

Дополнительные возможности FastAPI

Семинар 6





Что будет сегодня на семинаре. Наши цели:



Узнать про модели данных в FastAPI и дополнительные возможности валидации



Разобраться в работе с базой данных



Вопросы?

Вопросы?



Вопросы?





Задание №1






Разработать API для управления списком пользователей с использованием базы данных SQLite. Для этого создайте модель User со следующими полями:

- id: int (идентификатор пользователя, генерируется автоматически)
- username: str (имя пользователя)
- email: str (электронная почта пользователя)
- password: str (пароль пользователя)

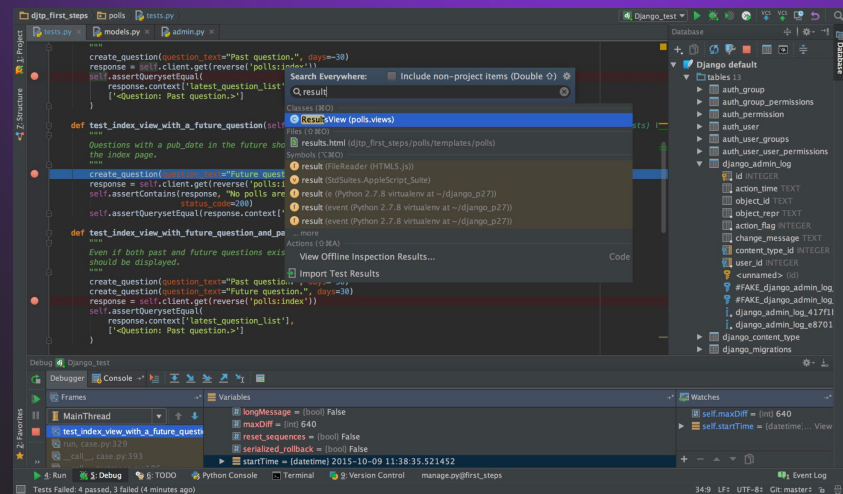


Задание №1 (продолжение)

-  API должно поддерживать следующие операции:
 - Получение списка всех пользователей: GET /users/
 - Получение информации о конкретном пользователе: GET /users/{user_id}/
 - Создание нового пользователя: POST /users/
 - Обновление информации о пользователе: PUT /users/{user_id}/
 - Удаление пользователя: DELETE /users/{user_id}/
-  Для валидации данных используйте параметры Field модели User.
-  Для работы с базой данных используйте SQLAlchemy и модуль databases.



Решение в IDE





Задание №2





Создать веб-приложение на FastAPI, которое будет предоставлять API для работы с базой данных пользователей. Пользователь должен иметь следующие поля:

- ID (автоматически генерируется при создании пользователя)
- Имя (строка, не менее 2 символов)
- Фамилия (строка, не менее 2 символов)
- Дата рождения (строка в формате "YYYY-MM-DD")
- Email (строка, валидный email)
- Адрес (строка, не менее 5 символов)

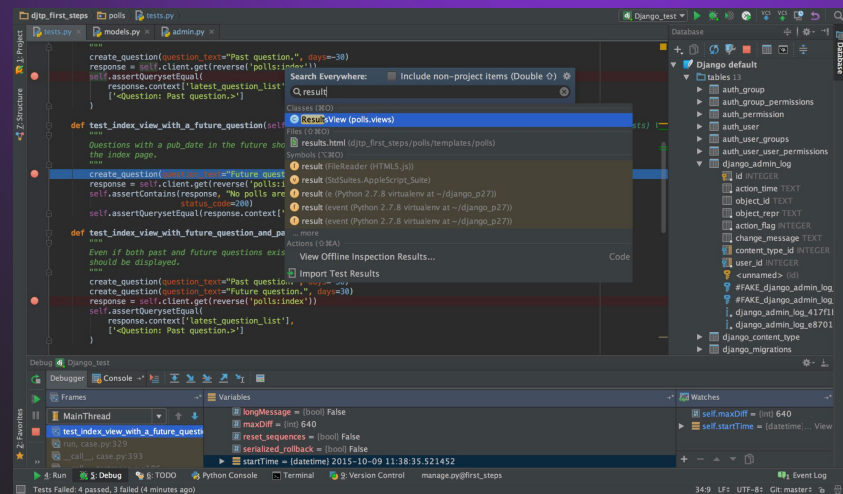


Задание №2 (продолжение)

-  API должен поддерживать следующие операции:
 - Добавление пользователя в базу данных
 - Получение списка всех пользователей в базе данных
 - Получение пользователя по ID
 - Обновление пользователя по ID
 - Удаление пользователя по ID
-  Приложение должно использовать базу данных SQLite3 для хранения пользователей.






Решение в IDE



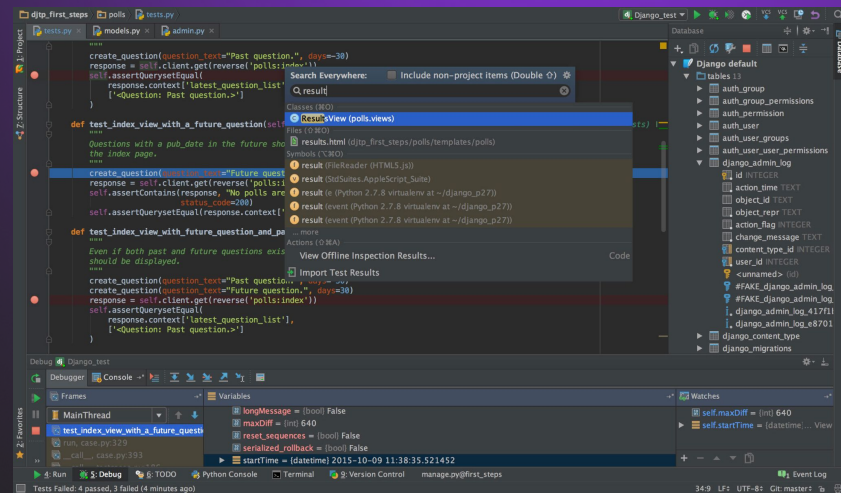


Задание №3

-  Создать API для управления списком задач.
-  Каждая задача должна содержать поля "название", "описание" и "статус" (выполнена/не выполнена).
-  API должен позволять выполнять CRUD операции с задачами.



Решение в IDE





Задание №4






Напишите API для управления списком задач. Для этого создайте модель Task со следующими полями:

- id: int (первичный ключ)
- title: str (название задачи)
- description: str (описание задачи)
- done: bool (статус выполнения задачи)

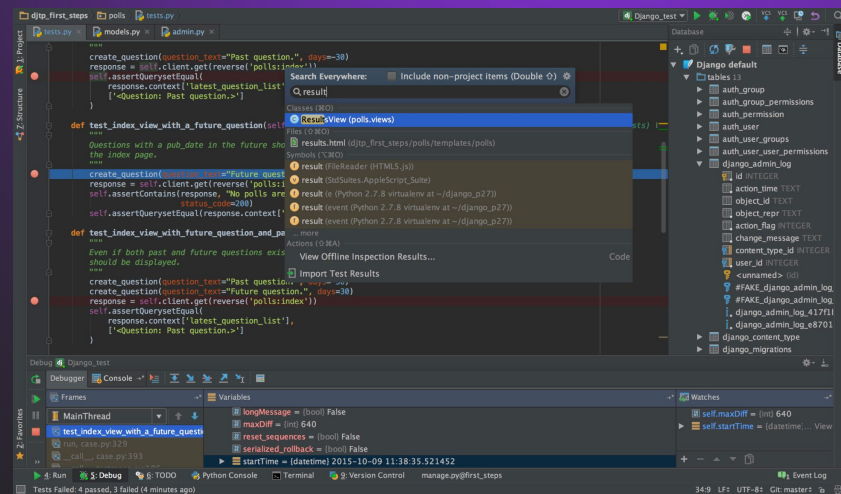


Задание №4 (продолжение)

-  API должно поддерживать следующие операции:
 - Получение списка всех задач: GET /tasks/
 - Получение информации о конкретной задаче: GET /tasks/{task_id}/
 - Создание новой задачи: POST /tasks/
 - Обновление информации о задаче: PUT /tasks/{task_id}/
 - Удаление задачи: DELETE /tasks/{task_id}/
-  Для валидации данных используйте параметры Field модели Task.
-  Для работы с базой данных используйте SQLAlchemy и модуль databases.



Решение в IDE





Перерыв?

Голосуйте в чате



Перерыв...

<<7:00->>



Задание №5*






Разработать API для управления списком задач с использованием базы данных MongoDB. Для этого создайте модель Task со следующими полями:

- id: str (идентификатор задачи, генерируется автоматически)
- title: str (название задачи)
- description: str (описание задачи)
- done: bool (статус выполнения задачи)

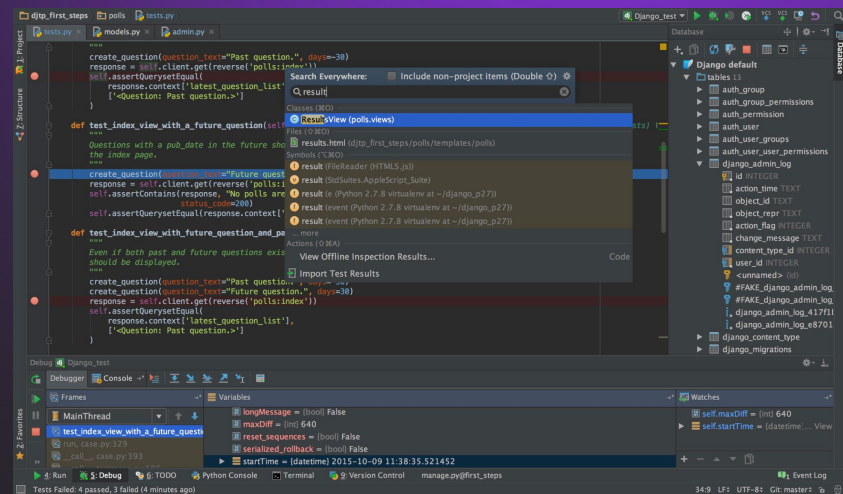


Задание №5* (продолжение)

-  API должно поддерживать следующие операции:
 - Получение списка всех задач: GET /tasks/
 - Получение информации о конкретной задаче: GET /tasks/{task_id}/
 - Создание новой задачи: POST /tasks/
 - Обновление информации о задаче: PUT /tasks/{task_id}/
 - Удаление задачи: DELETE /tasks/{task_id}/
-  Для валидации данных используйте параметры Field модели Task.
-  Для работы с базой данных используйте PyMongo.



Решение в IDE





Задание №6



Необходимо создать базу данных для интернет-магазина. База данных должна состоять из трех таблиц: товары, заказы и пользователи. Таблица товары должна содержать информацию о доступных товарах, их описаниях и ценах. Таблица пользователи должна содержать информацию о зарегистрированных пользователях магазина. Таблица заказы должна содержать информацию о заказах, сделанных пользователями.

- Таблица пользователей должна содержать следующие поля: id (PRIMARY KEY), имя, фамилия, адрес электронной почты и пароль.
- Таблица товаров должна содержать следующие поля: id (PRIMARY KEY), название, описание и цена.
- Таблица заказов должна содержать следующие поля: id (PRIMARY KEY), id пользователя (FOREIGN KEY), id товара (FOREIGN KEY), дата заказа и статус заказа.



Задание №6 (продолжение)



Создайте модели pydantic для получения новых данных и возврата существующих в БД для каждой из трёх таблиц (итого шесть моделей).

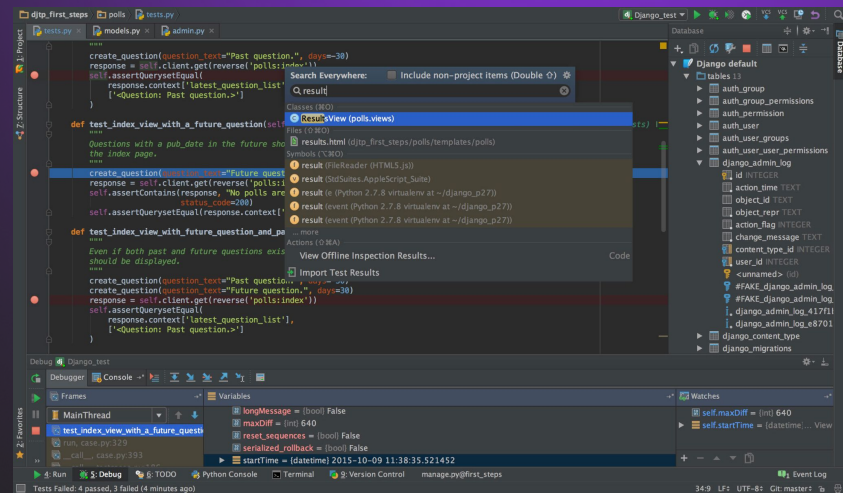


Реализуйте CRUD операции для каждой из таблиц через создание маршрутов, REST API (итого 15 маршрутов).

- Чтение всех
- Чтение одного
- Запись
- Изменение
- Удаление



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!