Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Методи тестування мобільних застосунків
(тема)

Виконав: студент II курсу, групи СПм-22-1

Шапошник М.С.
(прізвище, ініціали)

Спеціальність *123 _ Комп'ютерна інженерія*

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої прграми)

Керівник ст. викл. к.т.н. Єрьоміна Н.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Електронних обчислювальних машин
Рівень вищої освіти другий (магістерський)
Спеціальність 123_Комп'ютерна інженерія
(шифр і назва)
Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системне програмування
(повна назва

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Шапошнику Максиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи тестування мобільних застосунків

затверджена наказом по університету від " 06 " листопада 2023 р. № 1299 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024 р.

3. Вхідні дані до роботи 1) документація Android; 2) документація IOS; 3) існуючі алгоритми та методи тестування мобільних застосунків

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз предметної області;
2) визначення та аналіз основ мобільних застосунків;
3) порівняння існуючих методів та алгоритмів тестування мобільних застосунків;
4) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Слайд-презентація – 9 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження	06.11.2023 -07.11.2023	
2	Аналіз предметної області та опрацювання джерел	08.11.2023 -19.11.2023	
3	Аналіз мобільних операційних систем	20.11.2023 -22.11.2023	
4	Аналіз архітектури мобільних платформ	23.11.2023 -26.11.2023	
5	Аналіз інструментів для тестування мобільних застосунків	27.11.2023 -10.12.2023	
6	Аналіз методів та підходів до тестування мобільних застосунків	11.12.2023 -24.12.2023	
7	Оформлення пояснювальної записки	25.12.2023 -26.12.2023	
8	Перевірка виконаного проекту керівником,	27.12.2023 -09.01.2024	
9	Захист проекту	10.01.2024 -12.01.2024	

Дата видачі завдання 06.11.2023

Студент _____
(підпис)

Керівник роботи (проекту) _____ ст.викл. к.т.н. Єрьоміна Н.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 64 сторінки, 9 рисунків, 1 таблицю,1 лістинг, 6 джерел посилання, 1 додаток.

МЕТОД, ТЕСТУВАННЯ, ЗАСТОСУНОК, ANDROID, НАВАНТАЖЕННЯ, ОПЕРАЦІЙНА СИСТЕМА, МОБІЛЬНИЙ ПРИСТРІЙ

Метою кваліфікаційної роботи є дослідження методів та алгоритмів тестування мобільних застосунків

Методи тестування мобільних застосунків є важливим етапом у процесі розробки мобільних додатків. В даному дослідженні було розглянуто різноманітні підходи та методи, які дозволяють забезпечити якість та надійність програмних продуктів для мобільних пристроїв.

ABSTRACT

Master’s thesis 64 pages, 9 figures, 1 table, 1 listing, 6 sources, 1 appendices

METHOD, TESTING, APP, APPLICATION, ANDROID, LOAD, OPERATING SYSTEM, MOBILE DEVICE

The purpose of the qualification work is to research methods and algorithms for testing mobile applications.

Mobile application testing methods are an important stage in the mobile application development process. In this study, a variety of approaches and methods were considered to ensure the quality and reliability of software products for mobile devices.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Android — мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим вихідним кодом.

Wi-Fi — технологія бездротової локальної мережі з пристроями на основі стандартів IEEE 802.11

3G — технології мобільного зв'язку 3-го покоління — набір послуг, що поєднує як високошвидкісний мобільний доступ з послугами мережі Інтернет, так і радіозв'язку, що створює канал передачі даних.

LTE — стандарт високошвидкісної бездротової передачі даних для мобільних телефонів та інших терміналів, що працюють з даними.

Kotlin — це кросплатформова статично типізована мова програмування.

ВСТУП

Під час взаємодії людини з комп’ютером, процесу пошуку інформації чи виконання команд за допомогою компонентів графічного інтерфейсу, можливе встановлення з ними прямого фізичного контакту. Автоматизація процесу тестування має вирішальне значення для цього процесу. Макроси та сценарії, які керують компонентами графічного інтерфейсу або посилаються на компонент за його назвою, яка є сторонньою або недоступною для користувача, або область екрана, яка можна змінити.

У сучасному цифровому світі мобільні застосунки відіграють важливу роль і стають необхідним елементом життя мільйонів людей. З кожним днем кількість мобільних додатків зростає, і вони стають все більш варіативними, функціональними та інноваційними. Цей феномен створює нові можливості для користувачів і розробників, але одночасно викликає серйозні виклики у сфері якості, безпеки та надійності мобільних додатків.

Ця робота присвячена дослідженню проблем в тестуванні. Основна мета інтерфейсу користувача — сприяти швидкому розвитку автоматизованих тестів для мобільних додатків і методів, які можна використовувати для прискорення їх виконання.

Однією з основних причин актуальності дослідження методів тестування мобільних застосунків є різноманітність і складність функціональності таких додатків. Мобільні застосунки використовуються для найрізноманітніших завдань, починаючи від комунікації і закінчуючи фінансовим управлінням та медичною діагностикою. Оскільки вони стають невід'ємною частиною сучасного бізнесу, освіти та розваг, помилки та дефекти в мобільних додатках можуть призвести до серйозних наслідків, включаючи фінансові втрати, порушення конфіденційності даних та негативний вплив на репутацію розробника.

Крім того, сфера мобільних технологій є динамічною та постійно

змінюється. Постійно виходять нові версії операційних систем, апаратних платформ, а також змінюються тенденції в розробці програмного забезпечення. Це ставить виклик перед тестувальниками мобільних додатків, оскільки вони повинні постійно адаптувати та вдосконалювати свої методи тестування для забезпечення сумісності і надійності на нових платформах і пристроях.

Зазначу також, що створення високоякісних мобільних додатків стає важливим завданням для розробників не тільки з позиції конкурентоспроможності, але й з етичної точки зору, оскільки користувачі вірять своїм мобільним пристроям і передають їм важливу інформацію про себе. Тому дослідження та вдосконалення методів тестування мобільних застосунків є вкрай важливим завданням, яке спрямоване на захист інтересів користувачів і підвищення загальної якості цифрового середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Мобільні пристрої залишаються важливими в сучасному суспільстві та відіграють значну роль у нашому повсякденному житті. Вони важливі для забезпечення спілкування між людьми, дозволяючи їм здійснювати голосові та відео дзвінки, обмінюватися повідомленнями та використовувати соціальні мережі.

Інші мобільні пристрої забезпечують швидкий і простий доступ до Інтернету, ці пристрої необхідні для отримання інформації, новин і доступу до веб-сайтів у режимі реального часу. Вони стали звичайними розважальними пристроями, які дозволяють дивитися відео, слухати музику, грати в ігри та фотографувати. У сфері продуктивності та роботи мобільні пристрої допомагають виконувати завдання в дорозі, спілкуватися з колегами електронною поштою, використовувати офісні програми та співпрацювати над документами з іншими. Їх також залучають для самостійного тренування та навчання, надаючи доступ до інтерактивних програм та онлайн-курсів.

Мобільні пристрої також допомагають у питаннях здоров'я та благополуччя, дозволяючи людям відстежувати свою фізичну активність, контролювати показники здоров'я та використовувати додатки для моніторингу здоров'я. Вони також відіграють певну роль у забезпеченні безпеки, наприклад, завдяки системі розпізнавання обличчя та відбитків пальців.

У міру розвитку технологій мобільні пристрої продовжують впливати на наше життя, виводячи на ринок нові функції, такі як доповнена реальність (AR), штучний інтелект і високошвидкісний інтернет (5G). Загалом, мобільні пристрої продовжують залишатися важливою частиною нашого повсякденного життя, сприяючи зручності, комунікації та розвитку. Велику частину життя ми ,все ж таки, проводимо разом з цими технологіями, що вже стало невід’ємною частиною нашого існування.

1.1 Актуальність мобільних пристроїв

У 2023 році смартфони вже давно виросли з простого засобу зв'язку до невід'ємної частини повсякденного життя людей. З кожним роком вони стають все більш потужними, функціональними та важливими для різних сфер нашого існування. Актуальність смартфонів у 2023 році визначається рядом факторів, що охоплюють технологічний прогрес, зручність, комунікацію, розваги, організацію робочого та особистого часу. Технологічний прогрес є основним каталізатором актуальності сучасних смартфонів. У 2023 році вони вражають своєю потужністю, швидкістю та функціоналом. Нові процесори, вдосконалені камери, великі обсяги оперативної пам'яті та швидкість Інтернет-з'єднання роблять їх неймовірно ефективними для виконання різних завдань, від ведення бізнесу до гри в реалістичні ігри. Смартфони активно використовуються у сфері бізнесу та робочого оточення. Завдяки розвитку додатків для роботи, обміну документами та можливості працювати віддалено, вони стають невід'ємним інструментом для професійного росту та оптимізації робочих процесів. Сучасні смартфони об'єднують в собі не лише функції телефону, але і планшета, ноутбука та інших пристроїв, забезпечуючи повноцінний робочий інструмент в одному пристрої.

У 2023 році смартфони стали ефективним засобом для організації особистого життя. Додатки для планування, трекінгу фізичної активності, здоров'я та фінансів дозволяють людям більш ефективно використовувати свій час та керувати різними аспектами свого життя. Смартфони стали важливим елементом для здійснення покупок онлайн, забезпечуючи зручний доступ до різноманітних товарів та послуг в будь-який час.

Важливою складовою актуальності смартфонів є їх роль у забезпеченні комунікації. Соціальні мережі, месенджери та відеодзвінки дозволяють людям легко утримувати зв'язок з родиною, друзями та колегами. Смартфони стали не лише інструментом обміну повідомленнями, але й засобом для

вираження та спільного перегляду вражень, фотографій та відео.

Розважальний аспект смартфонів також важко переоцінити. Вони стали не тільки музичними плеєрами та відеопрогравачами, але й платформою для гри в ігри, перегляду стрімінгових сервісів та читання електронних книг. Високоякісні екрани, потужні аудіосистеми та ігрові можливості роблять їх ідеальними компаньйонами для розваг у будь-якому місці та часі.

Актуальність смартфонів у 2023 році визначається також їх впливом на розвиток технологій майбутнього. Розпізнавання обличчя, використання штучного інтелекту та розширена реальність вже стають стандартними функціями. Зростаюча кількість сенсорів та можливості взаємодії з оточенням відкривають нові перспективи в галузі розробки додатків та сервісів.

Згідно з результатами дослідження глобальної аналітичної компанії Counterpoint Research – користувачі проводять більше часу на своїх смартфонах, ніж на будь якому іншому пристрої. Половина користувачів смартфонів витрачає більше ніж 5 годин, 26% більше 7 годин на день. Користувачі смартфонів перевіряють свої телефони після ранкового пробудження та перед сном.

Рисунок 1.1 – Час користувачів у телефоні за даними Counterpoint Research

1.2 Аналіз мобільних операційних систем

Мобільні технології почали розвиватися трохи пізніше звичайних ноутбуків і стаціонарних комп'ютерів, тому не дивно, що перша офіційна операційна система з'явилася саме на другому типі пристроїв.

Проте згодом телефони, смартфони та планшети ставали все більш досконалими, їхні можливості розширювалися, тому виникла необхідність створювати спеціально підготовлені для них операційні системи.

Як і будь-який обчислювальний пристрій, мобільні пристрої мають операційну систему. Мобільна операційна система (мобільна ОС) — це операційна система для смартфонів, планшетів, КПК або інших мобільних пристроїв. Мобільна операційна система поєднує в собі функціональність операційної системи ПК із мобільною операційною системою та мобільними функціями: сенсорний екран , стільниковий зв’язок, Bluetooth, Wi-Fi, GPS-навігація, камера, відеозапис, ідентифікація, музичний плеєр, NFC (Near Field Communications) та інфрачервоний порт.

Хоча ноутбуки можна класифікувати як мобільні пристрої, операційні системи, на яких вони працюють, зазвичай не вважаються мобільними пристроями. Це пояснюється тим, що ці операційні системи розроблені для стаціонарних настільних комп’ютерів, які традиційно не мають спеціальних «мобільних» функцій і не потребують їх. Ця відмінність розмита в деяких нових операційних системах, які є комбінацією обох.

Портативні пристрої мобільного зв’язку (наприклад, смартфони) мають дві операційні системи. Основну програмну платформу інтерфейсу користувача доповнює друга власна низькорівнева операційна система реального часу, яка обслуговує радіообладнання. Дослідження показали, що ці низькорівневі операційні системи вразливі до зловмисних базових станцій, здатних контролювати мобільні пристрої, які можуть завдати шкоди користувачам, передаючи особисті дані третім особам.

Сучасні мобільні операційні системи: Android, iOS, CyanogenMod,

Cyanogen OS, Fire OS, Flyme OS, Firefox OS, Sailfish OS, Tizen, Ubuntu Touch.

Застарілі програмні платформи більше не підтримуються: Windows Phone, BlackBerry OS, Symbian, Palm OS, webOS, Maemo, MeeGo, LiMo.

1.2.1 Android

Операційна система та платформа для мобільних телефонів і планшетів, створена Google на основі ядра Linux. Підтримується Open Handset Alliance (ONA). Хоча Android базується на ядрі Linux, він виділяється серед спільноти Linux та інфраструктури Linux.

Ядром цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення та програми базуються на цій реалізації Java. Серед найвідоміших товарів на вітчизняному ринку – планшети та смартфони Samsung, HTC, Huawei, SONY, LG, Lenovo. Перша версія ОС вийшла друком у 2008 році на смартфоні HTC, і з того часу невпинно оновлюється.

Основою популярності Android є те, що Android можна встановити на пристрої різних виробників. Такі пристрої більш поширені в країнах, що розвиваються, в Азії, Індії, Африці та Європі.

Переваги ОС Android:

- безліч застосунків. Під цю операційну мережу було написано і щодня пишуться 100 млн ігор, додатків, симуляторів тощо. Вони є платні та безкоштовні;

- відкритий вихідний код. Ця система просто величезний порятунок для людей, які хочуть розробляти різні програми. У неї немає прихованих шифрів і паролів, що дозволяє впровадити та протестувати свої програми, ігри та багато іншого;

- добре підтримує багатозадачність. Багатозадачність — виконання кількох дій одночасно. Якщо у вашому житті бувають дуже часто випадки, що вам потрібно одночасно використовувати кілька соціальних мереж або

додатків, тоді вам найкраще підійте система Android;

- швидкі оновлення. Ця перевага означає те, що Google ретельно стежить за новити тенденціями. Це дозволяє покращувати систему, а саме: модернізувати дизайн, усувати баги, додавати нові алгоритми тощо.

Стрімко розвивається нова професія – розробка додатків для Android. Вважається, що найкращою мовою програмування для Android є Java, за нею йдуть C# і Python. Кожній версії системи присвоєно власну кодову назву на тему десерту (табл. 1.1).

Таблиця 1.1 – Історія версій ОС Android

Кодове ім’я версії	Номер	Дата випуску	Api рівень
Без кодового імені	1.0	23.09.2008	1
	1.1	09.02.2009	2
Cupcake	1.5	27.04.2009	3
Donut	1.6	15.09.2009	4
Eclair	2.0-2.1	26.10.2009	5-7
Fro vo	2.2-2.2.3	20.05.2010	8
Gingerbread	2.3-2.3.7	06.12.2010	9-10
Honeycomb	3.0-3.2.6	22.02.2011	11-13
Ice cream sandwich	4.0-4.0.4	18.10.2011	14-15
Jelly bean	4.1-4.3.1	09.07.2012	16-18
Kitkat	4.4-4.4A	31.10.2013	19-20
Lollipop	5.0-5.1.1	12.11.2014	21-22
Marshmallow	6.0-6.0.1	05.10.2015	23
Nougat	7.0-7.1.2	22.08.2016	24-25
Oreo	8.0-8.1	21.08.2017	26-27
Pie	9.0	06.08.2018	28
Android 10	10.0	03.09.2019	29

Продовження таблиці 1.1

Кодове ім’я версії	Номер	Дата випуску	Api рівень
Android 11	11.0	19.02.2020	29,30
Android 12	12.0	18.02.2021	31,32
Tiramisu	13.0	10.02.2022	33

1.2.2 IOS

Для смартфонів, планшетів, пристроїв, розроблених легендарною компанією Apple виключно для себе, на відміну від Windows Phone (Microsoft) і Android (Google). Ця операційна система була вперше представлена в 2007 році з iPhone. Спочатку він був створений для iPhone і iPod Touch, а пізніше для таких пристроїв, як iPad і Apple TV.

Ядро iOS майже ідентичне ядру операційної системи Apple MacOS (раніше OS X), тому воно використовує той самий набір основних компонентів.

Переваги IOS:

- покращена оптимізація;
- екосистема;
- безпека та захист даних;
- висока якість застосунків;
- оновлення ОС;
- інтуїтивно зрозумілий інтерфейс;
- ексклюзивні застосунки та ігри.

Розшифровка абревіатури iOS (читається як "айос") відсилає до найменувань фірмових мобільних пристроїв Apple (iPhone, iPod та iPad) плюс скорочення OS (Operating System, операційна система). Так фірма наголосила на ексклюзивному характері свого системного мобільного ПЗ.

2 ОСНОВИ МОБІЛЬНИХ ЗАСТОСУНКІВ

2.1 Основні характеристики мобільних застосунків

Мобільні застосунки, безумовно, стали необхідною складовою нашого сучасного життя. З кожним роком вони розвиваються та розширюють свої можливості, стаючи все більш важливими для нашої щоденної життєдіяльності. У цьому розділі буде розглянуто основні характеристики мобільних застосунків та їх вплив на наше сучасне суспільство.

1 Мобільні платформи

Сучасний ринок мобільних додатків домінується двома основними платформами: Android і iOS. Android, розроблена Google, має найбільшу частку ринку через широкий спектр пристроїв, які її підтримують. iOS, створена Apple, славиться своєю ексклюзивністю та вишуканістю. Кожна з цих платформ має свої особливості, вимоги та аудиторію.

Поза Android і iOS також є і інші операційні системи, такі як Windows Mobile, Black Berry OS та інші, які, хоч і не так поширені, все ще мають свою аудиторію та можуть бути важливими для певних сегментів ринку. Це ставить перед розробниками завдання забезпечення сумісності додатків з різними платформами, що вимагає додаткових зусиль.

2 Інтерфейс користувача

Вірно розроблений інтерфейс користувача (UI) – ключовий елемент успіху мобільного застосунку. Якщо інтерфейс не є зручним та інтуїтивно зрозумілим, користувачі можуть відмовитися від використання додатку. Інтерфейс повинен бути зрозумілим, привабливим та ефективним. З еволюцією мобільних додатків з'явилися нові тренди в дизайні, такі як плаский дизайн, мінімалізм та матеріальний дизайн. Ці стилі спрямовані на полегшення взаємодії користувача з додатком та надання йому сучасного та привабливого вигляду для досягнення поставлених цілей.

3 Функціональність

Мобільні додатки можуть бути різноманітними за своєю функціональністю. Вони використовуються для вирішення різних завдань, від спрощення щоденних обов'язків, таких як ведення списків покупок і календарів, до більш складних завдань, таких як відстеження медичних показників та управління фінансами.

Під час розробки мобільних додатків, розробники повинні дбати про ефективність додатка, а також про забезпечення безпеки даних користувачів. Важливо розробляти додатки, які відповідають потребам користувачів та пропонують їм значущий функціонал.

4 Архітектура застосунків

Вибір архітектурного рішення визначає майбутню масштабованість та продуктивність додатка. Розробники можуть обирати між різними підходами, такими як клієнт-серверна архітектура, розподілена архітектура та одноразові додатки. Кожна архітектура має свої плюси та мінуси і вибір залежить від задач та цілей додатку.

5 Тестування та якість

Однією з найважливіших фаз розробки мобільних додатків є тестування. Для забезпечення якості та надійності додатків важливо проводити ручне та автоматизоване тестування. Тестування допомагає виявляти та виправляти дефекти, а також забезпечує безпеку та захист даних користувачів.

6 Вплив на сучасне суспільство

Мобільні застосунки мають глибокий вплив на наше сучасне суспільство. Вони полегшують спілкування, полегшують доступ до інформації, сприяють цифровій трансформації бізнесу та покращують якість життя. Споживачі використовують мобільні додатки для всього, від замовлення їжі до ведення фітнес-журналів. Підприємства використовують додатки для розвитку свого бізнесу та залучення клієнтів. Насамперед, додатки, хоч і не всі, допомагають людям розвиватися, вчити щось нове для

себе, цим самим змушуючи розробників випускати ще більше оновлень та нових програм.

2.2 Особливості розробки мобільних додатків

Розробка мобільних додатків є сучасним і надзвичайно важливим напрямком в інформаційній технології. З кожним роком кількість смартфонів і планшетів зростає, що призводить до збільшення попиту на мобільні додатки для різних потреб. Розробка мобільних додатків має свої особливості та вимоги, які відрізняють її від розробки інших програмних продуктів.

Почнемо з важливості розуміння мобільних платформ. Існує кілька основних мобільних ОС: Android, iOS та Windows Mobile. Кожна з цих платформ має свої власні особливості та інструменти розробки. Розробники повинні визначити, для якої платформи вони хочуть створити застосунок, або вирішити розробляти його для кількох платформ одночасно.

Однією з ключових особливостей розробки мобільних додатків є інтерфейс користувача (UI). Інтерфейс графічного користувача повинен бути привабливим і інтуїтивно зрозумілим. Користувачі мають намір взаємодіяти з додатком через дотик, жести і тапи (доторкання до екрану), тому дизайн інтерфейсу має бути пристосованим до сенсорних пристроїв.

Функціональність є ще однією ключовою особливістю розробки мобільних додатків. Додатки можуть виконувати різні завдання, від надання інформації та розваг до роботи з даними і здійснення платежів. Розробники повинні ретельно вивчити потреби своєї цільової аудиторії і розробити функціонал, який задовольнить ці потреби.

Для розробки мобільних додатків розробники можуть використовувати різні мови програмування і фреймворки. Наприклад, для розробки додатків для iOS використовується мова програмування Swift, а для Android – Java або Kotlin. Вибір мови програмування залежить від конкретних потреб і обмежень проекту.

Мобільні додатки можуть також вимагати інтеграції з різними сервісами і API. Наприклад, застосунок може потребувати доступу до GPS для визначення місця розташування користувача або до камери для зйомки фотографій. Розробники повинні вивчити документацію і забезпечити правильну інтеграцію з необхідними сервісами.

Однією з важливих аспектів розробки мобільних додатків є тестування. Тестування допомагає виявляти та виправляти дефекти та помилки в додатку перед його випуском. Важливо проводити тестування на різних пристроях та платформах, оскільки можуть виникнути проблеми, які відмінні від однієї платформи до іншої.

Також важливо враховувати питання безпеки під час розробки мобільних додатків. Додатки можуть містити конфіденційну інформацію користувачів, тому важливо забезпечити захист від несанкціонованого доступу та втрати даних.

Розробка мобільних додатків є складним та багатоаспектним процесом, який вимагає глибокого розуміння платформи, функціональних вимог, дизайну і безпеки. Однак успішна розробка може принести значні переваги, включаючи розширення аудиторії, покращення користувацького досвіду та зростання бізнесу. Розробка мобільних додатків залишається актуальною та перспективною галуззю інформаційної технології, сегмент постійно розширюється і змінюється, відкриваючи перед розробниками безліч нових можливостей для інновацій та створюючи численні виклики в інформаційному суспільстві.

2.3 Архітектура мобільних платформ

Архітектура мобільних платформ є важливим аспектом розвитку та функціонування сучасних мобільних пристроїв. Мобільні платформи включають у себе як апаратну, так і програмну складову, які спільно працюють для забезпечення функціональності і продуктивності смартфонів

та планшетів.

Апаратна архітектура мобільних платформ є основою для функціонування смартфонів та планшетів. Сучасні мобільні пристрої використовують різноманітні компоненти, такі як процесори, пам'ять, датчики та екрани, для забезпечення високої продуктивності та зручності використання. Процесори в мобільних пристроях зазвичай розроблені спеціально для оптимізації роботи в умовах обмеженого живлення і обробки графіки та даних у реальному часі.

Операційні системи грають ключову роль в архітектурі мобільних платформ. Дві найпоширеніші операційні системи для смартфонів і планшетів – Android та iOS, які мають власні архітектурні рішення. Android базується на ядрі Linux і забезпечує відкритий доступ до додатків та налаштувань, що дозволяє більшу кількість виробників створювати різноманітні пристрої.

iOS, від Apple, відомий своєю високою безпекою та інтеграцією між апаратним і програмним забезпеченням. Він використовує спеціалізований процесор, відомий як Apple Silicon, для оптимізації роботи. Обидві операційні системи мають свої SDK (набори розробника), які дозволяють створювати додатки для платформ.

Архітектура мобільних платформ також включає в себе додатки, які встановлюються на смартфони та планшети. Додатки можуть бути розробленими для певної платформи (наприклад, Android або iOS) або бути крос-платформними, що дозволяє їх використання на різних пристроях.

Магазини додатків, такі як Google Play Store і App Store, виконують ключову роль у розповсюдженні та оновленні додатків. Вони також забезпечують механізми для забезпечення безпеки та перевірки додатків на віруси та шкідливі програми.

Безпека є важливим аспектом архітектури мобільних платформ. Мобільні пристрої містять велику кількість особистої інформації, і захист цієї інформації від несанкціонованого доступу є критичним завданням.

Операційні системи використовують різні механізми для забезпечення безпеки, такі як шифрування даних, автентифікація за відбитком пальця або розпізнавання обличчя.

Додатки також повинні дотримуватися високих стандартів безпеки, оскільки вони мають доступ до різних даних користувачів, які можуть нести в собі важливу інформацію. Для забезпечення безпеки розробники використовують безліч утиліт та власних розробок, щоб зберігати конфіденційність користувача при використанні програми.

Дизайн мобільних операційних систем еволюціонував від настільних операційних систем до вбудованих операційних систем до продуктів, які ми бачимо сьогодні на смартфонах. Під час цього процесу архітектура операційної системи змінюється від складної до простої та зупиняється десь посередині. Цей розвиток сам по собі зумовлений технологічним прогресом у галузі обладнання та програмного забезпечення, а також Інтернет-послуг.

Раніше модель використання мобільних пристроїв була дуже простою. Користувачі запускають програми або ігри для керування даними в режимі офлайн, іноді завантажують статичні веб-сторінки або використовують пошту. Зараз ситуація повністю змінилася: у більше немає функції «встановлення», пристрій в середу виконує роль певного порталу, де багато гравців є провайдерами послуг, створеними незалежними розробниками тощо. Надається багато послуг.

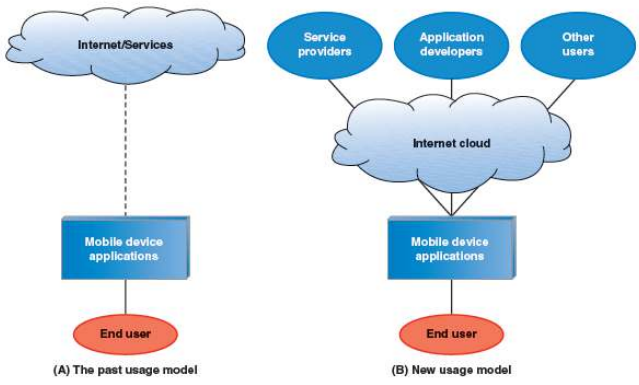

Рисунок 2.1 – Моделі використання мобільних пристроїв

З точки зору споживчих тенденцій, усі поточні мобільні операційні системи (такі як Apple iOS, Google Android, Microsoft Windows) мають більше подібностей, ніж відмінностей:

- всі вони мають документовані SDK з прописаними API, що дозволяє розробникам створювати програми під ці ОС;
- всі вони мають онлайн каталоги додатків, де розробники публікують свої програми та звідки користувачі їх скачують;
- у кожній реалізована багатозадачність та підтримка 3D-графіки, широко використовуються датчики та сенсорні екрани;
- у всіх системах велика увага приділена гладкості та чуйності у взаємодії з користувачем;
- використання інтернету далеко від статичних сторінок, HTML5 стає платформою за замовчуванням для Web-додатків;
- усі ОС підтримують мобільні системи платежів;
- усі системи сфокусовано на оптимізації енергоспоживання.

Спільність сучасних мобільних операційних систем зумовлена глобальними технологічними тенденціями в галузі апаратного та програмного забезпечення, а також у сфері комунікацій. Тепер давайте розберемо операційну систему нового покоління з точки зору наведених вище критеріїв. Одразу бачимо, що між ними досить сильні конфлікти.

Традиційні концепції продуктивності важко застосувати до мобільних пристроїв. Замість статичної продуктивності порівняно зі смартфонами доцільніше оперувати з метою забезпечення комфорту користувача, можливості оптимально реагувати на його дії, що виражається в чуйності, плавності, логічності та точності роботи. Досить часто пристрій А поступається Б за набором критеріїв, але з точки зору користувача цей пристрій оцінюється вище, оскільки тести вимірюють певні підсистеми пристрою. Смартфон не враховує взаємодію з користувачем, і він або вона спочатку оцінює це.

Візьмемо для прикладу відео. Традиційне тестування працює на основі

низки показників, таких як FPS або пропущені кадри. З цим підходом є принаймні дві проблеми. По-перше: відтворення відео — це лише одна дія з усього комплексу, включаючи запуск плеєра, завантаження відео, перемотування назад і т.д. З точки зору користувача, вам потрібно оцінити все разом. Інша проблема полягає в тому, що FPS, основний показник плавності взаємодії, не завжди відображає досвід користувача. Наприклад, під час прокручування зображення в додатку Gallery3D на пристрої B ми побачили помітне уповільнення, але на пристрої A все йшло гладко, хоча FPS там був нижчим. Щоб зрозуміти, в чому проблема, ми відклали період відображення зображення на осі часу. Тепер, мабуть, кожному зрозуміла причина: крім такого FPS необхідно враховувати його стабільність, тобто вводити дані про максимальне відхилення від середнього значення.

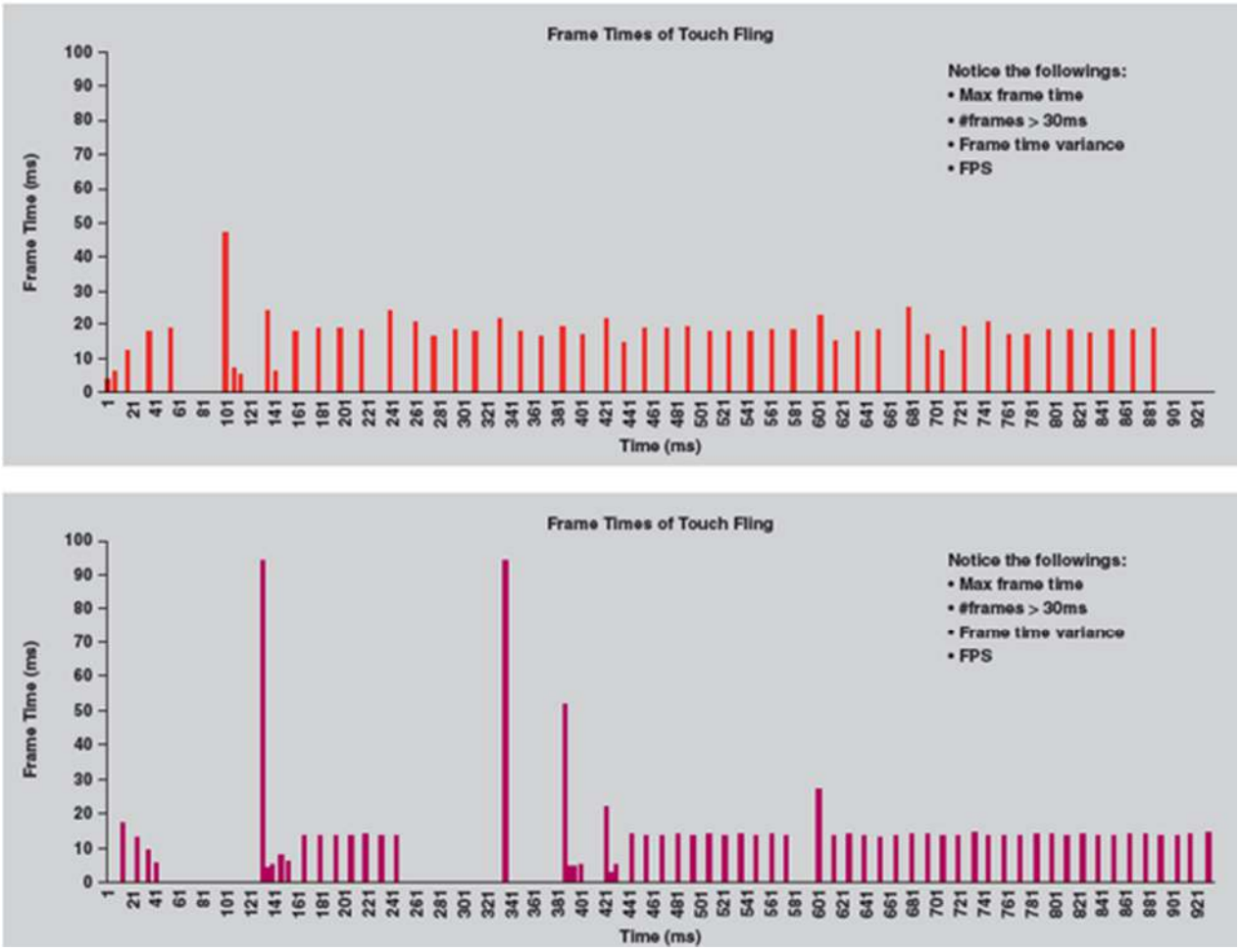

Рисунок 2.2 – Фреймрейти у додатку Gallery3D на пристроях А та Б

Для порівняння ми покажемо графік частоти кадрів пристрою В після оптимізації. Як бачимо, середній FPS практично не змінився, чого не скажеш про відчуття користувачів.

Не можна забувати, що сприйняття користувача — поняття суб'єктивне. Сучасна наука використовує декілька методів моніторингу реакції користувача: відстеження рухів очей, пульсу тощо. При створенні програмного забезпечення ми повинні підходити до проблеми системно, використовуючи всі можливі аналітичні методи. Після розробки метричної системи (наприклад, обговорено чутливість, плавність, логічність і точність) необхідно визначити межі їх зручності для користувача.

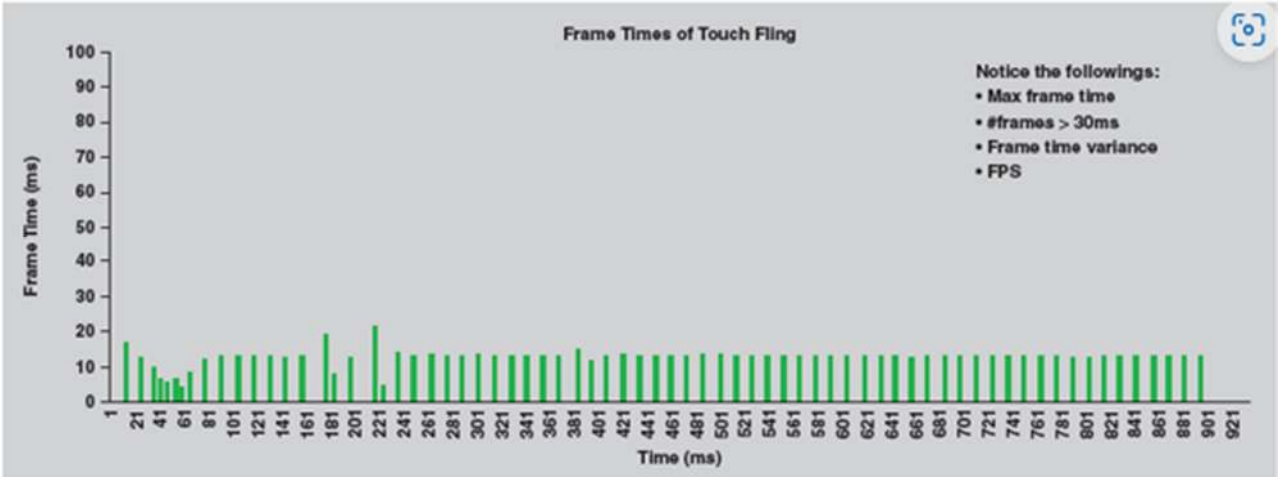

Рисунок 2.3 – Фреймрейт на пристрої Б після оптимізації

3 ІНСТРУМЕНТИ ДЛЯ ТЕСТУВАННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

3.1 Особливості тестування мобільних застосунків

Тестування — це, одна з технологій перевірки якості, воно здійснюється на стадії проектування і надалі, перед випуском програмного продукту, що розробляється. Основне завдання тестування виявити невідповідності в роботі програми, воно так само, є важливим етапом життєвого циклу будь-якого програми, що розробляється. Тестування дозволить виявити невідповідності між очікуваною та реальною поведінкою програми, така перевірка здійснюється на наборі тестів, обраних певним чином. У тестування включені певні процеси такі як: проектування тестів, виконання тестування та аналіз отриманих результатів. Всі ці процеси є невід'ємною частиною всього процесу тестування, де формуються тестові випадки чи сценарії, виконується тестування та аналізується поведінка тестованого додатка.

На рисунку 3.1 також продемонстрована логічна модель виконання тестування.

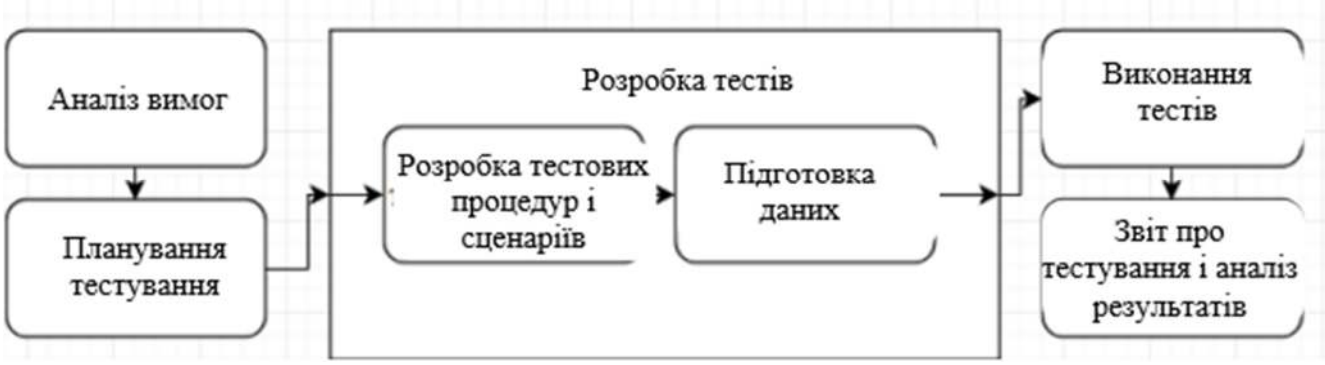

Рисунок 3.1 – Логічна модель тестування

Тестування програм на реальних пристроях дозволяє надати клієнтам

по-справжньому якісні продукти та підвищити ефективність бізнесу у сфері надання послуг.

Варто зауважити, що процес тестування відбувається в рамках послідовності виконання поставлених завдань, вибирається програма або застосунок, який проходитиме тестування, на відповідність вимогам встановлених при розробці та чинних стандартів якості. Після перевірки на критерії відповідності вимогам, відбувається аналіз та збір інформації, внаслідок чого буде отримано інформацію про невідповідності між очікуваним результатом та фактичним.

Тестування програми на мобільному пристрої зазвичай проводиться відповідно до загальних принципів тестування, але є також деякі особливості, характерні для тестування мобільного додатка.

Щоб зрозуміти можливість тестування прикладних програм на мобільних пристроях, необхідно розуміти фактори, що відрізняють мобільні програми від десктопних (настільних): певні операційні системи для мобільних пристроїв, різні конфігурації компонентів, комунікаторів і т.д. необхідно враховувати, наприклад, функціональність пристрою.

Через ці фактори підхід до тестів мобільних програм сильно різниться від настільних. Існує перелік додаткових нюансів та вимог, які необхідно перевірити. Ось головні моменти, на які треба звертати увагу при тестуванні мобільних програм:

а) розмір дисплею та сенсорний інтерфейс:

1) розмір всіх елементів графічного інтерфейсу користувача;

2) перевірка можливостей використання всіх активних елементів (кнопок, посилань і т.д.);

3) швидкість реакції активного елемента повинна бути досить високою;

4) програма не має завершувати роботу в терміновому порядку, навіть якщо кнопка натискається швидко та декілька разів;

5) підтримка мультитач – одночасне натискання декількох кнопок;

6) підтримка горизонтального (horizontal) і вертикального (vertical) позиціонування екрану;

7) підтримка спеціальних жестів, так як вони можуть спростити керування в додатку;

8) підтримка автоматичного повернення екрану.

б) телефонні ресурси:

1) слід контролювати можливість втрати пам'яті. Це часто відбувається в додатках, що мають вікна та містять великий обсяг інформації. Також може статися втрата пам'яті при тривалій роботі програми;

2) обробка ситуацій нестачі пам'яті для роботи операційної системи під час роботи програми в активному і фоновому режимах;

3) недостача місця для установки або роботи з додатком;

4) увімкнення Wi-Fi, 3G і LTE Інтернет без підключення до мережі, наприклад, перевірка заряду батареї пристрою (battery), коли застосунок запущено і працює у фоновому режимі.

в) різні розширення екрану і версії операційної системи:

1) потрібно перевірити роботу програми на пристроях з різною роздільною здатністю екрана. Дисплеї з більш високою роздільною здатністю (наприклад, дисплеї Retina) відображають менше елементів інтерфейсу та тексту, але якщо програма запускається на пристрої з дисплеєм з нижчою роздільною здатністю, елементи інтерфейсу можуть бути занадто великими;

2) потрібно переконатися, що застосунок не може бути встановлено на непідтримуваних пристроях. У той же час тестування програми на всіх заявлених підтримуваних пристроях є обов'язковим.

Користувачі мобільних пристроїв очікують, що додатки, які вони встановлюють, будуть простими, інтуїтивно зрозумілими, завжди функціональними та працюватимуть безперебійно. Якщо очікування не виправдаються, користувачеві доведеться встановити аналогічний застосунок від іншого розробника, чого завжди достатньо тільки в області мобільної розробки. Таким чином, якість програми, орієнтованість на користувача та

простота у використанні всіх функцій є одними з основних факторів її популярності.

3.2 Класифікація інструментів тестування мобільних застосунків

Класифікація інструментів тестування мобільних додатків важлива для вибору правильного набору інструментів при розробці та тестуванні мобільних додатків. Інструменти тестування можна розділити на кілька категорій залежно від їх функцій та характеристик, так як під кожну задачу рекомендується використовувати окремо визначений інструмент. Загальна класифікація інструментів тестування мобільних додатків виглядає наступним чином:

- функціональні інструменти тестування. Appium – це інструмент, який використовується для автоматизації функціонального тестування на різних мобільних платформах, включаючи Android та iOS. Calabash – функціональна тестова платформа, яка дозволяє писати тести в Ruby на платформі Xamarin. Espresso – платформа для автоматичного тестування програм Android за допомогою мови програмування Kotlin або Java. XCUITest – платформа для автоматичного тестування додатків iOS у Swift або Objective-C;

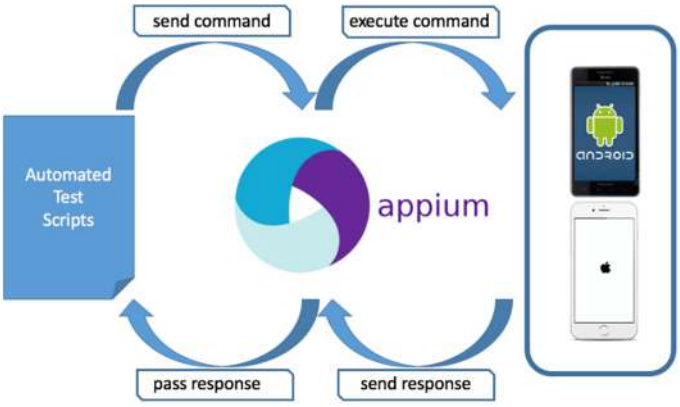

Рисунок 3.2 – Робота Appium

- інструмент візуального тестування. Applitools – автоматизована служба візуального тестування, яка допомагає виявити зміни в інтерфейсі мобільних додатків. SikuliX – інструмент візуального тестування, який дозволяє виконувати дії на основі знімків екрана;

- інструмент тестування навантаження. Apache JMeter – це інструмент тестування навантаження, який можна використовувати для тестування мобільних API та веб-служб. LoadRunner – комплексний інструмент тестування навантаження, який підтримує тестування мобільних додатків;

- засоби автоматизації та контролю якості. Selenium – це популярний інструмент автоматичного тестування веб-додатків, який також можна використовувати для тестування гібридних мобільних додатків. TestComplete це інструмент, який використовується для автоматизації тестів на різних платформах, включаючи мобільні пристрої;

- інструменти тестування безпеки. Mobile Security Framework (MobSF): це інструмент тестування безпеки мобільних додатків з відкритим кодом, який визначає потенційні загрози та вразливості. OWASP ZAP: це інструмент, який використовується для перевірки безпеки веб-додатків, але його також можна використовувати для аналізу трафіку мобільних додатків;

- інструменти для тестування і відновлення відмовостійкості. Crashlytics: послуга у Firebase, яка допомагає виявляти та аналізувати збої програм та надсилати звіти розробникам. HockeyApp: це послуга, яка використовується для моніторингу відмовостійкості додатків та розповсюдження оновлень.

Класифікація інструментів тестування мобільних додатків може бути більш детальною залежно від потреб конкретного проекту та вимог до тестування. Важливо вибрати інструменти, які найкраще відповідають вашим цілям і дозволяють пропонувати високоякісні мобільні додатки.

Також можливо кілька підходів при тестуванні продуктивності програмного забезпечення:

- у терміні робочого навантаження: програмне забезпечення

піддається тестуванню за сценаріями максимально наближеними до реальних умов;

- у рамках бета-тестування, коли програма піддається тестування реальним користувачем.

Тестування навантаження є простою формою тестування продуктивності. Таке тестування зазвичай виконується з метою оцінки поведінки програми при різних варіаціях та ступенях навантаження. Таким навантаженням може бути: велика кількість користувачів, що одночасно працюють з додатком, кількість яких-небудь транзакцій, що відбувається за певний інтервал часу. За допомогою такого типу тестування найчастіше вдається отримати час відгуку основних транзакцій. Якщо ж проводити спостереження за базою даних, сервером програми, мережею і т.п., то за допомогою такого типу тестування можна також виявити вразливі місця в додатку.

Стресове тестування, такий тип використовується в основному, для встановлення меж пропускної спроможності програми. Він також необхідний визначення надійності системи під час екстремальних чи диспропорційних навантажень, і показує результат продуктивності в момент перевищення поточного навантаження очікуваного максимуму.

Тестування стабільності, таке тестування дозволить переконатися, що програма під час тривалої роботи зможе витримати певне навантаження. Під час тестування, увага приділяється споживанню ресурсу пам'яті, додатком, що тестується, це дозволить виявити потенційні витоку. Так само за допомогою такого типу тестування можна виявити якусь деградацію продуктивності, яка буде виражатися у значному зниженні швидкості обробки інформації, а також виявити в процесі тривалої роботи програми, з моменту початку тестування, чи відбувається збільшення часу відповіді, після тривалої роботи з порівняно із початком тесту.

Конфігураційне тестування являє собою ще один тип тестування продуктивності. Під час проведення такого тестування продуктивність

оцінюється не з боку підвищення навантаження, а виявляється ефект впливу на зміни в конфігурації. Конфігураційне тестування, можна поєднувати з іншими типами тестування продуктивності: тестуванням навантаження, стресостійкості та стабільності.

У тестуванні продуктивності є також свої певні принципи та експериментальні факти, які застосовуються при проведенні даного типу тестування. Якщо розглядати загалом, то застосовуються вони при тестуванні продуктивності до будь-якого з типів тестування, зокрема, до тестування навантаження.

Рисунок 3.3 – Підходи в тестуванні

3.3 Інструменти автоматизації тестування мобільних застосунків

У тестах доцільно використовувати ці інтерфейси для взаємодії з додатком. При ручному тестуванні тестер є посередником між тестом і додатком, а текст тестового прикладу зберігається в одному з програмних інтерфейсів для ефективного тестування.

Для автоматизації тестер повинен бути замінений інструментом, який може взаємодіяти з одним або більше програмними інтерфейсами. Для цього також знадобиться утиліта для запуску та створення серії тестів.

Усі ці інструменти спільно називаються автоматичними тестовими стеками. Щоб зрозуміти, як вони взаємодіють у стеку, їх потрібно класифікувати. Представлена класифікація є умовною і в основному необхідна для розуміння інструментів та їх комбінацій.

Всього існує 4 групи інструментів: драйвери, плагіни, фреймворки та комбінації.

3.3.1 Драйвер

Утиліта автоматичного тестування, як і будь-яка інша програма, може взаємодіяти з додатком тільки через програмний інтерфейс – ніщо інше не може бути спеціальною програмою-драйвером для роботи через інші інтерфейси.

Драйвер – це програма, яка надає API одному з інтерфейсів програми. Насправді для кожного інтерфейсу, крім API, потрібен власний драйвер. Наприклад, якщо ви дасте драйверу графічного інтерфейсу команду «натиснути кнопку меню», він розпізнає її через API і відправить до програми, в якій проводиться тестування. Тут ця команда натисне графічну кнопку меню. Для взаємодії з API програми драйвер не потрібен або потрібен дуже рідко. Взаємодія – це програмне забезпечення. Однак, якщо використовуються інші інтерфейси, без них не обійтися.

Найскладнішим, як правило, є драйвер графічного інтерфейсу, оскільки цей інтерфейс сильно відрізняється від звичайного коду, з яким програма взаємодіє. У той же час графічний інтерфейс вважається найбільш актуальним для автоматичного тестування мобільних додатків, тому його часто доводиться використовувати в інтеграційних тестах. Найпопулярнішими драйверами графічного інтерфейсу в мобільному тестуванні є UIAutomator для Android і Espresso, а також xcuitest для iOS.

3.3.2 Надбудова

Якщо функціональності драйверу недостатньо або він незручний і складний, на ньому з'являється інший рівень. Це називається надбудовою.

Надбудова(плагін) – це програма, яка взаємодіє з додатком через один або більше драйверів для підвищення простоти використання або розширення функціональності. Плагін має наступні функції:

- зміна поведінки (без зміни API). Наприклад, додаткове протоколювання, валідація (перевірка достовірності) даних, очікування виконання дії протягом певного часу;

- поліпшити простоту API та/або рівень абстракції за рахунок використання «синтаксичного цукру» — зручні назви функцій, лаконічні їх виклики, єдиний стиль написання тестів. Наприклад, фонове керування драйвером запускається автоматично без необхідності вручну вводити кожну з цих дій. Спростіть складні команди, наприклад вибір подій із календаря або використання розкривних списків. Впроваджуйте альтернативні стилі програмування, такі як процедурний та гнучкий стилі;

- уніфікація API драйверу. Тут плагін надає єдиний інтерфейс для роботи з декількома драйверами одночасно. Це дозволяє використовувати той самий код для тестування на iOS та Android, наприклад, за допомогою звичайного плагіна Appium.

3.3.3 Фреймворк

Фреймворк – це програма, яка генерує, запускає та збирає результати серії тестових запусків

Завдання фреймворку включають:

- створення, групування та замовлення серії тестів;
- регулювання розпаралелювання (необов'язково);
- створення фікстур (автоматичних тестів для багаторазового використання);
- тестування;
- збір результатів виконання;

Ці функції стосуються не лише тестування мобільних додатків, але й успішно використовуються для тестування настільних та веб додатків. На практиці фреймворк не повинен забезпечувати взаємодію між тестуванням і додатком – він працює тільки з тестуванням, а тип програми такий же, як і тип додатку. Це не проблема.

Якщо драйвери та доповнення знаходяться між тестом та додатком, фреймворк ініціалізується. Тому важливо не плутати поняття «привід» і «каркас». Звичайно, в деяких фреймворках є свої драйвери для роботи з додатком, але це зовсім необов'язково. Найбільш помітними фреймворками для мобільного тестування є xUnit і Cucumber.

Фреймворки потрібні для створення як великих проектів, так і простих сайтів та програм, які планується розвивати в майбутньому. Вони дозволяють правильно побудувати бізнес-логіку. В основному фреймворки використовуються для створення калькуляторів, інтернет-магазинів з нестандартним функціоналом, власних CRM, які не можна створити на CMS, десктопних та мобільних програмах. На відміну від CMS, frameworks — це низькорівневе рішення, яке має більшу продуктивність і гнучкість. Дозволяє отримати готовий каркас для проекту без втрати гнучкості щодо функціоналу. На відміну від динамічних бібліотек, які є набором обмежених

функцій, на фреймворку вибудовується архітектура. Він визначає зв'язок між компонентами.

Фреймворк зазвичай складається з безлічі бібліотек, які він використовує в собі, і реалізує за допомогою них структуру/каркас проекту. Також він надає свої бібліотеки на вирішення поширених завдань. Крім того, у документації до фреймворку зазвичай суворо вказано, як використовувати ці бібліотеки, у яких місцях їх викликати та як вирішувати проблеми.

Щоб поділити проект на логічні частини, модулі, фреймворки найчастіше використовують архітектуру Model-View-Controller (MVC). Вона являє собою розподіл на три сегменти: модель, уявлення та контролер. Можна змінювати кожен, незалежно від інших.

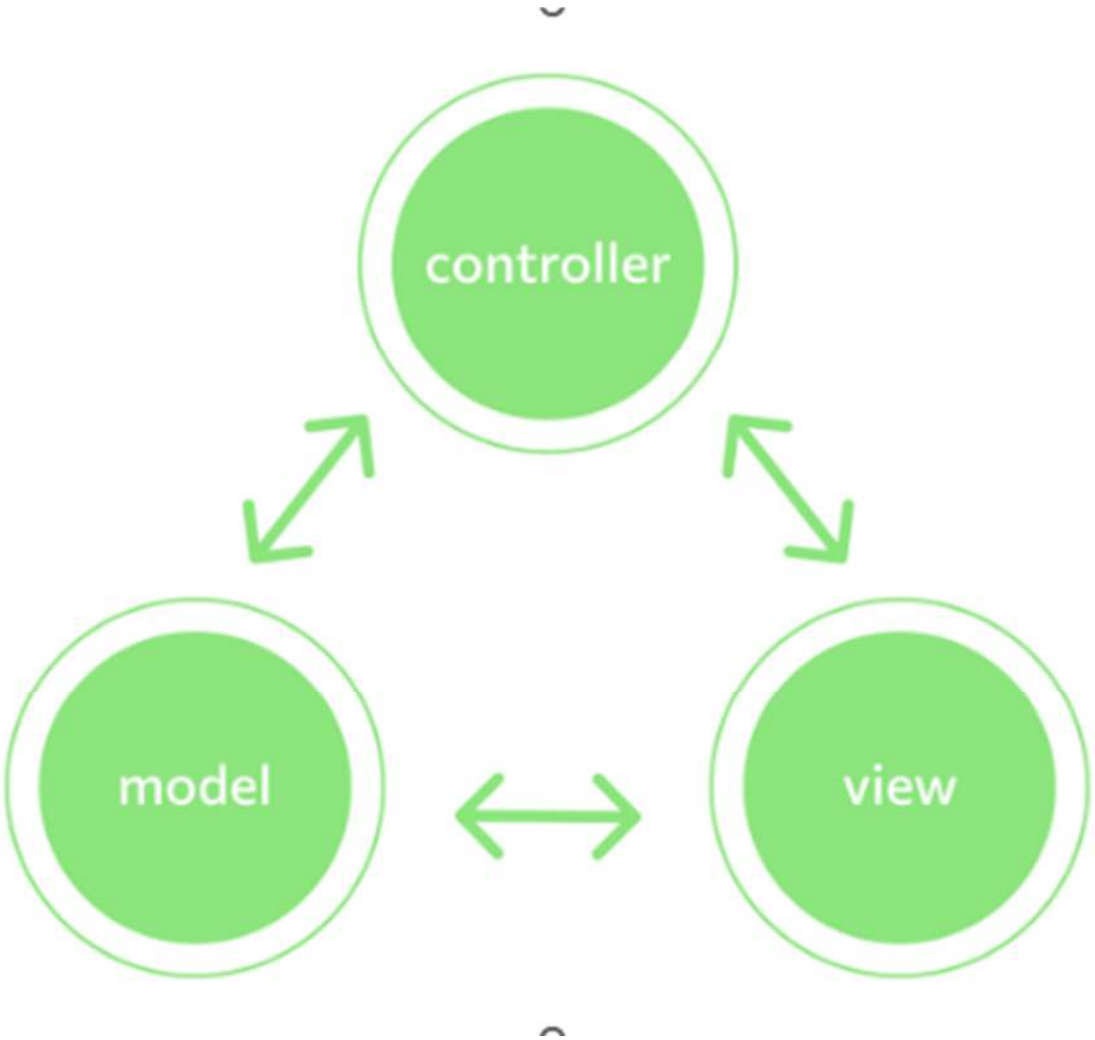

Рисунок 3.4 – Модель MVC

3.3.4 Комбайни

Нарешті, ще одна група утиліт, що використовуються для автоматизації тестування мобільних додатків – це поєднання фреймворків, драйверів (не лише мобільних) і навіть можливостей розробки. UITest, Squish, Ranorex від Xamarin підтримують автоматизацію тестування для iOS, Android та веб-додатків, а останні дві також підтримують настільні програми.

Без сумніву, справжній пристрій – найкраще рішення для тестування мобільних додатків. Коли ви тестуєте його на реальному пристрої, ви завжди отримуєте максимальну точність результатів. Насправді, вибрати найбільш придатний пристрій непросто.

Переваги тестування мобільних додатків на реальних пристроях:

а) висока точність результатів випробувань;

б) проста реплікація помилок (моменти, такі як: ємність акумулятора, геолокація, push-повідомлення і вбудовані датчики пристрою легко підтверджуються);

в) можливість перевірки вхідних переривань (дзвінок, СМС);

г) можливість тестування мобільного додатка в реальних умовах;

ґ) немає помилкових спрацювань.

Недоліки тестування мобільного додатка на реальному пристрої:

а) існує багато часто використовуваних пристроїв;

б) додаткові витрати на технічне обслуговування пристрою.

4 МЕТОДИ ТА ПІДХОДИ ДО ТЕСТУВАННЯ

4.1 Основні підходи до автоматизації тестування

4.1.1 Record-Runscript

Виділяють наступні види тестування:

- тестування на рівні коду;
- тестування користувацького інтерфейсу (GUI-тестування).

Перший тип включає, зокрема, модульні тести. Другий являє собою імітацію дій користувача з використанням спеціального тестового середовища. Найпоширенішою формою автоматизації є тестування програм за допомогою графічного інтерфейсу користувача.

Популярність цього типу тестів пояснюється двома факторами:

а) застосунок тестується так, як його використовують люди;

б) можливість протестувати свою програму без доступу до вихідного коду.

Однією з головних проблем автоматизованого тестування є його складність, хоча це може усунути деякі рутинні операції та пришвидшити виконання тестів. Ви можете витратити багато ресурсів на оновлення самого тесту. Це стосується обох типів автоматизації. При рефакторингу програми зазвичай потрібно оновити модульні тести та змінити тестовий код, тоді як при зміні програми потрібно переписати всі тести, пов'язані з оновленим вікном, та змінити основний код.

Проблема полягає в часі виконання автоматичного тестування. Тести графічного інтерфейсу для великих проектів зазвичай проводяться ввечері, тому результати їх проходження будуть готові до ранку. Це сповільнить випуск нової версії і не дозволить вам швидко виправити помилку, якщо в продукті вона раптово з'явиться.

В наші дні тестування програмного забезпечення проводилося вручну самим програмістом або користувачем, але це не можна назвати систематичним підходом, так як оцінити якість коду неможливо. Через деякий час тестування стало ще однією галуззю знань в рамках розробки програмного забезпечення, але швидко стало очевидним, що ручне тестування неефективне, оскільки вимагає багато робочої сили та багато часу.

Першим інструментом автоматизації тестування була, по суті, бібліотека, яку можна було використовувати для написання тестів, і тестери повинні були мати можливість програмувати на рівні розробника. Сучасні інструменти автоматизованого тестування дозволяють створювати автоматизовані тести з мінімальною участю людини (тестові сценарії створюються автоматично на основі взаємодії з користувачем).

Більшість сучасних фреймворків, призначених для автоматизації тестування, надають користувачеві можливість записати його дії однією зі скриптових мов програмування. Отриманий результат можна відредагувати, зберегти та за необхідності відтворити. Незважаючи на зручність даного методу, що здається, часто згенерований код не тільки важко читаємо, але і зовсім може бути незастосовний для реального тестування. Наприклад, натискання кнопкою може бути записано не через звернення до об'єкта, а як клік по координатах щодо екрана. Відповідно, при переміщенні або зміні розмірів форми програми, що тестується, скрипт видасть помилку.

Використання цього підходу не рекомендується при створенні автоматичних тестів, однак, може бути корисним у разі, коли необхідно в короткі терміни розробити скрипт, що робить велику кількість простих одноманітних дій.

Використання класів у розробці скрипту дозволяє структурувати код, що полегшує його підтримку та розробку надалі, а також спрощує планування та написання тестів. Взаємодія із зовнішнім джерелом даних дає можливість додавати та змінювати тести, не вносячи змін до коду, а отже

кінцевий користувач може працювати зі скриптом без участі розробника. Проста запис скрипта методами фреймворку корисна, коли необхідно в короткий термін реалізувати тестування невеликої частини функціоналу.

4.1.2 Data-Driven Testing

Тести, керовані даними — це методологія, що представляє собою тестування, виконання і верифікація якого виготовляють основі даних, одержуваних із зовнішнього джерела, у ролі якого може бути база даних, таблиця Excel чи будь-який інший упорядкований набір значень. Очевидно, що логіка, яка буде виконана в скрипті, також залежить від даних. Найбільший ефект від використання цього методу спостерігається при тестуванні форм введення, оскільки один скрипт, що управляє, може перевірити практично будь-яку форму з будь-якими даними.

Переваги цього підходу очевидні. Зовнішнє джерело даних дозволяє збільшити кількість тестів шляхом додавання нових значень таблицю без зміни коду. У разі ж, коли необхідно змінити дії тестів у зв'язку із зміною програми, що тестується, правки в скрипти доводиться вносити тільки один раз.

Однак є й недоліки. Зокрема, проектування та розробка скрипту, керованого даними, потребує більше часу, ніж простий запис автоматичного тесту. Також часто скрипт, що вийшов, орієнтований на виконання тільки одного конкретного завдання, внаслідок чого погано схильний до розширення, змін у логіці і рідко може бути використаний для тестування інших завдань.

Коли йдеться про автоматизоване тестування на C#, існує кілька тестових фреймворків, таких як NUnit, xUnit.net та MSTest. Кожен з них має свої переваги та недоліки, але NUnit та xUnit.net все ж мають величезну перевагу перед MSTest. Хоча MSTest є стандартним фреймворком тестування, що постачається з Visual Studio, він не був кращим для

автоматизації тестування до виходу MSTest 2.

MSTest 2 — версія MSTest з відкритим вихідним кодом. Попередня версія MSTest не мала багатьох функцій, які були присутні у інших тестових фреймворках. Однак друга версія підтримує паралельне виконання тестів та тестування на основі даних, а також розширюється за допомогою кастомних атрибутів тестів. Параметризовані тести з MSTest стали можливими у 2-й версії фреймворку з використанням атрибутів/анотацій.

Атрибут [DataRow] дозволяє встановити значення параметра тесту, при цьому можна встановити більше одного атрибуту [DataRow]. Для перетворення звичайного тесту на параметризований достатньо замінити атрибут [TestMethod] на [DataTestMethod] і передати необхідні параметри для тесту на [DataRow] атрибут. У прикладі нижче (Лістинг 4.1) розглянуто абстрактну валідацію числового інпуту, передаючи першим аргументом саме значення, а другим відповідне повідомлення про помилку в тултипі.

Лістинг 4.1 – Абстрактна валідація

```
      [DataTestMethod]
      [DataRow(NumberValues.NotNumberValue, Tooltips.NotNumber)]
      [DataRow(NumberValues.NotInAcceptableRangePositive,
Tooltips.AllowedRange)]
      [DataRow(NumberValues.NotInAcceptableRangeNegative,
Tooltips.AllowedRange)]
      [DataRow(NumberValues.ErrorInExponentialNumberNotation,
Tooltips.ErrorInExponentialNumberNotation)]
      [DataRow(NumberValues.OnlyOneSeparatorIsAllowed,
Tooltips.OnlyOneSeparatorIsAllowed)]
      public void NumberInputValidationTooltipShouldExist(string inputValue,
string tooltipText)
      {
          // Act
          SomeNumberInput
              .EnterValue(inputValue)
              .GetValidatorTooltip(out var currentTooltipText);

          // Assert
          Assert.AreEqual(tooltipText, currentTooltipText);
```

Використання [DataRow] та [DynamicData] атрибутів дозволить створювати параметризовані тести в MSTest 2 для розробки тестів на основі даних. Їх має бути достатньо більшості випадків. Якщо вбудованих атрибутів

недостатньо - вони можуть бути розширені під час створення власних [DataSource] атрибутів для створення джерела даних. Однак якщо стоїть завдання впровадити паралельне виконання Data Driven тестів, то MSTest 2, як і раніше, поступається іншим фреймворкам.

4.1.3 Object-DrivenTesting

Тести, керовані об'єктами — це підхід до автоматизації тестування, у якому скрипти проектуються як класів, реалізують логіку роботи з додатком. Дана методологія дозволяє формувати тести з урахуванням методів високого рівня, у своїй знання деталях реалізації цих методів є необхідними.

Даний підхід передбачає поділ додатка, що тестується, на логічні об'єкти, які будуть описуватися незалежно один від одного. Найчастіше один клас відповідає одній формі, проте за необхідності можливий опис форми кількома класами чи навпаки, опис кількох схожих форм одним класом.

Особливість цієї методології в тому, що такий підхід вимагає досить багато часу для аналізу архітектури додатка, що тестується. Чітке розуміння архітектури дозволяє забезпечити оптимальне поділ функціональності, що реалізується, на класи. Надмірна деталізація призведе до того, що код буде надмірно перевантажений невеликими окремими об'єктами. У зворотній ситуації будуть втрачені всі плюси методології, оскільки клас описуватиме надто велику частину логіки.

4.2 Класифікація засобів та план тестування

4.2.1 Тестування додатків, що надають API

Зовнішній програмний інтерфейс (API) дозволяє програмному забезпечення спілкування з іншими додатками. Як і будь-яке інше програмне забезпечення, інтерфейси мають бути протестовані. Виникає питання, як

можна розробити тести без впливу через графічний інтерфейс? Відповіддю може стати використання програмних інтерфейсів, які дозволяють отримати прямий доступ до тієї функціональності, яку треба протестувати.

SoapUIPro (або QualityLogic) — крос-платформне рішення для тестування веб-сервісів та API, за допомогою SoapUIPro можна швидко створити та запустити автоматизовані функціональні, регресійні та тести безпеки.

Тестування API істотно відрізняється від інших видів тестування, оскільки GUI рідко бере участь і є деякі труднощі, в тому числі і в тому, щоб просто переконатися, що система готова до тестування. Необхідно перевірити функціональність, відповідність вимогам та безпеці.

Під час тестування програми цікавими завданнями для тестувальників є:

- забезпечення тестування різними параметрами API дзвінків таким чином, щоб перевірити функціональність та безпеку. Це включає вивчення граничних умов;
- створення комбінацій значень параметрів для викликів із двома або більше параметрами;
- визначення вмісту API дзвінків. Це може включати створення зовнішніх умов навколишнього середовища (файли, периферійні пристрої тощо), а також внутрішніх даних, що зберігаються, які впливають на API;
- послідовність API викликів для перевірки функціональності.

Порядок тестів має відповідати плану тестування. Кожен тест має бути максимально автономним та ізольованим від залежностей наскільки це можливо. Основним орієнтиром є виявлення найбільш поширених параметрів та умов, які використовуються під час виклику API методів.

API тестування відрізняється від інших видів тестування тим, що GUI рідко бере участь у тестуванні API. Необхідно ініціалізувати тестовий стан: створення умов, за яких API використовуватиметься.

Тестові сценарії для тестування API базуються на вхідних даних:

- значення, що повертається на основі даних умов. Можна надсилати різні комбінації запитів та перевіряти відповідь на відповідність очікуваного результату;

- значення, що повертається, не визначено. Поведінка API в системі повинна бути перевірена коли немає значення, що повертається;

- запуск деяких інших API / подій / винятків. Методи API, якщо викликають певну подію або виняток, повинні бути відстежені відповідними обробниками. Набір тестів має перевіряти виклики API;

- оновлення даних. Оновлення даних матиме певний вплив на систему, тому воно має бути перевірено;

- зміна певних ресурсів. Якщо API виклик змінює деякі ресурси, наприклад, робить оновлення бази даних, то це повинен бути підтверджений доступ до відповідних ресурсів.

Порівняння API тестування із JUnit тестами (модульне тестування):

- API тестування не є модульним тестуванням. Модульним тестуванням найчастіше займаються розробників, тестуванням API — QA команда. Тестування API відноситься до тестування так званої чорної скриньки, коли як модульне тестування по суті відноситься до тестування білої скриньки;

- API-тестування та юніт-тестування орієнтовані на рівень коду. Існує кілька інструментів з відкритим вихідним кодом, доступних для тестування API: Webinject, JUnit, XMLUnit, HttpUnit, ANT і т.д;

- процес API тестування полягає у перевірці методів;

- модульне тестування забезпечує команда розробників, щоб кожен модуль надавав свої Unit тести перед збиранням програмного продукту;

- інша ключова відмінність між API та модульним тестуванням полягає у розробці тесту. Юніт-тести зазвичай призначені для перевірки кожного блоку коду окремо. Unit тестування часто не враховує рівень системи взаємодії різних елементів. API Тестування призначене для рівня функціональності системи, оскільки вона використовуватиметься кінцевим

користувачем. Це означає, що API тестування має бути набагато ширшим, ніж юніт-тестування. Таким чином, тестування API – це спеціальні команди, які викликають функції API з усіма основними варіантами параметром. Тестування API має свої особливості.

Для тесту навантаження використовують JMeter. За допомогою тестів для перевірки функціональності формується сценарій навантаження, яке властиве для веб-сервісів. Зміна параметрів дозволяє відстежити динаміку навантаження під час звернення (дзвінка) користувачем будь-якої функції програми

4.2.2 Рівні завдань тестування

На різних етапах і етапах розробки програмного забезпечення прийнято розділяти тести на рівні завдань і об’єктів:

- функціональні тести загальних підсистем і програмного забезпечення рекомендується проводити окремою групою тестування, не пов'язаною з розробником, для перевірки ступеня виконання функціональних вимог програмного забезпечення;

- модульне тестування частин (компонентів) програмного забезпечення проводиться розробником для перевірки правильності реалізації алгоритму. Модульне тестування штовхає програмістів писати максимально оптимізований код, проводити рефакторинг (спрощення програмного коду без шкоди для його функціональності), адже за допомогою модульного тестування можна легко перевірити працездатність розглянутого компонента. Необхідність відокремлення реалізації від інтерфейсу (через специфіку модульного тестування), це дозволяє мінімізувати системні залежності. Документація Юніт-тестів може бути прикладом «живого документа» кожного класу, тестованого даним способом. Модульне тестування допомагає краще зрозуміти роль кожного класу і натомість усієї програмної системи. Також, при «розробці через тестування», яка активно

використовується в екстремальному програмуванні, модульне тестування є одним з основних інструментів, що дозволяє розробляти модулі відповідно до вимог цього модуля;

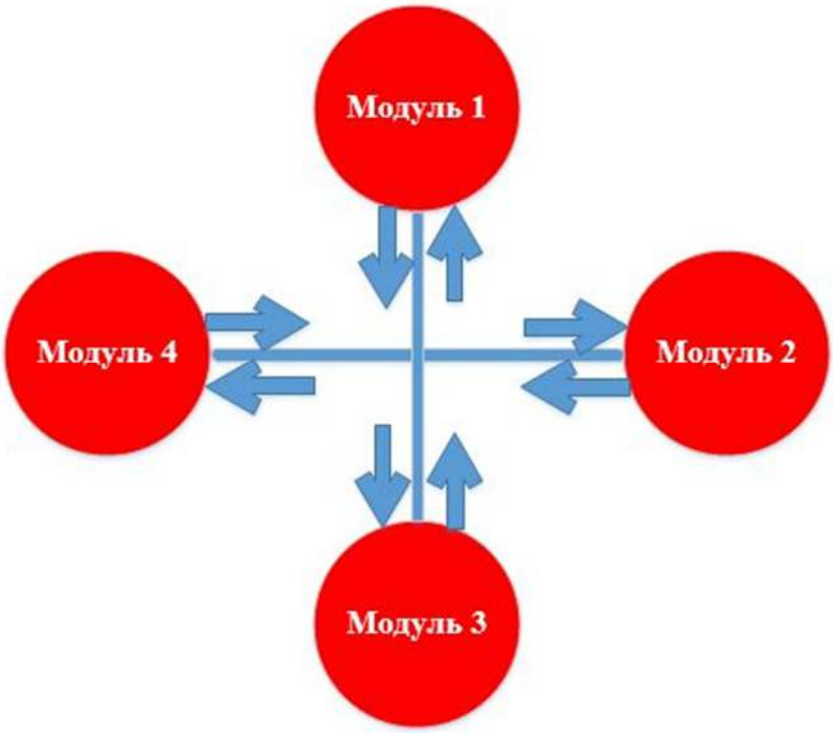

Рисунок 4.1 – Схема модульного тестування

- при зміні навантаження необхідно враховувати особливості функціональності програмного забезпечення (щільність доступу, заповнення бази даних і т.д.). Для виконання навантажувальних тестів (включаючи стрес-тести) потрібне висококваліфіковане випробувальне обладнання та дорогі інструменти для автоматизації експериментів.

Навантажувальне тестування — це один із видів тестування продуктивності, при якому оцінюється робота програмного забезпечення під очікуваним реальним навантаженням. Тестувальники проводять тести навантаження, щоб веб-сайт, мобільний застосунок або API змогли одночасно обробляти конкретну кількість користувачів.

Основна мета навантажувального тестування — виявлення та усунення вузьких місць у роботі програми. Крім того, воно дозволяє забезпечити стабільність та безперебійність роботи рішення перед релізом та початком використання реальними користувачами.

Сьогодні на ринку представлено багато продуктів для інструментів автоматичного тестування. Найпотужнішими з них є HP (QuickTest Professional, WinRunner), IBM (Robot, FunctionalTester), AutomatedQA (TestComplete) і Selenium, які зараз часто використовуються для тестування веб-додатків. Деякі використовують стандартні мови програмування (наприклад, Qtp використовується як мова розробки сценаріїв VB, а функціональний тестер, реалізований у середовищі Eclipse, може писати сценарії на Java).

Більшість інструментів орієнтовані на роботу з веб-додатками або звичайними програмами, створеними за допомогою технологій .Net або Java. Розробники інструментів автоматизованого функціонального тестування дуже швидко реагують на появу нових механізмів і платформ для розробки програмного забезпечення. На ринку існує безліч автоматизованих тестових продуктів, але деякі компанії, що займаються розробкою програмного забезпечення, створюють власні інструменти, придатні для тестування програм, які вони розробляють. Це пов'язано з високою вартістю інструментів автоматичного тестування, унікальністю програмного забезпечення, що тестується, та недоступністю стандартних інструментів автоматизації тестування. На застосунок до інструментів тестування існують також інструменти підтримки процесу тестування, які дозволяють вести облік вимог і тестових прикладів, аналізувати обсяг вимог в тестуванні, управляти ходом тестування і вести облік виявлених дефектів. Веб-застосунок HP Quality Center є лідером у цій галузі та є інструментом управління процесами тестування, включаючи управління вимогами та управління помилками та інструментами тестування навантаження.

Важливим моментом при створенні тестової системи і створенні тесту є

доступність плану тестування, як це визначено в міжнародному стандарті IEEE829-1983.

В ньому повинні бути наступні розділи:

а) що буде тестуватися (тестові вимоги, варіанти тестів);

б) якими методами буде тестуватися система та критерії, за якими буде визначатися успіх тестів;

в) план робіт та ресурси (тестувальники, техніка).

Потрібно визначити конкретні критерії, такі як успішне / невдале завершення тестів, плани управління ризиками та ризиками, а також структурування середовища та управління. Якщо ви чітко розумієте, що вам слід тестувати, чому вам слід тестувати, і у вас є план дій, настав час подумати про те, як стати більш ефективним, швидким і кращим. Сучасне програмне забезпечення – це складний об'єкт, з яким складно і дорого поводитися вручну. Крім того, при «ручному» тестуванні результати кожного тестового прогону втрачаються, і їх важко повторити. Щоб збільшити кількість перевірок та покращити якість тестів, вам потрібно створити базу даних тестів, щоб тести можна було повторно використовувати під час внесення змін до програмного забезпечення.

Як і при тестуванні будь-якого програмного забезпечення, в тестуванні мобільних додатків використовується функціональне тестування, тобто перевірка того, що функціонал, закладений у застосунок, працює відповідно до функціональних вимог. Також можлива перевірка підтримки горизонтального (landscape) та вертикального (portrait) положень. При мобільному тестуванні необхідно враховувати те, що мобільні пристрої – це в першу чергу пристрої для зв'язку, вони можуть дзвонити, надсилати SMS, відключатися. І це не повинно заважати роботі, додатку та навпаки.

Тестування установки мобільного додатка є важливою частиною комплексного тестування програми. Необхідно тестувати чисту установку з нуля, установку поверх попередньої версії, переривання та скасування установки програми. Також важливо тестувати оновлення: перевірка різних

шляхів встановлення оновлень, перевірка роботи встановлених змін, місць, куди вони вносилися; переконатися в підтримці оновлень старішими операційними системами, щоб елементи, які на новій системі працюють добре, не падали на старіших версіях.

Тестування локалізації аналогічне до тестування на веб-додатках. Перевірка локалізації полягає у перевірці інтерфейсу іншою мовою на екрані, для перекладу має вистачити місця для тексту, дати повинні відповідати формату встановленого регіону, тимчасові налаштування мають бути дотримані.

При тестуванні юзабіліті на мобільних пристроях використовуються ті самі засоби та методи, що й під час тестування веб-додатків. При цьому необхідно враховувати розміри екрану та touch-інтерфейс, в якому має бути зручний розмір кнопок, щоб не треба було шукати її на екрані та робити багато спроб потрапити по ній, і висока швидкість відгуку елементів (натиснена клавіша повинна візуально відрізнятися).

В ході тестування безпеки застосунок перевіряється на наявність вразливостей, стійкість до злому, можливості перехоплення трафіку з метою отримання несанкціонованого доступу до інформації, що передається додатком. В даний час даний вид тестування дуже важливий для програм, таких як мобільний банк, страхові програми або месенджери. Співробітники, які займаються тестуванням безпеки, повинні мати специфічні знання та навички, які відрізняються від навичок, які мають фахівці з інших видів тестування.

Випадкове тестування (fuzzy testing, «monkey» testing) – тестування, у якому застосунок має коректно реагувати виникнення випадкових і непередбачуваних подій. Мобільні пристрої найчастіше потрапляють в умови, в яких отримують хаотичну марну інформацію (наприклад, незаблокований девайс у кишені), тому програма має адекватно реагувати на подібні потоки даних.

У мобільній автоматизації використовувати реальні пристрої – дорого і

не завжди ефективно. Віддалені ферми пристроїв, як Browserstack, коштують досить дорого. Підтримка локальної ферми коштує ще дорожче – адміністрування парку пристроїв забирає багато часу. У такій ситуації рятує тестування на емуляторах та симуляторах. Їх встановлення та налаштування не займає багато часу, здебільшого вони безкоштовні, а утиліти автоматизації давно вміють підключатися до них самостійно.

Сьогодні Україна, на жаль, відстає від решти комп'ютерного світу щодо застосування засобів автоматизації тестування, і для цього є декілька причин:

а) неправильне ставлення до такого тестування – багато менеджерів вважають, що розробники можуть писати програми без помилок;

б) висока вартість інструментів автоматизації тестування;

в) бажання заощадити на кваліфікованому персоналі – робота фахівця з тестування засобів автоматизації обходиться дорожче, ніж робота звичайного тестера;

г) обмеження по термінах – автоматизація тестування вимагає значних тимчасових витрат і має сенс тільки в тому випадку, якщо проект як мінімум середньостроковий;

ґ) невдалий досвід використання такого інструменту і очікування негайного впливу від його застосування.

Зрештою, результати впровадження таких продуктів помітні не відразу, а витрати на автоматизацію окупаються протягом тривалого часу, і від ручного тестування не можна повністю відмовитися.

З вищевикладеного було обрано наступні інструменти для розробки автоматизованого тестування і прискорення тестування:

а) Java – мова програмування для написання тестів;

б) Appium – драйвер для запуску тестів;

в) TestNG – тестовий фреймворк;

г) Allure – фреймворк для генерації результатів автоматизованого тестування.

Прискорення автоматичного виконання тестів, що досягається за допомогою мови програмування Java, використання багатопотокового виконання тестів, що досягається за рахунок попередньої ініціалізації всіх графічних елементів і використання поділу кроків під час виконання тестових сценаріїв (для цього використовується середовище TestNG). На даному етапі це значно поліпшує процес тестування додатку.

4.3 Java

Мова програмування Java була створена в 1991 році в лабораторіях компанії Sun Microsystems Inc. Цікаво, що каталізатором появи Java не був Інтернет. Інтернет не був рушійною силою для розвитку Java. Основною мотивацією була потреба в незалежній від платформи (тобто архітектури) мові програмування. Існувала потреба в мові програмування, яку можна було б використовувати для створення програмного забезпечення, яке можна було б вбудовувати в різні пристрої побутової електроніки, наприклад, мобільний зв'язок та інші. Розробка першої робочої версії зайняла 18 місяців і отримала назву Oak. Незважаючи на те, що проект називався Oak, у 1995 році його було перейменовано на Java.

Роки становлення Java збіглися з розквітом Всесвітньої павутини, міжнародної інформаційної служби. Цей факт зіграв важливу роль у майбутньому Java. Майбутнє Java вимагало додатків, які також можна було б перенести в Інтернет. Як результат, фокус розвитку S Java змістився з побутової електроніки на програмування для Інтернету.

Java була мовою за замовчуванням для написання додатків Android з моменту появи платформи Android у 2008 році. Java — це об’єктно-орієнтована мова програмування, спочатку розроблена компанією Sun Microsystems у 1995 році (зараз належить Oracle). Вона була дуже популярною як чиста об’єктно-орієнтована мова (порівняно з C++) і швидко була прийнята платформою Android.

Java компілюється в «байт-код», який інтерпретується під час виконання основною віртуальною машиною Java (JVM), що працює в операційній системі. Критики Java кажуть, що для виконання простого завдання потрібно багато «шаблонного» коду, і що такі концепції важко зрозуміти. Зараз це мова, яка найчастіше використовується для розробки додатків для Android.

І все ж, за даними Google, станом на 2019 рік Kotlin є найкращою мовою для розробки Android.

Kotlin є однією з кількох мов віртуальної машини Java (JVM), розроблених JetBrains, творцями IntelliJ IDEA. У минулому розробникам було важко швидко оновити Java, оскільки через зворотну сумісність було майже неможливо внести значні зміни в синтаксис. Хоча розробники JetBrains створили Kotlin у 2010 році, на випуск версії 1.0 Kotlin знадобилося більше 5 років. Через рік, у 2017 році, на конференції Google I/O Kotlin було оголошено офіційною мовою розробки Android.

Kotlin було представлено для задоволення потреби у коротшій мові, сумісній з Java та JavaScript. Замість того, щоб зберегти його як внутрішній проект, JetBrains вирішила зробити його відкритим кодом у 2012 році. Відтоді розробники сприяли стрімкому злету Kotlin у світі розробки програм.

Java вимагає детального кодування; з іншого боку, компілятор Kotlin може зрозуміти код і легко написати велику кількість коду. Підраховано, що 25 розробників можуть написати на 20% менше коду в Kotlin, значно зменшуючи шаблонний код. У результаті розробники можуть витрачати більше часу на цікаві частини процесу кодування, дозволяючи Kotlin самостійно обробляти звичайний код.

Kotlin і Java — дві схожі мови, які, на щастя, можуть йти рука об руку. Kotlin на 100% сумісний з Java, тобто розробники можуть писати будь-який відсоток свого коду на Kotlin, а решту на Java. Якщо розробник має код Java і Kotlin в одному проекті, він може правильно скомпілювати.

Kotlin може легко використовувати всі існуючі фреймворки та

бібліотеки Java. Kotlin — це проста в інтеграції мова, яка часто інтегрується з Gradle, Maven та іншими системами збірки. Наразі Kotlin є другою за популярністю мовою JVM, і її популярність зростає, оскільки вона розроблена за ліцензією з відкритим кодом і проста у використанні. Згідно з індексом популярності мов програмування PYPL, Kotlin входить до 15 найкращих мов програмування у світі станом на вересень 2023 року. Багато Java розробників вже переходять на написання коду мовою Kotlin.

4.3.1 Безпечність

Всесвітня павутина вивела Java на передній план у програмуванні. Java значно вплинула і навіть змінила обличчя Інтернету, розширивши спектр об'єктів, які можна розгортати в кіберпросторі. З’явилися нові форми програмування – аплети. Аплети завантажуються з віддалених серверів і виконуються динамічно, До появи Java такий підхід був неприйнятним з міркувань безпеки та переносимості. Архітектура аплетів має ряд штучних обмежень, які роблять її абсолютно небезпечною. Перш за все, Java є інтерпретованою мовою. Ресурсний простір Java-додатку обмежений так званою віртуальною машиною Java (VJM). Ця машина контролює поведінку програми і захищає систему від побічних ефектів, спричинених помилками програми. Крім того, мова Java має обмеження для того, щоб додатки не ставали «троянськими конями». Зокрема. Java-аплети не можуть отримати доступ до локального жорсткого диска. Якщо вони намагаються отримати до нього доступ, то генерується виключення.

Мова Java є однією з наймолодших мов у сімействі мов програмування і була розроблена з розрахунком на те, що професійні програмісти зможуть досягти продуктивності, порівнянної з кодом на C++. Вона була розроблена з розрахунком на те, що професійні програмісти зможуть легко опанувати її та ефективно використовувати.

4.3.2 Ефективність

Java-додатки набагато простіше запускати на різних платформах, оскільки вони інтерпретуються, а не компілюються. У цьому випадку для кожної платформи необхідно створити середовище виконання Java. Якщо така система доступна, то програму на Java можна запустити в цьому середовищі без додаткової компіляції. Однак, Java не є інтерпретованою мовою в строгому розумінні цього слова; програми на Java компілюються. Результатом роботи компілятора Java є байт-код. Байт-код — оптимізований набір команд, призначений для виконання віртуальною машиною Java. Таким чином мінімізуються витрати на інтерпретацію. Оскільки байт-код вже оптимізовано, витрати на інтерпретацію зводяться до мінімуму, а продуктивність Java-програми значно підвищується. Перераховані вище функції забезпечують основу для трактування Java як окремої інформаційної технології, а не як окремої мови програмування. Отже, інтерпретація означає, що програма, реалізована за допомогою технології Java є найпростішим способом передачі програми, реалізованої в технології Java. Хоча мова Java була розроблена з урахуванням можливості інтерпретації, технічно ніщо не заважає компілювати байт-код у виконуваний код. Динамічний байт-код застосовується до байт-коду, що передається мережею, підлягає динамічній компіляції, але це не впливає на переносимість або безпеку. Цей підхід використовується у багатьох системах виконання. Java забезпечує продуктивність на рівні оптимізованого коду C++.

Розроблена з розрахунком на ефективне використання, мова Java базується на синтаксисі C++. Це одна з найпопулярніших мов програмування сучасності.

ВИСНОВКИ

Методи тестування мобільних застосунків є важливим етапом у процесі розробки мобільних додатків. В даному дослідженні було розглянуто різноманітні підходи та методи, які дозволяють забезпечити якість та надійність програмних продуктів для мобільних пристроїв.

Методи тестування мобільних додатків є комплексними підходами до перевірки функціональності, продуктивності та користувальницького досвіду при використанні додатків на мобільних пристроях. Основні методи включають ручне тестування, автоматизоване тестування, тестування сумісності, тестування продуктивності і тестування безпеки. Ручне тестування дозволяє виявити непередбачену поведінку програми, тоді як автоматизоване тестування прискорює процес і підвищує повторюваність тестів. Тестування сумісності забезпечує працездатність програми на різних пристроях та платформах. Тестування продуктивності та безпеки необхідно для забезпечення ефективності застосування та захисту від загроз. Комплексне використання цих методів допомагає розробникам створювати стабільні та задовільні мобільні програми.

По-перше, функціональне тестування є основним методом для перевірки коректності роботи додатків та їх відповідності функціональним вимогам. Наявність різноманітних платформ і операційних систем у мобільних пристроях вимагає уважного тестування на різних пристроях та браузерах.

По-друге, автоматизоване тестування дозволяє підвищити швидкість і точність процесу тестування. Воно особливо корисне в умовах швидкого реліз-циклу розробки мобільних додатків.

По-третє, тестування безпеки є надзвичайно важливим аспектом, оскільки мобільні додатки можуть містити конфіденційну інформацію користувачів. Методи тестування безпеки включають аналіз вразливостей та

тестування на проникнення.

Усі ці методи і підходи до тестування мобільних застосунків мають свої переваги та недоліки, і їх вибір залежить від конкретних вимог та умов розробки. Важливо пам'ятати, що якісне тестування сприяє поліпшенню досвіду користувача та довіри до мобільних додатків, що є ключовим фактором для їх успішного впровадження на ринку.

Виходячи із всього вище вказаного, кращим вибором для тестування мобільних застосунків буде не використання якогось конкретного методу, а об'єднання всіх методів в один. В такому випадку тестування буде проходити точніше та якісніше.Якщо ми будемо використовувати окремо якийсь конкретний метод, то завжди буде вірогідність пропустити якусь помилку, а при об'єднанні всіх методів дана вірогідність близька до нуля.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004

2. Android 2. Програмування додатків для планшетних комп'ютерів та смартфонів (Рето Майер, Эксмо, 2011)

3. Android Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/index.html

4. Хашимі, С. Розробка додатків для Android / С. Хашимі. - М.: Біном, 2011. - 2125 с

5. Липа, В. Надійність програмних засобів. - М .: Сінтег, 1998. - 358 с.

6. Різниця між тестуванням мобільних додатків та веб-додатків [Електронний ресурс] – Режим доступу до ресурсу: https://artjoker.ua/blog/raznitsa-mezhdu-testirovaniem-mobilnykh-prilozheniy-i-veb-prilozheniy/