

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

для індивідуальної роботи з дисципліни
“Розробка застосувань для мобільних пристроїв”

122

для студентів спеціальності
Комп'ютерні науки та інформаційні технології
”

2016

Методичні вказівки для індивідуальної роботи з дисципліни “Розробка застосувань для мобільних пристроїв ” для студентів спеціальності “Комп’ютерні науки та інформаційні технології”. / Укл.: О.О. Степаненко – Запоріжжя: ЗНТУ, 2016. – 20 с.

Автори: Степаненко О.О., к.т.зв., доцент;

Рецензент: Онищенко В.Ф., к. ф.-м. н., доцент

Відповідальний за випуск: Субботін С.О., д.т.н., проф.

Затверджено
вченою радою інституту
інформатики та
радіоелектроніки

Протокол № 1
від “19” березня 2014
р.

Затверджено
на засіданні кафедри
“Програмні засоби”

Протокол № 7
від “06” березня 2014 р.

ЗМІСТ

1. МОБІЛЬНИЙ ДОДАТОК FEEDER	4
1.1 Опис роботи програми	4
1.2 Структура програми	4
1.3 Авторизація користувача	5
1.4. Завантаження стрічки новин.....	5
1.5. Пошук інформації в соціальних мережах	6
1.6. Кешування зображень	6
1.7. Концепція користувацького інтерфейсу програми	7
2. ПРОГРАМНІ ЗАСОБИ ДЛЯ ДОДАТКУ FEEDER.....	8
2.1. Вибір технологій	8
2.2. Розробка мобільного додатку Android	15
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	20

1. МОБІЛЬНИЙ ДОДАТОК FEEDER

1.1. Опис роботи програми

Feeder – мобільний додаток, що є клієнтом для декількох соціальних мереж.

Ціль проекту Feeder – надати можливість власникам смартфонів, які користуються такими соціальними мережами як Facebook, Twitter, ВКонтакте в одному мобільному додатку мати доступ до всіх своїх акаунтів.

В даній програмі реалізований функціонал, раніше доступний лише частково в деяких мобільних проектах, а саме агрегація акаунтів вище названих соціальних мереж та пошук необхідної інформації серед всіх користувачів.

1.2. Структура програми

Програми виступає в ролі клієнта в моделі взаємодій «клієнт-сервер».

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних за стосунків і передбачає взаємодію та обмін даними між ними.

В даному разі є декілька серверів, які є незалежними один від одного. За допомогою прикладного програмного інтерфейсу (англ. Application Programming Interface, API) додаток взаємодіє з серверами соціальних мереж і отримує необхідні дані.

Компоненти програми:

- API соціальних мереж
- Модуль авторизації користувача у соціальних мережах, який зберігає стан підключених акаунтів
- Модуль, що виконує запити до відповідних серверів і отримує відповіді у форматі JSON або XML
- Модуль, який обробляє результати запитів і представляє їх у зручному вигляді
- Модуль кешування зображень

- Інтерфейс користувача

1.3. Авторизація користувача

Авторизація – надання певній особі або групі осіб прав на виконання певних дій; а також процес перевірки (підтвердження) даних при спробі виконання цих дій.

Авторизацію не слід плутати з аутентифікацією: аутентифікація – це лише процедура перевірки достовірності даних, наприклад, перевірки відповідності введеного користувачем пароля до облікового запису паролю в базі даних, або перевірка цифрового підпису по ключу шифрування, або перевірка контрольної суми файлу на відповідність заявленої автором цього файлу.

Алгоритм авторизації користувача:

1. Початок
2. Вибір мережі зі списку доступних
3. Відкриття вікна браузера для аутентифікації користувача на сайті відповідної мережі
4. Ввід логіна та пароля
5. Якщо введені дані правильні, то п.6, інакше – п.9
6. Дозвіл користувачем доступу до своїх даних
7. Передача в додаток ключа `access_token` для доступу API
8. Завантаження стрічки новин
9. Кінець

1.4. Завантаження стрічки новин

Для завантаження необхідних даних з серверів соціальних мереж потрібно використовувати їх API.

Інтерфейс програмування додатків (іноді інтерфейс прикладного програмування) (англ. Application Programming Interface, API) – набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання у зовнішніх програмних

продуктах. Алгоритм завантаження стрічки новин (доступно, коли користувач авторизований хоча б в одній соціальній мережі).

Алгоритм завантаження стрічки новин (доступно при хоч одному авторизованому акаунті):

1. Початок
2. Визначення доступних мереж
3. Для кожної мережі виконати запит на отримання певної кількості постів зі своєї стрічки новин
4. Отримати відповідь у форматі JSON або XML та обробити її
5. Видати на екран список
6. Кінець

1.5. Пошук інформації в соціальних мережах

Алгоритм пошуку інформації в соціальних мережах (доступно для неавторизованих користувачів):

1. Початок
2. Ввести шуканий текст або хештег в поле пошуку
3. Для кожної мережі виконати запит на отримання певної кількості постів в яких зустрічається даний текст
4. Отримати відповідь у форматі JSON або XML та обробити її
5. Видати на екран списки
6. Кінець

1.6. Кешування зображень

LRU Cache (Least Recently Used Cache) — це алгоритм кешування, збереження отриманих даних. Суть алгоритму полягає в наступному: у кеша є фіксований розмір N і в ньому зберігаються дані тільки для N найпопулярніших запитів, тобто ті дані, які, найвірогідніше, будуть запитані знову. При цьому, як тільки число кешованих даних перевищує задане N , алгоритм витісняє з кешу дані, які найдовше не запрошувалися (рис. 2.3.). Крім LRU Cache існує безліч інших алгоритмів кешування – це і алгоритм

Беладі, і MRU, і Pseud-LRU. Основна перевага алгоритму LRU Cache в тому, що він простий в реалізації і залишає в кеші останні запитувані дані, що в нашому випадку є найбільш підходящим.

У мові Java реалізація даного алгоритму не складе труднощів, адже для неї знадобляться всього дві структури даних:

1. Хеш-таблиця, яка буде безпосередньо зберігати кешовані дані
2. Черга з пріоритетами, підтримуюча додавання в чергу значення з пріоритетом і витяг (видалення) з неї

У нашому випадку дані, а саме картинки, зберігаються на флеш пам'яті мобільного пристрою, і додаються або видаляються в міру необхідності.

Тепер зображення з недавно переглянутих користувачем сторінок зберігаються у нього в пам'яті пристрою і не вимагають завантаження з інтернету, що економить його час та гроші.

1.7. Концепція користувацького інтерфейсу програми

Головною екранною формою програми є MainList. Вона завантажується при запуску і з нею проходять основні операції (рис. 2.3.). На верхній панелі знаходиться поле пошуку та кнопка налаштувань.

В залежності від того, які соціальні мережі доступні при завантаженні стрічки власних новин чи при отриманні результатів пошуку, для кожної мережі створюється окремий список постів. Він розміщується в окремій вкладці з назвою відповідної мережі. Переключатися між списками можна натискаючи на саму вкладку або за допомогою жестів вправо-вліво.

У вікні налаштувань (рис. 2.4.) можна підключити свої акаунти соціальних мереж.

2. ПРОГРАМНІ ЗАСОБИ ДЛЯ ДОДАТКУ FEEDER

2.1. Вибір технологій

Оскільки середовищем роботи клієнтської програми для соціальних мереж було обрано операційну систему Android, розглянемо існуючі технології розробки мобільних додатків під дану платформу.

2.1.1. Native додатки

Розробка додатків для платформи Android ведеться переважно мовою Java. Для створення програм мовою Java необхідно спеціальне програмне забезпечення. Найостанніші версії цього ПО можна завантажити з офіційного сайту розробника, Oracle Corporation. До цього програмному комплексу відносяться такі інструменти як: JRE (Java Runtime Environment) і JDK (Java Development Kit). Перший інструмент являє собою середовище виконання – мінімальну реалізацію віртуальної машини, в якій запускається і виконується програмний код на Java. Другий інструмент – це в свою чергу цілий набір інструментів, комплект розробника додатків мовою Java. Насправді, JRE також входить до складу JDK, так само як і різні стандартні бібліотеки класів Java, компілятор Javac, документація, приклади коду і різноманітні службові утиліти. Весь цей набір поширюється вільно і має версії для різних ОС, тому будь-хто може його завантажити і використовувати. У JDK не входить інтегроване середовище розробки, передбачається, що її розробник буде встановлювати окремо. Існують численні IDE для Java -розробки, наприклад, Eclipse, NetBeans, IntelliJ IDEA, Borland JBuilder та інші. Таким чином, перш ніж приступити до розробки програми на базі ОС Android, необхідно підготувати інструментарій.

На сайті компанії для розробників <http://developer.android.com/> можна знайти і завантажити SDK для своєї платформи.

До основних переваг native-розробки можна віднести:

- Зручність розробки та налагодження. Хоча сам процес і зручність інструментів відрізняється від платформи до платформи, в цілому розробляти native-додатки набагато зручніше, ніж, наприклад,

гібридні або кросплатформені. Пов'язано це з відсутністю додаткових прошарків між кодом програми і виконуваним файлом, і, в цілому, більш розвиненими інструментами.

- Документація і технічна підтримка
- Швидкість роботи. Тільки native-додаток може забезпечити максимальну швидкість роботи і продуктивність
- Зручність користування. Кожна платформа має свої архітектурні та дизайнерські рішення, знайоме користувачам розташування керуючих елементів. Native-додатки, як ніякі інші, надають можливість реалізувати інтерфейс і загальну поведінку програми найбільш природним для даної платформи способами.

До недоліків можна віднести:

- Охоплення платформ
- Наявність фахівців. Дійсно, знайти хорошого розробника на Java досить складно зважаючи специфіку даної області і більш високого порога при вивченні технології.

2.1.2. Веб-додатки

Якщо програма не вимагає роботи з ресурсами пристрою або роботи в офлайн режимі, то можна реалізувати його у вигляді веб-сервісу.

Користувачеві потрібно лише знати адресу в інтернеті, щоб звернутися до додатка через браузер.

Такий варіант добре підходить додаткам, які в будь-якому випадку вимагають постійну наявність інтернет підключення, і при цьому не пред'являють особливих вимог до графіки і використанню апаратних засобів пристрою.

Мобільний веб-додаток розробляється за допомогою класичних інструментів. HTML застосовується для розмітки елементів інтерфейсу. CSS описує візуальну складову і взаємне розміщення віджетів. Мова програмування JavaScript реалізує логіку програми.

Ці технології розвивалися тривалий час і на даний момент є достатньо зрілими для того, щоб використовуватися для розробки складних програм, вони адаптуються під різні пристрої додатків. Для розробника веб-додатків зараз є величезний вибір інструментів, фреймворків, які сильно прискорюють процес розробки. При створенні нового додатка не доводиться щоразу вирішувати типові завдання. Замість цього інструменти дозволяють сконцентруватися на вирішенні конкретної бізнес-проблеми.

Потрібно враховувати, що розробка під мобільні пристрої має свою специфіку. Зокрема, обмежені ресурси оперативної пам'яті і процесора мобільного пристрою, в порівнянні з настільними комп'ютерами, накладають великий відбиток на багато аспектів розробки і, відповідно, на специфіку знань розробника. Тому, якщо програміст успішно розробляв рішення для настільних комп'ютерів, це ще не означає, що він відразу зможе створювати ефективні мобільні додатки, використовуючи той же стек веб-технологій. Жоден готовий фреймворк не захищає від несподіваних падінь продуктивності через невірне управління пам'яттю.

Переваги веб-додатків:

- Охоплення платформ
- Час розробки. В цілому, якщо враховувати охоплення відразу декількох платформ, час розробки може бути нижче. Але, оскільки веб-додатки в цілому простіші, ніж native-додатки, ця перевага може бути спірною
- Наявність фахівців
- Документація та підтримка

До недоліків відносяться:

- Швидкість роботи і продуктивність. В цілому продуктивність нижче ніж у native-додатків, оскільки з'являється додатковий прошарок у вигляді веб-браузера

- Налаштування. Процес розробки веб-додатків досить простий. Первісне налаштування може бути зроблена на комп'ютері розробника у звичайному браузері. Налаштування ж на пристроях часто ускладнена і обмежується використанням спливаючих повідомлень (т.зв. *save man debugging* – примітивне налаштування)
- Використання апаратних ресурсів пристрою. Веб-додатки мають доступ тільки лише до невеликого набору сервісів – постійне сховище (*localStorage*), гео-координати. Ні файлова система, фото-камера не доступні.

2.1.3. Гібридні

Ідея використання веб-технологій дійсно дуже приваблива.

Розробивши додатки один раз, потенційно можна поширити його відразу на безліч платформ, так як веб-браузери вбудовані за замовчуванням в усі сучасні мобільні операційні системи. Враховуючи, що фреймворки та інструменти для розробки мобільних HTML5-додатків постійно удосконалюються, то багато програмістів, знайомих з веб-технологіями, можуть почати розробляти мобільні додатки прямо тут і зараз.

Однак, багатьом додаткам хотілося б читати і зберігати файли на SD-карті, робити знімки через фото-камеру, отримувати повідомлення про системні події і т.д. Веб-додатки, запущені в звичайному браузері, такої можливостей не мають.

У зв'язку з цим з'явилися інструменти, що дозволяють розробляти логіку і інтерфейс на HTML і JavaScript, маючи при цьому доступ до ресурсів пристрою. Практично всі SDK мобільних платформ надають спеціальний компонент браузера для впровадження в native-додаток – Web View в Андроїд. Це той же веб-браузер, але з можливістю передати в веб-додаток (в JavaScript-код) виклики з native коду, а також приймати повідомлення з JavaScript назад в native частину програми. Таким чином, веб-додатки,

«огорнуті» в native, можуть отримати доступ до всіх апаратних ресурсів пристрою.

На описаному підході заснована робота Phonegap – популярного інструменту для створення гібридних додатків. Він дозволяє в напівавтоматичному режимі упаковувати веб-додаток в native, роблячи це відразу для декількох платформ.

Phonegap підтримує створення розширень. Будь-яка функціональність може бути додана шляхом розробки плагіна на відповідній native мові. Існує велика кількість вже готових плагінів, тому розробка свого власного, швидше за все, не знадобиться.

За багатьма даними, технологія гібридних додатків стає все більш популярною, особливо в корпоративному секторі. Пов'язано це мабуть, насамперед, з великою популярністю веб-технологій взагалі. Також, гібридні додатки потенційно можуть підтримувати відразу кілька платформ, що може бути вирішальним аргументом на їх користь. Оскільки для корпоративних користувачів чуйність інтерфейсу додатку і загальна продуктивність не стоїть на першому місці, застосування даної технології може бути дійсно оптимальним вибором.

Переваги:

- Підтримка платформ. Зокрема, Phonegap заявляє про підтримку більш ніж семи платформ
- Наявність фахівців
- Час розробки. Правда, іноді процес доведення під конкретну платформу може привести до перегляду архітектури проекту і звести нанівець дану перевагу

Недоліки очевидні:

- Продуктивність. Потрібно постійно шукати компроміси, щоб програма не зупинялась через нестачу пам'яті або щоб його інтерфейс не застигав в процесі обробки великого дерева елементів

- Налаштування. Веб-додаток, упакований в native обгортку важко налагоджувати

2.1.4. Кросплатформенні

Поділ між гібридними і кросплатформенними засобами розробки досить умовний. У даній статті під кросплатформенними розуміються додатки, компільовані в native код або відразу у виконуваний файл. При цьому, існує можливість повторного використання значної частини коду.

Як правило, повторно використовується код, відповідаючий за бізнес-логіку. Інтерфейс користувача може (іноді і повинен) бути розроблений окремо. Кращою практикою може бути залучення різних фахівців для доведення під різні платформи. Кросплатформенні додатки стоять десь між native і гібридними. Від перших вони беруть продуктивність, від других – повторне використання коду.

У даній категорії можна згадати два інструменти.

Перший – Appcelerator Titanium. Код додатка пишеться на JavaScript. Потім він компілюється в native код для платформ (підтримується лише три – IOS, Android і BlackBerry). Далі, можна зібрати цей код в інсталяційний файл програми для кожної мобільної операційної системи. У Titanium є власне середовище розробки Titanium Studio з можливістю налаштування власного коду на JavaScript. Titanium, також як і Phonegap, підтримує розширення на Java або Objective-C для додавання необхідної функціональності.

Другий інструмент – MonoTouch від компанії Xamarin. Це дійсно багатообіцяючий інструмент. В якості мови розробки використовується мова C#, що може стати вирішальним фактором при виборі технології для розробників з досвідом роботи в .NET.

MonoTouch компілює проект на C# відразу в native-додаток. Крім того, Mono підтримує візуальне створення інтерфейсу IOS-додатків за допомогою InterfaceBuilder-a, що дозволяє використовувати файли nib з native-редактора X-Code.

MonoTouch може позиціонуватися не тільки як інструмент кросплатформенної розробки. Але і як просто можливість розробляти повністю native-додатки на мові C#. При цьому з'являється можливість пере використання коду для різних платформ. Наприклад, якщо додаток містить велику кількість бізнес- та інфраструктурної логіки, що не відноситься безпосередньо до інтерфейсу користувача, то, за деякими даними, пере використання коду може досягати 80%.

Переваги кросплатформенних технологій:

- Швидкість роботи
- Час розробки
- Підтримка платформ

Недоліки:

- Наявність фахівців. Дійсно, зважаючи на більшу специфічність, розробників з досвідом розробки в, наприклад, MonoTouch знайти ще складніше, ніж розробників для native-SDK
- Зручність налагодження. У Titanium можна налагоджувати тільки свій код, і не можна отримати доступ до самого фреймворка, що ускладнює пошук проблем. Що стосується MonoTouch, то, мабуть, через достатню зрілість інструменту, у багатьох розробників виникають постіні проблеми при налагодженні та запуску програми

2.1.5. Обрана технологія

Були розглянуті популярні технології, що дозволяють вирішити поставлене завдання. При виборі технології для розробки необхідно орієнтуватися на наступні характеристики:

- Платформа Android
- Зручність розробки та налагодження
- Документація та технічна підтримка
- Швидкість роботи
- Зручність користування

Варто зауважити, що веб- та гібридні технології Phonegap не підходять з причини своєї низької продуктивності.

Оскільки кросплатформенність програми не біло поставлено як основний критерій, то використання технології Appsembler Titanium і MonoTouch буде оптимальним для даної розробки.

Грунтуючись на перерахованих вище критеріях, а також наявність готового API для соціальних мереж Facebook, Twitter, ВКонтакте, мій вибір зупинився на розробці native-додатку, використовуючи мову програмування та Android SDK.

2.2. Розробка мобільного додатку Android

Програмний додаток для ОС Android складається з набору activity, кожній з яких відповідає екран програми. Кожна activity представлена в проекті класом, реалізованим на мові Java, що зберігається в однойменному файлі з розширенням .java. кожній activity відповідає xml файл-опис. У xml-файлі описано, у вигляді xml-коду, розташування візуальних об'єктів. При запуску activity система Android автоматично розпізнає розмір екрану мобільного пристрою і призводить виведений контент у відповідність з розміткою, описаною в xml-файлі. Таким чином, одна і та ж activity буде виглядати однаково незалежно від діагоналі використовуваного пристрою. Також, для кожної гри програми Android повинен існувати файл AndroidManifest.xml, в якому у вигляді xml-коду будуть прописані мінімальні вимоги до системи, а також активність, яка викликається при запуску програми.

Як можна побачити в коді файла AndroidManifest.xml є головна activity – MainActivity та activity для авторизації в мережах VK_LoginActivity та інші.

Елемент <uses-permission> запитує дозвіл, який повинен бути наданий системою для нормального функціонування додатку. Дозволи надаються під час установки програми, а не під час його роботи. Єдиним дозволом є дозвіл на підключення до інтернету.

3.2.1. Розробка частини авторизації в соціальних мережах

Основна задача програми – посилати запити до API соціальних мереж та отримувати відповіді з потрібною інформацією. Для цього було створено для кожної соціальної мережі додаток через який можливий доступ до API.

Для доступу до API ВКонтакте з будь-якого Standalone-додатку передбачено механізм клієнтської авторизації на базі протоколу OAuth 2.0. Як клієнт може виступати будь-який Desktop/мобільний додаток, що має доступ до управління Web-браузером.

OAuth – відкритий протокол авторизації, який дозволяє надати третій стороні обмежений доступ до захищених ресурсів користувача без необхідності передавати їй (третій стороні) логін і пароль. Наприклад, користувач, який хоче надати сервісу соціальної мережі доступ до книги контактів свого поштового акаунта, не повинен повідомляти мережі свій пароль від пошти. Замість цього він проходить авторизацію безпосередньо в поштовому сервісі, який (з дозволу користувача або адміністратора сервісу) надає сервісу соціальної мережі повноваження доступу до адресної книги.

Процес авторизації у додатку складається з 3-х кроків:

1. Відкриття вікна браузера для аутентифікації користувача на сайті
2. Дозвіл користувачем доступу до своїх даних
3. Передача в додаток ключа для доступу до API

При клієнтській авторизації ключ доступу до API видається додатком без необхідності розкриття секретного ключа додатку.

Перехід на нову activity виконується за допомогою класу Intent, в конструктор якого передається activity, яку треба відкрити.

Всі елементи інтерфесу користувача в Android додатку будуються з використанням об'єктів View і ViewGroup. Android надає колекцію View і ViewGroup підкласів, які пропонують загальні елементи управління введення (такі як кнопки та текстові поля) і різні моделі компоновання (наприклад, лінійну або відносну).

Facebook SDK надає клас `LoginButton`, який створює кнопку для авторизації. Клас `LoginButton` зберігає стан сеансу і використовується для входу користувачів в мережу та з неї.

Щоб викликати діалог авторизації Facebook створюється клас `FacebookLoginFragment`, який наслідує клас `Fragment`. В ньому перегружений метод `onCreateView()`

Після отримання ключа доступу до API `access_token` його потрібно зберегти у файловій системі, щоб використовувати його при подальших запусках програми. В Android існує клас `SharedPreferences` (Загальні налаштування), розроблений спеціально для цих цілей. Додаток автоматично створює файл у своїй папці і зберігає прості дані у вигляді «ключ-значення».

Класи `VK_Account`, `FB_Account`, `TW_Account` реалізують інтерфейс `Account`. Метод `save (Context context)` зберігає `access_token` у файлову систему, `restore (Context context)` – дістає з пам'яті.

2.2.2. Основні методи API

Facebook REST API:

Таблиця 3.1.

Web-методи Facebook API

GET <code>graph.facebook.com</code> <code>/search?</code> <code>q={your-query}&</code> <code>[type={object-type}]</code> <code>(#searchtypes)</code>	Повертає список постів, що відповідають вказаному запиту. Параметри: <code>q</code> – пошуковий запит <code>type=post</code> – пошук по публічних постах
GET <code>graph.facebook.com</code> <code>/me/home</code>	Повертає колекцію останніх оновлень зі стрічки новин.

В Android SDK запити посилаються за допомогою класу `Request`.

Twitter REST API:

Таблиця 3.2.

Web-методи Twitter API

GET search/tweets	Повертає список твітів, що заповідають вказаному запиту. q (необхідний параметр) – UTF-8, URL-кодований пошуковий запит.
GET Statuses/home_timeline	Повертає колекцію останніх твітів та ревітів розміщених авторизованим користувачем і користувачами, за якими він слідкує.

ВКонтакте REST API:

Таблиця 3.3.

Web-методи ВКонтакте API

GET newsfeed.search	Повертає список твітів, що відповідають вказаному запиту. q – пошуковий запит, по якому необхідно отримати результати. count – вказує, яке максимальне число записів потрібно повернути, але не більше 100.
GET newsfeed.get	Повертає дані, необхідні для показу списку новин для поточного користувача.

У разі успіху повертається загальна кількість записів і масив об'єктів, кожен з яких містить поля:

- id – локальний ідентифікатор запиту (для конкретного власника)
- owner_id – ідентифікатор власника стіни, на якій розміщений запис
- from_id – ідентифікатор автора запису
- date – час публікації записи у форматі unixtime
- text – текст запису

- `comments` – містить інформацію про кількість коментарів до запису
- `likes` – містить інформацію про кількість людей, яким сподобався даний запис
- `attachments` – знаходиться в записах зі стін і містить масив об'єктів, які прикріплені до поточної новини (фотографія, посилання тощо)

2.2.3. Завантаження постів

Завантаження постів для кожної мережі виконується в окремих потоках. Для цього використовується клас `AsyncTask`. Клас `AsyncTask` пропонує простий і зручний механізм для переміщення трудомістких операцій у фоновий потік. `AsyncTask` створює, синхронізує потоки, а також управляє ними, що дозволяє створювати асинхронні завдання, що складаються з операцій, що виконуються у фоновому режимі, і оновлювати користувацький інтерфейс по їх завершенні.

В методі `doInBackground()` виконується запит до відповідного API, що повертає потрібні дані у форматі JSON. Як тільки дані завантажуться і аналізуються, то можна оновити `ListView` з відповідними оновленими даними.

З цієї структури потрібно отримати автора, дату, текст і URL прикріплених ресурсів. Фрагмент коду для JSON розбору нижче.

Коли кожен з потоків закінчить парсинг даних, результат потрібно вивести на екран. Для цього в Android прийнято користуватися паттерном Адаптер. Для кожної стрічки створюємо клас `ListAdapter`, що наслідує `BaseAdapter` і перегружаємо необхідні методи.

Якщо до поста прикріплені зображення, в макеті динамічно створюються елементи `ImageView` і завантаження відбувається асинхронно.

2.2.4. Інтерфейс користувача

При налаштуванні інтерфейсу користувача було використано стандартні візуальні компоненти Android SDK.

Список використаних джерел

1. Майер Р. Android 2: Программирование для планшетных компьютеров и смартфонов / Ретро Майер ; пер. с англ. – М. : Эксмо, 2011. – 672 с.
2. Труфанов А. Н. Symbian C++. Программирование для мобильных телефонов / Труфанов А. Н. – М. : ИД "Вильямс", 2010. – 464 с. : ил.
3. Дейтел Х.М. и др. Технологии программирования на Java 2. Книга 1: Графика, JavaBeans, интерфейс пользователя – М.: Бином-пресс, 2003
4. Зdziarski Дж. iPhone. Разработка приложений с открытым кодом / Зdziarski Дж. ; пер. с англ. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2009. – 368 с. : ил.
5. Бишоп Д. Java 2 (Серия «Эффективное программирование») – СПб.: Питер, 2002 – 592 с.
6. Хорстманн К.С., Корнелл Г. Библиотека профессионала: Java 2. – М.: Изд. дом «Вильямс», 2004. Т. I: Основы – 848 с.
7. С. Хашими, С. Разработка приложений для Android / С. Хашими, С. Коматинени, Д. Маклин – СПб. : Питер, 2011. – 736 с. : ил.
8. Невдяев Л. М. Мобильная связь 3-го поколения. Серия изданий "Связь и бизнес" / Невдяев Л. М. – М. : МЦНТИ: 000 "Мобильные комму- никации" 2000. – 208 с.