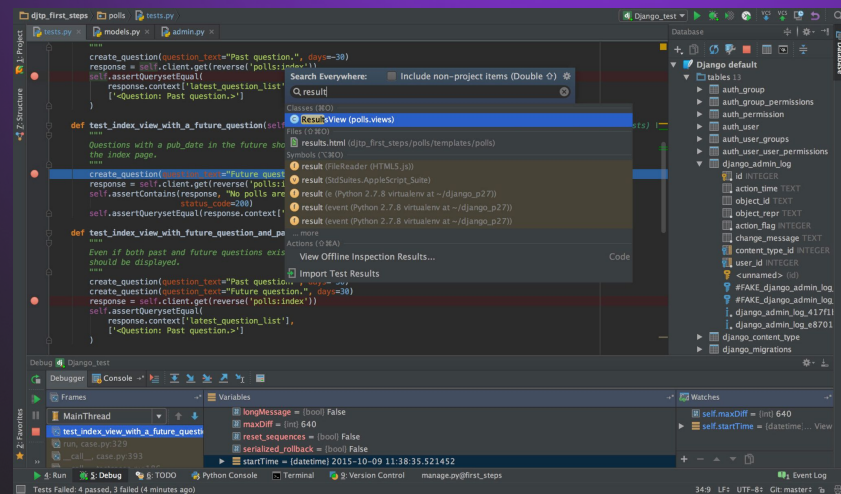


Задание №3

-  Создать API для добавления нового пользователя в базу данных. Приложение должно иметь возможность принимать POST запросы с данными нового пользователя и сохранять их в базу данных.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс User с полями id, name, email и password.
-  Создайте список users для хранения пользователей.
-  Создайте маршрут для добавления нового пользователя (метод POST).
-  Реализуйте валидацию данных запроса и ответа.









Решение в IDE



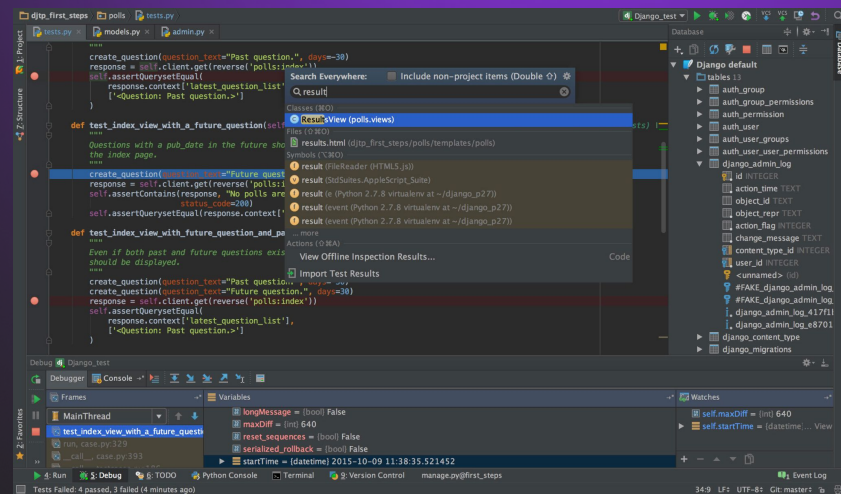


Задание №4

-  Создать API для обновления информации о пользователе в базе данных. Приложение должно иметь возможность принимать PUT запросы с данными пользователей и обновлять их в базе данных.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс User с полями id, name, email и password.
-  Создайте список users для хранения пользователей.
-  Создайте маршрут для обновления информации о пользователе (метод PUT).
-  Реализуйте валидацию данных запроса и ответа.









Решение в IDE



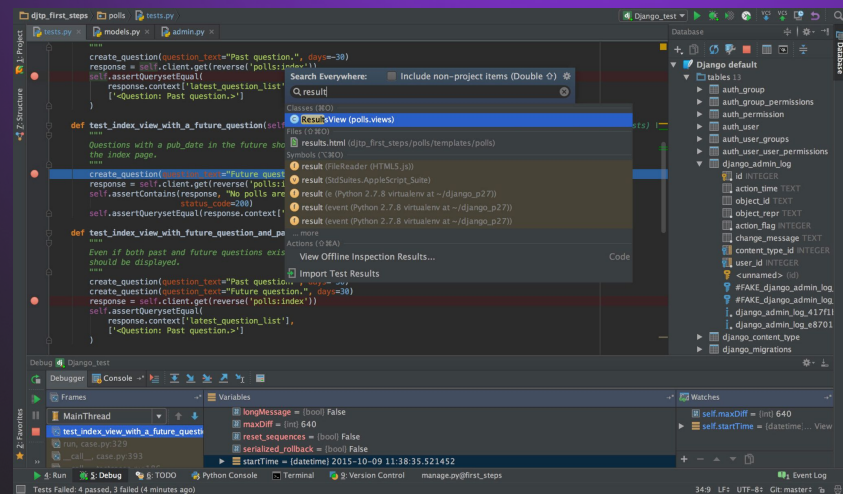


Задание №5

-  Создать API для удаления информации о пользователе из базы данных. Приложение должно иметь возможность принимать DELETE запросы и удалять информацию о пользователе из базы данных.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс User с полями id, name, email и password.
-  Создайте список users для хранения пользователей.
-  Создайте маршрут для удаления информации о пользователе (метод DELETE).
-  Реализуйте проверку наличия пользователя в списке и удаление его из списка.



Решение в IDE





Перерыв?

Голосуйте в чате










Перерыв...

<<7:00->>

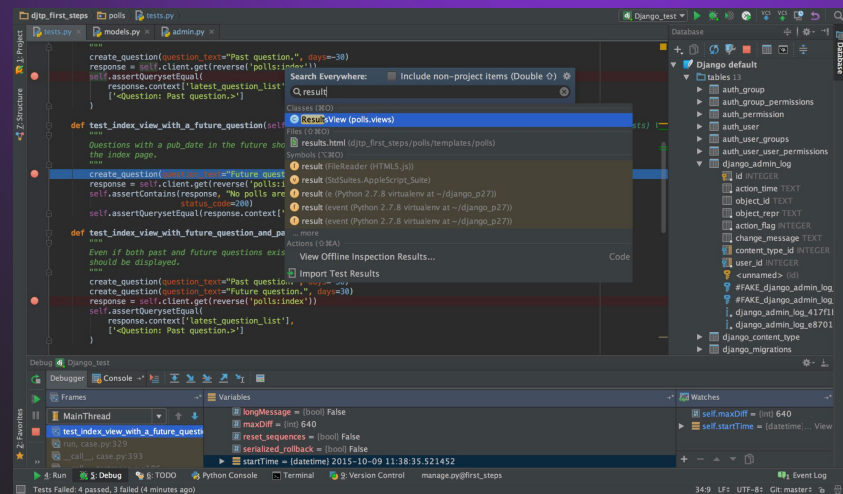


Задание №6

-  Создать веб-страницу для отображения списка пользователей. Приложение должно использовать шаблонизатор Jinja для динамического формирования HTML страницы.
-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс User с полями id, name, email и password.
-  Создайте список users для хранения пользователей.
-  Создайте HTML шаблон для отображения списка пользователей. Шаблон должен содержать заголовок страницы, таблицу со списком пользователей и кнопку для добавления нового пользователя.
-  Создайте маршрут для отображения списка пользователей (метод GET).
-  Реализуйте вывод списка пользователей через шаблонизатор Jinja.




Решение в IDE






Задание №7









 Создать RESTful API для управления списком задач. Приложение должно использовать FastAPI и поддерживать следующие функции:

- Получение списка всех задач.
- Получение информации о задаче по её ID.
- Добавление новой задачи.
- Обновление информации о задаче по её ID.
- Удаление задачи по её ID.

 Каждая задача должна содержать следующие поля: ID (целое число), Название (строка), Описание (строка), Статус (строка): "todo", "in progress", "done".

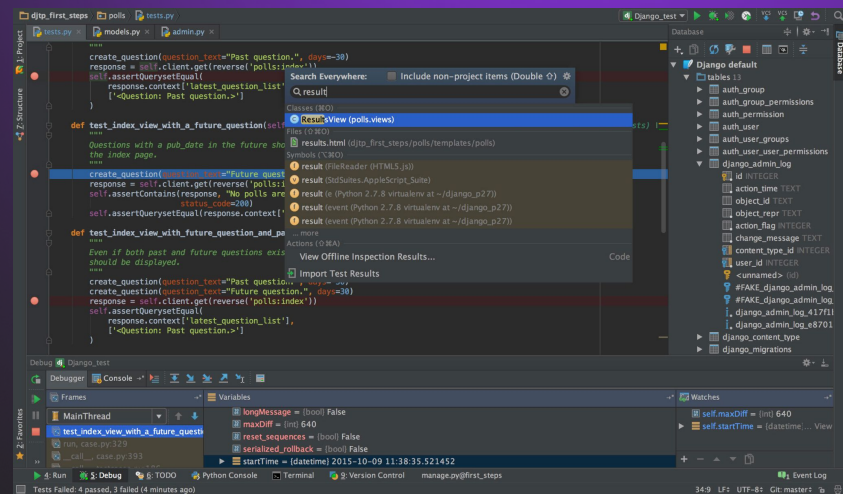


Задание №7 (продолжение)

-  Создайте модуль приложения и настройте сервер и маршрутизацию.
-  Создайте класс Task с полями id, title, description и status.
-  Создайте список tasks для хранения задач.
-  Создайте функцию get_tasks для получения списка всех задач (метод GET).
-  Создайте функцию get_task для получения информации о задаче по её ID (метод GET).
-  Создайте функцию create_task для добавления новой задачи (метод POST).
-  Создайте функцию update_task для обновления информации о задаче по её ID (метод PUT).
-  Создайте функцию delete_task для удаления задачи по её ID (метод DELETE).



Решение в IDE





Задание №8



Необходимо создать API для управления списком задач. Каждая задача должна содержать заголовок и описание. Для каждой задачи должна быть возможность указать статус (выполнена/не выполнена).



API должен содержать следующие конечные точки:

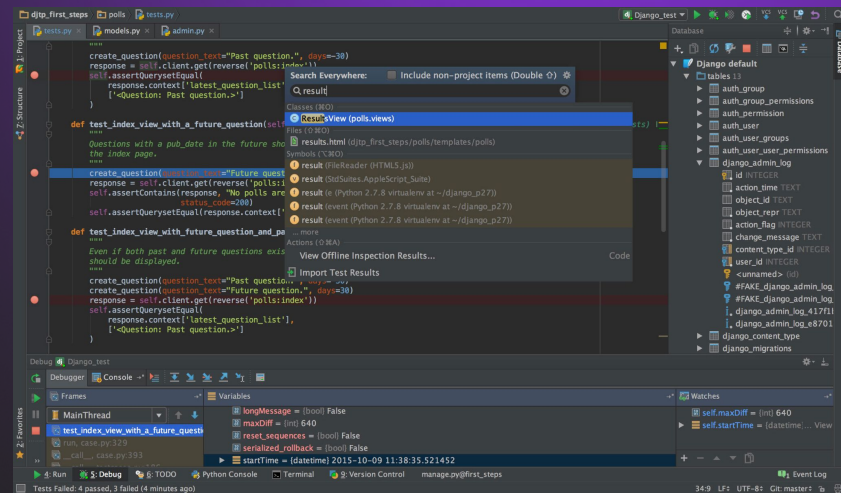
- GET /tasks - возвращает список всех задач.
- GET /tasks/{id} - возвращает задачу с указанным идентификатором.
- POST /tasks - добавляет новую задачу.
- PUT /tasks/{id} - обновляет задачу с указанным идентификатором.
- DELETE /tasks/{id} - удаляет задачу с указанным идентификатором.



Для каждой конечной точки необходимо проводить валидацию данных запроса и ответа. Для этого использовать библиотеку Pydantic.



Решение в IDE





Вопросы?

Вопросы?



Вопросы?





Домашнее задание



Задание



Решить задачи, которые не успели решить на семинаре.



Подведем итоги



Что было
сложного на
семинаре?





Напишите три вещи в
комментариях, которым
вы научились сегодня.





Как настроение?





Спасибо за работу!