ПЕРСПЕКТИВНІ НАПРЯМИ РОЗВИТКУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.4

Костирко В. С.,
к.фіз.-мат.н., доц., доцент кафедри комп'ютерних наук, Львівський торговельно-економічний університет, Львів

Костенко А. В.,
к.фіз.-мат.н., доц., завідувач кафедри комп'ютерних наук, Львівський торговельно-економічний університет, Львів

Плеша М. І.,
к.фіз.-мат.н., доц., доцент кафедри комп'ютерних наук, Львівський торговельно-економічний університет, Львів

СУЧАСНІ ПІДХОДИ ТА ТЕНДЕНЦІЇ В РОЗРОБЦІ ЗАСТОСУВАНЬ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

Анотація. *Описуються сучасні підходи до крос-платформної розробки мобільних застосувань на базі середовищ розробки та віртуальних пристроїв. Розглядаються тенденції розвитку мобільних застосувань: клієнт-серверні застосування, фреймворки і бібліотеки розширень, застосування хмарних сервісів та віртуальних оточень, шаблони проектування. У результаті проведених досліджень встановлено, що технології розробки мобільних застосувань знаходяться в стані швидкого розвитку і характеризуються широким залученням таких засобів автоматизації, як віртуальні пристрої та фреймворки, віртуальні оточення та хмарні сервіси. Запропоновано певне уточнення понять різних технологій розробки мобільних застосувань, особливо важливе для розуміння тенденцій їх майбутнього розвитку.*

Ключові слова: Android, Java, віртуальний пристрій, хмарний сервіс, фреймворк, застосування, бібліотека.

Kostyrko V. S.,
Ph.D., Associate Professor, Associate Professor of the Department of Computer Sciences, Lviv University of Trade and Economics, Lviv

Kostenko A. V.,
Ph.D., Associate Professor, Head of the Department of Computer Sciences, Lviv University of Trade and Economics, Lviv

Plesha M. I.,
Ph.D., Associate Professor, Associate Professor of the Department of Computer Sciences, Lviv University of Trade and Economics, Lviv

MODERN APPROACHES AND TRENDS IN THE DEVELOPMENT OF APPLICATIONS FOR MOBILE DEVICES

Abstract. *The modern approaches to cross-platform development of mobile applications based on environments development and virtual devices are described. The trends of mobile applications development are considered: client-server applications, frameworks and extension libraries, application of cloud services*

and virtual environments, design patterns. As a result of the research, it has been found that mobile applications technologies are in a state of rapid development and are characterized by extensive involvement of automation tools such as virtual devices and frameworks, virtual environments and cloud services. A certain clarification of the concepts of various technologies of mobile applications development is proposed, that is especially important for understanding their future development trends.

Keywords: Android, Java, virtual device, cloud service, framework, application, library.

DOI: https://doi.org/10.36477/2522-1221-2018-21-19

Постановка проблеми. Комп'ютерні технології захоплюють все нові сфери діяльності людини. Серед них особливе місце займають технології розробки самих комп'ютерних застосувань. Через бурхливий розвиток цієї сфери діяльності за технологічними проривами не встигають ні термінологія, ні їх теоретичне осмислення.

Особливо гостро ця проблема постає в галузі освіти. Адже швидкоплинні зміни в таких складних технологіях потрібно не лише чітко та зрозуміло описати в навчальних дисциплінах, але й правильно відобразити їх тенденції з тим, щоб студенти могли користатися цими знаннями не лише в момент навчання, але і після нього (протягом якогось розумного періоду часу).

Аналіз останніх досліджень і публікацій. В дослідженнях, присвячених сучасним комп'ютерним технологіям, часто ототожнюють багатоплатформність (мультиплатформність) та кросплатформність застосувань [5-7].

Уточнення потребують такі близькі поняття, як бібліотеки, розширення, надбудови та фреймворки [6, 10].

Сучасні підходи до верифікації застосувань потребують вдосконалення оцінки правильності їх функціонування. При цьому застосовуються ті чи інші моделі тестування застосувань [8-9].

Постановка завдання. Дана стаття присвячена побудові базису для використання в навчальних дисциплінах з галузі комп'ютерних технологій.

Виклад основного матеріалу дослідження. Побудова застосувань на базі мобільних пристроїв наштовхується на обмеженість ресурсів останніх в порівнянні з настільними комп'ютерами. До числа таких ресурсів відносяться не стільки процесор та оперативна пам'ять, скільки зовнішні пристрої – дисплей, миша, клавіатура, зовнішня пам'ять.

Через обмеженість ресурсів мобільних пристроїв в розробці застосувань для них майже виключно застосовується технологія крос-платформної розробки. У цій технології розрізняють платформу розробки та платформу виконання застосування. Звичайно платформою розробки служить настільний комп'ютер з відповідним програмним забезпеченням (середовищем розробки), а платформою виконання (цільовою платформою) – мобільний пристрій.

Платформа розробки дозволяє побудувати застосування та згенерувати файл для його інсталяції на мобільних пристроях вибраного типу. Пересилання цього файлу на мобільну платформу здійснюється за допомогою кабельної чи Wi-Fi-мережі і приводить до інсталяції застосування на мобільному пристрої.

Описана технологія успішно реалізується цілим рядом популярних середовищ розробки програм (IDE) – *Android Studio, .NET Compact Framework, Qt SDK, TotalCross, Unity* тощо.

Складність описаної крос-платформної розробки застосувань пояснюється величезною кількістю типів мобільних пристроїв, а також їх операційних систем. Оскільки розробник не може мати в своєму розпорядженні такої кількості мобільних пристроїв (та ще з різними версіями операційних систем), то актуальною стала віртуальна розробка з використанням замість реальних пристроїв їх моделей – так званих віртуальних пристроїв.

Віртуальні пристрої більш або менш точно моделюють роботу реальних пристроїв за допомогою емуляторів та віртуальних машин. Віртуальні пристрої, як правило, значно більш повільні, але реалізуються на самій платформі розробки, що прискорює пересилання інсталяційних файлів (рис. 2).

Крім того, на платформі розробки можна утворити не один віртуальний пристрій, а декілька.

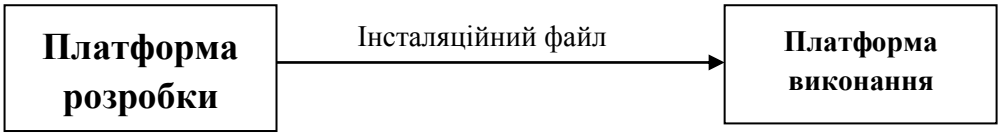

Рис. 1. Схема взаємодії платформ

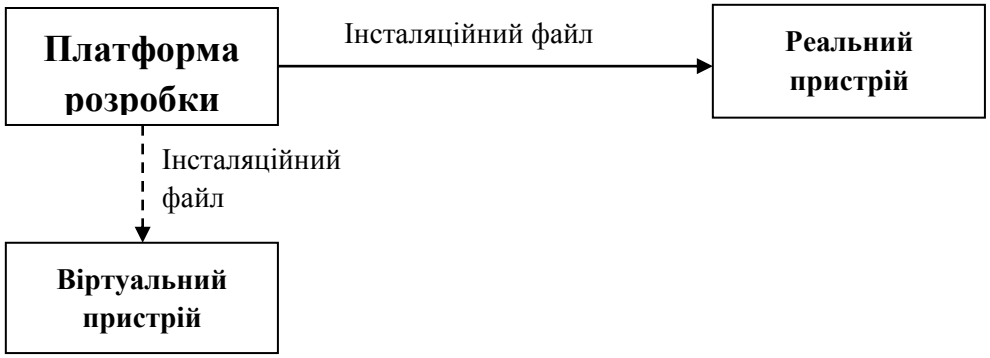

Рис. 2. Віртуальні та реальні пристрої

Середовища розробки характеризуються операційною системою та мовою програмування, а мобільні та віртуальні пристрої – типом та версією операційної системи. Середовища розробки існують для всіх основних платформ настільних комп'ютерів – *Windows, iOS* та *Linux*.

Середовища розробки надають компілятори, системи збірки програм та візуальні дизайнери інтерфейсів. Мови програмування більшості середовищ теж традиційні: *Java, C#, VB.NET, JavaScript, C++*.

Найпопулярнішими операційними системами мобільних пристроїв є *Android, iOS, Windows Mobile*. Кількість мобільних застосувань зростає з прискоренням кожного року. Станом на 2013 рік вона перевищила мільйон: 484 тисячі – для операційної системи *Android*, 617 тисяч – для *iOS* та 156 тисяч – для *Windows Mobile* [1].

Для кожного типу операційної системи мобільних пристроїв звичайно застосовуються свої середовища розробки. Зауважимо, що платформи розробки теж можуть мати різні операційні системи. І для кожної з них можуть застосовуватися декілька середовищ розробки. Тому маємо величезну кількість середовищ технологій крос-платформної розробки застосувань.

Для досягнення уніфікації технологій розробки вже давно розглядається ідея багатоплатформності застосувань. Ціль багатоплатформності полягає в можливості використання одного й того ж застосування на різних платформах.

Цю ідею давно і без особливих успіхів просуває корпорація Microsoft за допомогою середовища розробки Visual Studio. Однак справжня багатоплатформність була досягнута зовсім іншими технологіями. Такими можна вважати web-технології, які дозволяють реалізувати застосування в середовищі браузерів на всіх основних платформах – як комп'ютерних, так і мобільних.

Звичайно, передумовою для їх використання є наявність постійного і достатньо швидкого Інтернету. Але з реалізацією ідей Ілона Маска про запуск великої кількості супутників на низькій орбіті повсюдний і швидкий Інтернет обіцяє стати реальністю.

Іншим успішним прикладом багатоплатформності стали застосування на базі мови Python. Стандартна реалізація мови Python написана на мові ANSI C, для якої є компілятори для всіх основних операційних систем настільних комп'ютерів. Тому й застосування, написані мовою Python, працюють на всіх комп'ютерних платформах.

Повсюдне проникнення Інтернету формує цікавий тренд і в технології крос-платформної розробки мобільних застосувань. Вони набувають рис клієнт-серверних застосувань: на стороні мобільного пристрою реалізується клієнт такого мережевого застосування, а на стороні web-сервера – його сервер (рис. 3).

Це дозволяє звільнити мобільний пристрій від важких операцій з базами даних та файлами і зосередити його на інтерфейсах, однак вимагає надійного та швидкого Інтернету. Але досвід показує, що комп'ютерна інфраструктура розвивається з великим прискоренням, тому до цього тренду доречно віднестися з увагою.

В цьому плані показовим є середовище *Node.js* [12], яке застосовується для побудови серверних застосувань на базі мови програмування *JavaScript*. Середовище розробки генерує для клієнта web-сторінки, побудовані з застосуванням мови розмітки *HTML*, мови стилів *CSS*, а також тієї ж мови програмування *JavaScript*.

Застосування єдиної мови програмування для сервера та клієнта значно полегшує розробку програм застосування, що підтверджує і досвід застосування мови Java для побудови Android-застосувань.

Рис. 3. Схема клієнт-серверної взаємодії

Ще одна нова тенденція полягає в застосуванні фреймворків, які здатні стандартизувати технологію проектування застосувань. Фреймворки пропонують загальні схеми проектів (шаблони проектування) та надають бібліотеки. Фреймворки можуть надавати також додаткові застосування, спрощуючи процес розробки застосувань.

Таким чином, *фреймворк* являє собою більш широке поняття, ніж *бібліотека*. Однак такі поняття, як *розширення* та *надбудова,* фактично є синонімами поняття *бібліотека*. Тому доцільно уніфікувати це поняття як *бібліотека*. Для того, щоб підкреслити той факт, що бібліотека надає пакету чи фреймворку додаткові можливості, її також прийнято називати бібліотекою розширення, це допустимо, оскільки не змінює суті поняття.

Для розширення можливостей фреймворків часто застосовують інші фреймворки. Наприклад, фреймворки *Django* та *Flask* для спрощення генерації HTML-сторінок застосовують фреймворк *Bootstrap*. Фреймворк *Hamcrest* розширює можливості призначеного для модульного тестування фреймворку *jUnit.*

Особливу популярність в сучасних технологіях здобув шаблон *MVP* (*Model-View-Presenter*), який дозволяє відділити бізнес-логіку (*Model*) від інтерфейсу користувача (*View*), щоб зробити інтерфейс більш зрозумілим користувачу. Компонента *Presenter* реалізує взаємодію між компонентами *Model* та *View.*

Завдяки такому розділенню компонента *Model* стає відносно незалежною, що дозволяє реалізувати її на базі технології *ORM* (*Object-Relational Mapping*). Фреймворк *SQLAlchemy* реалізує технологію *ORM* стосовно до багатьох типів реляційних баз даних: *SQLite, Postgresql, MySQL, Oracle, MS SQL, Firebird, Sybase.*

В технології *ORM* структури баз даних представляються *моделями* – високорівневими описами у вигляді класів об'єктів. Зміни в моделях реєструються як *міграції*, які пізніше автоматично переносяться на структури бази даних.

Мета застосування *ORM* полягає в тому, щоб перевести маніпулювання базами даних з низького рівня (SQL) на рівень моделей. В переносі міграцій на структуру бази даних допомагає ще один фреймворк – *Migrate.*

На платформі *Java* технологію *ORM* реалізує фреймворк *Hibernate*, однак великої популярності він поки що не здобув, можливо, через те, що мобільні застосування ще не доросли до складних баз даних. Але знову-таки це справа часу.

Ряд фреймворків зараз успішно використовують шаблон *MVP* при побудові web-застосувань – наприклад, *Django* [14], *Flask* [13]. Однак можна очікувати, що успішний досвід їх використання в галузі web-програмування приведе до появи таких фреймворків і в галузі розробки застосувань для мобільних пристроїв.

Поки що реалізації мови Python на мобільних платформах ще немає, однак це справа часу. Але вже тепер фреймворк *Kivy* дозволяє вести кросплатформну розробку застосувань для мобільних пристроїв на базі мови Python [10]. Причому сам фреймворк *Kivy*, як і Python, є багатоплатформним.

При крос-платформній розробці застосувань для мобільних платформ *Android* та *iOS* успішно використовується фреймворк *Xamarin* [4]. При цьому мовою програмування служить *C#*, а середовищем розробки – *Microsoft Visual Studio*.

Ряд бібліотек покращують технологію розробки Android-застосувань на мові Java. Бібліотека *Dagger* допомагає у використанні шаблону *Dependency Injection*. Цей шаблон дозволяє покращити структуру проекту через впровадження залежностей між класами проекту. Бібліотека надає засоби для здійснення контролю над правильною побудовою таких залежностей.

Бібліотека *Picasso* спрощує роботу з зображеннями, здійснюючи їх кешування та перетворення. Бібліотека *Retrofit* надає мобільним застосуванням засоби для читання та відображення даних з web-серверів, тобто дозволяє їм утворити власного web-клієнта.

Бібліотека *RetroLambda* дозволяє спростити синтаксис слухачів подій завдяки введенню в Java лямбда-функцій.

Однією з трудомістких задач розробки мобільного застосування залишається розробка засобів для реалізації його інтерфейсу – обробників подій та слухачів діалогових елементів. Фреймворк *ButterKnife* надає інструменти для полегшення такої розробки.

Бібліотеки *RxJava* та *RxAndroid* надають класам Java засоби для взаємодії через обмін повідомленнями. В результаті виникла концепція реактивного програмування як альтернатива традиційній концепції імперативного програмування.

Для організації серверної частини застосування можна застосовувати хмарні сервіси класу *PaaS*, які надають ті чи інші фреймворки для мов програмування *Java*, *Python*, *Go*, *PHP* чи *JavaScript*. Наприклад, популярними сервісами цього класу є *Google App Engine* [2] та *Parse.com* [3].

Для того, щоб різні проекти могли застосовувати різні набори фреймворків та бібліотек, не заважаючи іншим проектам, в середовищі Python викристалізувалася ідея віртуального оточення. Такі віртуальні оточення успішно втілюються в життя середовищем проектування *PyCharm*. Можна очікувати появи цієї технології і в середовищах проектування мобільних застосувань.

Ще однією важливою проблемою розробки застосувань є їх верифікація. Загальновідомо, що формальні методи верифікації занадто складні для перевірки правильності реальних програм і успішно застосовуються лише до алгоритмів. Тому практичною методологією для перевірки правильності програм вважається тестування.

Прийнято розрізняти такі три рівні тестування: модульне (*unit testing*), інтеграційне (*integration testing*) та системне. Останнім часом системне поділяють на тестування користувацького інтерфейсу (*UI testing*) та приймальне тестування (*acceptance testing*) [11].

Відповідно класифікуються і самі тести. Ряд авторів вважають, що співвідношення в кількості тестів цих типів дещо відрізняється в залежності від типу застосування, його розмірів тощо, але близьке до 80:10:5:5 [11]. Цей факт наочно представляє так звана піраміда тестування (рис. 4).

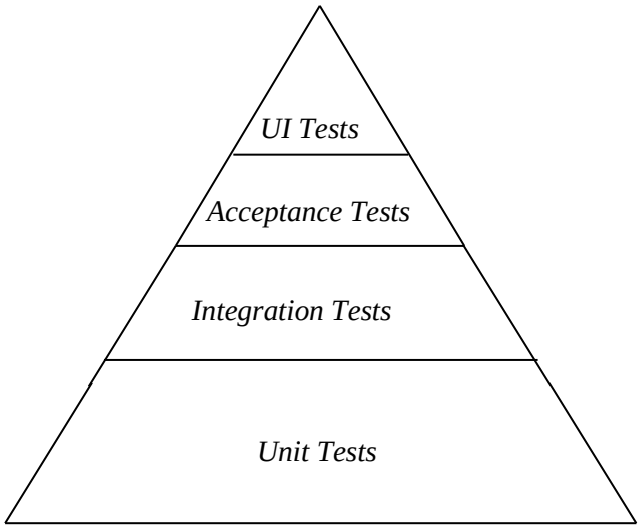

Рис. 4. Піраміда тестування

Тестування застосувань є одним з найбільш трудомістких етапів розробки програмного забезпечення [9]. Складність тестування зростає багатократно при застосуванні технології крос-платформної розробки мобільних застосувань [8].

Справа в тому, що тестування застосування при крос-платформній розробці передбачає цілу послідовність дій:

- побудова на платформі розробки інсталяційного модуля застосування;
- вивантаження його на платформу виконання;
- інсталяція застосування;
- передача на платформу виконання тестових даних;
- запуск застосування на тестових даних та одержання результатів;
- повернення результатів на платформу розробки;
- порівняння очікуваних та одержаних результатів.

Тому для автоматизації тестування при крос-платформній розробці застосувань використовують ряд нестандартних технологій та відповідних фреймворків. Серед них відзначимо технології локального та "інструментованого" тестування (*instrumentation testsing*).

Технологія локального тестування дозволяє радикально зменшити затрати на модульне тестування за рахунок переносу його на платформу розробки. Наприклад, тестування процедури, написаної лише з застосуванням засобів мови Java, можна здійснити не лише на Android-пристрої, а й у середовищі Windows за допомогою інтерпретатора Java.

Звичайно, можливості технології локального тестування обмежені, тому воно застосовується лише для модульного тестування. Оскільки модулі Android-застосувань можуть включати не лише методи класів Java, а й методи класів Android'у, то для тестування таких модулів всі зовнішні виклики підміняються викликами спеціально підготовлених заглушок (dummy) чи моків (mock-object).

На всіх інших рівнях піраміди тестування застосовується технологія інструментованого тестування, коли тести реалізуються в тісній взаємодії платформ розробки та виконання. До речі, термін "інструментоване тестування" неточно відображає суть такої технології. Замість нього більш доречно застосовувати термін "крос-платформне тестування".

Для тестування застосувань в ході їх крос-розробки теж використовуються фреймворки. Для локального тестування застосовується фреймворк *jUnit*. В технології модульного тестування тести описуються за допомогою *тверджень* – методів класу *Assert*.

Фреймворк *Hamcrest* розширює можливості фреймворку *jUnit* введенням так званих *мачерів* (предикатів особливого вигляду). Мачери спрощують читання та інтерпретацію тверджень, наближаючи їх рівень до природньої мови (англійської). До цього рівня мачери піднімають і діагностику причин непроходження тестів.

Мачери застосовуються в твердженнях виду *assertThat*:

assertThat(<Тестоване значення>, <Мачер>*)* .

Мачер – це предикат, який неявно залежить від тестованого значення та, можливо, явно заданих аргументів.

Наприклад, твердження про те, що значення *testValue* повинно бути більшим або рівним 3 з використанням мачерів *is* та *greaterThanOrEqualTo*, можна описати таким чином:

assertThat(testValue,
is(greaterThanOrEqualTo(3))); .

Зауважимо, що цей оператор читається так, ніби він висловлений природньою мовою: "Стверджуємо, що значення *testValue* є більшим або рівним 3".

Суперпозиція мачерів дозволяє підняти рівень тверджень до проблемного. Наприклад, твердження

assertThat(value,
both(startsWith("Hello")).and(endsWith("world!")));

можна прочитати таким чином: значення *value* є текстовим рядком, який починається фрагментом *"Hello"* і закінчується фрагментом *"world!"*.

Користувачі можуть розширювати бібліотеку мачерів власними класами. При модульному тестуванні для підміни Android-залежних класів та методів моками застосовуються фреймворки *Mockito* та *Robolectric*.

Модульне тестування здійснюється на базі методології "білої скриньки", тобто передбачає аналіз початкових текстів модулів.

Для тестування користувацьких інтерфейсів застосовується фреймворк *Espresso*. Цей фреймворк теж тісно співпрацює з середовищем розробки *Android Studio*, але застосовує методологію "сірої скриньки".

Побудова тестів інтерфейсу користувача здійснюється на підставі специфікації застосування та інструкції користувача. Однак початкові тексти модулів тут також потрібні – для запуску тестів з середовища розробки.

Для доступу до інтерфейсів класу діяльності проекту в тестовий клас потрібно включити так звані *правила* – оператори утворення об'єктів класу *ActivityTestRule* на базі цього класу діяльності, наприклад правило доступу до класу діяльності *MainActivity* виглядає таким чином:

```
@Rule
public ActivityTestRule<MainActivity>
mActivityRule = new
ActivityTestRule<>(MainActivity.class);
```

В тестових методах класу для ідентифікації діалогових елементів макету приєднаної діяльності застосовується метод *withId*, який використовує ідентифікатори класу *R* і діє аналогічно методу *findViewById*.

Метод *onView* визначається на об'єктах класу *View* (діалогових елементах макету). Метод *onView* перетворює заданий діалоговий елемент в об'єкт класу *ViewInteraction*, який надає доступ до властивостей і методів цього діалогового елемента. Основні методи класу *ViewInteraction* такі:

- *check* – перевіряє, чи задовольняє поточне значення діалогового елемента сформульованій за допомогою мачера умові;
- *perform* – виконує послідовність дій (об'єктів класу *ViewAction*) над діалоговим елементом; дії в цій послідовності відділяються комами.

Клас *ViewAction* задає операції над діалоговими об'єктами макетів, які моделюють дії користувача, наприклад:

- *typeText* – ввести в поле заданий текст;
- *click* – натиснути мишкою на кнопку;
- *closeSoftKeyboard* – згорнути екранну клавіатуру.

Для полегшення побудови таких тестів середовище Android Studio пропонує команду *Run→RecordEspressoTest*. Ця команда методично записує всі дії користувача в ході тестування інтерфейсів, приблизно так, як в офісному програмуванні генеруються макроси.

Для приймального тестування та тестування інтерфейсів на базі методології "чорної скриньки" використовується фреймворк *Robotium*. Ця методологія застосовна навіть в тому випадку, коли у розробника тестів є інсталяційний файл застосування, але немає доступу до початкових текстів його модулів. Перевагою фреймворку *Robotium* вважають простоту побудови та читабельність тестів, а також відносно високу швидкодію тестування.

Утиліта *Monkey*, яка входить до складу пакета ADB, також працює на базі методології "чорної скриньки". Однак її завдання полягає не в перевірці функціоналу застосування, а в тестуванні його стійкості в екстремальних ситуаціях.

Monkey генерує псевдовипадкові потоки подій, такі як кліки, дотики чи жести, а також ряд системних подій. Псевдовипадковість подій важлива для можливості відтворення ситуації.

Monkey може стежити за тестованим застосуванням: чи не звертається воно до будь-яких інших застосувань (і тоді може блокувати такі звертання). Monkey відстежує аварійні завершення застосування та появу необроблених виняткових ситуацій. Monkey стежить за тим, чи не перестало застосування відповідати на події (*application not responding*).

Висновки і перспективи подальших досліджень у даному напрямі. Технології розробки мобільних застосувань знаходяться в стані швидкого розвитку і характеризуються широким залученням таких засобів автоматизації, як віртуальні пристрої та фреймворки, віртуальні оточення та хмарні сервіси.

В роботі зроблено деякі узагальнення та огляд сучасного стану та тенденцій розвитку технологій крос-платформної розробки застосувань для мобільних пристроїв.

Не претендуючи на загальність, у даній роботі запропоновано певне уточнення понять різних технологій розробки мобільних застосувань, особливо важливе для розуміння тенденцій їх майбутнього розвитку.

ЛІТЕРАТУРА

1. Инфографика. Приложения в App Store, Google Play и Windows Phone Marketplace [Електронний ресурс]. – Режим доступу: http://www.-iphones.ru/iNotes/320838.

2. Sanderson, D. Programming Google App Engine. – 2nd ed., O'Reilly Media, 2012. – 536 p.

3. PARSE.COM [Електронний ресурс]. – Режим доступу: https://parseplatform.org.

4. Peppers J. Xamarin Cross-platform Application Development Second Edition. – Birmingham: Packt, 2015.

5. Маковський Д. Ю., Усата О. Ю. Розробка крос-платформних додатків для мобільних пристроїв [Електронний ресурс]. – Режим доступу: http://eprints.zu.edu.ua/22019/.

6. Злобін Г. Порівняльний аналіз засобів кросплатформного програмування / Злобін Г., Чмихало О. // Електроніка та інформаційні технології. – 2015. – Вип. 5. – С. 159-166.

7. Ткачёва Т. С. Исследование технологий кроссплатформенного программного обеспечения

для мобильных устройств / Ткачёва Т. С. // Наука і техніка Повітряних Сил Збройних Сил України. - 2011. – №2. – С. 130-133.

8. Харченко К. В. Методи та засоби розробки програмних додатків для операційної системи Android / Харченко К. В. // Вісник НТУ "ХПІ". – 2014. – №17. – С. 68-72.

9. Коротун Т. М. Моделі і методи тестування програмних систем / Коротун Т. М. // Проблеми програмування. – 2007. – № 2. – С. 76-84.

10. Schreiber A. Developing Apps for Android and Other Platforms with Kivy and Python [Електронний ресурс]. – Режим доступу: https://elib.dlr.de/86231-/1/20130409.1_Developing_Apps_for_Android_with_Kivy.pdf.

11. Cohn M. Succeeding with Agile: Software Development Using Scrum. – Addison-Wesley, 2010. – 475 p.

12. Node.js в действии / [Кантелон М. и др.]. – СПб. : Питер, 2014. – 548 с.

13. Гринберг М. Разработка веб-приложений с использованием Flask на языке Python. – М.: ДМК Пресс, 2014. – 272 с.

14. Django. Подробное руководство / [Головатый А. и др.]. – СПб. : Символ-Плюс, 2010. – 560 с.

REFERENCES

1. Infografika. Prilozheniia v App Store, Google Play i Windows Phone Marketplace, available at : http://www.iphones.ru/iNotes/320838.

2. Sanderson, D. (2012), Programming Google App Engine (2nd ed.), O'Reilly Media, 536 p.

3. PARSE.COM, available at : https://parseplatform.org.

4. Jonathan Peppers. (2015), Xamarin Cross-platform Application Development Second Edition, Packt, Birmingham.

5. Makovs'kyj, D.Yu. and Usata, O. Yu. Rozrobka kros-platformnykh dodatkiv dlia mobil'nykh prystroiv, available at : http://eprints.zu.edu.ua/22019/.

6. Zlobin H. and Chmykhalo O. (2015), Porivnial'nyj analiz zasobiv krosplatformnoho prohramuvannia, Elektronika ta informatsijni tekhnolohii, vol. 5, p. 159-166.

7. Tkachova, T. S. (2011), Cross-platform research technology software for mobile, Nauka i tekhnika Povitrianykh Syl Zbrojnykh Syl Ukrainy, №2, p. 130-133.

8. Kharchenko, K. V. (2014), Methods and tools for development of software applications for the android operating system, Bulletin of NTU "KhPI", № 17, p. 68-72.

9. Korotun, T. M. (2007), Modeli i metody testuvannia prohramnykh system, Problemy prohramuvannia, № 2, p. 76-84.

10. Schreiber A. Developing Apps for Android and Other Platforms with Kivy and Python, available at : https://elib.dlr.de/86231/1/20130409.1_Developing_Apps_for_Android_with_Kivy.pdf.

11. Cohn M. (2010), Succeeding with Agile: Software Development Using Scrum, Addison-Wesley, 475 p.

12. Kantelon M. i dr. (2014), Node.js v dejstvyy, Piter, SPb., 548 p.

13. Grinberg M. (2014), Razrabotka veb-prilozhenij s ispol'zovaniiem Flask na iazyke Python, DMK Press, M., 272 p.

14. Holovatyj A. i dr. (2010), Django. Podrobnoie rukovodstvo, Symvol-Plius, SPb., 560 p.