

Лабораторна робота 2. Звіти про дефекти, історія тестування та перший конвеєр

Тема. Розслідування дефектів і звітування про них, зокрема про переміжний дефект; звітування про саме тестування; перша автоматизована перевірка, яка виконується на кожен коміт.

Мета. Повідомляти про дефекти так, щоб їх виправляли; звітувати про тестування так, щоб йому вірили; побачити, що зелений конвеєр говорить і чого не говорить.

Застосунок для тестування

OrderFlow 1.1 постачається як вихідний код, `OrderFlow_lab2.zip`. Розпакуйте архів і запустіть застосунок із **тим самим номером варіанта, що й у лабораторній роботі 1**:

```
ORDERFLOW_VARIANT=<ваш номер у списку групи> dotnet run --  
project src/OrderFlow.Api
```

Реліз 1.1 не містить функціональних змін. У ньому додано засоби діагностики, описані в `kit/PRODUCT.md`: сторінка **Outbox (вихідні)**, яка зберігає копію кожного електронного листа, надісланого застосунком; сторінка **Log (журнал)** із кожним запитом, його статусом і тривалістю та ідентифікатором помилки для кожної внутрішньої помилки; кнопка **Reset demo data (скинути демонстраційні дані)** і `POST /api/test/reset`; сторінка **Support (підтримка)** (лише для менеджера) з повідомленням, яке переслали працівники відділу продажів або бухгалтерії. Додано п'ять демонстраційних замовлень.

Також ви маєте використати:

- власний звіт із лабораторної роботи 1: таблицю оракулів, аркуші сесій, список запитань, пропозиції щодо тестопридатності;
- `tests/OrderFlow.Smoke` — п'ять димових тестів, які виконуються командою `dotnet test`, і `.github/workflows/ci.yml` — заготовку робочого процесу (workflow) для GitHub Actions, у якій тригер налаштовано, а кроки залишено для заповнення;
- шаблони в `kit/templates/`: `investigation_note.md`, `bug_report.md`, `significance_table.md`, `proof_report.md`, `dashboard.md`;
- одну знахідку зі звіту одnogрупника з лабораторної роботи 1, призначену в парах (завдання 4);
- обліковий запис GitHub. Трекер issue вашого власного репозиторію є трекером дефектів для цієї роботи.

Облікові записи залишаються без змін: `manager / manager`; `alice / alice (VIP)`, `bob / bob`, `carol / carol`.

Правила роботи

- Записуйте спостереження окремо від висновків, як у лабораторній роботі 1. Там, де ви не впевнені, говоріть це безпечною мовою: «схоже», «я ще не бачив», «я припускаю».
- Знахідка, яка спирається на неоднозначну заяву, залишається питанням, а не дефектом; питання не стають issue.
- Кожен звіт називає свій оракул. «Очікувано: має працювати» — не оракул.
- Трекер — це запис. Не редагуйте issue після подання звіту; додайте коментар.
- Обмежуйте розслідування в часі. Правило з лекції — десять хвилин, а потім запитати розробника; так як тут ви розробника запитати не можете, вам дається ліміт в шістдесят хвилин. Якщо після завершення часу проблему не вдається відтворити на вимогу, повідомте те, що маєте, безпечною мовою, і скажіть, що робили б далі.
- Кожну навмисну зміну коду в завданні 6 робіть окремим комітом і потім відкочуйте. Ніколи не залишайте конвеєр червоним.

Хід роботи

Завдання 0. Підготовка (20 хвилин)

Розпакуйте архів. Створіть на GitHub новий репозиторій з назвою `orderflow-lab2` — публічний або приватний із викладачем, доданим як співавтор (collaborator), — і надішліть у нього вихідний код:

```
git init && git add . && git commit -m "OrderFlow 1.1 as received"
git branch -M main && git remote add origin <URL вашого репозиторію> && git push -u origin main
```

Увімкніть Issues у налаштуваннях репозиторію. Запустіть застосунок зі своїм номером варіанта. Увійдіть як менеджер і прочитайте повідомлення на сторінці Support: воно адресоване вам на все завдання 1. Виконайте `dotnet test` у теці рішення і переконайтеся, що п'ять тестів пройдено.

Зафіксуйте: URL репозиторію, ідентифікатор першого коміту, останній рядок виводу `dotnet test` і повідомлення зі сторінки Support, скопійоване дослівно.

Завдання 1. Розслідування повідомлення (60 хвилин)

Повідомлення — це свідчення очевидця. Поставтеся до нього так, як описано в теоретичних відомостях: вислухайте його, збережіть відстороненість від його висновків і з'ясуйте, що в ньому пропущено. Використайте `templates/investigation_note.md` і ведіть під час роботи

нотатки з мітками часу, записуючи спостереження й інтерпретації в окремих рядках.

- Спершу перевірте власні припущення: збірка і варіант, обліковий запис, стан даних, чи дивитеся ви на сторінку чи на API.
- Керуйте змінними по одній: ввід (яке замовлення, які товари, які кількості), стан (свіжі дані після скидання; дані, залишені попередніми замовленнями), час (як швидко одна дія йде за іншою; скільки ви чекаєте між ними).
- Поліпшіть спостереження, перш ніж будувати теорії. Outbox показує, що було надіслано; Log показує кожен запит із часом і статусом; вкладка Network у браузері та `curl -i` показують заголовки відповіді; API показує ті самі дані, що й сторінка, але без сторінки.
- Спрощуйте, потім ускладнюйте. Прибирайте кроки, доки проблема не зникне; додавайте навантаження, повторення і другого користувача, доки вона не з'явиться.
- Зупиніться, коли зможете або викликати проблему на вимогу, або точно сказати, чого саме вам не вдалося викликати.

Зафіксуйте: нотатки розслідування. Вони закінчуються висновком, який називає, **яка з п'яти можливостей** (спостереження, інша система, інший стан, інший ввід, неправильна модель) пояснює повідомлення, що виключило решту, яким було вирішальне спостереження, і кроки, які відтворюють проблему на вимогу, якщо ви їх знайшли (усе має бути написано безпечною мовою).

Завдання 2. Чотири формальні звіти про дефекти (60 хвилин)

Напишіть чотири звіти як issue у своєму репозиторії за шаблоном `templates/bug_report.md`: проблему із завдання 1 і три проблеми з вашої таблиці оракулів лабораторної роботи 1, які ви вважаєте найбільш значущими. Курйози й питання не включайте.

Для кожного звіту:

- **Назва:** одне речення приблизно з дванадцяти слів, проблема на початку, без префікса з місцем. Запит на вдосконалення починається зі слова «Будь ласка».
- **Версія та середовище:** OrderFlow 1.1, номер варіанта, браузер або клієнт, спосіб запуску застосунку. Підтвердьте проблему на 1.1; знахідка з лабораторної роботи 1, яка більше не відтворюється, у звіт не потрапляє.
- **Опис:** короткий. Не пишіть того, що всі й так знають.
- **Кроки:** лише там, де вони не очевидні. Останній крок починається словами «Зверніть увагу, що...».
- **Чому це проблема:** оракул, названий. Не «має працювати правильно».
- **Чому це важливо:** кого стосується, як часто, що втрачається.

— **Обхідний шлях**, якщо ви його знаєте; **вкладення**: знімок екрана або запит і відповідна частина відповіді, номери замовлень, записи журналу, ідентифікатор помилки.

Дайте кожному звіту найкращий фокус: найпереконливіше формулювання, яке при цьому залишається правдивим, найсильніший оракул першим, одна проблема на звіт.

Зафіксуйте: посилання на чотири issue і повний текст кожного issue в додатку до звіту.

Завдання 3. Значущість (20 хвилин)

Заповніть `templates/significance_table.md` для чотирьох звітів: частота, шкода, наслідки для інших ризиків, розголос, а потім запропоновані серйозність і пріоритет із обґрунтуванням в одне речення для кожного. Назвіть один дефект, для якого ви підняли б **чорний прапор**, і поясніть чому, або поясніть, чому жоден із чотирьох на це не заслуговує.

Зафіксуйте: таблицю та абзац про чорний прапор.

Завдання 4. Історія тестування (45 хвилин)

Напишіть історію всього вашого тестування OrderFlow на цей момент — трьох сесій лабораторної роботи 1 і розслідування із завдання 1 — у трьох частинах.

— **Звіт PROOF** за шаблоном `templates/proof_report.md`, написаний телескопічно: підсумок у два речення, який говорить, чи все гаразд із продуктом і чи все гаразд із тестуванням; абзац; потім п'ять розділів — Past (минуле), Results (результати), Obstacles (перешкоди), Outlook (перспективи), Feelings (відчуття).

— **Низькотехнологічна панель тестування** за шаблоном `templates/dashboard.md`: від п'яти до восьми областей, узятих із вашої схеми покриття з лабораторної роботи 1, для кожної — зусилля (Effort), покриття 0–3, колір якості та коментар. Кожну клітинку має бути можливо обґрунтувати вашими аркушами сесій і звітами.

— **Схема покриття релізу**: ваша схема з лабораторної роботи 1, у якій кожен запис розфарбовано за рівнями Голланда (нічого, базова перевірка, більше ніж базова, типові випадки, деякі межові випадки, глибокі межові випадки), а нижчі рівні схеми приховано, щоб читач бачив увесь продукт одним поглядом.

В історії ніде не має бути кількості тест-кейсів чи дефектів. Номер issue як посилання допустимі.

Зафіксуйте: звіт PROOF, панель, розфарбовану схему.

Завдання 5. Конвеєр (75 хвилин)

1. Увімкніть GitHub Actions (або аналогічний механізм у іншому сервісі контролю версій) у своєму репозиторії. Доповніть

.github/workflows/ci.yml так, щоб на кожне надсилання (push) і кожен pull request він відновлював залежності, збирав рішення і виконував димові тести (dotnet restore, dotnet build, dotnet test для OrderFlow.slnx). Надішліть зміни, відкрийте вкладку Actions і дочекайтеся зеленого завершення запуску.

2. **Навмисна зміна (а).** У `src/OrderFlow.Domain/Pricing.cs` змініть ставку звичайної знижки з 0.10 на 0.25. Зробіть окремий коміт із повідомленням «experiment: discount rate» і надішліть. Спостерігайте за запуском.
3. **Навмисна зміна (б).** У `src/OrderFlow.Api/ApiEndpoints.cs` перейменуйте маршрут `/products` на `/catalog`. Зробіть окремий коміт із повідомленням «experiment: route rename» і надішліть. Спостерігайте за запуском.
4. Відкотіть обидва експерименти командою `git revert`, надішліть і подивіться, як запуск знову стає зеленим.
5. Прочитайте червоний запуск як звіт і напишіть роздум на пів сторінки: які з трьох рівнів історії тестування запуск вам дає, а яких не дає; що довів зелений запуск після зміни (а) і чого він не довів; якого рівня покриття за шкалою Голланда досягають п'ять димових тестів; хто тепер має розслідувати і що йому потрібно з того, чого запуск не дає. Необов'язково: додайте друге завдання (job), яке запускає застосунок і виконує кілька ваших запитів із лабораторної роботи 1 за допомогою `curl`, або колекцію Postman за допомогою Newman.

Зафіксуйте: посилання на чотири запуски (зелений, зелений, червоний, зелений), знімок екрана з журналом кроку, що не пройшов, і роздум.

Вимоги до оформлення звіту

Подайте один документ `Lab2_<Прізвище>_<Варіант>.pdf` (або `.docx`), що містить у такому порядку:

1. Номер варіанта, URL репозиторію, ідентифікатори першого коміту та чотирьох комітів експериментів і відкатів.
2. Нотатку розслідування (завдання 1).
3. Посилання на чотири issue (завдання 2).
4. Таблицю значущості та абзац про чорний прапор (завдання 3).
5. Звіт PROOF, панель і розфарбовану схему (завдання 4).
6. Посилання на чотири запуски конвеєра, знімок екрана кроку, що не пройшов, і роздум (завдання 5).
7. Висновок на п'ять–десять рядків: який із ваших звітів, як ви очікуєте, виправлять першим і чому, і що тепер захищає конвеєр.
8. Додаток: повний текст чотирьох issue.

Контрольні питання

1. Що довів зелений запуск після зміни знижки і чого він не довів?
2. Яка з п'яти можливостей пояснила повідомлення у вашій скриньці?
Яким було вирішальне спостереження і що ви виключили до нього?
3. Чому у ваших звітах немає рядка «Очікуваний результат»? Коли ви його написали б?
4. Розробник відповідає на ваш звіт: «це проблема лише для новачків». Що ви скажете?
5. Покажіть одну клітинку своєї панелі та обґрунтуйте її своїми аркушами сесій.
6. Згадка мимохідь, формальний звіт, чорний прапор: що з цього ви застосували або застосували б до яких зі своїх знахідок і чому?
7. Який із ваших звітів має найслабші підстави для виправлення, і чи подали б ви його все одно?