МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Використання технології Appium для тестування мобільного застосунку**

Виконав: студент 4 курсу, групи 6.1219 – пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
М. О. Бойко
(ініціали та прізвище)
Керівник к.ф.-м.н. доцент, доцент кафедри ЕІС та ПЗ
Г.П. Коломоєць
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т. В. Критська
"_01_" __березня_____2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бойка Микола Олександровича
(прізвище, ім'я, по батькові)

1.Тема роботи Використання технології Appium для тестування мобільного застосунку
керівник роботи Коломоєць Геннадій Павлович, к.ф.-м.н. доцент_
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 29.12.2022 1893-с

2. Строк подання студентом кваліфікаційної роботи_____14.06.2023_____
3. Вихідні дані бакалаврської роботи:
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження технології Appium та його функціоналу;
- огляд емулятора Android та застосунку, який буде тестуватись;
- створення тестів та їх опис.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.03 – 10.03.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.03 – 12.03.23	виконано
3	Аналіз існуючих методів рішення	13.03 – 14.03.23	виконано
4	Дослідження засобів тестування Android застосунків	15.03 – 20.03.23	виконано
5	Дослідження емулятора Android та його функціональних можливостей	21.03 – 26.03.23	виконано
6	Вибір мобільного застосунку, який буде тестуватись	27.03 – 28.03.23	виконано
7	Огляд технології Appium, дослідження архітектури та функціональних можливостей	29.03 – 13.04.23	виконано
8	Розгортання оточення розробки для виконання тестів	14.04 – 16.04.23	виконано
9	Опис функціоналу мобільного застосунку	17.04 – 19.04.23	виконано
10	Розробка тестових сценаріїв	20.04 – 01.05.23	виконано
11	Запис тестових сценаріїв та експортування їх до JUnit 5 тестів	02.05 – 7.05.23	виконано
12	Рефакторинг JUnit 5 тестів	8.05 – 12.05.23	виконано
13	Оформлення звіту	13.05 – 20.05.23	виконано

Студент _____ М.О. Бойко
(підпис) (прізвище та ініціали)

Керівник роботи _____ Г.П. Коломоєць
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 89.

Рисунків – 45.

Джерел – 16.

Таблиць – 3.

Бойко М. О. Використання технології Appium для тестування мобільного застосунку: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Г. П. Коломоєць. Запоріжжя : ЗНУ, 2023. 89 с.

Для сучасних мобільних застосунків якість і надійність є пріоритетними, тому їх тестування стає необхідною складовою процесу розробки програмного забезпечення. У рамках даної роботи проведено аналіз основних підходів до тестування мобільних застосунків. В результаті проведеного дослідження було виявлено, що технологія Appium є одним з найбільш ефективних інструментів для автоматизації тестування мобільних застосунків.

Основна мета роботи полягає у дослідженні можливостей та переваг, які надає Appium, а також визначенні ефективності даної технології при розробці тестів для Android застосунків.

Для тестування був обраний застосунок Rozetka – це застосунок для мобільних пристроїв, який дозволяє замовляти різноманітні товари. Розроблено тести для мобільного застосунку, які тестують такі функціональності: реєстрацію користувача, пошук та замовлення товару, роботу фільтрів, сортування та коректність пошуку товару за категорією.

Ключові слова: *Appium, Android, iOS, Selenium WebDriver, JUnit, мобільний застосунок, тестування, емулятор*.

ABSTRACT

Pages – 89.

Figures – 45.

Sources – 16.

Tables – 3.

Boiko M. O. Using Appium technology for testing a mobile application: bachelor's thesis in speciality 121 "Software Engineering" / supervisor H. P. Kolomoyets. Zaporizhzhia: ZNU, 2023. 89 p.

For modern mobile applications, quality and reliability are a priority, so their testing becomes a necessary component of the software development process. This paper analyses the main approaches to testing mobile applications. As a result of the study, it was found that Appium technology is one of the most effective tools for automating mobile application testing.

The main purpose of the study is to investigate the capabilities and advantages provided by Appium, as well as to determine the effectiveness of this technology in developing tests for Android applications.

The Rozetka application was chosen for testing, which is an application for mobile devices that allows you to order various goods. Tests for the mobile application have been developed that test the following functionalities: user registration, product search and ordering, filtering, sorting, and correctness of product search by category.

Keywords: *Appium, Android, iOS, Selenium WebDriver, JUnit, mobile application, testing, emulator*.

ЗМІСТ

ВСТУП

Актуальність теми

В сучасному світі мобільні застосунки є необхідною частиною бізнесу та життя кожного користувача смартфона. Мобільні застосунки мають різноманітні функції від комунікації до покупок, тому вони повинні працювати стабільно та без помилок. З цього приводу, важливим є забезпечення правильного функціонування застосунків на різних платформах та пристроях, так як якість їх роботи має прямий вплив на задоволеність користувачів. Тестування мобільних застосунків допомагає виявляти помилки та забезпечувати якість роботи застосунку, також виявляти проблеми, пов'язані з сумісністю застосунків з різними версіями операційних систем, розмірами екранів та іншими факторами, що можуть впливати на роботу застосунку. Тому дослідження технологій тестування мобільних застосунків є актуальною та перспективною темою для тестувальників та інженерів з розробки програмного забезпечення.

Одним з найпопулярніших інструментів автоматизованого тестування мобільних застосунків є Appium [1]. Appium є відкритим фреймворком, призначеним для автоматизованого тестування мобільних застосунків на різних платформах, таких як iOS та Android. Цей інструмент надає можливість розробникам та тестувальникам ефективно перевіряти функціональність, стійкість та взаємодію мобільного застосунку з операційною системою пристрою.

Використання Appium спрощує процес тестування, оскільки Appium підтримує багатий вибір мов програмування та тестових фреймворків. Це забезпечується використанням фреймворку Selenium WebDriver, для якого існують прив’язки до багатьох мов програмування та тестових фреймворків, що дає можливість працювати тестувальникам та розробникам у відповідності до їхніх навичок та вподобань.

Зважаючи на постійний розвиток мобільних технологій та стрімке зростання ринку мобільних застосунків, використання технології Appium для

тестування стає необхідним елементом у розробці програмного забезпечення. Завдяки своїм можливостям, цей фреймворк дозволяє забезпечити високу якість та надійність мобільних застосунків, а також прискорити процес їх розробки та випуску на ринок.

Отже, дослідження та використання технології Appium для тестування мобільних застосунків є актуальною темою для тестувальників та розробників програмного забезпечення.

Мета дослідження

Дослідження можливостей та переваг, які надає Appium, а також визначення ефективності даної технології при розробці тестів для Android застосунків.

Завдання дослідження

Аналіз існуючих методів тестування мобільних застосунків. Вивчення Appium, визначення мобільного застосунку, який буде тестуватись, проектування тестів, розробка тестових сценаріїв, виконання тестів та аналіз результатів.

Об’єкт дослідження

Об’єктом дослідження є мобільний застосунок Rozetka – український онлайн – магазин, що спеціалізується на продажу різноманітних товарів, від електроніки та побутової техніки до моди, косметики та інших товарів. Цей мобільний застосунок, доступний для смартфонів і планшетів, дозволяє користувачам зручно та швидко здійснювати покупки з будь – якого місця за допомогою мобільного пристрою.

Предмет дослідження

Предметом дослідження є технологія Appium, дослідження ефективності та переваг використання цієї технології для тестування мобільного застосунку.

Методи дослідження

Теоретичні – аналіз і порівняння технологій тестування мобільних застосунків; порівняльний аналіз для вибору програмного забезпечення. Проведення експериментального дослідження, яке включає налаштування середовища для тестування з використанням технології Appium, вибір мобільного застосунку для тестування, розробку тестових сценаріїв для мобільного застосунку, виконання тестових сценаріїв та збір результатів тестування.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, щоб визначити, наскільки ефективно можна використовувати технологію Appium для тестування мобільних застосунків. Розроблені тестові скрипти, їх аналіз та результати демонструють, які конкретні переваги та можливості можуть бути використані розробниками та тестувальниками для автоматизованого тестування.

Апробація результатів

Робота була апробована на конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» / Запорізького національного університету [16].

Глосарій

Тестування — це процес перевірки програмного забезпечення з метою виявлення помилок та недоліків.

Appium — це інструмент мобільної автоматизації з відкритим кодом, який забезпечує автоматизацію на таких платформах, як Android та iOS.

Android — це операційна система та платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux.

iOS — мобільна операційна система, розроблена компанією Apple для використання на її мобільних пристроях.

Мобільний застосунок — це програмне забезпечення, призначене для р оботи на смартфонах, планшетах та інших мобільних пристроях.

Емулятор — програмне забезпечення, яке дозволяє емулювати роботу операційної системи на комп'ютері.

Selenium WebDriver — бібліотека для автоматизації веб-застосунків. Вона дозволяє розробникам тестувати веб-застосунки, взаємодіючи з ними на рівні браузера.

JUnit – фреймворк для автоматизованого тестування програмного забезпечення у мові Java.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

В наші дні IT – технології сягнули значного прогресу, включаючи мобільні розробки. Мобільні застосунки стали необхідною складовою нашого повсякденного життя, використовуються для різних цілей, таких як покупки, спілкування, ігри, розваги, робота і багато іншого, 87% користувачів смартфонів витрачають свій час саме у мобільних застосунках [2]. Кількість завантажень мобільних застосунків складала більше 200 мільярдів [2]. Використання мобільних програм зростає, а застосунки стають одним із найкращих способів комунікації з клієнтами.

Використання мобільних застосунків зросло з кількох причин:

1. Зручність: Мобільні програми надають зручний спосіб доступу до різних послуг та інформації прямо зі смартфонів. Вони дозволяють клієнтам отримувати доступ до продуктів і послуг в будь – який зручний для них час і місце. Немає потреби шукати комп'ютер або відвідувати фізичний магазин.
2. Зростання популярності смартфонів: Смартфони стали все більш поширеними і доступними для широкої аудиторії. Люди використовують свої смартфони для різних потреб, включаючи комунікацію, розваги та виконання різних завдань. Мобільні програми стали неодмінною складовою смартфонного досвіду.
3. Покращена швидкість та доступ до Інтернету: Зростання швидкості мобільного Інтернету дало змогу клієнтам зручно користуватися мобільними застосунками. Швидкість передачі даних та стабільність з'єднання роблять використання мобільних програм більш ефективним і приємним для користувачів.
4. Розширений функціонал: Мобільні програми надають широкий спектр функцій і можливостей. Вони можуть включати функції сповіщень,

персоналізацію, онлайн – платежі, GPS – навігацію та багато іншого. Це робить їх більш привабливими для користувачів і стимулює зростання використання.

5. Пандемія коронавірусу: Обмеження, які були пов'язані з пандемією, вимагали від людей зменшення фізичних контактів і переходу до безконтактних та онлайн – сервісів. Мобільні застосунки стали особливо актуальними для замовлення продуктів, отримання медичних консультацій, виконання фінансових операцій та багатьох інших потреб у період пандемії.

   Для огляду статистики кількості користувачів Інтернет та мобільних застосунків за 2022 – 2023 рік, можна розглядати глобальний огляд Digital 2023 Report , створений у партнерстві з Meltwater і We Are Social [3].

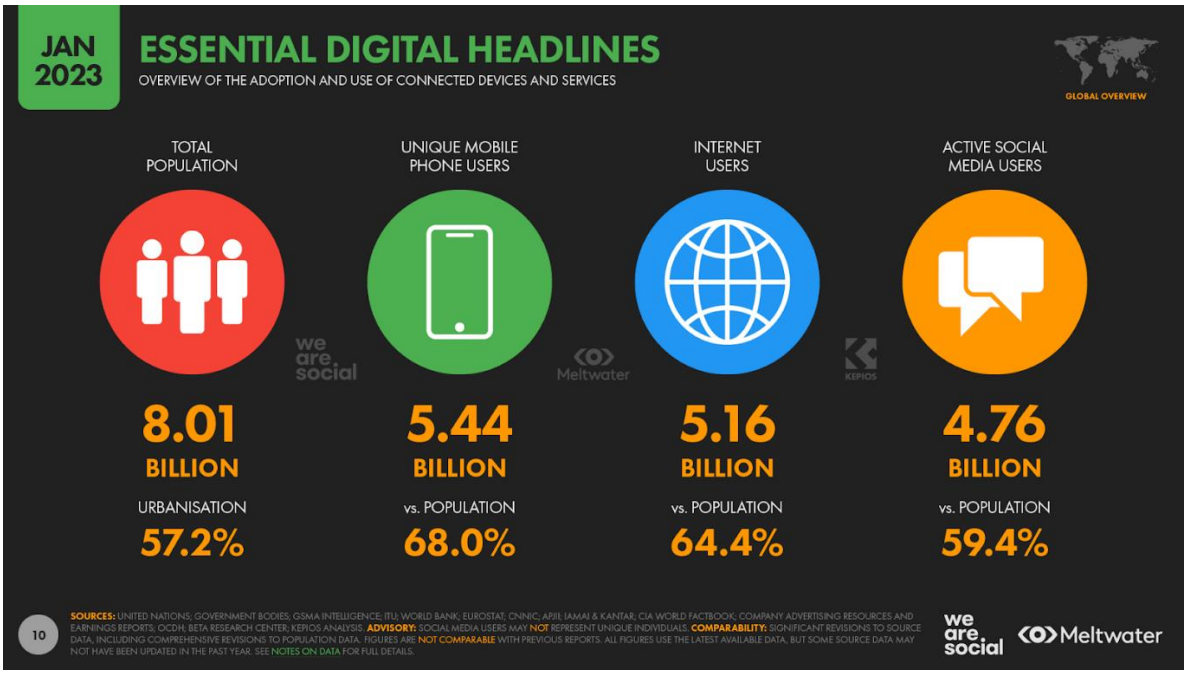

Рис. *1 Огляд впровадження та використання підключених пристроїв і послуг*

Як ми можемо побачити на рисунку 1 населення світу перевищило 8 мільярдів та досягло 8,01 мільярда на початку 2023 року. Зараз у містах проживає трохи більше 57 відсотків населення світу. Загалом на початок 2023 року мобільними пристроями скористалися 5,44 мільярда людей, що становить 68

відсотків від загального населення світу. За останні 12 місяців з'явилося 168 мільйонів нових користувачів. На початок 2023 року в світі налічується 5,16 мільярда користувачів Інтернету, що становить 64,4 відсотка від загального населення світу. За останні 12 місяців загальна кількість користувачів Інтернету зросла на 1,9 відсотка. Зараз у світі налічується 4,76 мільярда користувачів соціальних мереж, що становить 59,4 відсотка населення світу.

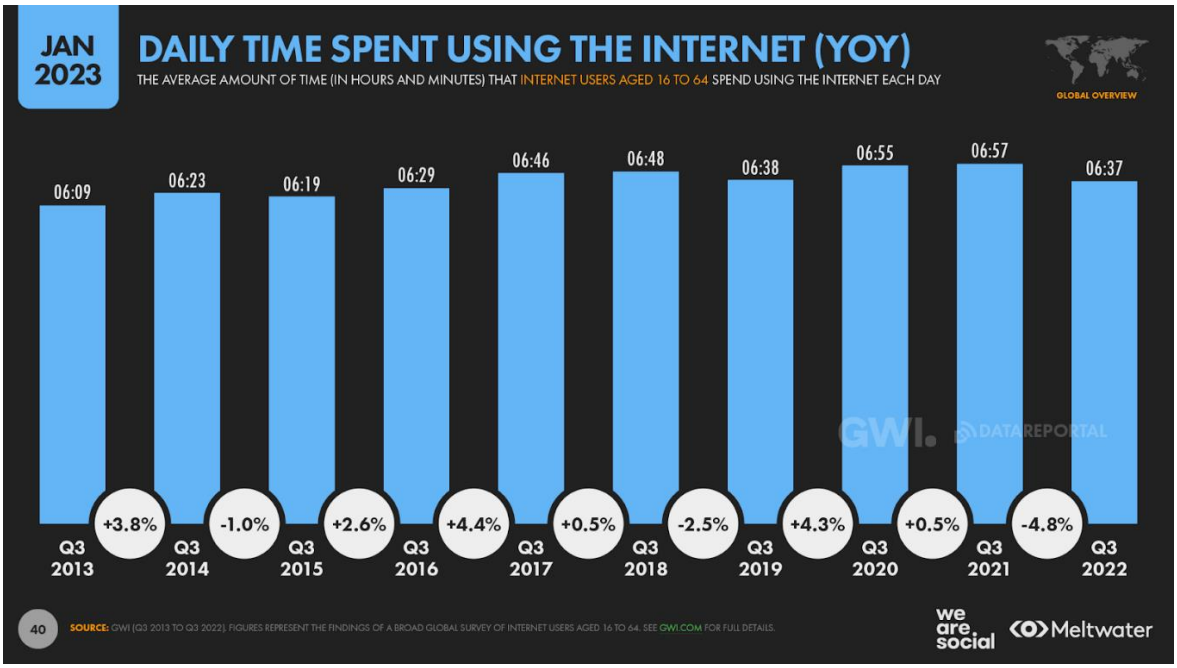

Рис. *2 Статистика щоденного часу, витраченого на використання Інтернету*

Розглянувши рисунок 2, можна побачити, що частка щоденного часу, витраченого на використанні Інтернету зменшилась, в порівнянні з 3 кварталом 2021 року, у 3 кварталі 2021 року цей час становив майже 7 годин, а в 3 кварталі 2022 року цей показник зменшився до 6 годин 37 хвилин на день. Показово, що ця остання цифра дуже близька до середньодобового показника за 3 квартал 2019 року – незадовго до того, як пандемія COVID – 19 завдала серйозного впливу на цифрову поведінку у світі.

Однак нещодавнє послаблення китайської політики "нульового COVID" може призвести до того, що китайські інтернет – користувачі протягом наступних тижнів проводитимуть більше часу за межами країни, що потенційно

може призвести до того, що вони проводитимуть менше часу в Інтернеті. А враховуючи, що на Китай припадає понад один з п'яти (20,4%) світових інтернет – користувачів, будь – яка зміна в онлайн – поведінці в Китаї, ймовірно, матиме значний вплив і на середні світові показники.

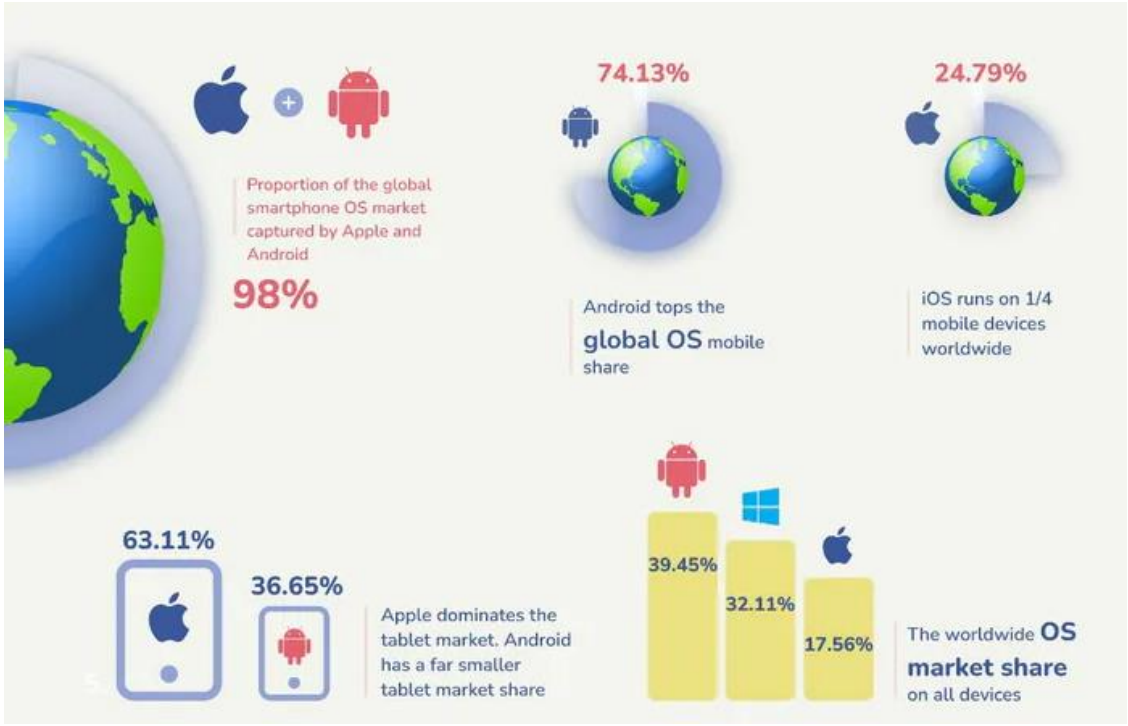

Рис. *3 Ринок операційних систем для смартфонів*

Розглянувши рисунок 3, можна побачити, що Android та iOS займають 98% ринку станом на 2022 рік [4]. Android займає 74,13 відсотки ринку, а iOS 24,79 відсотки. І iOS, і Android були випущені приблизно одночасно. Вперше Apple вивела на ринок свою операційну систему iOS і вона змогла швидко завоювати значну частину ринку. У той же час, Android, що був випущений в 2008 році, почав з відносно невеликою часткою ринку, що становила лише 10% у 2009 році. Однак з плином часу Android став найпоширенішою мобільною операційною системою у всьому світі. Варто зауважити, що перевага Android у популярності не означає, що iOS поступається в якості або рентабельності. Навпаки, iOS все ще має значний світовий вплив і, незважаючи на меншу частку ринку порівняно з Android, генерує більший потік доходів.

Рис. *4 Рейтинг мобільних застосунків Kantar CMeter за квітень 2022 року*

На рисунку 4 зображено рейтинг мобільних застосунків, складений компанією Kantar CMeter [5]. Проаналізувавши даний рейтинг, можна побачити, що Telegram збільшив свою аудиторію на 5,3 відсотки, майже наздогнав Facebook. Застосунок для електронних документів та державних послуг "Дія" нині можна вважати неодмінною частиною смартфонів українців. Охоплення застосунку збільшилося на 10 відсотків протягом трьох місяців й сягнуло позначки 84 відсотків. Також збільшилась популярність використання інтернет-

банкінгу, такі застосунки як Monobank та Privat24, що цікаво, у рейтингу немає застосунку Ochadbank. YouTube music вперше потрапив у топ 20 застосунків, зайнявши 19 місце. Продовжує набирати популярність китайська соціальна медіаплатформа TikTok, яка встановлена у більше ніж 50 відсотків населення України.

1.2 Аналіз програмних продуктів – аналогів

На ринку є кілька інструментів для мобільного тестування, які можуть допомогти автоматизувати тестування застосунків для Android та iOS і скоротити час, необхідний для процесу тестування. Розглянемо деякі приклади таких інструментів.

Xcode – це інтегроване середовище розробки (IDE) від Apple для macOS, яке використовується для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS та tvOS. Xcode [6]. Xcode надає широкий набір інструментів для тестування програмного забезпечення, включаючи наступні:

1. XCTest – це фреймворк для написання, виконання та управління тестами на платформі Apple. Він підтримує написання як юніт-тестів (Unit Tests), так і функціональних тестів.
2. Test Navigator – це панель у Xcode, яка відображає список тестів у проекті. Дозволяє переглядати, запускати та відлагоджувати тести тести безпосередньо з цієї панелі.
3. Test Schemes – дозволяють налаштовувати параметри тестування, такі як використання конкретних тестових кейсів, конфігурацію симулятора або пристрою, виконання певних тестових дій перед або після запуску тестів.
4. UI Testing – Xcode підтримує функціональне тестування інтерфейсу користувача. Є можливість записувати та відтворювати дії користувача на застосунку, перевіряти очікувані результати та виконувати автоматизовані функціональні тести.

Це лише кілька основних інструментів тестування, доступних у Xcode. Також можна використовувати сторонні бібліотеки тестування, які підтримуються Xcode, для розширення функціональності тестування застосунку.

Інтерфейс IDE можемо побачити на рисунку 5.

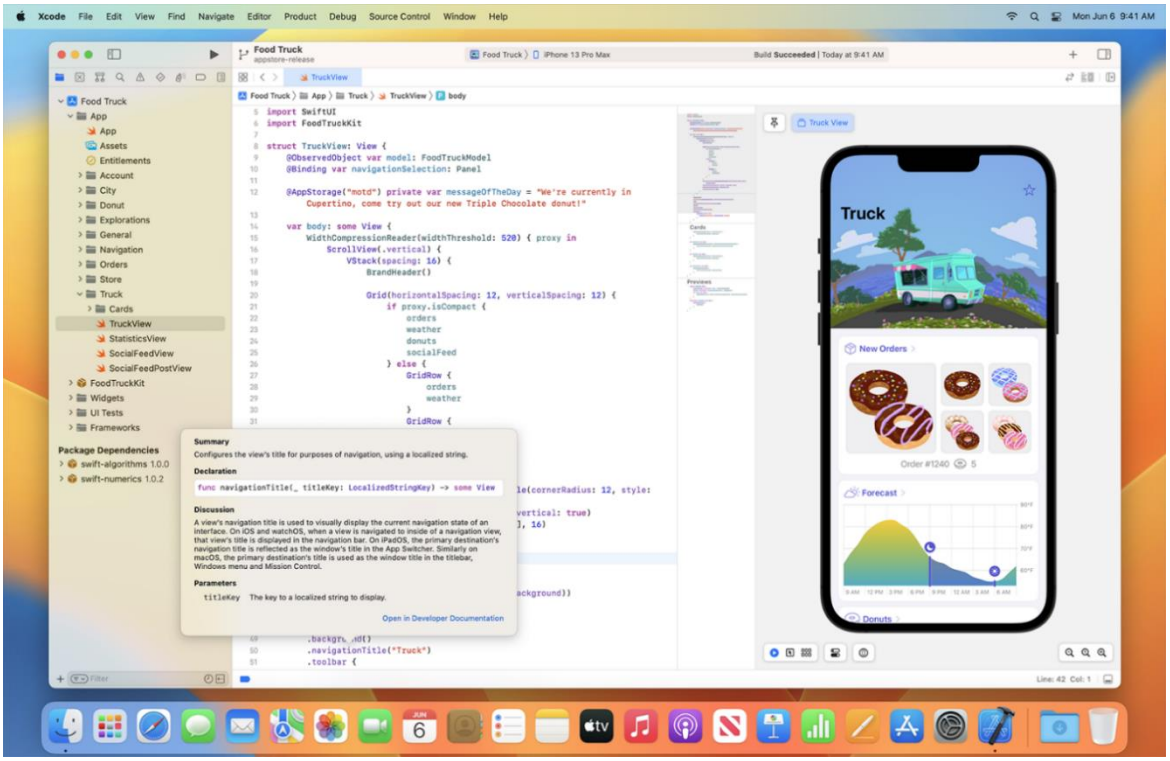

Рис. *5 Інтерфейс Xcode*

Переваги:

1. Xcode є офіційним інструментом розробки для платформи iOS та інших пристроїв Apple, що забезпечує максимальну сумісність і інтегрованість з їхніми технологіями та функціями.
2. Xcode надає повний набір інструментів, включаючи редактор коду, інтерфейс розробки користувача, компілятор, інструменти тестування та розгортання, що спрощує розробку та тестування застосунків.
3. Підтримує мови програмування Swift, Objective – C та AppleScript.
4. Безкоштовний для завантаження та використання.

Недоліки:

1. Працює тільки на macOS, що обмежує користувачів, які можуть використовувати це середовище розробки.
2. Вимагає доволі потужний комп'ютер для роботи з ним.
3. Не підтримує мови програмування, які не є частиною екосистеми Apple.

Selendroid – це інструмент для автоматизації тестування мобільних застосунків на платформі Android. Він надає розширення для фреймворку Selenium, який зазвичай використовується для автоматизації веб-застосунків [7].

Інтерфейс Selendroid Inspector можемо побачити на рисунку 6.

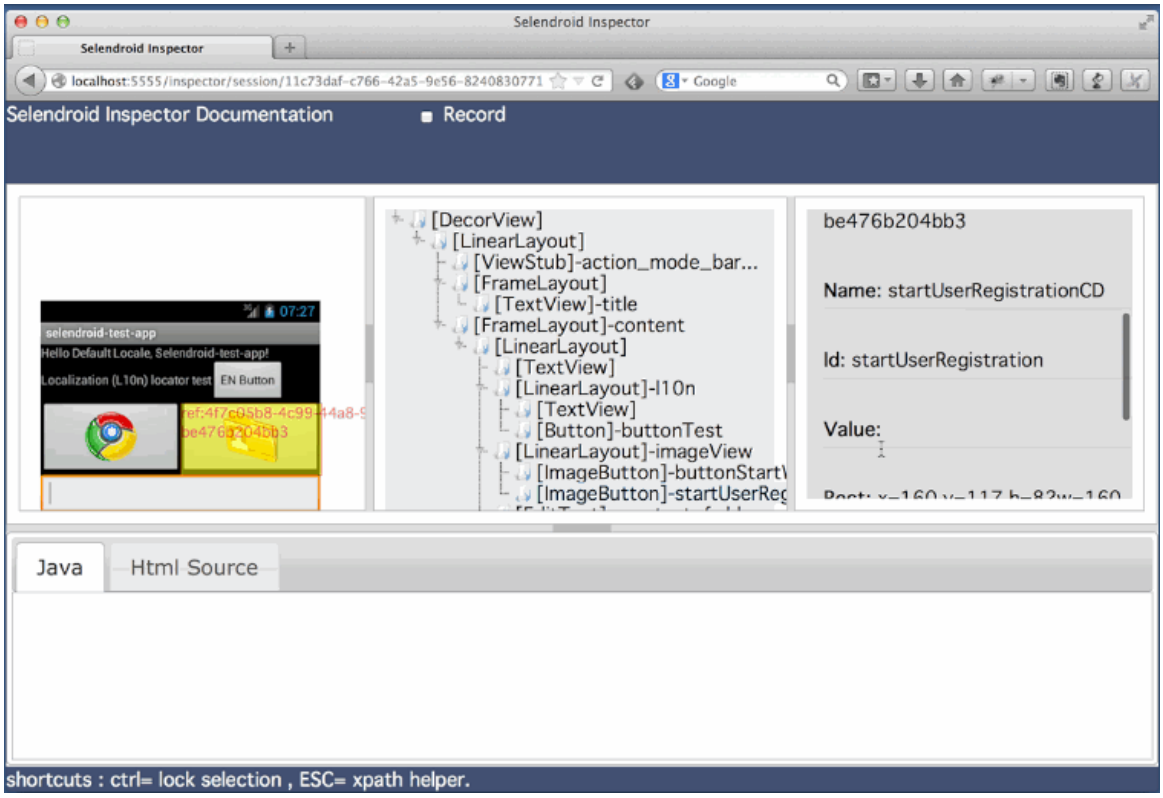

Рис. *6 Інтерфейс Selendroid Inspector*

Переваги:

1. Selendroid дозволяє тестувати мобільні застосунки на різних версіях Android, починаючи з 2.3 і вище. Це дозволяє тестувати застосунки на широкому спектрі пристроїв з різними версіями операційної системи Android.

2. Selendroid є розширенням для фреймворка Selenium, що дозволяє використовувати відому синтаксичну структуру тестування Selenium WebDriver для автоматизації мобільних застосунків на Android. Це забезпечує зручну інтеграцію і спрощує використання Selendroid для автоматизованого тестування.
3. Може використовуватися як на реальних пристроях, так і на емуляторах/симуляторах, що робить його відмінним інструментом для тестування сумісності.

Недоліки:

1. Не підтримує інші мобільні операційні системи, крім Android.
2. Selendroid може мати обмежену підтримку для певних пристроїв або вимагати додаткових налаштувань для сумісності з окремими моделями пристроїв. Це може обмежити здатність тестувати застосунки на певних пристроях або може вимагати додаткових зусиль для налаштування.
3. Selendroid може не мати такого багатого набору функцій та можливостей, які можуть бути доступні в більш нових або популярних інструментах автоматизації тестування, таких як Appium. Це може обмежити здатність досягти повноцінного тестування або реалізувати певні сценарії тестування.

TestComplete – це інструмент для автоматизованого тестування програмного забезпечення, який дозволяє тестувати графічний інтерфейс, веб-застосунки, мобільні застосунки та інші типи програмного забезпечення [8].

Інтерфейс TestComplete можемо побачити на рисунку 7.

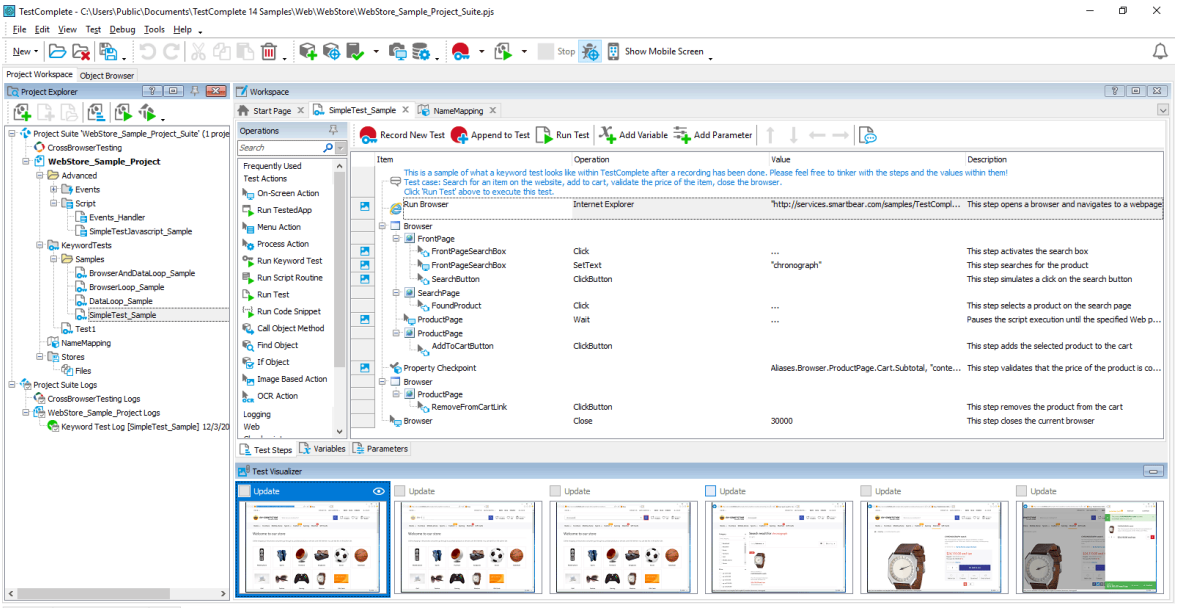

Рис. *7 Інтерфейс TestComplete*

Переваги:

1. TestComplete підтримує автоматизоване тестування на різних платформах, включаючи Windows, macOS, Linux, Android та iOS.
2. Інструмент має вбудовані засоби для створення тестових скриптів, що дозволяє швидко створювати тестові сценарії.
3. TestComplete має вбудовану функціональність для збору даних про продукт, що дозволяє аналізувати результати тестування та виявляти проблеми.
4. Інструмент має можливість інтеграції з іншими інструментами для автоматизованого тестування, такими як Selenium та Jenkins.

Недоліки:

1. TestComplete є комерційним інструментом, що може бути досить дорогим для деяких користувачів.
2. Інструмент може бути складним у використанні для деяких користувачів, особливо для тих, хто не має досвіду в автоматизованому тестуванні.
3. Хоча TestComplete підтримує багато технологій, в ньому можуть бути обмеження для деяких специфічних платформ або технологій. Це може

бути проблемою, якщо проект використовує рідкісну або спеціалізовану технологію.

Appium – це відкритий фреймворк для автоматизації тестування мобільних застосунків. Він дозволяє розробникам тестувати програми на різних платформах, таких як Android, iOS та Windows [9].

Переваги:

1. Відкритий фреймворк, що дозволяє розробникам та тестувальникам використовувати його безкоштовно.
2. Підтримує різні мови програмування, такі як Java, Python, Ruby, JavaScript, та інші.
3. Дозволяє тестувати мобільні застосунки на різних платформах, таких як iOS , Android та Windows.
4. Можливість використовувати один і той же код для тестування на різних платформах.
5. Є підтримка для різних інструментів автоматизації, таких як Selenium WebDriver.

Недоліки:

1. Не підтримує тестування на різних пристроях одночасно.
2. Не підтримує тестування на різних версіях ОС одночасно.
3. Є проблеми зі стабільністю тестів на деяких платформах.
4. Не підтримує тестування застосунків, які використовують нативні елементи управління, такі як календарі, камери, контакти тощо.

Нижче наведена таблиця з порівняннями властивостей Selendroid, TestComplete та Appium.

Таблиця *1*

Порівняння властивостей Selendroid, TestComplete та Appium

Властивість	**Selendroid**	**TestComplete**	**Appium**
Мови програмування	Java	C#, VBScript, JavaScript	Java, C#, Ruby, Python, JavaScript
Платформи	Android	Windows, macOS, iOS, Android	Android, iOS, Windows
Тестові можливості	Обмежені	Широкий спектр можливостей	Широкий спектр можливостей
Інтеграція з іншими інструментами	Часткова	Широкий спектр інтеграцій	Широкий спектр інтеграцій
Вартість	Відкритий код, безкоштовний	Комерційний, платний	Відкритий код, безкоштовний
Можливість тестування веб-застосунків	За допомогою Selenium WebDriver	Ні	За допомогою Selenium WebDriver
Підтримка тестування на реальних пристроях	Ні	Так	Так

Переглянувши та дослідивши інструменти для тестування мобільних застосунків, обраним інструментом для тестування був Appium.

1.3 Постановка завдання

Метою кваліфікаційної роботи є розробка автоматизованих тестів для мобільного застосунку за допомогою технології Appium.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати сучасні підходи до тестування мобільних застосунків та ознайомитись з основними проблемами, з якими стикаються розробники при реалізації тестування.

2. Виконати дослідження технології Appium: вивчити принципи роботи та можливості технології, ознайомитись з документацією.
3. Проаналізувати емулятори пристроїв на платформі Android та обрати більш зручний для роботи.
4. Обрати мобільний застосунок, який буде тестуватись за допомогою Appium.
5. Розгорнути та налаштувати середовище розробки тестів.
6. Реалізувати автоматизоване тестування мобільного застосунку з використанням технології Appium, використовуючи розроблену методологію та набір тестових сценаріїв.
7. Написати пояснювальну записку з описанням, використаних технологій та інструментів, обраного для тестування застосунку, процесу розробки тестових сценаріїв, аналізом результатів тестування та висновками і рекомендаціями щодо подальшого вдосконалення мобільного застосунку.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ МОБІЛЬНОГО ТЕСТУВАННЯ

2.1 Огляд технології Appium

Appium – це відкритий інструмент для автоматизації тестування мобільних застосунків. Він забезпечує можливість тестувати застосунків на різних платформах, таких як Android, iOS та Windows. Appium дозволяє тестувати застосунки на реальних пристроях, емуляторах або симуляторах. На офіційному сайті фреймворку визначають його філософію, яка міститься у чотирьох тезах та характеризує фреймворк [9]:

1. При створенні автоматизованих тестів відсутня необхідність перекомпіляції або будь – якої зміни програми, що тестується. Appium використовує такі фреймворки, які були надані постачальниками:

– Apple XCUITest для iOS (версії 9.3 і наступних версій);
– Apple UIAutomation для iOS (версії 9.3 і попередніх версій);
– Google UiAutomator/UiAutomator2 для Android (версій 4.3+);
– WinAppDriver від Microsoft для Windows Mobile.

2. Appium підтримує багатий вибір мов програмування та тестових фреймворків. Ця теза забезпечується використанням фреймворка Selenium WebDriver, для якого існують прив’язки до багатьох мов програмування та тестових фреймворків.
3. Appium не "винаходить колесо", коли справа доходить до API автоматизації. WebDriver став стандартом де – факто для автоматизації веб-браузерів і є робочим проектом W3C. Appium використовує засоби WebDriver, розширюючи їх додатковими методами API, корисними для мобільної автоматизації.
4. Структура мобільної автоматизації має бути з відкритим вихідним кодом, Appium відноситься до такого програмного забезпечення.

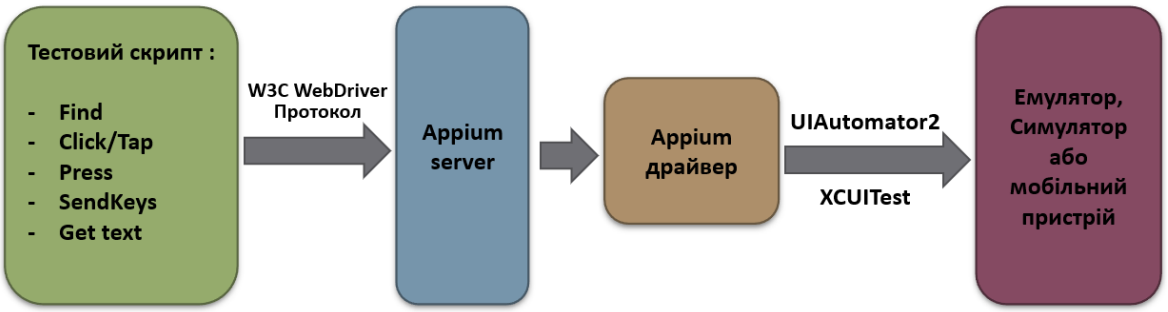

Рис. *8 Архітектура Appium*

На рисунку 8 наведена узагальнена архітектура Appium [10]. Основною складовою є Appium Server – це HTTP – сервер, який базується на платформі Node.js. За допомогою HTTP – запитів протоколу W3C WebDriver [11], команди тестових скриптів, що створені з використанням Appium, Selenium та обраного тестового фреймворка (залежно від мови програмування), передаються з комп'ютера, на якому запущений тестовий скрипт, до Appium Server через Інтернет або локальну мережу. Appium Server передає ці команди до програм – драйверів, які керують фреймворками автоматизації мобільних застосунків відповідних платформ (UIAutomator2 – Android, XCUITest – iOS), в рамках встановлених з ними сесій. Ці драйвери взаємодіють з програмами, що керують виконанням тестів на мобільних пристроях або емуляторах/симуляторах за допомогою нативних протоколів. Ці програми відповідають за виконання тестів у наітивних та гібридних застосунках, які підлягають тестуванню (для тестування веб-застосунків підтримується протокол W3C WebDriver мобільними браузерами).

WebDriver API є стандартним інтерфейсом для автоматизації браузерів, який дозволяє створювати тестові скрипти для тестування веб-застосунків. Цей інтерфейс дозволяє взаємодіяти з браузером і виконувати різні дії, такі як введення тексту, натискання кнопок, отримання даних з елементів сторінки та інші.

Appium використовує WebDriver API для взаємодії з мобільними застосунками, оскільки мобільні застосунки також можуть бути розглянуті як "браузери". Замість того, щоб взаємодіяти з елементами HTML сторінки, як в

випадку з веб-застосунками, Appium взаємодіє з елементами користувацького інтерфейсу мобільного застосунку, використовуючи інструменти, які надаються WebDriver API.

WebDriver API забезпечує різноманітні методи для навігації по веб-сторінці, знаходження елементів, виконання дій з елементами та отримання даних зі сторінок. Наприклад, деякі з основних методів WebDriver API включають:

1. get(url): Завантажує веб-сторінку за вказаною URL – адресою.
2. findElement(by): Знаходить перший елемент на сторінці, що відповідає заданому локатору (наприклад, id, клас, CSS – селектор).
3. click(): Клікає на елемент.
4. sendKeys(text): Вводить текст у поле вводу або елемент.
5. getText(): Отримує текст, що відображається на елементі.
6. getAttribute(attributeName): Отримує значення атрибута елемента.
7. submit(): Відправляє форму.

Це лише декілька прикладів методів, доступних у WebDriver API. Даний API є стандартом для багатьох інструментів автоматизації, включаючи Appium, і дозволяє розробникам створювати автоматизовані тестові сценарії для веб-застосунків і мобільних застосунків.

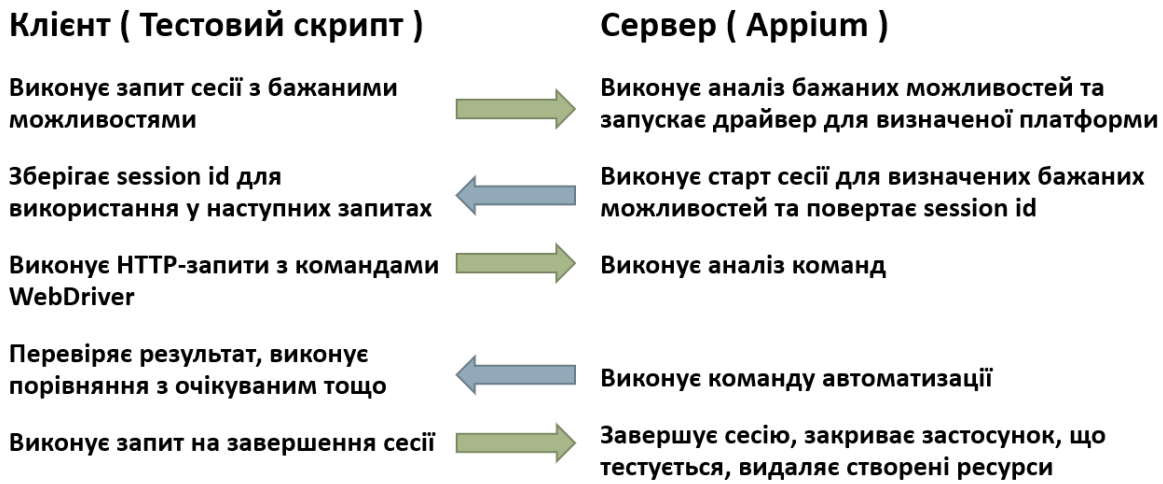

Рис. *9 Клієнт – серверна взаємодія при виконанні тесту Appium*

Розглянемо більш детально взаємодію між клієнтом і Appium Server під час проведення тестування на платформі Android (див. Рис. 9). Виконання тесту починається з клієнта, який може бути комп'ютером, на якому запускається тестовий скрипт. Клієнт створює об'єкт UiAutomator2Options, який містить бажані можливості (desired capabilities). Ці можливості є параметрами для нової сесії тестування і передаються у вигляді JSON – об'єкта через HTTP – запит команди WebDriver New Session. Бажані можливості визначають потрібне тестувальне середовище (див. Лістинг 1).

Лістинг *1 Приклад створення об’єкта з бажаними можливостями*

```
UiAutomator2Options options = new UiAutomator2Options()
    .setPlatformVersion("10.0")
    .setDeviceName("Android Phone")
    .setApp("/path/to/TestApp.apk")
    .eventTimings();
```

Бажані можливості (Desired Capabilities) представляють собою набір пар "ключ – значення" у форматі JSON, який клієнт (комп'ютер, на якому виконується тестовий скрипт) надсилає серверу. Ці пари "ключ – значення" визначають параметри для створення сесії на певній платформі, мобільному пристрої (або емуляторі/симуляторі) та застосунку. Існують загальні бажані можливості, які застосовуються незалежно від мобільної операційної системи, а також специфічні можливості для кожної платформи. В таблиці 2 наведені найпоширеніші загальні бажані можливості.

Таблиця *2*

Клієнтські бібліотеки Appium

Можливість	Опис	Найчастіше вживані значення
automationName	Фреймворк автоматизації, що буде використовуватись	Appium (за замовчуванням) або UiAutomator2 або Espresso для Android, або XCUITest для iOS
platformName	Операційна система мобільного пристрою	iOS або Android
platformVersion	Версія операційної системи мобільного пристрою	наприклад, 7.1, 11.0
deviceName	Встановлює тип мобільного пристрою або емулятора	Наприклад, iPhone Simulator, Android Emulator, Galaxy S4. На iOS визначається командою instruments – s, на Android – adb devices.
app	Абсолютний шлях до файлу або URL для скачування файлу в форматі .ipa, .apk, або .zip	наприклад, "e:\Calculator.apk" або "https://www.example.com/apps/Calculator.apk"
browserName	Використовується при тестуванні Веб-застосунків, визначає мобільний браузер для тестування	"Safari" для iOS та "Chrome", "Chromium" або "Browser" для Android
autoWebview	Запускає затосунок безпосередньо у контексті Webview.	true або false, за замовчуванням false
noReset	Не виконує перевстановлення стану застосунку	true або false. Якщо true – для Android не видаляє apk, для iOS – не видаляє симулятор
fullReset	Виконує повне перевстановлення	true або false. Якщо true – для Android видаляє apk, для iOS – видаляє симулятор

У прикладі на лістингу 1 наведені налаштування, які визначають драйвер (тип об'єкта), платформу, на якій буде тестуватися застосунок (це автоматично встановлюється відповідно до драйвера), версію платформи, назву пристрою

(для Android визначається командою adb devices) і повний шлях до тестового застосунку в проекті. Варто зазначити, що в самому тестовому скрипті об'єкт бажаних можливостей зазвичай будується у форматі, що не є простим JSON – рядком, наприклад, це може бути словник у Python або спеціальний об'єкт у Java.

Таким чином, клієнт Appium запитує сервер про створення сесії з цими можливостями. Сервер аналізує ці можливості, визначає, що потрібно автоматизувати, і запускає відповідний драйвер для визначеної платформи. Після цього розпочинається сесія тестування. На мобільному пристрої або емуляторі/симуляторі запускається тестовий застосунок, і виконується необхідна ініціалізація на конкретній платформі. Сервер Appium повертає клієнту ідентифікатор сесії, який клієнт зберігає для майбутніх запитів.

На даному етапі клієнт може надсилати різноманітні команди автоматизації (WebDriver команди) для пошуку та взаємодії з елементами інтерфейсу. Кожна з цих команд проходить аналіз параметрів у JSON – об'єкті сервером Appium, що визначає, які дії потрібно виконати, і пересилає відповідну команду відповідному драйверу, який активний у поточній сесії.

Після виконання команди у тестовому застосунку з'являється певний результат. Це може бути запитаний текст або просто значення null, що вказує на успішне виконання команди. Результат повертається клієнту у форматі JSON, який аналізується та перетворюється відповідно до обраної мови програмування та тестового фреймворку. Тепер оператор тестового фреймворку може порівняти отриманий результат з очікуваним.

Після завершення клієнт – серверної взаємодії, яка реалізує логіку тесту, незалежно від того, був він пройдений чи ні, необхідно завершити сесію. На цьому етапі Appium закриє тестовий застосунок, очистить всі виділені ресурси, що були використані протягом сесії, і звільнить пристрій для майбутніх сесій.

Варто зазначити, що протокол WebDriver був розроблений для автоматизації веб-браузерів і не підтримує всі можливості мобільних застосунків, наприклад, переміщення файлів у файловій системі пристрою. Однак завдяки розширенню специфікації WebDriver, розробники Appium додали

до стандартного функціоналу WebDriver додаткові інструменти, корисні для автоматизації мобільних застосунків. Ці інструменти використовують ті ж самі механізми, стилі маршрутизації та методи HTTP, що й стандартні методи API.

Якщо розглянути роботу клієнта Appium, який використовує протокол WebDriver, можна очікувати, що клієнти Selenium WebDriver (для визначених мов програмування та тестових фреймворків) можуть бути використані для виконання тестів з Appium. Це насправді правда, однак клієнти Selenium WebDriver не можуть використовувати розширення Appium. Тому були розроблені спеціальні клієнтські бібліотеки Appium для кожної з підтримуваних мов програмування (див. Табл. 3). Ці клієнтські бібліотеки Appium, як правило, надають обгортку навколо клієнтських засобів Selenium та додають підтримку розширень Appium.

Таблиця *3*

Клієнтські бібліотеки Appium

Клієнт Appium	**Адреса репозиторію**
Java	https://github.com/appium/java – client
Python	https://github.com/appium/python – client
Ruby	https://github.com/appium/ruby_lib
JavaScript	https://github.com/admc/wd
C#	https://github.com/appium/dotnet – client
PHP	https://github.com/appium – boneyard/php – client

Драйвери Appium є частиною фреймворку і відповідають за автоматизацію конкретної платформи. Ці драйвери є модулями, які вибираються сервером Appium з доступного набору драйверів в Appium залежно від запиту на створення сесії. Вони перетворюють команди протоколу WebDriver на команди, зрозумілі для фреймворків автоматизації на конкретній платформі. Appium підтримує широкий спектр драйверів, як для популярних платформ мобільних пристроїв – Android, iOS та Windows, так і для специфічних платформ. Наприклад, драйвер

для телевізійної платформи під назвою UI.TV, драйвер для мобільних або вбудованих операційних систем Samsung Tizen, драйвер для розробки крос – платформних програм з використанням Google Flutter та інші.

Сервер Appium виконує аналіз команди WebDriver і передає її відповідному драйверу. Кожен драйвер приймає відповідальність за виконання отриманої команди. В ідеалі, всі драйвери виконують однакові дії для однакових команд, але іноді це неможливо. Деякі платформи можуть не мати певних команд, або спосіб реалізації певної функціональності може відрізнятися між платформами. У таких випадках можуть виникати незначні відмінності в поведінці на різних платформах. Проте, драйвери намагаються уникати цих відмінностей в поведінці, наскільки це можливо.

Часто для однієї платформи існує кілька драйверів, які залежать від різних базових технологій, і деякі з них можуть мати значно різні набори функцій. При тестуванні необхідно вибрати один з цих драйверів. На щастя, Appium має перевагу у тому, що можна легко переключатись між різними драйверами.

Розглянемо більш детально драйвери для найпопулярніших мобільних платформ. Для Android на сьогоднішній день існує два актуальних драйвери: Appium UiAutomator2 Driver, який є поточним стандартом, підтримуваним Google, і Espresso Driver, який також підтримується Google. Останній має деякі переваги у швидкості та надійності тестування, але одним з його недоліків є обмеження на тестування тільки програм, розроблених вами, тобто неможливість тестування довільних застосунків. Для iOS наразі актуальний лише один драйвер – Appium XCUITest Driver, який базується на XCUI API від Apple.

Кожен з цих драйверів має свою внутрішню структуру. Наприклад, драйвер UIAutomator2 містить JavaScript – частину, яка є складовою частиною сервера Appium, а також Java – частину, яка запускається як застосунок appium – uiautomator2 – server – vX.X.X.apk (де X.X.X – версія), на пристрої Android і встановлюється в каталог device/emulator.

Аналогічно для драйвера iOS: Appium XCUITest Driver, написаний на JavaScript, є складовою сервера Appium, а WebDriverAgent, написаний на Objective – C, встановлюється на мобільні пристрої та відповідає за автоматизацію мобільних застосунків.

2.2 Локатори Веб-елементів та основні команди WebDriver

Локатори веб-елементів-це методи знаходження елементів на веб-сторінці для їх подальшої взаємодії під час автоматизованого тестування за допомогою Selenium WebDriver [12]. Локатори дозволяють знайти елементи за їх ідентифікатором, назвою тегу, класом, текстом або іншими атрибутами. Selenium WebDriver надає різні типи локаторів, такі як:

1. Локатор за ID (By.id): Ідентифікатор є унікальним атрибутом елемента і найшвидшим способом знаходження елементів.
2. Локатор за назвою класу (By.className): Клас може бути присвоєний одному або декільком елементам і використовується для знаходження групи елементів з однаковим класом.
3. Локатор за назвою тегу (By.TagName): Імена тегів, такі як , <input>, <a>, використовуються для знаходження всіх елементів з відповідним тегом.
4. Локатор за ім’ям (By.name): використовується в Selenium WebDriver для знаходження веб-елементів за їх іменем. Цей локатор шукає елементи на веб-сторінці по атрибуту name.
5. Локатор XPath (By.XPath): XPath є мовою для навігації і виразами пошуку в структурі документа, такі як шляхи до елементів або їх атрибути.
6. Локатор за CSS – селектором (By.cssSelector): CSS – селектори використовуються для знаходження елементів за допомогою правил CSS.
7. Локатор за текстом посилання (By.LinkText): Використовується для знаходження посилань за їхнім текстовим вмістом.

8. Локатор за частковим текстом посилання (By.PartialLinkText): Знаходить посилання за частковим текстом, що міститься в ньому.

WebDriver – це інтерфейс, який надає набір методів для автоматизації дій у веб-браузері. Основні команди WebDriver включають наступне:

1. get(url): Відкриває вказану URL – адресу у веб-браузері.
2. navigate().to(url): Аналогічна до get(url), відкриває вказану URL – адресу у веб-браузері.
3. getTitle(): Повертає заголовок сторінки.
4. getCurrentUrl(): Повертає поточний URL – адресу сторінки.
5. findElement(by): Знаходить перший елемент на сторінці, що відповідає заданому селектору.
6. findElements(by): Знаходить всі елементи на сторінці, що відповідають заданому селектору.
7. sendKeys(string text): Вводить текст або клавіші у вибраний елемент.
8. click(): Клікає на вибраний елемент.
9. getText(): Повертає текстове значення елемента.
10. getAttribute(String name): Повертає значення вказаного атрибуту елемента.
11. clear(): Очищує вміст вибраного елемента (якщо це поле вводу).
12. isDisplayed(): Перевіряє, чи видимий елемент на сторінці.
13. isEnabled(): Перевіряє, чи доступний для взаємодії елемент.
14. isSelected(): Перевіряє, чи вибраний елемент (якщо це елемент вибору).
15. submit(): Відправляє форму, до якої належить вибраний елемент.

Ці команди дозволяють взаємодіяти з елементами сторінки, отримувати їх властивості, виконувати дії та отримувати дані з веб-браузера.

2.3 Команди WebDriver, специфічні для мобільного застосунку

WebDriver також можна використовувати для автоматизації мобільних застосунків за допомогою мобільних браузерів або спеціальних бібліотек. Приклад команд WebDriver, специфічних для мобільних застосунків:

1. driver.context(name): Перемикає контекст Appium на заданий контекст. Наприклад, driver.context("WEBVIEW_com.example.app") переключиться на контекст WebView з пакетом com.example.app.
2. driver.startActivity(appPackage, appActivity): Запускає заданий пакет застосунку та активність на пристрої. Наприклад, driver.startActivity("com.example.app", "com.example.app.MainActivity") запустить застосунок з пакетом com.example.app та активністю MainActivity.
3. driver.hideKeyboard(): Ховає віртуальну клавіатуру на мобільному пристрої.
4. driver.getContextHandles(): Повертає список доступних контекстів на мобільному пристрої. Можна використовувати цю команду для переключення між контекстами (наприклад, між веб – вмістом та нативним застосунком).
5. driver.lockDevice(): Блокує мобільний пристрій. Ця команда може бути використана для симуляції заблокованого екрану або використана в інших тестових сценаріях, де потрібно перевірити поведінку на заблокованому пристрої.
6. driver.isDeviceLocked(): Перевіряє, чи заблокований мобільний пристрій. Ця команда повертає булеве значення (true або false) в залежності від того, чи пристрій заблокований.
7. driver.installApp(appPath): Встановлює мобільний застосунку на пристрій. Потрібно вказати шлях до APK – файлу застосунку.
8. driver.removeApp(appId): Видаляє встановлений мобільний застосунок за його ідентифікатором (appId).
9. getBatteryInfo(): повертає інформацію про заряд батареї на мобільному пристрої.
10. drag_and_drop(source, target): перетягування об'єктів.
11. release(selector): Відпускання натискання на екрані.
12. swipe(selector,xoffset,yoffset,speed): проведення пальцем по екрану.

13. rotate(x,y,radius,rotation,touchCount,duration): симуляція жесту повороту на сенсорних екранах або тачпадах.
14. shake(): симуляція жесту тряски на мобільних пристроях.
15. scroll(selector,xoffset,yoffset): прокрутка сторінки або елемента вниз або вгору на сенсорних екранах або тачпадах.

2.4 Розгортання середовища розробки тестів із використанням Appium

Для проведення тестування мобільних застосунків з використанням фреймворку Appium, потрібно налаштувати тестове середовище на комп'ютері. Необхідними попередніми вимогами є наявність на комп'ютері Java Development Kit (JDK), з налаштованими змінними оточення JAVA_HOME та PATH. Оскільки Appium використовується для тестування мобільного застосунку на Android, потрібно встановити Android SDK, можна встановити Android Studio, при цьому буде встановлений як Android SDK, так і необхідні інструменти, включаючи Android Debug Bridge та Virtual Device Manager, які дозволяють запускати Android Emulator.

Так як Appium Server написаний для платформи Node.js, спочатку потрібно встановити цей фреймворк, перевірити правильність інсталяції на версію фреймворку можливо командами (див. Рис. 10).

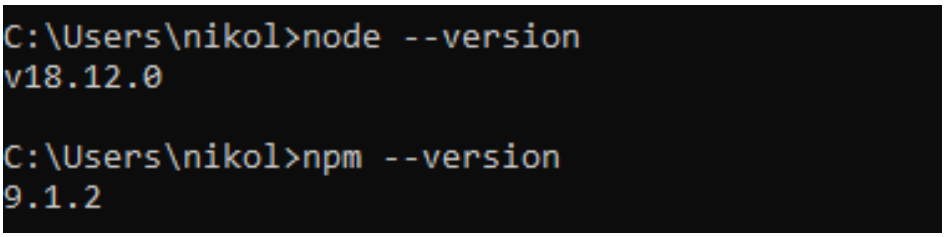

Рис. *10 Перевірка правильності встановлення Node.js*

За допомогою менеджера пакетів Node.js можна встановити сервер Appium командою npm install – g appium. Також можна перевірити правильність інсталяції та версію сервера Appium (див. Рис. 11).

```
C:\Users\nikol>appium -v
1.22.3
```

Рис. *11 Перевірка правильності встановлення Appium Server*

Для зручності встановлена утиліта Appium Server GUI, яка має графічний інтерфейс та використовується для запуску сервера. Встановлення інструмента виконується стандартним чином, після інсталяції на робочому столі виводиться піктограма інструмента. Подвійне натискання по піктограмі запускає утиліту, яка надає засоби запуску сервера Appium (див. Рис. 12).

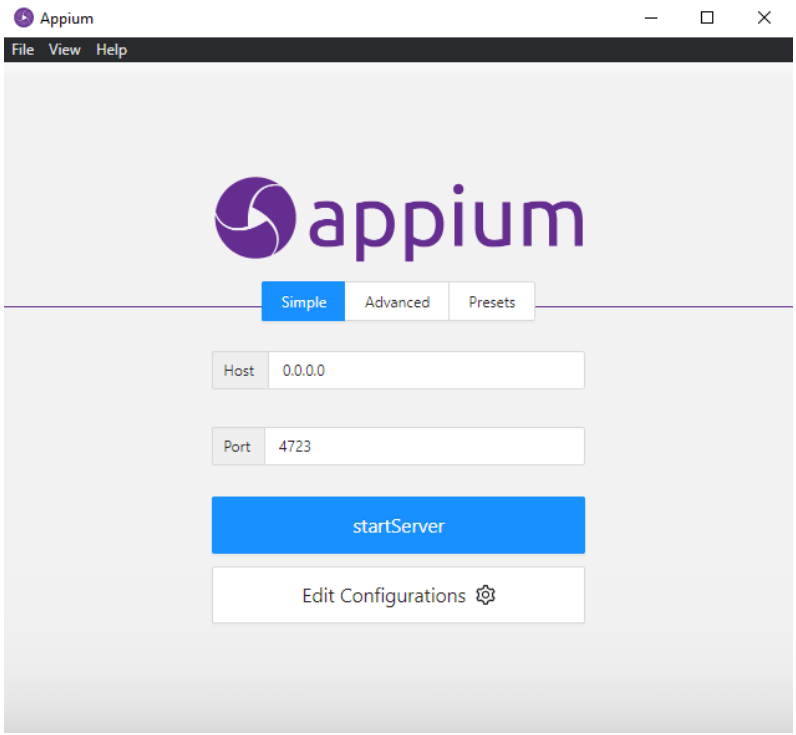

Рис. *12 Інтерфейс утиліти Appium Server GUI*

Друга утиліта, яка потрібна для тестування це Appium Inspector. Appium Inspector є інструментом, який фактично виступає в ролі клієнта сервера Appium. Він є аналогом клієнтських бібліотек Java або Python, але відрізняється тим, що надає графічний інтерфейс для зручного введення бажаних можливостей, пошуку елементів і взаємодії з інтерфейсом мобільного застосунку.

Appium Inspector дозволяє розробникам легко знаходити та взаємодіяти з елементами інтерфейсу за допомогою графічного інтерфейсу, що спрощує відладку та тестування мобільних застосунків. Зокрема, він надає можливість перегляду ієрархії елементів, їх властивостей та атрибутів, що дозволяє точно визначити, з якими елементами потрібно взаємодіяти.

Крім того, Appium Inspector надає засоби записування користувацьких дій із застосунком у форматі скрипту команд Appium. Це дозволяє автоматизувати тести шляхом створення скриптів, які відтворюють послідовність дій, виконаних користувачем, і забезпечують повторне виконання тих самих дій під час тестування.

Встановлюється він стандартним чином, після інсталяції на робочий стіл виводиться піктограма (див. Рис. 13).

Рис. *13 Піктограма інструмента Appium Inspector*

Для перевірки правильності встановлення Appium, необхідних та опціональних залежностей бажано встановити appium – doctor. Після виконання перевірок Appium Doctor надає звіт, що вказує на потенційні проблеми або відсутність необхідних компонентів. Звіт містить рекомендації щодо вирішення проблем та посилання на відповідну документацію або ресурси для отримання додаткової інформації (див. Рис. 14).

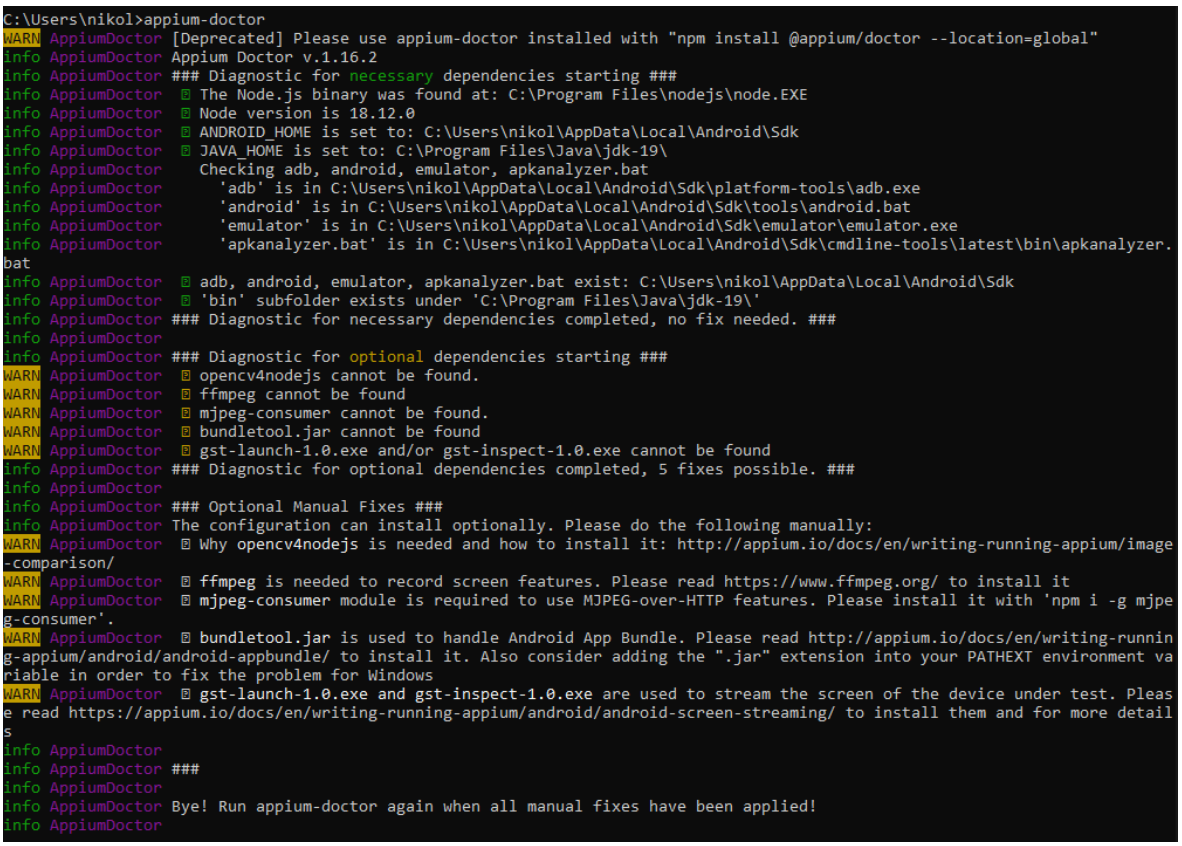

Рис. *14 Перевірка встановлення залежностей утилітою appium-doctor*

Також потрібно встановити Android емулятор Genymotion. Він дозволяє розробникам, тестувальникам та іншим користувачам створювати віртуальні пристрої з різними версіями операційної системи Android і різними налаштуваннями пристрою.

Однією з ключових переваг Genymotion є швидкість роботи. Він базується на технології віртуалізації, такій як VirtualBox або VMware, що дозволяє ефективно використовувати ресурси комп'ютера та швидко емулювати Android – пристрої. Це особливо важливо для розробників, які виконують швидкі тести та розробляють програмне забезпечення для різних версій Android.

Genymotion підтримує широкий спектр Android – пристроїв, включаючи смартфони та планшети різних виробників, таких як Samsung, Google, HTC, Sony та інші. Крім того, є можливість налаштувати різні параметри пристрою, такі як розмір екрану, кількість оперативної пам'яті, версія Android тощо, що дозволяє проводити тестування в різних умовах.

Ще одна важлива риса Genymotion – його інтеграція з різними інструментами розробки. Ви можете легко підключити Genymotion до популярних інструментів розробки, таких як Android Studio або IntelliJ IDEA, і прямо з них керувати емуляторами Genymotion.

Після встановлення та запуску у програмі Genymotion можна створити віртуальний мобільний пристрій з ОС Android, виконавши клік по кнопці  , обравши необхідний пристрій зі списку, або створити власний, з необхідними параметрами. Після створення віртуального пристрою кліканням на трьох вертикальних крапках у його рядку можна відкрити меню, де відображені команди швидкого запуску, «холодного» запуску, редагування, видалення, створення дуплікату та відновлення пристрою до налаштувань по замовчуванню (див. Рис. 15).

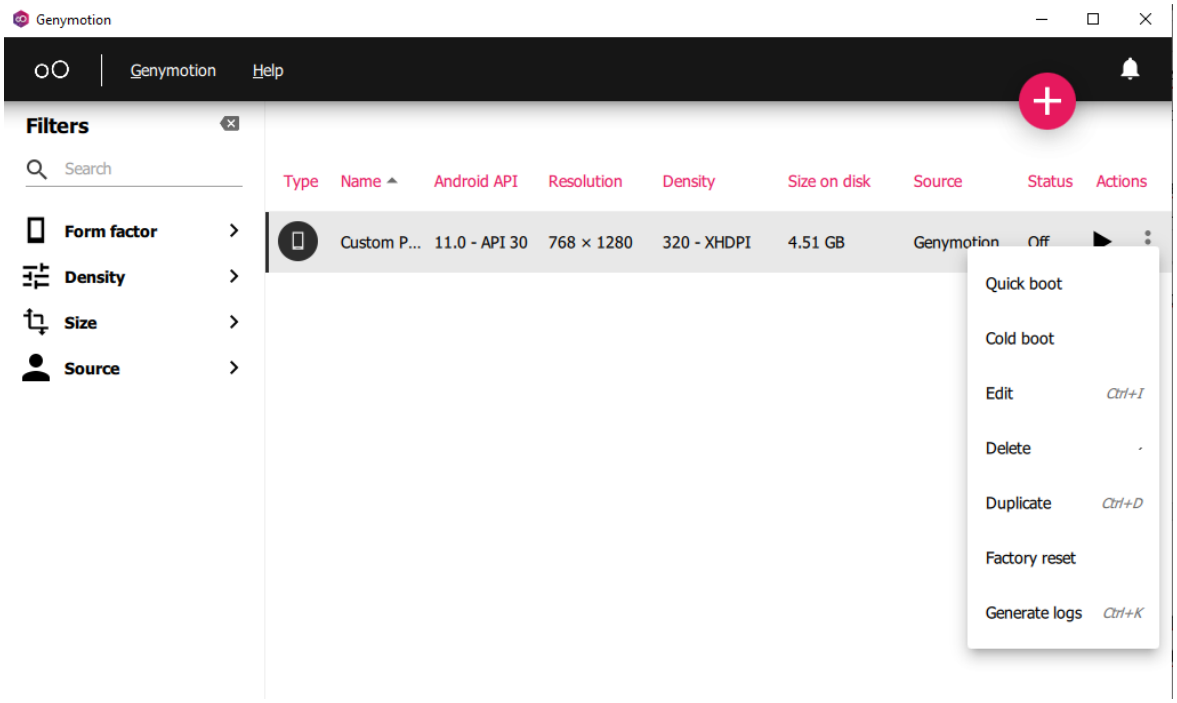

Рис. *15 Інтерфейс Genymotion та меню взаємодії з віртуальним пристроєм*

На цьому налаштування середовища в ОС Windows розробки тестів мобільних застосунків для Android із використанням фреймворку Appium можна вважати завершеним.

3 ОПИС ЗАСТОСУНКУ ТА ВИЗНАЧЕННЯ ЙОГО ФУНКЦІОНАЛУ, ЩО БУДЕ ТЕСТУВАТИСЬ

3.1 Опис функціоналу мобільного застосунку

У якості мобільного застосунку для тестування був обраний Rozetka, який є Android-застосунком популярного онлайн магазину Rozetka.com.ua. Android-застосунки, зазвичай, розповсюджуються як файли з розширенням .apk, тому було завантажено файл (rozetka_5.40.1.apk) з сайту apkpure.com для інсталяції застосунку на віртуальний пристрій Android.

Програма, у частині свого функціоналу, що забезпечує пошук товарів, передбачає заповнення користувачем на початковому екрані поля для пошуку, також є можливість голосового введення та сканер штрихкодів, який дозволяє відсканувати товар за штрихкодом у магазині Rozetka та ознайомитись з повним описом товару у застосунку (див. Рис. 16).

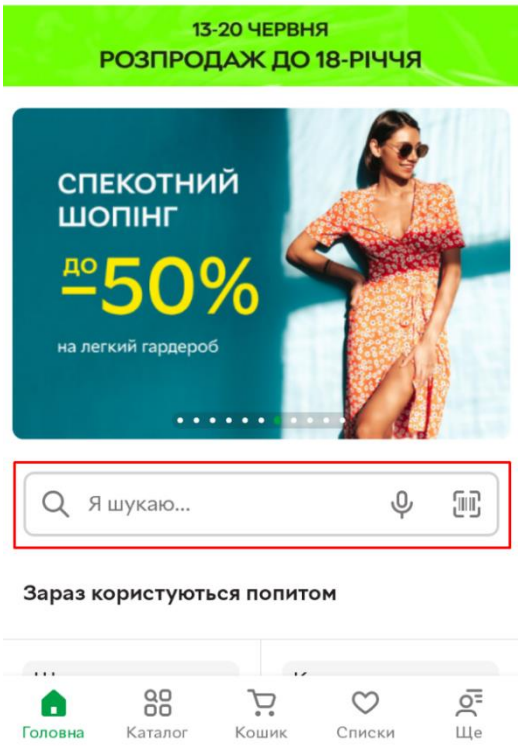

Рис. *16 Початковий екран застосунку Rozetka*

Бажаний товар вводиться після натискання по полю пошуку шляхом введення з віртуальної клавіатури (див. Рис. 17).

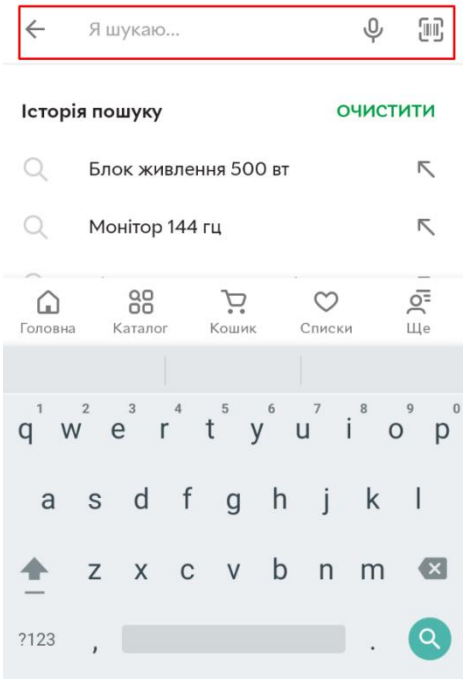

Рис. *17 Екран пошуку товарів*

Після введення вхідних даних та натисканню на кнопці шукати виводиться список з запропонованими товарами, кожен елемент якого містить назву товару, рейтинг (кількість зірок), який регулюється відгуками про сам товар, кількість відгуків, ціну (поточку та без знижки), кнопка додавання в кошик, кнопка додавання у список бажань. На екрані зі списком товарів угорі наявна панель з елементами сортування, фільтрації за певними ознаками, зміна розташування елементів та кнопка, яка дозволяє скопіювати посилання на знайдені товари (див. Рис. 18).

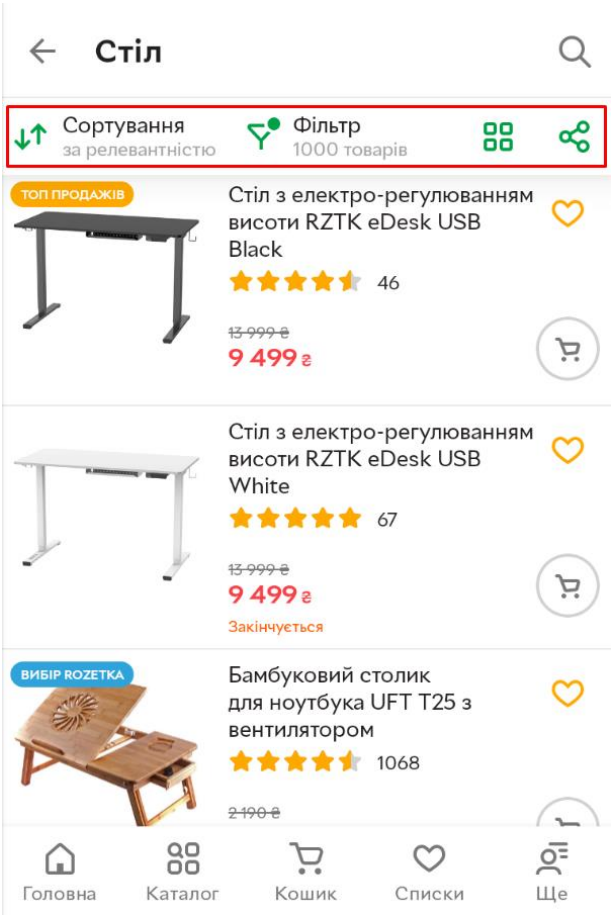

Рис. *18 Результати пошуку товару*

На рисунку 19 показані ознаки, за якими можливе сортування, а саме: сортування від дешевих до дорогих, від дорогих до дешевих, новинки та сортування за рейтингом. На рисунку 20 показані ознаки фільтрації товарів – їх досить велика кількість, та їх кількість залежить від категорії шуканого товару, але є такі фільтри, які належать майже всім товарам, це: продавець, виробник та ціна.

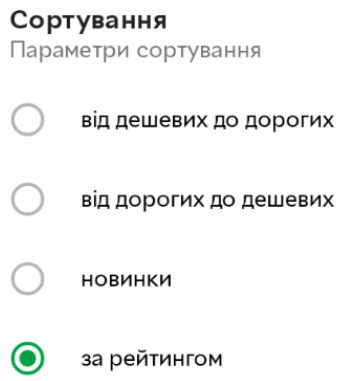

Рис. *19 Параметри сортування*

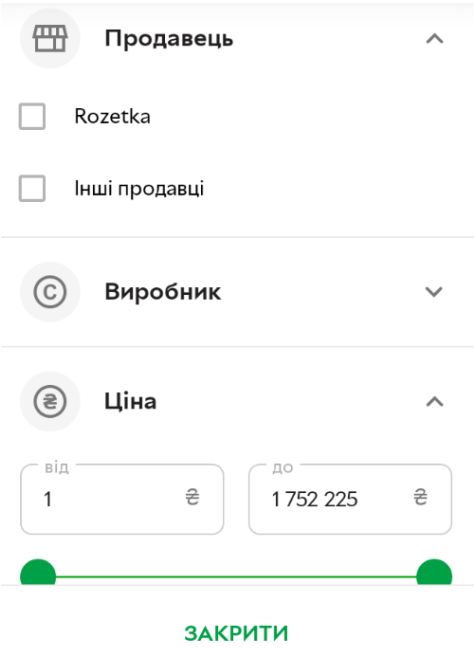

Рис. *20 Ознаки фільтрації*

Також товари можна знайти на головній сторінці, на ній відображаються новинки, акційні товари та товари, які користуються попитом (див. Рис. 21). Вони відображаються разом з категорією, в яких можна ознайомитись зі схожими пропозиціями, але є відмінність у відображенні цих елементів, порівнюючи їх з елементами, які відображаються після пошуку з вхідними даними. Ці елементи мають лише назву та ціну (поточну та без знижки, якщо вона є).

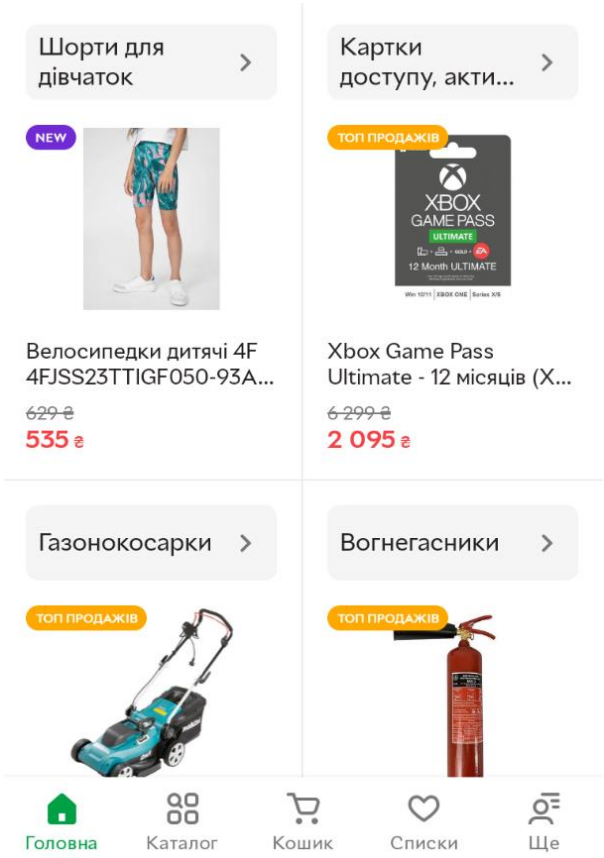

Рис. *21 Товари на головній сторінці*

На екрані застосунку є кнопка «каталог», натиснувши на неї можна побачити велику кількість категорій товарів, що дозволяє користувачам знаходити товари у різних сферах, включаючи електроніку, комп'ютери, мобільні пристрої, побутову техніку, одяг, косметику, товари для дому та багато іншого (див. Рис. 22).

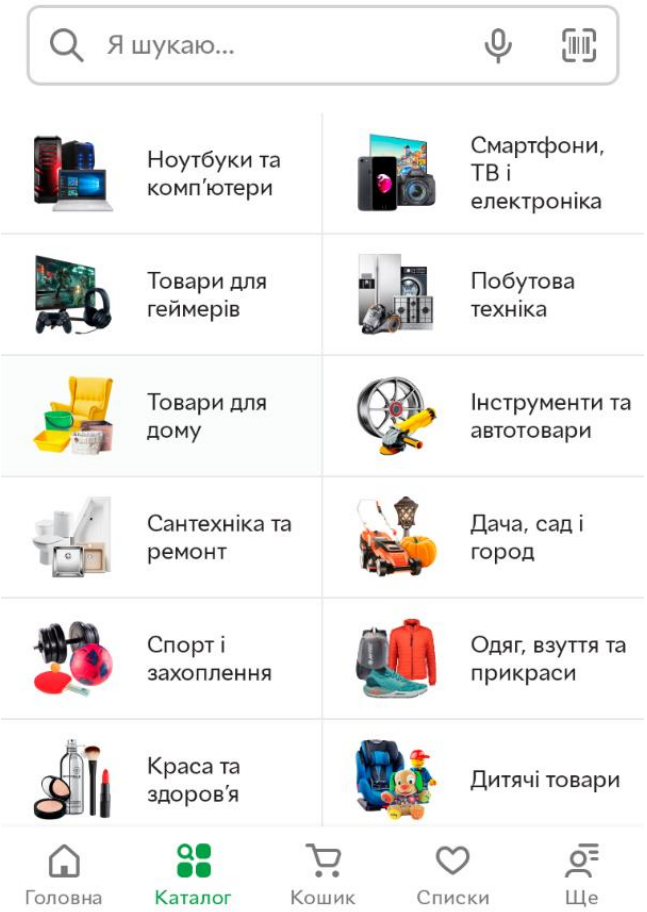

Рис. *22 Каталог та відображені категорії товарів*

Користувач може ознайомитись з більш детальною інформацією про товар, кожен товар має свою сторінку з докладною інформацією про його характеристики, відгуки покупців, питання, фотографії та ціни. На головній сторінці товару наявні панелі з елементами. На нижній панелі розташовані елементи додавання товару у порівняння, додавання у кошик, додавання у бажаний список та кнопка «купити зараз», яка переадресує користувача на сторінку оформлення замовлення. Вище фотографій товару розташована панель з елементами «все про товар», «характеристики», «відгуки», «питання», «фото/відео», «купують разом» (див. Рис. 23).

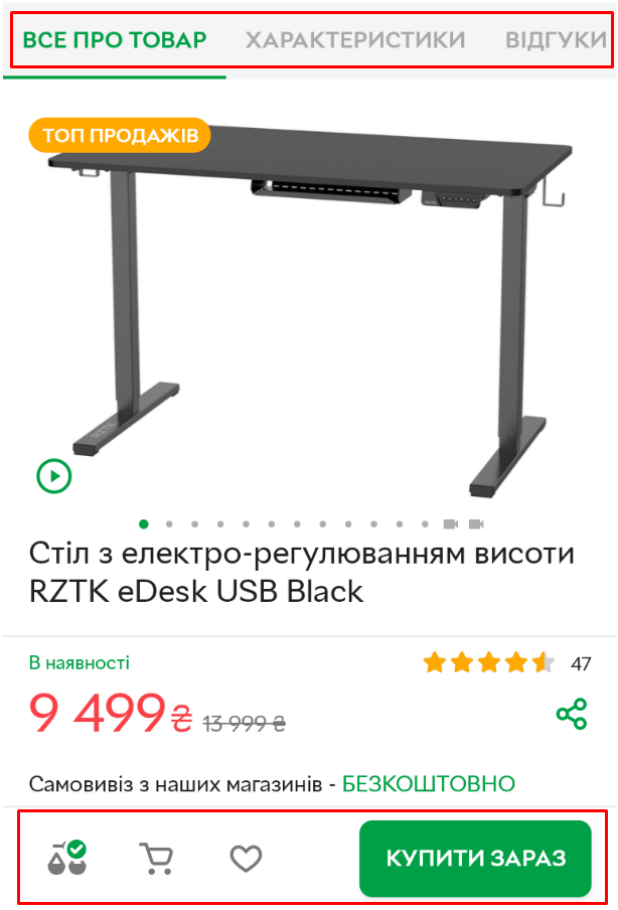

Рис. *23 Сторінка товару та елементи на ній*

У розділі «все про товар» можна ознайомитись з галереєю фотографій товару, його короткий опис, ціну, способи оплати та доставки. Користувач може дізнатися про всі основні технічні деталі товару. У розділі «характеристики» відображена детальна інформація про технічні характеристики конкретного товару. Цей розділ надає конкретні числові та технічні деталі, які допомагають користувачу краще розібратися в особливостях товару. Натиснувши на «відгуки» користувач може ознайомитися з відгуками та оцінками інших покупців про конкретний товар та залишити свій, увійшовши до свого облікового запису. Цей розділ надає користувачу можливість оцінити думки та досвід інших людей, які вже придбали той самий товар. На вкладці «питання» є можливість питання щодо конкретного товару і отримувати відповіді від інших користувачів або представників магазину. Цей розділ створений для полегшення комунікації та вирішення невизначеностей щодо товару перед придбанням. У розділі "відео" користувач може переглядати відеоматеріали, пов'язані з конкретним товаром.

Цей розділ надає можливість користувачам побачити товар у дії, ознайомитися з його функціональністю та дизайном через відеоогляди, огляди експертів та інші відеоматеріали

Серед функціоналу взаємодії користувача з товаром можна виділити такі пункти: додавання товару у порівняння, додавання товару у списки бажаного, додавання товару у кошик, для подальшого замовлення.

Порівняння товарів можна знайти натиснувши на елемент «Ще» на нижній панелі застосунку (див. Рис. 24).

Рис. *24 Нижня панель застосунку з елементами*

Додані товари у списки бажаного можна переглянути, натиснувши на елемент «Списки» на нижній панелі. Додаючи товар до списку бажаних, за замовчуванням вони сортуються у групи, залежно від пристрою на якому товар був доданий до списку (див. Рис. 25), також є кнопка +, виконавши тап по ній, з’являється екран, на якому можна створити власний список та зробити його за замовчуванням (див. Рис. 26).

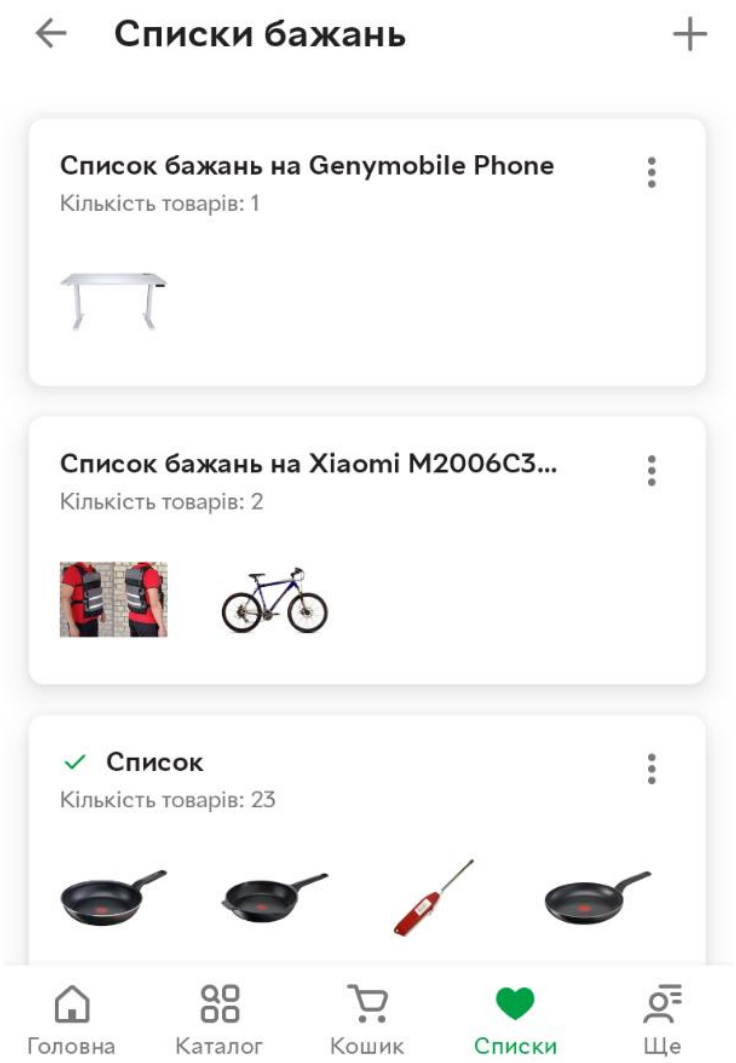

Рис. *25 Екран списків бажань*

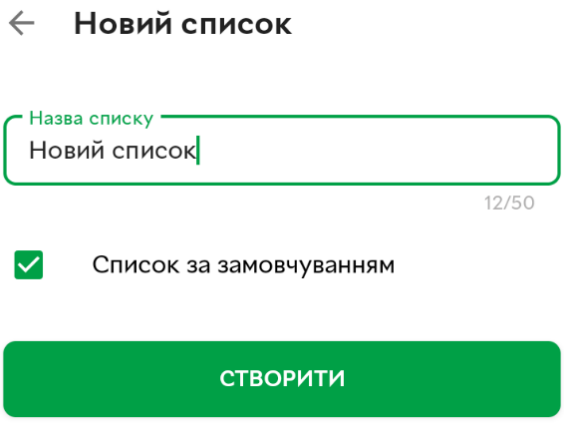

Рис. *26 Екран створення нового списку за замовчуванням*

Кожен елемент списку бажань містить: назву списку, кількість товарів у ньому, зображення товарів, доданих у список та меню з функціями видалення списку, редагування списку, перемістити товари у інший список та поділитися посиланням на список бажань (див. Рис. 27).

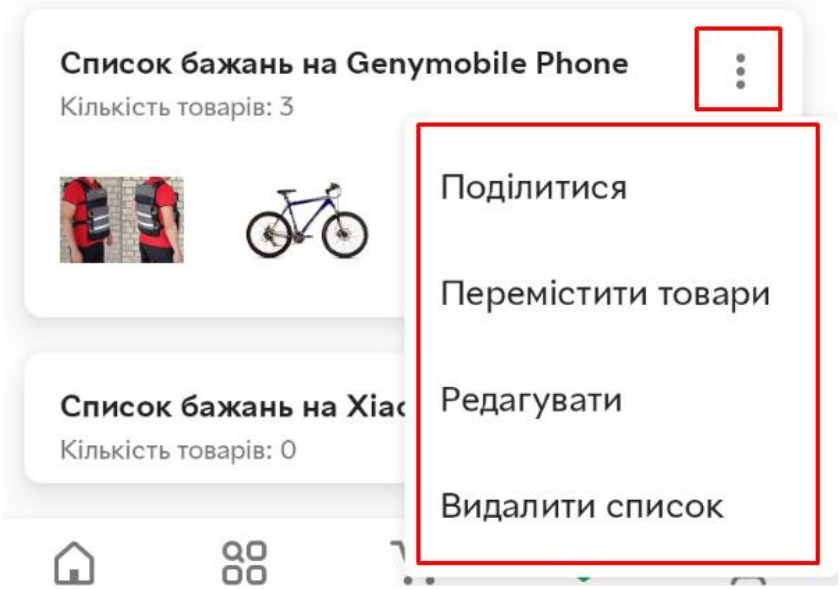

Рис. *27 Меню з функціями взаємодії зі списком бажань*

Виконавши тап по списку бажань, з’являється екран з доданими до нього товарами, кожен елемент у списку містить: назву, рейтинг (кількість зірок), кількість відгуків, ціну (поточна та без урахування акції), кнопка додавання у кошик та меню з функціями перемістити товар (переміщення товару з поточного списку у інший) та видалити товар зі списку (див. Рис. 28).

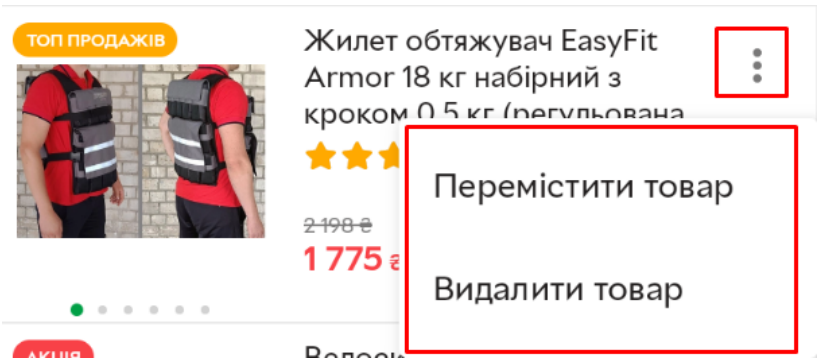

Рис. *28 Меню з функціями взаємодії з товаром у списку бажань*

У списку бажань на верхній панелі відображається кількість товарів, доданих у список, сума цих товарів, кнопка «купити все», та кнопка копіювання посилання на список бажань (див. Рис. 29).

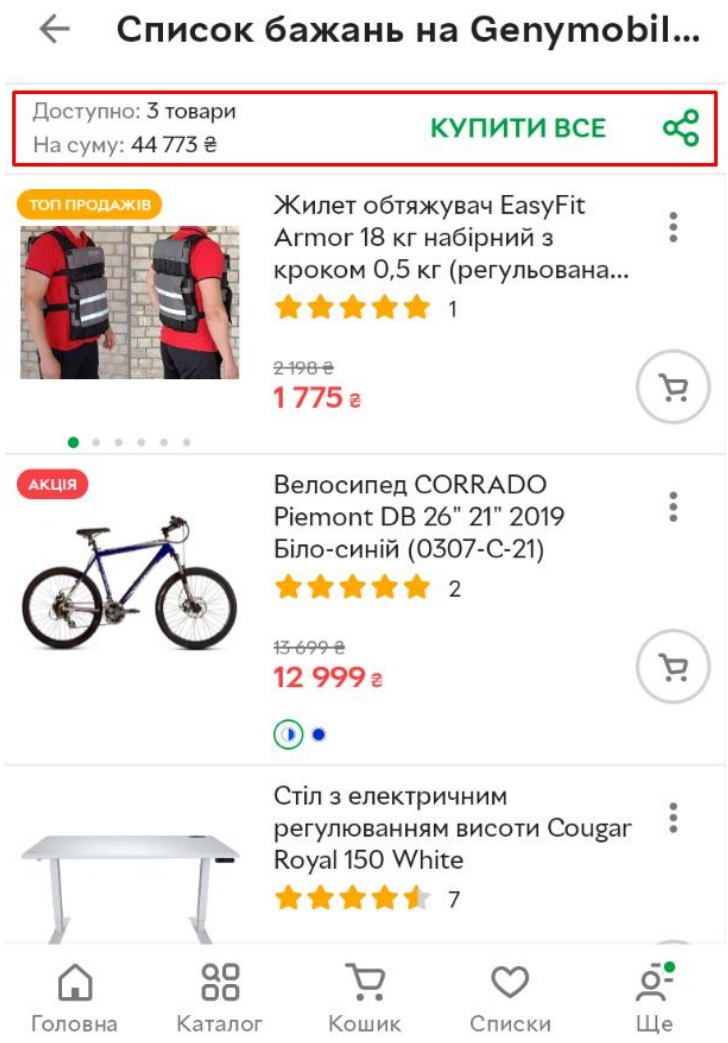

Рис. *29 Екран списку бажань*

Виконавши тап по кнопці «купити все» товари додаються до кошика, перейшовши на екран кошика відображається список товарів, які були додані до нього, кожний елемент у списку містить: checkbox, який додає або прибирає товар із замовлення, назву товару, продавець товару, ціну (поточну та без урахування знижки), меню з 3 крапками, яке має функцію видалення товару з кошика, кількість пропозицій заданого товару у кошику, яку можна змінити

натиснувши на елементи + та – на елементі та додаткові послуги, якщо вони прикріплені до товару (див. Рис. 30).

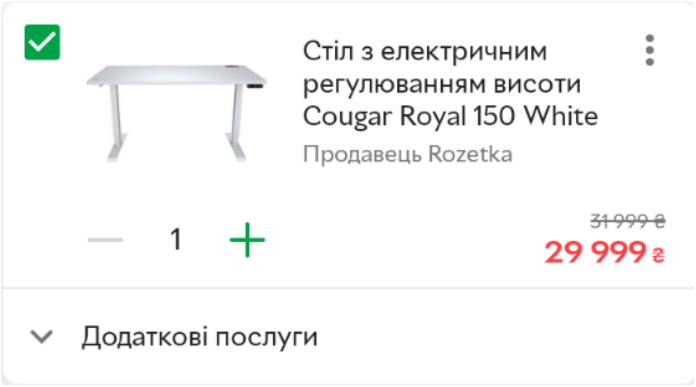

Рис. *30 Елемент списку у кошику*

На екрані кошика розташовані такі елементи: checkbox, який додає або прибирає всі товари у списку із замовлення, кількість обраних товарів зі списку, кнопка, яка дозволяє перемістити всі товари з кошика у список бажаного, кнопка видалення усіх товарів з кошика та кнопка «оформити замовлення», на якій вказано суму обраних товарів (див. Рис. 31).

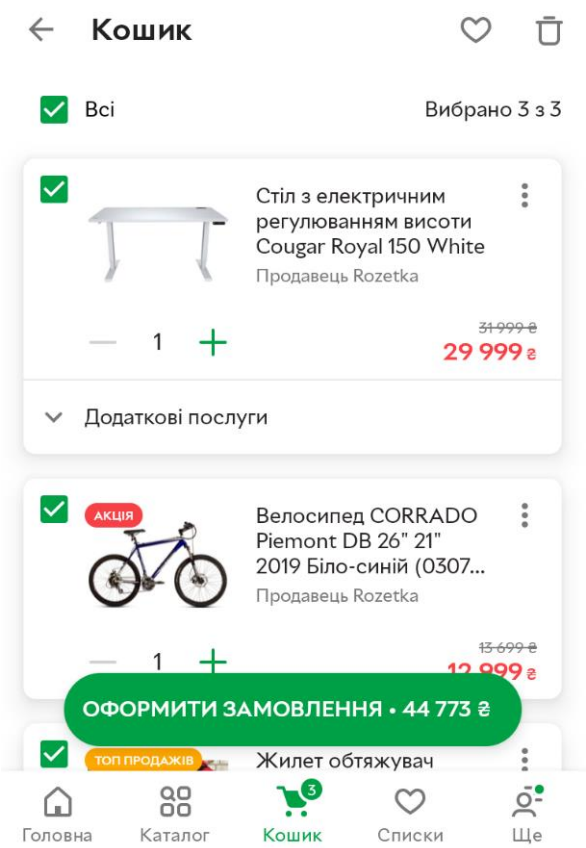

Рис. *31 Екран кошика*

Виконавши тап по кнопці «оформити замовлення» з’являється екран оформлення замовлення з кнопкою для вибору міста доставки, в якому бажане місце вводиться після тапу на відповідному полі та полями для вхідних даних, а саме: прізвище, ім’я та номер телефону (див. Рис. 32). Після натискання на кнопку «продовжити» замовлення підтверджується.

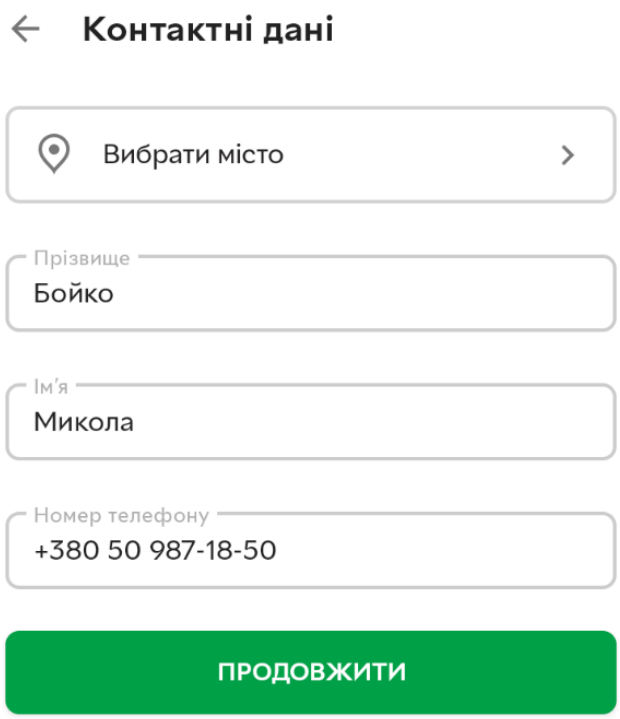

Рис. 32 *Екран оформлення замовлення*

Виконавши тап по елементу «Ще» на нижній панелі головного екрану відображаються такі елементи: зміна мови застосунку, вхід, реєстрація, мої замовлення, мої повідомлення, моє листування, мій гаманець, порівняння, переглянуті, знижки, акції, служба підтримки, адреси та час роботи, інформація, налаштування (див. Рис. 33).

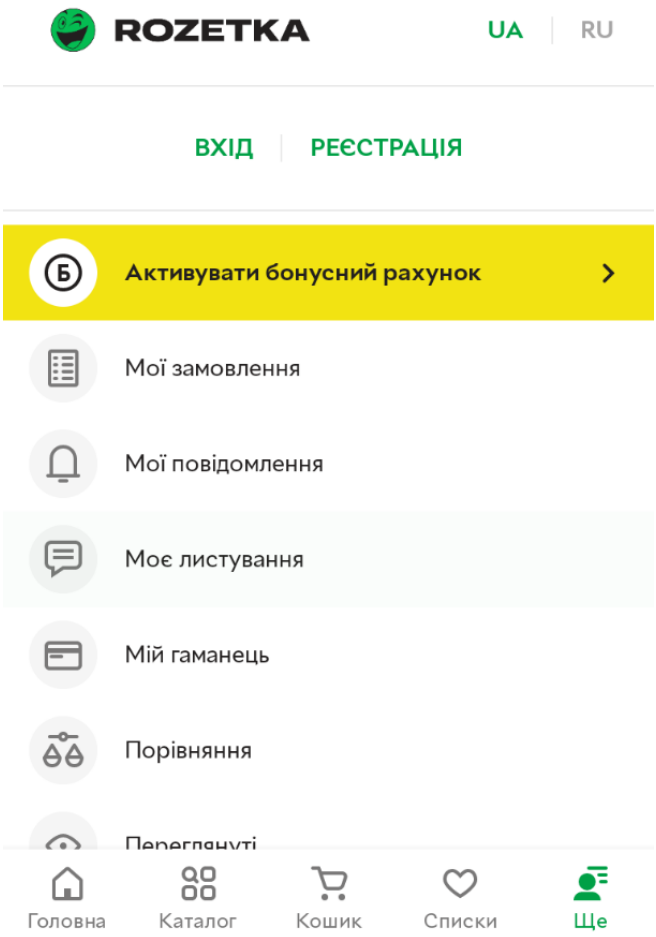

Рис. *33 Екран мобільного застосунку в меню «Ще»*

Мобільний застосунок підтримує дві мови, а саме українську – основну мову взаємодії з користувачем та російську.

Виконавши тап по кнопці «реєстрація» з’являється екран, який передбачає заповненням користувачем таких вхідних даних: прізвище, ім’я, номер телефону, електронна пошта та пароль (див. Рис. 34).

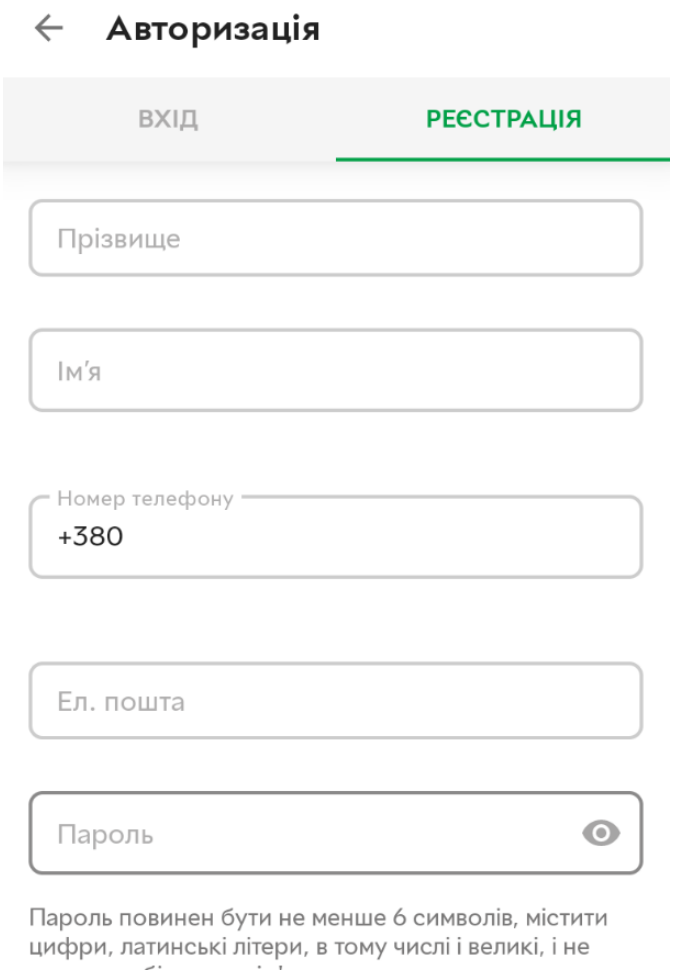

Рис. *34 Екран реєстрації облікового запису*

Виконавши тап по кнопці «вхід» з'являється екран, який передбачає заповненням користувачем таких вхідних даних: телефон або електронна пошта та пароль. Також є можливість увійти за допомогою аккаунта Facebook або Google (див. Рис. 35).

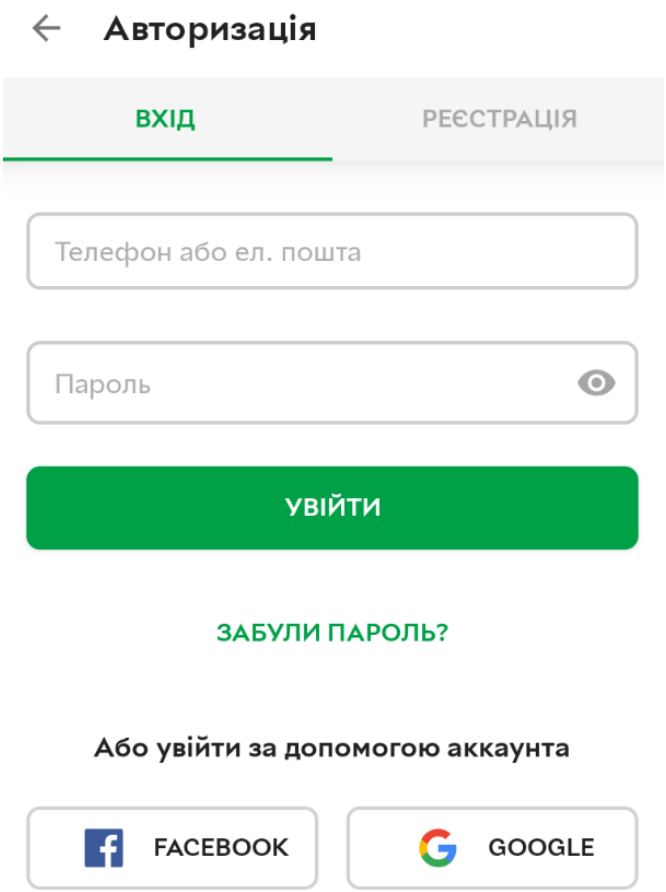

Рис. *35 Екран входу у обліковий запис*

Дослідження застосунку показало, що бажані товари обираються з дуже великого переліку зареєстрованих у системі товарів. Прізвище та ім’я користувача при реєстрації не може вводитись латиницею, номер телефону для реєстрації повинен мати український код +380.

В ході дослідження функціональності знайдені такі обмеження:

1. Замовлення товарів за межами покриття: якщо магазин Rozetka не доставляє товари до певного регіону або країни, користувачі з цього регіону не зможуть зробити замовлення на ці товари через застосунок. Обмеження доставки може бути зумовлено географічними обмеженнями або ліцензійними умовами.
2. Обмеження оплати: Rozetka може мати обмеження щодо доступних способів оплати. Наприклад, якщо користувач не має певного виду

платіжної картки або електронного гаманця, він не зможе використовувати цей спосіб оплати в застосунку.

3. Недоступність певних функцій для гостьових користувачів: деякі функціональні можливості застосунку, наприклад, залишення відгуків, питань та створення списків бажаного можуть бути доступні лише для зареєстрованих користувачів. Гостьові користувачі можуть мати обмежений доступ до певних функцій або можуть бути викликані до реєстрації акаунта.
4. Недоступність товарів: в деяких випадках товари, які були розміщені в застосунку, можуть бути відсутніми або недоступними на даний момент через тимчасову або постійну недоступність у магазині.

3.2 Розробка тест-вимог для мобільного застосунку

Для тестування функціоналу мобільного застосунку онлайн-магазину Rozetka можна запропонувати тестування пошуку товару, його відповідність до запропонованої у пошуку категорії, тестування реєстрації користувача, тестування сортування, тестування фільтрації, тестування оформлення замовлення користувачем. Сформовано відповідні тест-вимоги:

1. Перевірка ініціалізації застосунку (після первинного завантаження):
    a. Перевірити виведення спливаючого вікна, якщо воно є, закрити це вікно.
    b. Перевірити наявність на поточному екрані поля введення пошуку товарів.
2. Перевірка можливостей вхідних даних:
    2.1. Перевірити функцію пошуку на співпадіння товару з відповідною йому категорією:
        2.1.1. Виконати тап на полі введення назви товару на початковому екрані.
        2.1.2. В поле пошуку ввести бажану назву товару;

2.1.3. На екрані обрання категорій у пошуку обрати бажану категорію зі списку знайдених категорій (обрати перший елемент списку);

2.1.4. Перевірити співпадіння знайденої категорії з бажаною категорією.

2.2. Перевірка функції реєстрації користувача:

2.2.1. На головному екрані застосунку виконати тап на елемент «Ще»;

2.2.2. На екрані меню «Ще» виконати тап на елемент «реєстрація»;

2.2.3. На екрані реєстрації ввести прізвище користувача у поле «Прізвище»;

2.2.4. На екрані реєстрації ввести ім’я користувача у поле «Ім’я»;

2.2.5. На екрані реєстрації ввести номер телефону користувача у поле «Номер телефону»;

2.2.6. На екрані реєстрації ввести електронну пошту користувача у поле «Електронна пошта»;

2.2.7. На екрані реєстрації ввести пароль користувача у поле «Пароль»;

2.2.8. На екрані реєстрації виконати тап на елемент «зареєструватись»;

2.2.9. Перевірити, що з’явилось вікно з підтвердженням реєстрації.

2.3. Перевірка функції пошуку товару та оформлення замовлення:

2.3.1. Виконати тап на полі введення назви товару на початковому екрані;

2.3.2. В поле пошуку ввести бажану назву товару;

2.3.3. На екрані обрання категорій у пошуку обрати бажану категорію зі списку знайдених категорій (обрати перший елемент списку);

2.3.4. На екрані зі списком знайдених товарів за назвою та категорією виконати тап по бажаному товару (обрати перший елемент списку);

2.3.5. Здійснити перевірку відповідності обраного товару до того, який був шуканий, шляхом перевірки часткового збігу назви шуканого товару з назвою обраного товару;

2.3.6. Здійснити тап на елемент «відгуки» та перевірити функціональність свайпу у розділі;

2.3.7. Здійснити тап на елемент «характеристики» та порівняти серію продукту обраного товару з очікуваною;

2.3.8. Перевірити функціональність свайпу у розділі «характеристики»;

2.3.9. Здійснити подвійний тап на елемент кошику, перший тап додає товар у кошик, другий відкриває вікно кошика;

2.3.10. На екрані кошика здійснити тап на елемент «оформити замовлення»;

2.3.11. На екрані замовлення здійснити тап на елемент «Вибрати місто»;

2.3.12. На екрані «Вибір міста» обрати бажане місто;

2.3.13. На екрані замовлення ввести прізвище у поле «Прізвище»;

2.3.14. На екрані замовлення ввести ім’я у поле «Ім’я»;

2.3.15. На екрані замовлення ввести номер телефону у поле «Номер телефону»;

2.3.16. На екрані замовлення здійснити тап на елемент «Продовжити» для завершення оформлення замовлення.

2.4. Перевірка функції фільтрації та сортування списку товару:

2.4.1. На головному екрані застосунку здійснити тап на елемент «Каталог»;

2.4.2. На екрані «Каталог» здійснити тап на елемент категорії;

2.4.3. На екрані обраної категорії здійснити тап на елемент у списку запропонованих товарів за обраною категорією;

2.4.4. На екрані зі списком товарів здійснити тап на елемент «Фільтр»;

2.4.5. На екрані «Фільтр» здійснити тап на checkbox;

2.4.6. Запам'ятати кількість товарів у верхній частині екрану з фільтрами;

2.4.7. На екрані «Фільтр» застосувати інший фільтр, змінивши максимальну ціну товарів;

2.4.8. На екрані «Фільтр» здійснити тап на елемент «Застосувати», для оновлення кількості товарів у верхній частині екрану.

2.4.9. На екрані зі списком товарів здійснити тап на елемент «Фільтр»;

2.4.10. Запам'ятати оновлену кількість товарів у верхній частині екрану з фільтрами;

2.4.11. Перевірити, що кількість запропонованих товарів у верхній частині екрану стала меншою;

2.4.12. Запам’ятати мінімальну та максимальну ціну у фільтрах за ціною;

2.4.13. На екрані зі списком товарів здійснити тап на елемент «Сортування»;

2.4.14. На екрані «Сортування» обрати сортування «від дешевих до дорогих»;

2.4.15. Запам’ятати ціну першого елемента у відсортованому списку товарів;

2.4.16. Перевірити, що мінімальна ціна у фільтрах не перевищує ціну найдешевшого товару;

2.4.17. На екрані зі списком товарів здійснити тап на елемент «Сортування»;

2.4.18. На екрані «Сортування» обрати сортування «від дорогих до дешевих»;

2.4.19. Запам’ятати ціну першого елемента у відсортованому списку товарів;

2.4.20. Перевірити, що максимальна ціна товару не перевищує максимальну ціну у фільтрах;

3.3 Запис тестових сценаріїв у Appium Inspector та експортування їх до JUnit 5 тестів

Для розробки тестів для визначених тест-вимог для початку потрібно запустити Genymotion та віртуальний пристрій Android (Запускаємо створений CustomPhone) , після цього потрібно запустити Appium Server GUI і Appium Inspector (див. Рис. 36).

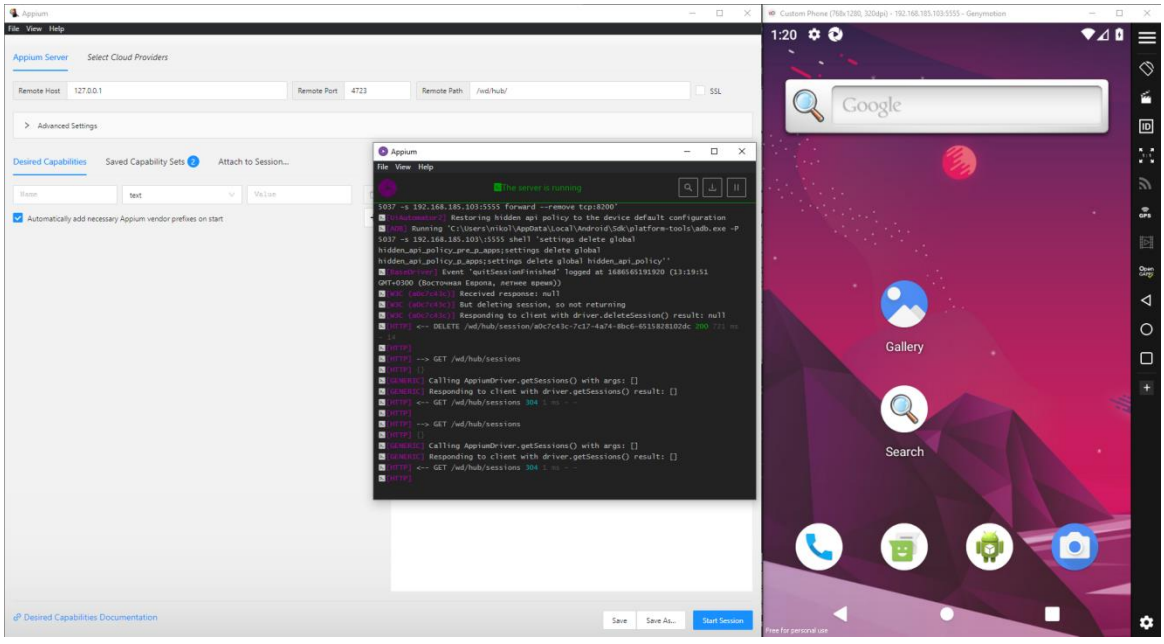

Рис. *36 Запуск середовища розробки тестів*

Тепер необхідно в Appium Inspector налаштувати сесію з сервером, вказавши для неї бажані можливості - Desired Capabilities (див. Рис. 37).

Визначаються такі бажані можливості, як назва платформи, її версія, драйвер, ім’я пристрою та повний шлях до застосунку, який буде тестуватись. Після цього можна запускати сесію Appium Inspector кліком по кнопці Start Session.

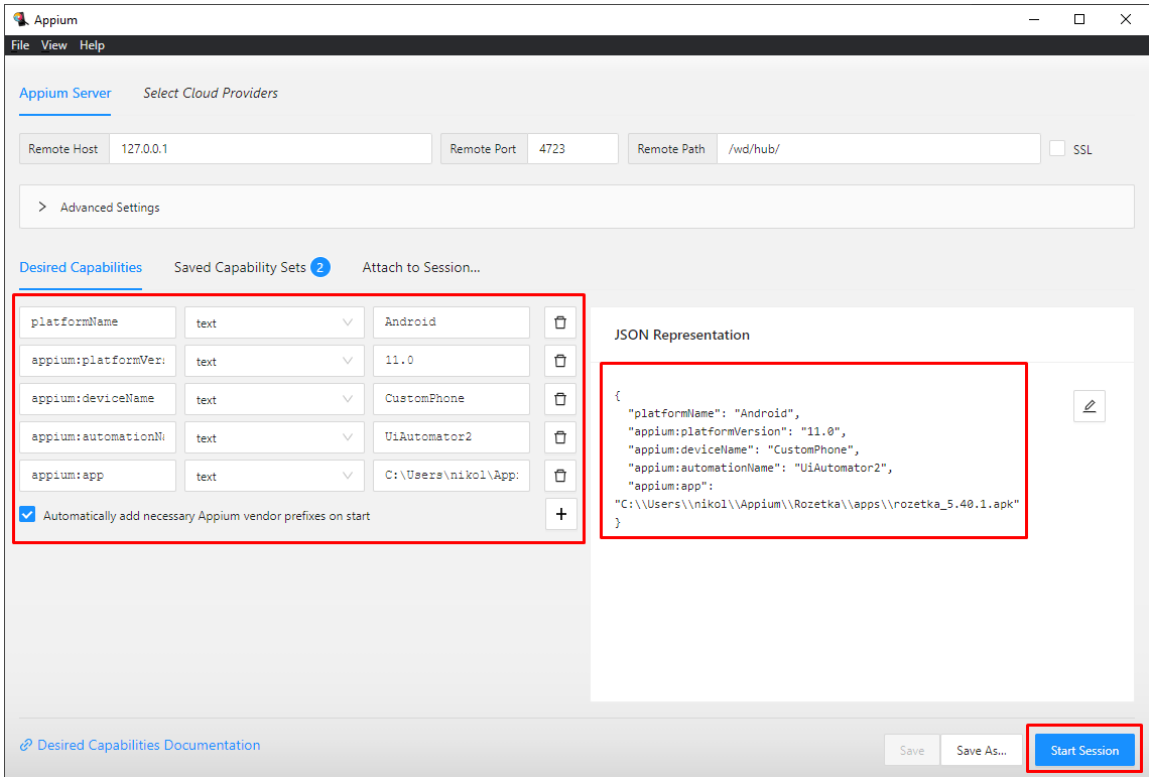

Рис. *37 Визначення бажаних можливостей для сесії в Appium Inspector*

Перебіг запуску сесії можна спостерігати у вікні Appium Server – туди виводяться службові повідомлення, у тому числі і про інсталяцію застосунку до мобільного пристрою. Після запуску сесії у лівій частині вікна Appium Inspector з’явиться інтерфейс застосунку, ідентичний інтерфейсу на емуляторі Android (див. Рис. 38).

Вікно Appium Inspector поділене на три частини: у лівій відображається поточний інтерфейс мобільного застосунку, у середній – код XML-розмітки інтерфейсу застосунк, а у правій – значення локаторів та атрибутів виділеного

елементу інтерфейсу (див. Рис. 39). Appium Inspector надає засоби роботи з виділеними елементами інтерфейсу мобільного застосунку – кнопки на панелі дій з виділеним елементом (Рис. 39). Це кнопка Tap для тапу по елементу, кнопка Send Keys для введення до елементу символів (тексту), кнопка Clear для очищення елемента, кнопка Copy Attributes to Clipboard для копіювання атрибутів та кнопка Get Timing, що дозволяє виміряти час завантаження елемента за кожною з його стратегій пошуку (за кожним з типів локаторів).

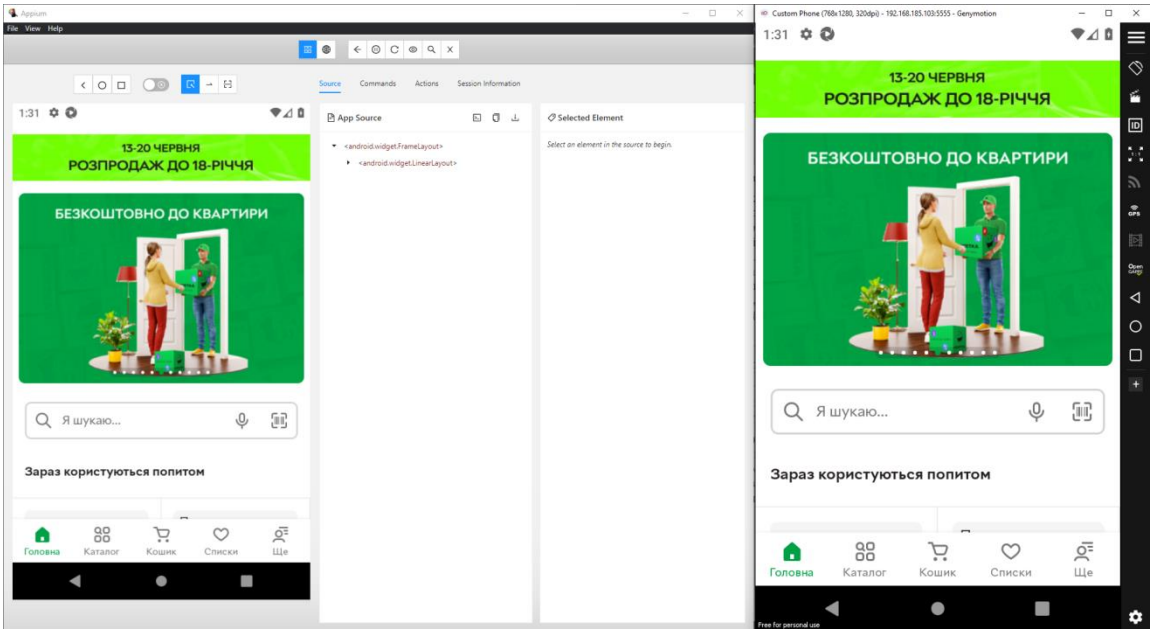

Рис. *38 Відображення інтерфейсу застосунку у Appium Inspector та емуляторі*

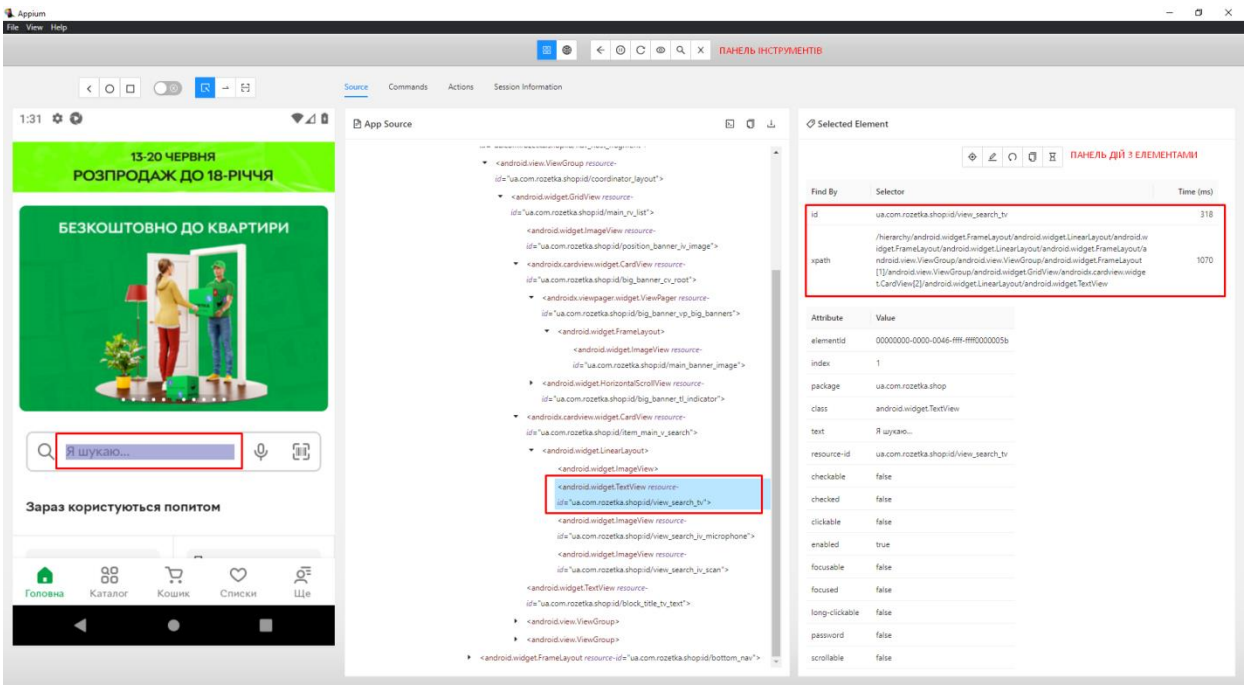

Рис. *39 Елементи управління Appium Inspector*

Розглянемо також елементи управління Панелі інструментів у верхній частині вікна Appium Inspector. Перша група елементів дозволяє перемикати режим роботи застосунку між нативним (Native App Mode) та Веб або гібридним (Web/Hybrid App Mode) відповідно до типу застосунку, що тестується (зараз тестується нативний застосунок).

Друга група елементів управління дозволяє обрати режим роботи з застосунком: виділення елемента інтерфейсу (Select Elements), режим свайпу (Swipe By Coordinates) відповідно лінії, яку можливо намалювати у цьому режимі на зображенні інтерфейсу у лівому вікні Appium Inspector та режим тапу (Tap By Coordinates) у точці, вказаній координатами на зображенні інтерфейсу.

Третя група елементів управління має кнопку, що імітує кнопку повернення на попередній екран (Back), кнопку призупинення синхронізації зображення інтерфейсу (Pausing refreshing source), кнопку синхронізації

зображення інтерфейсу у лівому вікні Appium Inspector з поточним інтерфейсом мобільного емулятора, на якому запущений застосунок, що тестується (Refresh Source & Screenshot), кнопку початку запису дій користувача (Start Recording), кнопку пошуку елементу інтерфейсу (Search for element) за значенням селектора обраного зі списку типу локатора та кнопку закриття сесії (Quit Session & Close Inspector).Виконаємо запис дій користувача у Appium Inspector, для цього виконаємо клік по кнопці (Start Recording) на панелі інструментів (див. Рис. 40).

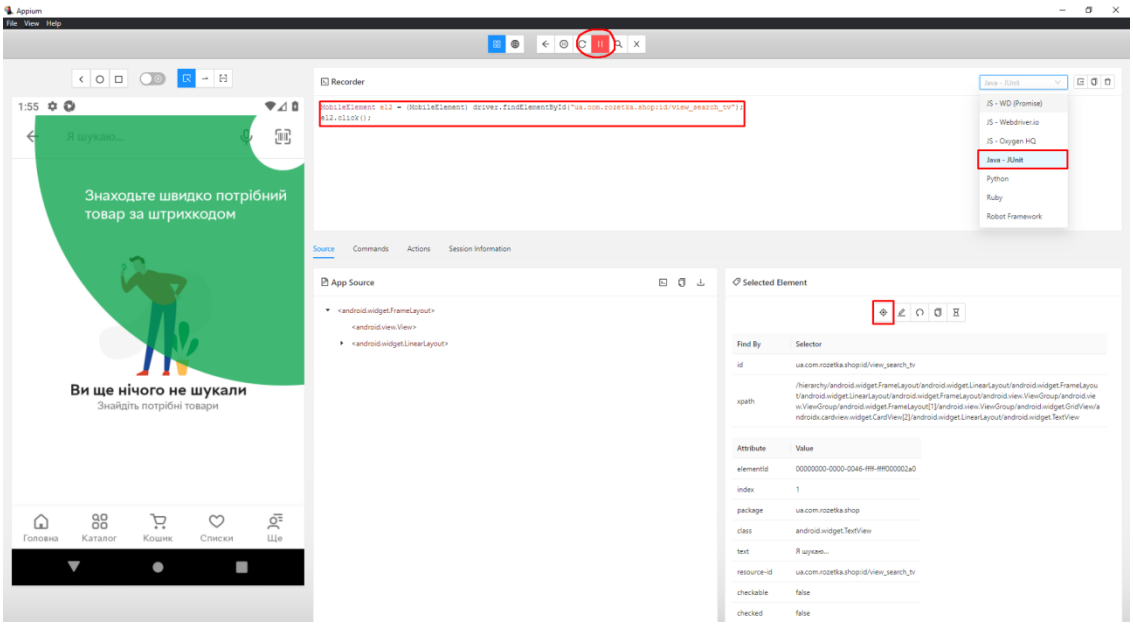

Рис. *40 Запис дій користувача у Appium Inspector*

Після виконаних дій, зупиняємо запис тестового скрипту, виконаємо клік по кнопці (Show/Hide Boilerplate Code) та скопіюємо його за допомогою кнопки (Copy code to clipboard). Записаний тест імітує пошук телефону Samsung Galaxy S23, перегляд характеристик та оформлення замовлення. Вміст записаного тестового скрипту описаний у лістинг 2.

Лістинг *2 Експортований тестовий скрипт з Appium Inspector*

```
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import junit.framework.TestCase;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;

public class SampleTest {

  private AndroidDriver driver;

  @Before
  public void setUp() throws MalformedURLException {
    DesiredCapabilities desiredCapabilities = new
DesiredCapabilities();
    desiredCapabilities.setCapability("platformName",
"Android");

desiredCapabilities.setCapability("appium:platformVersion",
"11.0");
    desiredCapabilities.setCapability("appium:deviceName",
"CustomPhone");

desiredCapabilities.setCapability("appium:automationName",
"UiAutomator2");
    desiredCapabilities.setCapability("appium:app",
"C:\\Users\\nikol\\Appium\\Rozetka\\apps\\rozetka_5.40.1.apk"
);
```

```
desiredCapabilities.setCapability("appium:ensureWebviewsHaveP
ages", true);

desiredCapabilities.setCapability("appium:nativeWebScreenshot
", true);

desiredCapabilities.setCapability("appium:newCommandTimeout",
3600);

desiredCapabilities.setCapability("appium:connectHardwareKeyb
oard", true);

    URL remoteUrl = new URL("http://127.0.0.1:4723/wd/hub/");

    driver = new AndroidDriver(remoteUrl,
desiredCapabilities);
  }

  @Test
  public void sampleTest() {
    MobileElement el6 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/view_search_tv
");
    el6.click();
    MobileElement el7 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/et_query");
    el7.sendKeys("Samsung galaxy s23");
    el7.click();
    MobileElement el8 = (MobileElement)
driver.findElementByXPath("/hierarchy/android.widget.FrameLay
out/android.widget.LinearLayout/android.widget.FrameLayout/an
droid.widget.LinearLayout/android.widget.FrameLayout/android.
```

```
view.ViewGroup/android.view.ViewGroup/android.widget.FrameLay
out[1]/android.view.ViewGroup/androidx.recyclerview.widget.Re
cyclerView/androidx.cardview.widget.CardView[1]/android.widge
t.RelativeLayout/android.widget.TextView");
    el8.click();
    (new TouchAction(driver))
      .press(PointOption.point(381, 1009}))
      .moveTo(PointOption.point(389, 258}))
      .release()
      .perform();
    MobileElement el9 = (MobileElement)
driver.findElementByXPath("/hierarchy/android.widget.FrameLay
out/android.widget.LinearLayout/android.widget.FrameLayout/an
droid.widget.LinearLayout/android.widget.FrameLayout/android.
view.ViewGroup/android.view.ViewGroup/android.widget.FrameLay
out[1]/androidx.drawerlayout.widget.DrawerLayout/android.widg
et.LinearLayout/android.widget.ScrollView/android.widget.Grid
View/android.view.ViewGroup[1]/androidx.cardview.widget.CardV
iew[1]/androidx.viewpager.widget.ViewPager/androidx.recyclerv
iew.widget.RecyclerView/android.widget.FrameLayout/android.wi
dget.ImageView");
    el9.click();
    MobileElement el10 = (MobileElement)
driver.findElementByXPath("//android.widget.LinearLayout[@con
tent-desc=\"Характеристики\"]/android.widget.TextView");
    el10.click();
    (new TouchAction(driver))
      .press(PointOption.point(352, 1046}))
      .moveTo(PointOption.point(361, 160}))
      .release()
      .perform();
    MobileElement el11 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/iv_cart");
```

```
    el11.click();
    MobileElement el12 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/iv_cart");
    el12.click();
    MobileElement el13 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/fab_checkout")
;
    el13.click();
    MobileElement el14 = (MobileElement)
driver.findElementByXPath("/hierarchy/android.widget.FrameLay
out/android.widget.LinearLayout/android.widget.FrameLayout/an
droid.widget.LinearLayout/android.widget.FrameLayout/android.
view.ViewGroup/android.view.ViewGroup/android.widget.FrameLay
out/android.view.ViewGroup/android.widget.ScrollView/android.
widget.LinearLayout/androidx.cardview.widget.CardView[2]/andr
oid.widget.LinearLayout/android.widget.LinearLayout");
    el14.click();
    MobileElement el15 = (MobileElement)
driver.findElementByXPath("/hierarchy/android.widget.FrameLay
out/android.widget.LinearLayout/android.widget.FrameLayout/an
droid.widget.LinearLayout/android.widget.FrameLayout/android.
view.ViewGroup/android.view.ViewGroup/android.widget.FrameLay
out/android.view.ViewGroup/android.widget.FrameLayout/android
x.recyclerview.widget.RecyclerView/android.widget.LinearLayou
t[5]/android.widget.TextView");
    el15.click();
    MobileElement el16 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/et_last_name")
;
    el16.sendKeys("Бойко");
    MobileElement el17 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/et_first_name"
);
```

```
    el17.sendKeys("Микола");
    MobileElement el18 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/et_phone");
    el18.sendKeys("509871350");
    (new TouchAction(driver))
      .press(PointOption.point(359, 1078}))
      .moveTo(PointOption.point(366, 195}))
      .release()
      .perform();
    MobileElement el19 = (MobileElement)
driver.findElementById("ua.com.rozetka.shop:id/b_continue");
    el19.click();
  }

  @After
  public void tearDown() {
    driver.quit();
  }
}
```

Наступим кроком є створення проекту Gradle в Intellij IDEA. Необхідно обрати команду File – New project, зазначити назву проекту, обрати створений каталог, обрати мову програмування Java, Build System обираємо Gradle. Gradle – це система автоматичного збирання проектів, яка розвиває принципи, закладені до систем автоматичного збирання проектів Apache Ant та Apache Maven і використовує предметно-орієнтовану мову (Domain-Specific Language – DSL) на основі мов програмування Groovy або Kotlin замість традиційної XML-подібної форми представлення конфігурації проєкту. Наступним обираємо JDK, Gradle DSL (Groovy Domain Specific Language) обираємо Groovy, також зазначаємо у GroupId - ua.edu.znu (див. Рис. 41).

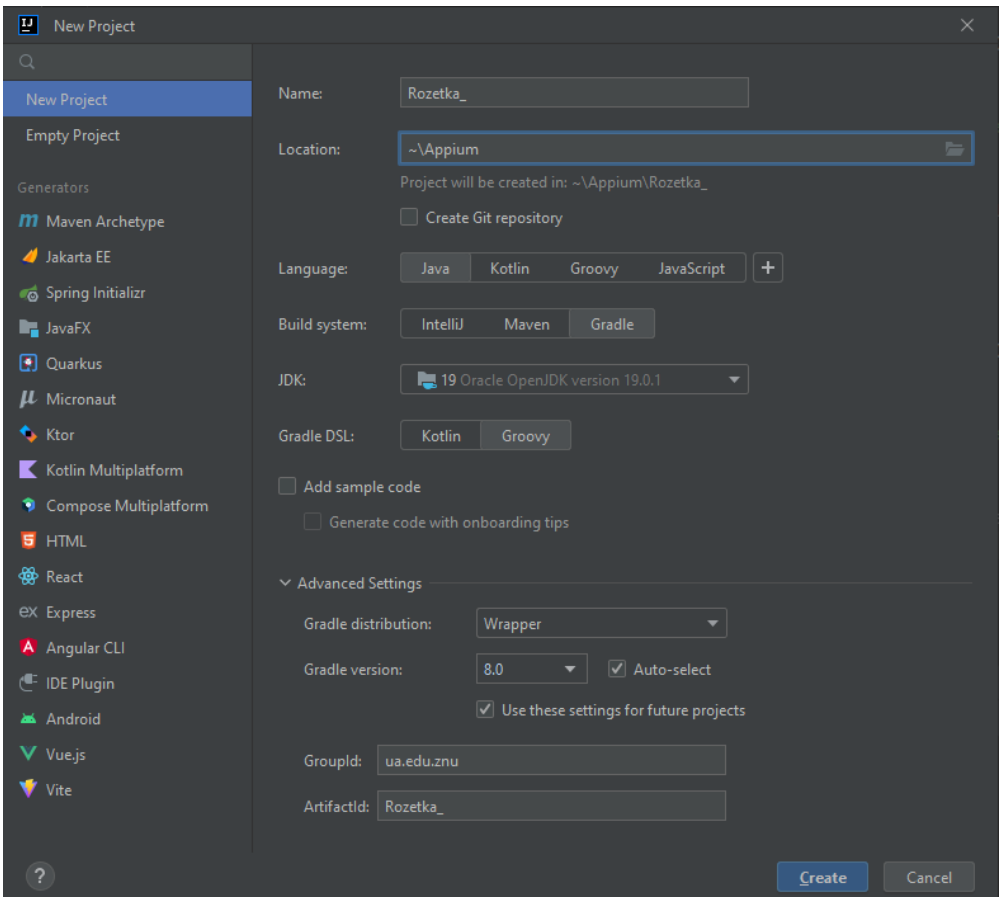

Рис. *41 Створення проекту типу Gradle у Intellij IDEA*

Після кліку по кнопці Finish в IDE буде завантажений проект і відкриється вікно з файлом на мові програмування Groovy build.gradle, який знаходиться у кореневому каталозі проекту і містить конфігурацію проекту (він є аналогом файла pom.xml для Maven-проектів) (див. Рис. 42).

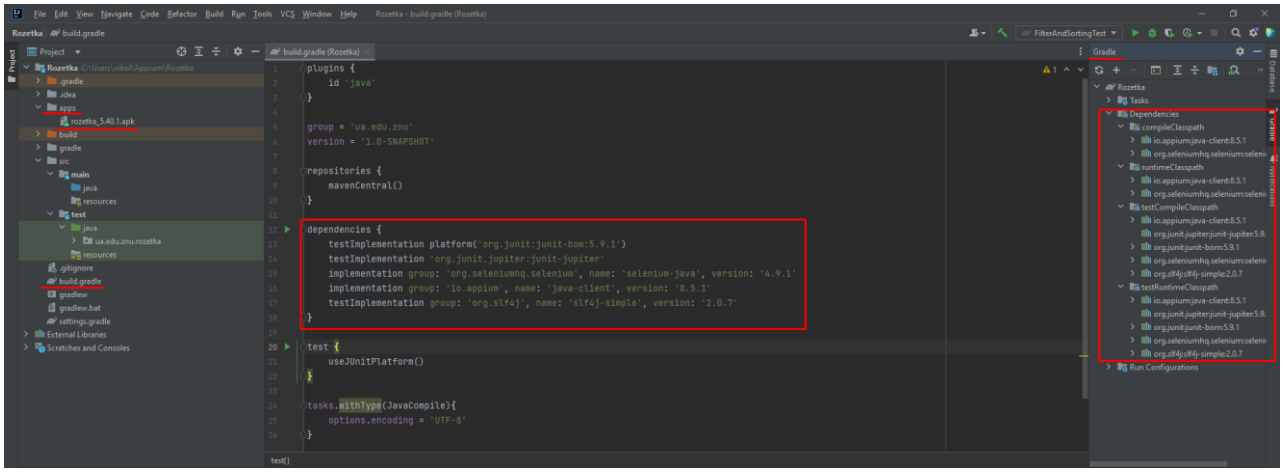

Рис. *42 Структура Gradle-проекту та вміст файлу build.gradle*

При створенні проекту автоматично формується файл build.gradle, у ньому вже є залежності фреймворка модульного тестування JUnit 5, до нього додані залежності останніх на момент написання модуля версій: selenium-java, Appium java-client, а також залежність slf4j-simple. Вікно справа на рисунку 39 це інструмент IntelliJ IDEA, який відображає задачі Gradle та підключені бібліотеки залежностей. У стандартній структурі проекту був створений каталог apps, у який був скопійований файл мобільного застосунку, що тестується, а також у каталозі src\test\java створений пакет ua.edu.znu.rozetka, до якого скопійований файл з вихідним кодом тесту SearchAndBuyProductTest.java, експортований з Appium Inspector. Поточна версія Appium Inspector виконує експорт, орієнтуючись на попередню версію фреймворку JUnit та на попередні 7 версії Appium java-clint. В результаті експортований скрипт можна вважати лише заготовкою для подальшого рефакторингу, але можливість автоматичного запису скрипту за діями користувача та автоматичне визначення локаторів інтерфейсу застосунку є дуже важливим.

3.4 Рефакторинг JUnit 5 тестів

Перехід до версії 8.0.0 Appium java-client призвів до значних змін у коді, що вимагає змін у розробці тестів. Головною метою цього переходу було забезпечити відповідність засобів Appium до актуальної 4 версії Selenium, яка строго відповідає стандарту W3C WebDriver. Було змінено спосіб конфігурування драйвера з використанням об'єкта DesiredCapabilities на створення об'єктів класів UiAutomator2Options для Android. Ці класи мають методи для встановлення бажаних можливостей. Також були змінені сигнатури методів пошуку елементів інтерфейсу, де методи findBy_тип_локатора були замінені на findElement[s](By.тип_локатора або AppiumBy.тип_локатора).

Крім того, було видалено деякі класи, наприклад, DefaultGenericMobileElement разом з його нащадками MobileElement,

IOSElement, AndroidElement, замість яких рекомендується використовувати стандартний клас WebElement. Також класи TouchAction та MultiTouchAction, що використовувались для автоматизації жестів на мобільних пристроях, були оголошені застарілими й рекомендується використовувати альтернативні підходи.

Спочатку був доданий оператор оголошення пакету, замінені анотації JUnit 4 на JUnit 5 та виконано заміну коду створення об'єкта DesiredCapabilites на об'єкт UiAutomator2Options, який розміщено в методі-фікстурі SetUp(), анотованому @BeforeAll (див. Лістинг 3).

Лістинг *3 Метод SetUp() після рефакторингу*

```
@BeforeAll
    public static void setUp() throws MalformedURLException {
        UiAutomator2Options options = new
UiAutomator2Options()
                .setPlatformName("Android")
                .setPlatformVersion("11.0")
                .setDeviceName("CustomPhone")
                .setAutomationName("UiAutomator2")

.setApp("C:\\Users\\nikol\\Appium\\Rozetka\\apps\\rozetka_5.4
0.1.apk")
                .eventTimings();
        URL remoteUrl = new
URL("http://127.0.0.1:4723/wd/hub/");
        driver = new AndroidDriver(remoteUrl, options);
}
```

Було виконано заміну видаленого у 8 версії java-client типу MobileElement на WebElement та корекцію методів пошуку елементів на актуальну для java-client 8 версії форму, наприклад, замість

MobileElement el6 = (MobileElement) driver.findElementById("ua.com.rozetka.shop:id/view_search_tv");

Тепер використовується

```
WebElement el6 = driver.findElement(AppiumBy.id("\"ua.com.rozetka.shop:id/view_search_tv\""));
```

Замість

```
MobileElement el8 = (MobileElement) driver.findElementByXPath("/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGroup/android.view.ViewGroup/android.widget.FrameLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/androidx.cardview.widget.CardView[1]/android.widget.RelativeLayout/android.widget.TextView");
```

Тепер використовується

```
WebElement el8 = driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGroup/android.view.ViewGroup/android.widget.FrameLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/androidx.cardview.widget.CardView[1]/android.widget.RelativeLayout/android.widget.TextView"))
```

Було виконано заміну застарілого класу TouchAction на код із використанням W3CActions API для моделювання свайпу, організація свайпу по елементу була винесена до окремого методу, оскільки свайп може використовуватись декілька разів (див. Лістинг 4).

Лістинг *4 Метод реалізації свайпу у вертикальному напрямку вниз*

```
/**
```

```
     * This method is used to make vertical swipe down on
element.
     *
     * @param element element that used for swipe
     */
    private static void verticalSwipeDown(WebElement element)
{
        int centerX = element.getRect().x +
(element.getSize().width/2);
        double startY = element.getRect().y +
(element.getSize().height * 0.9);
        double endY = element.getRect().y +
(element.getSize().height * 0.1);

        PointerInput finger = new
PointerInput(PointerInput.Kind.TOUCH, "finger");
        Sequence swipe = new Sequence(finger, 1);

swipe.addAction(finger.createPointerMove(Duration.ofSeconds(0
), PointerInput.Origin.viewport(), centerX, (int)startY));
        swipe.addAction(finger.createPointerDown(0));

swipe.addAction(finger.createPointerMove(Duration.ofMillis(70
0),
                PointerInput.Origin.viewport(), centerX,
(int)endY));
        swipe.addAction(finger.createPointerUp(0));

        driver.perform(Arrays.asList(swipe));
    }
```

До тесту додані оператори ствердження JUnit 5, які перевірятимуть, що отриманий результат збігається з очікуваним. Наприклад, для перевірки що обраний товар має схожість у назві з товаром, який шукався (див. Лістинг 5).

Лістинг *5 Перевірка, що назва обраного товару містить очікуваний частковий текст товару, який шукався*

```
String expectedPartialProductNameText = "Samsung Galaxy S23";
        WebElement firstSearchedProduct =
driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.
FrameLayout/android.widget.LinearLayout/android.widget.FrameL
ayout/android.widget.LinearLayout/android.widget.FrameLayout/
android.view.ViewGroup/android.view.ViewGroup/android.widget.
FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout/andr
oid.widget.LinearLayout/android.widget.ScrollView/android.wid
get.GridView/android.view.ViewGroup[1]/androidx.cardview.widg
et.CardView[1]/androidx.viewpager.widget.ViewPager/androidx.r
ecyclerview.widget.RecyclerView/android.widget.FrameLayout/an
droid.widget.ImageView"));
        firstSearchedProduct.click();
        WebElement productName =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/tv_tit
le"));
        String productNameText = productName.getText();

Assertions.assertTrue(productNameText.contains(expectedPartia
lProductNameText), "Element text does not contain the
expected partial text.");
```

Був доданий метод, який закриває початковий банер при першій ініціалізації застосунку (див. Лістинг 6).

Лістинг *6 Код методу, який закриває банер при першій ініціалізації застосунку*

```
/**
     * This method is used to close the first banner that
appears on the first app launch.
     */
```

```
private static void closeFirstBanner() {
        driver.manage().timeouts().implicitlyWait(20,
TimeUnit.SECONDS);
        if
(!driver.findElements(AppiumBy.xpath("/hierarchy/android.widg
et.FrameLayout/android.widget.LinearLayout/android.widget.Fra
meLayout/android.widget.FrameLayout/android.widget.FrameLayou
t/androidx.appcompat.widget.LinearLayoutCompat/android.widget
.FrameLayout/android.widget.FrameLayout/android.widget.FrameL
ayout/android.widget.ImageView[2]")).isEmpty()) {
            driver.navigate().back();
        }
    }
```

Був доданий метод, який закриває спливаюче вікно скану штрихкода при першому натисканні на поле пошуку на головній сторінці (див. Лістинг 7).

Лістинг *7 Метод, який закриває вікно скану штрихкода*

```
/**
     * This method is used to close the IV scan
functionality.
     */
    public static void CloseIVScan() {
        WebElement ivScanButton =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/iv_sca
n"));
        ivScanButton.click();
        WebElement permissionButtonDeny =
driver.findElement(AppiumBy.id("com.android.permissioncontrol
ler:id/permission_deny_button"));
        permissionButtonDeny.click();
```

```
        WebElement canselButton =
driver.findElement(AppiumBy.id("android:id/button2"));
        canselButton.click();
        WebElement closeIVScan =
driver.findElement(AppiumBy.accessibilityId("Перейти
вгору"));
        closeIVScan.click();
    }
```

Для тесту реєстрації користувача був доданий метод, який перевіряє, чи виник елемент після натискання кнопки (див. Лістинг 8).

Лістинг *8 Метод, який перевіряє чи присутній елемент на сторінці за заданим локатором*

```
/**
     * Checks if an element is present on the page based on
the given locator.
     *
     * @param locator the locator used to find the element
     * @return true if the element is present, false
otherwise
     */
    private boolean isElementPresent(By locator) {
        try {
            Thread.sleep(3000);
            driver.findElement(locator);
            return true;
        } catch (NoSuchElementException |
InterruptedException ex) {
            return false;
        }
    }
```

Наступним етапом рефакторингу був розподіл тесту на 6 тестових методів: перший виконує пошук товару за назвою, другий – перевіряє чи збігається назва обраного першого товару зі списку з тим, який шукався, третій – тестує функціональність кнопки «відгуки» та тестує скролл по розділу, четвертий – тестує функціональність кнопки «характеристики» та перевіряє, чи збігається серія обраного товару з очікуваним, п’ятий – тестує скролл по розділу «характеристики», шостий – тестування оформлення замовлення обраного товару.

Останнім було визначити змінні-рядки для очікуваних значень та перейменування змінних для елементів інтерфейсу так, щоб вони описували свій вміст та додані пояснювальні коментарі.

Після рефакторингу можна виділити основні тестові методи. На лістингу 9 наведений код методу пошуку товару та методу перевірки правильності пошуку. У методі пошуку товару спочатку виконується тап на елемент пошуку та закривається сканер штрих-коду. У поле пошуку вводиться product name (Samsung Galaxy S23) та виконується тап по першому знайденому елементу. У методі перевірки правильності пошуку записується назва товару, який було обрано та перевіряється, чи є у назві товару очікувана часткова назва товару. (Samsung Galaxy S23).

Лістинг *9 Методи пошуку товару та перевірки правильності пошуку*

```
/**
     * This test case performs a product search.
     * It searches for the product "Samsung galaxy s23" and
clicks on the first search result.
     * It then scrolls down to the section with product
results.
     */
    @Test
    @Order(1)
```

```
    public void SearchProductTest() {
        String productName = "Samsung Galaxy S23";
        WebElement searchButton =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/view_s
earch_tv"));
        searchButton.click();
        /*Close IV Scan the first time search field is used*/
        CloseIVScan();
        WebElement searchField =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/et_que
ry"));
        searchField.sendKeys(productName);
        WebElement firstSearchedResult =
driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.
FrameLayout/android.widget.LinearLayout/android.widget.FrameL
ayout/android.widget.LinearLayout/android.widget.FrameLayout/
android.view.ViewGroup/android.view.ViewGroup/android.widget.
FrameLayout[1]/android.view.ViewGroup/androidx.recyclerview.w
idget.RecyclerView/androidx.cardview.widget.CardView[1]/andro
id.widget.RelativeLayout/android.widget.TextView"));
        firstSearchedResult.click();
        WebElement swipeElementProductResults =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/sectio
n_rv_offers"));
        verticalSwipeDown(swipeElementProductResults);
    }
    /**
     * This test case verifies if the searched product is
correct.
     * It expects the partial product name "Samsung Galaxy
S23" to be present in the search result.
     * The test clicks on the first searched product and
retrieves its name.
```

```
     * It then compares the retrieved name with the expected
partial product name and asserts if it contains the expected
text.
     * If the assertion fails, it means that the element text
does not contain the expected partial text.
     */
    @Test
    @Order(2)
    public void SearchedProductIsCorrectTest() {
        String expectedPartialProductNameText = "Samsung
Galaxy S23";
        WebElement firstSearchedProduct =
driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.
FrameLayout/android.widget.LinearLayout/android.widget.FrameL
ayout/android.widget.LinearLayout/android.widget.FrameLayout/
android.view.ViewGroup/android.view.ViewGroup/android.widget.
FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout/andr
oid.widget.LinearLayout/android.widget.ScrollView/android.wid
get.GridView/android.view.ViewGroup[1]/androidx.cardview.widg
et.CardView[1]/androidx.viewpager.widget.ViewPager/androidx.r
ecyclerview.widget.RecyclerView/android.widget.FrameLayout/an
droid.widget.ImageView"));
        firstSearchedProduct.click();
        WebElement productName =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/tv_tit
le"));
        String productNameText = productName.getText();

Assertions.assertTrue(productNameText.contains(expectedPartia
lProductNameText), "Element text does not contain the
expected partial text.");
    }
```

Ще одним з основних методів тесту є метод, який виконує замовлення товару. На сторінці товару виконується подвійний тап на елемент кошику, перший тап додає товар у кошик, другий перенаправляє користувача до сторінки замовлення товару. Після тапу на елемент «Оформити замовлення». Виконується тап на елемент «Вибір міста» та обирається місто доставки. Далі на екрані «Контактні дані» заповнюється інформація про прізвище, ім’я та номер телефону для замовлення. Після введення даних виконується тап по кнопці «продовжити» та перевіряється, чи з’являється вікно з підтвердженням замовлення. Код методу описаний у лістингу 10.

Лістинг *10 Метод оформлення замовлення товару*

```
/**
     * This test case performs an order product test.
     * It clicks on the cart button twice to navigate to the
cart view.
     * Then clicks on the order product button to proceed to
the checkout process.
     * It selects the destination for the order by choosing a
city.
     * The test enters the last name, first name, and phone
number for the order.
     * It performs a vertical scroll down on the contact data
section.
     * Finally, the test clicks on the confirm order button
to complete the order process.
     */
    @Test
    @Order(6)
    public void OrderProductTest() {
        String lastName = "Бойко";
        String firstName = "Микола";
        String phoneNumber = "509861320";
```

```
        WebElement cartButton =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/iv_car
t"));
        cartButton.click();
        cartButton.click();
        WebElement orderProduct =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/fab_ch
eckout"));
        orderProduct.click();
        WebElement chooseDestinationButton =
driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.
FrameLayout/android.widget.LinearLayout/android.widget.Frame
ayout/android.widget.LinearLayout/android.widget.FrameLayout/
android.view.ViewGroup/android.view.ViewGroup/android.widget.
FrameLayout/android.view.ViewGroup/android.widget.ScrollView/
android.widget.LinearLayout/androidx.cardview.widget.CardView
[2]/android.widget.LinearLayout/android.widget.LinearLayout")
);
        chooseDestinationButton.click();
        WebElement cityDestination =
driver.findElement(AppiumBy.xpath("/hierarchy/android.widget.
FrameLayout/android.widget.LinearLayout/android.widget.Frame
ayout/android.widget.LinearLayout/android.widget.FrameLayout/
android.view.ViewGroup/android.view.ViewGroup/android.widget.
FrameLayout/android.view.ViewGroup/android.widget.FrameLayout
/androidx.recyclerview.widget.RecyclerView/android.widget.Lin
earLayout[5]/android.widget.TextView"));
        cityDestination.click();
        WebElement lastNameField =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/et_las
t_name"));
        lastNameField.sendKeys(lastName);
```

```
        WebElement firstNameField =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/et_fir
st_name"));
        firstNameField.sendKeys(firstName);
        WebElement phoneNumberField =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/et_pho
ne"));
        phoneNumberField.sendKeys(phoneNumber);
        WebElement swipeElementOrder =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/contac
t_data_nsv_scroll"));
        verticalSwipeDown(swipeElementOrder);
        WebElement confirmOrder =
driver.findElement(AppiumBy.id("ua.com.rozetka.shop:id/b_cont
inue"));
        confirmOrder.click();
        if
(isElementPresent(By.id("ua.com.rozetka.shop:id/parentPanel")
)){
            Assertions.assertTrue(true);
        }
    }
```

Результати тесту можна побачити на рисунку 43.

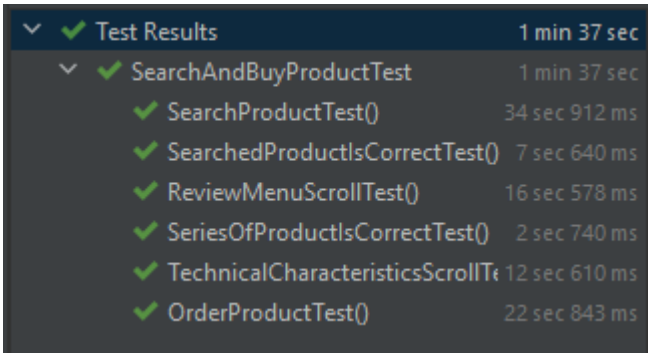

Рис. *43 Результати тесту для перевірки визначеної функціональності застосунку*

Параметризовані тести були розроблені у тесті реєстрації (див. Лістинг 11) та тесті правильності підібраної категорії для шуканих товарів (див. Лістинг 12).

Лістинг *11 Параметризований тест для перевірки реєстрації користувача*

```
@ParameterizedTest (name = "last name: {0}, first name: {1}, phoneNumber: {2}, mail: {3}, password: {4}")
    @Order(2)
    @CsvSource({
            "Бойко, Микола, 509871350, yourusername1@gmail.com, Password1",
            "Ігнатенко, Данило, 966047217, yourusername1@hotmail.com, Password2",
            "Шевченко, Сергій, 683180433, yourusername1@outlook.com, Password3",
            "Шостак, Олександр, 639571063, yourusername1@icloud.com, Password4"
    })
    void registrationTest(String lastName, String firstName, String phoneNumber, String mail, String password)
```

Даний тест був розроблений, щоб перевірити які електронні пошти та які мобільні оператори можуть бути використані для реєстрації. Результати тесту можна побачити на рисунку 44.

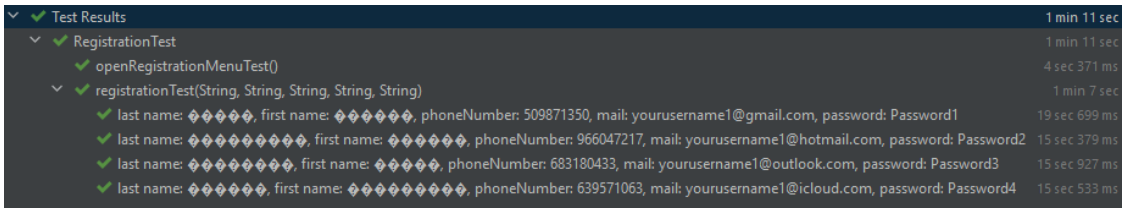

Рис. *44 Результати параметризованого тесту реєстрації користувача*

Лістинг *12 Параметризований тест для перевірки правильності підібраної категорії до товару*

```
@ParameterizedTest (name = "Product: {0}, Category: {1}")
    @Order(2)
    @CsvSource({
            "Відеокарта RTX 3060 Ti, Відеокарти",
            "Монітор 144 гц, Монітори",
            "Блок живлення 500 вт, Блоки живлення"
    })
    void categoryCorrectnessTest (String searchableProduct,
String expectedCategory )
```

Даний тест був розроблений, щоб перевірити, що пошук мобільного застосунку запропоновує правильні категорії для певного товару, який шукається. Результати тесту можна побачити на рисунку 45.

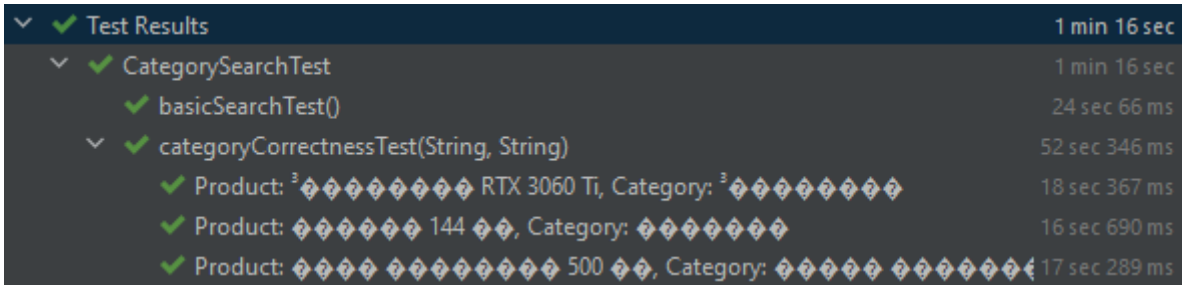

Рис. *45 Результати параметризованого тесту перевірки підібраних категорій пошуку*

ВИСНОВКИ

1. У процесі виконання кваліфікаційної роботи були опрацьовані літературні джерела за темою тестування мобільних застосунків та технології Appium. Також проведено аналіз аналогічних інструментів для тестування мобільних застосунків.
2. Проведено дослідження та розробку тестового середовища для тестування мобільного застосунку Rozetka з використанням технології Appium. В ході дослідження були визначені основні переваги та можливості Appium для автоматизованого тестування мобільних додатків.
3. Проведений опис функціоналу мобільного застосунку Rozetka, В ході дослідження були розглянуті різні аспекти та можливості застосунку, які роблять його популярним серед користувачів. Опис функціоналу мобільного застосунку "Rozetka" включав огляд його основних функцій, таких як пошук та перегляд товарів, додавання їх до кошика, здійснення покупки, реєстрація та авторизація користувача.
4. Проведена розробка тест-вимог та запис тестових сценаріїв для мобільного застосунку "Rozetka". Процес розробки тест-вимог включав аналіз функціоналу застосунку та визначення ключових функцій, які потрібно протестувати. На основі розроблених тест-вимог були записані тестові сценарії, які описували послідовність кроків для тестування кожної функції. Вони включали в себе вхідні дані, очікувані результати та необхідні дії користувача.
5. В результаті роботи був проведений рефакторинг JUnit 5 тестів для мобільного застосунку Rozetka. Перехід до версії 8.0.0 Appium java-client призвів до суттєвих змін у коді, що вимагає змін у розробці тестів. Розроблено тести, які тестують функціональні можливості застосунку, такі як пошук товару, оформлення замовлення, функції сортування та фільтрів та реєстрація користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Verma N. Mobile Test Automation with Appium: Mobile application testing made easy. Packt Publishing, 2017. 256 с.
2. 55+ Jaw Dropping App Usage Statistics in 2023 [Infographic]. Techjury Site. URL: https://techjury.net/blog/app – usage – statistics/ (дата звернення 30.05.2023).
3. The changing world of digital in 2023. Wearesocial Site. URL: https://wearesocial.com/uk/blog/2023/01/the – changing – world – of – digital – in – 2023/ (дата звернення 30.05.2023).
4. Частка ринку Android та iOS: оприлюднено статистику 2022 року. Root – nation Site. URL: https://root – nation.com/ua/news – ua/it – news – ua/ua – android – ios – statistika – 2022/ (дата звернення 30.05.2023).
5. Названі найпопулярніші мобільні застосунки в Україні: YouTube вилетів із трійки лідерів. McToday Site. URL: https://mc.today/uk/nazvani – najpopulyarnishi – mobilni – zastosunki – v – ukrayini – youtube – viletiv – iz – trijki – lideriv/ (дата звернення 30.05.2023).
6. Xcode documentation. Apple Developer Site. URL: https://developer.apple.com/documentation/xcode (дата звернення 30.05.2023).
7. Getting started. Selendroid Site. URL: http://selendroid.io/setup.html (дата звернення 30.05.2023).
8. TestComplete 15 Documentation. Smartbear Site. URL: https://support.smartbear.com/testcomplete/docs/ (дата звернення 30.05.2023).
9. Introduction to Appium. Appium Site. URL: https://appium.io/docs/en/about – appium/intro/ (дата звернення 30.05.2023).
10. Appium Architecture. HeadSpin Site. URL: https://www.headspin.io/course – material/appium – architecture (дата звернення 30.05.2023).

11. WebDriver. W3C Working Draft Site. URL: https://www.w3.org/TR/webdriver/ (дата звернення 30.05.2023).
12. Selenium WebDriver як інструмент для автоматизованого тестування. QATestLab Site. URL: https://training.qatestlab.com/blog/technical – articles/selenium – webdriver/ (дата звернення 30.05.2023).
13. Hans M. Appium Essentials. Packt Publishing, 2015. 188 с.
14. Bångerius S., Fröberg F. Functional testing of an Android application. 2016. 40 с.
15. Milano D. T., Blundell P. Learning Android Application Testing: Improve Your Android Applications Through Intensive Testing and Debugging. Packt Publishing, Limited, 2015. 274с.
16. Бойко М.О., Коломоєць Г.П. доцент, канд. фіз.-мат. наук – науковий керівник. Використання технології Appium для тестування мобільного застосунку. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» / Запорізький національний університет. – Запоріжжя : ЗНУ, 2023. – Т.5. – С. 76-78.