Кейс 1: Создание простого REST API с использованием Flask

Задание:

1. Создать простой REST API с использованием Flask.
2. Реализовать эндпоинты для получения, создания, обновления и удаления ресурса (например, задачи в TODO списке).

Ожидаемый результат:

- REST API создан с использованием Flask.
- Эндпоинты для CRUD операций работают корректно.

Кейс 2: Тестирование API с использованием Postman

Задание:

1. Создать коллекцию в Postman для тестирования созданного API.
2. Написать запросы для каждого эндпоинта и проверить их работоспособность.

Ожидаемый результат:

- Коллекция в Postman создана.
- Все эндпоинты протестированы и работают корректно.

Кейс 3: Обработка запросов и ответов

Задание:

1. Реализовать обработку запросов с параметрами (query parameters) и передача данных в теле запроса (request body).
2. Возвращать ответы в формате JSON с соответствующими HTTP статусами.

Ожидаемый результат:

- Запросы с параметрами и передача данных в теле запроса обрабатываются корректно.
- Ответы возвращаются в формате JSON с правильными HTTP статусами.

Кейс 4: Аутентификация и авторизация

Задание:

1. Добавить аутентификацию к API с использованием токенов (например, JWT).
2. Реализовать защиту эндпоинтов с проверкой токенов.

Ожидаемый результат:

- Аутентификация с использованием JWT реализована.

- Эндпоинты защищены и доступны только для аутентифицированных пользователей.

Кейс 5: Работа с внешними API

Задание:

1. Написать клиент для работы с внешним API (например, OpenWeatherMap для получения данных о погоде).
2. Реализовать запросы к внешнему API и обработку ответов.

Ожидаемый результат:

- Клиент для работы с внешним API написан.
- Запросы к внешнему API выполняются и ответы обрабатываются корректно.

Кейс 6: Документация API с использованием Swagger

Задание:

1. Добавить документацию к созданному API с использованием Swagger (Flask-RESTPlus или FastAPI).
2. Убедиться, что документация корректно описывает все эндпоинты и параметры.

Ожидаемый результат:

- Документация API добавлена и доступна через интерфейс Swagger.
- Документация корректно описывает все эндпоинты и параметры.

Кейс 7: Версионирование API

Задание:

1. Реализовать версионирование API, чтобы поддерживать несколько версий одновременно.
2. Добавить новую версию API с изменениями в структуре данных или эндпоинтах.

Ожидаемый результат:

- Версионирование API реализовано.
- Новая версия API добавлена и работает параллельно с предыдущей версией.

Кейс 8: Обработка ошибок и логирование

Задание:

1. Реализовать обработку ошибок в API и возвращать понятные сообщения об ошибках.
2. Добавить логирование запросов и ошибок.

Ожидаемый результат:

- Обработка ошибок реализована.
- Логирование запросов и ошибок работает корректно.

Кейс 9: Тестирование API с использованием pytest

Задание:

1. Написать тесты для API с использованием pytest и pytest-flask.
2. Покрыть тестами все эндпоинты и основные сценарии использования API.

Ожидаемый результат:

- Тесты для API написаны и выполняются успешно.
- Все эндпоинты покрыты тестами.

Кейс 10: Развертывание API в облаке

Задание:

1. Настроить развертывание API в облаке (например, Heroku или AWS).
2. Убедиться, что API доступен и работает корректно в облаке.

Ожидаемый результат:

- API развернут в облаке.
- API доступен и работает корректно.