РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки:потоки відео контенту з хостів інтернету.

Мета кваліфікаційної роботи: проектування та розробка мобільного програмного додатку для операційної системи Android, за допомогою якого можна переглядати потокове відео з обраного хосту за допомогою парсингу.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні мобільного додатка, що працює на сучасній Android системі, має зручний графічний інтерфейс та забезпечує за допомогою парсингу пришвидшений та якісний процес збору та пошуку відео контенту.

Актуальність даного програмного продукту визначається великим попитом на мобільні додатки, що спрощують пошук та доступ до необхідної інформації розважального характеру за допомогою сучасних технологій.

Список ключових слів: ВІДЕОКОНТНЕТ, ПАРСИНГ, МОБІЛЬНИЙ ДОДАТОК, ОПЕРАЦІЙНА СИСТЕМА, СТРІМ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: video content streams from Internet hosts.

The purpose of the qualification work: design and development of a mobile software application for the Android operating system, with which you can view streaming video from the selected host using parsing.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, designs and develops the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of a mobile application that runs on a modern Android system, has a user-friendly graphical interface and provides a fast and high-quality process of collecting and searching video content through parsing.

The relevance of this software product is determined by the high demand for mobile applications that simplify the search and access to the necessary information of an entertaining nature with the help of modern technologies.

List of keywords: VIDEO CONTNET, PARSING, MOBILE APPLICATION, OPERATING SYSTEM, STREAM.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

API – Application Programming Interface, Інтерфейс програмування програм;

IDE – Integrated Development Environment, інтегроване середовище розробки.

ЗМІСТ

ВСТУП

Одною із основних потреб людського життя є інформація. Ми з кожним днем пізнаємо щось нове, читаємо, дивимось телебачення, слухаємо радіо, спілкуємося з людьми. Інколи людина, користуючись Інтернетом, втрачає багато свого дорогоцінного часу, щоб знайти конкретну інформацію, адже потрібно переглянути не один сайт, наткнутися на рекламу.

Інтерфейс деяких сайтів є доволі таки складним і знайти там щось потрібне займає багато часу. Виникає потреба в пришвидшені процесу збору інформації по сайтах. В даний час всі процеси, де застосовується синтаксичний аналіз, використовують парсери – програми для проведення візуального або програмно-автоматизованого синтаксичного і лексичного аналізу або розбору будь-якого документа з метою вилучення з нього необхідних даних. Наприклад, автоматизовані перекладачі з однієї мови на іншу, транслятори мов програмування, які формують програмний код на машинно-орієнтовану мову, мова SQL-запитів і тому подібні застосування.

Парсер контенту – це не що інше, як скрипт, здатний сортувати інформацію, виділяючи найважливішу і обробляючи її згідно з алгоритмом, створеному для вирішення того чи іншого завдання.

Розробка мобільних додатків для операційних систем Android та IOS, на сучасному етапі є вкрай затребуваною. У наш час люди завжди носять із собою мобільні пристрої такі як сматрфони, розумний годинник та інші. Проводять багато часу з ними дивлячись відео чи переглядаючи повідомлення у соціальних мережах. Зараз мобільні пристрої не відстають за функціональністю від стаціонарних, а кількість користувачів збільшується з кожним днем.

Популярність операційної системи (ОС) Android пояснюється тим, що на її базі випускається дуже багато пристроїв, причому не тільки смартфони та планшети, а й годинник, велосипеди і навіть автомобілі. ОС Android одночасно унікальна і різноманітна – кожен користувач може дуже тонко налаштувати оригінальну оболонку на своєму пристрої. Для публікації додатку на Android

треба зробити одноразовий внесок у розмірі 25$ та завантажити додаток, і вже через кілька годин його зможуть завантажувати користувачі.

Зараз доля Android пристроїв переважає, їм належить більше 80% ринку. Отже, перспективи очевидні, аудиторія Android користувачів переважає, а розробка під нього є менш затратною.

Метою кваліфікаційної роботи бакалавра є проектування та розробка мобільного програмного додатку для операційної системи Android, за допомогою якого можна переглядати потокове відео з обраного хосту за допомогою парсингу.

Для розробки даного додатку застосовані інструменти розробки кросплатформених програмних додатків – Xamarin мовою C# та API хоста.

Перевірка роботоспроможності додатка виконується на прикладі підключення до хосту http:SeasonVar.ru, що містить в собі велику кількість ліцензованих відеопродуктів.

Розроблений програмний додаток містить функції: відображення списку стрімів (відео), пошук за назвою та вивід інформації по обраному стріму, де є можливість переглянути опис серіалу, обрати бажаний сезон, переклад та серію, після чого обрати плеєр, у якому відтворити стрім.

Створений мобільний додаток, що працює на сучасній Android системі, має зручний графічний інтерфейс та забезпечує за допомогою парсингу пришвидшений та якісний процес збору та пошуку відео контенту.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Парсинг

Парсинг (від англ. Parse) – процес аналізу або розбору певного контенту на складові за допомогою роботів-парсерів (спеціальних програм або скриптів). У SEO цим контентом є html-код сторінок сайтів. Найвідоміші парсери в мережі – це пошукові роботи, які аналізують сторінки, зберігають дані аналізу у себе в базі і потім при пошуку видають релевантні та актуальні документи. Часто парсинг плутають з граббінгом. Це близькі поняття, але все ж мають різні значення. Граббер дозволяє скачувати інформацію з мережі (html-сторінки, rss-стрічки, xml-документи) в свою базу, а парсер дозволяє виявити з цієї купи корисну інформацію і обробити її, залежно від поставлених завдань. У галузі пошукової оптимізації парсинг використовується дуже часто. Всі SEO-інструменти щось парсингують (посилання, ключові слова) і на основі цього надають корисні дані для аналізу.

Парсер це скрипт, призначення якого полягає в автоматичному створенні статей на сайті. Парсер, за заданими параметрами, виконує пошук в мережі Інтернет потрібного контенту та переносить його на вказаний сайт, тому парсер часто називають грабером сайтів. Застосування скриптів-парсерів дозволяє швидко наповнити новий сайт великою кількістю тематичної інформації практично без участі веб-розробника.

В результаті використанні парсерів веб-розробник отримує сайт, вся інформація якого вже існує в мережі. Такий метод отримання контенту не схвалюється пошуковими системами, так як отримані тексти не є унікальними. Ранжування пошуковими системами сайтів, вся інформація яких отримана з допомогою парсерів, завжди буде дуже низькою. За плагіат інформації такий сайт навіть може бути вилученим з результатів пошуку.

У більшість сучасних парсерів є функції автоматичної обробки отриманих текстів перед їх викладенням на сайт. Для цього вони застосовують різні алгоритми автоматизованого рерайтингу статей. Однак, яким би гарним не був алгоритм рерайтингу, він завжди поступається контенту написаному людиною.

Використання парсеру може бути виправданим для підтримки актуальності сайту. Копіювання свіжих новин зі сторонніх ресурсів може призвести до ситуації коли вони будуть першими проіндексовані на вашому сайті, а не на сайті з якого отримані. В результаті чого пошукова система буде вважати оригіналом контенту викрадену статтю.

Крім того, постійна публікація свіжих актуальних новин також позитивно сприймається пошуковими системами.

Застосування скриптів-парсерів може бути виправданим для наповнення ресурсів, для яких просування в пошукових системах не є актуальним. Наприклад, форумів або блогів соціальних спільнот, де користувачі можуть отримати актуальну інформацію агреговану з різних інтернет-сайтів.

1.1.2. Огляд існуючих інтернет ресурсів

SeasonVar (http:SeasonVar.ru) (рис. 1.1) [20]. На сайті зібрані кращі фільми та серіали всіх сезонів різних країн. Зручний графічний інтерфейс забезпечує доступ до обраного контенту. Проте даний сайт не має мобільної версії, а також зареєстрований в РФ, тому доступ до нього може бути заблокованим у зв’язку політичними рішеннями державцями України.

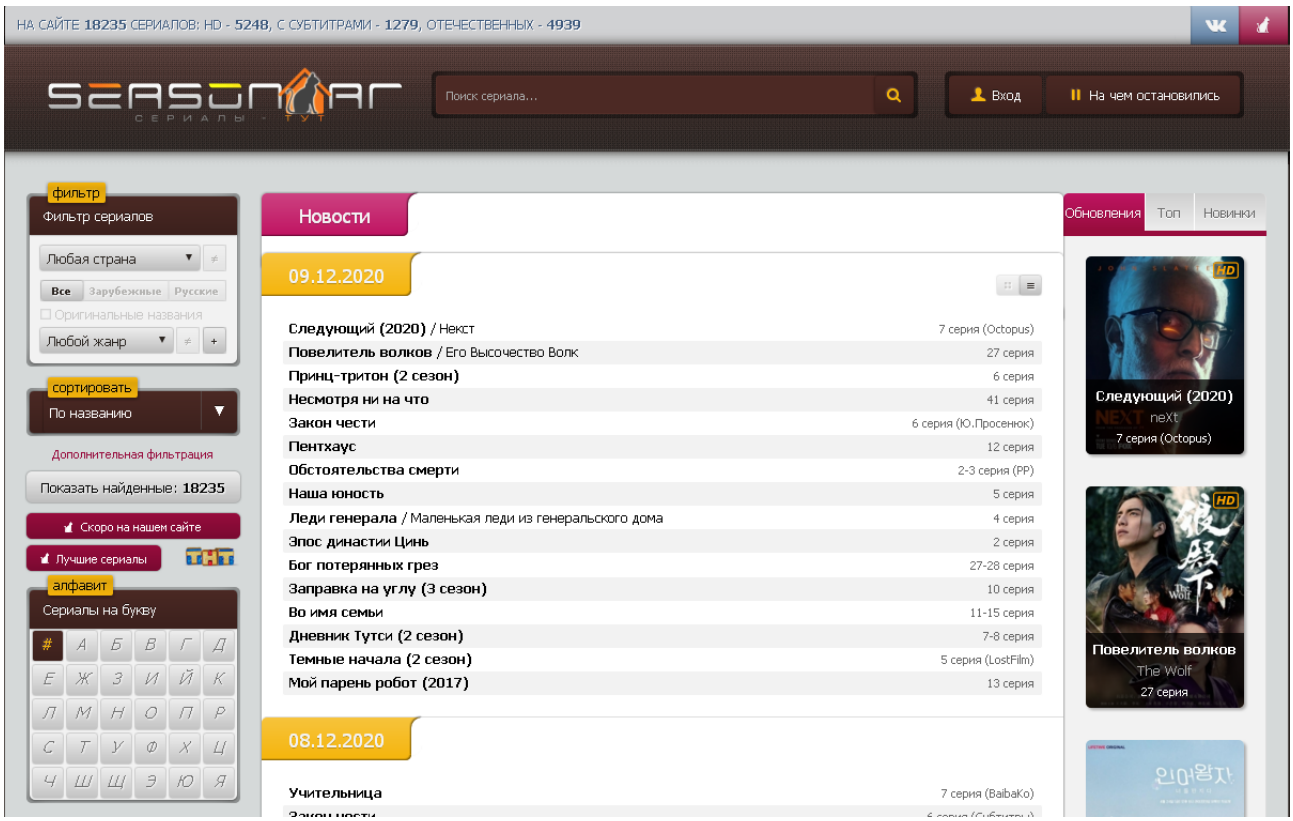

Рис. 1.1. Головне вікно хосту SeasonVar

HD VideoBox (https://hdvideobox.ru/) (рис. 1.2). Унікальний пошуковик - каталог фільмів, серіалів і мультфільмів, з описом, можливість вибрати якість відео, озвучку, переклад і багато іншого.

HD Videobox шукає доступне відео на різних ресурсах, список яких постійно поповнюється, ось список деяких з них: filmix.net, zona.mobi, moonwalk.cc, seasonvar.ru, tivio.net, kinokong.net, hdgo.cc, uafilm, kinosha.net, kino-live.life, 1kinobig.ru, kinokiwi.com. Пошук по торрентів доступний тільки у версії Plus.

Додаток оптимізований для роботи на Андроїд пристроях, медіа програвачах, смарт ТВ, планшетах і телефонах.

Додаток HD VideoBox не доступно в офіційному Google Play, це трохи ускладнює спосіб установки на Андроїд пристрої. Для установки необхідно завантажити інсталяційний apk файл [16].

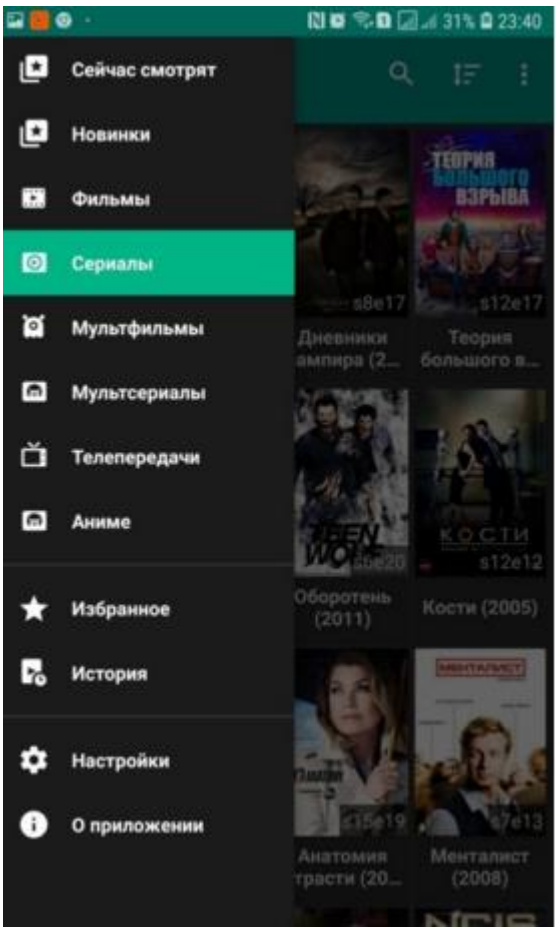

Рис. 1.2. HD VideoBox

1.2. Призначення розробки та область застосування

Метою кваліфікаційної роботи бакалавра є проектування та розробка мобільного програмного додатку для операційної системи Android, за допомогою якого можна переглядати потокове відео з обраного хосту за допомогою парсингу.

Розроблений програмний додаток містить функції: відображення списку стрімів (відео), пошук за назвою та вивід інформації по обраному стріму, де є можливість переглянути опис серіалу, обрати бажаний сезон, переклад та серію, після чого обрати плеєр, у якому відтворити стрім.

Створений мобільний додаток, що працює на сучасній Android системі, має зручний графічний інтерфейс та забезпечує за допомогою парсингу пришвидшений та якісний процес збору та пошуку відео контенту.

Перевірка роботоспроможності додатка виконується на прикладі підключення до хосту http:SeasonVar.ru, що містить в собі велику кількість ліцензованих відеопродуктів.

Даний мобільний додаток може працювати з будь-якими інтернет-ресурсами, що надають користувачу необхідні відеодані розважального контенту (фільми, серіали, мультфільми).

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка мобільного додатку онлайн перегляду відеопотоків на операційній системі Android» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Завданням даної кваліфікаційної роботи бакалавра є спроектувати та розробити мобільний додаток «VideoBox SeasonVar» для ОС Android, метою якого буде пошук та онлайн перегляд відеопотоків методом парсингу з хосту надання відеоконтенту за допомогою власного API.

Під час виконання даної роботи необхідно:

– описати принципи та інструменти взаємодії програмних додатків з іншими за допомогою власного АР-інстерфейсу;

– описати механізми розробки кросплатфомермених додатків – Xamarin, мови програмування - C#, XML, стандарт – JSON, IDE – MS Visual Studio 2019;

– описати характеристику середи функціонування програмного додатку та їх функціональних можливостей;

– сформулювати вимоги до інформаційної та програмної сумісності, складу програми та параметрів технічних систем;

– описати процес проектування та результати її реалізувати.

Для виконання даного завдання необхідно розробити графічний інтерфейс користувача, а саме: екрану форму пошуку серіалу/передачі, меню програми, екрани перегляду списку сезонів та серій серіалу/передачі, запуск обраної серії у програвачі відео потоків.

У якості вхідних даних виступає введена у поле пошуку назва відео, вихідними даними є отримана інформація про обраний відеопоток.

В програмі передбачити функціонали для:

− відправки запиту до API, який буде приймати словник з даними Dictionary та повертати результат у вигляді рядку формату JSON;

− прийому рядка формату JSON і перетворює його у контейнер з вкладеним словником та вивід у список на головній сторінці.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмний додаток призначений для пошуку та перегляду відео потоків на прикладі хосту з надання відеоконтенту http://seasonvar.ru/. У додатку повинно бути передбачено наступні функції:

− пошук за ключовим словом відео потоків та інформації по ним;

− відображення усіх відео на головній сторінці;

− список сезонів та серій до відео потоку.

Також необхідно розробити спливаючу навігаційну панель для обрання категорії відео, перегляд історії переглядів та листу обраних відео.

1.5.2. Вимоги до інформаційної безпеки

Розроблена програма повинна забезпечити дотримання таких вимог до надійності:

— використовувати ліцензійне програмне забезпечення на сервері;

— здійснювати захист від вірусів на сервері;

— здійснювати захист від несанкціонованого доступу;

— застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні

— при псуванні обладнання робота програми повинна бути продовжена при повторному запуску.

1.5.3. Вимоги до складу та параметрів технічних засобів

До технічних засобів висуваються наступні вимоги:

– операційна система: Android 5.0 і вище;

– процесор: MediaTek MT6580A і вище;

– кількість ядер: 2 і вище;

– частота Ггц: 1.3 і вище;

– оперативна пам’ять: 2 ГБ і більше [5].

1.5.4. Вимоги до інформаційної та програмної сумісності

Додаток має бути розроблений на кросплатформовій мові програмування, завдяки чому його можна буде використовувати під різними платформами (Android , Windows, IOS тощо) та мати засоби для доступу до мережі Інтернет.

Програма повинна являти собою самостійний виконуваний модуль, бути структурована і за коментована

Додаток має стабільно працювати на усіх останніх версіях ANDROID.

Для повноцінної роботи програми необхідно встановити MX Player, глобальний офлайновий відеоплеєр і постачальник потокового мовлення Over the Top (OTT), створений J2 Interactive і належить Times Internet, підрозділу цифрових медіа Times Group.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою кваліфікаційної роботи бакалавра є проектування та розробка мобільного програмного додатку для операційної системи Android, за допомогою якого можна переглядати потокове відео з обраного хосту за допомогою парсингу.

Програма призначена для забезпечення наступного функціоналу:

- відправки запиту до API, який буде приймати словник з даними Dictionary та повертати результат у вигляді рядку формату JSON;
- прийому рядка формату JSON і перетворює його у контейнер з вкладеним словником та вивід у список на головній сторінці;
- пошук за ключовим словом відео потоків та інформації по ним;
- відображення усіх відео на головній сторінці;
- збереження списку сезонів та серій до відеопотоку.

В рамках виконання даного завдання необхідно розробити графічний інтерфейс користувача, а саме: екрану форму пошуку серіалу/передачі, меню програми, екрани перегляду списку сезонів та серій серіалу/передачі, запуск обраної серії у програвачі відео потоків.

2.2. Опис застосованих математичних методів

Використання математичних методів під час проектування та розробки мобільного додатку не передбачено.

2.3. Опис використаної архітектури та шаблонів проектування

2.3.1. Application Programming Interface

Application Programming Interface (API) програмний інтерфейс взаємодії між системами, що дозволяє:

- отримувати доступ до бізнес-сервісів підприємства;
- обмінюватися інформацією між системами і додатками;
- спростити взаємодію між компаніями, партнерами, розробниками і клієнтами. Реалізація концепції Open API допомагає трансформувати бізнес, вбудовувати його в гнучку проектну екосистему гравців ринку, створювати умови для постійної генерації нових ідей і формування додаткової цінності при управлінні масивами корпоративних даних [14].

Для знаходження потоків відео у хості SeasonVar передбачений саме Application Programming Interface. Він визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована.

Програмні компоненти хосту взаємодіють один з одним за допомогою API. При цьому зазвичай компоненти утворюють ієрархію – високорівневі компоненти використовують API низькорівневих компонентів, а ті, в свою чергу, використовують API ще більш низькорівневих компонентів.

За таким принципом побудовані протоколи передачі даних по Internet. Стандартний протокол Internet (мережева модель OSI) містить 7 рівнів (від фізичного рівня передачі пакетів біт до рівня протоколів програм, подібних протоколів HTTP і IMAP). Кожен рівень користується функціональністю попереднього рівня передачі даних і, в свою чергу, надає потрібну функціональність наступного рівня.

Важливо зауважити, що поняття протоколу близьке за змістом до поняття API. Протокол та API є абстракцією функціональності. Протокол передає данні, а API призначений для побудови програмних додатків. API бібліотеки функцій і класів включає в себе опис сигнатур і семантики функцій.

Для звернення до API використовують методи (рис. 2.1) [19]:

– GET використовується для отримання (або читання) представлення ресурсу. У разі "вдалого" (або не маючого помилок) адреси, GET повертається представлення ресурсу у форматі XML або JSON в поєднанні з кодом стану HTTP 200 (OK). У разі наявності помилок зазвичай повертається код 404 (NOT FOUND) або 400 (BAD REQUEST);

– POST запит найбільш часто використовується для створення нових ресурсів. На практиці він використовується для створення вкладених ресурсів. Іншими словами, при створенні нового ресурсу, POST запит відправляється до батьківського ресурсу і, таким чином, сервіс бере на себе відповідальність на встановлення зв'язку створюваного ресурсу з батьківським ресурсом, призначення нового ресурсу ID та інші. При успішному створенні ресурсу повертається HTTP код 201, а також в заголовку “Location” передається адреса створеного ресурсу;

– PUT зазвичай використовується для надання можливості поновлення ресурсу. Тіло запиту при відправленні PUT-запиту до існуючого ресурсу URI має містити оновлені дані оригінального ресурсу (повністю, або тільки оновлену частина);

– DELETE запит використовується для видалення ресурсу, ідентифікованого конкретним URI (ID). При успішному видаленні повертається 200 (OK) код HTTP, спільно з тілом відповіді, яке містить дані видаленого ресурсу (негативно позначається на економії трафіку). Також можливе використання HTTP коду 204 (NO CONTENT) без тіла відповіді.

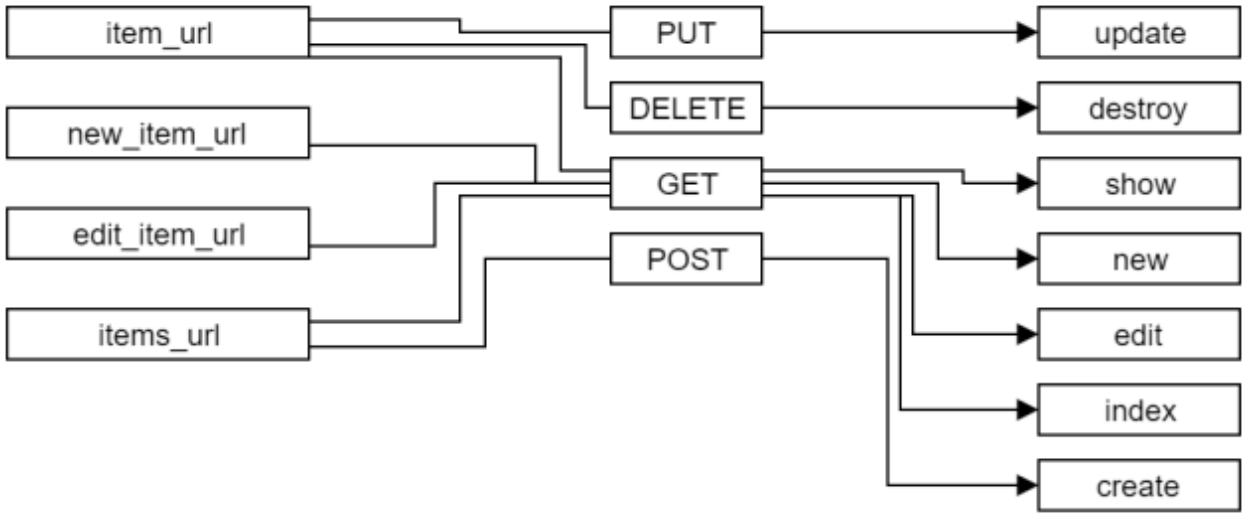

Рис. 2.1. Методи API та їх функції

2.3.2. JSON

Для відправки та отримання результатів запитів зручніше використання JSON, до того ж API написаний на JavaScript.

Для відправки та отримання результатів запитів від API використовуємо JSON – це (нотація об'єктів JavaScript) простий формат обміну даними, зручний для читання і написання як людині, так і комп'ютеру. Він заснований на підмножині мови програмування JavaScript, визначеного в стандарті ECMA-262. JSON – текстовий формат, повністю незалежний від мовних реалізацій, але він використовує угоди, знайомі програмістам мов C, таких як C, C ++, C#, Java, JavaScript, Perl, Python і багатьох інших. Ці властивості роблять його ідеальною мовою обміну даними.

JSON заснований на двох структурах даних:

– колекція пар ключ/значення. У різних мовах, ця концепція реалізована як об'єкт, запис, структура, словник, хеш, іменований список або асоціативний масив (рис. 2.2) [17].

– упорядкований список значень. У більшості мов програмування це реалізовано як масив, вектор, список або послідовність[6].

Рис. 2.2. Структура JSON

2.3.3. XML

Платформа Xamarin використовує XML для запису даних. XML являє собою набір правил для розробки текстових форматів, які дозволять структурувати дані. Це не мова програмування, і не потрібно бути програмістом, щоб використовувати або вивчати його. Він полегшує комп'ютеру задачі створення і читання даних, забезпечуючи при цьому однозначність їх структури. XML дозволяє уникнути поширених помилок проектування мов: він розширюваний, незалежний від платформи, включає підтримку інтернаціоналізації та локалізації. Він повністю сумісний з Unicode. Також як і в HTML, в XML використовуються теги (слова, укладені в '<' і '>') і атрибути (виду ім'я = "значення"). Але якщо в HTML фіксується смислове значення кожного тега і атрибута і часто то, як текст між ними буде виглядати в браузері, в XML теги використовуються тільки для логічної розмітки даних, і їх інтерпретація залишається на розсуд обробної програми[13].

2.4. Опис використаних технологій та мов програмування

2.4.1. Особливості застосування мови програмування C#

Мова програмування C# розроблялась як мова прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо[15].

Мова C# підтримує строго типізовані неявні оголошення змінних з ключовим словом var і неявно типізовані масиви з ключовим словом new[], за яким слідує ініціалізатор колекції. Вирази, які приймають умови, такі як while та if, вимагають висловлювання, що реалізує оператор true або false. C# безпечніший в порівнянні з C++. Єдиними неявними перетвореннями за умовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції і в деяких випадках, під час виконання. Не відбувається неявних перетворень між булевими і цілими числами, а також між членами перерахування і цілими числами (крім літерала 0, який може бути неявно перетворений в будь-який нумерований тип). Будь-яке призначене для користувача перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C++ і операторів перетворення, які за умовчанням є неявними.

C# має явну підтримку коваріації та контраваріантності в родових типах, на відміну від C++, яка має певний рівень підтримки контраваріантності просто

через семантику типів, що повертаються, на віртуальні методи. Члени перерахування розміщуються в своєму власному обсязі. Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції[7].

На відміну від C++, C# має контракти, які надають спосіб вказівки передумов, постумов і інваріантів об'єктів в коді. Передумови – це вимоги, які повинні бути виконані при вході в метод або властивість. Постумови описують очікування під час виходу з коду методу або властивості. Інваріанти об'єктів описують очікуваний стан класу, який знаходиться в робочому стані.

2.4.2. Опис фреймворку Xamarin

Xamarin – це фреймворк для кросплатформеної розробки мобільних додатків (iOS, Android, Windows Phone) з використанням мови C#.

Фреймворк складається з декількох основних частин:

- Xamarin.IOS – бібліотека класів для C#, що надає розробнику доступ до iOS SDK;
- Xamarin.Android – бібліотека класів для C#, що надає розробнику доступ до Android SDK;
- компілятори для iOS і Android;
- IDE Xamarin Studio (рис. 1.3);
- плагін для Visual Studio[9].

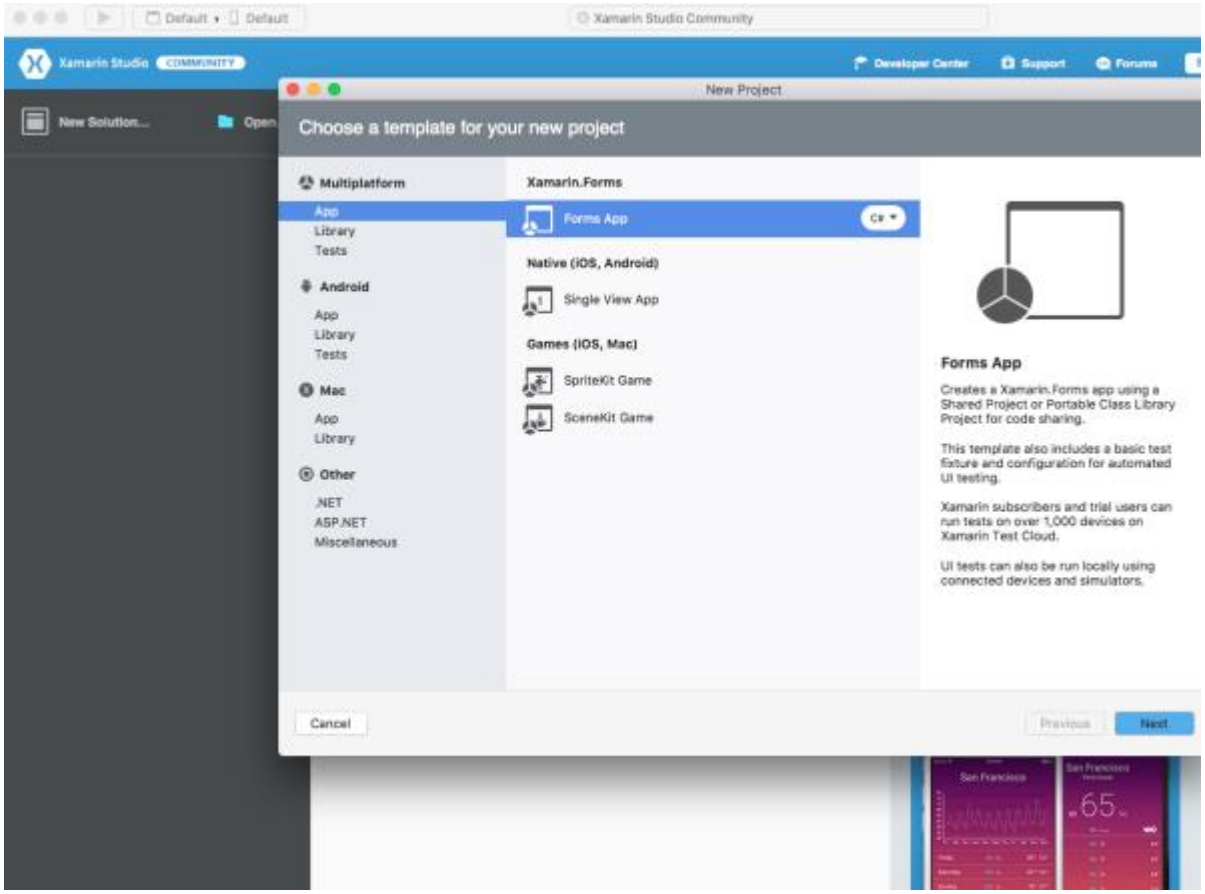

Рис. 2.3. Вікно Xamarin Studio

2.4.3. Середа розробки MS Visual Studio 2019

Visual Studio – це інтегроване середовище розробки (IDE) від компанії Microsoft.

За допомогою Visual Studio можна розробляти:

– класичні додатки для комп'ютера під керуванням операційної системи Windows;

– мобільні додатки (Windows, IOS, Android) (рис. 1.4);

– web-додатки;

– хмарні додатки;

– різні розширення для Office, SharePoint, а також створення власних розширень для Visual Studio;

– ігри;

– бази даних SQL Server і SQL Azure.

У Visual Studio є можливість використовувати наступні технології і мови програмування: .NET, Node.js, C, C#, C ++, Python, Visual Basic, F#, JavaScript[18].

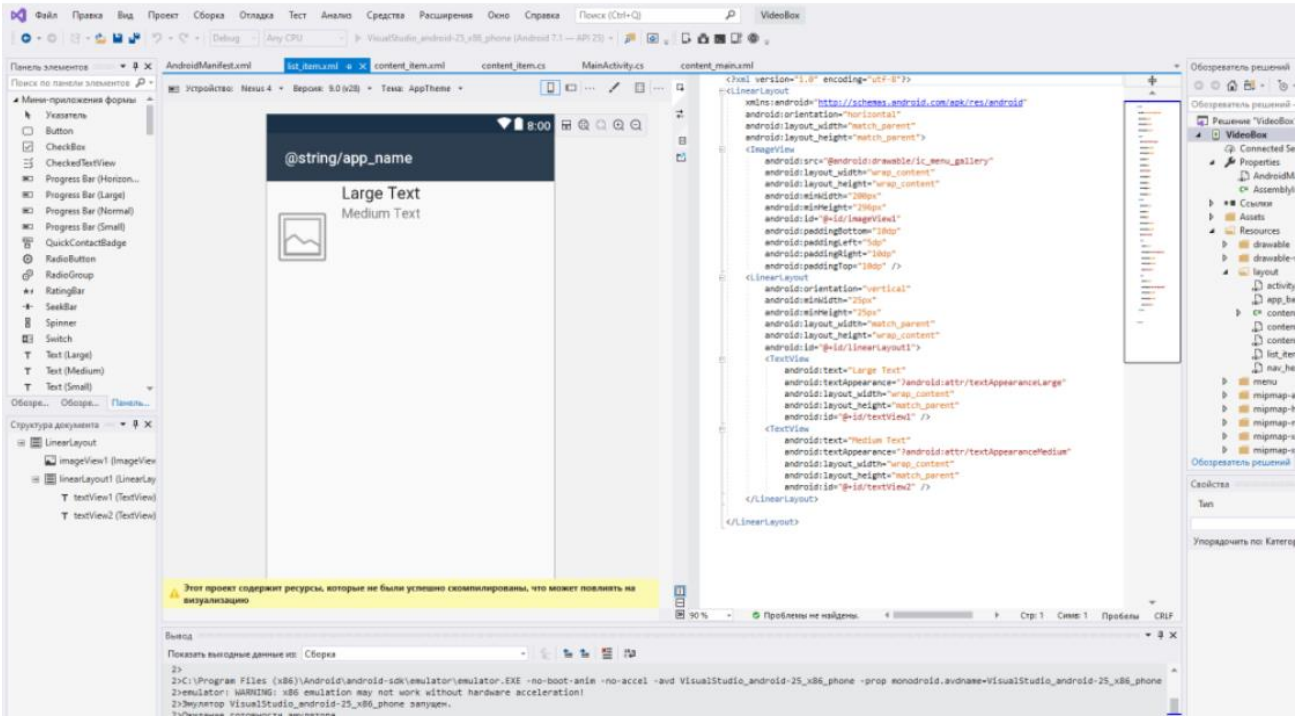

Рис. 2.4. Вікно MS Visual Studio 2019, розробка інтерфейсу мобільного додатку на платформі Xamarin

2.4.4. Характеристики ОС Android

Програмний додаток розроблений під операційну систему Android. Характеристики ОС Android:

− платформа легко пристосовується для використання VGA, бібліотек двовимірної і тривимірної графіки, розроблених на основі OpenGL ES 1.0-3.1 специфікації, традиційних інструментів для смартфонів;

− SQLite для структурованих даних;

− ОС Android підтримує багато технологій, що забезпечують зв'язок, у тому числі: GSM, Bluetooth, EDGE, 3G, 4G, 5G та WiFi;

- для обміну повідомленнями доступні як SMS, так і MMS сервіси, у тому числі й потокові повідомлення;
- на Android доступний веб-браузер, розроблений на основі SCRUM framework;
- програми, написані на Java, можна скомпілювати в Dalvik байткод і виконувати на Dalvik virtual machine, яка являє собою розроблену спеціально для використання на мобільних пристроях віртуальну машину, незважаючи на те, що не є стандартною Java Virtual Machine;
- ОС Android підтримує такі формати для аудіо/відео даних та зображень: MPEG-4, MP3, MP4 та AAC, AMR, JPG, PNG, GIF;
- Android підтримує відеокамери, фотоапарати, сенсорні екрани, GPS, компаси, акселерометри та прискорювачі 3D графіки;
- офіційним середовищем розробки є Android Studio, створене на базі Intellij IDEA, Містить емулятор, засоби відлагодження, профілювання пам’яті та швидкодії. Також доступні плагіни для Intellij IDEA, Eclipse та NetBeans.

Архітектура ОС Android поділяється на такі рівні (рис. 2.5)[1]:

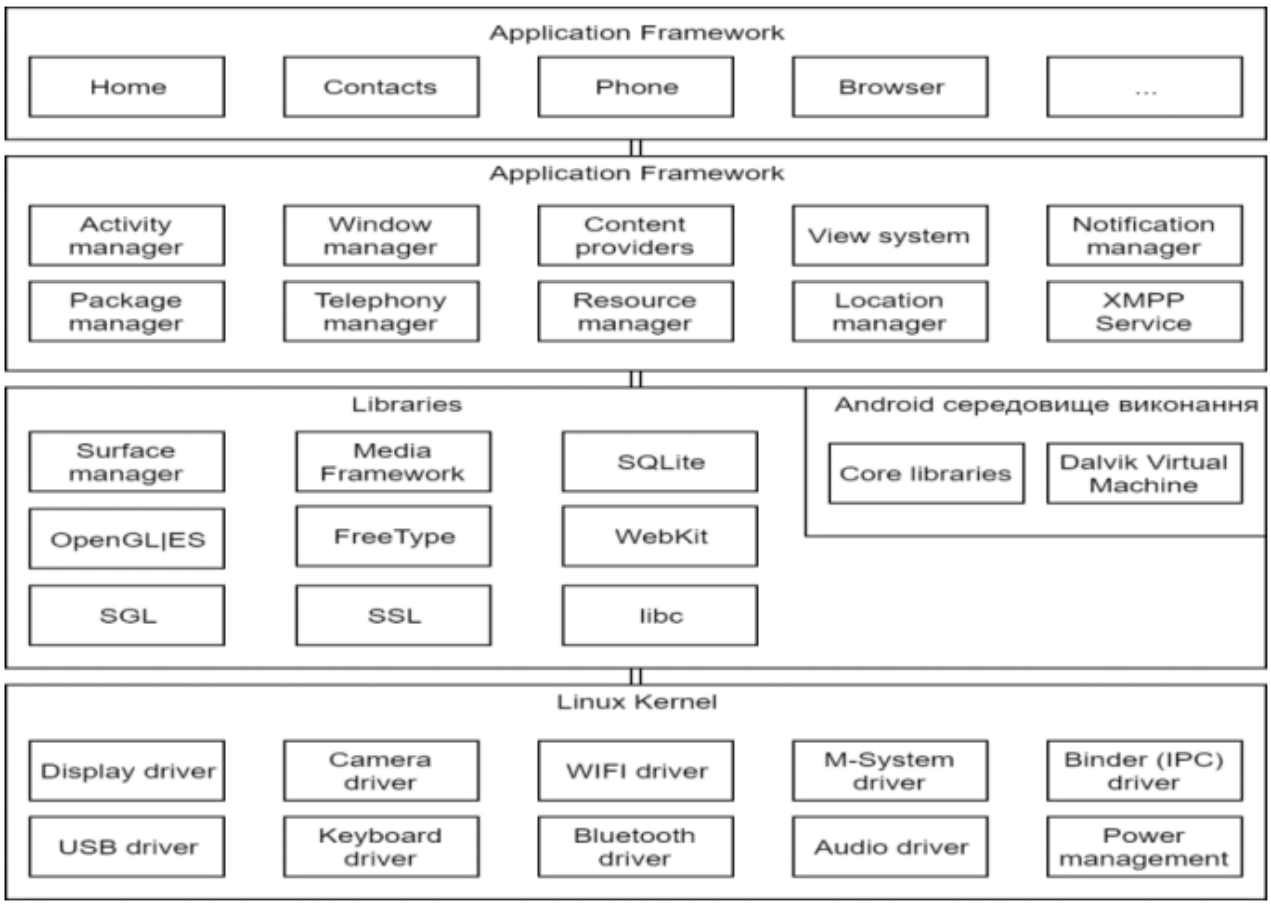

Рис. 2.5. Архітектура ОС Android

– рівень додатків (Applications). До складу Android входить комплект базових додатків: клієнти електронної пошти і SMS, календар, різні карти, браузер, програма для управління контактами і багато іншого. Всі додатки, що запускаються на платформі Android написані на мові Java;

– рівень каркаса додатків (Application Framework). Android дозволяє використовувати всю міць API, використовуваного в додатках ядра. Архітектура побудована таким чином, що будь-який додаток може використовувати вже реалізовані можливості іншої програми за умови, що останнім відкриє доступ на використання своєї функціональності. Таким чином, архітектура реалізує принцип багаторазового використання компонентів ОС і додатків. Основою всіх додатків є набір систем і служб:

1. Система уявлень (View System) – це багатий набір уявлень з розширеною функціональністю, який служить для побудови зовнішнього

вигляду додатків, що включає такі компоненти, як списки, таблиці, поля введення, кнопки і т.п.

2. Контент-провайдери (Content Providers) – це служби, які дозволяють додаткам отримувати доступ до даних інших додатків, а також надавати доступ до своїх даних.

3. Менеджер ресурсів (Resource Manager) призначений для доступу до строкових, графічним і іншим типам ресурсів.

4. Менеджер повідомлень (Notification Manager) дозволяє будь-якому додатком відображати призначені для користувача повідомлення в рядку статусу.

5. Менеджер дій (Activity Manager) управляє життєвим циклом додатків і надає систему навігації по історії роботи з діями.

6. Рівень бібліотек (Libraries). Платформа Android включає набір C/C++ бібліотек, використовуваних різними компонентами ОС. Для розробників доступ до функцій цих бібліотек реалізований через використання Application Framework;

– рівень середовища виконання (Android Runtime). До складу Android входить набір бібліотек ядра, які надають більшу частину функціональності бібліотек ядра мови Java. Платформа використовує оптимізовану, регістр-орієнтовану віртуальну машину Dalvik, на відміну від неї стандартна віртуальна машина Java – стек-орієнтована. Кожна програма запускається в своєму власному процесі, зі своїм власним примірником віртуальної машини. Dalvik використовує формат Dalvik Executable (*.dex), оптимізований для мінімального використання пам'яті додатком. Це забезпечується такими базовими функціями ядра Linux, як організація потокової обробки і низькорівневе управління пам'яттю. Байт-код Java, на якому написані ваші програми, компілюються в dex-формат за допомогою утиліти dx, що входить до складу SDK;

– рівень ядра Linux (Linux Kernel). Android заснований на операційній системі Linux версії 2.6, тим самим платформі доступні системні служби ядра,

такі як управління пам'яттю і процесами, забезпечення безпеки, робота з мережею і драйверами. Також ядро служить шаром абстракції між апаратним та програмним забезпеченням.

Доступні бібліотеки:

- Bionic – бібліотека стандартних функцій, несумісна з libc;
- SSL – шифрування;
- Media Framework (PacketVideo OpenCORE, MPEG4, H.264, MP3, AAC, AMR, JPG, PNG);
- Surface Manager;
- LibWebCore (на базі WebKit);
- SGL – 2D-графіка;
- OpenGL ES – 3D-бібліотека;
- FreeType – шрифти;
- SQLite – легка СУБД.

У порівнянні зі звичайними додатками Linux, додатки Android підкоряються додатковим правилам:

- Content Providers – обмін даними між додатками;
- Resource Manager – доступ до таких ресурсів, як файли XML, PNG, JPEG;
- Notification Manager – доступ до рядка стану;
- Activity Manager – управління активними додатками[1].

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Проектування програми

Для реалізації програмного додатку «VideoBox», призначеного для пошуку та перегляду серіалів для користувачів сматфонів на операційній системі Android, слід виділити наступні об'єкти:

- макет програмного додатку;
- меню і взаємодії з користувачем;

– інтернет підключення.

Програмний додаток повинен забезпечувати:

– відображення списку серіалів;
– фільтрація та сортування списку;
– відображення результатів пошуку у вигляді списку;
– відображення повної інформації по обраному серіалу;
– запис в історію перегляду переглянуті серіали;
– додавання серіалів до списку обраного.

Для визначення послідовності процесів використання та взаємодії: користувача, додатку та системи мобільного пристрою(сматрфону) – під час проектування програмного додатку були розроблені UML діаграми.

Діаграма варіантів використання, необхідна для опису функціональності та поведінки додатку наведена на рис. 2.6:

– користувач запускає додаток «VideoBox»;
– додаток робить запит до сайту SeasonVar та формує список стрімів;
– далі користувач може виконувати подальші дії.

Рис. 2.6. Діаграма варіантів використання (Use Case)

Діаграма послідовності є однією з різновиду діаграм взаємодії та призначення моделювання взаємодій об’єктів системи у часі, а також обміну повідомленнями між ними наведена на рис. 2.7:

- користувач відкриває додаток;
- додаток робить запит до сайту та формує список;
- користувач вибирає бажаний стрім;
- відкривається вікно з усією інформацією по стріму та сам стрім.

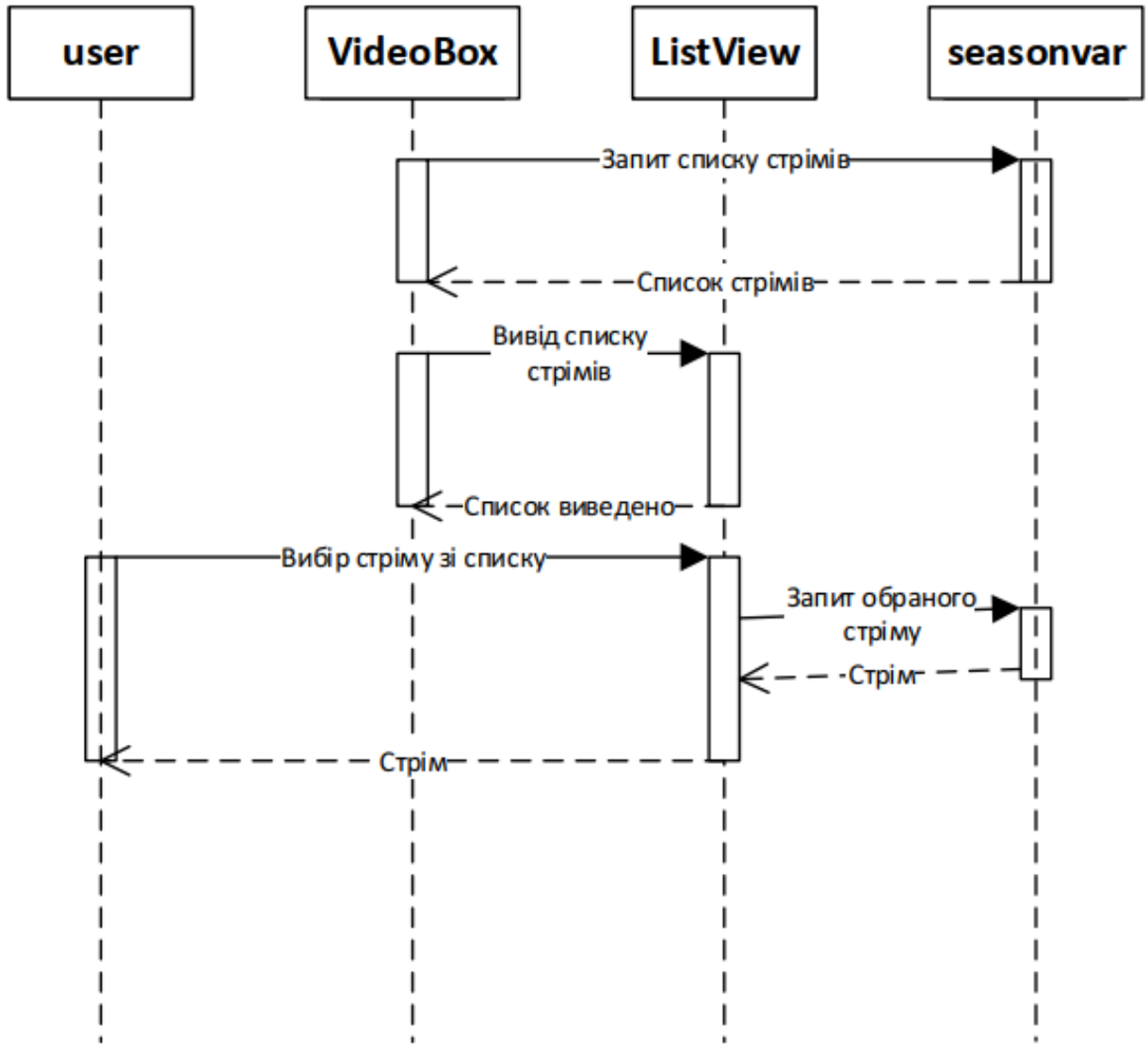

Рис. 27. Діаграма послідовності програмного додатку

Діаграма діяльності детально відображає, як користувачу досягнути певного функціоналу, наведена на рис. 2.8.

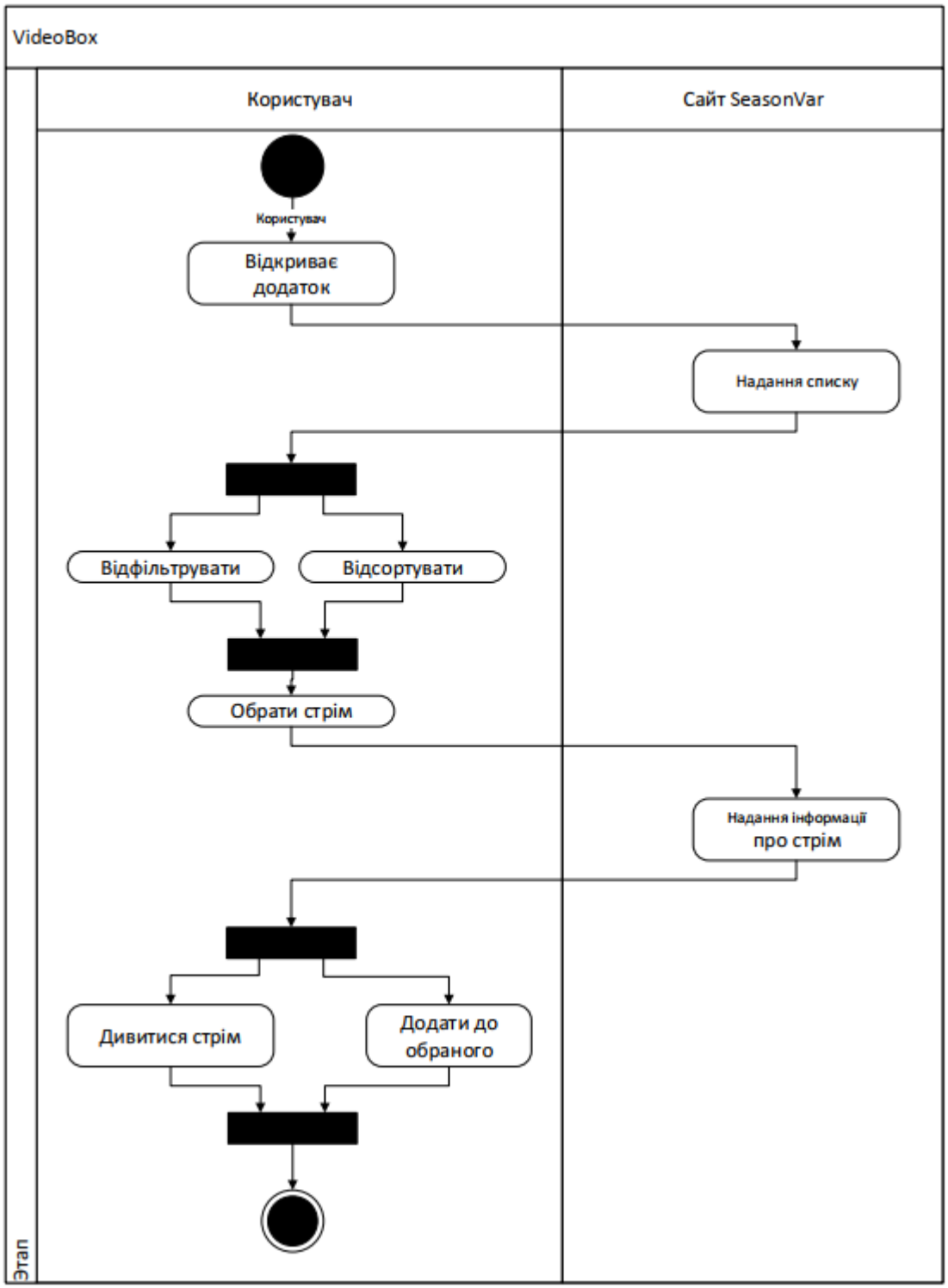

Рис. 2.8. Діаграма діяльності програмного додатку

Діаграма діяльності програмного додатку відображає етапи, які користувачу потрібно виконати, щоб досягнути функціоналу:

– користувач відкриває додаток;

– до сайту надходить запит, якщо запит надійшов то сайт надає інформацію;

– користувачу надано список стрімів, він може його відсортувати або відфільтрувати чи просто обрати вподобаний стрім;

– за обраним стрімом до сайту надходить запит по ньому;

– користувачу виводиться уся інформація про стрім, він може переглянути стрім чи додати його до листу обраного.

Діаграма потоку даних (DFD) представлена у вигляді трьох діаграм потоків:

− контекстна діаграма відображає інтерфейс системи з зовнішнім світом, а саме інформаційні потоки між системою та зовнішніми сутностями з якими вона буде зв’язана (рис. 2.9);

− діаграма декомпозиції призначена для деталізації функцій (рис. 2.10);

− діаграма процесу відображає перетворення вхідних потоків даних у вихідні (рис. 2.11).

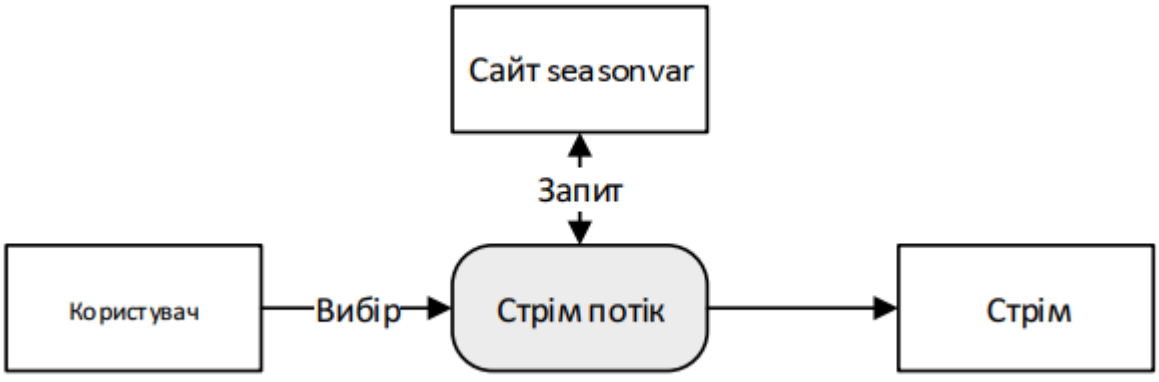

Рис. 2.9. Контекстна діаграма

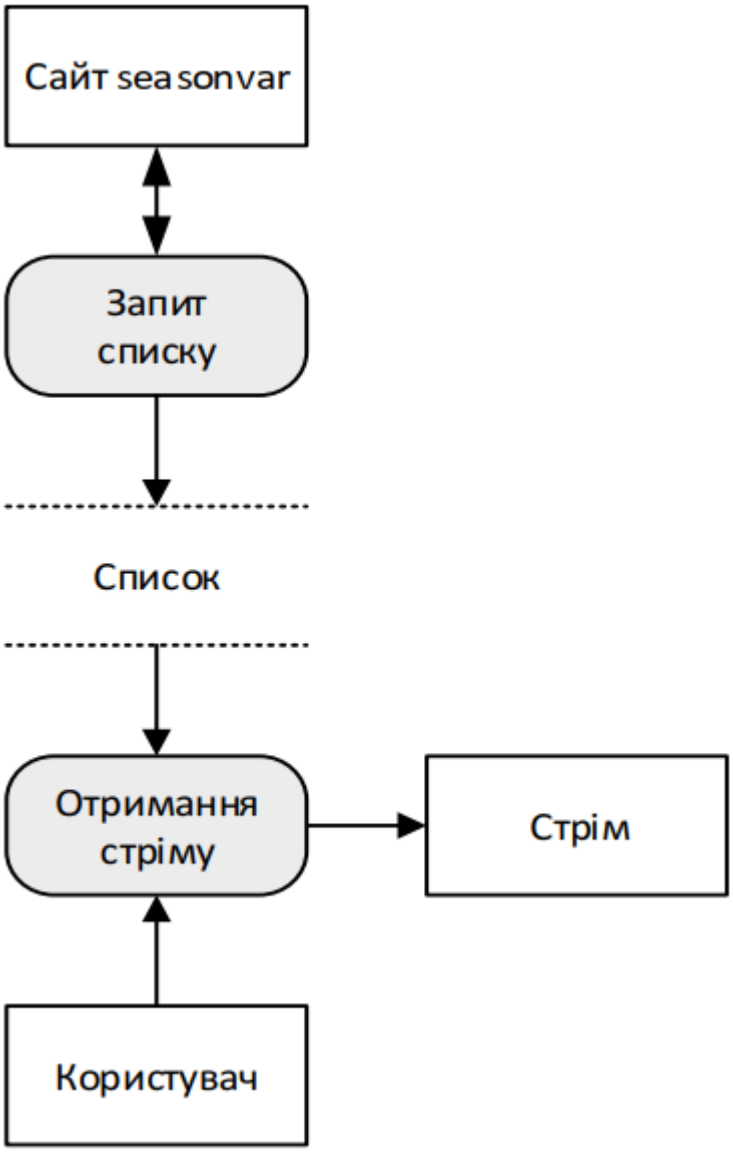

Рис. 2.10. Діаграма декомпозиції

Рис. 2.11. Діаграма процесу

2.5.2. Опис файлової структури додатку

Діаграма компонентів відображає розбиття програмної системи на структурні компоненти та зв'язки між компонентами. В якості фізичних компонентів можуть виступати бібліотеки, класи тощо (рис. 2.12).

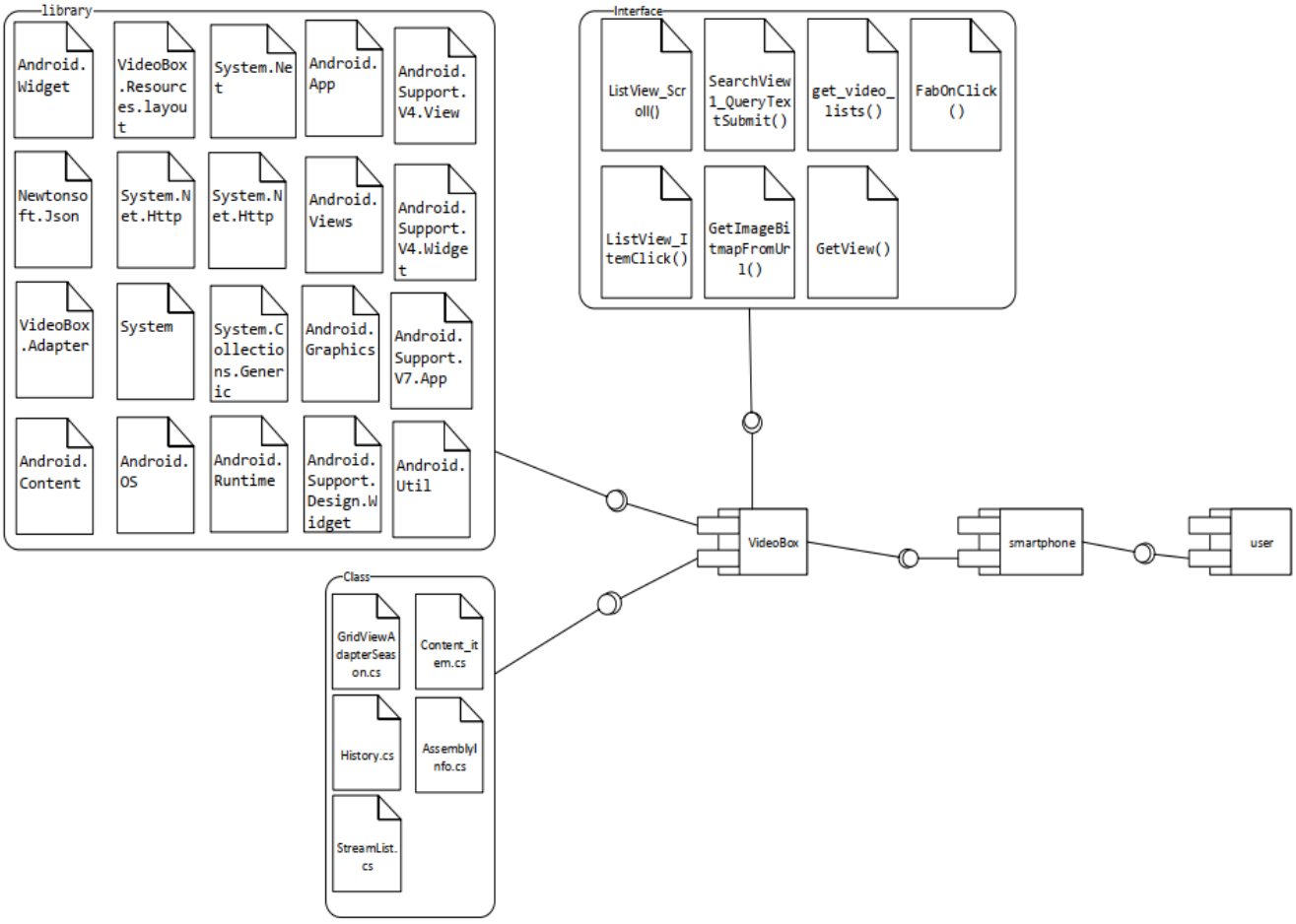

Рис. 2.12. Діаграма компонентів

Для кожної програми на Xamarin android створюються файли:

– MainActivity.cs – файл, з якого починається виконання програми;

– AndroidManifest.xml – файл, який містить усю важливу інформацію, яку потребує ОС Android для того, щоб запустити додаток;

– AssemblyInfo.cs – файл з кодом, який встановлює налаштування проекту;

– Resource.designer.cs – файл, який містить усі ідентифікатори ресурсів;

– Strings.xml – таблиця строк для локалізації додатку.

2.5.3. Програмна реалізація додатку

2.5.3.1. Опис функцій класів

Функції класу MainActivity.cs:

– OnCreate(Bundle savedInstanceState) – при першому створенні активності;

– ListView_Scroll(object sender, AbsListView.ScrollEventArgs e) – подія, яка при досягненні кінця списку підвантажує у нього серіали;

– ListView_ItemClick(object sender,AdapterView.ItemClickEventArgs e) – виникає, коли користувач натисне на елемент у списку;

– GetImageBitMupFromUrl(string url) – функція, яка завантажує зображення з заданого посилання;

– SearchView1_QueryTextSubmit(object sender, SearchView. QueryTextSubmit-EventArgs e) – виникає, коли користувач натискає кнопку пошук;

– Get_Video_List() – функція, яка робить запит до сайту SeasonVar та з отриманих даних формує масив;

– Get_Serial() – функція, яка робить запит за обраним серіалом до сайту та з отриманих даних формує список сезонів;

– Get_Season() – функція, яка робить запит по заданому списку сезонів до сайту та з отриманих даних формує детальні списки по сезонах.

Для заповнення списків були використані перевизначені адаптери GridViewAdapterSeason, які успадковується від BaseAdapter та ExpandableListAdapter від BaseExpandableListAdapter.

GridViewAdapterSeason містить такі методи:

– GetView(int position, View convertView, ViewGroup parent) – перевизначений метод, який створює кнопку з відповідними параметрами та

назначає їй список який відкривається після натискання на неї;

– GetItem(int position) – базовий метод, який повертає позицію елемента у списку;

– GetItemId(int position) – базовий метод, який повертає id елемента.

ExpandableListAdapter містить такі методи:

– GetChild(int groupPosition, int childPosition) – базовий метод, який повертає елемент з групи;

– GetChildId(int groupPosition, int childPosition) – базовий метод, який повертає позицію елемента у групі;

– GetChildView(int groupPosition, int childPosition, bool isLastChild, View convertView, ViewGroup parent) – перевизначений метод, який створює елемент визначеної групи;

– GetChildrenCount(int groupPosition) – базовий метод, що повертає кількість елементів у групі;

– GetGroup(int groupPosition) – базовий метод, що повертає масив елементів з заданої групи;

– GetGroupId(int groupPosition) – базовий метод, що повертає id групи;

– GetGroupView(int groupPosition, bool isExpanded, View convertView, ViewGroup parent) – перевизначений метод, який створює групу;

Для відображення інформації по обраному серіалу визначимо такі змінні:

– List<Dictionary<string, object>> lstDictionary = new List<Dictionary <string, object>>() – призначений для зберігання даних плей-листу по сезону: переклади, серії, трейлери до сезону, опис серій та їх назви;

– Dictionary<string, object> dictionaryL = new Dictionary<string, object>() – призначений для зберігання та окремого виділення плей-листу сезону, тому що дані, які повертає сайт, містять повну інформацію про серіал, яка дублюється до кожного сезону: назва, опис, жанр, запитуваний сезон, рейтинг;

– Dictionary<string, List<string>> listDataChild = new Dictionary <string, List<string>>() – призначений для зберігання серій під їхнім перекладом;

– List<string> listDataHeader = new List<string>() – призначений для зберігання доступного перекладу у поточному сезоні, далі використовується для угрупування серій з поточним перекладом у listDataChild;

– List<string> keys = new List<string>() – використовується для зберігання тимчасових ключів до lstDictionary, тому що звернення до словника відбувається через строкові індекси;

– List<string> links = new List<string>() – використовується для зберігання посилань на стріми серій поточного сезону, далі використовується у listDataChild, де додається під свій переклад.

У методі Get_Serial створюємо словник з ключем доступу, командою та параметрами, де за допомогою HttpClient робимо запит до сайту та отриманий результат десереалізуємо (перетворення з об’єкту JSON у об’єкт мови C#) у лист зі словниками.

Для збереження команди та ключа доступу до сайту http://seasonvar.ru/ оголошуємо словник - dict, який містить запит на список сезонів до обраного серіалу:

Dictionary<string, string> dict = new Dictionary<string, string>(){…}

У визначені словника dict встановлюємо значення параметру «key», який потрібен для доступу до бази даних сайту http://seasonvar.ru/:

{"key","d3a00523" }

Параметр «command» у визначені словника dict є командою, яку повинен виконати сайт та повертає список сезонів (розділів) до обраного серіалу.

Останній параметр словника dict - «name», містить назву обраного серіалу:

{"name",dictionary[0]["name"].ToString() }

Для швидкого та зручного доступу до URL сайту http://seasonvar.ru/ оголошуємо глобальну змінну з посиланням на API сайту:

string url = "http://api.seasonvar.ru/";

Для відправки та отримання запитів через URL посилання, у методі Get_Serial оголошено змінну client типу HttpClient:

HttpClient client = new HttpClient();

Також у методі Get_Serial оголошено FormUrlEncodedContent для формування запиту у вигляді JSON об’єкту:

FormUrlEncodedContent form = new FormUrlEncodedContent(dict);

Відправка запиту до серверу та отримання його результату виконується об’єктом resp типу HttpResponseMessage:

HttpResponseMessage resp = await client.PostAsync(url, form);

Зчитування результату запиту списку сезонів з сайту SeasonVar зберігається у рядку:

string result = await resp.Content.ReadAsStringAsync(); Отриманий результат запиту сезонів серіалу перетворюється з об’єкту JSON у словник мови C# засобами класу JsonConvert:

dictionary = JsonConvert.DeserializeObject<List<Dictionary <string, object>>>(result);

Отриманим результатом конвертації заповнюємо макет екрану

програмного додатку даними про стрім:

textView1.Text = dictionary[0]["name"].ToString(); textView2.Text = dictionary[0]["genre"].ToString(); textView3.Text = dictionary[0]["description"].ToString(); var imageBitmap = GetImageBitmapFromUrl(dictionary[0]["poster_small"].ToString()); imageView1.SetImageBitmap(imageBitmap);

При пролістуванні екрана даними стріму необхідно виконувати очищення списку сезонів для подальшого заповнення новими даними:

lstSeason.Clear();

Фільтрацію та сортування сезонів за перекладом виконується методом - get_season().

За допомогою оператору циклу do-while формуємо список сезонів, список переводів під кожний сезон та список серій до кожного переводу відповідного сезону. У середині циклу:

– оголошуємо список назв груп стрімів (переклад, трейлери тощо.):

listDataHeader = new List<string>();

– оголошення словника зі списком для групування серій серіалу:

listDataChild = new Dictionary<string, List<string>>();

– створюємо словник з ключем та командою з параметром:

Dictionary<string, string> dict = new Dictionary<string, string>()
{

{"key","d3a00523" }, {"command", "getSeason" },

{"season_id", dictionary[i]["id"].ToString() }

};

– за допомогою класу HttpClient робимо запит до сайту:

HttpClient client = new HttpClient();

FormUrlEncodedContent form = new

FormUrlEncodedContent(dict);

HttpResponseMessage resp = await client.PostAsync(url, form); string result = await resp.Content.ReadAsStringAsync();

– десереалізуємо отримані дані:

dictionaryL=JsonConvert.DeserializeObject<Dictionary<string, object>>(result);

– окремо виділяємо плей лист з серіями до сезона: string ss = dictionaryL["playlist"].ToString();

– десереалізуємо список серій у сезоні:

lstDictionary=JsonConvert.DeserializeObject<List<Dictionary<string, object>>>(ss);

– додаємо назву перекладу «Original» для тих серій які не матимуть переклад:

listDataHeader.Add("Original");

– за допомогою for циклічно перевіряємо серії на переклад та формуємо список перекладу:

```
for (int j = 0; j < lstDictionary.Count; j++)
{
keys = lstDictionary[j].Keys.ToList(); if (keys.IndexOf("perevod") > -1)
{
if(listDataHeader.IndexOf(lstDictionary[j]["perevod"].
ToString())<0) listDataHeader.Add(lstDictionary[j]["perevod"].ToString());
}
}
```

– за допомогою вкладеного оператору циклу for циклічно перевіряємо кожну серію на наявність перекладу та додаємо її у список з відповідним перекладом.

```
for(int j = 0; j < listDataHeader.Count; j++){ links = new List<string>();
for (int h = 0; h < lstDictionary.Count; h++)
{
keys = lstDictionary[h].Keys.ToList();
if (keys.IndexOf("perevod") > -1)
{
if (listDataHeader[j] == lstDictionary[h]["perevod"].ToString())
links.Add(lstDictionary[h]["link"].ToString()); }else if(j == 0)
links.Add(lstDictionary[h]["link"].ToString()); } if(links.Count > 0)
listDataChild.Add(listDataHeader[j], links); } lstSeasons.Add(listDataChild);
i++;
} while (i < lstSeason.Count);
```

За допомогою перевизначених адаптерів (GridViewAdapterSeason, ExpandableListAdapter) виводимо списки у макет екрана програмного додатку:

```
GridViewAdapterSeason gridViewAdapter=new
```

```
GridViewAdapterSeason(lstSeasons, this);
        gridView1.Adapter = gridViewAdapter;
        }
```

2.5.3.2. Програмування парсингу

Процес і скрипт парсинга сайту складається з двох частин:

1. Потрібно отримати HTML код сторінки, якої нам необхідний.

2. Розбір отриманого коду зі збереженням даних і подальшої обробки їх (будемо використовувати phpQuery).

Для вирішення першого пункту в програмі написано простий клас з одним статичним методом, який буде обгорткою над CURL. Так код можна буде використовувати в подальшому і, якщо необхідно, модифікувати його.

Перше, з чим нам потрібно визначитися - як буде називатися клас і метод і які будуть у нього обов'язкові параметри:

```
class Parser{
        public static function getPage($params = []){
                if($params){
                        if(!empty($params["url"])){
                                $url = $params["url"];
                                // код
                        }
                }
                return false;
        }
}
```

Основний метод, який у нас буде - це getPage () і у нього всього один обов'язковий параметр URL сторінки, яку ми будемо аналізувати. Крім цього цей метод оброблятиме наступні значення:

– $ useragent - нам важливо мати можливість встановлювати

заголовок User-Agent, так ми зможемо зробити наші звернення до сервера схожими на звернення з браузера;

– $ timeout - буде відповідати за час виконання запиту на сервер;

– $ connecttimeout - так само важливо вказувати час очікування з'єднання;

– $ head - якщо нам буде потрібно перевірити тільки заголовки, які віддає сервер на наш запит цей параметр нам просто буде необхідний;

– $ cookie_file - файл, в який будуть записувати куки нашого донора контенту і при зверненні передаватися;

– $ cookie_session - забороняти передачу сесійних кук;

– $ proxy_ip – параметр IP проксі-сервера;

– $ proxy_port - відповідно порт проксі-сервера;

– $ proxy_type - тип проксі CURLPROXY_HTTP, CURLPROXY_SOCKS4, CURLPROXY_SOCKS5, CURLPROXY_SOCKS4A або CURLPROXY_SOCKS5_HOSTNAME;

– $ headers - масив заголовків;

– $ post - для відправки POST запиту.

Додамо їх в скрипт:

```
$useragent = !empty($params["useragent"]) ? $params["useragent"] : "Mozilla/5.0 (Windows NT 6.3; W…) Gecko/20100101 Firefox/57.0";
$timeout = !empty($params["timeout"]) ? $params["timeout"] : 5;
$connecttimeout = !empty($params["connecttimeout"]) ? $params["connecttimeout"] : 5;
$head = !empty($params["head"]) ? $params["head"] : false;
$cookie_file = !empty($params["cookie"]["file"]) ? $params["cookie"]["file"] : false;
$cookie_session = !empty($params["cookie"]["session"]) ? $params["cookie"]["session"] : false;
$proxy_ip = !empty($params["proxy"]["ip"]) ? $params["proxy"]["ip"] : false;
```

```
$proxy_port = !empty($params["proxy"]["port"]) ? $params["proxy"]["port"] : false;
$proxy_type = !empty($params["proxy"]["type"]) ? $params["proxy"]["type"] : false;
$headers = !empty($params["headers"]) ? $params["headers"] : false;
$post = !empty($params["post"]) ? $params["post"] : false;
```

У всіх параметрів є значення за замовчуванням.

Приклад коду, який буде очищати файл з куками при запиті:

```
if($cookie_file){
    file_put_contents(__DIR__."/".$cookie_file, "");
}
```

Так ми забезпечимо себе від ситуації, коли з якої-небудь причини не створився файл.

Для роботи з CURL нам необхідно спочатку ініціювати сеанс, а по завершенню роботи його закрити, також при роботі важливо врахувати можливі помилки, які напевно з'являться, а при успішному отриманні відповіді повернути результат, зробимо ми це таким образам:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
// код...
curl_setopt($ch, CURLINFO_HEADER_OUT, true);
$content = curl_exec($ch);
$info   = curl_getinfo($ch);
$error = false;
if($content === false){
    $data = false;
    $error["message"] = curl_error($ch);
    $error["code"]        = self::$error_codes[
```

```
                curl_errno($ch)
        ];
}else{
        $data["content"] = $content;
        $data["info"]        = $info;
}
curl_close($ch);
return [
        "data"        => $data,
        "error" => $error
];
```

$ error_codes - це масив з розшифровкою кодів функції curl_errno ():

```
private static $error_codes = [
        "CURLE_UNSUPPORTED_PROTOCOL",
        "CURLE_FAILED_INIT",
        ...
        "CURLE_FTP_BAD_FILE_LIST",
        "CURLE_CHUNK_FAILED"
];
```

Після того, як ми ініціалізували з'єднання через функцію curl_setopt (), встановимо кілька параметрів для поточного сеансу:

– CURLOPT_URL - перший і обов'язковий - це адреса, на який ми звертаємося;

– CURLINFO_HEADER_OUT - інформація про поточне з'єднання.

Використовуючи функцію curl_exec (), ми здійснюємо безпосередньо запит за допомогою CURL, а результат зберігаємо в змінну $ content, за замовчуванням. Після успішного відпрацювання результат відобразиться на екрані, а в $ content впаде true. Відстежити попутну інформацію при запиті нам допоможе функція curl_getinfo (). Також важливо, якщо станеться помилка -

результат спілкування буде false, тому, нижче за кодом ми використовуємо сувору рівність з урахуванням типів. Залишилося розглянути ще дві функції: це curl_error () - поверне повідомлення про помилку, і curl_errno () - код помилки. Результатом роботи методу getPage () буде масив. Для тесту зробимо запит на сервіс httpbin для отримання свого IP.

При успішному запиті ми отримуємо заповнену комірку масиву data з контентом та інформацією про запит, при помилці заповнюється осередок error. Щоб вирішити це, нам потрібно додати ще один параметр сеансу CURLOPT_RETURNTRANSFER.

```
curl_setopt ($ ch, CURLOPT_RETURNTRANSFER, true);
```

Звертаючись до сторінок, ми можемо виявити, що вони здійснюють редирект на інші, щоб отримати кінцевий результат додаємо:

```
curl_setopt ($ ch, CURLOPT_FOLLOWLOCATION, true);
```

Далі ми описали змінні $ useragent, $ timeout і $ connecttimeout. Додаємо їх в наш скрипт:

```
curl_setopt ($ ch, CURLOPT_USERAGENT, $ useragent);
curl_setopt ($ ch, CURLOPT_TIMEOUT, $ timeout);
curl_setopt ($ ch, CURLOPT_CONNECTTIMEOUT, $ connecttimeout);
```

Для того, щоб отримати заголовки відповіді, нам буде потрібно додати наступний код:

```
if ($ head) {
curl_setopt ($ ch, CURLOPT_HEADER, true);
curl_setopt ($ ch, CURLOPT_NOBODY, true);
```

```
}
```

Ми відключили висновок тіла документа і включили висновок шапки.

Для роботи з посиланнями з SSL сертифікатом, додаємо:

```
if (strpos ($ url, "https")! == false) {
curl_setopt ($ ch, CURLOPT_SSL_VERIFYHOST, true);
curl_setopt ($ ch, CURLOPT_SSL_VERIFYPEER, true);
}
```

Отриманий скрипт парсеру контенту і кук іноді можуть не зберігатися. Однією з основних причин може бути вказівка відносного шляху, тому нам варто це врахувати і написати такі рядки:

```
if ($ cookie_file) {
curl_setopt ($ ch, CURLOPT_COOKIEJAR, __DIR __. "/". $ cookie_file);
curl_setopt ($ ch, CURLOPT_COOKIEFILE, __DIR __. "/". $ cookie_file);
if ($ cookie_session) {
curl_setopt ($ ch, CURLOPT_COOKIESESSION, true);
}
}
```

Далі залишилося додати в параметри сеансу: проксі, заголовки і можливість відправки запитів POST:

```
if ($ proxy_ip && $ proxy_port && $ proxy_type) {
curl_setopt ($ ch, CURLOPT_PROXY, $ proxy_ip. ":". $ proxy_port);
curl_setopt ($ ch, CURLOPT_PROXYTYPE, $ proxy_type);
}
if ($ headers) {
```

```
curl_setopt ($ ch, CURLOPT_HTTPHEADER, $ headers);
}
if ($ post) {
curl_setopt ($ ch, CURLOPT_POSTFIELDS, $ post);
```

2.6. Обґрунтування та організація вхідних та вихідних даних програми

У якості вхідних даних виступає введена у поле пошуку назва відео для завантаження його в додаток.

Вихідними даними є отримана інформація про обраний відеопоток.

Програма відправляє запит до API, який буде приймати словник з даними Dictionary та повертати результат у вигляді рядку формату JSON та приймає рядок формату JSON і перетворює його у контейнер з вкладеним словником та виводить у список на головній сторінці.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного веб-сайту використовувався ПК з наступними характеристиками:

– CPU 1,3 + Hz;

– VGA 16 + Mb;

– RAM 128 + Mb;

– монітор (для найкращої роботи програми рекомендується монітор з роздільною здатністю 1680/1050 і більше);

– клавіатура: стандартна - 101/102 клавіші;

– миша;

– доступ в мережу Інтернет.

Для роботи даного додатку необхідно мобільний пристрій (наприклад, смартфон), що відповідає таким вимогам:

- операційна система Android версії 2.2 (Froyo) і вище;
- оптимальна роздільна здатність дисплею – 1280*720;
- доступ до мережі Інтернет.

Для тестування та демонстрації роботи даного додатку використовувався мобільний пристрій з наступними характеристиками:

Смартфон Huawei Y5 2019 (CRO-U00) DualSim з наступними характеристиками: екран (5", TFT, 854x480)/ MediaTek MT6580M (1.3 ГГц)/ / RAM 1 ГБ/ 16 ГБ вбудованої пам'яті + підтримка microSD/SDHC (до 128 ГБ)/ 3G/ GPS/ підтримка 2-х SIM-карток (Micro-SIM)/ Android 6.0 (Marshmallow).

2.7.2. Використані програмні засоби

Для розробки даного додатку застосовані інструменти розробки кросплатформених програмних додатків – Xamarin мовою C# та API.

2.7.3. Виклик та завантаження програми

Для повноцінної роботи програми необхідно встановити MX Player, глобальний офлайновий відеоплеєр і постачальник потокового мовлення Over the Top (OTT), створений J2 Interactive і належить Times Internet, підрозділу цифрових медіа Times Group.

На початку роботи з програмним додатком, його необхідно завантажити на смартфон у вигляді файлу VideoBox.apk. Варіанти завантаження файла можуть бути різні:

- з інтернету за посиланням хмарного сховища;
- за допомогою кабелю USB з підключенням смартфон-ПК.

При встановленні файлу VideoBox.apk на смартфон користувачу необхідно запустити додаток та увімкнути інтернет на смартфоні.

2.7.4. Опис інтерфейсу користувача

Після запуску додатку з’являється головне меню додатку з функціональними кнопками (рис. 2.13).

Рис. 2.13. Функціональні кнопки головного меню додатку

Користувач може обрати відеоконтент зі списку (рис. 2.14) або скористатися пошуком (рис. 2.15-2.16).

Рис. 2.13. Початковий екран додатку

Рис. 2.15. Кнопка пошуку контенту

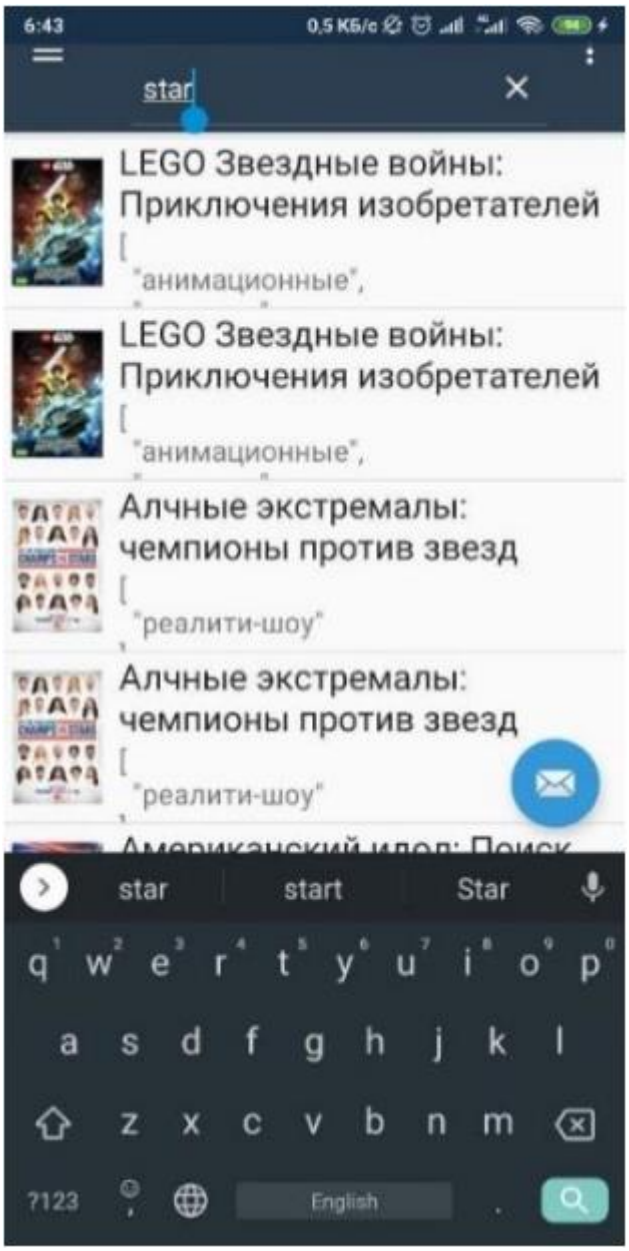

Рис. 2.16. Застосування пошуку

Для перегляду історії переглядів або плей листу обраних стрімів, необхідно натиснути на елемент у вигляді трьох рисок у верхньому лівому куті (рис. 2.17-2.18).

Рис. 2.17. Кнопка для перегляду історій переглядів

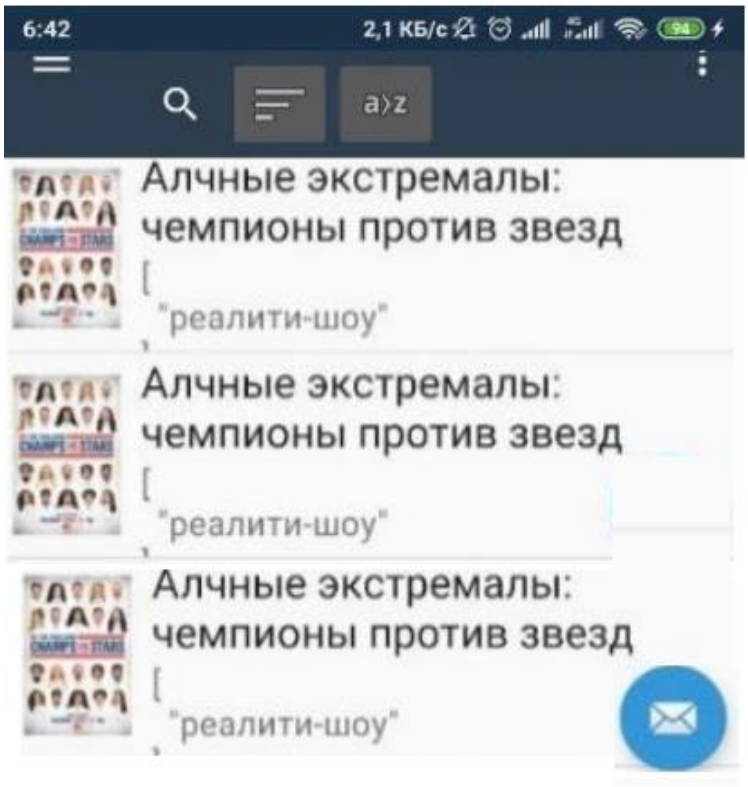

Рис. 2.18. Історія перегляду

Після того, як користувач обрав серіал, йому відкриється вікно серіалу, де він може додати до обраного стрім, обрати сезон та переклад серіалу і після обрати потрібну серію (рис. 2.19).

Рис. 2.19. Екран обраного серіалу

Коли користувач обрав потрібну серію, система запропонує йому обрати додаток через, який буде запущено стрім на його мобільному пристрої (рис. 2.20). Обравши необхідний додаток відтворення відео, обраний відео файл запуститься на екрані (рис. 2.21).

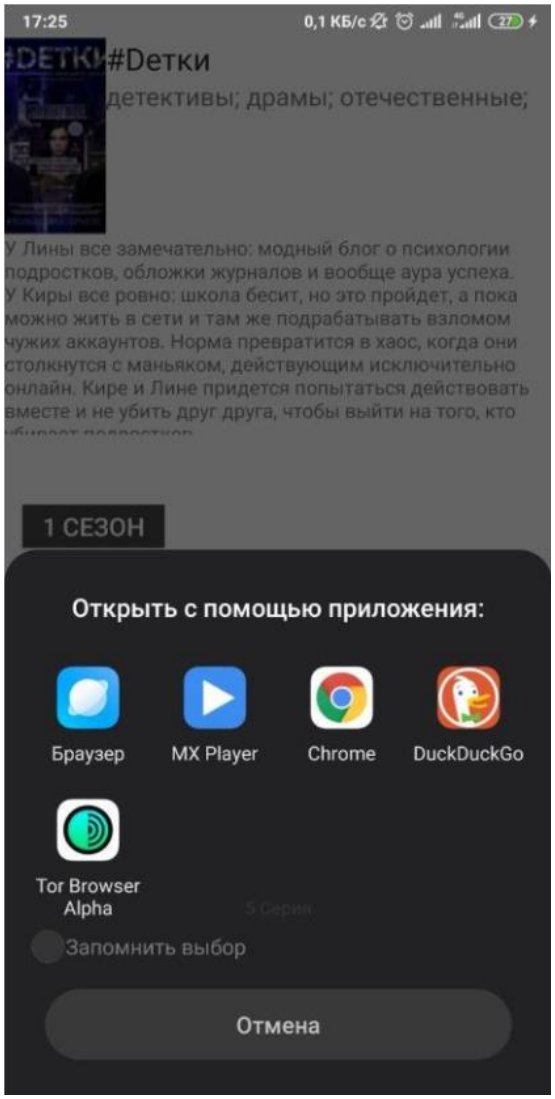

Рис. 2.20. Вибір додатку, через який буде відкрито стрім

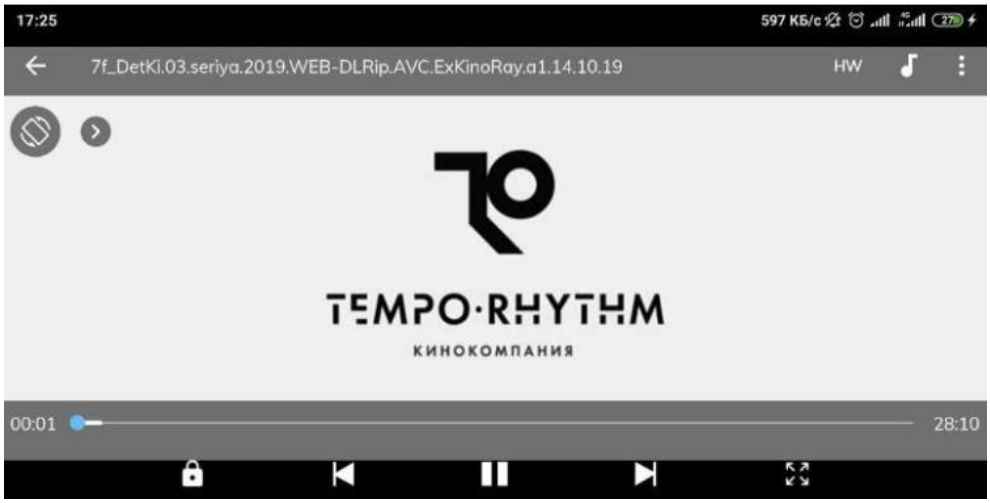

Рис. 2.21. Стрім через додаток «MX Player»

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1200;
- коефіцієнт складності програми – 1,7;
- коефіцієнт корекції програми в ході її розробки – 0,1;
- годинна заробітна плата програміста, грн / год – 30.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти – витрати праці на дослідження алгоритму рішення задачі;

та – витрати праці на розробку блок-схеми алгоритму;

тп – витрати праці на програмування по готовій блок-схемі;

тотл – витрати праці на налагодження програми на ЕОМ;

тд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q – передбачуване число операторів;

С – коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1200 \cdot 1{,}7 \cdot (1 + 0{,}1) = 2244 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2244 \cdot 1{,}2}{80 \cdot 0{,}8} = 42 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2244}{20 \cdot 0{,}8} = 140 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2244}{25 \cdot 0{,}8} = 112 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2244}{4 \cdot 0{,}8} = 701 \text{ людино-годин.}$$

За умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл} = 701 * 1{,}2 = 841{,}5$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр – трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.9}$$

$$t_{др} = \frac{2244}{15 \cdot 0{,}8} = 187 \text{ людино-годин,}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.10)$$

$$tdo = 0{,}75 \cdot 187 = 140$$

$$t_{до} = 140 + 187 = 327 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 42 + 140 + 112 + 841 + 327 = 1513 \text{ людино-годин.}$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

Спр – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1513 \cdot 30 = 45398 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{М}, \text{ грн,} \quad (3.13)$$

де tотл – трудомісткість налагодження програми на ЕОМ, год.

Смч – вартість машино-години ЕОМ, 5 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 841 \times 5 = 4207 \text{ грн.}$$

$$Кпо = 4207 + 45398 = 49605 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.14)$$

де Bk – число виконавців;

Fp – місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1531}{1 \cdot 176} = 8{,}6 \text{ міс.}$$

Визначено трудомісткість розробленої інформаційної системи (1513 люд-год), проведений підрахунок вартості роботи по створенню програми (49605 грн.) та розраховано час на його створення (8,6 міс).

ВИСНОВКИ

Під час виконання завдання кваліфікаційної роботи відповідно до постановки задачі, було спроектовано та розроблено мобільний додаток «VideoBoxSeasonVar» для операційної системи Android, який призначений для перегляду відеопотоків з сайту http://seasonvar.ru/ засобами парсингу – технологій API.

Даний додаток розроблено за допомогою інструменту Xamarin, мовою C# та мовою розмітки XML для розробки візуального інтерфейсу.

Програма відправляє запит до API, який приймає словник з даними Dictionary та повертає результат у вигляді рядку формату JSON та приймає рядок формату JSON і перетворює його у контейнер з вкладеним словником та виводить у список на головній сторінці.

Програмний додаток забезпечує:

- відображення списку відеофійлів;
- фільтрацію та сортування списку;
- відображення результатів пошуку у вигляді списку;
- відображення повної інформації по обраному відеофайлу;
- запис в історію перегляду переглянуті відео;
- додавання відео до списку обраного.

Розроблений мобільний додаток успішно протестовано.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (1513 люд-год), проведений підрахунок вартості роботи по створенню програми (49605 грн.) та розраховано час на його створення (8,6 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура ОС Android / URL: http://refua.in.ua/rozrobka-programnogoza bezpechennya-dlya-vidtvorennya-fajliv-m.html?page=5. дата звернення 21.12.20

2. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

3. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

4. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2018.

5. Бюджетные смартфоны 2020 / URL: http://www.smart-android.info/2020/04/21/ luchshie-byudzhetnye-smartfony-2020-goda-za-100-dollarov/3592/. дата звернення 21.12.20

6. Введение в JSON / URL: https://www.json.org/json-ru. дата звернення 21.12.20

7. Дібрівний О. А. Вступ до об’єктно орієнтованого програмування C#: Навчальний посібник / О. А. Дібрівний, В. В. Гребенюк., 2018.Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

8. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

9. Кулик М. С. ПОЛІТ СУЧАСНІ ПРОБЛЕМИ НАУКИ / М. С. Кулик. – Київ, 2014. – 110 с. – (Національний авіаційний університет Інститут комп'ютерних інформаційних технологій).Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності "Комп'ютерні системи " / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

10. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

11. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

12. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

13. Шматко О. В. Аналіз методів і технологій розробки мобільних додатків для платформи Android. частина 2. / О. В. Шматко, А. О. Поляков, В. М. Федорченко., 2018.

14. Applic ation_Programming_Interface_(API/ URL: https:// www.tadviser.ru/ index.php/Статья:Application_Programming_Interface_ (API). https://itchief.ru/javascript/json. дата звернення 21.12.20

15. C# programing language / URL: https://sites.google.com/site/ cprograminglanguage1/osoblivosti -movi. дата звернення 21.12.20

16. Hdvideobox/ URL: https://hdvideobox.ru/ дата звернення 21.12.20

17. JavaScript – формат JSON / URL: https://itchief.ru/javascript/json. дата звернення 21.12.20

18. Microsoft Visual Studio / URL: https://uk.wikipedia.org/wiki/Microsoft_ Visual_Studio. дата звернення 21.12.20

19. REST API / URL: https://leodev.ru/blog/webservices/rest-api-заметка/. дата звернення 21.12.20

20. Seasonvar / URL: http://seasonvar.ru/ дата звернення 21.12.20

ДОДАТОК А

КОД ПРОГРАМИ

VideoBox.cs

```
using System;
using System.Collections.Generic;
using System.Net.Http;

using Android;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;

using Android.Support.Design.Widget;
using Android.Support.V4.View;
using Android.Support.V4.Widget;
using Android.Support.V7.App;
using Android.Views;
using Android.Widget;

using Newtonsoft.Json;
using Newtonsoft;
using VideoBox.Resources.layout;
using VideoBox.Resources;
using System.Net;
using Newtonsoft.Json.Linq;

using System.IO;
using Android.Graphics;

namespace VideoBox
{
```

```
[Activity(Theme = "@style/AppTheme.NoActionBar", MainLauncher = true)]

public class MainActivity : AppCompatActivity,
NavigationView.IOnNavigationItemSelectedListener
{
SearchView searchView1;
ListView listView;
List<Dictionary<string, object>> dictionary = new List<Dictionary<string, object>>();

public static string get_quary { get; set; }
protected override void OnCreate(Bundle savedInstanceState)
{

base.OnCreate(savedInstanceState);
Xamarin.Essentials.Platform.Init(this, savedInstanceState);
SetContentView(Resource.Layout.activity_main);

Android.Support.V7.Widget.Toolbar toolbar =
FindViewById<Android.Support.V7.Widget.Toolbar>(Resource.Id.toolbarOld);
SetSupportActionBar(toolbar);

FloatingActionButton fab = FindViewById<FloatingActionButton>(Resource.Id.fab);
fab.Click += FabOnClick;

DrawerLayout drawer = FindViewById<DrawerLayout>(Resource.Id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, toolbar,
Resource.String.navigation_drawer_open, Resource.String.navigation_drawer_close);
drawer.AddDrawerListener(toggle);
toggle.SyncState();

NavigationView navigationView =
FindViewById<NavigationView>(Resource.Id.nav_view);
navigationView.SetNavigationItemSelectedListener(this);

searchView1 = FindViewById<SearchView>(Resource.Id.searchView1);
```

```
searchView1.QueryTextSubmit += SearchView1_QueryTextSubmit;
        get_video_lists();

        listView = FindViewById<ListView>(Resource.Id.listView1); listView.ItemClick +=
ListView_ItemClick; listView.Scroll += ListView_Scroll; List<ListItem> ListItems = new
List<ListItem>();
        ListItems.Add(new ListItem()
        {

        name = "Loading..."
        });

        listView.Adapter = new VideoAdapter(this, ListItems);
        }

        private void ListView_Scroll(object sender, AbsListView.ScrollEventArgs e)
        {
        if(listView.LastVisiblePosition == listView.Count-1 && listView.LastVisiblePosition > 1)

        {
        int posietion = listView.LastVisiblePosition;

        Toast.MakeText(this, listView.LastVisiblePosition.ToString() + ":" +
listView.Count.ToString(), ToastLength.Short).Show();

        List<ListItem> ListItems = new List<ListItem>();
        int cc = listView.Count + 10;
        for (int i = 0; i < cc; i++)
        {

        string genre = "";

        List<string> aaa =
JsonConvert.DeserializeObject<List<string>>(dictionary[i]["genre"].ToString()); for (int j = 0; j <
aaa.Count; j++)
```

```
genre += aaa[j]+", ";

var imageBitmap = GetImageBitmapFromUrl(dictionary[i]["poster_small"].ToString());
ListItems.Add(new ListItem()

{
name = dictionary[i]["name"].ToString(),
text = genre,
imageBitmap = imageBitmap
});
}
ListItems.Add(new ListItem()
{
name = "swipe up"
});
ListItems.Add(new ListItem());

listView.Adapter = new VideoAdapter(this, ListItems);
listView.ScrollTo(posietion,posietion);
}
}

private void ListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
{
var dd2 = e.Position.ToString();
Intent intent = new Intent(this, typeof(content_item));
StartActivity(intent);

content_item.dictionary = new List<Dictionary<string, object>>() {
dictionary[Int32.Parse(dd2)] };
}
public List<Dictionary<string,object>> get_dictionary()
{
return dictionary;
}
```

```
private Bitmap GetImageBitmapFromUrl(string url)
{
Bitmap imageBitmap = null;
using (var webClient = new WebClient())
 {
var imageBytes = webClient.DownloadData(url);
if (imageBytes != null && imageBytes.Length > 0)
{
imageBitmap = BitmapFactory.DecodeByteArray(imageBytes, 0, imageBytes.Length);
}
}

return imageBitmap;
}

private async void SearchView1_QueryTextSubmit(object sender,
SearchView.QueryTextSubmitEventArgs e)//search
{

List<ListItem> ListItems1 = new List<ListItem>();
string query = searchView1.Query;
Dictionary<string, string> dict = new Dictionary<string, string>()
{
{"key","d3a00523" },
{"command","search" },
{"query", query }
};
string url = "http://api.seasonvar.ru/";
HttpClient client = new HttpClient();
FormUrlEncodedContent form = new FormUrlEncodedContent(dict);
HttpResponseMessage resp = await client.PostAsync(url, form);

string result = await resp.Content.ReadAsStringAsync();
```

```
dictionary = JsonConvert.DeserializeObject<List<Dictionary<string, object>>>(result);
for (int i = 0; i < dictionary.Count; i++)
{
var imageBitmap = GetImageBitmapFromUrl(dictionary[i]["poster_small"].ToString());

ListItems1.Add(new ListItem()
{
name = dictionary[i]["name"].ToString(),
text = dictionary[i]["genre"].ToString(),
imageBitmap = imageBitmap

});
}

listView.Adapter = new VideoAdapter(this, ListItems1);
}
private async void get_video_lists()
{
List<ListItem> ListItems = new List<ListItem>();

Dictionary<string, string> dict = new Dictionary<string, string>()
{
{"key","d3a00523" },
{"command","getSerialList" }
};
string url = "http://api.seasonvar.ru/";
HttpClient client = new HttpClient();
FormUrlEncodedContent form = new FormUrlEncodedContent(dict);

HttpResponseMessage resp = await client.PostAsync(url, form);
string result = await resp.Content.ReadAsStringAsync();
if (result.IndexOf("this ip is not allowed") < 0)
{

//Dictionary<string, object> dicttionary =
```

```
JsonConvert.DeserializeObject<Dictionary<string, object>>(result); dictionary =
JsonConvert.DeserializeObject<List<Dictionary<string, object>>>(result);
        for (int i = 0; i < 10; i++)
        {
        string genre = "";

        List<string> aaa =
JsonConvert.DeserializeObject<List<string>>(dictionary[i]["genre"].ToString()); for (int j = 0; j <
aaa.Count; j++)
        genre += aaa[j]+", ";

        var imageBitmap = GetImageBitmapFromUrl(dictionary[i]["poster_small"].ToString());
        ListItems.Add(new ListItem()
        {
        name = dictionary[i]["name"].ToString(),
        text = genre,
        imageBitmap = imageBitmap
        }) ;
        }
        ListItems.Add(new ListItem()
        {
        name = "swipe up"
        });
        ListItems.Add(new ListItem());
        listView.Adapter = new VideoAdapter(this, ListItems);
        }

        else
        {
        Toast.MakeText(this, "this ip is not allowed", ToastLength.Short).Show();
        }
        }

        private void Content_item_create()
        {
```

```
Intent intent = new Intent(this, typeof(content_item));
StartActivity(intent);
}
public class VideoAdapter : BaseAdapter<ListItem>
{
List<ListItem> items;
Activity context;
public VideoAdapter(Activity context, List<ListItem> items): base()
{
this.context = context;
this.items = items;
}
public override long GetItemId(int position)
{
return position;
}
public override ListItem this[int position]
{
get { return items[position]; }
}

public override int Count
{
get { return items.Count; }
}
public override View GetView(int position, View convertView, ViewGroup parent)
{
var item = items[position];

View view = convertView;
if (view == null) // no view to re-use, create new

view = context.LayoutInflater.Inflate(Resource.Layout.list_item, null);
view.FindViewById<TextView>(Resource.Id.textView1).Text = item.name;
view.FindViewById<TextView>(Resource.Id.textView2).Text = item.text;
```

```
view.FindViewById<ImageView>(Resource.Id.imageView1).SetImageBitmap(item.imageBitmap);

        return view;
        }
        }

        public class ListItem
        {
        public string name { get; set; }
        public string text { get; set; }
        public Bitmap imageBitmap  { get; set; }
        }

        public override void OnBackPressed()
        {
        DrawerLayout drawer = FindViewById<DrawerLayout>(Resource.Id.drawer_layout);
if(drawer.IsDrawerOpen(GravityCompat.Start)) {
        drawer.CloseDrawer(GravityCompat.Start);

        }
        else
        {
        base.OnBackPressed();
        }
        }

        public override bool OnCreateOptionsMenu(IMenu menu)
        {
        MenuInflater.Inflate(Resource.Menu.top_menus, menu);
        return base.OnCreateOptionsMenu(menu);
        }

        public override bool OnOptionsItemSelected(IMenuItem item)
        {
        Toast.MakeText(this, "Action selected: " + item.TitleFormatted,
```

```
ToastLength.Short).Show();

        return base.OnOptionsItemSelected(item);
        }

        private void FabOnClick(object sender, EventArgs eventArgs)
        {
        View view = (View) sender;
        Snackbar.Make(view, "Replace with your own action", Snackbar.LengthLong)
        .SetAction("Action", (Android.Views.View.IOnClickListener)null).Show();
        }

        public override void OnRequestPermissionsResult(int requestCode, string[] permissions,
[GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
        Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions,
grantResults);

        base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }
        }
        }
```

Content_item.cs

```
        using System;
        using System.Collections.Generic;
        using System.Linq;

        using System.Text;
        using System.Json;
        using Android.App;
        using System.Dynamic;
         using Android.Content;
        using Android.OS;
```

```
using Android.Runtime;

using Android.Views;
using Android.Widget;
using System.Net;
using Newtonsoft.Json;

using Newtonsoft.Json.Linq;
using VideoBox.Resources.layout;
using System.Net.Http;
using Android.Graphics;
using Android.Webkit;

using VideoBox.Adapter;
namespace VideoBox.Resources.layout
{

[Activity(Label = "@string/app_name", Theme = "@style/AppTheme.NoActionBar",
MainLauncher = true)]
class content_item : Activity
{

public static List<string> lstSeason = new List<string>();

public static List<Dictionary<string, object>> dictionary { get; set; } string quary;
ImageView imageView1;
TextView textView1;
TextView textView2;
TextView textView3;
ExpandableListView expListView1;
GridView gridView1;

protected override void OnCreate(Bundle savedInstanceState)
{
base.OnCreate(savedInstanceState);
```

```
SetContentView(Resource.Layout.content_item);
gridView1 = FindViewById<GridView>(Resource.Id.gridView1);
expListView1 = FindViewById<ExpandableListView>(Resource.Id.expListView1);
imageView1 = FindViewById<ImageView>(Resource.Id.imageView1); textView1 =
FindViewById<TextView>(Resource.Id.textView1); textView2 =
FindViewById<TextView>(Resource.Id.textView2); textView3 =
FindViewById<TextView>(Resource.Id.textView3); get_serial();
}

public async void get_season()
{
List<Dictionary<string, object>> lstDictionary = new List<Dictionary<string, object>>();
Dictionary<string, object> dictionaryL = new Dictionary<string, object>(); Dictionary<string,
List<string>> listDataChild = new Dictionary<string, List<string>>(); List<Dictionary<string,
List<string>>> lstSeasons = new List<Dictionary<string, List<string>>>(); List<string>
listDataHeader = new List<string>(); List<string> keys = new List<string>();
List<string> links = new List<string>();
int i = 0;
do
{

listDataHeader = new List<string>();

listDataChild = new Dictionary<string, List<string>>(); Dictionary<string, string> dict =
new Dictionary<string, string>() {
 {"key","d3a00523" },
{"command", "getSeason" },
{"season_id", dictionary[i]["id"].ToString() }
};
string url = "http://api.seasonvar.ru/";
HttpClient client = new HttpClient();
FormUrlEncodedContent form = new FormUrlEncodedContent(dict);

HttpResponseMessage resp = await client.PostAsync(url, form);
string result = await resp.Content.ReadAsStringAsync();
```

```
dictionaryL = JsonConvert.DeserializeObject<Dictionary<string, object>>(result); string ss = dictionaryL["playlist"].ToString();
lstDictionary = JsonConvert.DeserializeObject<List<Dictionary<string, object>>>(ss); listDataHeader.Add("Original");

for (int j = 0; j < lstDictionary.Count; j++)
{
keys = lstDictionary[j].Keys.ToList();
if (keys.IndexOf("perevod") > -1)
{
if (listDataHeader.IndexOf(lstDictionary[j]["perevod"].ToString()) < 0)
listDataHeader.Add(lstDictionary[j]["perevod"].ToString());
}
}
for(int j = 0; j < listDataHeader.Count; j++)
{
links = new List<string>();
for (int h = 0; h < lstDictionary.Count; h++)
{
keys = lstDictionary[h].Keys.ToList();

if (keys.IndexOf("perevod") > -1)
{
if (listDataHeader[j] == lstDictionary[h]["perevod"].ToString())
links.Add(lstDictionary[h]["link"].ToString());
}else if(j == 0) links.Add(lstDictionary[h]["link"].ToString());
}
if(links.Count > 0)
listDataChild.Add(listDataHeader[j], links);
}
lstSeasons.Add(listDataChild);
i++;
} while (i < lstSeason.Count);

GridViewAdapterSeason gridViewAdapter = new GridViewAdapterSeason(lstSeasons,
```

```
this); gridView1.Adapter = gridViewAdapter;

        }
        private async void get_serial()
        {
        Dictionary<string, string> dict = new Dictionary<string, string>()
        {
        {"key","d3a00523" },
        {"command","getSeasonList" },
        {"name",dictionary[0]["name"].ToString() }
        };
        string url = "http://api.seasonvar.ru/";
        HttpClient client = new HttpClient();
        FormUrlEncodedContent form = new FormUrlEncodedContent(dict);
        HttpResponseMessage resp = await client.PostAsync(url, form);
        string result = await resp.Content.ReadAsStringAsync();
        dictionary = JsonConvert.DeserializeObject<List<Dictionary<string, object>>>(result);
        List<string> genre =
JsonConvert.DeserializeObject<List<string>>(dictionary[0]["genre"].ToString());
         string genreG = "";
        for (int j = 0; j < genre.Count; j++)
        genreG += genre[j] + "; ";

        textView1.Text = dictionary[0]["name"].ToString();
        textView2.Text = genreG;
        textView3.Text = dictionary[0]["description"].ToString();

        var imageBitmap = GetImageBitmapFromUrl(dictionary[0]["poster_small"].ToString());
imageView1.SetImageBitmap(imageBitmap); lstSeason.Clear();

        for (int j = 1; j < dictionary.Count + 1; j++)
        lstSeason.Add(j.ToString() + " Сезон");
        get_season();
        }
```

```
private Bitmap GetImageBitmapFromUrl(string url)
{
Bitmap imageBitmap = null;
using (var webClient = new WebClient())
{
var imageBytes = webClient.DownloadData(url);
if (imageBytes != null && imageBytes.Length > 0)
{
imageBitmap = BitmapFactory.DecodeByteArray(imageBytes, 0, imageBytes.Length);
}
}
return imageBitmap;
}
}
}
```

GridViewAdapterSeason.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.Graphics;
using Android.OS;
using Android.Runtime;
using Android.Util;

using Android.Views;
using Android.Widget;
using VideoBox.Resources.layout;
namespace VideoBox.Adapter
{
```

```
public class ExpandableListAdapter : BaseExpandableListAdapter {
private Activity _context;
private List<string> _listDataHeader;
private Dictionary<string, List<string>> _listDataChild;
public ExpandableListAdapter(Activity context, List<string> listDataHeader,
Dictionary<string, List<string>> listChildData)

{
_context = context;
_listDataHeader = listDataHeader;
 _listDataChild = listChildData;
}
public override Java.Lang.Object GetChild(int groupPosition, int childPosition)
{
return _listDataChild[_listDataHeader[groupPosition]][childPosition];
}
public override long GetChildId(int groupPosition, int childPosition)
{
return childPosition;
}

public override View GetChildView(int groupPosition, int childPosition, bool isLastChild,
View convertView, ViewGroup parent)
{
string childText = (childPosition + 1) + " Серия";
if (convertView == null)
{
convertView = _context.LayoutInflater.Inflate(Resource.Layout.ListItemCustomLayout,
null);

}
TextView txtListChild =
(TextView)convertView.FindViewById(Resource.Id.lblListItem1);
txtListChild.Text = childText;
convertView.Click += delegate
```

```
{

List<string> keys = _listDataChild.Keys.ToList(); Toast.MakeText(_context, childText, ToastLength.Short).Show(); var coo = this._context as Activity;

var uri = Android.Net.Uri.Parse(_listDataChild[keys[groupPosition]][childPosition]); var intent = new Intent(Intent.ActionView, uri); coo.StartActivity(intent);
};
return convertView;
}
public override int GetChildrenCount(int groupPosition)
{
return _listDataChild[_listDataHeader[groupPosition]].Count;
}
//For header view
public override Java.Lang.Object GetGroup(int groupPosition)
{
return _listDataHeader[groupPosition];
}

public override int GroupCount
{
get
{
return _listDataHeader.Count;
}
}
public override long GetGroupId(int groupPosition)
{
return groupPosition;
}
public override View GetGroupView(int groupPosition, bool isExpanded, View convertView, ViewGroup parent)
{
string headerTitle = (string)GetGroup(groupPosition);
```

```
        convertView = convertView ??
_context.LayoutInflater.Inflate(Resource.Layout.HeaderCustomLayout, null);
        var lblListHeader = (TextView)convertView.FindViewById(Resource.Id.lblListHeader);
        lblListHeader.Text = headerTitle;
        return convertView;
        }
        public override bool HasStableIds
         {
        get
        {

        return false;
        }
        }
        public override bool IsChildSelectable(int groupPosition, int childPosition)

        {
        return true;
        }

        class ViewHolderItem : Java.Lang.Object
        {
        }
        public void AddChild(int groupPosition, string newitem)
        {
        var children = _listDataChild[_listDataHeader[groupPosition]]; children.Add(newitem);
        }
        }

        public class GridViewAdapterSeason : BaseAdapter
        {
        List<Dictionary<string, List<string>>> lstSeries;
        Context context;
```

```
public GridViewAdapterSeason(List<Dictionary<string, List<string>>> lstSeries, Context
context) {
this.lstSeries = lstSeries;
this.context = context;
}

public override int Count
{
get
{
return lstSeries.Count;
}
}

public override Java.Lang.Object GetItem(int position)
{
return position;

}

public override long GetItemId(int position)
{
return position;
}

public override View GetView(int position, View convertView, ViewGroup parent)
{
Button button = null;
if(convertView==null)
{
button = new Button(context);
button.LayoutParameters = new GridView.LayoutParams(280, 85);
button.SetPadding(8, 8, 8, 8);
button.SetBackgroundColor(Color.DarkGray);
button.SetTextColor(Color.White);
```

```
button.SetTextSize(ComplexUnitType.Fraction, 50);
button.Text = (position + 1) + " Сезон";
button.Click += delegate
{

Toast.MakeText(context, button.Text, ToastLength.Short).Show(); //View view =
convertView;
var coo = this.context as Activity;
ExpandableListAdapter gridViewAdapter1 = new ExpandableListAdapter(coo,
lstSeries[position].Keys.ToList(), lstSeries[position]);
coo.FindViewById<ExpandableListView>(Resource.Id.expListView1).SetAdapter(gridVie
wAdapter1); //view.FindViewById<GridView>(Resource.Id.gridView2).Adapter =
gridViewAdapter1;
};

}
else
button = (Button)convertView;
return button;
}
}

public class GridViewAdapterSerial:BaseAdapter
{
List<List<string>> lstSource;
Context context;
int season;
public GridViewAdapterSerial(List<List<string>> lstSource, int season, Context context)
{
this.lstSource = lstSource;
this.context = context;
this.season = season;
}
public override int Count
{
```

```
get
{
return lstSource[season].Count;
}
}

public override Java.Lang.Object GetItem(int position)
{
return position;
}

public override long GetItemId(int position)
{
return position;
}

public override View GetView(int position, View convertView, ViewGroup parent)
{
Button button = null;
if (convertView == null)
{
button = new Button(context);
button.LayoutParameters = new GridView.LayoutParams(200, 85);
button.SetPadding(8, 8, 8, 8);
button.SetBackgroundColor(Color.DarkGray);
button.SetTextColor(Color.White);
button.Text = (position + 1).ToString()+" серия";
button.Click += delegate
{

Toast.MakeText(context, lstSource[season][position], ToastLength.Short).Show(); var coo =
this.context as Activity;
var uri = Android.Net.Uri.Parse(lstSource[season][position]);
var intent = new Intent(Intent.ActionView, uri);
 coo.StartActivity(intent);
```

```
};
}

else
button = (Button)convertView;
return button;
}
}
}
```

list_item.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"

android:layout_width="match_parent"
android:layout_height="match_parent">

<LinearLayout
android:orientation="horizontal"
android:minWidth="25px"
android:minHeight="25px"

android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/linearLayout2">
<ImageView
android:src="@android:drawable/ic_menu_gallery"
android:layout_width="wrap_content"

android:layout_height="wrap_content"
android:minWidth="200px"
android:minHeight="296px"
android:id="@+id/imageView1"
android:paddingBottom="10dp"
```

```
android:paddingLeft="5dp"
android:paddingRight="10dp"
android:paddingTop="10dp" />
<LinearLayout
android:orientation="vertical"
android:minWidth="25px"

android:minHeight="25px"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/linearLayout1">
<TextView
android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/textView1" />
<TextView
android:text="Medium Text"

android:textAppearance="?android:attr/textAppearanceMedium"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/textView2" />
</LinearLayout>

</LinearLayout>
</LinearLayout>
```

listItemCustomLayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
```

```
android:layout_width="match_parent"
android:layout_height="55dip">
<TextView

android:text="Text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:minWidth="25px"

android:minHeight="25px"
android:textSize="16dp"
android:paddingLeft="?android:attr/expandableListPreferredChildPaddingLeft"
android:textColor="#fff69212"
android:textStyle="bold"

android:background="#ffffffff"
android:paddingTop="5dp"
android:paddingBottom="5dp"
android:id="@+id/lblListItem1" />

</LinearLayout>
```

headerCustomLayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="8dp"
android:background="#fff69212">

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:minWidth="25px"
android:paddingLeft="?android:attr/expandableListPreferredItemPaddingLeft"

android:minHeight="25px"
android:textSize="17dp"
android:text="test"
android:textColor="#ffffffff"
android:textStyle="bold"
android:id="@+id/lblListHeader" />

</LinearLayout>
```

Content_item.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"

android:layout_width="match_parent"
tools:showIn="@layout/app_bar_main"
android:layout_height="match_parent"
android:overScrollMode="always"
android:scrollbars="none|vertical"
android:verticalScrollbarPosition="defaultPosition">

<LinearLayout
android:orientation="horizontal"
android:minWidth="25px"
android:minHeight="25px"
android:layout_width="match_parent"

android:layout_height="141.5dp"
```

```
android:id="@+id/linearLayout1">
<ImageView
android:src="@android:drawable/ic_menu_gallery"
android:layout_width="wrap_content"

android:layout_height="match_parent"
android:scaleType="centerCrop"
android:id="@+id/imageView1"

android:maxWidth="400px"
android:maxHeight="600px"
android:minWidth="200px"
android:minHeight="2960px"

android:layout_marginRight="0.0dp"
android:layout_marginBottom="0.0dp" />
<LinearLayout
android:orientation="vertical"
android:minWidth="25px"

android:minHeight="25px"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/linearLayout2">

<TextView
android:text="Large Text"
android:textAppearance="?android:attr/textAppearanceLarge"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/textView1" />
<TextView
android:text="Medium Text"

android:textAppearance="?android:attr/textAppearanceMedium"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/textView2" />
</LinearLayout>
</LinearLayout>

<LinearLayout
android:orientation="vertical"
android:minWidth="25px"
android:minHeight="25px"
android:layout_width="match_parent"

android:layout_height="195.0dp"
android:id="@+id/linearLayout3">
<TextView
android:text="Small Text"
android:textAppearance="?android:attr/textAppearanceSmall"
android:layout_width="match_parent"

android:layout_height="144.5dp"
android:id="@+id/textView3"
android:overScrollMode="ifContentScrolls"

android:scrollbars="none|vertical"
android:verticalScrollbarPosition="right" />
</LinearLayout>

<GridView
android:minWidth="25px"
android:minHeight="25px"
android:layout_width="match_parent"
android:layout_height="101.0dp"

android:columnWidth="100dp"
```

```
android:numColumns="auto_fit"
android:verticalSpacing="4dp"
android:horizontalSpacing="4dp"
android:gravity="center"
android:id="@+id/gridView1" />

<ExpandableListView
android:minWidth="25px"
android:minHeight="25px"
android:layout_width="match_parent"

android:layout_height="match_parent"
android:id="@+id/expListView1" />
</LinearLayout>
```

App_bar_main.xml

```
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"
android:layout_height="match_parent">

<android.support.design.widget.AppBarLayout
android:layout_width="match_parent"

android:layout_height="wrap_content"
android:theme="@style/AppTheme.AppBarOverlay">

<android.support.v7.widget.Toolbar
android:id="@+id/toolbarOld"
android:layout_width="match_parent"

android:layout_height="?attr/actionBarSize"
```

```
android:background="?attr/colorPrimary"
app:popupTheme="@style/AppTheme.PopupOverlay"
android:minWidth="25px"
android:minHeight="25px">
<SearchView

android:minWidth="25px"
android:minHeight="25px"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/searchView1"

android:layout_marginRight="0.0dp"
android:gravity="right" />
<ImageButton
android:src="@android:drawable/ic_menu_sort_by_size"
android:layout_width="wrap_content"
android:layout_height="wrap_content"

android:id="@+id/imageButton1" />
<ImageButton
android:src="@android:drawable/ic_menu_sort_alphabetically"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageButton2" />

</android.support.v7.widget.Toolbar>

</android.support.design.widget.AppBarLayout>

<include
layout="@layout/content_main" />

<android.support.design.widget.FloatingActionButton
android:id="@+id/fab"
```

```
android:layout_width="wrap_content"

android:layout_height="wrap_content"
android:layout_gravity="bottom|end"
android:layout_margin="@dimen/fab_margin"

app:srcCompat="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

Content_main.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"
android:layout_height="match_parent">
<ListView
android:minWidth="25px"
android:minHeight="25px"
android:layout_width="wrap_content"

android:layout_height="wrap_content"
android:id="@+id/listView1" />
</RelativeLayout>
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи