ОТЧЕТ

О практической работе по дисциплине “Тестирование программного обеспечения для разработчиков”

Лабораторная работа №2
Тема: Интеграционное тестирование.

Студент: Сорокин Андрей Александрович, К3321
tg: @androndef

Санкт-Петербург,
2025

Цель работы:

Закрепить навыки интеграционного тестирования: научиться выявлять места взаимодействия между компонентами системы, писать тесты для проверки таких взаимодействий и анализировать результаты.

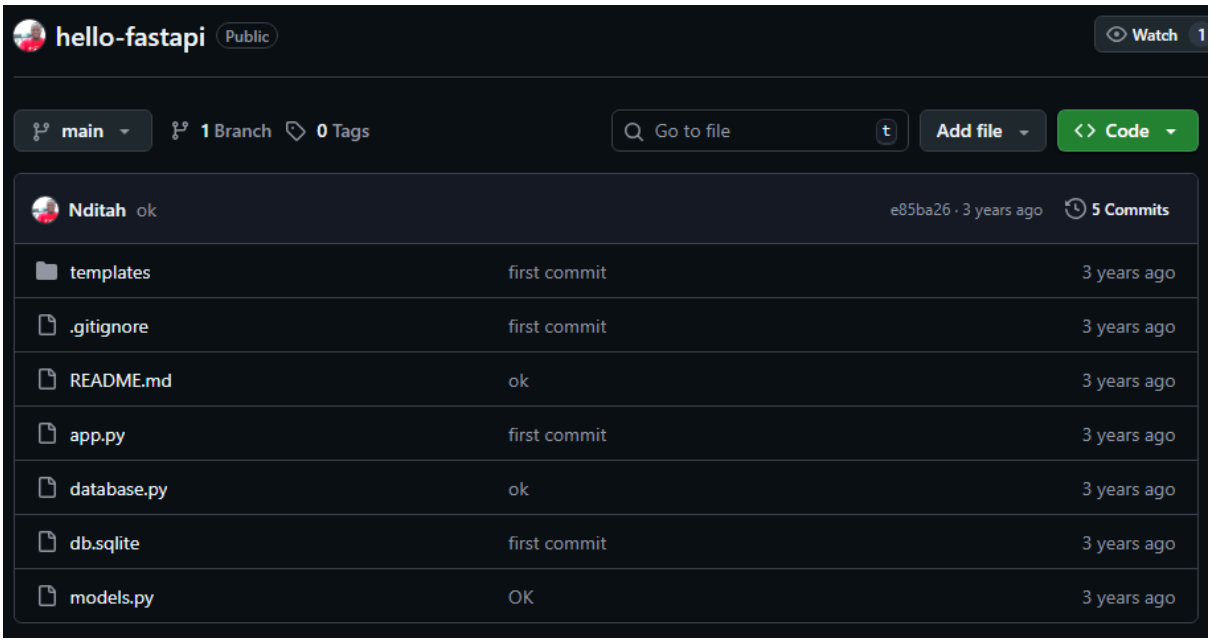
Задачи:

- Выбрать открытый проект на GitHub (можно свой пет-проект).
- Определите ключевые точки интеграции в проекте.
- Критические участки.
- Ключевые сценарии взаимодействия (работа с БД, запрос к API).
- Создать не менее 5 интеграционный-тестов
- Использовать тестовое окружение
- Подготовить отчёт

Ссылка на репозиторий - https://github.com/AndronIxI/hello-fastapi-fork

Ход работы:

1. Поиск проекта
Как и в предыдущей лабе, я решил использовать публичный проект с гита. На этот раз жертвой стал проект hello-fastapi



Это простенькое веб-приложение на питоне с использованием FastAPI, SQLAlchemy и SQLite. В нём есть несколько модулей, которые явно взаимодействуют друг с другом:

- database.py отвечает за создание подключения к базе данных и настройку SQLAlchemy.
- models.py содержит ORM-модель Todo, соответствующую таблице todos в SQLite.
- app.py поднимает FastAPI-приложение, подключает шаблоны, определяет зависимость get_db и реализует маршруты для CRUD-операций с задачами.
- каталог templates/ содержит HTML-шаблон base.html, в который передаются данные из базы.

Благодаря такой структуре проект хорошо подходит для интеграционного тестирования

2. Тесты

Первый тест проверяет, нормально ли отображается список задач на главной странице.

```python
def test_home_returns_existing_todos():
    create_todo(title="First task")
    response = client.get("/")
    assert response.status_code == 200
    assert "First task" in response.text
```

Сначала он создаёт тестовую задачу в базе данных, а потом отправляет запрос на / и смотрит, есть ли эта задача в HTML-ответе. Если всё в порядке, значит приложение правильно берёт данные из БД и подставляет их в шаблон. Если нет — пользователь просто не увидит свои задачи, и смысл интерфейса сильно теряется.

Второй тест проверяет, корректно ли работает сценарий добавления новой задачи через форму.

```python
def test_add_todo_creates_item_and_redirects():
    response = client.post("/add", data={"title": "New task"}, follow_redirects=False)
    assert response.status_code == 303
    db = TestingSessionLocal()
    todos = db.query(models.Todo).all()
    db.close()
    assert len(todos) == 1
    assert todos[0].title == "New task"
    assert todos[0].complete is False
```

Он отправляет POST-запрос на /add с заголовком задачи и ожидает, что сервер вернёт редирект, а в тестовой базе появится новая запись с нужными полями. Фактически это проверка основной функции приложения: если тест падает, пользователи не смогут добавлять задачи, а приложение перестанет выполнять свою главную роль.

Третий тест проверяет, правильно ли работает изменение статуса задачи.

```python
def test_update_todo_toggles_complete_status():
    todo = create_todo(title="Toggle task", complete=False)
    response = client.get(f"/update/{todo.id}", follow_redirects=False)
    assert response.status_code == 303
    db = TestingSessionLocal()
    updated = db.query(models.Todo).filter(models.Todo.id == todo.id).first()
    db.close()
    assert updated.complete is True
    response = client.get(f"/update/{todo.id}", follow_redirects=False)
    assert response.status_code == 303
    db = TestingSessionLocal()
    updated_again = db.query(models.Todo).filter(models.Todo.id == todo.id).first()
    db.close()
    assert updated_again.complete is False
```

Тест создаёт задачу с complete=False, затем дважды вызывает маршрут /update/{id} и каждый раз проверяет, что значение флага complete в базе инвертируется. Так мы убеждаемся, что переключение «выполнено/не выполнено» работает так, как ожидает пользователь. Если тут что-то сломано, управление статусами задач становится просто бесполезным.

Четвертый тест проверяет, корректно ли выполняется удаление задач.

```python
def test_delete_todo_removes_item():
    first = create_todo(title="First")
    second = create_todo(title="Second")
    response = client.get(f"/delete/{first.id}", follow_redirects=False)
    assert response.status_code == 303
    db = TestingSessionLocal()
    titles = [t.title for t in db.query(models.Todo).all()]
    db.close()
    assert "First" not in titles
    assert "Second" in titles
```

Создаются две задачи, затем через запрос /delete/{id} удаляется одна из них. После этого тест смотрит в базу и проверяет, что первая задача исчезла, а вторая осталась. Это важный сценарий: если удаление работает неправильно, список задач быстро превращается в свалку, которую невозможно нормально почистить.

Пятый тест проверяет, как приложение ведет себя при попытке добавить задачу без названия.

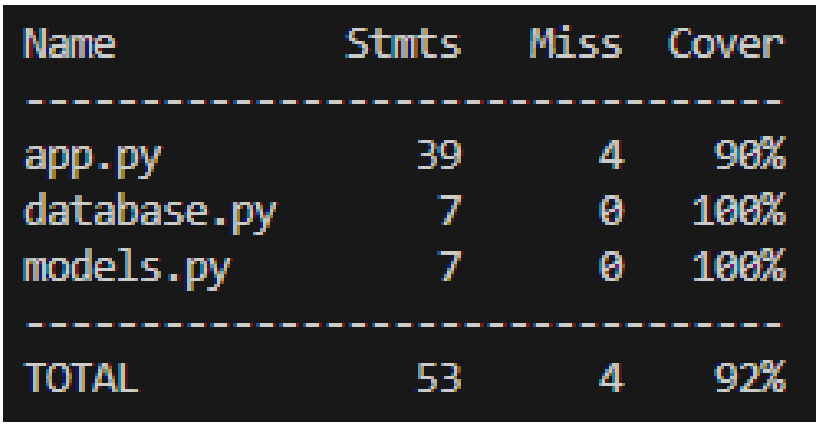
```python
def test_add_todo_without_title_returns_validation_error():
    response = client.post("/add", data={}, follow_redirects=False)
    assert response.status_code == 422
```

Он отправляет POST-запрос на /add без поля title и ожидает код ответа 422. Так мы убеждаемся, что встроенная валидация FastAPI не пропускает некорректные данные. Если этот тест не проходит, в базе могут появляться «пустые» задачи, что в будущем приведёт к странному поведению приложения и непонятному отображению списка для пользователя.

3. Результаты тестов
Запуск тестов через pytest в vs code.
По результатам теста выяснилось, что автор работы красавчик)
К сожалению (хотя скорее к счастью) все работает и нет ни одного заваленного теста.

```
PS C:\Users\tebil\Downloads\TestPO2\hello-fastapi-fork> pytest -v
============================================= test session starts =============================================
platform win32 -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\tebil\AppData\Local\Programs\Python\Python31
3\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\tebil\Downloads\TestPO2\hello-fastapi-fork
plugins: anyio-4.12.0, cov-7.0.0
collected 5 items

tests/test_integration.py::test_home_returns_existing_todos PASSED                                       [ 20%]
tests/test_integration.py::test_add_todo_creates_item_and_redirects PASSED                               [ 40%]
tests/test_integration.py::test_update_todo_toggles_complete_status PASSED                               [ 60%]
tests/test_integration.py::test_delete_todo_removes_item PASSED                                          [ 80%]
tests/test_integration.py::test_add_todo_without_title_returns_validation_error PASSED                   [100%]

============================================== warnings summary ===============================================
database.py:11
  C:\Users\tebil\Downloads\TestPO2\hello-fastapi-fork\database.py:11: MovedIn20Warning: The ``declarative_base()`` f
unction is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Background on SQLAlchemy 2.0
 at: https://sqlalche.me/e/b8d9)
    Base = declarative_base()

tests/test_integration.py::test_home_returns_existing_todos
  C:\Users\tebil\AppData\Local\Programs\Python\Python313\Lib\site-packages\starlette\templating.py:162: DeprecationW
arning: The `name` is not the first parameter anymore. The first parameter should be the `Request` instance.
  Replace `TemplateResponse(name, {"request": request})` by `TemplateResponse(request, name)`.
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================== 5 passed, 2 warnings in 1.00s ==========================================
```

Метрика code coverage



Name	Stmts	Miss	Cover
app.py	39	4	90%
database.py	7	0	100%
models.py	7	0	100%
TOTAL	53	4	92%

4. Выводы

Были проверены основные пользовательские сценарии приложения: открытие списка задач, добавление новой записи, изменение ее статуса, удаление и реакция на некорректный ввод. То есть мы прошлись по тем местам, без которых ToDo-приложение просто теряет смысл. Все интеграционные тесты отработали успешно, и это показывает, что веб-слой FastAPI, база данных SQLite и ORM-модель Todo между собой взаимодействуют правильно.

Coverage в 92% и 5 из 5 успешных тестов говорят о том, что почти весь важный код приложения был реально выполнен во время тестов, а непокрытыми остались всего 4 строки. Это значит, что вероятность скрытых критических багов в базовых сценариях работы достаточно низкая, но полностью исключать ошибки в непокрытых ветках всё равно нельзя. Вообще такое покрытие можно считать достаточно хорошим, а в будущем к нему уже удобно постепенно добавлять тесты для более специфических и граничных случаев.