МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Проєктування програмних систем для мобільних пристроїв Лабораторний практикум

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра
за спеціальностями F1 «Інженерія програмного забезпечення», F3 «Комп'ютерна інженерія» та F6 «Інформаційні системи та технології»

Укладачі: Б. Я. Корнієнко, А.О. Неструк

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2026

Рецензент *Ладієва Л. Р.*, канд. техн. наук, доцент,
доцент кафедри технічних та програмних засобів автоматизації,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Відповідальний
редактор *Ролік О.І.,* докт. техн. наук, професор

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 6 від 03.04.2026 р.)
за поданням Вченої ради Факультету інформатики та обчислювальної техніки
(протокол № 8 від 30.03.2026 р.)

Навчальний посібник розрахований на одержання студентами практичних навичок з програмування Android-додатків і складається з 7 лабораторних робіт, що охоплюють широкий спектр розробки. В навчальному посібнику приділено увагу основним підходами до програмування додатків під операційну систему Android, особливостям середовища програмування, основним елементам інтерфейсу, інструментам і середовищам програмування, основам проектування мобільних додатків.

Містить теоретичні положення, порядок виконання лабораторних робіт, завдання та список літератури з дисципліни «Проєктування програмних систем для мобільних пристроїв».

Для студентів всіх форм навчання, які навчаються за спеціальностями F1 «Інженерія програмного забезпечення», F3 «Комп'ютерна інженерія» та F6 «Інформаційні системи та технології» факультету інформатики та обчислювальної техніки КПІ ім. Ігоря Сікорського.

Реєстр. № 25/26-297 . Обсяг 4,5 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Стрімкий розвиток технологій та мобільних телефонів за останні двадцять років призвів до заслуженого масового визнання смартфонів. Смартфони завоювали користувачів своєю багатофункціональністю. Камера, телефон, плеєр та інші доповнення в одному пристрої – це справедливо користується високим попитом. Але особливої популярності смартфони набули завдяки додаткам, які на них можна встановлювати. Сучасні web-додатки базуються на різних операційних системах та програмних платформах. Таким чином, вибір при розробці зводиться до створення нативної програми, гібридної програми або web-додатка.

Основна мета лабораторних робіт з дисципліни «Проєктування програмних систем для мобільних пристроїв» – вивчення принципів розробки web-систем із використанням сучасних мов програмування для різних операційних систем. Система, що розробляється за допомогою цих засобів, будується за клієнт-серверною архітектурою, основні принципи якої розглядаються в рамках лекційних занять. Необхідно опанувати особливості роботи з різними програмними платформами, фреймворками, визначити структуру програмної системи.

Лабораторні роботи розраховані на одержання студентами практичних навичок з програмування Android додатків і складається з 7 лабораторних робіт, що охоплюють широкий спектр розробки. Кожна лабораторна робота містить короткі теоретичні відомості і практичні завдання, які студент повинен виконати індивідуально відповідно до свого варіанту. Лабораторні роботи орієнтовані на студентів, які мають початковий рівень підготовки в галузі об'єктно-орієнтованого програмування, інформаційних технологій, володіють основами проектування, мають початкові навички програмування Java додатків.

В результаті виконання лабораторних робіт студенти ознайомляться з основними підходами до програмування додатків під операційну систему Android, з особливостями середовища програмування, з основними

елементами інтерфейсу; вивчать інструменти і середовища програмування, основи проектування мобільних додатків, структуру Android-додатків, що використовується при програмуванні зовнішніх ресурсів, одержать навички роботи з файлами, базами даних, способами створення фонових служб, елементами об'єктно-орієнтованого аналізу і дизайну в Android додатках, принципами роботи з файловою системою, програмуванням додатків на мові Java, використанням SDK Android та розробці інтерфейсу.

ЛАБОРАТОРНА РОБОТА 1

РОЗРОБКА WEB-ДОДАТКІВ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

Мета роботи: ознайомитися з основними поняттями та особливостями розробки web-додатків для мобільних пристроїв.

Теоретичні положення

Розробку web-додатків

Сьогодні смартфони – основний кишеньковий пристрій, тому мобільні web-додатки стали необхідністю як з технічної, так і з комерційної точки зору. Є кілька підходів до розробки таких додатків, і, враховуючи, що компанії, що нині лідирують, можуть за лічені місяці піти на другий план, а нові гаджети з'являються практично безперервно, сучасному фахівцеві важливо розбиратися в основних технологіях створення мобільних web-додатків.

В даний час новий додаток у багатьох випадках буде web-додатком, тоді як у минулому зазвичай починали зі створення програми спеціально для будь-якої операційної системи, наприклад, Windows або Unix. Зараз актуальне завдання створення додатків, здатних працювати на всіх мобільних пристроях, тому дуже непросто прийняти рішення про вибір відповідного інструментарію хоча б через зростаючу кількість платформ і інструментальних середовищ.

Мобільний додаток сильно відрізняється від desktop-програми, тому необхідно враховувати цей факт при плануванні процесу розробки.

Основними відмінностями мобільних додатків та desktop-додатків є:

1) мобільний пристрій – це система, яка не має потужної начинки, таким чином, вона не може працювати як персональний комп'ютер;

2) тестування мобільних програм проводиться на мобільних телефонах (Apple, Samsung, Motorola), в той час як desktop-додатки тестуються на центральному процесорі;

3) розмір екрану мобільного телефону менший, ніж у desktop;

4) здійснення та отримання дзвінків є основним завданням телефону, тому програма не повинна втручатися в цю важливу функцію;

5) мобільні програми мають широкий спектр конкретних операційних систем та компонентних конфігурацій (Android, iOS);

6) операційна система мобільного телефону швидко застаріває;

7) мобільні пристрої використовують мережеві підключення (3G, 4G, 5G), широкосмугове підключення до настільного персонального комп'ютера (ПК) або Wi-Fi;

8) мобільні пристрої постійно здійснюють пошук мережі, тому є необхідність протестувати програму з різною швидкістю передачі даних;

9) інструменти, які добре підходять для тестування desktop-додатків, не повністю підходять для тестування мобільних додатків;

10) мобільні програми повинні підтримувати кілька вхідних каналів (клавіатура, голос, жести тощо), мультимедійні технології та інші функції, що підвищують їх зручність.

Також при розробці звичайних сайтів нерідко доводиться стикатися з тим, що ті чи інші речі у різних браузерах відображаються по-різному: десь підтримується такий стандарт, а десь ні. З цією проблемою доведеться зіткнутися і під час розробки мобільних сайтів.

Особливості web-програмування під мобільні системи

Поняття мобільної web-програми можна трактувати по-різному. Згідно з одним тлумаченням це додаток, що працює в web і спроектований так, щоб коректно відображатися на мобільному пристрої, а згідно з іншим – це додаток, створений спеціально для конкретної мобільної операційної системи (ОС), який з'єднується з web для надсилання та прийому даних. Щоб розмежувати ці позиції, можна скористатися шкалою від стандартних до нативних web-додатків. Якщо нативні працюють зі швидкістю апаратної платформи, то гібридні та мобільні виконуються над додатковими рівнями, які витрачають обчислювальні ресурси, знижуючи швидкодію пристрою.

Введемо поняття стандартних та швидко реагуючих додатків, а також "мобільного web". Відповідні технології, а також їх переваги та недоліки дуже

схожі, проте тут наведемо їх окремо, наголосивши, що це різні рішення. Деяким користувачам та розробникам подобається адаптивний інтерфейс, тоді як інші віддають перевагу спеціалізованим додаткам для конкретних пристроїв. Мобільно-орієнтовані web-сайти (наприклад, відповідні варіанти Facebook або Google Docs) зазвичай пропонують користувачеві можливість доступу до стандартної web-версії. Стандартна програма на смартфоні, найімовірніше, працюватиме повільніше, але деякі користувачі виберуть саме її, щоб мати доступ до всіх можливостей сайту.

Виділимо основні види мобільних web-додатків:

1. Стандартні web-додатки. Терміном «web-додатки» позначаються додатки, розраховані на виконання в браузерах для настільних комп'ютерів. Вони також зможуть працювати на мобільних пристроях, якщо не покладаються на специфічні технології, які відсутні на багатьох мобільних пристроях (наприклад, Adobe Flash).

2. Адаптивний web-інтерфейс. Програми з адаптивним web-інтерфейсом автоматично змінюють його зовнішній вигляд залежно від розміру пристрою – найчастіше використовується технологія CSS (Cascading style sheets). Дизайн може бути обраний сервером під час доставки програми, змінений лише на рівні клієнта чи обома методами. Сенс у тому, щоб контент з одного джерела відображався по-різному в залежності від особливостей конкретного пристрою. Даний варіант застосовується як для мобільних web-додатків, так і для інших видів пристроїв, наприклад, на ігрових консолях та телевізорах.

3. Мобільний web. Термін "мобільний web" використовується для випадків, коли є спеціальний сайт чи онлайн-сервіс для доставки контенту на мобільні пристрої. Зазвичай інтерфейс виглядає ефектніше і працює швидше, ніж адаптивний, оскільки елементи (клавіші, селектори і текстові поля) відображаються аналогічно нативним. При цьому підході може знадобитися створити кілька варіантів одного сайту.

4. Нативні програми. Компанії, що розробляють мобільні операційні

системи, прагнуть до появи якомога більшої кількості нативних програм для їх ОС, і такі програми розробляються за допомогою інструментарію, рекомендованого відповідним постачальником: наприклад, на Objective-C та Xcode – для iOS або Java та Eclipse – для Android. Для кожної ОС зазвичай потрібен окремий проект нативного додатку.

5. Гібридні програми. Гібридним є мобільний додаток, «упакований» у нативну оболонку. Такі програми, як і нативні, встановлюються з онлайн-супермаркету і мають доступ до тих же можливостей пристрою, але розробляються за допомогою HTML5, CSS та JavaScript.

Вибір виду мобільного додатку для кожної ситуації не завжди очевидний. Звузити коло варіантів вибору допоможе розгляд низки технічних факторів:

1. Підтримка платформ та версій. Насамперед слід визначитися, скільки платформ та їх версій належить підтримувати. При цьому потрібно взяти до уваги діапазон пристроїв, відмінності в наборах засобів розробки для кожного з них, а також можливості браузера на кожній платформі та навички. Якщо мета полягає у створенні програми з підтримкою відразу безлічі платформ, то гібридний або мобільний web-додаток підійде більше, ніж нативний, який доведеться розробляти окремо для кожної з платформ.

2. Можливості пристрою. Необхідно врахувати, які вам знадобляться функції пристрою. Якщо програма потребує доступу до камери, сканера штрих-кодів, файлової системи або периферійного Bluetooth-пристрою, то краще скористатися нативним або гібридним підходом. Нові браузери підтримують апаратне прискорення векторної графіки, але поки що не можуть повною мірою користуватися власними графічними можливостями пристрою та іншими функціями, доступними через API.

3. Користувальницький інтерфейс. У нативних додатків більш функціонально багаті і привабливі інтерфейси користувача, швидше реагують і більш інтерактивні, а у web-додатків можуть бути складності з доступом до апаратних функцій пристрою. Гібридні додатки дають розробнику більше

свободи, дозволяючи коду HTML звертатися до нативних API, але необхідність обліку застосування web-технологій робить інтерфейс користувача зовні не зовсім відповідним «рідному».

4. Швидкодія. Швидкодія - один із головних факторів, який потрібно враховувати розробнику. Якщо інтерфейс багатий графікою або потрібна інтенсивна обробка даних, то при мобільному і гібридному підходах досягти високої швидкодії важче, оскільки в цих випадках програми працюють поверх додаткових рівнів, що витрачають ресурси процесора. У будь-якому випадку перед початком розробки слід перевірити швидкодію на прототипі або схожих програмах, що вже існують.

5. Оновлення. При розробці нативних програм потрібно враховувати, що користувачі не завжди хочуть переходити на нову версію, так що, швидше за все, доведеться одночасно підтримувати кілька версій, що ускладнить серверну частину. Те саме стосується гібридних додатків, якщо значна частина коду виконується локально.

Визначитись з вибором виду мобільного додатка також допоможе розгляд низки нетехнічних факторів:

1. Розповсюдження. Мобільні програми просто розповсюджувати, але важко рекламувати, оскільки поза онлайн-супермаркетом їх знайти нелегко. При цьому слід врахувати, що у загальнодоступних магазинів додатків є свої обмеження, немає можливості впливати на політику управління магазином.

2. Цикл узгодження. Процес створення мобільних програм може конфліктувати з методологіями швидкої (agile) розробки, що мають на увазі частий випуск оновлень і безперервний зворотний зв'язок з користувачами. При виборі нативного чи гібридного підходу слід пам'ятати, що частиною проекту розробки стане процес узгодження, і якщо відхилень нічого очікувати, то час узгодження зазвичай мінімальний. Ситуація також полегшується, якщо ви розробляєте програму лише для якогось конкретного смартфона або користуєтеся локальним магазином програм підприємства.

3. Монетизація. За допомогою платформних магазинів додатків можна не тільки розповсюджувати додатки, але й домагатися окупності завдяки простим у використанні та надійним платіжним шлюзам. Але в цій ситуації значну частку доходу отримує власник платформи (у випадку iOS – близько 30%), тому потрібно ретельно зважити, чи варто обирати платформний підхід.

Узагальним усі перелічені чинники, що впливають вибір виду мобільного докладання, в таблиці 1.1.

Таблиця 1.1 – Критерії вибору між підходами

Чинники	**Нативний**	**Гібридний**	**Web**
Ціна підтримки різних платформ	Висока	Середня	Низька
Доступ до можливостей пристрої	Повний	Повний	Частковий
Користувальницький інтерфейс	Повноцінний	Повноцінний	З обмеженнями
Швидкодія	Дуже висока	Дуже висока	Висока
Оновлення версій клієнтської програми	Потрібно	Потрібно	Не вимагається
Простота публікації / поширення	Середня	Середня	Висока
Цикл узгодження	Обов'язковий	Потрібно деяких випадках	Не вимагається
Монетизація через супермар-кет додатків	Доступна	Доступна	Недоступна

Всі види мобільних програм можна створювати без допомоги фреймворків, проте використання готових бібліотек та інструментів спрощує процес розробки та зменшує трудовитрати. Можна обрати гібридний чи нативний підхід і при цьому створювати кросплатформні програми, тобто бібліотеки і для цих випадків.

Сьогодні ринок мобільних пристроїв – це фактично монополія платформ iOS та Android. Наприклад, платформу Firefox OS підтримують великі оператори мереж мобільного зв'язку, а оскільки програми для неї засновані на HTML, їх асортимент може зрости досить швидко.

При створенні нативних та гібридних додатків завжди потрібно враховувати ризик, пов'язаний з відсутністю у розробника контролю над платформою. Наприклад, можна втратити користувачів, якщо власник платформи через побоювання втрати доходів або з іншої причини обмежить доступ до вашої програми. Можливі також проблеми, якщо власники платформ заборонятимуть будь-які програми, розроблені на певному фреймворку, з економічних міркувань або через проблеми з безпекою.

Огляд об'єктно-орієнтованих мов та технологій

Об'єктно-орієнтовані мови програмування користуються останнім часом великою популярністю серед програмістів, оскільки дозволяють використовувати переваги об'єктно-орієнтованого підходу як на етапах проектування і конструювання програмних систем, так і на етапах їх реалізації, тестування і супроводу.

Перша об'єктно-орієнтована мова програмування Simula-67 була розроблена наприкінці 60-х років XX століття у Норвегії. Автори цієї мови дуже точно відчули перспективи розвитку програмування: їхня мова набагато випередила свій час. Однак їхні сучасники (програмісти 60-х рр.) виявилися не готовими сприйняти цінності мови Simula-67, і вона не витримала конкуренції з іншими мовами програмування (передусім з мовою Fortran). Прохолодному відношенню до мови Simula-67 сприяла і та обставина, що вона була реалізована як інтерпретована (а не компілювана) мова, що було абсолютно неприйнятною в 60-ті рр., оскільки інтерпретація пов'язана зі зниженням ефективності програм.

Мова Smalltalk була розроблена командою Xerox Palo Alto Research Center Learning Research Group як програмна частина Dynabook – фантастичного проекту Алана Кея. В основу було покладено ідеї Simula.

Smalltalk є одночасно мовою програмування, і середовищем розробки програм. Це чисто об'єктно-орієнтована мова, в якій абсолютно все сприймається як об'єкти; навіть цілі числа – це класи. Smalltalk є найважливішою об'єктно-орієнтованою мовою, наступною після Simula, оскільки він не лише вплинув на наступні покоління мов програмування, але й заклав основи сучасного графічного інтерфейсу користувача, на яких безпосередньо базуються інтерфейси Macintosh та Windows.

Відомі п'ять випусків мови Smalltalk, що позначаються за роком їх появи: Smalltalk-72, -74, -76, -78, -80. Реалізації 1972 та 1974 років заклали основу мови, зокрема, ідею передачі повідомлень та поліморфізм, хоча на той час механізм успадкування ще не з'явився. У наступних версіях повноправне громадянство набули класів; до цього призвела думка, що це складається з об'єктів. Smalltalk-80 був перенесений на багато комп'ютерних платформ.

В основу мови покладено дві прості ідеї:

- все є об'єктом;
- об'єкти взаємодіють, обмінюючись повідомленнями.

Поява Delphi не пройшла непоміченою серед численних користувачів комп'ютера. Оцінки експертів, які вивчають можливості цього продукту фірми Borland, зазвичай є високими. Основна перевага Delphi полягає в тому, що тут реалізовано ідеї візуального програмування. Середовище візуального програмування перетворює процес створення програми на приємне і легко доступне в плані розуміння конструювання програми з великого набору графічних та структурних примітивів.

Система Delphi дозволяє вирішувати безліч завдань, зокрема:

- створювати закінчені додатки для Windows найрізноманітнішої спрямованості: від обчислювальних та логічних до графічних та мультимедіа;
- швидко створювати (навіть програмістам-початківцям) професійно оформлений віконний інтерфейс для будь-яких додатків;
- створювати потужні системи роботи з локальними та віддаленими

базами даних;

- створювати довідкові системи (файли .hlp) для своїх додатків тощо. Delphi - система, що надзвичайно швидко розвивається. Перша версія Delphi 1.0 була випущена в лютому 1995 р. А потім щорічно випускалися нові версії, кожна наступна версія Delphi доповнювала попередню. Більшість версій Delphi випускається у кількох варіантах: Standart – стандартному, Professional – професійному, Client/Server – клієнт/сервер, Enterprise – розробка баз даних предметних областей. Варіанти відрізняються переважно різним рівнем доступу до систем управління базами даних. Останні варіанти – Client/Server і Enterprise – у цьому відношенні найбільш потужні.

Мова програмування C++ була розроблена Б'єрном Страустрапом, співробітником AT&T Bell Laboratories. Безпосереднім попередником C++ є З with Classes, створений тим самим автором 1980 р. Мова З with Classes, своєю чергою, було створено під сильним впливом З і Simula. C++ – це значною мірою надбудова над З. У певному сенсі можна назвати C++ поліпшеним З, тим З, який забезпечує контроль типів, навантаження функцій та інших зручностей. Але головне в тому, що C++ додає С об'єктну орієнтованість.

Відомі кілька версій С ++. У версії 1.0 реалізовані основні механізми об'єктно-орієнтованого програмування, такі як одиночне успадкування та поліморфізм, перевірка типів та перевантаження функцій. У створеній 1989 р. версії 2.0 знайшли відображення багато додаткових властивостей, що виникли на базі широкого досвіду застосування мови численною спільнотою користувачів. У версії 3.0 1990 з'явилися шаблони та обробка винятків. C++ продовжує вдосконалюватися і зараз. Так, у 1998 р. вийшла нова версія стандарту, що містить у собі деякі досить суттєві зміни. Мова стала основою для розробки сучасних великих та складних проектів.

Мова Java зародилася як частина проекту створення передового програмного забезпечення (ПЗ) для різних побутових приладів. Реалізація проекту була запущена мовою C++, але незабаром виникла низка проблем, найкращим засобом боротьби з якими була зміна самого інструменту – мови

програмування. Мова Java знадобилася для створення інтерактивних продуктів для мережі Internet. Фактично більшість архітектурних рішень, прийнятих при створенні Java, були продиктовані бажанням надати синтаксис, подібний до C і C++. У Java використовуються дійсно ідентичні угоди для оголошення змінних, передачі параметрів, операторів та управління потоком виконанням коду. У Java додані всі найкращі риси C.

Три ключові елементи об'єдналися в технології мови Java, що зробило її докорінно відмінною від усього, що існує на сьогодні:

1) Java надає для широкого використання свої аплети (applets) – невеликі, надійні, динамічні активні мережеві програми, що не залежать від платформи, що вбудовуються в сторінки web. Аплети Java можуть налаштовуватись і поширюватися споживачам з такою ж легкістю, як будь-які документи HTML;

2) Java вивільняє міць об'єктно-орієнтованої розробки додатків, поєднуючи простий та знайомий синтаксис з надійним та зручним у роботі середовищем розробки. Це дозволяє широкому колу програмістів швидко створювати нові програми та нові аплети;

3) Java надає програмісту багатий набір класів об'єктів для явного абстрагування багатьох системних функцій, що використовуються під час роботи з вікнами, мережею та для введення-виведення. Ключова риса цих класів полягає в тому, що вони забезпечують створення незалежних від платформи абстракцій для широкого спектра системних інтерфейсів.

Програми на Java транслюються в байт-код Java, що виконується віртуальною машиною Java (JVM), - програмою, що обробляє байтовий код і передає інструкції обладнанню як інтерпретатор.

Перевагою такого способу виконання програм є повна незалежність байт-коду від операційної системи та обладнання, що дозволяє виконувати Java-програми на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється

віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до вад концепції віртуальної машини відносять зниження продуктивності. Наведемо ряд удосконалень, які дещо збільшили швидкість виконання програм на Java:

- застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді;
- широке використання платформно-орієнтованого коду (native-код) у стандартних бібліотеках;
- апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, яку підтримують деякі процесори фірми ARM).

За даними сайту shootout.alioth.debian.org для семи різних завдань час виконання на Java становить у середньому у півтора-два рази більше, ніж для C/C++. У деяких випадках Java швидше, а в окремих - у 7 разів повільніше. З іншого боку, споживання пам'яті Java-машиной було у 10–30 разів більше, ніж програмою на C/C++. Також дослідження, проведене компанією Google, згідно з яким відзначається суттєво нижча продуктивність і більше споживання пам'яті в тестових прикладах на Java в порівнянні з аналогічними програмами на C++.

Ідеї, закладені в концепцію, та різні реалізації середовища віртуальної машини Java надихнули багатьох ентузіастів на розширення переліку мов, які могли б бути використані для створення програм, що виконуються на віртуальній машині. Ці ідеї знайшли також вираз у специфікації про мовну інфраструктуру CLI, закладену в основу платформи .NET компанією Microsoft.

C# – об'єктно-орієнтована мова програмування. Розроблено у 1998–2001 роках групою інженерів під керівництвом Андерса Хейлсберга в компанії

Microsoft як мова розробки додатків для платформи Microsoft .NET Framework і згодом була стандартизована як ECMA-334 та ISO/IEC 23270.

C# відноситься до сім'ї мов з C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, навантаження операторів (у тому числі операторів явного та неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи та методи, ітератори, анонімні функції з підтримкою замикань, LINQ, винятки, коментарі у форматі XML.

Перейнявши багато від своїх попередників – мов C++, Pascal, Модула, Smalltalk і особливо Java – C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе проблематичними при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинне успадкування інтерфейсів).

C# розроблявся як мова програмування прикладного рівня для CLR і, як такої, залежить насамперед від можливостей CLR. Це стосується переважно системи типів C#, яка відбиває Base Class Library (BCL) – стандартну бібліотеку класів платформи .NET Framework. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи конкретна мовна особливість може транслювати у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився сам C#; подібної взаємодії слід очікувати і надалі (проте ця закономірність була порушена з виходом C# 3.0, що є розширення мови, що не спираються на розширення плат форми .NET). CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені "класичні" мови програмування. Наприклад, складання сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# так само, як це робиться для програм на VB.NET, J# тощо.

Hypertext Preprocessor (PHP) – препроцесор гіпертексту – є найпоширенішою мовою веб-програмування. Переважна більшість сайтів та web-сервісів у мережі Інтернет написана за допомогою PHP. За деякими

оцінками PHP застосовується більш ніж на 80% сайтів, серед яких такі сервіси як facebook.com, baidu.com тощо. Простота мови дозволяє швидко та легко створювати сайти та портали різної складності.

PHP був створений у 1994 р. данським програмістом Расмусом Лердорфом і спочатку був набором скриптів іншою мовою – Perl. Згодом цей набір скриптів був переписаний в інтерпретатор мовою C. І з самого виникнення PHP надавав зручний набір інструментів для спрощеного створення web-сайтів та web-додатків.

PHP надає наступні переваги:

- для всіх операційних систем, що мають найбільше поширення (Windows, MacOS, Linux), є свої версії пакетів розробки на PHP, а це означає, що ви можете створювати web-сайти на будь-якій з цих операційних систем;
- PHP може працювати у зв'язці з різними web-серверами: Apache, Nginx, IIS;
- простота та легкість освоєння. Як правило, маючи навіть невеликий досвід у програмуванні на PHP, можна створювати прості веб-сайти;
- PHP схожий на мову C, тому, знаючи C або одну з мов із C-подібним синтаксисом, буде простіше опанувати PHP;
- PHP підтримує роботу з безліччю систем баз даних (MySQL, MSSQL, Oracle, Postgre, MongoDB тощо);
- поширеність хостингових послуг та їхня дешевизна. Як правило, хостингові компанії розміщують web-сайти на PHP на web-серверах Apache або Nginx, які працюють на одній із операційних систем сімейства Linux;
- web-сервери та операційні системи на базі Linux є безкоштовними, що знижує загальну вартість використання хостингу.

PHP продовжує розвиватися, виходять нові версії, які несуть нові функції, адаптуючи мову програмування до нових викликів.

Практична частина

1. Вивчити теоретичну частину.
2. Розробити Android додаток згідно з варіантом. Запустити програму на емуляторі. Реалізувати програму однією з об'єктно-орієнтованих мов, наведених у теоретичній частині. Виділіть ключові особливості синтаксису мови.
3. Підготуйте звіт з лабораторної роботи.

Таблиця 1.2 – Варіанти завдань на лабораторну роботу

№ варіанту	Завдання
1	**Student**: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса,Телефон. Створити масив об'єктів. Вивести: a) список студентів заданого факультету; b) списки студентів для кожного факультету та курсу; c) список студентів, які народилися після цього року; d) перелік навчальної групи.
2	**Customer**: id, Прізвище, Ім'я, По-батькові, Адреса, Номер кредитної картки, Номер банківського рахунку. Створити масив об'єктів. Вивести: a) список покупців у алфавітному порядку; b) список покупців, у яких номер кредитної картки знаходиться у заданому інтервалі.
3	**Patient**: id, Прізвище, Ім'я, По-батькові, Адреса, Телефон, Номер медичної карти, Діагноз. Створити масив об'єктів. Вивести: a) список пацієнтів, які мають цей діагноз; b) список пацієнтів, номер медичної картки у яких знаходить у заданому інтервалі.

4	**Abiturient**: id, Прізвище, Ім'я, По-батькові, Адреса, Телефон, Оцінки. Створити масив об'єктів. Вивести: a) список абітурієнтів, які мають незадовільні оцінкики; b) список абітурієнтів, середній бал у яких вище заданого; c) вибрати задане число n абітурієнтів, що мають найбільш високій середній бал (вивести також повний список абітурієнтів, які мають напівпрохідний бал).
5	**Book**: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна, Обкладинка. Створити масив об'єктів. Вивести: a) список книг заданого автора; b) список книг, випущених даним видавництвом; c) список книг, випущених після цього року.
6	**House**: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, термін експлуатації. Створити масив об'єктів. Вивести: a) список квартир, що мають задану кількість кімнат; b) список квартир, що мають задану кількість кімнат і розташовані на поверсі, що знаходиться в заданому проміжку; c) список квартир, що мають площу, яка перевищує задануну.
7	**Phone**: ID, Прізвище, Ім'я, По-батькові, Адреса, Номер кредитної картки, Дебет, Кредит, Час міських та міжміських розмов. Створити масив об'єктів. Вивести: a) відомості про абонентів, у яких час внутрішньоміських розмов перевищує заданий; b) відомості про абонентів, які користувалися міжміським зв'язком;

	c) відомості про абонентів у алфавітному порядку.
8	**Car**: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер. Створити масив об'єктів. Вивести: a) список автомобілів заданої марки; b) список автомобілів заданої моделі, які експлуатуютьються більше n років; c) список автомобілів заданого року випуску, ціна яких більше вказаної.
9	**Product:** id, Назва, Виробник, Ціна, Термін зберігання, Кількість. Створити масив об'єктів. Вивести: a) список товарів для заданого найменування; b) перелік товарів для заданого найменування, вартість яких не перевищує задану; c) список товарів, термін зберігання яких більший за заданий.
10	**Train**: Пункт призначення, Номер потягу, Час відправлення, Кількість місць (загальних, купе, плацкарт, люкс). Створити масив об'єктів. Вивести: a) список потягів, що прямують до заданого пункту призначення; b) список потягів, що прямують до заданого пункту призначення та відправляються після заданої години; c) список потягів, що відправляються до заданого пункту призначення та мають спільні місця.
11	**Bus**: Прізвище та ініціали водія, Номер автобуса, Номер маршруту, Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів. Вивести: a) список автобусів для заданого номера маршруту;

	b) список автобусів, які експлуатуються понад 10 років; c) список автобусів, пробіг яких більше 100000 км.
12	**Airlines**: Пункт призначення, Номер рейсу, Тип літака, Час вильоту, Дні тижня. Створити масив об'єктів. Вивести: a) список рейсів для заданого пункту призначення; b) список рейсів для заданого дня тижня; c) список рейсів для заданого дня тижня, час вильоту для яких пізніший за заданий.
13	**Coffee bar:** Напої, Наїдки, Сувеніри, Вартість. Створити масив об'єктів. Вивести: a) список гарячих напоїв; b) список холодних напоїв; c) список наїдків.
14	**Home accounting:** Прибутки, Витрати, Комунальні рахунки. Створити масив об'єктів. Вивести: a) список прибутків; b) список витрат; c) перелік комунальних рахунків.
15	**Fitness center:** Прізвище та ініціали клієнтів, Прізвище та ініціали тренерів, Кількіст годин в абонементі кожного клієнта, Вартість абонементів, Графік занять для клієнта, Графік занять для тренера. Створити масив об'єктів. Вивести: a) список клієнтів з абонементами та використаними годинами; b) список тренерів; c) графік занять для тренерів; d) графік занять для клієнтів.

16	**Cinema:** Фільми, Сеанси, Схеми залів, Вартість квитків. Створити масив об'єктів. Вивести: a) список фільмів; b) список сеансів; c) схеми залів із вільними місцями; d) схеми залів із вартістю квитків.
17	**Beauty salon**: Клієнти, Майстри, Послуги, Вартість послуг, Графік роботи майстрів. Створити масив об'єктів. Вивести: a) список клієнтів; b) список майстрів; c) список послуг; d) графік роботи кожного майстра.
18	**Sports competitions**: Учасники, Види спорту, Судді, Результати змагань. Створити масив об'єктів. Вивести: a) список учасників змагань; b) список видів спорту; c) список результатів змагань; d) список призових місць.
19	**Canteen at the enterprise:** Назва блюда, Вартість, Назва напоїв, Варінти меню (вегетаріанське, особливе). Створити масив об'єктів. Вивести: a) список можливих меню; b) список кожного меню; c) список напоїв.
20	**Home library**: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна. Створити масив об'єктів. Вивести: a) список книг заданого автора; b) список книг, випущених даним видавництвом;

	c) список книг, випущених після цього року.

Контрольні питання

1. Дайте визначення поняття web-додатку.
2. Назвіть основні відмінності між мобільними і desk-top додатками.
3. Наведіть види мобільних web-додатків та охарактеризуйте їх.
4. Перелічіть сучасні об'єктно-орієнтовані мови програмування.

ЛАБОРАТОРНА РОБОТА 2

ШАБЛОНИ ПРОЕКТУВАННЯ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

Мета роботи: вивчити основні шаблони проектування; розглянути відносини та взаємодії між класами або об'єктами у даних шаблонах проектування.

Теоретичні положення

Шаблони проектування для мобільних пристроїв

Паттерн – повторна архітектурна конструкція, що являє собою вирішення проблеми проектування в рамках деякого контексту, що часто виникає.

Найбільш широко використовуваним шаблоном є Model – View – Controller (MVC). MVC – це фундаментальний патерн, який знайшов застосування у багатьох технологіях, дав розвиток новим технологіям і щодня полегшує розробникам життя.

Вперше патерн MVC виник у мові SmallTalk. Розробники мали придумати архітектурне рішення, яке дозволяло б відокремити графічний інтерфейс від бізнес-логіки, а бізнес-логіку від даних. Таким чином, у класичному варіанті MVC складається з трьох частин, які дали йому назву (рис. 2.1).

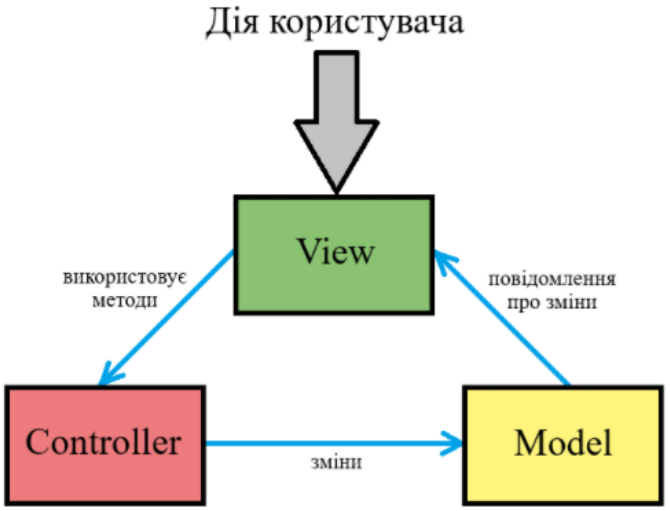

Рис. 2.1. Схема патерну MVC

Під моделлю (Model) зазвичай розуміють частину, що містить у собі

функціональну бізнес-логіку програми. Модель має бути повністю незалежною від інших елементів продукту. Модельний шар нічого не повинен знати про елементи дизайну та спосіб його відображення. Досягається результат, що дозволяє змінювати представлення даних, те, як вони відображаються, не торкаючись самої моделі.

Модель має такі ознаки:

- модель – це бізнес-логіка програми;
- модель має знання про себе саму і не знає про контролерів;
- для деяких проектів модель – це просто шар даних (DAO, база даних, XML-файл);
- для деяких проектів модель – менеджер бази даних, набір об'єктів або просто логіка програми.

До обов'язків Вигляду (View) входить відображення даних, отриманих від моделі. Однак вигляд не може безпосередньо впливати на модель. Можна говорити, що вигляд має доступ до даних «тільки читання».

Вигляд має такі ознаки:

- у Вигляді реалізується відображення даних, які витягуються з моделі будь-яким способом;
- у деяких випадках Вигляд може мати код, який реалізує деяку бізнес-логіку.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. Контролер управляє запитами користувача (отримані як запити HTTP GET чи POST, коли користувач натискає на елементи інтерфейсу до виконання різних дій). Основна функція – викликати та координувати дію необхідних ресурсів та об'єктів, потрібних для виконання дій, що задаються користувачем.

Перевага, яку ми отримуємо від використання концепції MVC, – чіткий поділ логіки вигляду (інтерфейсу користувача) та логіки програми.

Підтримка різних типів користувачів, які використовують різні типи

пристроїв, є загальною проблемою наших днів. Інтерфейс, що надається, повинен відрізнятися, якщо запит надходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність у тому, що контролер вибирає різні види виведення даних.

Крім ізолювання видів від логіки програми концепція MVC істотно зменшує складність великих додатків. Код виходить набагато структурованішим, і тим самим полегшується підтримка, тестування та повторне використання рішень.

Model – View – Presenter

Model – View – Presenter (MVP) – шаблон, який вперше з'явився у IBM, а потім використовувався в Taligent в 1990-х роках. MVP є похідним від MVC при цьому має дещо інший підхід. У MVP Вигляд не тісно пов'язаний з моделлю, як це було в MVC.

Цей підхід дозволяє створювати абстракцію Вигляд. Для цього необхідно виділити інтерфейс Вигляду з певним набором властивостей та методів. Презентер, у свою чергу, отримує посилання на реалізацію інтерфейсу, підписується на події Вигляду та на запит змінює модель.

На рисунку 2.2 показано структуру MVP.

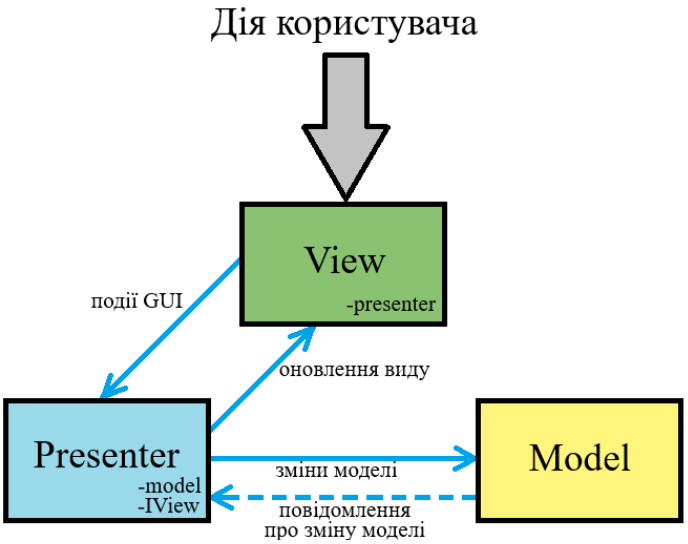

Рис. 2.2. Схема шаблону Model – View – Presenter

Presenter зайняв місце контролера та відповідає за переміщення даних, введених користувачем, а також за оновлення Вигляду при змінах, що

відбуваються в моделі. Presenter спілкується з Виглядом через інтерфейс, який дозволяє збільшити тестування, оскільки модель може бути замінена на спеціальний макет для модульних тестів. Так само, як і для MVC, розглянемо структуру MVP з боку бізнес-логіки програми.

Ознаки Presenter:

- двостороння комунікація з Виглядом;
- Вигляд взаємодіє безпосередньо з презентером шляхом виклику відповідних функцій чи подій екземпляра презентера;
- презентер взаємодіє з View шляхом використання спеціального інтерфейсу, реалізованого Виглядом;
- один екземпляр презентера пов'язаний з одним Виглядом.

Інтерфейс Вигляду визначає набір функцій та подій, необхідних для взаємодії з користувачем (наприклад, IView.ShowErrorMessage(string msg)). Презентер повинен мати посилання реалізацію відповідного інтерфейсу, яку зазвичай передають у конструкторі.

Model – View – ViewModel

Шаблон Model – View – ViewModel (MVVM) застосовується під час проектування архітектури програми. Запропонований Джоном Госсманом у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтовано на сучасні платформи розробки, такі як Windows Presentation Foundation, Silverlight від компанії Microsoft.

MVVM зручно використовувати замість класичного MVC у тих випадках, коли у платформі, на якій ведеться розробка, є "зв'язування даних".

Зв'язування даних – це процес, який встановлює з'єднання між UI (інтерфейсом користувача) програми і бізнес-логікою. Якщо налаштування та сповіщення встановлено правильно, дані відображають зміни, коли вони зроблені. Це може також означати, що, коли UI змінюється, дані, що лежать в його основі, будуть відображати ці зміни.

Основна ідея MVVM Pattern'a - відокремити View, він же (UI) від логіки,

яка може відбуватися в рамках форми (View) і зробити зв'язування даних між цими двома шарами. Це дозволяє змінювати їх окремо один від одного. Наприклад, програміст задає логіку роботи з даними, а дизайнер відповідно працює з інтерфейсом користувача. Схематично робота патерну на рис. 2.3.

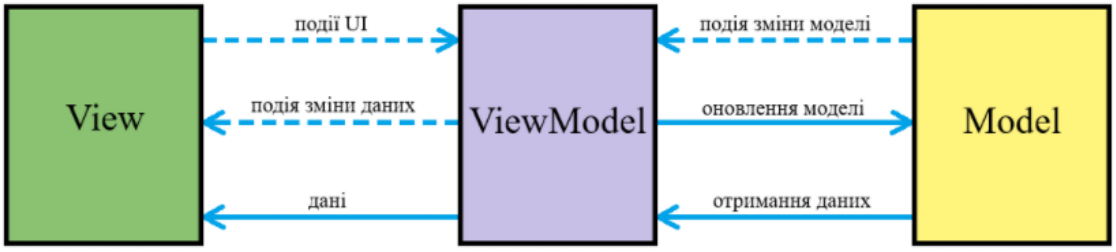

Рис. 2.3. Схема патерну Model – View – ViewModel

Модель (Model) так само, як у класичному патерні MVC, є фундаментальними даними, необхідними для роботи програми (класи, структури).

Вигляд (View) так само, як у класичному патерні MVC, – це графічний інтерфейс, тобто вікно, кнопки тощо.

Модель вигляду (ViewModel, що означає Model of View) є, з одного боку, абстракцією Вигляду, з другого – надає обгортку даних із моделі, які підлягають зв'язуванню. Тобто вона містить модель, яка перетворена на вигляд, а також містить у собі команди, якими може скористатися Вигляд, щоб впливати на модель.

Реалізація MVVM- і MVP-патернів виглядає досить простою, схожою. Однак для MVVM зв'язування Вигляду з View-моделлю здійснюється автоматично, а MVP – необхідно програмувати. MVC має більше можливостей з управління Виглядом.

Загальні правила вибору патерну:

1) MVVM:

а) використовується в ситуації, коли можливе зв'язування даних без необхідності введення спеціальних інтерфейсів Вигляду (тобто відсутня необхідність реалізовувати View);

б) прикладом є технологія WPF;

2) MVP:

а) використовується в ситуації, коли неможливе зв'язування даних (не можна використовувати Binding);

б) прикладом може бути використання Windows Forms;

3) MVC:

а) використовується в ситуації, коли зв'язок між Виглядом та іншими частинами програми неможливий (і ви не можете використовувати MVVM або MVP);

б) прикладом використання може бути ASP.NET MVC.

Варто зазначити, що MVC часто трактують просто як поділ трьох рівнів програми і ніяк не регламентують зв'язок між ними. Тому досить часто зустрічаються діаграми, на яких модель і вигляд пов'язані стрілками, хоча очевидно, що таким чином втрачаються корисні властивості масштабованості при використанні різних виглядів та ієрархічність контролерів.

Вивчення та застосування на практиці архітектурного шаблону проектування HMVC

Розглянемо приклад: є модель чогось, вигляд для відображення даних та контролер для здійснення дій у відповідь на дії користувача (рис. 2.4).

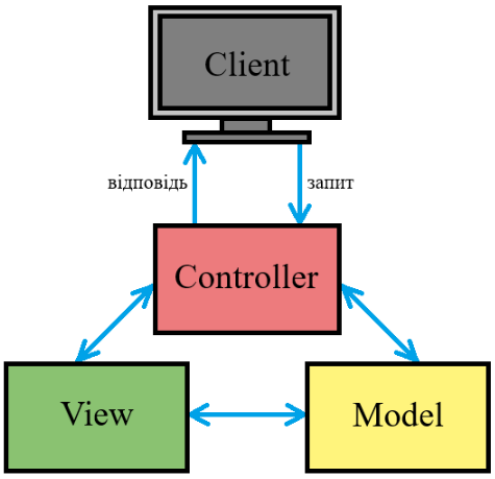

Рис. 2.4. Шаблон MVC

На практиці доводиться оперувати одночасно кількома моделями, наприклад, статтями, користувачами, коментарями. Це не критично, патерн

MVC це передбачає, але це збільшує кількість залежностей: Вигляд та контролер залежать більш ніж від однієї моделі, а від однієї моделі залежать більше одного Вигляду та контролера.

Для відображення даних із різних моделей хотілося б використовувати вигляди, створені спеціально для них. Є логічним відображення коментарів однаково для статей та товарів. Це MVC не передбачає, але цей патерн обходиться частково використанням шаблонів. Тобто використовується один вигляд, який отримує дані з моделей, а для їх відображення використовується комбінація декількох шаблонів. Відповідно зростає кількість залежностей.

Основна ідея HMVC

Оскільки проблеми виникають через те, що замість MVC виходить щось на кшталт MMMVVVCC, де кожна модель, вигляд та контролер можуть належати різним підсистемам, відповідь очевидна – повернутися до MVC, в якому є лише по одній моделі, вигляду та контролеру.

Перший принцип HMVC: у додатку використовуються лише жорстко фіксовані тріади модель – вигляд – контролер, які взаємодіють із рештою підсистем виключно через контролер.

З цього випливає другий принцип: для більш складних комбінацій використовуються ієрархії тріад (рис. 2.5).

На перший погляд, може здатися, що для реалізації HMVC достатньо здійснити виклик контролера з іншого контролера. Але у додатку поведінка залежить непросто від команди, переданої контролеру, як від http-запиту загалом. І модель, і вигляд можуть самі аналізувати запит і певним чином змінювати свою поведінку.

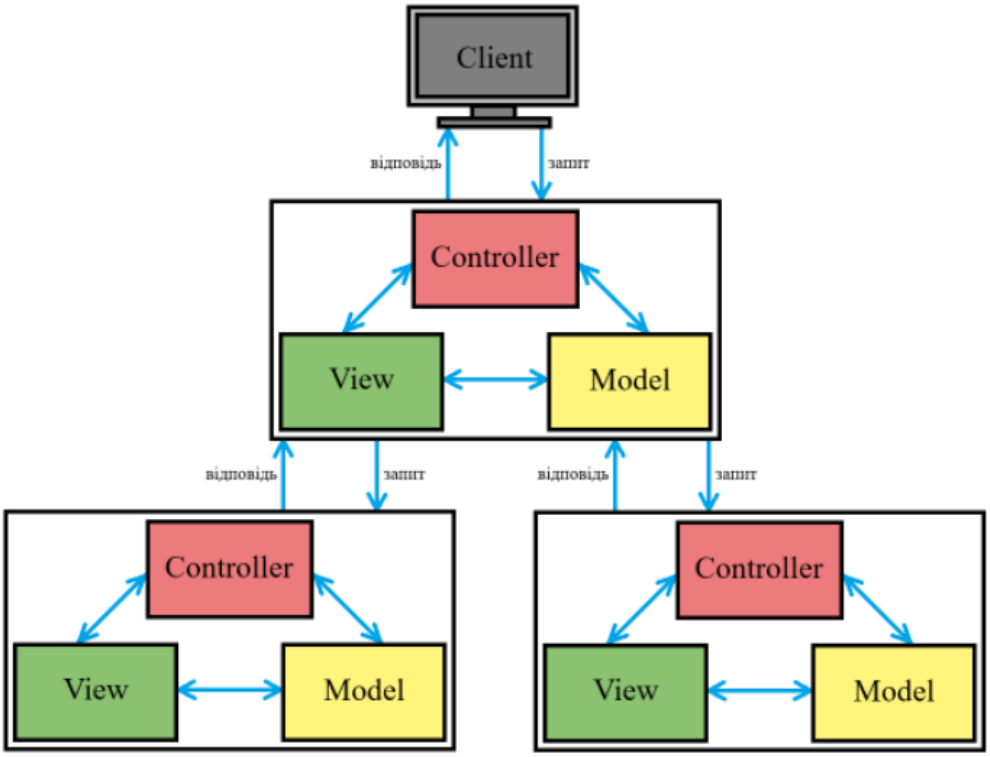

Рис. 2.5. Тріада HMVC

Методи реалізації

1. Клієнт-серверний HMVC

Найпростіший спосіб відправити http-запит – використовувати для цього браузер. У цьому випадку навіть не потрібна якась особлива підтримка фреймворку. Цей підхід використовується скрізь і називається AJAX. Розглянемо приклад зі статтею та коментарями. Можна використовувати одну базову тріаду модель - вигляд - контролер для відображення основного вмісту сторінки (наприклад, тексту статті), а решта необхідних фрагментів (наприклад, коментарі) отримувати за допомогою ajax-запитів до таких же тріад (рис. 2.6).

Такий підхід дозволяє для деяких частин сторінки використовувати http-кешування (кешування даних у кеші браузера, проксі-сервера або проксуючого http-сервера), а частину завантажувати в режимі real-time. Наприклад, сторінка статті може бути завантажена частинами таким чином:

- сама сторінка з текстом статті статична і, по можливості, витягується з кеша браузера, в якому може зберігатися досить довго;

- коментарі постійно оновлюються і тому не кешуються і щоразу запитуються з веб-сервера;
- блок останніх новин оновлюється час від часу, може вилучатися з кешу, але зберігається у ньому, як стаття.

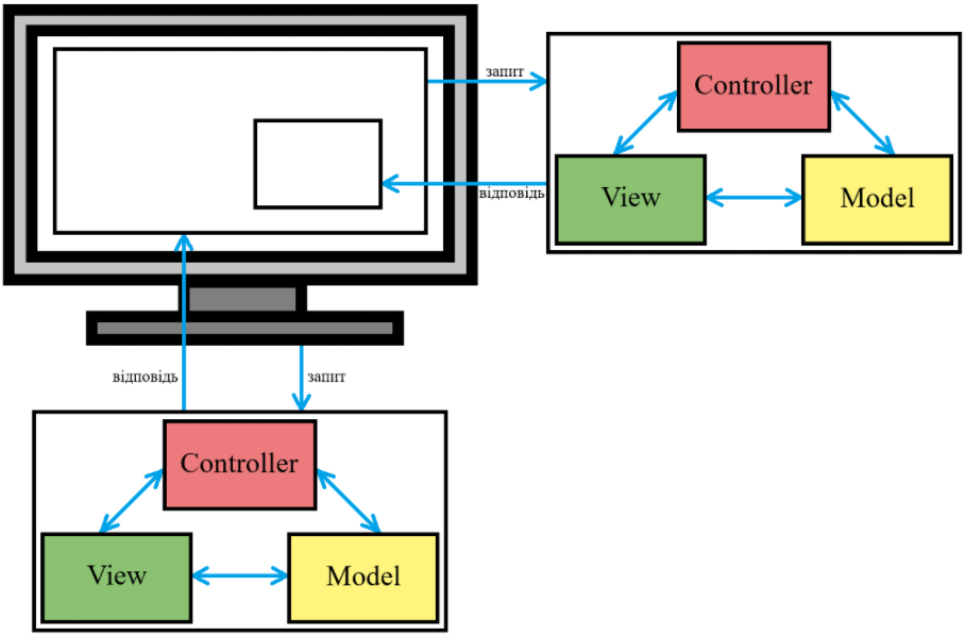

Рис. 2.6. Тріада HMVC з ajax-запитами

Переваги такого підходу:

- немає необхідності підтримки фреймворком;
- можливість гнучкого http-кешування;
- можливість надіслати запит на інший web-сервер, розподіливши таким чином навантаження між кількома серверами.

Недоліки:

- необхідність писати java-скрипти;
- збільшення кількості запитів від браузера серверу, що може збільшити час завантаження сторінки та навантаження на веб-сервер.

2. Сервер-серверний HMVC

Наступним кроком є надсилання запиту web-сервером самому собі. Може використовуватися curl чи інша бібліотека, здатна відправляти http-запити. Цей підхід схожий на попередній, але різниця в тому, що запити

надсилає не браузер користувача, а сам веб-сервер у процесі формування документа. У цьому випадку за допомогою http-запиту, який ініціює запуск паралельного процесу, що видає готовий блок коментарів для статті (рис. 2.7).

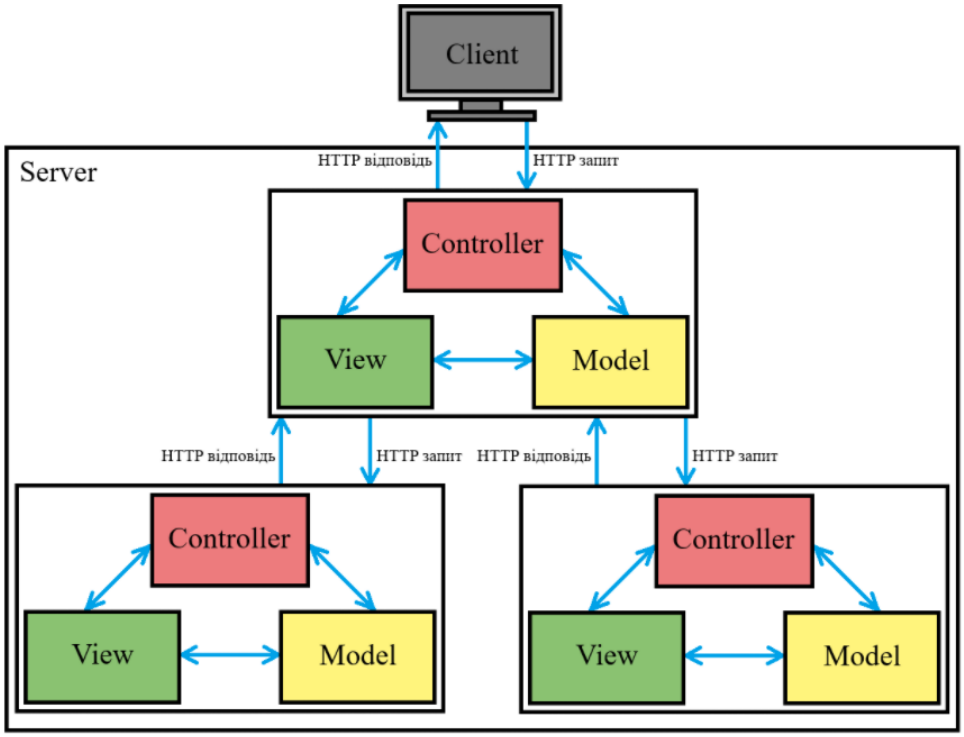

Рис. 2.7. Надсилання запиту сервером

Як і в попередньому випадку, при такому підході можливе використання http-кешування, але для цього необхідно застосувати як проксі http-сервер.

Переваги такого підходу:

- клієнт отримує готову сторінку;
- можливість гнучкого http-кешування;
- можливість надіслати запит на інший web-сервер, розподіливши таким чином навантаження між кількома серверами.

3. Внутрішній сервер HMVC

Під терміном «внутрішній сервер» мається на увазі, що все відбувається всередині процесу застосування. Як і в попередньому випадку, тріади модель – вигляд – контролер спілкуються між собою за допомогою запитів, і сприйматися вони повинні ними так само, як звичайні http-запити, проте

передачу цих запитів забезпечує фреймворк. З погляду програміста, два останні варіанти відрізняються між собою несуттєво. У випадку фреймворка різниця має бути лише в одному параметрі, який вказує, повинен підзапит бути внутрішнім (всередині процесу) або зовнішнім (викликати справжній http-запит).

Практична частина

1. Вивчити теоретичну частину.
2. Розробити Android додаток з використанням архітектури "Модель-Вигляд-Контролер", що виконує введення даних, виведення та редагування відповідно до варіанту (Таблиця 1.2.). Для виконання кожного пункту завдання (a-d) використовувати окрему Activity та модель. Запустити програму на емуляторі.
3. Підготувати звіт з лабораторної роботи.

Контрольні питання

1. Дайте визначення поняття «патерн».
2. Дайте характеристику шаблону Model – View – Controller (MVC).
3. Дайте характеристику шаблону Model – View – Presenter (MVP).
4. Дайте характеристику шаблону Model – View – ViewModel (MVVM).
5. Перелічіть проблеми, з якими стикається шаблон MVC.
6. Назвіть основні принципи шаблону HMVC.
7. Назвіть види HMVC.

ЛАБОРАТОРНА РОБОТА 3
МОБІЛЬНІ ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТАРІЙ

Мета роботи: вивчити основні засади програмування для мобільного пристрою, провести класифікацію мобільних пристроїв, вивчити основні середовища розробки додатків Android Studio та PHPStorm.

Теоретичні положення

Програмування для мобільного пристрою

Операційна система Android побудована на унікальній основі, що передбачає повну відкритість вихідного коду. Ця операційна система перебуває у вільному поширенні. Це спрощує розробникам можливість отримати безпосередній доступ до вихідного коду Android та зрозуміти, як правильно реалізовані властивості та функції різних програм. Абсолютно будь-який користувач може взяти участь у вдосконаленні операційної системи Android. Для цього достатньо надіслати звіт про виявлені помилки або взяти участь в одній із груп. У мережі Інтернет доступні різні додатки Android з відкритим вихідним кодом, які пропонують найбільша корпорація Google і ряд сторонніх виробників.

Відкритість платформи сприяє швидкому її оновленню. На відміну від закритої системи iOS компанії Apple, доступної тільки на пристроях Apple, система Android доступна на пристроях десятків виробників обладнання (OEM, Original Equipment Manufacturer) та телекомунікаційних компаній.

При розробці програм для Android використовується об'єктно-орієнтована мова програмування Java – на даний момент це одна з найпоширеніших мов програмування. Використання мови Java стало цілком обґрунтованим вибором для платформи Android, тому що це потужна, вільна і відкрита мова, відома мільйонам розробників. Програмісти – фахівці мови Java – можуть дуже швидко опанувати Android-програмування, використовуючи інтерфейси Google Android API (Application Programming Interface) та інші розробки незалежних фірм.

Мова Java є об'єктно-орієнтованою, надає розробникам доступ до

потужних бібліотек класів, що прискорюють розробку тих чи інших програм. Програмування графічного інтерфейсу користувача є керованою подією. Крім безпосереднього написання коду додатків, можна скористатися спеціальними середовищами розробки додатків Eclipse та Android Studio, що дозволяють збирати графічний інтерфейс з готових об'єктів, таких як різні кнопки та текстові поля, перетягуючи їх у певні місця екрану, додаючи різні підписи та змінюючи їх розміри. Ці середовища розробки дозволяють швидко та зручно створювати, тестувати та налагоджувати програми Android.

У сучасному світі універсальні смартфони Android поєднують функції мобільних телефонів, інтернет-клієнтів, MP3-плеєрів, ігрових консолей, цифрових фотоапаратів тощо. Ці портативні пристрої обладнані повнокольоровими мульти-сенсорними екранами. Простий дотик пальців дозволяє легко перемикатися між використанням телефону, запуском програм, відтворенням музики, переглядом веб-сторінок оснащуються фізичними клавіатурами.

У комплект поставки пристроїв Android входять різні вбудовані програми, набір яких залежить від пристрою, виробника або оператора мобільного зв'язку. Зазвичай це програми "Телефон" (Phone), "Пошта" (EMail), "Браузер" (Browser), "Камера" (Camera) та ряд інших додатків.

Web-Служби (web-services) являють собою програмні компоненти, що зберігаються на одному комп'ютері, до яких можуть звертатися додатки (або інші програмні компоненти) з іншого комп'ютера через Інтернет. На базі web-служб можуть створюватися звані мешапи (mashups), які прискорюють розробку додатків шляхом комбінування різних web-служб, що у різних організаціях, і з різними типами введеної інформації. Наприклад, сайт 100 Destinations об'єднує фотографії та публікації в Twitter з картами Google Maps, щоб користувачі могли познайомитися з різними країнами за фотографіями інших користувачів.

На сайті Programmableweb розміщено каталог 9400 API та 7000 мешапів, а також посібники та приклади коду з самостійного створення мешапів. За

даними Programmableweb для створення мешапів найпопулярнішими API на сьогоднішній день є Google Maps, Twitter та YouTube.

Види мобільних пристроїв

Мобільні пристрої припускають використання будь-якого виду комп'ютера в середовищі, що рухається. Безпосередньо у русі можуть бути використані такі пристрої, як ноутбуки, портативні міні-комп'ютери, мікрокомп'ютери та мобільні телефони, а також цифрові камери та MP3-програвачі. Всі мобільні пристрої використовують технології бездротового з'єднання, такі як LAN, Wi-Fi, GPRS, 3G, 4G, 5G.

Мобільні пристрої можна класифікувати за двома тематичними категоріями: портативні пристрої та переносні пристрої.

Портативні пристрої фактично відносяться до провідних з'єднань. Самі портативні пристрої мобільні, але щоб скористатися ними, існує одна потреба - підключення їх до мережних портів. Це означає, що портативними пристроями можна скористатися скрізь, де є доступ до мережевих портів.

В даний час мобільні пристрої також називають переносними пристроями. Це – справжній бездротовий зв'язок. Мало того, що самі пристрої рухливі, то ще й ними скористатися можна практично скрізь. Сьогодні портативні пристрої перебувають у стадії зникнення; а мобільні пристрої стали невід'ємною частиною життя.

Вперше мобільні пристрої були використані у транспортних засобах. Нинішні спідометри були першими комп'ютеризованими пристроями. Практично у кожному сучасному транспортному засобі існує кілька мобільних пристроїв, розташованих на панелі приладів.

Широке поширення на сьогоднішній день мають такі мобільні пристрої:

- стільниковий телефон;
- електронна книга;
- планшет.

Стільниковий телефон – мобільний телефон, призначений для роботи в мережах стільникового зв'язку, використовує приймач радіодіапазону та

традиційну телефонну комутацію для здійснення телефонного зв'язку на території зони покриття стільникової мережі.

В даний час стільниковий зв'язок - найпоширеніший з усіх видів мобільного зв'язку, тому, як правило, мобільним телефоном називають саме стільниковий телефон, хоча мобільними телефонами, крім стільникових, є також супутникові телефони, радіотелефони та апарати магістрального зв'язку.

Стільниковий телефон – складний високотехнологічний електронний пристрій, що включає:

- приймач на піддіапазони 1–2 ГГц (GSM) та 2–4 ГГц (UMTS) НВЧ-діапазону;
- спеціалізований контролер управління;
- дисплей;
- інтерфейсні пристрої;
- акумулятор.

Більшість апаратів мають унікальний номер, наприклад, IMEI – міжнародний ідентифікатор мобільного пристрою. IMEI присвоюється під час виробництва стільникового телефону і складається з п'ятнадцяти цифр; він записується в частину прошивки телефону, що не модифікується. Сам цей номер надрукований на етикетці телефону під акумулятором, а також на упаковочній коробці до телефону (під штрих-кодом).

Залежно від вимог, які ви ставите до апарата, необхідно вибирати дисплей відповідних розмірів. Якщо однією з функцій телефону, крім дзвінків, є вихід до мережі Інтернет або читання електронних книг, то рекомендується вибирати телефони з великим сенсорним екраном – від 3,5″. З іншого боку, якщо при використанні потрібна максимальна зручність і компактність, варто вибрати пристрій з меншим розміром діагоналі.

Крім вибору розміру, необхідно звернути увагу на роздільну здатність дисплея: кількість точок на екрані (чим більше, тим чіткіше зображення) і технологію його виконання.

Можна виділити три основні технології виробництва екранів для мобільних пристроїв:

- LCD (або SLCD);
- TFT (або його різновид IPS);
- AMOLED.

До переваг LCD-дисплеїв можна віднести невеликі розмір та масу, відсутність видимого мерехтіння, дефектів фокусування та зведення променів, перешкод від магнітних полів, проблем з геометрією зображення та чіткістю. IPS-екрани мають хороші кути огляду і підвищену роздільну здатність (велику кількість точок на екрані). Особливістю AMOLED-дисплеїв є їхня висока яскравість, контрастність і хороша передача кольору.

Електронна книга – це книга, в якій інформація представлена в електронному вигляді. Донедавна електронні книги існували тільки в програмній інтерпретації, у різних форматах, як звичайних (наприклад, .txt, .doc, .htm, .chm, .pdf, .rtf, .djvu), так і специфічних (наприклад, .fb2). Деякі програмні електронні книги створені як самостійні програми у форматі exe-файлів. Електронні книги реалізовані також в апаратній інтерпретації, так звані цифрові гаджети (пристрої для читання електронних книг).

Важливим фактором є роздільна здатність екрану (для 5–6-дюймових книг – зазвичай 800×600 пікселів, для 9–10-дюймових – 1200×825). У книгах із дисплеєм E-Ink вказується кількість градацій сірого кольору (зазвичай 16). Чим вище ці показники, тим чіткішу картинку ми побачимо.

Наявність вбудованої пам'яті зазвичай дозволяє зберегти більше 1000 книг, але при бажанні зберегти в пристрої багато улюблених книг слід звернути увагу на наявність слота для карт пам'яті.

Показники частоти процесора та ємності оперативної пам'яті дадуть вам уявлення про швидкодію електронного пристрою. Ця інформація буде цікавою, якщо ви плануєте одночасно читати кілька книг, слухати при цьому фонову музику та користуватися доступом Wi-Fi до Інтернету.

З важливих додаткових функцій електронних книг виділимо наявність

встановленого словника, можливості перетворення тексту на мову. З розважальних, але водночас корисних функцій відзначимо можливість підключати до книг електронні накопичувачі (флешки), підтримку мультимедіа-, відео-, аудіо-, фотоформатів.

Наявність в електронній книзі відкритої операційної системи (наприклад, Android) перетворює електронний пристрій, по суті, на невеликий планшет, дозволяючи завантажувати та встановлювати додаткові програми для навчання або розваги.

Підтримка Wi-Fi та Bluetooth можуть дозволити швидко поповнити вашу бібліотеку як через Інтернет, так і з іншого пристрою.

Планшет – це пристрій із сенсорним екраном, який має розміри звичайної книги та виконує функції комп'ютера.

Планшети мають менший розмір та вагу, ніж ноутбуки, що робить їх транспортабельними. Через розміри та обчислювальні потужності планшетів менше, але при цьому час автономної роботи більше. А в порівнянні зі смартфонами вони мають більший розмір екрану, що комфортніше при тривалому перегляді відео або текстової інформації.

Перші моделі планшетних комп'ютерів ще називалися Tablet PC та Slate PC. Tablet PC відрізнялися від інших планшетів тим, що вже встановлювалася операційна система з комп'ютерів (Windows XP, Windows 7).

А вже ближче до 2010 року з'явилися і Slate PC (тонкий ПК). Вони були меншого розміру і зберігали сумісність з IBM PC, що дозволяло використовувати програмне забезпечення зі звичайного комп'ютера. Там вже встановлювалися операційні системи, перероблені для сенсорних екранів.

Революційним етапом розвитку планшетів можна назвати 2010. Саме цього року фірма Apple випустила свій перший інтернет-планшет під назвою iPad. Його поява довела виграшність думки, що планшети повинні споживати контент, а не створювати. Tablet PC та Slate PC пішли в минуле, ринок з тих пір міцно завоювали інтернет-планшети. З того часу всі пристрої із сенсорним екраном розміром діагоналі дисплея від 7 до 12 дюймів без апаратних

клавіатури та мишки називаються планшетами або планшетними комп'ютерами. Винятком є електронні книги, які також підходять під цей опис.

На функціональні можливості планшетного комп'ютера впливають розміри екрана та продуктивність, що залежить від процесора та обсягу оперативної пам'яті.

Оскільки сьогоднішні планшети є здебільшого пристроями споживання контенту, а не його створення, то й виконувані функції цим визначаються.

Сьогодні планшети надають такі можливості:

- інтернет-серфінг – дозволяє отримувати інформацію з Інтернету (текстову, відео, фото, музику), відвідування сайтів;
- у мережі Інтернет користування соціальними мережами;
- електронна пошта;
- перегляд відео (фільми будь-якої роздільної здатності, кліпи, навчальні фільми тощо);
- прослуховування музики;
- ігри (можливість завантажувати зі сховищ виробників операційних систем як платні, і безкоштовні);
- фото- та відеозйомка на вбудовані камери з подальшою обробкою нескладними редакторами на планшеті;
- робота з текстовими документами, таблицями та іншими документами (враховуючи, що працювати потрібно з екранною клавіатурою або додатково підключати апаратну клавіатуру);
- читання електронних книг різного формату (.fb2, .pdf, .djvu тощо);
- GPS-Навігація.

Функції планшетів можна розширити, встановивши додаткові програми з магазинів програм від Apple, Google, Microsoft.

Варто пам'ятати, що повністю замінити персональний комп'ютер планшет не зможе через обмеження на продуктивність. Він скоріше є доповненням до ПК, яке завжди можна взяти із собою у дорогу чи подорож. Саме мобільність і є однією з головних переваг планшетного комп'ютера.

Android Studio

Android Studio – інтегроване середовище розробки виробництва Google, за допомогою якого розробникам стають доступні інструменти для створення програм на платформі Android OS. Android Studio можна встановити на Windows, Mac та Linux. Обліковий запис розробника додатків у Google Play App Store коштує 25 дол. Android Studio створювалася з урахуванням IntelliJ IDEA (рис. 3.1).

IDE можна завантажити та користуватися безкоштовно. Studio містить інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Рішення для Android розробляються в Android Studio за допомогою Java або C++. В основі робочого процесу Android Studio закладено концепт безперервної інтеграції, що дозволяє відразу ж виявляти проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного додатка в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard та App Signing.

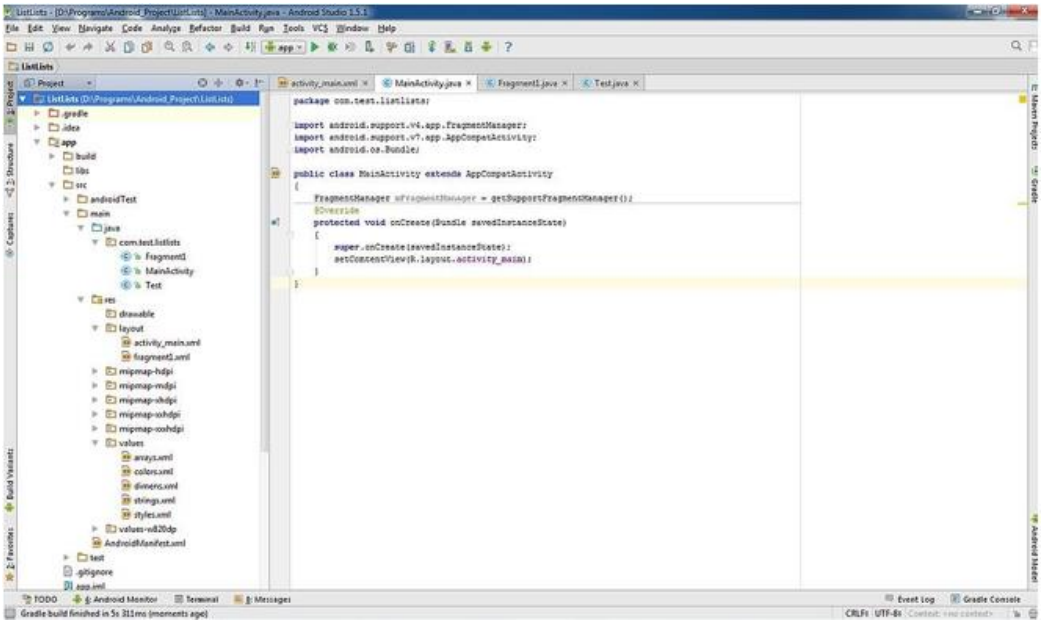

Рис. 3.1. Вікно Android Studio

Android Studio сумісна з платформою Google App Engine для швидкої інтеграції у хмарі нових API та функцій. У розробці ви знайдете різні API, такі як Google Play, Android Pay і Health. Є підтримка всіх платформ Android, починаючи з Android 1.6. Є варіанти Android, які суттєво відрізняються від версії Google Android. Найпопулярніша з них – це Amazon Fire OS. В Android Studio можна створювати APK для цієї ОС. Підтримка Android Studio обмежується онлайн-форумами.

Практична частина

1. Вивчити теоретичну частину.

2. Додати у програму, розроблену у лабораторній роботі № 2 можливості введення даних, виведення та редагування відповідно до варіанта (таблиця 1.2). Під час запуску програми зчитує дані з файлу. При закінченні програми зберігати дані у файл. Зі зміною стану життєвого циклу Activity записувати інформацію до журналу. Запустити програму на емуляторі.

3. Підготувати звіт з лабораторної роботи.

Контрольні питання

1. Класифікуйте мобільні пристрої та назвіть основні їх особливості.
2. Охарактеризуйте середовище розробки Android Studio.

ЛАБОРАТОРНА РОБОТА 4. ЕМУЛЯТОР ТА РЕСУРСИ ANDROID

Мета роботи: вивчити програмні засоби емуляції операційної системи Anroid; розглянути ресурси Anroid; основні методи роботи із файловою системою Anroid; ознайомитись з методами підключення бібліотек та віджетів у додаток для Anroid; вивчити маніфест операційної системи Anroid.

Теоретичні положення

Встановлення емулятора

Для роботи емулятора операційної системи Android потрібна наявність на комп'ютері віртуальної машини Java (Java Runtime Environment), яку можна завантажити на офіційному веб-сайті Java. Завантаживши цю віртуальну машину на свій комп'ютер, її слід встановити, дотримуючись підказок майстра установки.

Власне, сам емулятор Android можна завантажити з офіційного сайту Android Studio, де слід вибрати 32- або 64-розрядну версію програми, що відповідає тій Windows-платформі, на яку вона встановлюватиметься.

Щоб запустити емулятор, спочатку слід створити віртуальний пристрій, робота якого емулюватиметься. Для цього слід перейти в розпакованій директорії (за замовчуванням ім'я директорії може бути, наприклад, adt-bundle-windows-x86-20130729, а можна і перейменувати його) в піддиректорію /sdk/tools і запустити файл android.bat, так що повний шлях до файлу буде, наприклад, такий: C: Program Files adt-bundle-windows-x86-20130729 sdk tools android.bat [33].

Внаслідок цього з'явиться вікно менеджера SDK (Android SDK Manager) (рис. 4.1).

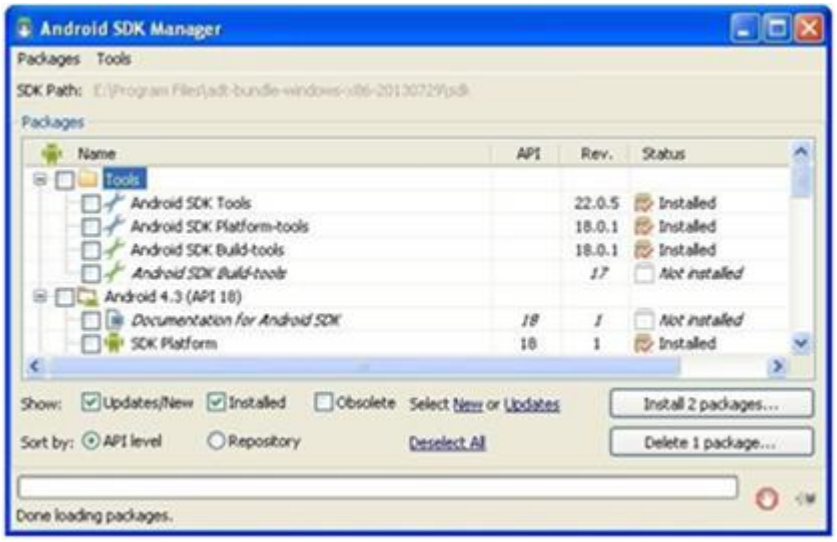

Рис. 4.1. Стартове вікно Android SDK Manager

У цьому вікні із системного меню tools слід вибрати пункт Manage AVDs (Android Virtual Devices (AVD) – віртуальні пристрої). З'явиться вікно менеджера віртуальних пристроїв Android Virtual Device Manager (рис. 4.2).

Рис. 4.2. Вікно менеджера віртуальних пристроїв

Android Virtual Device Manager

Натиснувши кнопку New..., переходимо до вікна створення нового віртуального пристрою (рис. 4.3).

Рис. 4.3. Створення нового віртуального пристрою

У цьому вікні слід дати будь-яку назву новому віртуальному пристрою AVD Name (в даному прикладі вибрано ім'я G1) і вибрати пристрій Device (тут вибрано Galaxy Nexus). Інші параметри можна залишити за замовчуванням.

Після введення параметрів нового пристрою та натискання кнопки OK з'явиться вікно з параметрами нового віртуального пристрою (рис. 4.4).

Рис. 4.4. Параметри створеного віртуального пристрою

Тепер, закривши раніше відкриті вікна, переходимо безпосередньо до запуску емулятора. Для цього слід створити ярлик для програми emulator.exe, розташованої у цій директорії /sdk/tools. Як параметри запуску необхідно вказати наступне: emulator.exe –avd G1. Тут G1 це ім'я створеного віртуального пристрою (рис. 4.5).

Замість ярлика можна створити командний bat – пакетний файл, розташувавши його у директорії програми /sdk/tools з вмістом emulator.exe-avd G1 (рис. 4.6).

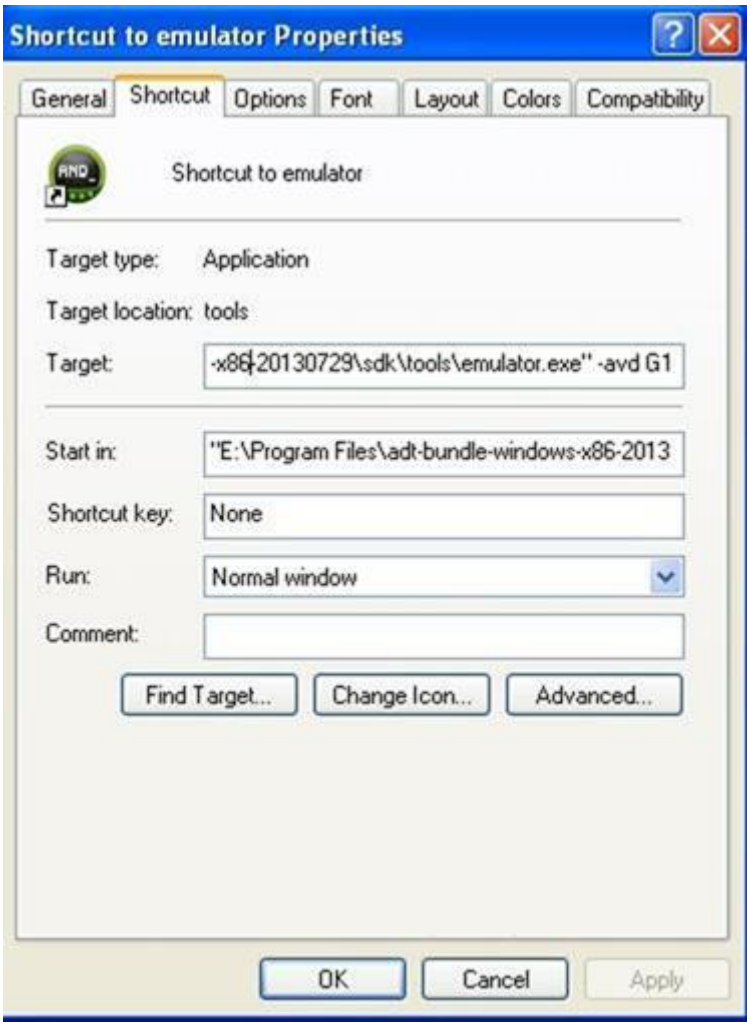

Рис. 4.5. Властивості ярлика емулятора

Рис. 4.6. Стартове вікно емулятора Android

Завантаження системи триватиме деякий час. Після завантаження вікно емулятора Android матиме вигляд (рис. 4.7).

Рис. 4.7. Запущений емулятор Android

У цьому випадку на правому полі вгорі видно кілька кнопок, з них активні лише три верхні (для даного віртуального пристрою). Встановлений

емулятор Android дозволяє запускати і тестувати програми без мобільного пристрою на базі Android.

Приклади використання ресурсів

Розглянемо способи застосування, формат та синтаксис основних типів ресурсів програми Android, які ви можете надати в каталозі проекту res (це стосується як Eclipse, так і Android Studio).

Проаналізуємо такі типи ресурсів:

- Animation Resources;
- Property Animation;
- View Animation;
- ресурс Color;
- ресурс Color State List;
- ресурси типу Drawable.

Animation Resources

Тут задаються ресурси анімації. Tween-анімації (анімації з використанням проміжних кадрів) зберігаються в папці res/anim/, і доступ до них здійснюється за допомогою класу R.anim. Frame-анімації (покадрові анімації) зберігаються в папку res/drawable/ (доступ через клас R.drawable).

Ресурс анімації може задати два типи анімацій:

– Property Animation (анімація якості). У цьому випадку анімація відбувається через модифікацію значень властивостей об'єкта через встановлений період з використанням класу Animator;

– View Animation (анімація вигляду). Цей тип включає ще два підтипи анімацій, які ви можете створити в робочому оточенні View: Tween animation – анімація створюється через серію перетворень однієї картинки в класі Animation; Frame animation – в цьому випадку анімація буде показана як послідовність зображень, що чергуються один за одним, що робиться за допомогою класу AnimationDrawable.

Анімація Tween задається в XML і виконує такі перетворення, як поворот, затінення (fading), переміщення (moving) та розтягування/стискання (stretching) елементів графіки.

Анімація Frame також задається в XML і працює як показ послідовності зображень (так само, як влаштований класичний кінематограф).

Ресурс Color

Color – це ціле число, що кодує колір. Ресурс кольору Color – це ціле число, яке задано в XTML. Колір вказується у вигляді значення RGB та каналу альфа (alpha channel – число, яке кодує прозорість кольору). Можна використовувати ресурс кольору в будь-якому місці, яке набуває значення кольору в шістнадцятковому вигляді. Також можна використовувати ресурс кольору там, де мається на увазі використання XML графічного (drawable) ресурсу.

Ресурс Color State List

Це так званий колірний перелік станів. ColorStateList є об'єктом, заданим у XML, який ви можете застосувати як колір, але він насправді змінюватиме кольори залежно від стану об'єкта View, до якого застосований. Наприклад, віджет кнопки Button може існувати в одному з деяких різних станів (кнопка натиснута, отримала фокус або ні одне, ні інше), так що ви можете, використовуючи список кольорів станів (color state list), надати різний колір для кожного стану кнопки.

Ресурси типу Drawable

Є кілька різних типів drawable-ресурсів:

- Bitmap File – растрова картинка, графічний файл із розширенням .png, *.jpg* чи .gif. Створюється за допомогою класу BitmapDrawable;

- Nine-Patch File – це файл PNG з регіонами, що розтягуються, використовується для зміни розмірів зображення, базуючись на його вмісті (.9.png). Створюється за допомогою класу NinePatchDrawable;

- Layer List – список шарів, спеціальний drawable-ресурс, який управляє масивом з інших drawable-ресурсів. Цей список відображається в порядку проходження елементів у масиві, так що елемент з найбільшим індексом буде намальований на вгорі. Створюється за допомогою класу LayerDrawable;

- State List – список станів, файл XML, який містить посилання на різні растрові (bitmap) графічні зображення для різних станів (наприклад, може використовуватися для анімації кнопки, коли вона натиснута). Створюється за допомогою класу StateListDrawable;

- Level List – список рівнів, файл XML, що визначає drawable-ресурс, який керує деякою кількістю альтернативних Drawables, кожному з яких надано максимальне числове значення. Створюється за допомогою класу LevelListDrawable;

- Transition Drawable – файл XML, що описує drawable, який може перетікати (cross-fade) між двома drawable-ресурсами. Створюється за допомогою класу TransitionDrawable;

- Inset Drawable – файл XML, що визначає drawable, який буде вставлений в інший drawable із зазначеною відстанню. Це корисно, коли для View потрібен Рисунок заднього плану (background drawble), який менший, ніж реальні межі View;

- Clip Drawable – файл XML, що визначає drawable, який відсікає інший drawable на основі значення поточного рівня. Створюється за допомогою класу ClipDrawable;

- Scale Drawable – файл XML, що задає drawable, змінює розмір іншого Drawable на основі значення його поточного рівня. Створюється за допомогою класу Scale Drawable.

Робота з файловою системою Android

ОС Android побудовано на основі Linux. Цей факт знаходить своє відображення у роботі з файлами. Так, у шляхах до файлів як розмежувач у

Linux використовує «/», а не зворотний «\» (як у Windows).

Програма Android зберігає свої дані в каталозі /data/data/<назва_пакета>/ і, як правило, щодо цього каталогу буде йти робота.

Усі файли, які створюються та редагуються в програмі, зазвичай зберігаються в підкаталозі /files в каталозі програми.

Для безпосереднього читання та записування файлів застосовуються також стандартні класи Java з пакету java.io.

Система дозволяє створювати файли з двома різними режимами: MODE_PRIVATE – файли можуть бути доступні лише власнику програми (за замовчуванням режим); MODE_APPEND – дані можуть бути додані до кінця файлу.

Формат зберігання даних JSON

JSON – текстовий формат обміну даними, що базується на JavaScript. За рахунок своєї лаконічності, порівняно з XML, формат JSON може бути більш підходящим для серіалізації складних структур. Якщо говорити про web-додатки, то в такому ключі він доречний у завданнях обміну даними як між браузером та сервером (AJAX), так і між самими серверами (програмні HTTP-сполучення).

Оскільки формат JSON є підмножиною синтаксису мови JavaScript, він може бути швидко десеріалізований вбудованою функцією eval(). Крім того, можлива вставка цілком працездатних JavaScript-функцій. У мові PHP, починаючи з версії 5.2.0, підтримка JSON включена в ядро у вигляді функцій json_decode() та json_encode(), які самі перетворять типи даних JSON у відповідні типи PHP, і навпаки.

Підключення та використання сторонніх бібліотек

Android Studio дозволяє підключити три типу бібліотек у свій проект:

1) з репозиторію Maven;
2) бібліотеку у вигляді зібраного .jar файлу;
3) бібліотеку як вихідний код.

Відкриваємо вікно структури проекту (File → Project Structure) та переходимо до основного модуля проекту – зліва знаходиться секція Modules, слід натиснути на назву модуля цієї програми (рис. 4.8).

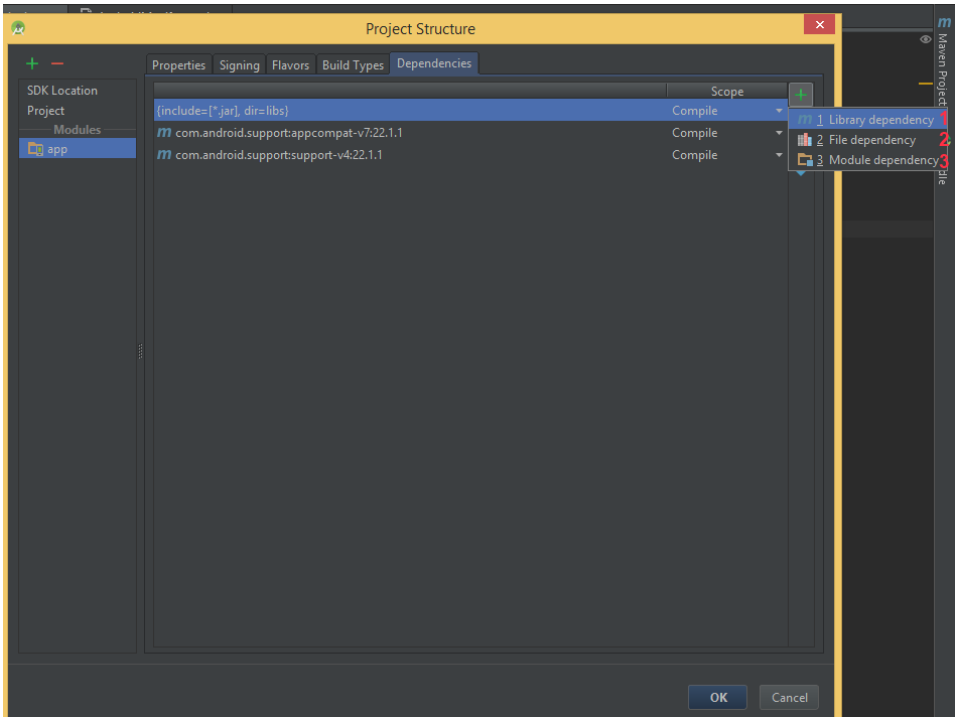

Рис. 4.8. Вікно Project Structure

У вікні переходимо до вкладки Dependencies. Тут можна керувати підключеними залежностями: додати нові, видалити непотрібні та переміщати їх ієрархією.

Маніфест. Огляд файлу manifest.xml

Файл маніфесту AndroidManifest.xml надає основну інформацію про програму. Кожна програма повинна мати свій файл AndroidManifest.xml. Редагувати файл маніфесту можна вручну, змінюючи XML-код або через візуальний редактор Manifest Editor (редактор файлу маніфесту), який дозволяє здійснювати візуальне та текстове редагування файлу маніфесту програми.

Призначення маніфесту:

- оголошує ім'я Java-пакета програми, яка є унікальним ідентифікатором;

- описує компоненти програми (діяльність, служби, приймачі широкомовних намірів та контент-провайдери, що дозволяє викликати класи, які реалізують кожен із компонентів) та оголошує їх наміри;
- містить список необхідних дозволів для звернення до захищених частин API та взаємодії з іншими програмами;
- оголошує дозволи, які сторонні додатки повинні мати для взаємодії з компонентами цієї програми;
- оголошує мінімальний рівень API Android, необхідний роботи програми;
- перераховує пов'язані бібліотеки.

Файл маніфесту інкапсулює всю архітектуру Android-програми, його функціональні можливості та конфігурацію. У процесі розробки програми вам доведеться постійно редагувати цей файл, змінюючи його структуру та доповнюючи новими елементами та атрибутами.

Підключення віджетів

Віджети є структурними елементами, з яких складається інтерфейс користувача. Віджет може виводити текст або графіку, взаємодіяти з користувачем або розміщувати інші віджети на екрані. Кнопки, текстові поля, прапорці – це різновиди віджетів.

Щоб створити найпростіший віджет, знадобляться три деталі:

1) Layout-Файл;
2) XML-файл з метаданими;
3) клас, успадковуючи AppWidgetProvider.

Layout-файл

У ньому формується зовнішній вигляд віджету. Все аналогічно layout-файлам для Activity та фрагментів, тільки набір доступних компонентів тут обмежений наступним списком:

- FrameLayout;

- LinearLayout;
- RelativeLayout;
- GridLayout;
- AnalogClock;
- Button;
- Chronometer;
- ImageButton;
- ImageView;
- ProgressBar;
- TextView;
- ViewFlipper;
- ListView;
- GridView;
- StackView;
- AdapterViewFlipper;

XML-файл з метаданими

У файлі задаються різні характеристики віджету. Файлу властиві такі параметри:

- layout-файл, щоб віджет знав, як він виглядатиме;
- розмір віджету, щоб віджет знав, скільки місця він має зайняти на екрані;
- інтервал оновлення, щоб система знала, як часто їй потрібно буде оновлювати віджет.

Клас AppWidgetProvider

У цьому класі нам треба буде реалізувати Lifecycle-методи віджету.

Наведемо схему створення віджету та його lifecycle-методів. Створимо проект без Activity:

Project name: P1171_SimpleWidget Build Target: Android 2.3.3 Application name: SimpleWidget

Package name: p1171simplewidget

Додамо рядки до strings.xml:

```
<string name="widget_name">My first widget</string>
<string name="widget_text">Text in widget</string>
```

Створюємо layout-файл widget.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="match_parent"
android:layout_height="match_parent" android:orientation="vertical">
<TextView android:id="@+id/tv"
android:layout_width="match_parent"
android:layout_height="match_parent" android:background="#6600ff00"
android:gravity="center" android:text="@string/widget_text"
android:textColor="#00ze" android ="18sp">
</TextView>
</RelativeLayout>
```

RelativeLayout, а всередині зелений TextView з текстом по центру. Тобто віджет просто показуватиме текст на зеленому тлі.

Створюємо файл метаданих res/xml/widget_metadata.xml:

```
<appwidget-provider
xmlns:android=http://schemas.android.com/apk/res/android
android:initialLayout="@layout/widget" android:minHeight="40dp"
android:minWidth="110dp" android:updatePeriodMillis="2400000">
</appwidget-provider>
```

В атрибуті initialLayout вказуємо layout-файл для віджету.

Атрибути minHeight та minWidth містять мінімальні розміри віджету за висотою та шириною.

Існує певний алгоритм розрахунку цих цифр. При розміщенні віджета екран ділиться на комірки, і віджет займає одну або кілька з цих осередків шириною та висотою. Щоб конвертувати комірки в dp, використовується формула 70 n – 30, де n – це кількість осередків. Тобто, якщо, наприклад, хочемо, щоб віджет займав два осередки завширшки і один заввишки, то вираховуємо ширину – 70*2 – 30 = 110 – і висоту – 70*1 – 30 = 40. Ці отримані значення і будемо використовувати в атрибутах minWidth та minHeight.

Атрибут updatePeriodMillis містить кілька мілісекунд (мс). Це інтервал оновлення віджету. Тут можна вказати інтервал хоч від 5 с, але частіше ніж один раз на 30 хв, віджет оновлюватись все одно не буде – це системне обмеження. Наприклад поставимо інтервал 40 хв.

Приступаємо до створення класу, який успадковує AppWidgetProvider.

```
package p1171simplewidget; import java.util.Arrays;
import android.appwidget.AppWidgetManager; import
android.appwidget.AppWidgetProvider; import android.content.Context;
import android.util.Log;
public class MyWidget extends AppWidgetProvider { final String
LOG_TAG = "myLogs";
@Override
public void onEnabled(Context context) { super.onEnabled(context);
Log.d(LOG_TAG, "onEnabled");
}
@Override
public void onUpdate(Context context, AppWidgetManager
appWidgetManager,int[] appWidgetIds) {
super.onUpdate(context, appWidgetManager, appWidgetIds);
Log.d(LOG_TAG, "onUpdate" + Arrays.toString(appWidgetIds));
```

```
}
@Override
public void onDeleted(Context context, int[] appWidgetIds) {
super.onDeleted(context, appWidgetIds);
Log.d(LOG_TAG, "onDeleted" + Arrays.toString(appWidgetIds));
}
@Override
public void onDisabled(Context context) { super.onDisabled(context);
Log.d(LOG_TAG, "onDisabled");
}
}
```

onEnabled викликається системою при створенні першого екземпляра віджета (адже ми можемо додати в Home кілька екземплярів одного і того ж віджету).

onUpdate викликається при оновленні віджету. На вхід, крім контексту, метод отримує об'єкт AppWidgetManager та список ID-екземплярів віджетів, які оновлюються. Саме цей метод зазвичай містить код, який оновлює вміст віджету. Для цього потрібно буде AppWidgetManager, який ми отримуємо на вхід.

onDeleted викликається при видаленні кожного екземпляра віджету. На вхід, крім контексту, метод отримує список ID-екземплярів віджетів, що видаляються.

onDisabled викликається при видаленні останнього екземпляра віджету.

У всіх методах виводимо в лог однойменний текст і список ID для *onUpdate*.

В onUpdate створюється код для якогось оновлення віджету. Тобто, якщо віджет відображає, наприклад, поточний час, то при черговому оновленні (дзвінку onUpdate) треба отримувати поточний час і передавати цю інформацію у віджет для відображення.

Залишилося заповнити маніфест (рис. 4.9).

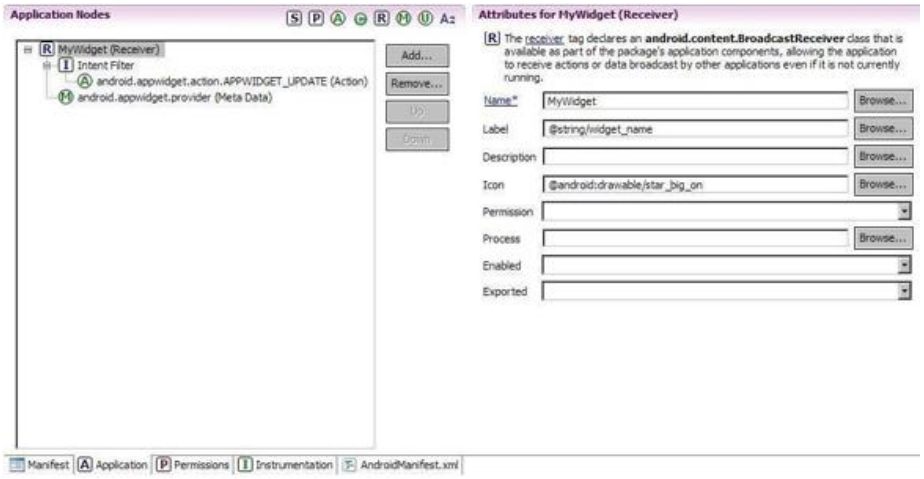

Рис. 4.9. Маніфест

Додайте туди клас Receiver:

- вкажіть йому свої label і icon. Цей текст і цю іконку ви побачите у списку вибраних віджетів, коли додаватимете віджет на екран;
- налаштуйтедля нього фільтр з *action* = android.appwidget.action. APPWIDGET_UPDATE;
- додайте метадані з іменем android.appwidget.provider і вказівкою файлу метаданих xml/widget_metadata.xml як ресурс.

Після цього в маніфесті має вийти приблизно така секція Receiver:

```
<receiver android:name="MyWidget"
android:icon="@android:drawable/star_big_on"
android:label="@string/widget_name">
<intent-filter>
<action
android:name="android.appwidget.action.APPWIDGET_UPDATE">
</action>
</intent-filter>
<meta-data android:name="android.appwidget.provider"
```

```
android:resource="@xml/widget_metadata">
    </meta-data>
    </receiver>
```

Віджет готовий. Все зберігаємо та запускаємо. Жодних Activity, зрозуміло, не спливе. У консолі має з'явитися текст:

```
\P1171_SimpleWidget\bin\P1171_SimpleWidget.apk installed on device Done!
```

Відкриваємо діалог створення віджету та знаходимо у списку наш віджет з іконкою та текстом, які вказували у маніфесті для Receiver (рис. 4.10).

Рис. 4.10. Список віджетів

Вибираємо його та додаємо на екран (рис. 4.11).

Рис. 4.11. Віджет на робочому столі

Дивимося логи:

onEnabled onUpdate [8]

Відразу після цього додавання спрацював метод onUpdate для конкретного екземпляра. Бачимо, що йому призначено ID = 8. Тобто система додала екземпляр віджету на екран і викликала метод оновлення із зазначенням ID-екземпляра.

Додаємо ще один примірник (рис. 4.12).

Рис. 4.12. Додавання другого віджету на робочий стіл

Дивимося лог:

onUpdate [9]

onEnabled не спрацював, так як екземпляр віджету, що додається, вже не перший. onUpdate знову відпрацював для нового доданого екземпляра і отримав на вхід ID = 9.

Тепер розглянемо видалення з екрана двох цих екземплярів віджету.

Спочатку видаляється останній. У логах побачимо:

onDeleted [9]

Спрацював наDeleted і отримав на вхід ID видаленого екземпляра віджета. Видаляємо перший екземпляр. У логах відображається:

onDeleted [8] onDisabled

Віджет оновлюється (отримує виклик методу onUpdate) один раз на 40 хв. Додайте знову віджети на екран і зачекайте. Коли вони знову оновляться, це в логах позначиться. Так створюється та працює простий віджет. Звернімо увагу на деякі нюанси.

Клас AppWidgetProvider є розширенням класу BroadcastReceiver (у маніфесті ми його прописали як Receiver). Він просто отримує від системи повідомлення в onReceive, визначає за значеннями з Intent, яка саме подія відбулася (додавання, видалення або оновлення віджету), і викликає відповідний метод (onEnabled, onUpdate тощо).

Якщо ми розташуємо кілька примірників віджету, побачимо наступний його стан (рис. 4.13).

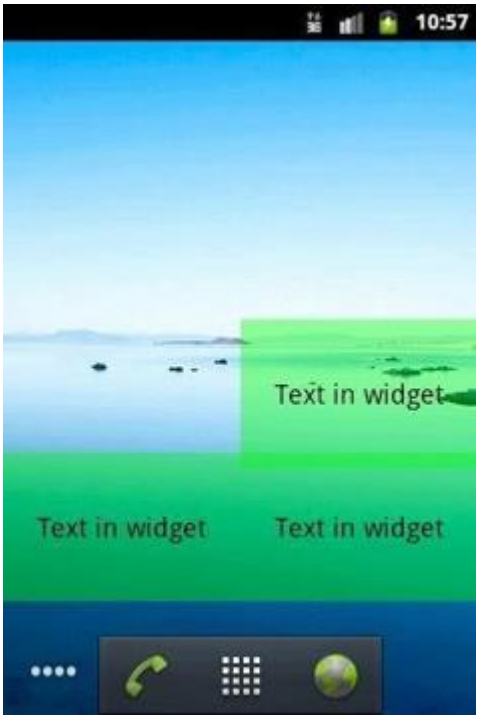

Рис. 4.13. Розташування віджетів

Додамо android:padding="8dp" до RelativeLayout у нашому layout-файлі (рис. 4.14).

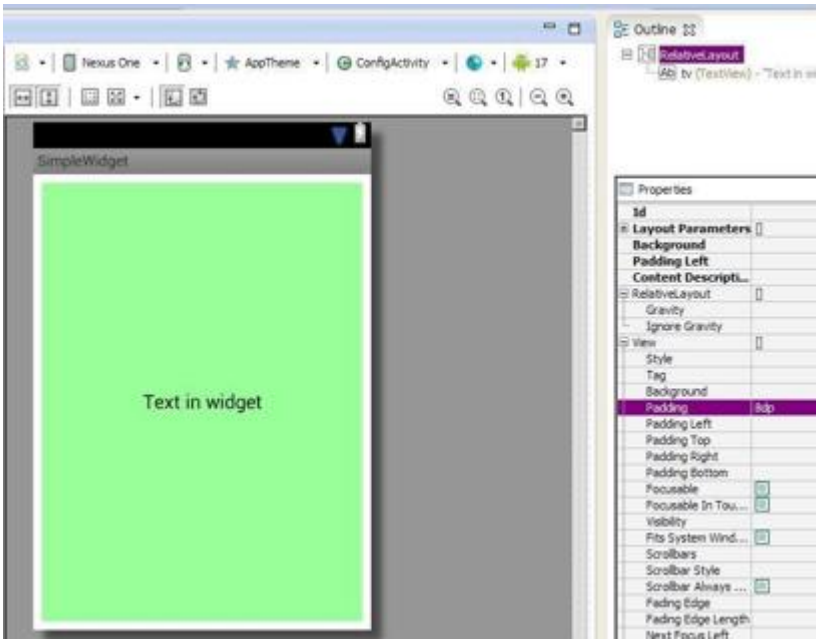

Рис. 4.14. Додавання відступів

Зберігаємо, запускаємо. Віджети на екрані змінилися автоматично і тепер відображаються коректно (рис. 4.15).

Рис. 4.15. Перевірка роботи віджетів з відступами

В Android 4.0 (API Level 14) та пізніших версіях ця незручність з відступами була усунена, і необхідність робити відступи вручну відпала.

Практична частина

1. Вивчити теоретичну частину.
2. Додати у програму, розроблену у лабораторній роботі № 3 елементи управління (Button, TextView, EditText, ListView). Запустити програму на емуляторі.
3. Оформити звіт з лабораторної роботи.

Контрольні питання

1. Дайте визначення поняття «емулятор».
2. Назвіть основні етапи встановлення емулятора Android.
3. Наведіть види емуляторів. Назвіть їх переваги та недоліки.
4. Назвіть основні ресурси Android.
5. Назвіть методи роботи з файлами.
6. Охарактеризуйте формат зберігання даних JSON.
7. Назвіть основні етапи підключення бібліотек до Android Studio.
8. Назвіть призначення файлу manifest.xml та охарактеризуйте його.

ЛАБОРАТОРНА РОБОТА 5. АРХІТЕКТУРА ДОДАТКІВ ТА ДОПОМІЖНІ БІБЛІОТЕКИ

Мета роботи: вивчити особливості архітектури програм для операційної системи Android та виділити особливості операційної системи Android як платформи для розробки програм.

Теоретичні положення

Огляд архітектури програми Android

Екосистема засобів розробки під Android розвивається дуже швидко. Щотижня хтось створює нові інструменти, оновлює існуючі бібліотеки, пише нові статті чи виступає із доповідями.

До 2012 року структура проектів виглядала дуже просто. Не було жодних бібліотек для роботи з мережею, і тільки AsyncTask надавав допомогу. На рис. 5.1 наведено архітектуру цих рішень.

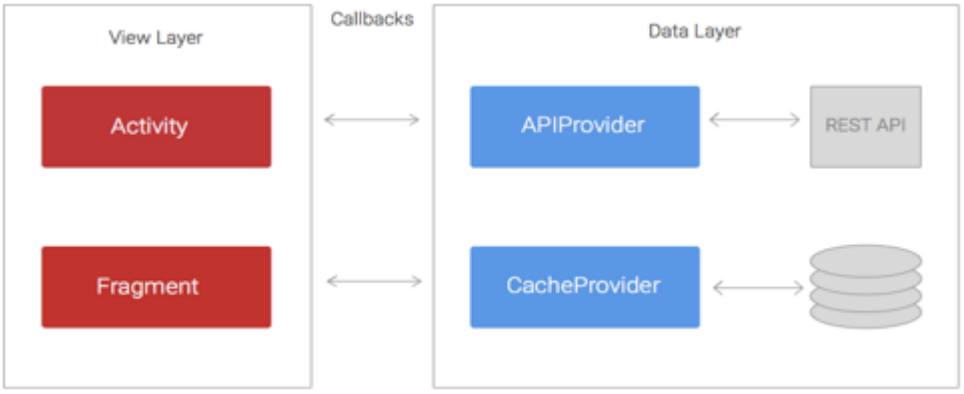

Рис. 5.1. Архітектура проектів до 2012 р.

Код був поділений на два рівні: рівень даних (data layer), який відповідав за отримання/збереження даних, що одержуються через REST API, так і через різні локальні сховища, і рівень вигляду (view layer), що відповідає за обробку та відображення даних.

APIProvider надає методи, що дозволяють Activity та Fragment взаємодіяти з REST API. Ці методи використовують URLConnection і

AsyncTask для виконання запиту у фоновому потоці, а потім доставляють результати в Activity через функції зворотного виклику. Аналогічно працює і CacheProvider: є методи, які дістають дані із SharedPreferences або SQLite, і є функції зворотного виклику, які повертають результати.

Головна проблема такого підходу полягаєу тому, що рівень вигляду має надто багато відповідальності. Простий сценарій, в якому програма має завантажити список постів з блогу, закешувати їх у SQLite, а потім відобразити в ListView. Activity має виконати наступне:

1. Викликати метод APIProvider#loadPosts(Callback).
2. Зачекати виклик методу onSuccess() у переданому Callback'e і потім викликати CacheProvider#savePosts(Callback).
3. Почекати виклик методу onSuccess() в переданому Callback'e і відобразити дані в ListView.
4. Окремо обробити дві можливі помилки, які можуть виникнути як в APIProvider, так і CacheProvider.

На практиці може статися так, що API поверне дані не в тому вигляді, в якому їх очікує наш рівень, відповідно Activity повинна якось трансформувати та відфільтрувати дані перш, ніж зможе з ними працювати. Наприклад loadPosts() прийматиме аргумент, який потрібно звідкись отримати (наприклад, адресу електронної пошти, яку запросимо через Play Services SDK). Напевно, SDK повертатиме адресу асинхронно, через функцію зворотного повідомлення, а отже, тепер є три рівні вкладення функцій зворотного повідомлення.

Ситуація почала змінюватися у 2014 році, коли вийшла версія RxJava. RxJava дозволяє керувати даними через асинхронні потоки і надає безліч операторів, які можна застосовувати до потоків, щоб трансформувати, фільтрувати, або комбінувати дані так, як потрібно.

Запропонованана архітектура нової програми (рис. 5.2).

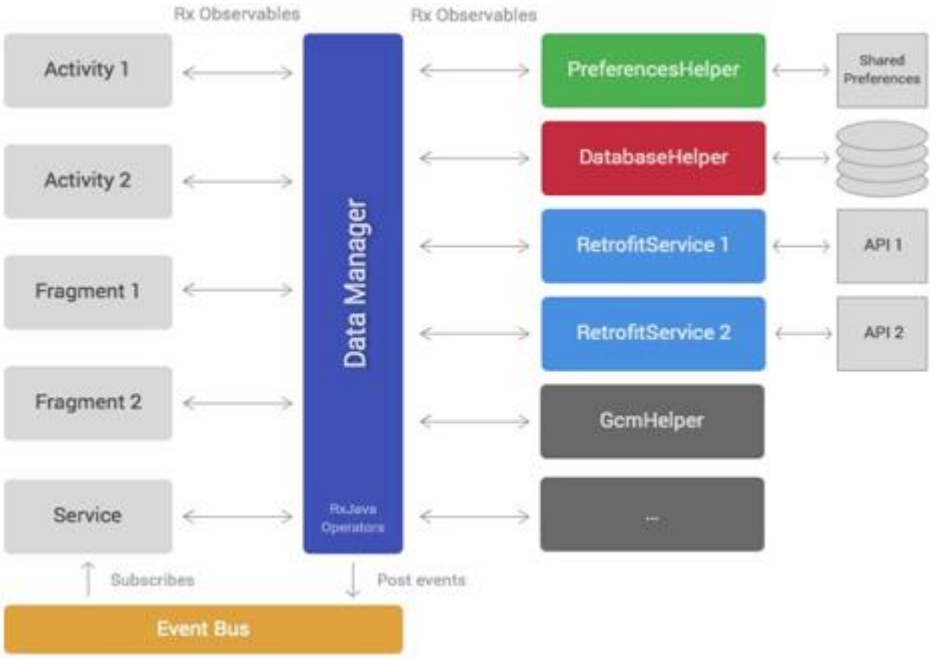

Рис. 5.2. Архітектура нової програми

Код так само розділений на два рівні: рівень даних, який містить DataManager і набір класів-помічників, і рівень вигляду, що складається з класів Android SDK, таких як Activity, Fragment, ViewGroup тощо.

Класи-помічники (третя колонка в діаграмі) мають дуже обмежені сфери відповідальності, і реалізують їх у певній послідовності. Наприклад, більшість проектів мають класи доступу до REST API, читання даних із бази даних (БД) чи взаємодії з SDK від сторонніх виробників. У різних додатків буде різний набір класів-помічників, але найчастіше використовуються такі:

- PreferencesHelper: працює з даними у SharedPreferences;
- DatabaseHelper: працює з SQLite;
- сервіси Retrofit, що виконують звернення до REST API. Використання Retrofit замість Volley краще, тому що він підтримує роботу з RxJava, причому API у нього краще.

Багато методів класів-помічників повертають RxJava Observables.

DataManager є центральною частиною нової архітектури. Широко використовує оператори RxJava для того, щоб комбінувати, фільтрувати та трансформувати дані, отримані від помічників. Завдання DataManager полягає

в тому, щоб звільнити Activity і Fragment від роботи зі «згладжування» даних - він буде проводити всі необхідні трансформації всередині себе і віддавати назовні дані, готові до відображення.

Компоненти рівня вигляду просто викликатимуть цей метод і підписуватимуться на повернутий ним Observable. Як тільки підписка завершиться, пости, повернені отриманим Observable, можуть бути додані в Adapter, щоб відобразити їх у RecyclerView.

Останній елемент цієї архітектури – це event bus. Event bus дозволяє нам запускати повідомлення про деякі події, що відбуваються лише на рівні даних, а компоненти, що є лише на рівні вигляду, можуть підписуватися на ці повідомлення. Наприклад, метод signOut() у DataManager може запустити повідомлення, що повідомляє про те, що відповідний Observable завершив свою роботу, і тоді Activity, підписані на цю подію, можуть перемалювати свій інтерфейс, щоб показати, що користувач вийшов із системи.

Переваги цього підходу:

- Observables і оператори з RxJava позбавляються функцій зворотного виклику;
- DataManager бере на себе роботу, яка раніше виконувалася на рівні вигляду, розвантажуючи таким чином Activity і Fragment;
- переміщення частини коду в DataManager і класи-помічники робить unit-тестування Activity та Fragment більш простим;
- ясний поділ відповідальності та виділення DataManager як єдиної точки взаємодії з рівнем даних робить усю архітектуру адаптованою до тестування.

В Android-спільноті почали набирати популярності окремі архітектурні шаблони, такі як MVP або MVVM (рис. 5.3). Виявили, що MVP може привнести значні зміни в архітектуру проектів.

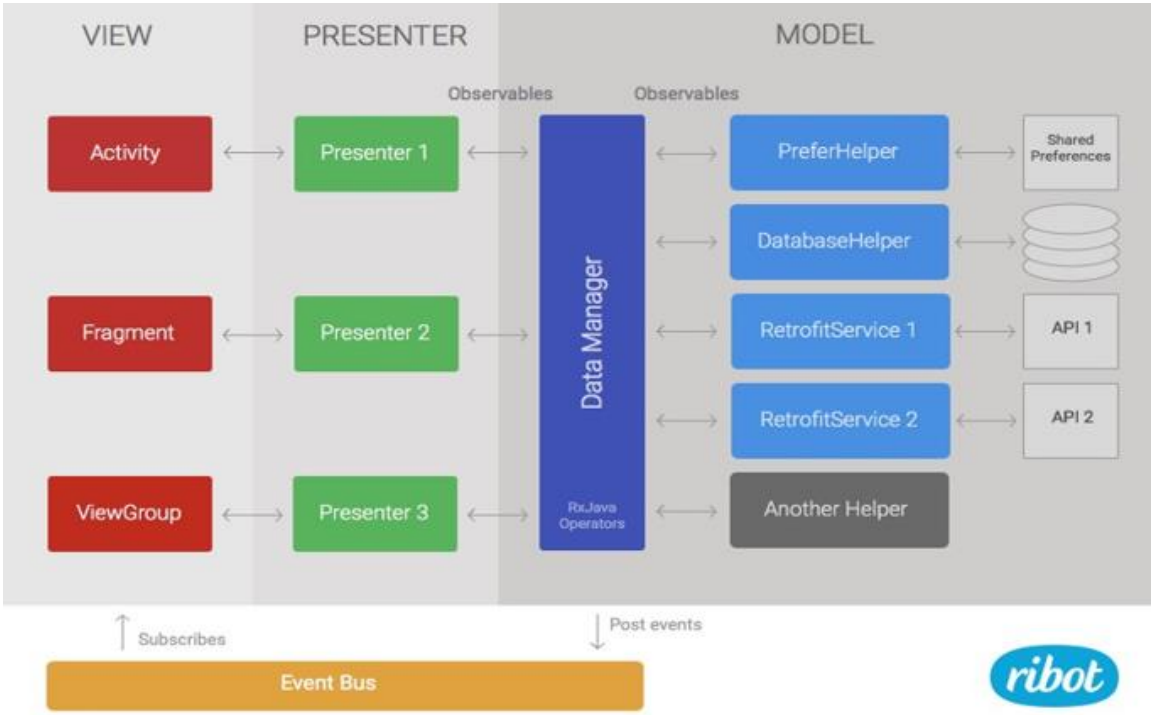

Рис. 5.3. Архітектурний шаблон MVP

Рівень даних залишається незмінним, але він називається моделлю, щоб відповідати імені відповідного рівня з MVP.

Підключення та використання сторонніх бібліотек

Дуже часто під час створення програми доводиться користуватися сторонніми бібліотеками під час вирішення тих чи інших завдань. Але не всі Android-розробники вміють підключати їх до свого проекту. Розглянемо, як підключаються бібліотеки. Як IDE використовуватиметься Eclipse.

Розберемо три випадки:

- підключення jar;
- підключення проекту із вихідними джерелами;
- що робити, якщо пункт 2 не спрацював.

Випадок перший: бібліотека постачається у jar-файлі

Це найлегший варіант підключення бібліотеки. На сьогоднішній день не дуже поширений і, швидше за все, втрачатиме популярність. Однією з причин є неможливість «запаковування» ресурсів разом із бібліотекою. Нещодавно з'явився новий формат бібліотек – aar. Це той же jar-файл тільки з ресурсами.

Тут також є два способи підключення jar-бібліотеки до проекту. Перший спосіб:

1. Розкрийте дерево проекту в Package Explorer.
2. Перетягніть у папку libs ваш jar-файл (якщо папки libs немає, то створіть її).
3. У вікні виберіть Copy files і натисніть кнопку OK.
4. Натисніть правою кнопкою миші на проекті та виберіть пункт *Refres*.

Другий спосіб:

1. Натисніть правою кнопкою миші на проекті та виберіть пункт *Properties*.
2. У вікні зліва є меню, виберіть в ньому пункт Java Build *Path*.
3. Виберіть вкладку Libraries.
4. Натисніть кнопку Add External JARs.
5. У вікні виберіть файл із вашою бібліотекою та натисніть OK.

Випадок другий: бібліотека підключається у вигляді проекту

1. Відкрийте File > Import, виберіть Android > Existing Android Code into Workspace.
2. Натисніть кнопку Browse.
3. У вікні знайдіть папку з бібліотекою, натисніть OK.
4. Поставте галочку на пункті Copy projects в workspace.
5. Натисніть Finish.
6. Натисніть правою кнопкою миші на проекті та виберіть пункт *Properties*.
7. У вікні зліва є меню, виберіть в ньому пункт Android.
8. Натисніть Add.
9. У вікні виберіть бібліотеку та натисніть OK.

Третій випадок: під час імпортування бібліотеки Eclipse видає помилку.

Якщо при імпортуванні бібліотеки в workspase Eclipse видає помилку, то виконуємо такі дії:

1. Виберіть File > New > Android Application Project.
2. У полях Application Name та Project Name напишіть назви бібліотеки, наприклад PullToRefreshLibrary.
3. Натисніть кнопку Next.
4. Заберіть галочку з пунктів Create custom launcher icon та Create activity. Поставте галочку на Mark this project as a library. Натисніть Finish.
5. Видаліть папки src, libs та res, а також файл AndroidManifest.xml.
6. Перетягніть папки src, libs, res і AndroidManifest.xml з бібліотеки до папки проекту.
7. У вікні виберіть Copy files and folders.
8. Виконайте дії з другого випадку, починаючи з пункту 6.

Практична частина

1. Вивчити теоретичну частину.
2. Додати у програму, розроблену у лабораторній роботі № 4 сторонні бібліотеки, різними методами. Зробіть висновок про зручність цих методів. Скористайтеся бібліотеками з відкритих джерел Інтернету.
3. Оформити звіт з лабораторної роботи.

Контрольні питання

1. Охарактеризуйте архітектуру Android-програми.
2. Назвіть класи-помічники.
3. Назвіть основні етапи підключення сторонніх бібліотек у проект.

ЛАБОРАТОРНА РОБОТА 6. БАЗА ДАНИХ MYSQL

Мета роботи: вивчити принцип використання бази даних MySQL для серверних завдань та керування базою даних MySQL.

Теоретичні положення

Встановлення та налаштування бази даних MySQL

Сервер баз даних MySQL є одним із найбільш популярних серед серверів баз даних з відкритим вихідним кодом, які використовуються при розробці web-додатків.

Розглянемо послідовність дій щодо настроювання сервера бази даних MySQL версії 5.6 в ОС Windows. Відомості про конфігурацію MySQL не розглядаються, наводиться лише послідовність необхідних кроків. Останню версію MySQL можна завантажити на офіційному сайті.

Наповнення бази

Корисним та зручним інструментом, який дозволяє створювати бази даних MySQL та працювати з ними, є Denwer. Він також дає змогу тестувати код PHP. Розглянемо наповнення бази даних MySQL за допомогою web-програми для адміністрування систем управління базами даних (СУБД) MySQL phpMyAdmin. Перейдемо на сторінку адміністрування бази даних MySQL (рис. 6.1), ввівши адресу: localhost/tools/phpmyadmin/ у командному рядку браузера.

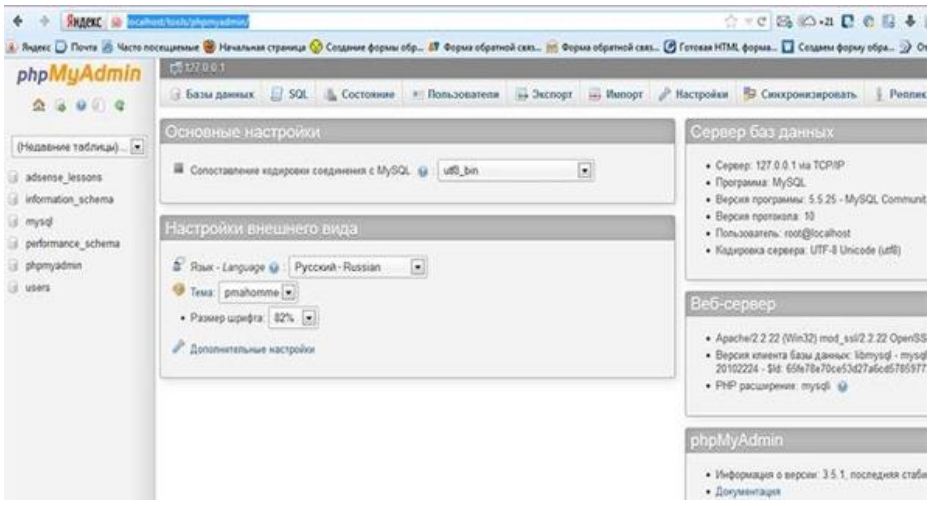

Рис. 6.1. Стартова сторінка phpMyAdmin

Створення бази даних MySQL

Щоб створити нову базу даних, необхідно натиснути на верхню вкладку «Бази даних» – на центральному полі відкриється список всіх баз даних MySQL. Потрібно створити нову. Для цього в полі «Створити базу даних» потрібно вписати назву бази, що створюється, і натиснути на кнопку «Створити» (рис. 6.2). Після натискання кнопки «Створити» база даних додасться до списку баз даних у панелі зліва та на центральному полі. Тепер оберіть нову базу даних.

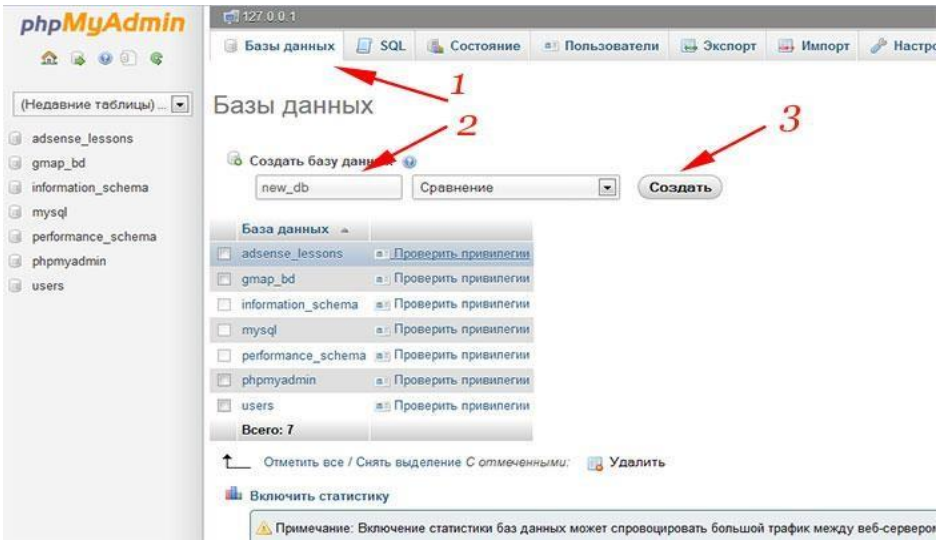

Рис. 6.2. Створення нової бази даних

Тут буде запропоновано створити таблицю (рис. 6.3). Для цього заповніть поля «Ім'я» та «Кількість стовпців» та натисніть OK.

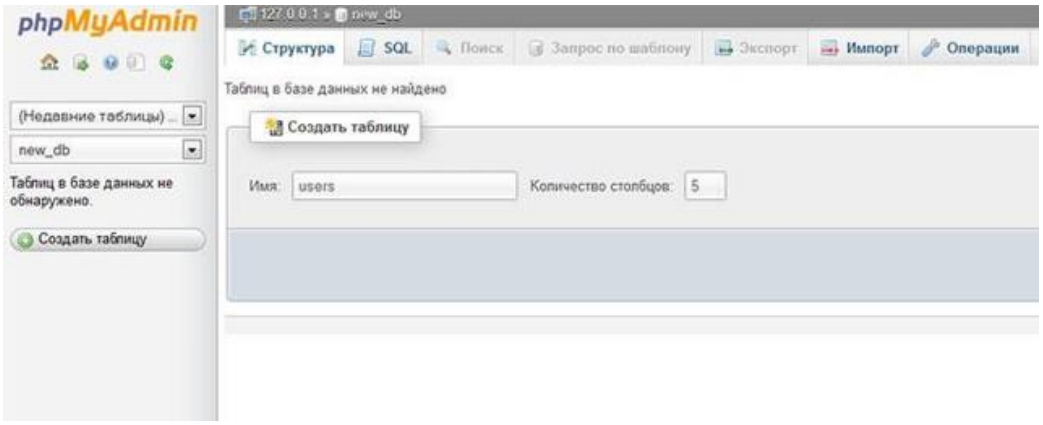

Рис. 6.3. Створення нової таблиці

Після цього відкриється сторінка заповнення полів нової таблиці бази даних (рис. 6.4). Тут кожному полю потрібно присвоїти ім'я, тип збережених даних, довжину (якщо потрібно для даного атрибута) і для такого поля, як ідентифікатор (id), також потрібно вказати авто-інкремент та первинний ключ. Це має виглядати так, як на скріншоті (рис. 6.4).

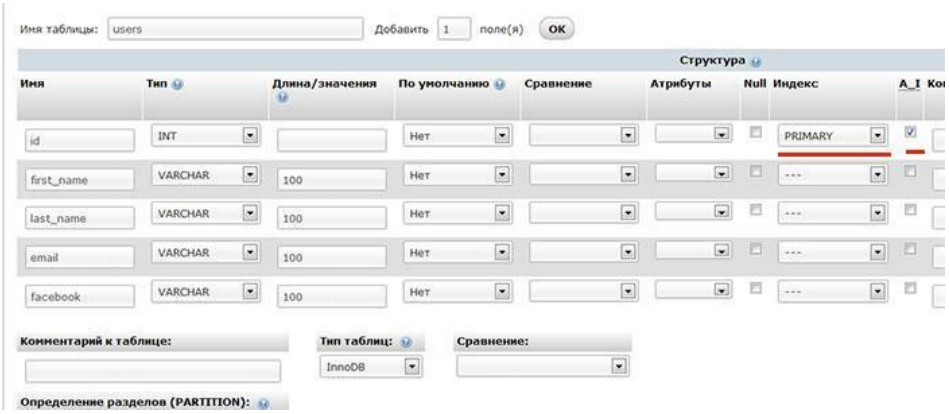

Рис. 6.4. Сторінка для заповнення полів нової таблиці

Існують різні типи даних, призначені для зберігання дати, тексту та інших даних. Далі натискаємо кнопку «Зберегти», і відкривається створена таблиця, в якій поки що немає жодного запису (рис. 6.5). Таблиця з'явиться на панелі ліворуч і на центральній частині екрана. Натисніть на її ім'я, щоб побачити структуру.

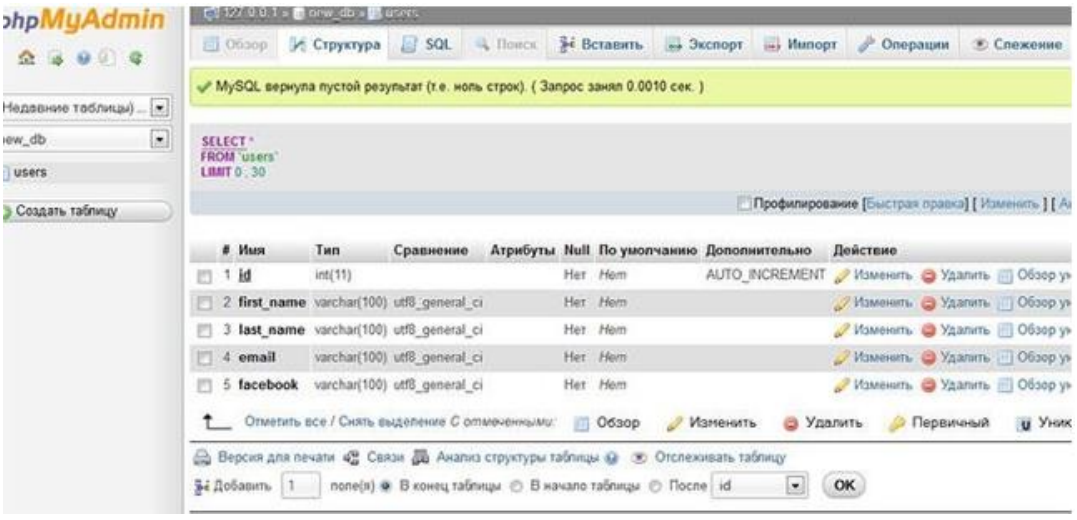

Рис. 6.5. Структура таблиці

Вставлення нового елементу в таблицю бази даних

Для цього натисніть на верхній вкладці «Вставити» – відкриється сторінка для вставлення нового елемента в таблицю бази (рис. 6.6). Заповніть усі поля (крім поля id, воно заповнюватиметься автоматично) та натисніть кнопку ОК.

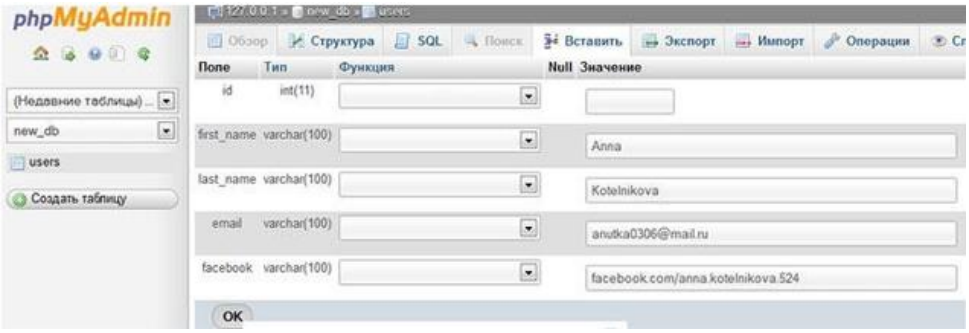

Рис. 6.6. Вставлення нового елемента до таблиці

Після натискання ОК перейдіть до вкладки «Огляд» (вона знаходиться вгорі) – ви побачите новий доданий елемент у таблиці бази даних MySQL (рис. 6.7).

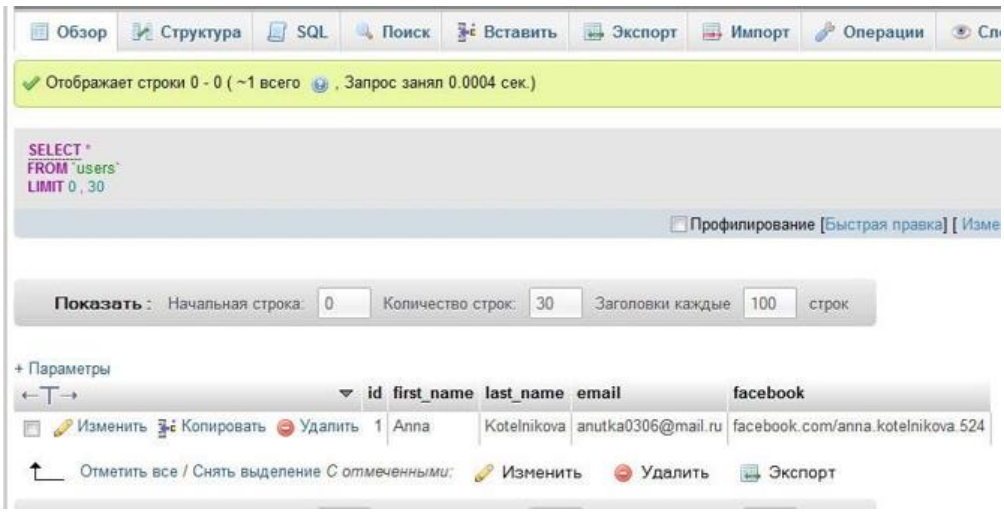

Рис. 6.7. Перегляд елемента таблиці бази даних

Створити нового користувача для бази даних – значить створити йому ім'я та пароль і виставити певні привілеї. Інформація про імені користувача та паролі знадобиться, коли з'єднуватимемося з базою за допомогою PHP-скрипту.

Отже, створюємо нового користувача бази даних (рис. 6.8). Для цього натискаємо вгорі на назву бази даних, після цього у верхніх вкладках з'явиться пункт "Привілеї", натискаємо по ньому.

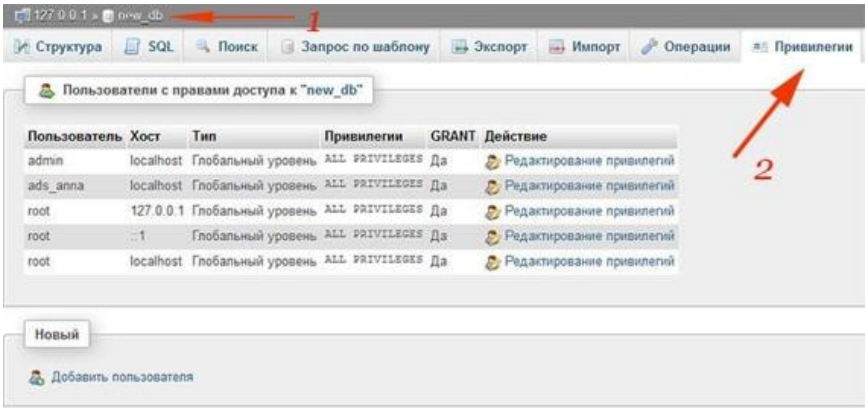

Рис. 6.8. Створення нового користувача бази даних

Натискаємо «Додати користувача». Відкриється сторінка з полями, які потрібно заповнити (ім'я користувача, хост, пароль та підтвердження пароля) (рис. 6.9). Як хост потрібно вибрати локальний хост.

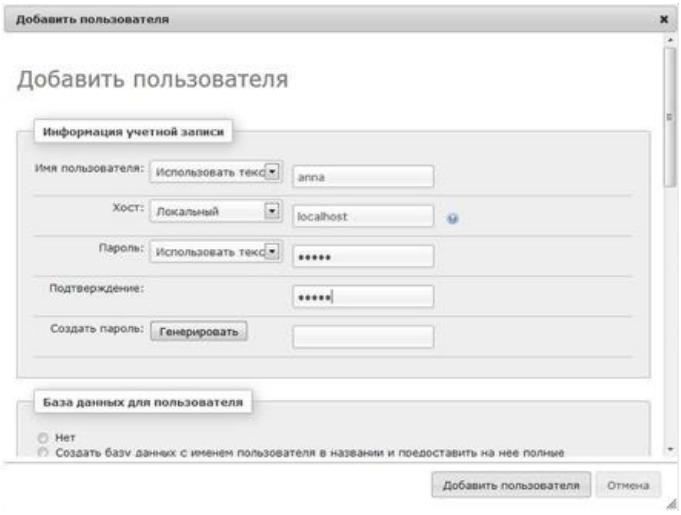

Рис. 6.9. Додавання нового користувача

Після чого натискаємо "Додати користувача", і новий користувач буде додано. Буде видно повідомлення, що доданий користувач до бази даних new_db з усіма привілеями.Тут також можна редагувати привілеї, натиснувши на редагування привілеїв. Це може знадобитися в тому випадку, якщо хтось повинен мати доступ до бази, але необхідно обмежити цю людину в привілеях (наприклад, вона не може видаляти дані). Тоді створюєте нового користувача бази даних, але виставляєте йому певні привілеї.

Щоб видалити базу даних, потрібно знову перейти до вкладки «Бази даних», вибрати базу для видалення та натиснути «Видалити» (рис. 6.10).

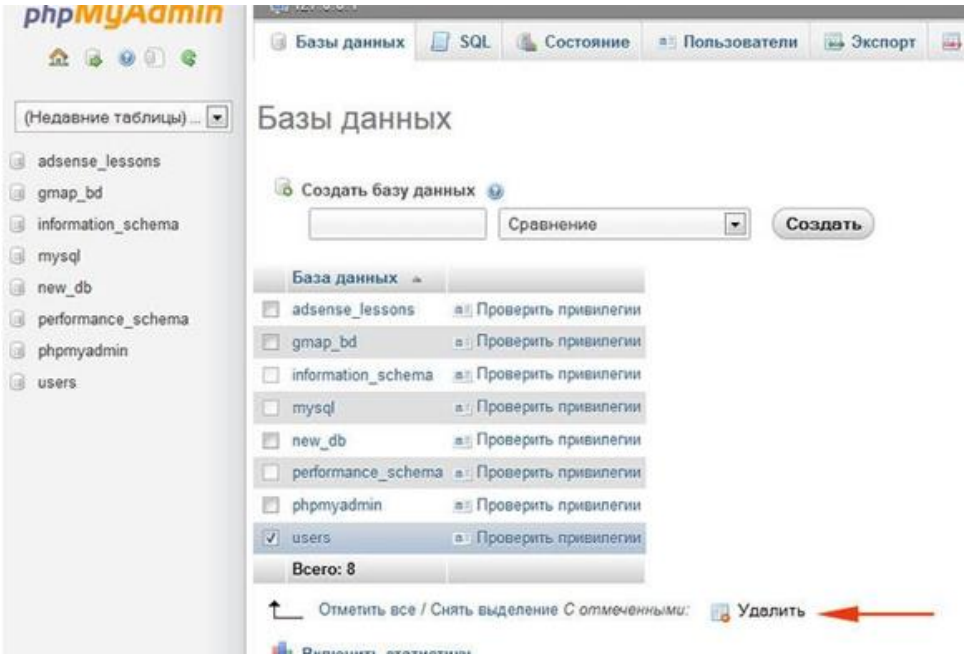

Рис. 6.10. Видалення бази даних

Бази даних MySQL та web-інтерфейс php MyAdmin дозволяють реалізовувати проекти різної складності схем даних.

Практична частина

1. Вивчити теоретичну частину.
2. Додати у програму, розроблену у лабораторній роботі № 5 функціонал, що виконує збереження всіх даних програми в БД Android та вибірку їх за потреби. Запустити програму на емуляторі.
3. Оформити звіт з лабораторної роботи.

Контрольні питання

1. Назвіть основні етапи встановлення бази даних MySQL.
2. Назвіть основні етапи заповнення бази даних MySQL.
3. Опишіть довготривале зберігання даних мобільного додатка.
4. Опишіть локальні бази даних та SQLite.
5. Як відбувається зберігання серій та позицій у базі даних?

ЛАБОРАТОРНА РОБОТА 7. МОВА ПРОГРАМУВАННЯ PHP

Мета роботи: вивчити основні принципи розробки додатків об'єктно-орієнтованою мовою програмування PHP та розглянути основні фреймворки мови PHP.

Теоретичні положення

Мова програмування PHP

PHP (Рекурсивний акронім словосполучення PHP (Hypertext Preprocessor) – це скриптова мова програмування загального призначення з відкритим вихідним кодом. PHP сконструйований спеціально для ведення web-розробок, і його код може впроваджуватися безпосередньо в HTML. Розглянемо синтаксис мови PHP.

Класи та об'єкти

Приклад оголошення класу:

```
class Article {
// Тіло класу
}
```

Об'єкт – сукупність конкретних даних та функцій для їх обробки. Фактично об'єкт – це змінна, тип даних якої задається відповідним класом.

Для оголошення об'єкта використовується ключове слово new:

```
$a = new Article();
```

Властивості та методи

Клас може містити методи та властивості.

Методи – це функції класу.

Властивості – це змінні класу. Значення властивостей може бути порожнє значення, число, рядок або масив.

```
class Article
{
var $id;
var $myint = 3; var $mystr = 'str';
var $myarr = array('Jesse');
```

```
public function content(){
//Вміст методу
}
}
```

При спробі привласнити будь-яке інше значення відбувається аварійне завершення PHP.

Щоб надати змінній неконстантне значення, потрібно виконати присвоєння всередині методу класу:

```
class Article{ var $str;
public function update($id){
$this->str = 'My id is '.$id;
}
}
```

де $this – спеціальна змінна, містить посилання на об'єкт поточного класу.

Символ → служить для звернення до методів та властивостей класу через об'єкт:

```
$a = new Article(); echo $a->id;
echo $a->content()
```

Успадкування

При наслідуванні використовується ключове слово extents:

```
class Foo
{
public function printItem($string)
{
echo 'Foo:'. $string. PHP_EOL;
}

public function printPHP()
{
```

```
echo 'PHP is great.' . PHP_EOL;
}
}
class Bar extends Foo
{
public function printItem($string)
{
echo 'Bar:'. $string. PHP_EOL;
}
}
$ Foo = New Foo ();
$ Bar = New Bar ();
$foo->printItem('baz'); // Виведе: 'Foo: baz'
$foo->printPHP();       //Виведе: 'PHP is great'
$bar->printItem('baz'); // Виведе: 'Bar: baz'
$bar->printPHP();       //Виведе: 'PHP is great'
```

Поліморфізм

Поліморфізм - взаємозамінність об'єктів з однаковим інтерфейсом. Мова програмування підтримує поліморфізм, якщо класи з однаковою специфікацією можуть мати різну реалізацію, наприклад, реалізація класу може бути змінена у процесі спадкування.

Розглянемо властивість поліморфності класів на основі такого прикладу:

```
class A {
// Виводить, функція якогось класу була викликана function Test() { echo "Test from A\n"; }
// Тестова функція – просто переадресує на Test() function Call() { Test(); }
}
class B extends A {
// Функція Test() для класу B
```

```
function Test() { echo "Test from B\n"; }
}
$a=new A();
$ b = New B ();
$a->Call(); // виводить "Test from A"
$b->Test(); // виводить "Test from B"
$b->Call(); // Увага! Виводить "Test from B"!
```

Специфікатори доступу до PHP

Модифікатор public дозволяє звертатися до властивостей та методів звідусіль. Модифікатор private дозволяє звертатися до властивостей та методів лише усередині поточного класу. Модифікатор protected дозволяє звертатися до властивостям і методам лише поточного класу та класу, який успадковує властивості та методи батьківського класу:

```
class Person { public $name; protected $age; private $salary;
public function_construct(){ } protected function set_age(){ } private function set_salary(){ }
}
```

Статичні методи

Крім символу «→», можна використовувати символ «::»для доступу до методів або властивостей. Статичні методи працюють незалежно один від одного та від інших методів та властивостей. У статичному методі не можна використовувати посилання $this.

Статичний метод виглядає так:

```
class Article{
public static function getUser(){
//Вміст методу.
}
}
Article::getUser()
```

Усередині класу, у якому визначено статичний метод, для звернення до

нього використовується позначення self:

```
class Article{
public static function getuser(){
//Вміст методу.
}
public function moreuser(){ return self::getuser();
}
}
```

Константи класу

Константи визначаються як властивості класу, але з використанням мітки const:

```
class Math{
const pi = 3.14159;
}
// math::pi
```

До констант, як і до статичних властивостей, можна звертатися без попереднього створення об'єкта, при цьому використовується синтаксис двох двокрапок (::).

Для звернення до константи всередині методу класу використовується префікс self:

```
class Math{
const pi = 3.14159; protected $radius; public function cirle(){
return self::pi * $this→radius;
}
}
```

Фреймворки PHP

Фреймворк – програмна платформа, що визначає структуру програмної системи; програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекту.

Переваги PHP-фреймворків:

- прискорюють процес розробки;
- допомагають писати структурований код, придатний для повторного використання;
- дозволяють легко масштабувати проекти;
- дотримуються схеми MVC (Model - View - Controller, модель - вигляд - контролер);
- заохочують сучасні практики розробки, наприклад об'єктно-орієнтоване програмування.

Розглянемо основні фреймфорки PHP.

SLIM

Slim – мікрофреймворк, що ідеально підходить для невеликих проектів або додатків, де повноцінний фреймворк здасться зайвим. Використовують багато PHP-розробників для створення RESTful API та сервісів. Серед функцій Slim – кешування HTTP на стороні клієнта, URL-маршрутизація, шифрування сесій та cookie, а також миттєві повідомлення за запитами HTTP. Документація повна та зроблена якісно.

PHALCON

Phalcon був створений у 2012 році та швидко набув популярності серед PHP-розробників. Вважається дуже швидким, оскільки написаний на C і C++ з метою досягати найвищого можливого рівня продуктивності. Знання C не потрібні, тому що вся функціональність укладена в PHP-класи, які можна використовувати з будь-якою метою.

Так як Phalcon спочатку був створений як розширення на C, його архітектура реалізована на низькому рівні, що значно знижує витрати ресурсів, типове для додатків, заснованих на схемі MVC. Phalcon не лише підвищує швидкість виконання, а й знижує рівень витрат ресурсів. У цього PHP-фреймворку є багато інших чудових функцій: універсальний автозавантажувач, менеджмент ресурсів, безпека, переклад, кешування. Документація для Phalcon досить велика, а використовувати його нескладно.

CAKEPHP

Фреймворк CakePHP теж досить популярний. Роботі з ним легко навчитися, а шаблонування – швидке і настроюється. Вбудована функція CRUD дуже допомагає під час взаємодії з базою даних.

В останньому релізі – CakePHP 3.x – покращилися керування сесіями та модульність, а також розширилися можливості створення більшої кількості окремих бібліотек. Обирайте CakePHP, якщо веб-додатку потрібен високий рівень безпеки. CakePHP містить такі функції безпеки:

- валідація введення;
- захист від атак з використанням SQL-ін’єкцій;
- запобігання міжсайтовому скриптингу;
- захист від підробки міжсайтових запитів.

ZEND FRAMEWORK 2

Zend Framework 2 – фреймворк з відкритим вихідним кодом, що використовується для розробки web-додатків та сервісів на PHP 5.3+. Він використовує в повному обсязі об'єктно-орієнтований код і більшість нових функцій PHP 5.3: простір імен, пізніше статичне зв'язування, лямбда-функції та замикання. Zend – надійне рішення з безліччю варіантів конфігурації. Зазвичай його не рекомендують використовувати для невеликих додатків, а для великих проектів цей фремворк підходить найкраще.

Серед функцій Zend Framework 2:

- інструменти криптографічного кодування;
- простий у використанні drag-and-drop-редактор із підтримкою фронтенд-технологій (HTML, CSS, JavaScript);
- миттєве онлайн-налагодження;
- інструменти для unit-тестування;
- майстер конфігурації бази даних.

Творці цього фреймворку врахували методологію Agile, що дозволяє створювати високоякісні програми для корпоративних клієнтів.

YII 2

Yii обирають, щоб підвищити продуктивність сайту. Швидший за інші PHP-фреймворки, оскільки використовує технологію завантаження на вимогу (lazy loading). Yii 2 повністю об'єктно-орієнтований і заснований на принципі Don't-Repeat-Yourself, тому основа для коду буде чистою і логічною.

Yii 2 інтегрований з jQuery та поставляється з набором AJAX-функцій. Механізми скінінгу та вибору тем тут прості, тому фреймворк сподобається тим, хто раніше займався фронтенд-розробкою. Тут також є потужний генератор вихідного коду Gii, що сприяє об'єктно-орієнтованому програмуванню та швидкому прототипуванню, а також надає web-інтерфейс, в якому можна інтерактивно генерувати потрібний код.

PHPIXIE

PHPixie – порівняно новий фреймворк, створений для сайтів-візиток. Як і FuelPHP, PHPixie підтримує схему HMVC. Він побудований на незалежних компонентах, які можна використовувати навіть без фреймворку. Модулі компонентів PHPixie повністю протестовані та вимагають мінімум інших компонентів для своєї роботи.

Функції PHPixie:

- робота з БД на рівні об'єктів (ORM);
- кешування;
- валідація введення;
- автентифікація та можливості для авторизації.

SYMFONY 2

Компоненти фреймворку Symfony 2 використовуються в багатьох чудових проектах, наприклад Drupal 8 CMS, phpBB і Laravel. У Symfony велика спільнота розробників і величезна кількість відданих шанувальників. Компоненти Symfony – це PHP-бібліотеки, які можна використовувати повторно та виконувати за їх допомогою безліч завдань. Завдання Symfony 2:

- створення форм;
- конфігурація об'єктів,

- маршрутизація;
- автентифікація;
- створення шаблонів.

Будь-який компонент можна встановити за допомогою Composer-менеджера пакетів PHP. А в розділі Showcase на сайті вказані проекти, виконані за допомогою цього фреймворку.

Laravel

Laravel – безкоштовний web-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC. Laravel випущено під ліцензією MIT. Вихідний код проекту розміщується на GitHub.

«Фреймворк корпоративного рівня» та «Фреймворк для особистих проектів». Ключові особливості, що лежать в основі архітектури Laravel:

- пакети – дозволяють створювати та підключати модулі у форматі Composer до програми на Laravel. Багато додаткових можливостей вже доступні у вигляді таких модулів;
- Eloquent ORM – реалізація шаблону проектування ActiveRecord на PHP. Дозволяє чітко визначити відносини між об'єктами бази даних. Стандартний для Laravel виробник запитів Fluent підтримується ядром Eloquent;
- логіка програми – частина програми, що розробляється, оголошена або за допомогою контролерів, або маршрутів. Синтаксис оголошень схожий на синтаксис, що використовується у каркасі Sinatra;
- зворотна маршрутизація – пов'язує між собою посилання, що генеруються додатком, і маршрути, дозволяючи змінювати останні з автоматичним оновленням пов'язаних посилань. При створенні посилань за допомогою іменованих маршрутів Laravel автоматично генерує кінцеві URL-адреси;
- REST-контролери – додатковий шар для поділу логіки обробки GET- та POST-запитів HTTP;
- автозавантаження класів – механізм автоматичного завантаження

класів PHP без необхідності підключати файли їх визначень у include. Завантаження на вимогу запобігає завантаженню непотрібних компонентів; завантажуються лише ті, які дійсно використовуються;

- міграції – система управління версіями баз даних. Дозволяє пов'язувати зміни в коді програми зі змінами, які потрібно внести до структури БД, що спрощує розгортання та оновлення програми;
- модульне тестування (unit-тести) - відіграє дуже велику роль в Laravel, який сам по собі містить велику кількість тестів для запобігання регресії (помилок внаслідок оновлення коду або виправлення інших помилок).

Розробка проекту та тестування

Процес написання проекту можна розбити на такі етапи:

- написання коду;
- тестування коду;
- складання проекту та розгортання коду.

Процес написання коду

Розробка проекту можлива у будь-якій операційній системі, у тому числі й на Windows. Для цього необхідне IDE (інтегроване середовище розробки (від англ. Integrated Development Environment)), наприклад, PhpStorm. Можна використовувати текстовий редактор Atom або Sublime Text. Звичайно, можна писати код і в звичайному блокноті, наприклад Notepad++, але IDE використовувати доцільніше.

Крім IDE, при написанні коду необхідно встановити Composer (Composer – це пакетний менеджер рівня додатків для мови програмування PHP, який надає засоби управління залежностями в PHP-додатку). Саме через Composer встановлюється (оновлюється) Laravel, додаються додаткові пакети до web-проекту. Composer – це дуже важливий та корисний інструмент.

Після підготовчої роботи йде сам процес написання коду з використанням можливостей IDE та фреймворку.

Тестування коду

Для тестування web-проекту немає потреби завантажувати файли на

FTP-сервер, встановлювати локальний Apache (той самий Denwer чи XAMPP). Це неправильно і не позбавляє помилок у коді. На сьогоднішній день із цим завданням справляються відповідні інструменти, які заощадять багато часу.

Так, Laravel пропонує встановити Homestead.

Homestead – це образ операційної системи Ubuntu, де вже встановлено все необхідне для роботи.

Для встановлення образу необхідні Vagrant та VirtualBox. Завдяки цьому образу розробник точно знає, які модулі треба встановити і як поведеться код на Ubuntu.

Якщо коротко, то в системі з'являться спільні папки з кодом, які будуть доступні всередині образу Ubuntu, і код буде записаний саме всередині Ubuntu.

У браузері вводиться site.app і браузер відображає сайт з Ubuntu. При цьому також можливий доступ до Ubuntu SSH.

У програмістів-початківців установка і налаштування Homestead займає чимало часу, але ці дії обов'язкові. Варто відзначити, що Homestead можна встановити не лише на Linux, а й на Windows. Вважатимемо, що Homestead встановлений, і сайт зі свіжою версією Laravel відкривається в браузері.

Laravel пропонує нам інструменти для повноцінного тестування web-проекту з усіх боків. Причому тестування різноманітні: можна створити тимчасову базу даних, перевірити заповнення HTML-форм, перевірити завантаження файлів, навіть зміст PHP-сесій та надсилання повідомлень.

Laravel створений для якісного тестування всіх можливостей проекту. У Laravel тести знаходяться в папці tests і виконуються командою phpunit в консолі або одночасно з IDE. Тести бувають кількох типів: функціональні – Feature-тести; модульні – Unit-тести.

Feature-тести - функціональні тести

Тести, які перевіряють функціонал web-проекту, наприклад, реєстрацію користувачів, відправлення повідомлень, заповнення web-форм, завантаження файлів. Дозволяють нам перевірити, які саме дані відображаються у браузері: так, не потрібно заповнювати web-форми вручну, щоб дізнатися, чи вони

працюють.

Також можна проводити тестування за допомогою Laravel Dusk, не просто надсилаючи HTTP-запити, а використовуючи справжній движок браузера Chromium. У цьому допомагає Laravel Dusk.

Unit-Тести - модульні тести

Ці типи тестів перевіряють логіку програми, кожну функцію, відловлюють події, визначають, чи надіслано повідомлення, а також звіряють текст повідомлення, перевіряють, чи додано завдання в чергу повідомлень та різні помилки, якщо невдало змінено код. Кожна функція проекту повинна мати свої тести, а коли завершено проект, всі тести повинні успішно запускатися. За зміни функціоналу можна дописати тести. Це гарантія від помилок та допомога при діагностиці проблеми. Unit-тестування дозволяє уникнути помилок у логіці програми.

Існує методика розробки TDD (test-driven development) – розробка через тестування. Спочатку пишуться тести, потім поступово реалізується код. Коли всі тести виконані, можна сказати, що завершено написання коду.

Крім тестів, є ще інші помічники для аналізу продуктивності web-додатку.

Laravel пропонує використовувати Laravel Debugbar. За допомогою нього можна відстежити всі SQL-запити до бази даних з метою подальшої оптимізації.

Складання проекту

Після створення web-проекту та проходження тестів необхідно підготувати проект до розміщення на сервері. Laravel надає нам Laravel Mix. Laravel Mix використовує Webpack та вміє працювати з CSS, JS, Less, Saas, Stylus, PostCSS. Це чудовий інструмент, який, використовуючи спеціальний збирач модулів Webpack, збирає разом усі JS- та CSS-файли, а також, найголовніше, вміє створювати версії цих файлів.

Таким чином, кожна редакція нашого проекту дозволяє мати різні назви JS- та CSS-файлів у HTML-коді, що вирішує проблему з кешуванням при зміні

вмісту файлу.

У шаблоні нашого проекту пишемо

```
<link rel="stylesheet" href="{{ mix('/css/app.css') }}">
```

Після збирання він перетворюється:

```
<link href="/css/app.289df32d2d2c47df3b16.css" rel="stylesheet">
```

При цьому браузер відвідувача відразу завантажить новий файл із сайту.

Варто відзначити, що Laravel чудово працює з прогресивним Javascript-фреймворком Vue і дозволяє дуже зручно створювати web-додатки на базі JS-фреймворку. При цьому кожен компонент можна зручно розміщувати в окремому файлі.

Розгортання коду

Зазвичай після складання проекту його файли необхідно завантажити на сервер та оновити структуру таблиць у базі даних. Не будемо використовувати FTP і php MyAdmin, інакше, поки вносяться зміни, всі користувачі, які зайдуть на сайт web-проекту, побачать безліч помилок про відсутність файлів або полів у БД. Є дуже просте та грамотне рішення, яке дозволяє оновлювати код web-проекту без його відключення, і жоден користувач при цьому не отримає повідомлення про помилку.

Перше, що нам необхідно вивчити – Git. Git – це розподілена система керування версіями файлів. За допомогою Git можна відстежувати зміни у файлах, повертати стару версію файлів і працювати в команді над одним і тим самим кодом, при цьому нічого не переплутавши. Можна створити або загальнодоступний код або приватний (для приватних репозиторіїв він платний). Також можна використовувати інший сервіс – BitBucket, який дозволяє безкоштовно створювати приватні репозиторії із кодом. Крім цього, сам Git можна налаштувати так, щоб при внесенні змін відбувалися певні дії:

- запуск тестів проекту через Travis CI;
- форматування коду за стандартом;
- аналіз якості коду через інструмент

Таким чином, весь код web-проекту зберігатиметься в Git, причому

завжди буде якісний та перевірений.

Наприклад, якщо необхідно внести зміни до офіційного коду PHP-фреймворку Laravel, то при внесенні змін автоматично запускаються тести, які перевіряють роботу фреймворку з огляду на новий код.

Раніше йшлося про процес розгортання web-додатку. Саме для цього нам і потрібний Git. З локальної машини завантажується код web-додатку до Git, після чого відбудеться автоматичний запуск розгортання програми на сервері.

Laravel Forge – надійний сервер. Для автоматичного розгортання з Git надає допомогу сервіс Laravel Forge. Через Laravel Forge можна створити віртуальний сервер у DigitalOcean, Linode або вказати доступ до власного сервера. При цьому буде налаштовано абсолютно все необхідне програмне забезпечення для роботи PHP-фреймворку Laravel. Laravel Forge автоматично встановлює оновлення, пов'язані із безпекою системи. Також Forge легко встановить безкоштовний SSL сертифікат від Let's Encrypt.

Можна дати сервісу Laravel Forge доступ до Git-репозиторію, і при кожній зміні коду на сервері буде автоматично розгорнута його свіжа версія.

У разі потреби використання, наприклад десяти серверів, Laravel Forge може встановити балансувальник навантаження, створити десять віртуальних серверів, на кожен сервер копіювати код з Git і запустити проект.

Практична частина

1. Вивчити теоретичну частину.
2. Додати у програму, розроблену у лабораторній роботі №6 функціонал, який виконує визначення найближчого об'єкта, здатного забезпечити надання сервісу, необхідного програмі, відповідно до логіки її роботи (для програми «книги» найближчої книгарні, для програми «розклад потягів» найближчого залізничного вокзалу). Дані про доступні об'єкти підвантажувати з файлу. Запустити програму на емуляторі.
3. Оформити звіт з лабораторної роботи.

Контрольні питання

1. Назвіть основні особливості мови програмування PHP.
2. Назвіть переваги PHP-фреймворків.
3. Охарактеризуйте фреймворк Laravel.
4. Назвіть основні етапи розробки та тестування проекту.
5. Опишіть використання Internet комунікацій у пристроях Android (HTTP).
6. Опишіть використання Internet комунікацій у пристроях Android (FTP).
7. Опишіть використання Internet комунікацій у пристроях Android (e-mail).
8. Опишіть використання Internet комунікацій у пристроях Android (сокети).
9. Формати даних (html, xml, json).
10. Опишіть відстеження розташування пристрою (GeoLocation).

СПИСОК ЛІТЕРАТУРИ

1. Friesen, J. Java XML and JSON: Document Processing for Java SE. Second Edition. — Berkeley (CA): Apress, 2019. — 546 p. DOI: 10.1007/978-1-4842-4330-5.

2. Mikkonen T. Programming mobile devices: an introduction for practitioners. - London: John Wiley & Sons Ltd., 2007. - 245 p.

3. Paavilainen J. Mobile business strategies - understanding the technologies and opportunities. - London: IT Press, 2002. - 257 p.

4. Lee V., Schneider H., Schell R. Mobile Applications: architecture, design, and development. - Prentice Hall, 2004. - 368 p.

5. Fling B. Mobile design and development: practical concepts and techniques for creating mobile sites and web apps. - O'Reilly Media, 2009. -336 p.

6. Verbraeck A. Designing mobile service systems. - Amsterdam: IOS Press, 2007. - 249 p.

7. Zheng P., Lionel N. Smart Phone and next-generation mobile computing. - Morgan Kaufmann, 2005. - 350 p.

8. Friesen J. Learn Java for Android development. - Apress, 2010. - 656 p.

9. Jackson W. Android apps for absolute beginners. - Apress, 2011. - 344 p.

10. Burnette E. Hello, Android: introducing Google's mobile development platform. - Pragmatic Bookshelf, 2010. - 300 p.

11. Ableson WF, Sen R., King C. Android in action. - Manning Publications, 2011. - 592 p.

12. Rogers R, Lombardo J, Mednieks Z, Blake Meike G. Android application development: programming with the Google SDK. - O'Reilly Media, 2009. - 336 p.

13. Murphy ML Android programming tutorials. - CommonsWare, 201L- 334 p.

14. Meier R. Professional Android 2 application development. - Wrox, 2010. -576 p.

15. Sayed Y. Hashimi. Pro Android 2. - Apress, 2010. - 500 p.

16. Conder S., Darcey L. Android wireless application development. - Addison-Wesley Professional, 2009. - 600 p.

17. To N., Steele J. The Android developer's cookbook: building applications with the Android SDK (Developer's Library). - Addison-Wesley Professional, 2010. - 400 p.

18. Di Marzio J.F. Android: a programmer's guide. - McGraw-Hill Osborne Media, 2008. - 400 p.

19. Komatineni S., MacLean D., Hashimi S. Pro Android 3. - Apress, 2011. - 1200 p.

20. Корнієнко Б.Я., Галата Л.П. Дослідження імітаційного полігону захисту критичних інформаційних ресурсів методом IRISK // Моделювання та інформаційні технології. 2018. Вип. 83. С. 34-42.

21. Korniyenko B., Yudin A., Galata L. Risk estimation of information system. // *Wschodnioeuropejskie Czasopismo Naukowe*. 2016. № 5. P. 35-40.

22. Корнієнко Б.Я., Юдін О.К., Снігур О.С. Безпека аутентифікації у web-ресурсах // *Захист інформації*. 2012. № 1 (54). С. 20-25. DOI: 10.18372/2410- 7840.14.2056 (ukr).

23. Корнієнко Б.Я., Максімов Ю.О., Марутовська Н.М. Прикладні програми управління інформаційними ризиками // *Захист інформації*. 2012. № 4 (57). С. 60 – 64. DOI: 10.18372/2410-7840.14.3493 (ukr).

24. Корнієнко Б.Я. Безпека інформаційно-комунікаційних систем та мереж // Навчальний посібник для студентів спеціальності 125 «Кібербезпека». – К.: НАУ, 2018. 226 с.

25. Корнієнко Б.Я., Галата Л.П. Оптимізація системи захисту інформації корпоративної мережі // *Математичне та комп'ютерне моделювання. Серія: Технічні науки*. 2019. Випуск 19. С. 56-62.

26. Корнієнко Б. Я. Дослідження моделі взаємодії відкритих систем з погляду інформаційної безпеки // Наукоємні технології. 2012. № 3 (15). С. 83–89, doi.org/10.18372/2310- 5461.15.5120 (ukr).

27. Корнієнко Б. Я., Галата Л. П. Побудова та тестування імітаційного полігону захисту критичних інформаційних ресурсів // Наукоємні технології. 2017. № 4 (36). С. 316–322. doi.org/10.18372/2310-5461.36.12229.

28. Корнієнко Б.Я. Інформаційні технології оптимального управління виробництвом мінеральних добрив: монографія. – К.: Вид-во Аграр Медіа Груп. 2014. – 288 с.

29. Galata, L., Korniyenko, B., Yudin, A.: Research of the simulation polygon for the protection of critical information resources. In: CEUR Workshop Proceedings, Information Technologies and Security, Selected Papers of the XVII International Scientific and Practical Conference on Information Technologies and Security (ITS 2017), 30 Nov 2017, Kyiv, Ukraine. vol. 2067. pp. 23–31., urn:nbn:de:0074-2067-8.

30. Korniyenko B., Galata L. Implementation of the information resources protection based on the CentOS operating system. Conference Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON -2019) July 2 – 6, 2019, Lviv, Ukraine. - pp. 1007-1011.

31. Галата Л.П., Корнієнко Б.Я., Заболотний В.В. Математична модель протидії загрозам у системі захисту критичних інформаційних ресурсів. Наукоємні технології, Том 43, № 3, 2019. – С. 300 – 306.

32. Корнієнко Б.Я. Modeling of information security system in computer network. Безпека інформаційних систем і технологій, Том №1 (1), 2019. – С.36-41.

33. Korniyenko B., Galata L., Ladieva L. Research of Information Protection System of Corporate Network Based on GNS3. Conference Proceedings of 2019 IEEE Inernational Conference on Advanced Trends in Information Theory (IEEE ATIT -2019) Dezember 18 – 20, 2019, Kyiv, Ukraine. - pp. 244-248.

34. Korniyenko B., Galata L., Ladieva L. Mathematical model of threats resistance in the critical information resources protection system. CEUR Workshop Proceedings, Selected Papers of the XIX International Scientific and Practical Conference "Information Technologies and Security" (ITS 2019) Kyiv, Ukraine, November 28, 2019. Vol-2577. P.281-291.

35. Korniyenko B.Y., Galata L.P. Design and research of mathematical model for information security system in computer network. Науковий журнал «Наукоємні технології». – 2017, № 2 (34), С. 114 - 118.

36. Korniyenko B., Galata L., Kozuberda O. Modeling of security and risk assessment in information and communication system. Sciences of Europe. – 2016. – V. 2. – No 2 (2). – P. 61 -63.

37. Korniyenko B. The classification of information technologies and control systems. International scientific journal. – 2016. –№ 2. – P. 78 - 81.

38. Korniyenko B., Galata L., Ladieva L. Security Estimation of the Simulation Polygon for the Protection of Critical Information Resources / B. Korniyenko, //CEUR Workshop Proceedings, Selected Papers of the XVIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2018) Kyiv, Ukraine, November 27, 2018, Vol-2318, - P. 176-187, urn:nbn:de:0074-2318-4

39. Kornienko Y.M., Liubeka A.M., Sachok R.V., Korniyenko B.Y. Modeling of heat exchangement in fluidized bed with mechanical liquid distribution // ARPN Journal of Engineering and Applied Sciences. 2019. №14(12). P. 2203-2210.

40. Korniyenko B., Yudin O., Novizkij E. Open systems interconnection model investigation from the viewpoint of information security // The Advanced Science Journal. 2013. №8. P. 53-56.

41. Zhulynskyi A.A., Ladieva L.R., Korniyenko B.Y. Parametric identification of the process of contact membrane distillation // ARPN Journal of Engineering and Applied Sciences Volume 14. 2019. №17. P. 3108-3112.

42. Korniyenko, B., Ladieva, L., Galata, L. Control system for the production of mineral fertilizers in a granulator with a fluidized bed. ATIT 2020 - Proceedings: 2020 2nd IEEE International Conference on Advanced Trends in Information Theory, 2020, pp. 307–310.

43. Kornienko Y.M., Haidai S.S., Sachok R.V., Liubeka A.M., Korniyenko B.Y. Increasing of the heat and mass transfer processes efficiency with the application of non-uniform fluidization // ARPN Journal of Engineering and Applied Sciences. 2020. No 15(7). P. 890-900.

44. Korniyenko B., Kornienko Y., Haidai S., Liubeka A., Huliienko S. Conditions of Non-uniform Fluidization in an Autooscillating Mode // Advances in Computer Science for Engineering and Manufacturing. ISEM 2021 Lecture Notes in Networks and Systems. 2022. No 463. P. 14-27.

45. Korniyenko B., Kornienko Y., Haidai S., Liubeka A. The Heat Exchange in the Process of Granulation with Non-uniform Fluidization //Advances in Computer Science for Engineering and Manufacturing. ISEM 2021 Lecture Notes in Networks and Systems. 2022. No 463. P. 28-37

46. Korniyenko B.Y. The two phase model of formation of mineral fertilizers in the fluidized–bed granulator // The Advanced Science Journal. 2013. №4. P. 41-44.

47. Korniyenko B. Y. Modeling of transport processes in disperse systems. // The Advanced Science Journal. 2013. No. 1. P. 7–10.

48. Korniyenko B. Y. Research modes of a fluidized bed granulator. // The Advanced Science Journal. 2013. No. 5. P. 12–15.

49. Korniyenko, B., Kornienko, Y., Haidai, S., Liubeka, A. Innovative Process for Granulation of Heterogeneous Liquid Systems in a Fluidized Bed. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024. 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_44

50. Ладієва Л.Р., Корнієнко Б.Я., Пелипенко Н.С. Математичне моделювання процесу виробництва етанолу гідролізом деревини. Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, Том 36 (75), № 1, 2025. – С. 134-140. DOI https://doi.org/10.32782/2663-5941/2025.1.2/20

51. Ладієва Л. Р., Корнієнко Б. Я., Сазонов А. Ю., Пелипенко Н. С. Оптимальне керування процесом виробництва етанолу гідролізом деревини. Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, Т. 36(75), № 2, Ч. 2, 2025. - С. 104-111. DOI: 10.32782/2663-5941/2025.2.2/14

52. Ладієва Л.Р., Корнієнко Б.Я., Пилипон О.В. Оптимальне керування процесом паро-кисневої конверсії метану. Прикладні питання математичного моделювання. Том 7, № 1, (2024), - С. 164-174.

53. Корнієнко Б.Я., Ладієва Л.Р., Ульяницька К.О., Галата Л.П., Нестерук А.О. Захист критичних ресурсів веб-застосунку з оренди нерухомості. Інформаційні технології та суспільство. 2024, Випуск 4 (15), – С. 80-87.

54. Нестерук, А., Корнієнко, Б. Системи управління процесами зневоднення та гранулювання у псевдозрідженому шарі. *Інформаційні технології та суспільство*, 2023, 1 (7), - С. 50-58. https://doi.org/10.32689/maup.it.2023.1.7

55. Нестерук, А., Корнієнко, Б. Математична модель процесу виробництва мінеральних добрив у грануляторі з псевдозрідженим шаром. *Інформаційні технології та суспільство*, 2023, 2 (8), -С. 51-61. https://doi.org/10.32689/maup.it.2023.2.6

56. Корнієнко Б.Я., Нестерук А.О. Система управління виробництвом гранульованих мінеральних добрив у псевдозрідженому шарі. Вчені записки Таврійського національного університету імені В.І. Вернадського, Серія: Технічні науки, 2023, Том 34(73), № 5, - С. 133-139. DOI https://doi.org/10.32782/2663-5941/2023.5/22

57. Корнієнко Б.Я., Нестерук А.О. Система управління виробництвом мінеральних добрив у грануляторі із псевдозрідженим шаром, Новітні технології, 2022, №1(13), - С. 47-65.

58. Корнієнко Б.Я., Нестерук А.О. Математичне моделювання процесу гранулювання у псевдозрідженому шарі, Інформаційні технології та суспільство, 2022, №3(5), - С. 20-28.

59. Корнієнко Б.Я., Нестерук А.О. Математичне моделювання процесу гранулювання у псевдозрідженому шарі (огляд моделей), Вісник НТУУ “КПІ імені Ігоря Сікорського”. Серія: Хімічна інженерія, екологія та ресурсозбереження, 2022, №2(21), - С. 51-59.

60. Корнієнко Б.Я., Ладієва Л.Р., Осипа Р.А., Семенцов В.К. Оптимальне керування процесом гранулювання в псевдозрідженому шарі. Новітні технології, № 1(11), 2021. - С. 17-32.

61. Корнієнко Б.Я., Ладієва Л.Р., Галата Л.П. Оптимальна система управління виробництвом мінеральних добрив у грануляторі із пседозрідженим шаром. Новітні технології, № 2(12), 2021. - С. 18-28.

62. Сучасні мобільні операційні системи: Лабораторний практикум [Електронний ресурс]: навчальний посібник для здобувачів ступеня магістра за спеціальністю 126 Інформаційні системи та технології / КПІ ім. Ігоря Сікорського; уклад. Б.Я. Корнієнко. – Електронні текстові данні (1 файл: 1300 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 73 с. https://ela.kpi.ua/handle/123456789/55779

63. Управління проєктами: Лабораторний практикум [Електронний ресурс]: навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 126 Інформаційні системи та технології / КПІ ім. Ігоря Сікорського; уклад. Б.Я. Корнієнко. – Електронні текстові данні (1 файл: 1200 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 68 с. https://ela.kpi.ua/handle/123456789/55780

64. Компоненти програмної інженерії: Архітектура програмного забезпечення: Лабораторний практикум [Електронний ресурс]: навчальний

посібник для здобувачів ступеня бакалавра за спеціальністю 121 Інженерія програмного забезпечення/ КПІ ім. Ігоря Сікорського; уклад. Б.Я. Корнієнко. – Електронні текстові данні (1 файл: 1790 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 88 с. https://ela.kpi.ua/handle/123456789/55781

65. Korniyenko B., Ladieva L., Kornienko Y., Galata L., Nesteruk A., Savina O. Control System for Mineral Fertilizer Production in a Fluidized Bed Granulator with a Fuzzy Controller. Studies in Systems, Decision and Control, 2025, 622, pp. 115 - 128. DOI: 10.1007/978-3-032-02770-2_7

66. Korniyenko B., Ladieva L., Nesteruk A. Chaos Control System of the Granulation Process of Mineral Fertilizers in a Fluidized Bed. Studies in Systems, Decision and Control, 2025, 622, pp. 103 - 113. DOI: 10.1007/978-3-032-02770-2_6

67. Korniyenko, B.Y., Ladieva, L.R., Pisarenko, V.G., Pisarenko, J.V., Nesteruk, A.O. Control Systems for the Granulation of Mineral Fertilizers in a Fluidized Bed. Cybernetics and Systems Analysis, 2024, 60(5), pp. 726–735. DOI: 10.1007/s10559-024-00710-6

68. Korniyenko, B., Nesteruk, A. Mathematical Model of the Process of Production of Mineral Fertilizers in a Fluidized Bed Granulator. Lecture Notes on Data Engineering and Communications Technologies, 2023, 180, pp. 55-64. DOI: 10.1007/978-3-031-36115-9_6

69. Korniyenko, B., Ladieva, L., Nesteruk, A., Berezianko, K. Control System for the Production of Granular Mineral Fertilizers in a Fluidized Bed. Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS*, 2023, pp. 38–41. DOI: 10.1109/IDAACS58523.2023.10348752*

70. Korniyenko, B., Kornienko, Y., Haidai, S., Liubeka, A. Hydrodynamics of Inhomogeneous Jet-Pulsating Fluidization. Lecture Notes on Data Engineering and Communications TechnologiesVolume 181, Pages 560 – 573. 2023. DOI http://doi.org/10.1007/978-3-031-36118-0_50

71. Korniyenko, B., Ladieva, L., Bereza, O. Optimal Control with Prediction for the Process of Vacuum Membrane Distillation. In: Hu, Z., Zhang, Q., Petoukhov, S., He, M. (eds) Advances in Artificial Systems for Logistics Engineering. ICAILE 2022. Lecture Notes on Data Engineering and Communications Technologiesthis link is disabled, 2022, vol. 135, pp. 37–50. Springer, Cham. ISSN 23674512. https://doi.org/10.1007/978-3-031-04809-8_4