Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчет по интеграционному тестированию

по дисциплине «Технология разработки качественного программного продукта»

Выполнили студенты гр. 3530904/80105: Пинаев Н. Д.
Распереза А. Д.

Руководитель: Маслаков А. П.

Санкт-Петербург
2022 г.

Задание

Интеграционное тестирование предполагает наличие нескольких модулей или, если приложение построено в соответствии с микросервисной архитектурой, микросервисов разработанного приложения. В качестве модулей могут быть функциональные части (например: модуль авторизации\аутентификации, модуль взаимодействия с пользователем, модуль интеграции со сторонним сервисом).

Необходимо выполнить интеграционное тестирование нескольких модулей в соответствии с предварительно описанными сценариями тестирования. Каждый сценарий тестирования должен представлять собой некоторый законченный вариант использования системы (кейс) пользователя.

Интеграционное тестирование предполагает командную работу над программным продуктом, поэтому необходимо проводить его с использованием одного из инструментов CI, доступных на рынке: Gitlab, Jenkins, Bamboo, TeamCity, etc.

Минимальное количество сценариев - 10.

Обязательные требования:

1. Предварительное формирование документа, описывающего тестовые сценарии (утверждение сценариев у преподавателя).
2. Применение заглушек (mock-сервисов или mock-объектов) для изоляции от окружения и внешних сервисов или для ускорения прохождения тестов.
3. Рассмотрение негативных сценариев тестирования.
4. Поднятие сервера непрерывной интеграции и запуск задачи по интеграционному тестированию по временному триггеру или по событию изменения кода приложения\интеграционных тестов.

Отчёт по интеграционному тестированию должен содержать:

1. Отчёт о выполненной работе, использованных инструментах, применённых техниках тест-дизайна.
2. Тест-план со словесным описанием тестовых сценариев, которые планируется реализовать в интеграционном тестировании (необходимо предварительно утвердить у преподавателя)
3. Отчёт о прохождении тестов с результатами на сервере непрерывной интеграции.
4. Описание процедуры расширения тестового набора на примере добавления новой функциональной части (или модуля)

Описание выполненной работы, использованных инструментов, применённых техниках тест-дизайна.

В CI/CD пайплайн был добавлен шаг, который запускает интеграционные тесты при пуше в мастер

Инструменты:

- Pytest
- GitHub actions
- protobuf

В интеграционных тестах упор был сделан на работу внутренних модулей программы и на авторизацию/аутентификацию в Google Play/AppStore, не были затронуты конечные решения для пользователя такие как PyPi и Docker. И также на взаимодействие с Google Play API и App Store ITunes клиентом для проверки работы модулей авторизации и аутентификации.

Модули которые взаимодействуют между собой:

AppStore, StoreClient для работы с Apple Store маркетом, GooglePlayAPI и DeviceBuilder который является заглушкой(mock) реального телефона

С помощью фикстуры capfd получаем вывод из консоли, который и можем проверять на корректность.

Техники тест дизайна

- **фаззинг(fuzzing)**

Для использования фаззинга мы отправляем вместо корректных параметров рандомные наборы символов и проверяем что программа корректно их обрабатывает, пример:

```
def test_app_store_login_first_try_fuzzing(capfd):
    apple_id = ''.join(random.choices(string.ascii_lowercase, k=10))
    test_pass = ''.join(random.choices(string.ascii_lowercase, k=10))
    login_no_2fa = subprocess.run(
        ['python3', f'{os.getcwd()}/mdast_cli/mdast_scan.py', '--
distribution_system', 'appstore', '--appstore_apple_id',
         f'{apple_id}', '--appstore_password2FA', f'{test_pass}', '--
appstore_app_id',
         'aaaa'])
    out, err = capfd.readouterr()
    assert login_no_2fa.returncode == 4
    assert "ERROR Store authenticate failed! Message:
MZFinance.BadLogin.Configurator_message" in out
```

- **Угадывание ошибок (error guessing)**

Нужно отлавливать сбои прогаммы, которые будут ожидаемы, например заданием некорректной последовательности операций. Для использования такой техники тест дизайна необходимо хорошо знать продукт.

Пример:

```
def test_google_play_authenticate_fuzzing_check_failed():
    gsfId = 1337
    auth_token = ''.join(random.choices(string.ascii_lowercase, k=10))
    test_package_name = config['gp']['insta_package_name']
    gp_api = GooglePlayAPI()
    gp_api.login(email=None, password=None, gsfId=int(gsfId),
authSubToken=auth_token)
    try:
        gp_api.details(test_package_name)
    except ConnectionError as e:
        assert True
        assert "Error retrieving information from server" in e.args[0]
```

ConnectionError при отсутствии успешной аутентификации по токену.

Тест план

Модули:

- AppStore API
- Google Play API
- DeviceBuilder
- Poly App Downloader entrypoint
- Класс gp_api
- Класс appstore

№	Сценарий	Ожидаемое поведение	Тестируемые модули	Ход выполнения
1	test_get_help (Получить информацию о заголовках)	Вывод информации о параметрах для работы в виде мануала(man)	Poly App Downloader entrypoint	1)Вызвать энтрипоинт с параметром ‘-h’ 2)Проверить успешность работы и корректность мануала

2	test_device_builder_mock_device_setup_with_mocked_data (создание фэйкового девайса с данными из мока)	Девайс с данными из мока создан	DeviceBuilder Класс gp_api	1)Инициализировать класс DeviceBuilder с device=’mocked’ 2)Получить информацию о девайсе 3)Проверить данные на правильность сравнив их с мокированными данныыми
3	test_device_builder_get_mocked_android_build(получение билда андроида для отправки в Google API)	Билд готовый к отправке в Google Play для создвания нового устройства получен	DeviceBuilder Класс gp_api	1)Вызвать метод getAnroidBuild 2)Получить его значения 3)Сравнить их с ожидаемыми 4)Проверить что был использован mock
4	test_google_play_login_needs_browser_check_for_error_no_token(проверка ошибки при первом логине в гугл плэй по логину + паролю)	Ошибка: local variable 'ac2dmToken' referenced before assignment при вызове метода из класса Google Play APi	Google Play API Класс gp_api	1)Попытаться залогиниться по логину + паролю 2)Проверить что при попытке логина после 15 мая 2022 года гугл временно больше не дает залогиниться через браузер, а в ответе BadAuth, поэтому программа выдает ошибку про токен, который не пришел в ответе 3)Поймать UnboundLocalError
5	test_google_play_login_fail_fuzzing (проверить что при попытке логина с рандомными данными в	Ошибка: local variable 'ac2dmToken' referenced before assignment	Google Play API Класс gp_api	1)Попытаться залогиниться по логину + паролю полученных из случайных символов 2)проверить что в ответе BadAuth,

	логине + пароле ошибка)	при вызове метода из класса Google Play APi		поэтому программа выдает ошибку про токен, который не пришел в ответе 3)Поймать UnboundLocalError
6	test_google_play_authenticate_and_check_api_works (проверка успешной аутентификации через gsfid + auth_token)	Аутентификация пройдена успешно Апи работает	Google Play API Класс gp_api	1)Подать на вход корректные gsfid + auth_token 2)Попытаться пройти аутентификацию 3)Она пройдена успешно 4)Попытаться вызвать любой метод из апи 5)он работает корректно
7	test_google_play_authenticate_fuzzing_check_failed()	Токен задан, ошибок нет Апи не работает, ConnectionError	Google Play API Класс gp_api	1)Подать на вход gsfid + auth_token из случайных данных 2)Попытаться пройти аутентификацию 4)Попытаться вызвать любой метод из апи 5)апи не работает, ошибка “ConnectionError’
8	test_app_store_login_first_try_wrong_pass(прохождение авторизации с неправильным паролем))	авторизация не будет пройдена, в логах будет отображен код ошибки = 4	AppStore API Класс appstore Poly App Downloader entrypoint	1. Получить apple id и test pass из конфига 3. Вызвать entry point с полученными apple id и test pass 4. Проверить, что авторизация не прошла, exit code = 4, в логах отображена информация о процессе.
9	test_app_store_login _fuzzing(прохожение авторизации с рандомными	авторизация не будет пройдена, в логах будет отображен код	AppStore API Класс appstore Poly App	1. Получить рандомное значение строки для apple id и test pass 2. Вызвать entry point

	значениями))	ошибки = 4	Downloader entrypoint	с полученными apple id и test pass 3. Проверить, что аутентификация не прошла, exit code = 4, в логах отображена информация о процессе.
10	test_app_store_login_and_check_api_works(прохождение авторизации и дальнейшая возможность работы с api)	авторизация будет пройдена и имеется возможность дальнейшей работы с api	AppStore API Класс appstore Poly App Downloader entrypoint	1. Получить креды из конфига 2. Получить валидный bundle id 3. Вызвать entry point с полученными кредами и bundle id 4. Проверить, что аутентификация прошла успешно и возможна дальнейшая работа с api.

Описание работы

В пайплайн был добавлен шаг с интеграционными тестами

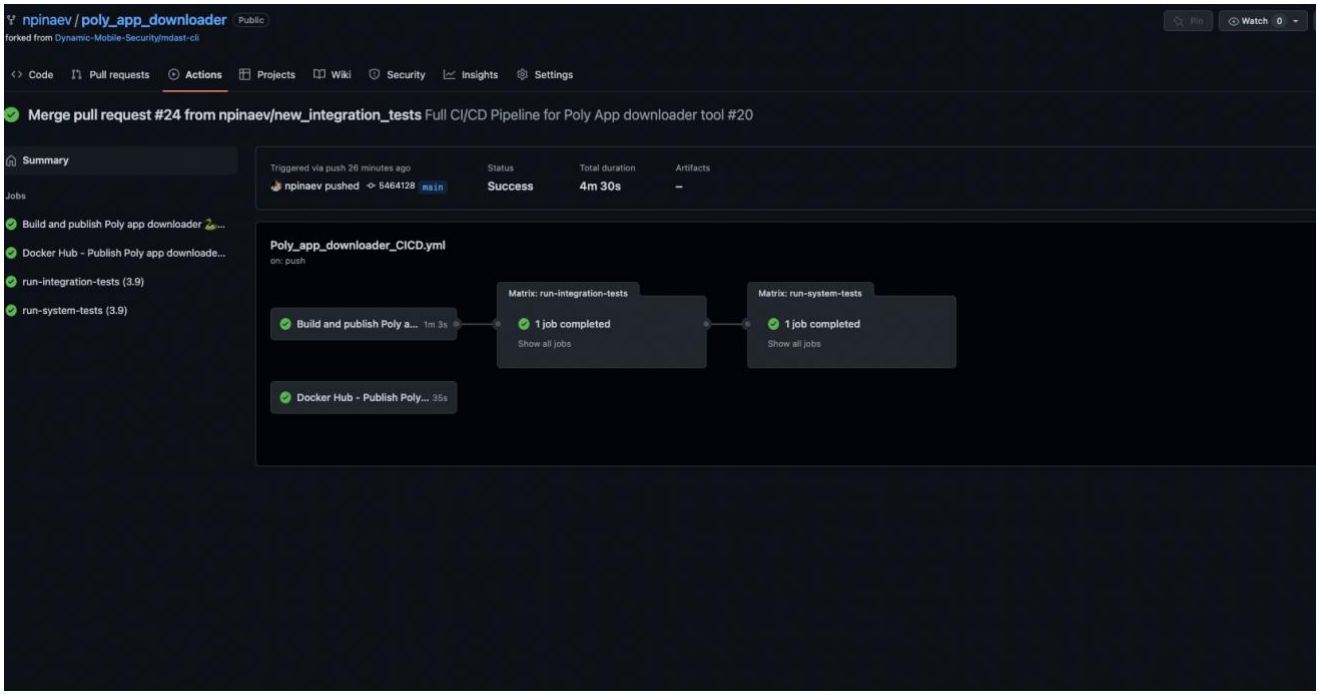
Для проверки и использования mock-данных возьмем взаимодействие Google Play API и DeviceBuilder.

В качестве мок-объекта был реализован фэйк-девайс в классе DeviceBuilder с device_codename='mocked'

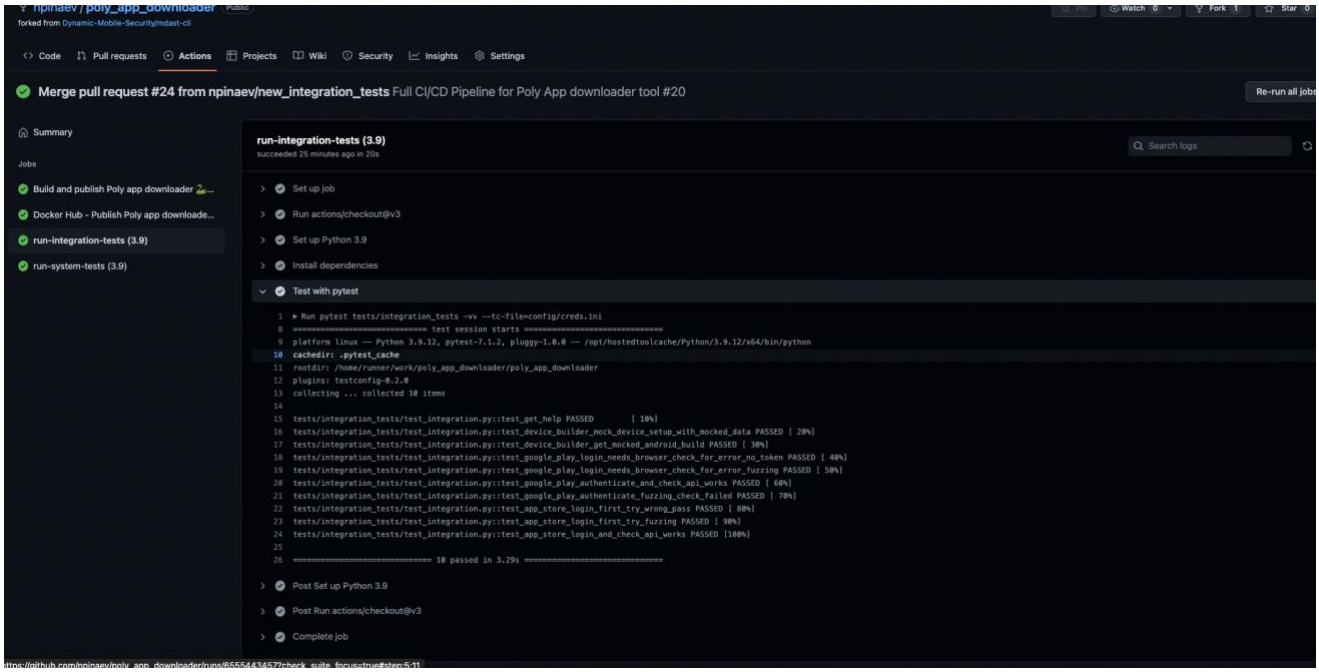
Остальные тесты в основном на модули авторизации/аутентификации и описаны в тест плане.

Отчёт о прохождении тестов с результатами и оценкой покрытия кода тестами.

Скрин пайплайна:



Скрин прохождения тестов:



Описание процедуры расширения тестового набора на примере добавления новой функциональной части (или модуля)

Для добавления новой функциональности и расширения тестового набора надо учитывать такие факторы как:

- Изменение нынешнего кода – если в процессе рефакторинга поменялась старая функциональность, то необходимо исправить все тесты, которые

это затронуло

- Тесты должны быть написаны в одном стиле, используются те же фикстуры, те же методы, которые были реализованы, нельзя реализовывать одну и ту же функциональность дважды
- Для каждого автоматизированного теста должен быть написан тест-кейс, описывающий необходимые требования, предусловия, ожидаемый результат и шаги выполнения
- Тесты не должны падать без изменения кода, то есть должны быть выполнены критерии по стабильности