Урок 9. Годовой бюджет, повторяющиеся операции и автоматизированное тестирование Work List Planner

1. Цель занятия

Продолжить развитие Work List Planner — веб-приложения для управления задачами и личными финансами — и реализовать функциональность планирования регулярных доходов и расходов на год.

Дополнительно проверить работоспособность API, обеспечить изоляцию тестовой базы данных и актуализировать документацию проекта.

В ходе занятия мы работали с Python, FastAPI, SQLite, HTML, JavaScript, pytest и Git.

2. Развитие модуля «Годовой бюджет»

В приложении развивается модуль для планирования доходов и расходов по месяцам выбранного года.

Пользователь может указывать сумму операции, дату, категорию и статус. Это позволяет планировать финансовые обязательства, учитывать регулярные платежи и видеть

структуру бюджета.

2.1. Повторяющиеся доходы и расходы

Добавлена возможность создавать ежемесячно повторяющиеся операции.

При включении соответствующей настройки приложение формирует отдельные операции на каждый месяц до конца выбранного года.

Например, если пользователь указывает регулярный платёж за аренду жилья, не нужно вручную создавать двенадцать одинаковых записей. Приложение формирует отдельную операцию для каждого месяца.

При этом записи остаются самостоятельными: сумму, дату или другие доступные параметры каждой операции можно корректировать независимо от остальных.

2.2. Обработка календарных дат

При создании повторяющихся операций важно учитывать, что календарные месяцы имеют разную продолжительность.

Реализована обработка дат с учётом длины месяца и високосных лет.

Примеры ожидаемого поведения:

- операция на 31-е число переносится на последний день более короткого месяца;
- для февраля учитывается фактическое количество дней;
- в високосном году учитывается 29 февраля;
- отдельные созданные операции можно изменять независимо друг от друга.

Это предотвращает ошибки при формировании годового плана и делает календарную логику предсказуемой.

2.3. Статусы операций

Для повторяющихся операций используется статус планирования `planned`, чтобы созданные на будущие месяцы записи не воспринимались как уже исполненные платежи.

Для разовой операции сохраняется выбранный пользователем статус.

Так разделяются планирование будущих расходов и работа с отдельными операциями.

2.4. Разделение данных пользователей

При работе с финансовыми данными особенно важно не допустить доступа одного пользователя к операциям другого.

В API учитывается принадлежность данных пользователю. Это необходимо для сохранения изоляции личных бюджетов.

3. Автоматизированное тестирование

Одной из важных задач занятия стало обеспечение безопасного запуска автоматических тестов.

3.1. Изоляция тестовой базы данных

Тесты не должны изменять реальные данные пользователей. Для этого используется отдельная временная база SQLite, настраиваемая через `conftest.py`.

Таким образом, автоматизированные проверки выполняются независимо от рабочей базы приложения.

Основное правило: **не запускать автоматические тесты на базе данных рабочего приложения**.

3.2. Запуск тестов

Для запуска тестов используется виртуальное окружение проекта:

```
.venv/bin/pytest -q
```

3.3. Результаты проверки

После внесения изменений был выполнен полный запуск тестов.

Результат:

- **40 тестов пройдены успешно**;
- ошибок, приводящих к провалу тестов, не обнаружено;
- зафиксировано 74 предупреждения, связанные с устаревающими API и функциями (deprecated warnings).

Предупреждения не означают, что тесты провалились. Однако их следует учитывать при дальнейшем сопровождении проекта и обновлении зависимостей.

Успешный запуск тестов подтверждает прохождение существующего набора проверок, но сам по себе не гарантирует отсутствие всех возможных ошибок в рабочем приложении.

4. Работа с Git и GitHub

После проверки изменений код был сохранён в Git и отправлен в удалённый репозиторий.

Репозиторий проекта:

https://github.com/im449/work-list-planner

Ветка разработки: `main`.

Основной коммит с реализацией повторяющихся операций:

`d801ece` — `Add monthly recurring annual budget items`

Изменения включали работу с интерфейсом годового бюджета, серверной логикой и API, а также тестами.

4.1. Обновление документации

После публикации функциональности обнаружилось, что `README.md` не описывает новые возможности и содержит устаревшую инструкцию по тестированию.

Документация была исправлена.

В README добавлено описание годового бюджета:

- планирование доходов и расходов по месяцам;
- указание суммы, даты, категории и статуса;
- ежемесячное повторение операций;
- обработка календарных дат;
- независимое редактирование операций;
- разделение данных между пользователями.

Инструкция тестирования обновлена с учётом текущей конфигурации `pytest` и изолированной тестовой базы.

Коммит документации:

`7a6d5f0` — `Document annual budget and test setup`

Изменения успешно отправлены в GitHub:

https://github.com/im449/work-list-planner/commit/7a6d5f0

Проверка `git diff --check` прошла без ошибок. В коммит документации включён только файл `README.md`.

Обновление документации не потребовало перезапуска приложения.

5. Что изучено в ходе занятия

В ходе работы закреплены следующие практические навыки:

1. Реализация повторяющихся операций в финансовом приложении.
2. Обработка календарных дат и границ месяцев.
3. Разделение плановых и фактически исполненных операций.
4. Организация доступа к данным с учётом пользователя.
5. Изоляция тестовой базы данных от рабочей.
6. Запуск и интерпретация результатов автоматизированных тестов.
7. Работа с Git: проверка изменений, подготовка коммита и отправка в GitHub.
8. Поддержание README в соответствии с текущей функциональностью проекта.

Важный принцип разработки: недостаточно реализовать новую возможность — необходимо проверить её поведение, убедиться в безопасности работы с данными и обновить документацию.

6. Текущее состояние проекта

На момент завершения занятия:

- функциональность ежемесячных повторяющихся операций реализована;
- обработка дат с учётом длины месяца и високосных лет предусмотрена;
- существующий набор из 40 тестов проходит успешно;
- тестовая база изолирована от рабочей;
- README обновлён;

- изменения опубликованы в GitHub;
- работающий сервис не перезапускался при публикации документации.

Проект продолжает развиваться как полноценное веб-приложение, а не только как учебный пример.

7. Следующее занятие

Общая сумма расходов за месяц без детализации операций

Следующая задача — добавить возможность указать общую сумму расходов за выбранный месяц одной строкой, даже если подробные операции за этот месяц отсутствуют.

Например, пользователь хочет зафиксировать, что в сентябре было потрачено 45 000 ₽, но не хочет вручную создавать записи для продуктов, транспорта, бытовых покупок и других расходов.

В следующем занятии необходимо:

1. Продумать, как хранить месячный итог отдельно от детализированных операций.
2. Добавить интерфейс ввода и редактирования общей суммы.
3. Реализовать сохранение данных через API.
4. Не допустить двойного учёта суммы, если позднее пользователь добавит отдельные расходы.
5. Добавить тесты на сохранение, изменение, чтение и разделение данных пользователей.
6. Проверить, как месячный итог будет отображаться в сводке бюджета.

Главная цель следующего занятия: научить Work List Planner учитывать общий месячный расход без обязательного внесения каждой отдельной операции, сохранив корректность финансовых итогов.

Итог

В этом занятии мы завершили важный этап развития годового бюджета, проверили существующую функциональность автоматизированными тестами и привели документацию в соответствие с кодом.

Работа с проектом продолжает строиться по практическому циклу: реализация → тестирование → проверка изменений → публикация в GitHub → документирование → следующий этап развития.

Revision #1
Created 2026-10-15 21:45:00 UTC by im
Updated 2026-10-15 21:45:29 UTC by im