**АНОТАЦІЯ**

**ABSTRAСT**

**ЗМІСТ**

[ВСТУП 7](#_Toc453441982)

[Розділ 1. Огляд програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 8](#_Toc453441983)

[1.1. Опис предметної області 8](#_Toc453441984)

[1.2. Огляд аналогів 8](#_Toc453441985)

[1.3. Алгоритм опрацювання даних з мережі 12](#_Toc453441986)

[1.4. Висновок 13](#_Toc453441987)

[Розділ 2. Постановка задачі опрацювання і відображення спортивних подій і букмекерських ставок 15](#_Toc453441988)

[2.1. Загальна постановка задачі 15](#_Toc453441989)

[2.2. Вибір інструментальних засобів 15](#_Toc453441990)

[2.3. Специфікація вимог до програмного продукту 17](#_Toc453441991)

[2.4. Висновок 25](#_Toc453441992)

[Розділ 3. архітектурА і Проектування ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Для ОПРАЦЮВАННЯ та відображення спортивних подій і букмекерських ставок 26](#_Toc453441993)

[3.1. Архітектура програмного продукту. Опис впровадження архітектури MVP (Model View Presenter) в програмному продукті 26](#_Toc453441994)

[3.2. Проектування компонент та класів 27](#_Toc453441995)

[3.2.1 Проектування компонент та класів MVP (Model View Presenter) 27](#_Toc453441996)

[3.2.2 Проектування компонент та класів для взаємодії з мережею а також опрацювання спортивних подій та букмекерських ставок. 28](#_Toc453441997)

[3.3. Проектування алгоритмів 33](#_Toc453441998)

[3.4. Проектування графічного інтерфейсу користувача 35](#_Toc453441999)

[3.5. Висновок 38](#_Toc453442000)

[Розділ 4. Реалізація та тестування Програмного продукту 39](#_Toc453442001)

[4.1. Реалізація мобільного додатку 39](#_Toc453442002)

[4.2. Опис роботи з програмним продуктом 40](#_Toc453442003)

[4.3. Тестування програмного продукту 44](#_Toc453442004)

[4.4. Висновок 48](#_Toc453442005)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 49](#_Toc453442006)

[5.1. Економічна характеристика проектного рішення (програмного продукту) 49](#_Toc453442007)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару. 50](#_Toc453442008)

[5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ. 51](#_Toc453442009)

[5.4. Формування стратегічних альтернатив 54](#_Toc453442010)

[5.5. Бюджетування 58](#_Toc453442011)

[5.6. Остаточний вибір стратегії 61](#_Toc453442012)

[Висновки 62](#_Toc453442013)

[список Літератури 63](#_Toc453442014)

[Додаток А. Блок-схема алгоритму опрацювання спортивних подій та букмекерських ставок 64](#_Toc453442015)

[Додаток Б. Діаграма варіантів використання 65](#_Toc453442016)

[Додаток В. Діаграма діяльності 66](#_Toc453442017)

[Додаток Д. Звіт по тестуванню 67](#_Toc453442018)

[Додаток Е. Інструкція користувача 71](#_Toc453442019)

[Додаток Ж. Лістинг коду 73](#_Toc453442020)

[Додаток З. Статичний аналіз коду за допомогою Android Studio 82](#_Toc453442021)

[Додаток И. Список багів та зауважень від замовника 83](#_Toc453442022)

[Додаток К. Результати тестування першої версії перед релізом 86](#_Toc453442023)

# ВСТУП

Сьогодні люди мають велику потребу у пересуванні, чи це з особистих причин, чи з професійних. Мільйони людей подорожують кожен день, для цього існує багато видів транспорту, але найшвидший — авіатранспорт.

Пасажирські авіаперевезення – це найшвидший та найкомфортніший транспорт, адже можна подолати лічені години промайнувши кілька тисяч кілометрів. Різноманітні авіакомпанії надають цікаві послуги та сервіси в обслуговуванні клієнтів. Часто буває що клієнтові потрібно переглянути весь список авіарейсів, або переглянути стасус якогось рейсу, або замовити квиток не виходячи з домівки, для цього всього потрібно їхати в аеропорт, тратити свій час та запитувати все у асистентів.

Таким чином з’явилась необхідність у розробці програмного продукту для моніторингу та планування пасажирських авіаперевезень, який суттєво зекономить час клієнтів та дозволить здійснити усі вище згадані послуги не покинувши домівки.

# Розділ 1. Огляд програмного забезпечення для моніторингу та планування пасажирських авіаперевезень

## 1.1. Опис предметної області

Подорож — переміщення якоюсь певною територією з метою її вивчення, а також із загальноосвітньою, пізнавальною, спортивною цілями. Подорожі відіграють велику роль у житті людини, а з появою авіалайнерів вони стали набагато швидшими.

Загалом подорож літаком має такі переваги:

— Найважливішою перевагою подорожей повітряним транспортом є його висока швидкість пересування. Всього за кілька годин можна перетнути тисячі кілометрів і опинитися в потрібному вам місці. Особливо це зручно при обмеженому часі.

— Подорож літаком не здається такою нудною, як, наприклад, на поїзді або автобусі. Адже тут не потрібно декілька днів витрачати на поїздку.

— Літаки відрізняються високим сервісом. Пасажиру під час польоту надається їжа, напої. До того ж тут пропонуються всілякі способи, що дозволяють скоротати час, наприклад, подивитися фільм, ознайомитися з пресою або погортати журнал.

— Під час польоту пасажир відчуває себе максимально комфортно і зручно, що забезпечується зручними сидіннями, що дозволяють чудово відпочити і навіть поспати.

— Дуже вражає вид з ілюмінатора, красиві пейзажі, розташовані внизу або білосніжні хмари, що огортають літак.

— Сьогодні існує унікальна можливість забронювати квиток по телефону або Інтернету і забути про незліченні черги. Причому, доступна така послуга у будь-який час, найбільш зручний для вас.

— Літаком можна відправитися в будь-яку країну, куди не завжди можна дістатися залізничним або автомобільним транспортом.

Дивлячись на всі ці переваги нe дивно, що сьогодні люди в основному подорожують на літаку. Він поєднує в собі комфорт і швидкість, ви доберетеся до місця призначення дуже швидко. Без сумніву, подорож літаком – найзручніше і комфортний засіб пересування.

## 1.2. Огляд аналогів

Переглянувши декілька аналогів мною було вибрано три найяскравіші приклади:

WizzAir —  [угорсько-польська бюджетна авіакомпанія, із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)м серверним [веб-застосунок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA)ом для користування послугами компанії.

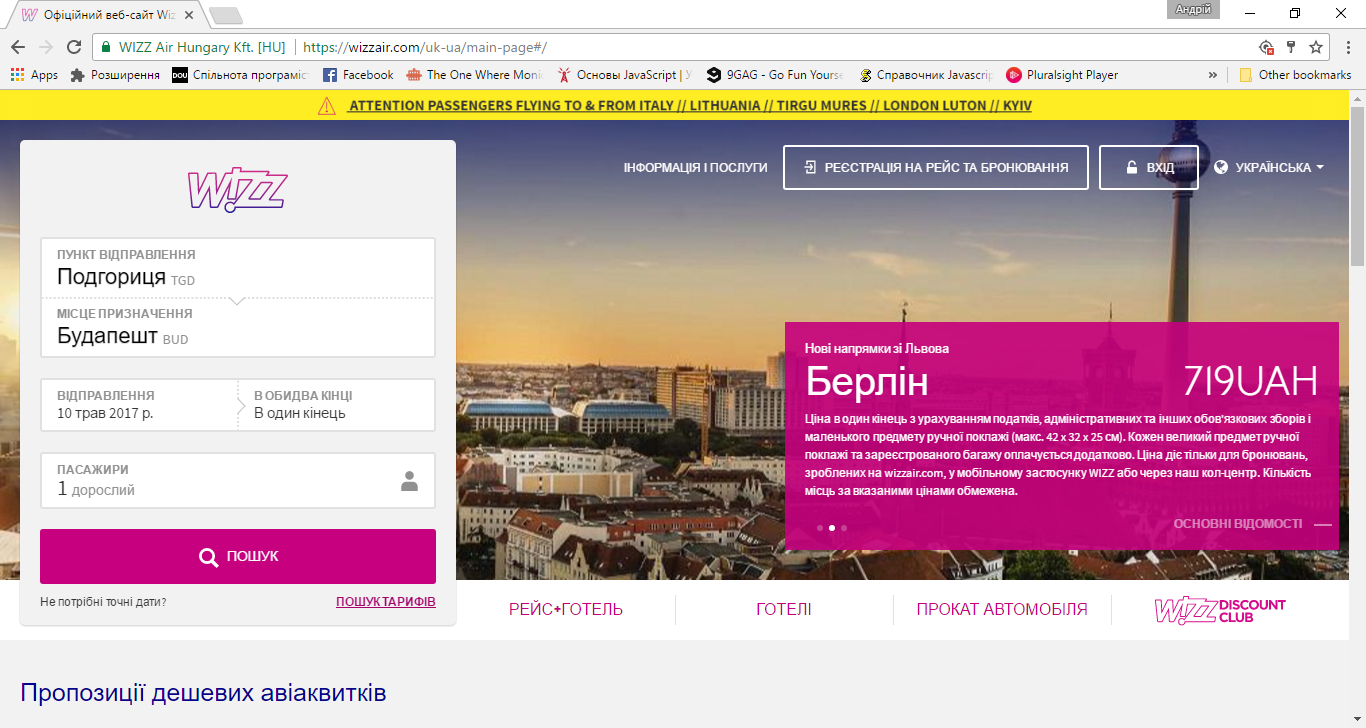


Рис. 1.1 Головна сторінка сервісу WizzAir

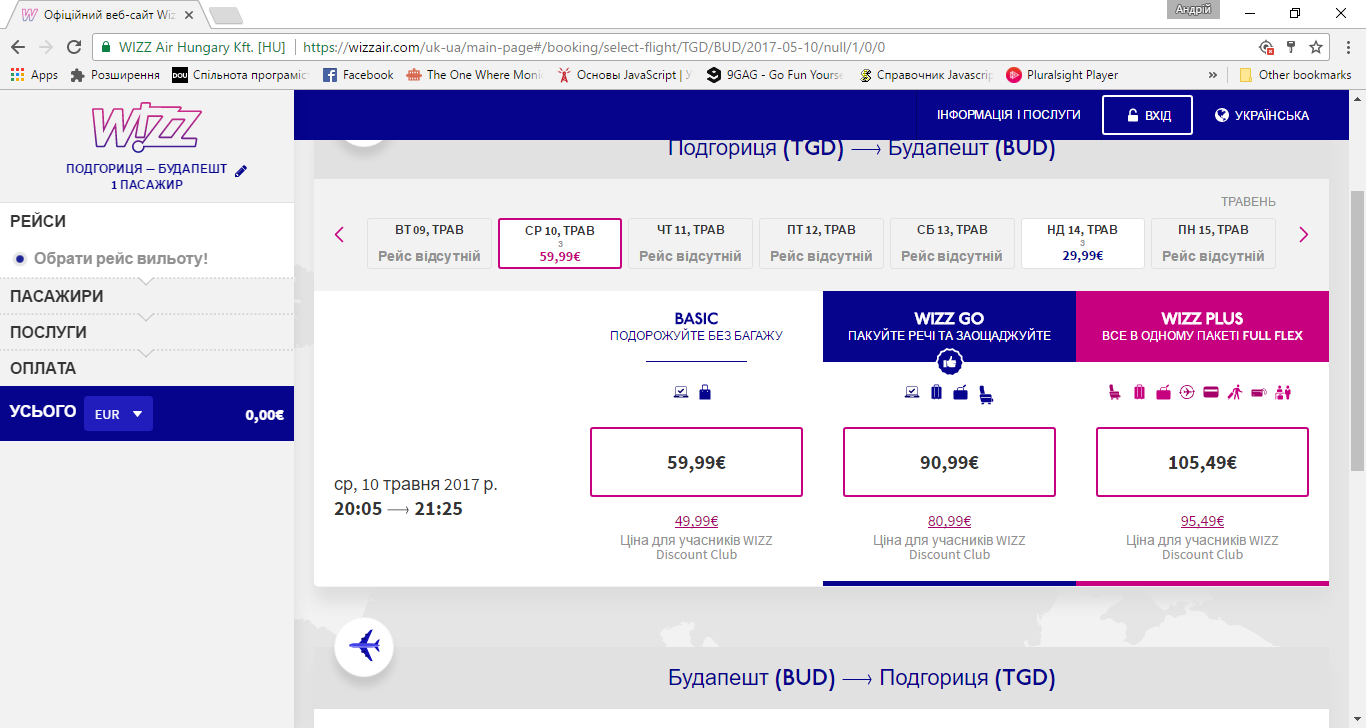


Рис. 1.2. Сторінка результату пошуку рейсів та подальшим замовленням сервісу WizzAir

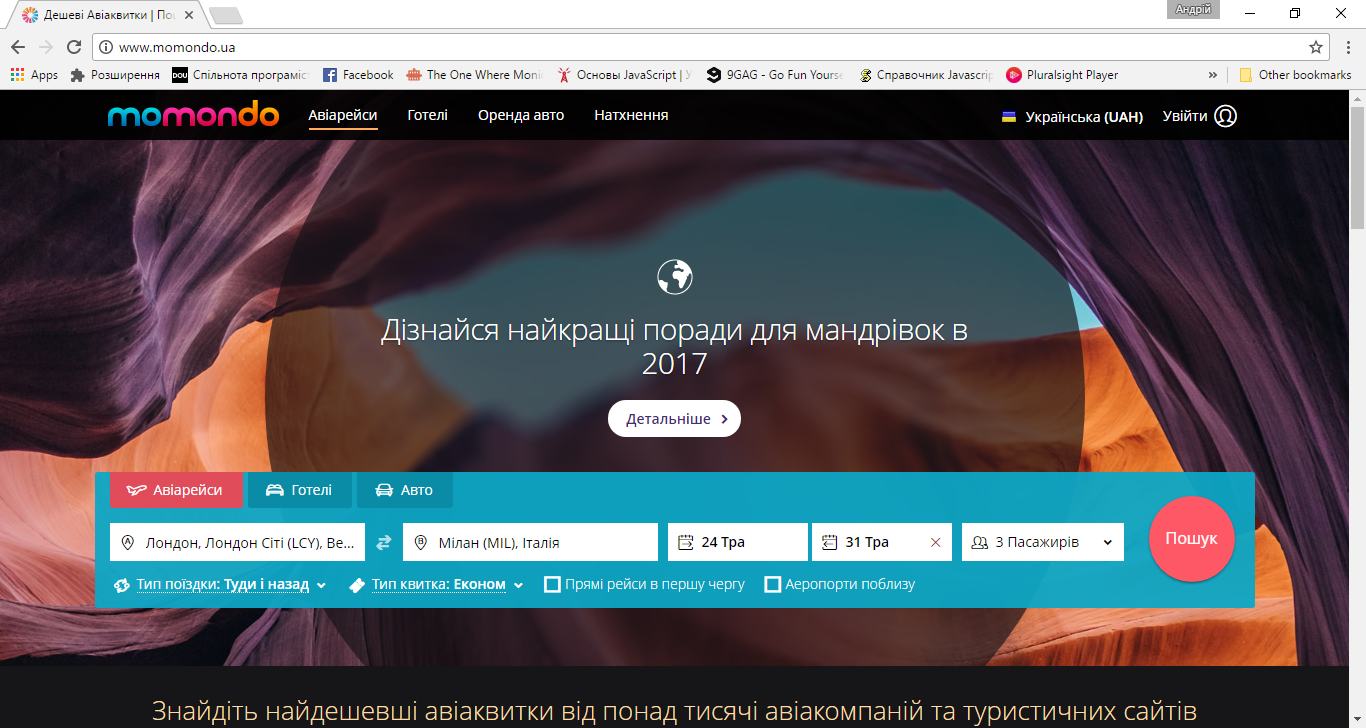
Переваги:

* Можливість додаткових послуг, таких як: бронювання готелів, виклик таксі та перегляд найблищих закладів харчування.
* Наявна українська локалізація
* Синхронізація із соціальними мережами
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Складний інтерфейс.
* Потребує багато кроків для купівлі або бронювання квитка.

Momondo — сервіс для пошуку та замовлення авіаквитків

Рис. 4.1.2.3 Головна сервісу Momondo

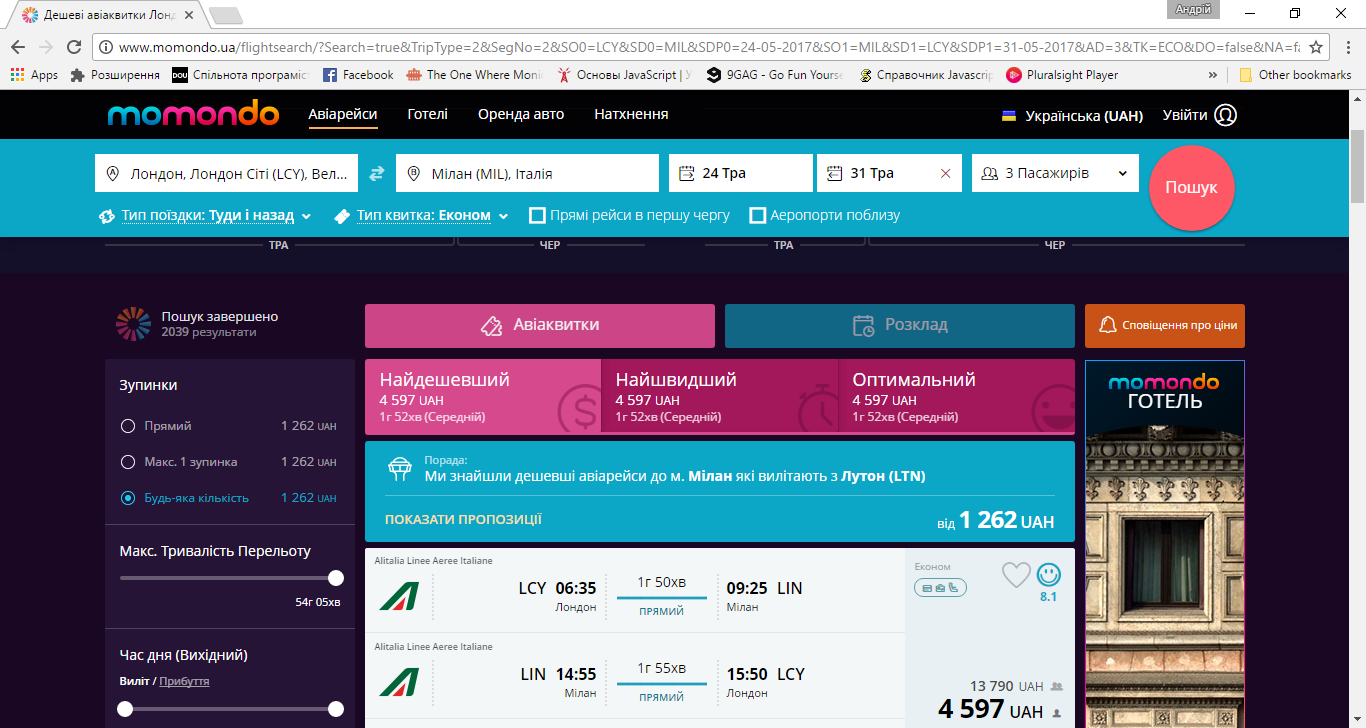


Рис. 4.1.2.4 Сторінка результату пошуку рейсів та подальшим замовленням сервісу Momondo

Переваги:

* Простота і зручність інтерфейсу
* Пропозиції найпопулярніших рейсів.
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Відслідковування статусу рейса.
* Присутня реклама

British Airways — найбільша авіакомпанія та національний авіаперевізник Великої Британії, одна з найбільших в Європі, [із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)м серверним [веб-застосунок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA)ом для користування послугами компанії.

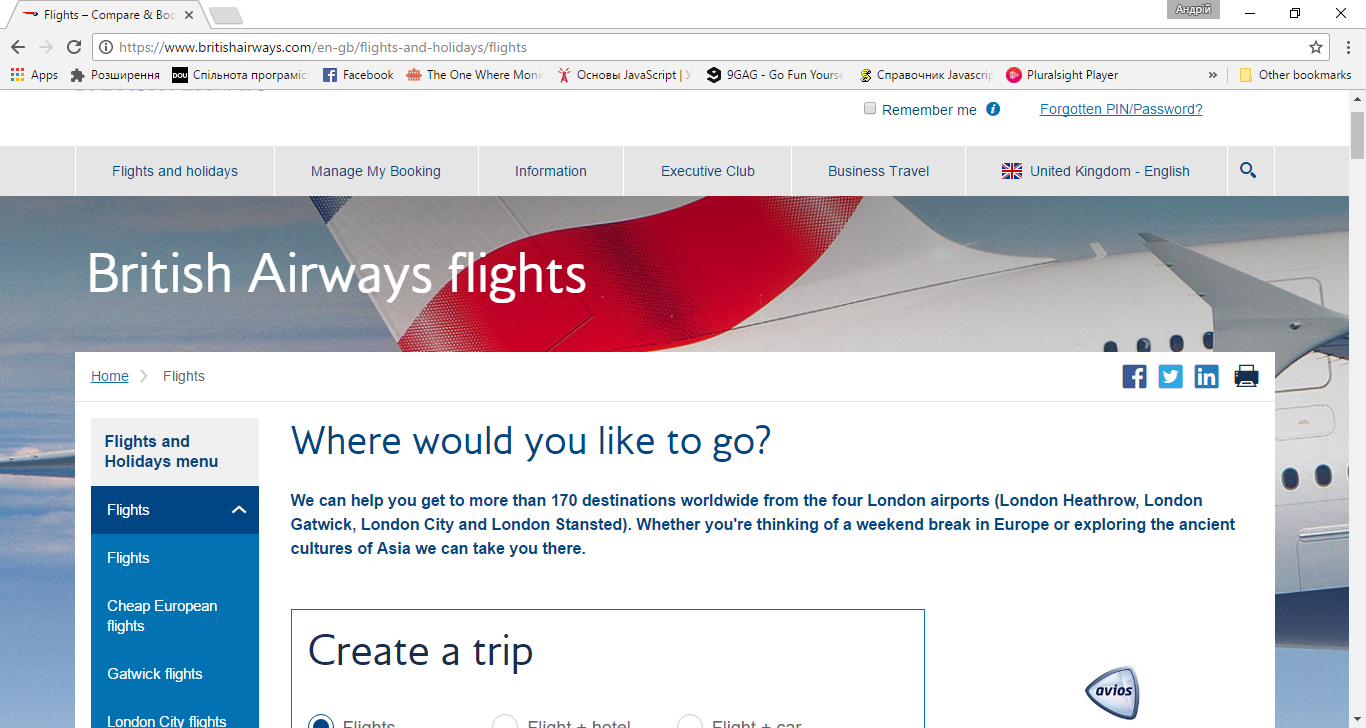


Рис. 4.1.2.5 Сторінка пошуку рейсів сервісу British Airways

Переваги:

* Простота і зручність інтерфейсу.
* Пропозиції найдешевших квитків.
* Відслідковування статусу рейса.
* Синхронізація із соціальними мережами
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Немає укрїнської локалізації

Таблиця 1.1

Порівняння функціоналу

|  |  |  |  |
| --- | --- | --- | --- |
| Функціонал | WizzAir | Momondo | British Airways |
| Відслідковування статусу рейса | - | + | + |
| Наявність інформації про конмпанію та авіаперевезення | + | + | + |
| Українська локалізація | + | + | - |
| Пропозиції додаткових послуг | + | + | + |
| Пропозації найпопулярніших рейсів | - | + | + |
| Присутність реклами | - | + | - |
| Синхронізація із соціальними мережами | + | - | + |

## 1.3. Алгоритм опрацювання даних з мережі

Однією з важливих задач при розробці клієнт- серверних застосунків є опрацювання даних з мережі. В більшості випадків це завдання є типовим і потребує автоматизації для економії часу розробки ПЗ. Досить часто це завдання виглядає приблизно так. Сформувати запит на сервер, передавши йому всі необхідні дані. Здійснити запит. Отримати дані в певному стандартизованому форматі, найчастіше XML або JSON. Проаналізувати ці дані та сформувати з них моделі даних з якими зручно працювати. Бібліотека Retrofit [4]. Параметри запиту передаються методам у вигляді параметрів методів, роблячи процес розробки зручним. Крім того, бібліотека Retrofit вміє також «парсити» популярні формати передачі даних, такі як XML, JSON. Недоліком цієї бібліотеки є погана можливість налаштувати її під свої потреби. Для того, щоб бібліотека коректно працювала з JSON форматом, його поля повинні бути заздалегідь заданими. Не всі методи API сервера відповідають цим вимогам, тому бібліотеку Retrfit не можна було використати на всі методи API. Інший варіант при роботі з мережею є класичний для Java клас – HttpConnection [5]. В такому разі процес обробки даних з мережі виглядає так. За допомогою класу AssyncTask я передаю параметри фоновому потоку. В фоновому потоці я створюю HttpConnection. Налаштовую його. Здійснюю запит та отримую результат у бінарному форматі. Потім формую із нього формат String і передаю його класу JsonTokener, який аналізує текстові дані у форматі JSON та формує з них об’єкт класу JsonObject. По суті JsonObject – це деревоподібна структура, яка складається з інших об’єктів JsonObject. Знаючи назви атрибутів, я отримую їх значення. На основі цих даних формую модель даних, яку вже потім опрацьовую. Кінцевий результат передаю головному потоку програми. Цей спосіб роботи з мережею є значно складнішим, але дозволяє здійснювати обробку будь якої структури даних у якої назви атрибутів є змінними.

## 1.4. Висновок

В результаті огляду програмного забезпечення моніторингу та планування пасажирських авіаперевезень було проаналізовано предметну область. Описано головні тенденції та потреби. Також проаналізовано декілька продуктів аналогів. В результаті огляду аналогів було виявлено переваги та недоліки, які будуть братись до уваги при розробці програмного забезпечення.

# Розділ 2. Постановка задачі опрацювання і відображення спортивних подій і букмекерських ставок

## 2.1. Загальна постановка задачі

У межах даної бакалаврської роботи було поставлено завдання розробити мобільний додаток для операційної системи Android, який представлятиме собою мобільну версію сайту tvevents.org. Мобільний додаток повинен отримувати та опрацьовувати (фільтрувати, групувати, впорядковувати) інформацію про спортивні події близько 30 видів спорту та інформацію букмекерських контор. Інформація отримуватиметься із сервера сайту в форматі JSON. Крім того, додаток повинен мати зручний графічний інтерфейс. Відповідно до цього було поставлено ряд задач:

* знайти та проаналізувати мобільні додатки на Google Play, які задовольняють хоча б деякі із цих вимог;
* проаналізувати технології та середовища розробки мобільних аплікацій під Android;
* розробити інтерфейс додатку згідно Android guidelines;
* проаналізувати алгоритми та бібліотеки для опрацювання даних які приходять з мережі в форматі JSON;
* продумати та спроектувати архітектуру системи. Вибрати хороший архітектурний шаблон;
* програмно реалізувати програмний продукт;
* провести тестування, системи;
* виконати реліз мобільного додатку для публікації в Google Play.

## 2.2. Вибір інструментальних засобів

Для того щоб розробляти додатки для ОС Android потрібно установити Android SDK – бібліотека класів та компонентів, як дозволяють взаємодіяти із елементами цієї операційної системи. Також треба установити Java, тому що Android SDK написаний на Java і загалом розробка ведеться на цій мові програмування.

Із відомих IDE (Integrated development environment) в мене було два можливих варіанти Android Studio та Eclipse. Я вибрав Android Studio, тому що вона на мою думку краще підходить для програмування на Android [6].

Android Studio – середовище розробки, створена компанією Google спеціально для розробки Android додатків. Середовище базується на Intelij IDEA – IDE розроблене компанією JetBrains, яке написане на Java, і спочатку створювалось для програмування на мові Java, але згодом воно стало підтримувати багато інших технологій, таких як JavaScript, PHP, Python та ін. Оскільки Android Studio написане на Java, то її можна використовувати на операційних системах Windows, Linux, MacOS.

Інтегрована інсталяція дозволяє автоматично завантажувати та установлювати сторонні бібліотеки для проекту. Для цього достатньо написати один рядок коду в налаштуваннях. Android Studio володіє системою автоматичного інспектування коду, що дозволяє знайти потенційні джерела помилок, оптимізувати код.

Для тестування Android додатку я використовував реальний пристрій (смартфон) і емулятор, що дозволяє запускати додаток на PC. Для цього можна використовувати стандартні емулятори, які є інтегровані в Android Studio, але я використав емулятор Genymotion. По-перше, він дозволяє встановити на емуляторі Google Services. По-друге, він менше використовує менше ресурсів і відповідно швидше працює.

Також для при розробці проекту я використав декілька сторонніх бібліотек, які дозволяють спростити процес програмування. ButterKnife – бібліотека яка дозволяє зв’язувати об’єкти із елементами View використовуючи просто анотації. Це спрощує написання коду. Він стає компактніший. Його стає легше підтримувати. Також я використав бібліотеку Picasso, яка спрощує процес завантаження графічних ресурсів як з мережі, так і з локального сховища. Вона також володіє автоматичним кешуванням.

## 2.3. Специфікація вимог до програмного продукту

1. * 1. **Вступ**
        1. Призначення, мета

Призначення даного продукту – дати можливість спортивним фанатам робити ставки на матчі вибираючи при цьому найвигідніші для цього букмекерські контори, а також отримати оригінальні прогнози, поточні результати і кінцеві результати подій для усіх основних видів спорту.

Мета – надати користувачу зручний спосіб отримання даних про події усіх основних видів спорту та спростити процес вибору букмекерської контори для здійснення ставки на результат спортивної події.

* + - 1. Продукти-аналоги
* Live Score;
* Goal Live Scores;
* Predictions Foot;
  + 1. **Загальний опис**
       1. **Характеристики продукту**

Функції, що будуть реалізовані у програмі:

* Вибрати вид спорту;
* Отримання інформації про всі спортивні події;
* Фільтр спортивних подій;
* Отримати детальну інформацію про подію;
* Отримання статистики;
* Здійснювати пошук спортивних подій;
* Перейти на сайт букмекера;
* Отримання інформації про вибрані спортивні події;
* Підписатись на події;
* Налаштувати додаток;
* Поділитись подією з друзями в Facebook.

Діаграма варіантів використання системи наведена в Додатку Б.

* + - 1. **Класи користувачів та їх характеристики**

В даній програмі передбачено лише один клас користувачів, тому функціонал не буде мати обмеження.

Мобільний додаток призначений для спортивних фанатів, щоб вони завжди були в курсі спортивних новин різних видів спорту, щоб спростити процес вибору найвигіднішої букмекерської контори для здійснення ставки на результат спортивної події.

* + - 1. **Середовище функціонування**

Апаратні вимоги:

* 20 Mb вільного простору на диску;
* Частота процесора 1 GHz;
* Оперативна пам’ять 512 Mb.

Системні вимоги:

* Операційна система Android 4.0.4 або вище;
* Доступ до мережі.
  + 1. **Характеристики системи**
       1. **Вибрати вид спорту**
          1. Опис і пріоритет

Завантаження та відображення всіх доступних видів спорту із можливістю вибрати поточний. Після цього користувач отримуватиме інформацію тільки про вибраний вид спорту. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувач вибирає вкладку "Види спорту". Відображається список із видами спорту. Користувач натискає на іконку із певним видом спорту. Програма зберігає вибраний вид спорту в локальній базі даних. На екрані завантажується головне вікно програми з інформацією про вибраний вид спорту.

* + - * 1. Функціональні вимоги

REQ-1.1: Завантаження списку доступних видів спорту;

REQ-1.2: Введення виду спорту;

REQ-1.3: Збереження виду спорту в налаштуваннях.

* + - 1. **Отримати інформацію про всі спортивні події**
         1. Опис і пріоритет

Користувач отримує список всіх подій для вибраного виду спорту на вибрану дату. Дані завантажуються та опрацьовуються програмою та виводяться користувачеві в зручному по групованому виді. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно "Explore". Програма завантажує та опрацьовує інформацію про спортивні події. Програма виводить дані користувачеві.

* + - * 1. Функціональні вимоги

REQ-2.1: Завантаження та опрацювання спортивних подій;

REQ-2.2: Вивід спортивних подій на екран користувачу.

* + - 1. **Фільтр спортивних подій**
         1. Опис і пріоритет

Завантаження та опрацювання тільки тих подій, які відповідають критеріям фільтру. Фільтрувати можна активні події, тобто ті події, які відбуваються в даний момент та події, для яких наявний спортивний прогноз від сайту tvevents.org. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно зі списком спортивних подій. Відкриває головне меню. Та ставить (забирає) прапорці на елементах меню: "Active matches", "Matches with predictions". Програма перезавантажує спортивні події, та відображає тільки ті, які відповідають вказаним критеріям.

* + - * 1. Функціональні вимоги

REQ-3.1: Вибір опції "Active Matches";

REQ-3.2: Вибір опції "Matches with predictions";

REQ-3.3: Завантаження, опрацювання та фільтрування списку спортивних подій;

REQ-3.4: Вивід отриманих спортивних подій користувачу.

* + - 1. **Отримати детальну інформацію про подію**
         1. Опис і пріоритет

Користувач може отримати детальну інформацію про спортивну подію. До неї може входити детальний рахунок по таймах, букмекерську інформацію з різних букмекерських контор та інше в залежності від виду спорту. Також деякі спортивні події містять спортивні прогнози. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувач вибирає зі списку конкретну подію. Програма завантажує інформацію про подію. Окремо завантажується букмекерська інформація. Користувачу відображається результат на екрані.

* + - * 1. Функціональні вимоги

REQ-4.1: Завантаження та опрацювання інформації про подію;

REQ-4.2: Завантаження та опрацювання даних букмекерських контор;

REQ-4.3: Отримання спортивного прогнозу;

REQ-4.4: Відображення даних користувачу;

REQ-4.5: Перейти на сайт букмекера.

* + - 1. **Отримання статистики**
         1. Опис і пріоритет

Користувачу надається можливість переглядати статистичні дані, рейтингові таблиці улюблених команд, та учасників спортивних ігор. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувача вибирає конкретну лігу. Програма завантажує дані спортивних учасників вибраної ліги. Також програма опрацьовує дані і на їх основі формує рейтингову таблицю із статистичними даними. Готова таблиця виводиться користувачу на екран.

* + - * 1. Функціональні вимоги

REQ-5.1: Опрацювання статистичної інформації;

REQ-5.2: Вивід таблиці з результатами на екран.

* + - 1. **Здійснювати пошук спортивних подій**
         1. Опис і пріоритет

Можливість пошуку, завдяки якій користувач зможе легко знайти спортивну подію. Для цього користувачу достатньо ввести фразу, яка наявна в інформації про спортивну подію. Наприклад частина назви ліги чи команди, якщо вибраний вид спорту - футбол, або назва гравця в крикет. Також користувач зможе здійснювати пошук для заданої дати. Пріоритет середній.

* + - * 1. Послідовності дія/відгук

Користувач натискає на кнопку пошуку. Відображається поле вводу тексту. Користувач вводить текст. В процесі вводу користувачем тексту, програма автоматично здійснює пошук фрази в кожній спортивній події, тобто виконується "живий" пошук. Із тих подій, які задовольняють критерії пошуку, формується новий список, який відображається користувачу.

* + - * 1. Функціональні вимоги

REQ-6.1: Введення пошукової фрази;

REQ-6.2: Перевірка спортивної події на наявність в ній пошукової фрази;

REQ-6.3: Формування списку подій, які задовольняють критерії пошук;

REQ-6.4: Виведення результату.

* + - 1. **Перейти на сайт букмекера**
         1. Опис і пріоритет

Користувач відкриває вікно, на якому відображається список букмекерських контор, завдяки якому користувач може перейти на сайт контори, яка йому інтересна. Пріоритет низький.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно зі списком букмекерських контор. Програма завантажує список букмекерських контор. Користувач натискає на елемент списку. Програма відкриває браузер, в якому завантажується веб-сайт вибраної букмекерської контори.

* + - * 1. Функціональні вимоги

REQ-7.1: Завантаження списку всіх букмекерських контор;

REQ-7.2: Вибір букмекерської контори;

REQ-7.3: Завантаження веб-сайту вибраної букмекерської контори.

* + - 1. **Отримання інформації про вибрані спортивні події**
         1. Опис і пріоритет

Користувач отримує інформацію тільки про ті команди, ліги та спортсменів, які представляють йому інтерес. Пріоритет високий.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно Feed з вибраними подіями. Програма завантажує та опрацьовує інформацію про спортивні події. Програма завантажує "Вибрані об'єкти" (команди, ліги, спортсмени, які користувач вказав як вибрані). Програма фільтрує тільки ті події, які відносяться до "Вибраних об'єктів". Програма відображає отримані події.

* + - * 1. Функціональні вимоги

REQ-8.1: Завантаження та опрацювання спортивних подій;

REQ-8.2: Фільтр спортивних подій за "Вибраними";

REQ-8.3: Вивід спортивних подій на екран користувачу.

* + - 1. **Підписатись на події**
         1. Опис і пріоритет

Користувач, якого інтересує окремі команди гравці, ліги або спортсмени (наприклад, бокс), можуть на них підписуватись. Після цього вони зможуть отримувати інформацію про спортивні події (включаючи сповіщення). Пріоритет середній.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно із усіма спортивними подіями. Програма завантажує та виводить список всіх подій. Користувач натискає на зірочку біля об'єкту спортивної події (команди, ліги, спортсмена). Програма зберігає "вибраний" об'єкт в локальній базі даних. Зірочка стає активна (або неактивна, якщо раніше була активна).

* + - * 1. Функціональні вимоги

REQ-9.1: Вибирати команди, ліги, які представляють інтерес;

REQ-9.2: Відписуватись від "Вибраних";

REQ-9.3: Зберігати вибрані об'єкти (команди, ліги, спортсмени);

REQ-9.4: Виводити "Вибрані" спортивні події, тобто ті, які стосуються вибраних об'єктів (команди, ліги, спортсмени);

REQ-9.5: Отримувати сповіщення про нові спортивні події.

* + - 1. **Налаштувати додаток**
         1. Опис і пріоритет

Додаток можна налаштувати. Є можливість налаштувати частому автоматичного оновлення даних, або взагалі відключити оновлення. Можна включити (відключити) автоматичні сповіщення. Пріоритет середній.

* + - * 1. Послідовності дія/відгук

Користувач відкриває вікно налаштувань. Користувач натискає на пункт "Автоматичні оновлення". Відкривається контекстний діалог. Користувач задає інтервал оновлення та натикає "Apply". Вказаний інтервал зберігається в локальних налаштуваннях. Користувач натискає на повзунок "Сповіщення". Повзунок переходить в протилежний стан. Його значення зберігається в локальних налаштуваннях.

* + - * 1. Функціональні вимоги

REQ-10.1: Вмикання та вимикання автоматичних оновлень даних з мережі.

REQ-10.2: Вмикання та вимикання автоматичних сповіщень про події.

REQ-10.3: Налаштування частоти автоматичного оновлення даних з мережі та сповіщень;

REQ-10.4: Очистка кешу.

REQ-10.2: Налаштування сповіщень.

* + - 1. **Поділитись подією з друзями в соціальних мережах**
         1. Опис і пріоритет

Користувач може "поділитись" подією з друзями в соціальних мережах. Для цього він публікує інформацію про спортивну подію себе на сторінці в соціальній мережі, автоматично відправляє її другові в повідомленні, чи на електронну пошту, SMS. Пріоритет низький.

* + - * 1. Послідовності дія/відгук

Користувач вибирає подію та натискає кнопку "Поділитись". Програма отримує інформацію про спортивну подію та формує з неї публікацію. Програма публікує спортивну подію в соціальній мережі.

* + - * 1. Функціональні вимоги

REQ-11.1: Запропонувати програму друзям відправивши інформацію про програму в соціальну мережу, e-mail, SMS та ін.

REQ-11.2: Надіслати текст з інформацією про спортивну подію в соціальну мережу, e-mail, SMS та ін.

* + 1. **Вимоги зовнішніх інтерфейсів**
       1. **Користувацькі інтерфейси**
* Користувацький інтерфейс повинен бути розроблений відповідно до «Android Human Interface Guidelines» [7].
  + - 1. **Програмні інтерфейси**
* Android SDK;
* Бібліотека Retrofit;
* Бібліотека ButterKnife.
  + 1. **Інші нефункціональні вимоги**
       1. **Вимоги продуктивності**

Швидкість завантаження та опрацювання даних буде прямо залежати від швидкості передачі даних в мережі та від кількості даних, які отримуються з мережі.

* + - 1. **Вимоги безпеки**

Немає вимог безпеки.

* + - 1. **Атрибути якості програмного продукту**

В програмному продукті реалізований мінімальний набір функцій необхідний для зручного використання. Користувацький інтерфейс проектувався у відповідності до стандартів Android, тому він повинен бути зрозумілий для користувачів цієї операційної системи [5].

Програмний продукт використовує ресурси мережі. Це може бути причиною уповільнення завантаження даних з мережі іншими програмами.

* + - 1. **Інші вимоги**

Необхідний дозвіл на можливість працювати з файловою системою для створення файлів з локальними даними.

## 2.4. Висновок

Впродовж постановки задачі опрацювання і відображення спортивних подій і букмекерських ставок було сформовано завдання від замовника розробити мобільний додаток, який представлятиме інформацію із сайту tvevents.org. Для реалізації програми вибрано середовище розробки Andorid Studio. Для тестування системи я вирішив використовувати Android емулятор на базі Genymotion, адже завдяки цьому програма можна тестувати на різних версіях операційної системи та віртуальних пристроях. Крім того розроблена специфікація вимог, в які вирішено який набір функціоналу буде реалізовано в додатку, а саме вибір виду спорту, можливість отримати інформацію про всі події вибраного виду спорту, фільтрувати спортивні події, пошук, отримання статистики ліг та змагань, можливість підписуватись на команди, ліги, гравців, та налаштувати додаток. Програма повинна бути зручною в користуванні та відповідати типовим правилам розробки мобільних додатків для операційної системи Android.

# Розділ 3. архітектурА і Проектування ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Для ОПРАЦЮВАННЯ та відображення спортивних подій і букмекерських ставок

## 3.1. Архітектура програмного продукту. Опис впровадження архітектури MVP (Model View Presenter) в програмному продукті

Програма розділяється на три основні компоненти Model View Presenter [8]. Тут важливо не плутати View з однойменним класом – базовим класом всіх графічних елементів. В MVP (далі буде йтися про MVP під ОС Android) View – це всі класи Activity та Fragment. Для цього для кожного з них створюється інтерфейс View. Наприклад, класу MainActivity ми створимо інтерфейс, MainView. Тоді MainActivity буде реалізовувати інтерфейс MainView. В цьому інтерфейсі будуть описуватись всі методи, які здійснюватимуть передачу даних на користувацький інтерфейс. Важливо, щоб в класі MainActivity не була присутня бізнес логіка, та логіка опрацювання вводу та виводу. Для цього будуть створені класи MainPresenter та MainModel. MainPresenter не повинен знати про MainActivity, а взаємодіяти з ним через його інтерфейс MainView. MainActivity матиме посилання на MainPresenter і надсилатиме є йому дані та команди, які приходитимуть з графічного інтерфейсу. Презентер отримуватиме дані, опрацьовуватиме їх. У разі помилок відправлятиме відображенню повідомлення про це, використовуючи його інтерфейс MainView. Якщо дані не містять помилок вводу та виводу MainPresenter надсилатиме дані MainModel. MainModel нічого не знає про існування View, Presenter. Вона повністю незалежна. MainPresenter отримуватиме від неї дані через методи зворотного виклику, тобто через інтерфейс Listener. Тут застосований дизайн патерн Observer.

MainModel після отримання вхідних даних здійснюватиме їхнє опрацювання. Компоненти Model також взаємодіють з різними сервісами такими як, наприклад, Retrofit, SQLite база даних, та іншими, абстрагуючи при цьому логіку роботи із сервісами. На рівні Presenter взагалі не важливо звідки дані будуть отримуватись і куди вони передаватимуться. Важливо тільки те, що ми передаємо якісь дані моделі і через інтерфейс Listener отримуємо дані у випадку успіху чи повідомлення про помилку у випадку помилки.

На рис. 3.1. наведена діаграма класів, яка демонструє реалізацію архітектури Model View Presenter в програмному продукті на прикладі головного фрагменту, який отримує список спортивних подій.



Рис. 3.1. Діаграма класів архітектури Model View Presenter.

## 3.2. Проектування компонент та класів

## 3.2.1 Проектування компонент та класів MVP (Model View Presenter)

Головними пакетами в проекті виступають data, ui, utils, widgets [9]. Весь список пакетів зображено на рис. 3.2.

Data – пакет, який містить класи та інтерфейси призначені для роботи з моделями даних. Тут також містяться класи завантаження даних з мережі, класи опрацювання даних та сервіси.

Ui – пакет який містить класи для роботи з графічним інтерфейсом користувача. Ділиться на під пакети, які містять класи та інтерфейси для конкретного вікна або фрагмента. Оскільки в проекті використано архітектуру Model View Presenter, то структура під пакета така: інтерфейс відображення (View), клас активності або фрагменту, який реалізовує даний інтерфейс, класи Presenter, моделі та інтерфейс Listener, за допомогою якого здійснюється передача даних з моделі в presenter.

Utils – пакет, який містить допоміжні класи, такі як конвертор дати та інші.

Widgets – пакет, який містить реалізовані мною елементи користувацького інтерфейсу.

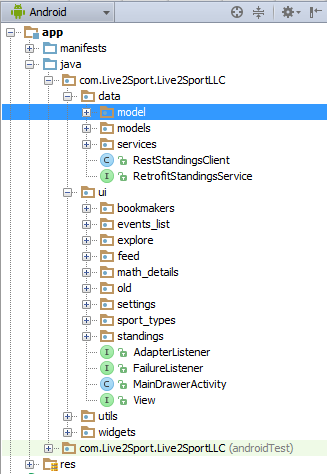


Рис. 3.2. Структура пакетів мобільного додатку

## 3.2.2 Проектування компонент та класів для взаємодії з мережею а також опрацювання спортивних подій та букмекерських ставок.

Головними класами в проекті виступають FetchItemsTask, UpdateService, TvEventsFetchr, MatchLab, Match, League, Bookmaker, інтерфейс RetrofitStandingsService. На рис. 3.3. зображені всі класи проекту.

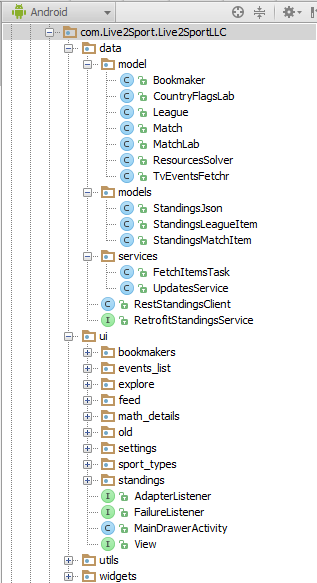


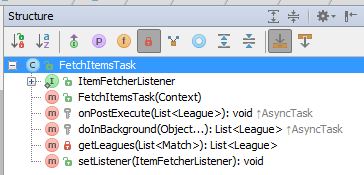
Рис. 3.3. Головні класи мобільного додатку

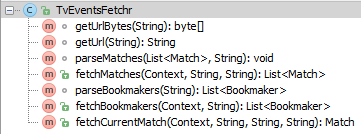
* FetchItemsTask – клас який відповідає за асинхронне опрацювання даних. Оскільки завантаження даних з серверу та їх подальше опрацювання – ресурсоємний процес, то виконувати його в головному потоці програми недоцільно. Для цього створюється об’єкт класу FetchItemsTask, який приймає вхідні дані, створює фоновий потік, в якому виконує завантаження та опрацювання даних сервера та повертає результат головному потоку програми.
* RetrofitStandingsService – головний інтерфейс підсистеми Retrofit, який містить методи для роботи з REST запитами сервера, за допомогою яких здійснюється запит до сервера із необхідними параметрами, які вказуються як аргументи методів.
* UpdateService – служба призначена для отримання даних з сервера в фоновому режимі. Користувач може закрити додаток, але служба продовжуватиме працювати, надсилаючи сповіщення про події, на які підписався користувач.
* TvEventsFetchr – клас, який містить методи для завантаження даних з мережі а також здійснює «парсинг» подій з формату JSON в користувацькі моделі даних.
* MatchLab – клас призначений для опрацювання спортивних подій, які представлені в форматі списку. Формує з нього структуру, фільтрує та впорядковує їх відповідно до вимог.
* Match – головна модель спортивної події, містить всю необхідну інформацію.
* League –модель «ліги» або групи подій, які об’єднані. Містить в собі впорядкований список подій.
* Bookmaker – модель букмекера. Містить назву букмекера, інформацію про букмекерські ставки певної спортивної події, посилання на сайт букмекера та відео трансляцію.



Рис. 3.4. Діаграма класів

В програмі реалізовані наступні методи:







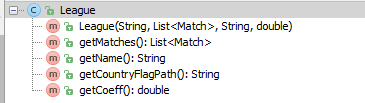


Рис. 3.5. Представлення методів

Опис основних методів:

* 1. getUrlBytes – за вказаним запитом отримує дані з мережі в бінарному форматі.
  2. getUrl – повертає текстову інформацію з бінарного представлення результату запиту.
  3. parseMatches - отримує модель списку спортивних подій із текстових даних у форматі JSON.
  4. parseBookmakers – на вхід отримує список дані букмекерських контор в форматі JSON. Здійснює «парсинг» та повертає на список букмекерів із їхньою інформацією про спортивну події.
  5. groupMatchesByLeagues – на вхід отримує список спортивних подій. Здійснює їхнє опрацювання, тобто групує по лігах, впорядковує ліги та спортивні події всередині ліг відповідно до їхньої «важливості». Також фільтрує спортивні події в залежності від налаштувань додатку.

## 3.3. Проектування алгоритмів

***Проектування алгоритму завантаження та парсингу даних з мережі***

Завантаження і опрацювання даних – ресурсоємний процес, тому недоцільно його виконувати в головному потоці програми (крім того система Android це не дозволяє). Тому ця робота виконується в фоновому потоці. Для цього Android SDK забезпечує клас AssyncTask [10]. Суть полягає в тому, що ми в головному потоці передаємо об’єкту цього класу параметри, такі як дата, вид спорту, атрибути, за якими ми фільтруватимемо дані. Потім ми перезначаємо метод doInBackground – метод, який вже виконуватиметься в окремому потоці. Він отримує з головного потоку дані і після цього виконує нижче описані дії (із фонового потоку не можна доступатись до елементів графічного інтерфейсу).

1. Формування та відправлення запиту на сервер за допомогою за допомогою об’єкту класу HttpURLConnection;
2. Отримання результату в бінарному вигляді та конвертування його в текстовий (текст буде в форматі JSON);
3. За допомогою JsonTokener отримання JSONObject із текстових даних;
4. Формування моделі списку спортивних подій із JSONObject. JSONObject являє собою об’єкт даних, який має деревовидну структуру. Він має в собі дочірні об’єкти класу JSONObject або JSONArray, який представляє собою масив інших JSONObject. Знаючи структуру JSON я отримую з нього необхідні значення об’єктів та зберігаю їх окремому списку.
5. Здійснюється опрацювання списку з даними (оскільки описується загальний принцип завантаження та парсингу даних з мережі, то це стосується і спортивних подій і букмекерів).
6. Сформовавши таким чином готову модель фоновий потік передає дані головному після чого дані відображаються на графічному інтерфейсу користувача.

***Проектування алгоритму опрацювання спортивних подій та букмекерських ставок***

Блок схема алгоритму подана в додатку А.

На вхід отримуємо список неопрацьованих спортивних подій, тобто той, що отримувався в результаті «парсингу» JSON даних, які прийшли з мережі. Після цього дані необхідно потрібним чином опрацювати сформувавши з них структуру, зручну для представлення користувачу.

Послідовність кроків алгоритму:

1. Профільтрувати події за певними атрибутами.
   1. Активні спортивні події – ті, що йдуть в даний момент (матчі, поєдинки, змагання)
   2. Події з прогнозуваннями. Для деяких подій сервер надає можливість переглянути конкретний прогноз.
   3. Події, які представляють інтерес користувача, тобто ті, на які він підписаний. Наприклад, якщо це футбол, то користувач може бути підписаний на лігу чи конкретну команду.
   4. Особливо, в яких трапився цікавий момент, наприклад, у футболі це може бути нещодавній гол.
2. Після цього події групуються за певними категоріями, наприклад, футбольні матчі групуються за лігами, до яких вони належать.
3. Всередині певної категорії події впорядковуються в порядку їхньої важливості.
   1. Спочатку йдуть активні матчі, потім неактивні, але нерозпочаті, вкінці завершені.
   2. Всередині кожної з вище описаної події також впорядковуються в порядку важливості, як серед активних, так і серед неактивних нерозпочатих і завершених.

## 3.4. Проектування графічного інтерфейсу користувача

Розробка прототипу здійснювалась за допомогою сервісу uxpin [11].

Головне вікно програми містить випадаюче меню, яке містить такі переходи: Feed, Explore, Sport types, Bookmakers, Settings (рис. 3.6.).

Feed – тут відображаються матчі поточного виду спорту, класифіковані по лігах, які користувач виділив як «Вибране» (рис. 3.7.). Якщо вікно Feed порожнє, то воно матиме посилання на вікно Explore. Натиснувши на лігу ми переходитимемо на вікно Standings – турнірна таблиця яка відображає рейтингову таблицю команд в Лізі з детальною інформацією. Натиснувши на матч ми переходитимемо в вікно Mathc details. Користувач має змогу вибирати дату на яку він бажає переглянути матчі за допомогою календаря або пересуваючи горизонтально пальцем по екрану. Також є пошук по назві ліги чи команди. Є також меню, яке дозволяє фільтрувати матчі тільки по тими, які йдуть зараз, по «Prediction» (наявність коефіцієнту виграшу) а також оновлювати дані (дані можуть автоматично оновлюватись, задавши відповідну опцію в налаштуваннях).

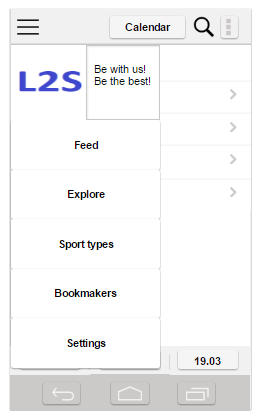


Рис. 3.6. Прототип головного вікна з випадаючим меню

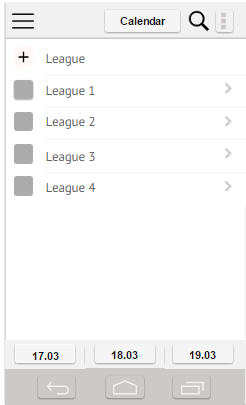


Рис. 3.7. Вікно Feed

Explore – відображаються всі ліги, або країни, які містять матчі на поточну дату (рис. 3.8.). Натиснувши на лігу вона анімаційно розгортається, відображаючи матчі. Також наявні «зірочки» біля кожної ліги та команди, які дозволяють користувачу занести її в «Вибране». В головній панелі є «зірочка», яка дозволяє відобразити тільки вибрані матчі з можливістю видалити з вибраних. Також наявні опції такі як і в Feed.

Sport types – відображається список видів спорту у вигляді «карток» із картинками та надписами (рис. 3.9.). Якщо користувач хоче отримувати інформацію про певний вид спорту він натискає на картку з цим видом спорту і програма автоматично завантажує інформацію про нього.

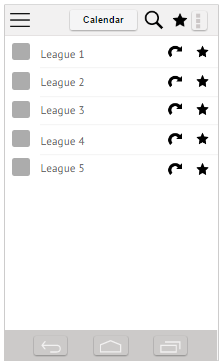


Рис. 3.8. Вікно Explore

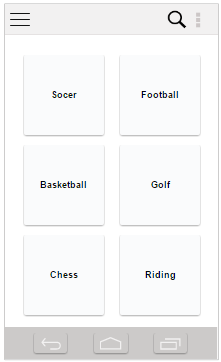


Рис. 3.9. Вікно Sport types

Bookmakers – список букмекерських контор, із посиланнями на їхні сайти (рис. 3.10.). Користувач натискає на елемент списку з певним букмекером. Після цього програма переходить в WEB – браузер, в якому завантажується сайт цієї контори.

Settings – налаштування додатку (рис. 3.11.). В ньому користувач зможе налаштовувати автоматичні оновлення даних, налаштовувати сповіщення, ділитись інформацією про додаток в Facebook, ставити рейтинг додатку на GooglePlay.

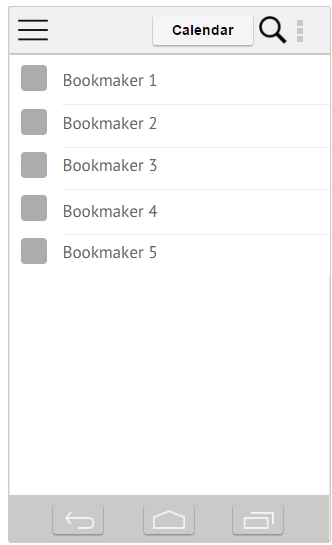


Рис. 3.10. Вікно Bookmakers

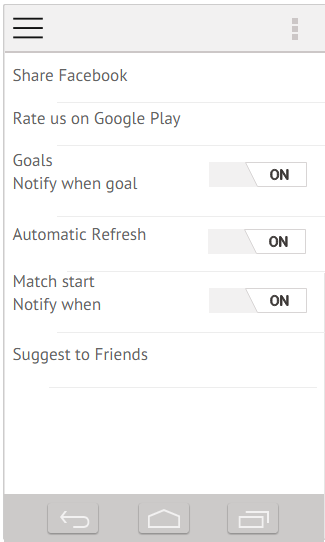


Рис. 3.11. Вікно Settings

Mathc details – вікно деталей матчу із вказанням ліги, матчу, часу, детального рахунку по тайму, «Prediction» (при наявності, прогноз виграшу команди із коефіцієнтом від сайту tvevents.org), список сайтів букмекерів, із інформацією про цей матч та посиланням (при наявності) на онлайн трансляції (рис. 3.12.).

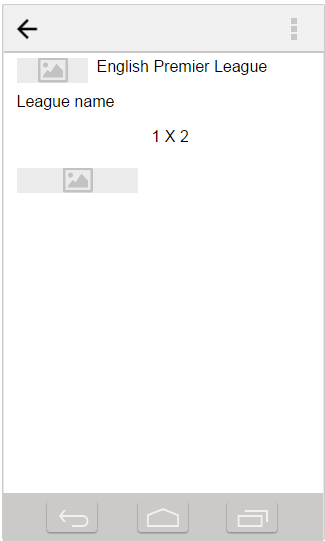


Рис. 3.12. Вікно з детальною інформацією про спортивну подію.

## 3.5. Висновок

В результаті виконання 3 розділу бакалаврської кваліфікаційної роботи спроектована архітектура мобільного додатку. В якості архітектурного шаблону вибрано Model View Presenter. На рівні Presenter не потрібно слідкувати за життєвим циклом активностей та фрагментів, адже цей рівень від нього не залежить. Це спрощує процес опрацювання даних. Спрощується процес модульного тестування. Також спроектована ієрархія пакетів та класів, які розділяються на чотири основні пакети: data, ui, utils, widgets. Визначені основні класи системи. Які відповідають за отримання та обробку даних. Спроектовано алгоритм завантаження та парсингу даних за мережі а також алгоритм опрацювання спортивних подій та букмекерських ставок, що опрацьовує дані, які приходять із сервера. Також створений прототип інтерфейсу системи за допомогою сервісу uxpin.

# Розділ 4. Реалізація та тестування Програмного продукту

## 4.1. Реалізація мобільного додатку

Для контролю версій було створено репозиторій на bitbucket. В ньому я розмістив папку з проектом. Список останніх комітів із їхнім описом та датою створення наведено на рис. 4.1.

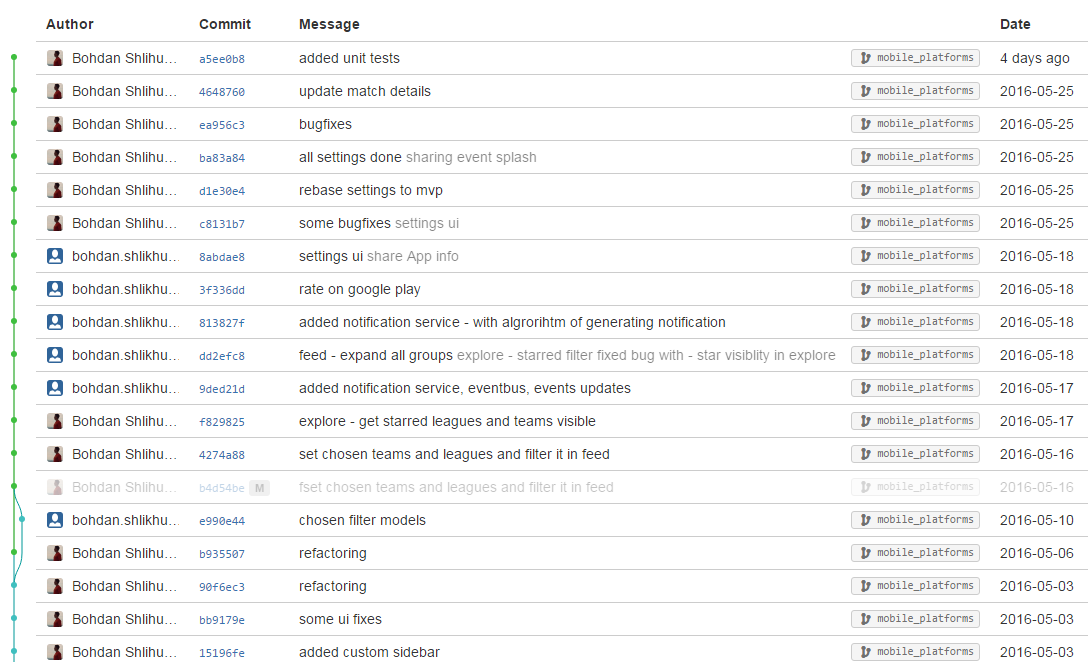


Рис. 4.1. Список комітів на репозиторії bitbucket.

Блок-схема алгоритму опрацювання спортивних подій та букмекерських ставок наведено в додатку А.

Вихідні коди програми із детальними поясненнями та коментарями наведені в додатку Ж.

Завдання виконувались ітераційно завжди добавлялись нові побажання та зауваження від замовника. Список завдання та їхні статуси зображено на рис. 4.2.

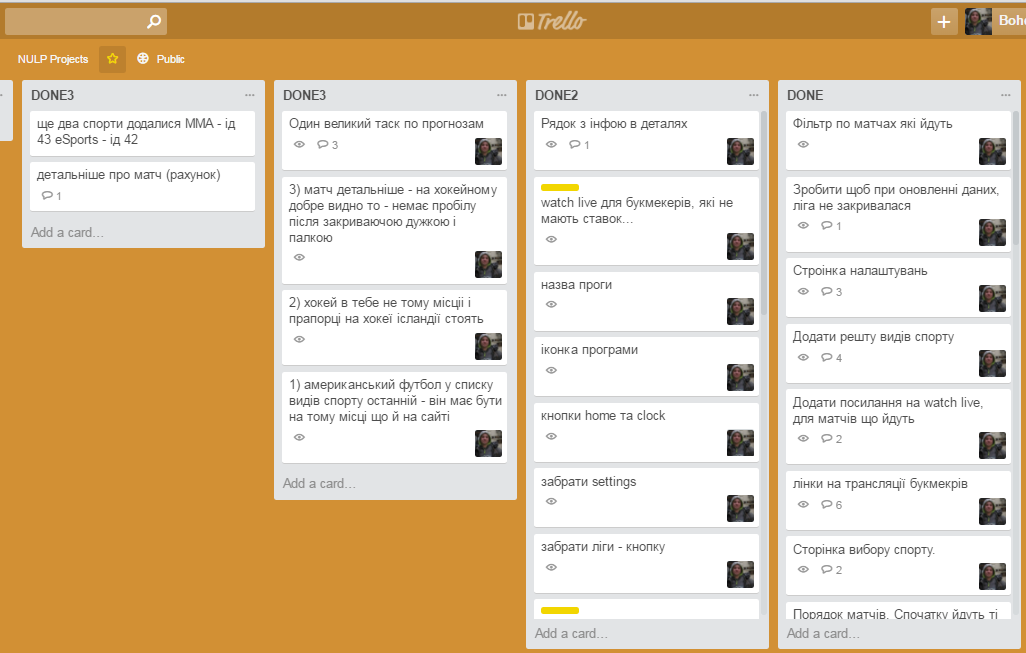


Рис. 4.2. Спискок завдань для виконання на сайті Trello.

## 4.2. Опис роботи з програмним продуктом

Спочатку користувачу необхідно запустити додаток. Для цього він знаходить додаток Live2Sport в списку лаунчера Android та запускає. Після цього перед користувачем з’являється стартове вікно додатку із логотипом. Потім появляється головне вікно додатку та бічна панель користувача. В ній міститься список із основними переходами, які може здійснювати користувач (рис. 4.3.). Також за замовчуванням завантажується вікно Feed – «вибрані спортивні події» (рис. 4.5.). При першому запуску програми цей список порожній, на його місці повідомлення, яке каже користувачі що слід перейти в вікно Explore.

Користувач переходить в вікно Explore (рис. 4.4.), яке схоже до вікна Feed (зручний пошук, фільтри, вибір дати, оновлення даних), але тут відображаються всі події на задану дату для вибраного виду спорту, а також їх можна відзначати як вибрані, та фільтрувати вибрані події. Крім того за замовчуванням групи подій є згорнутими, що спрощує навігацію коли список дуже великий. Користувач переглядає події та відзначає спортивні команди, ліги, спортивних гравців як «Вибрані».

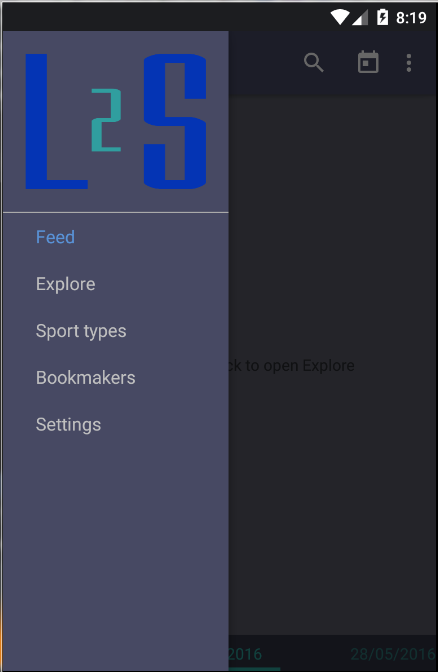


Рис. 4.3. Бічна панель із основними переходами.

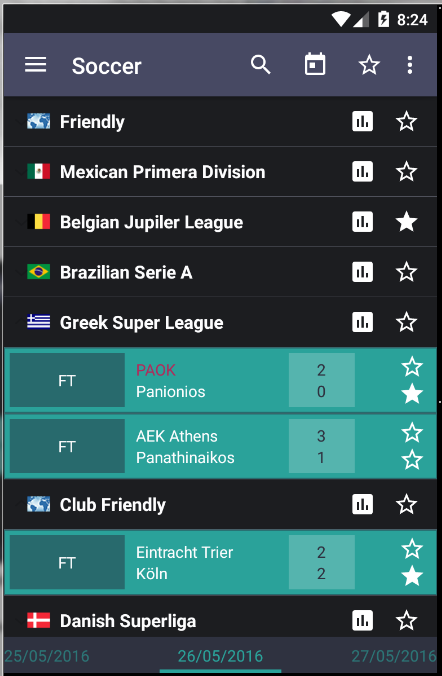


Рис. 4.4. Список всіх спортивних подій вибраного виду спорту.

Після цього користувач переходить назад в Feed (рис. 4.5.), де відображатиметься список «вибраних спортивних подій» вибраного виду спорту на задану дату, тобто ті, які відносяться до вибраних ліг, турнірів, команд, чи учасників. Наявний зручний пошук події за пошуковою фразою, фільтри (фільтрувати події які відбуваються в даний момент, фільтрувати події, які містять прогнози отримані із сайту tvevents.org), вибір дати, оновлення даних з сервера.

В вікні Feed (або в Explore) користувач вибирає певну подію (натискає на неї). Після цього відображається детальна інформація про цю подію (рис. 4.6.). Якщо це футбольний матч (програма підтримує більше тридцяти видів спорту), то там завантажується дата початку матчу, поточний час, детальний рахунок по таймах, кінцевий рахунок, ліга, до якої відноситься матч. Список букмекерських контор, які мають букмекерські коефіцієнти, що дозволяє користувачу визначити де саме найвигідніше здійснювати ставку. Натиснувши на логотип букмекера, користувач переходить на його сайт. Також деякі букмекери мають відео трансляції матчів (Watch Live). Користувач може оновити дані та поділитись подією з друзями в соціальних мережах.

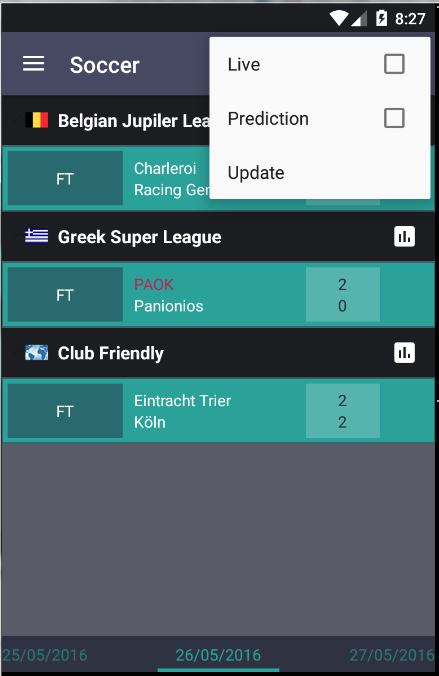


Рис. 4.5. Список «вибраних спортивних подій»

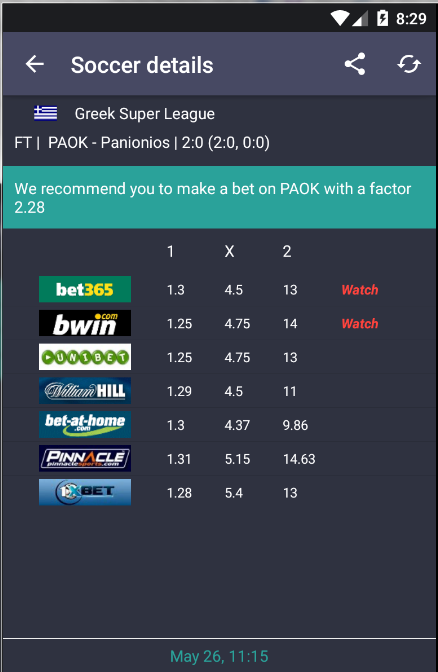


Рис. 4.6. Детальна інформація про спортивну подію

В вікнах Explore, Feed користувач може відкрити турнірну таблицю ліги (рис. 4.7.) натиснувши на неї в списку. Тут отримуються результати для певного турніру, наприклад, футбольної ліги.

Додаток за замовчуванням працює зі спортивними подіями футболу. Для того, щоб це змінити треба зайти в вікно Sport types та вибрати вид спорту (рис. 4.8.). Після перезавантаження програми завантажиться інформація вибраного виду спорту.

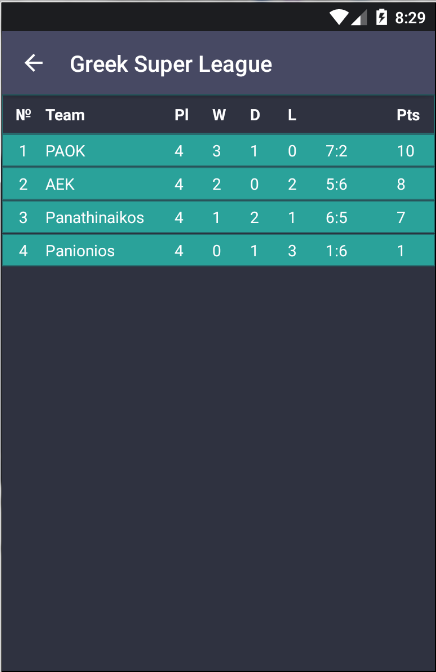


Рис. 4.7. Турнірна таблиця ліги

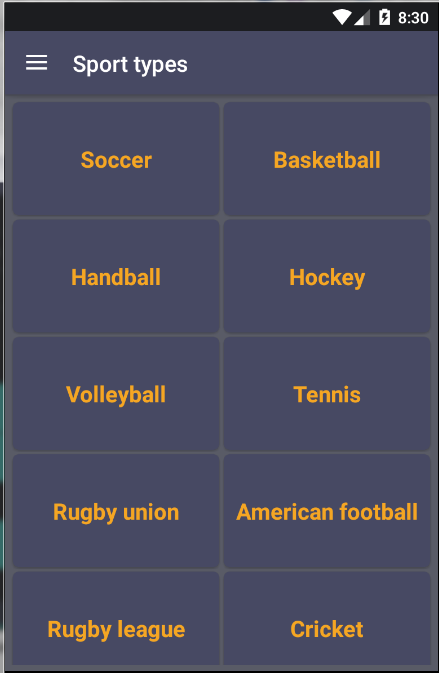


Рис. 4.8. Вікно з вибором виду спорту.

Також в бічній панелі є перехід на список, із яких програма отримує дані. Є можливість переходу на їхні сайти (рис. 4.9.).

Користувач може налаштувати мобільний додаток під свої потреби. Для цього в бічній панелі він вибирає відповідний пункт, після цього завантажується вікно налаштувань (рис. 4.10.). Користувач може включити чи відключити автоматичні оновлення – регулярне автоматичне оновлення даних із сервера із перезавантаженням вмісту вікон Feed, Explore, деталей матчу. Включити чи відключити автоматичні сповіщення, які відсилатимуться користувачу у випадку таких подій як – команда забила гол (в футболі). Можна також налаштувати частоту оновлень та сповіщень. Користувач може очистити локальні дані (список «вибраного») а також проголосувати за додаток на Google Play та запропонувати його друзям.

Інструкція користувача наведене в Додатку Е.



Рис. 4.9. Вікно зі списком букмекерських контор.

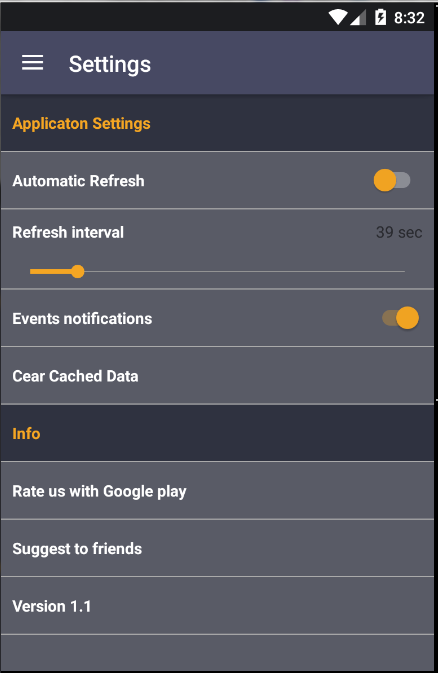


Рис. 4.10. Вікно налаштувань мобільного додатку.

## 4.3. Тестування програмного продукту

**4.3.1. Вступ.**

В цьому розділі наведені результати тестування мобільного додатку. Також показано оцінку якості. Було здійснено функціональне тестування готового мобільного додатку.

**4.3.2. Розробка тестів.**

Під час фази тестування було розроблено тести функціонального тестування, а також проведено статистичний аналіз коду за допомогою Android Studio.

**4.3.3. Функціональне тестування**

4.3.3.1. Результати тестування

Під час функціонального тестування провірялись усі реалізовані функціональні можливості додатку.

Звіт по функціональному тестуванню наведено в додатку Д.

4.3.3.2. Підсумки тестування

Мобільний додаток розроблявся ітераційно, тобто від замовника постійно надходили нові побажання та зауваження. Тому процес тестування та відлагодження продовжувався протягом усього проекту. Часто були були зауваження, пов’язані інтерфейсом програми, наприклад, «Підправити відображення прапорів країн», поміняти графічне відображення спортивних подій в списку. Також були проблеми із пам’яттю, програма аварійно завершувалась після тривалого користування програмою. Тому мені прийшлось оптимізувати програмний код та алгоритми, для економії ресурсів. Також деякі дані сервер генерував некоректно. Наприклад, JSON даних про бейсбол мав іншу структуру. Це стало причиною того, що програма аварійно завершувалась після вибору цього виду спорту. Для вирішення цієї проблеми я модифікував алгоритм парсингу спеціально для цих даних, після чого проблема вирішилась. Дані завантажувались не в тому порядку, в якому потрібно. Для цього прийшлось розробляти алгоритми для їх впорядкування.

Крім того, замовник завжди вносив свої поправки. Добавлялись нові види спорту, які треба було інтегрувати в додаток. Також появлялись нові букмекерські контори. В процесі, змінювались посилання на них, логотип, назви. Це все виправлялось. В JSON появлялись нові поля, наприклад, посилання на «Watch Live» для букмекерів. Які я добавляв в додаток. Всі помилки були виправлені і всі зауваження замовника були задоволені. Більш детальна інформація про список багів та зауважень від замовника знаходиться в додатку З.

Перед релізом першої версії. Я здійснив тестування функціоналу системи. В результаті програма вже працювала стабільніше, ніж раніше, але все-таки появились декілька багів. Не завжди коректно завантажувався список подій, хоча мав би. Це було причиною поганої обробки виняткових ситуацій. Коефіцієнти в списку букмекерів некоректно відображались, якщо кількість цифр в числі була надто великою. Іноді список матчів в додатку не відповідав списку матчів на сайті. Також було декілька дефектів в інтерфейсі додатку. В результаті, я виправив всі помилки, провів повторне тестування, яке не виявило цих помилок. Згенерував підписаний apk файл, який згодом був опублікований на Google Play. Більш детальна інформація про список багів та зауважень від замовника знаходиться в додатку К.

Після редезайну та розробки нової версії продукту я провів функціональне тестування, результати якого наведені в таблиці 4.1.

Таблиця 4.1

Розподіл варіантів використання для функціонального тестування

| **Варіанти використання** | **Тестові дані** |
| --- | --- |
| Вибір виду спорту | 28 |
| Додавання команд (учасників) у «вибране» | 10 |
| Додавання ліг у «вибране» | 5 |
| Фільтрування «активних» подій | 3 |
| Фільтрування подій з прогнозами | 3 |
| Користувацьке оновлення даних | 5 |
| Фільтрування «вибраних» подій в Explore | 4 |
| Вибір дати для завантаження за допомогою календаря | 5 |
| Пролистування подій за датою | 4 |
| Пошук подій за пошуковою фразою | 10 |
| Перевірка деталей події на відповідність із сайтом | 10 |
| Перехід на сайт букмекера | 4 |
| Користувацьке оновлення деталей події | 3 |
| Поділитись подією з друзями в соціальних мережах | 4 |
| Отримання даних турнірної таблиці та перевірка її на відповідність із сайтом | 5 |
| Отримання даних «вибраних подій» | 10 |
| Перевірка роботи сповіщень | 3 |
| Зміна інтервалу оновлень | 2 |
| Очищення даних користувача | 3 |
| **Загалом** | **121** |

Всі тестові випадки пройшли успішно, тому функціональне тестування вважається успішним.

**4.3.4. Статичний аналіз коду в Android Studio**

Для визначення та усунення потенційно небезпечних ділянок в коді програми був проведений статистичний аналіз коду, результати якого наведені в Додатку З. Він допоміг знайти такі критичні місця програми як непроініціалізовані змінні, що можуть бути причиною небезпечних помилок, чи непотрібні ресурси, які погіршують продуктивність програми за рахунок надмірного використання оперативної пам’яті. Також аналіз виявив хардкод в коді програми, а також посилання, які можуть спричинити переповнення пам’яті. Після усунення цих критичних місць в програмі, якість коду та ресурсів значно покращилася.

**4.3.5. Критерій успіх/провал проекту.** Тут наведені умови, виконання яких свідчить про успішність проведення тестування та якість продукту:

Розробка тестів:

* Розроблені тести для кожного варіанту використання системи;
* Тести покривають усі варіанти використання;
* Тести покривають усі програмні вимоги.

Виконання тестів:

* Всі розроблені тести виконано;
* Програма успішно пройшла всі тести;
* Здійснено статистичний аналіз проекту засобами Android Studio.

Наведені умови задоволено, проект вважається успішним.

## 4.4. Висновок

В результаті реалізації та тестування програмного продукту розроблений мобільний додаток для ОС Android. Реалізовано всі класи та алгоритми необхідні для роботи всіх функцій програми. Реалізовано алгоритми завантаження та парсингу даних з мережі, алгоритм опрацювання даних спортивних подій. Реалізовано графічний інтерфейс. Виконане тестування програмного продукту, яке відбувалось на протязі всього процесу розробки. При цьому враховувались побажання та зауваження замовника. Також здійснено функціональне тестування, під час якого виконано перевірку всіх варіантів використання продукту. Для перевірки правильності написання коду був здійснений статичний аналіз коду за допомогою Androi Studio. Всі баги виправлено.

# РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

## 5.1. Економічна характеристика проектного рішення (програмного продукту)

Завданням дипломної роботи є створити мобільний додаток, за допомогою якого користувач зможе отримувати інформацію про різні спортивні події в режимі реального часу. Крім того, програма надаватиме інформацію про букмекерські ставки на результат спортивної події із декількох букмекерських контор. Це допоможе користувачу правильно здійснити ставку а також вибрати букмекерську контору із кращими ставками. Користувач зможе спостерігати за вибраними командами, гравцями, лігами та видами спорту, підписуючись на них. В результаті отримаємо програмний продукт, із зручним інтерфейсом, який спростить процес пошуку та отримання інформації про спортивні події.

Дана програма призначена для азартних спортивних фанатів, які бажають заробити гроші, здійснюючи ставки на результати спортивних подій, бути завжди в курсі останніх спортивних подій, прогнозів та розкладу майбутніх спортивних подій.

Розробка даного мобільного додатку є доцільна, з економічної точки зору. Є програми аналоги, які містять результати спортивних подій, але часто вони містять інформацію тільки про один вид спорту (в основному футбол), без букмекерської інформації. Даний мобільний додаток отримує інформацію про понад 30 видів спорту, з інформацією букмекерських контор, крім того в ньому наявні прогнози результатів спортивних подій, що дозволяє користувачу правильно прийняти рішення при здійсненні ставки на результат спортивної події без потреби аналізування безлічі букмекерських контор, що є досить тривалим та рутинним процесом.

## 5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару.

В процесі аналізу ринку конкурентів, я зрозумів, що дуже багато продуктів аналогів, які дають користувачу майже всю інформацію про спортивну подію. Але практично всі вони мають низку обмежень. Є додатки, які дають користувачу дуже детальну інформацію про спортивну подію, наприклад, матч, але призначена тільки для одного виду спорту. Є багато сайтів букмекерських, але важко зустріти серед аналогів додатки із букмекерською інформацією декількох контор. Якщо користувач спортивний фанат декількох видів спорту, то йому доводиться мати окремий додаток на кожен вид спорту. А якщо він, крім того, азартний, то часто йому не надається інформація необхідна при виборі ставки. Нижче наведені додатки конкуренти:

1. Live Score – мобільний додаток призначений для перегляду результатів спортивних матчів в режимі онлайн. Є інформація про такі спортивні матчі: футбол, хокей, баскетбол, теніс, крикет. Є можливість переглядати інформацію про пройдені матчі, а також матчі, які йдуть в даний момент. Основний акцент приділяється саме футболу, тому що додаток спочатку розроблявся саме для відображення даних про футбол. Наявна велика база ліг та матчів з усього світу.
2. Goal Live Scores – мобільний додаток для перегляду інформації в реальному часі про футбольні матчі та турніри. Відображає також рейтингову таблицю та детальну інформацію про матч.
3. Predictions Foot – мобільний додаток призначений для прогнозування футбольних результатів. Прогнози базуються на багаторічних статистичних даних. Також відображається інформація про різні команди. Для кожного матчу відображається прогноз (перемога першої команди, перемога другої команди, нічия). Для кожного матчу можна переглянути більш детальну інформацію. Ми можем переглянути результати попередніх матчів цих команд із вказанням рахунків. Також окремо відображається статистика кожної з двох команд, останні матчі команд, турнірна таблиця. Ці дані допомагають зробити ставку.

Я також проаналізував відгуки користувачів про додатки-конкуренти:

* Часто в згадувалось про те, що в додатках відсутня опція “Вибране”. Додатки мають інформацію про багато спортивних подій, а користувачів цікавило не все, а тільки окрема інформація. І було б добре надати користувачам можливість відзначати цікаві для них події. Наприклад опція «мої матчі», «мої ліги», «мої клуби». Користувачі також хочуть отримувати статистику про «Вибране».
* Майже у всіх додатках були відгуки пов’язані з тим, що користувачі хотіли бачити в них можливість налаштовувати сповіщення. Користувачі хотіли мати можливість отримувати сповіщення від матчів, команд, ліг, на які вони підписались. Дехто бажав мати можливість підписатись на матчі, які ще не розпочались, щоб програма сповістила їх про початок гри.

## 5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ.

Фактори зовнішнього середовища оцінюються за шкалою в межах від -5 до 5. Оцінка -5 означає найбільш негативний вплив факторів, а позитивний вплив факторів – оцінка 5. Якщо фактор не впливає на організацію, то він оцінюється оцінкою 0.

Щодо факторів внутрішнього середовища, то вони оцінюються за шкалою в межах [0; 5], при цьому 0 демонструє низький рівень розвитку або поганий стан фактору. Відповідно оцінка 5 відображає розвиненість фактору.

Сумою усіх факторних ваг є одиниця. Відповідно середня вагомість фактору – коефіцієнт. За допомогою цього коефіцієнта отримують рівень впливу факторів, помноживши його на експерту оцінку.

Результати оцінок впливу факторів зовнішнього та внутрішнього середовищ на організацію наведено у табл. 5.1.

*Таблиця 5.1*

Результати експертних оцінок

|  |  |  |  |
| --- | --- | --- | --- |
| Фактори | Середня експертна оцінка, бали | Середня вагомість факторів | Зважений рівень впливу, бали |
| 1 | 2 | 3 | 4 |
| *Фактори зовнішнього середовища* |  | | |
| Споживачі | 3 | 0,11 | 0,33 |
| Постачальники | 0 | 0,1 | 0 |
| Конкуренти | -4 | 0,1 | -0,4 |
| Державні органи влади | 0 | 0,05 | 0 |
| Інфраструктура | 1 | 0,06 | 0,06 |
| Законодавчі акти | 0 | 0,1 | 0 |
| Профспілки, партії та інші громадські організації | 0 | 0,05 | 0 |
| Система економічних відносин в державі | -1 | 0,06 | -0,06 |
| Організації-сусіди | 1 | 0,01 | 0,01 |
| Міжнародні події | 2 | 0,01 | 0,02 |
| Міжнародне оточення | 2 | 0,03 | 0,06 |
| Науково-технічний прогрес | 3 | 0,07 | 0,21 |
| Політичні обставини | 0 | 0,06 | 0 |
| Соціально-культурні обставини | 0 | 0,05 | 0 |
| Рівень техніки та технологій | 0 | 0,04 | 0 |
| Особливості міжнародних економічних відносин | -1 | 0,02 | -0,02 |
| Стан економіки | 0 | 0,08 | 0 |
| **Загальна сума** | — | 1 | 0,21 |

*Продовження табл. 5.1*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *Фактори внутрішнього середовища* |  |  |  |
| Цілі | 0 | 0,11 | 0 |
| Структура | 1 | 0,16 | 0,16 |
| Завдання | 3 | 0,07 | 0,21 |
| Технологія | 4 | 0,2 | 0,8 |
| Працівники | 4 | 0,21 | 0,84 |
| Ресурси | 2 | 0,25 | 0,5 |
| **Загальна сума** | — | 1 | 2,51 |

Відповідно до результатів оцінювання найбільш позитивний вплив на розробку програмного продукту становлять споживачі, тому що чим більше користувачів скачуватимуть додаток, тим більшим буде рейтинг додатку в Google Play. Позитивні відгуки та оцінки користувачів приваблюватимуть нових користувачів, завдяки чому продукт ставатиме ще популярнішим. Також позитивний вплив становить науково-технічний прогрес, тому що на даний час покращується мобільна мережа Інтернет. Мобільні провайдери забезпечують користувачів високошвидкісним інтернетом 3G, технічні характеристики смартфонів покращуються. Все це робить користування смартфоном невід’ємною частиною життя.

Найбільш негативним фактором є конкуренти на ринку. Цей фактор часто є критичним серед Android додатків. Магазин Google Play містить мільйони додатків на різні жанри і тематики. Серед них є тисячі додатків, пов’язаних із спортивними подіями. Багато із них мають досить високі рейтинги і кількість завантажень. Відповідно під час пошуку додатків вони є першими в рейтингу. Як і для розробка будь якого Android додатку, розробка цього додатку передбачає надання йому таких можливостей, які б могли його виділити на фоні конкурентів.

Фактори, які здійснюють найбільший вплив – це фактори внутрішнього середовища. Особливо, технологія та працівники. Вибір технології є важливим фактором, адже від нього залежить час розробки ПЗ, його якість, а отже і ціна. Працівники теж важливий фактор, а точніше їхній досвід та особисті якості, такі як відповідальність, організованість, швидкість у навчанні та рівень комунікації в команді.

## 5.4. Формування стратегічних альтернатив

* + 1. **Стратегічні альтернативи, група №1**

Критеріями поділу альтернативних стратегій розвитку є існуючий продукт (ПЗ) та новий, а також відповідні послуги.



Рис. 5.1. Стратегічні альтернативи, група №1.

**Стратегія розробки повністю нового продукту (проектного рішення**)передбачає розробку повністю нового програмного забезпечення, яке призначене для вирішення нових проблем людини, економіки чи суспільства.

**Стратегія розвитку уже існуючого програмного продукту**, характеризується модифікацією та удосконаленням, продукту, який існує на даний момент.

**Стратегія розвитку наявного на ринку продукту (проектного рішення) з супутніми послугами** означає пропонування зміненого програмного забезпечення із додатковими послугами, такими як: встановлення, коригування, супроводження.

**Стратегія розвитку нового продукту з супутніми послугами** передбачає розробку нового проектного рішення, включаючи при цьому додаткові послуги., такі як супроводження, коригування, встановлення.

* + 1. **Стратегічні альтернативи, група №2**

Альтернативні стратегії другої групи поділяються за такими критеріями, як новий або існуючий продукт (чи проектне рішення), а також відсутній чи наявний ринок.

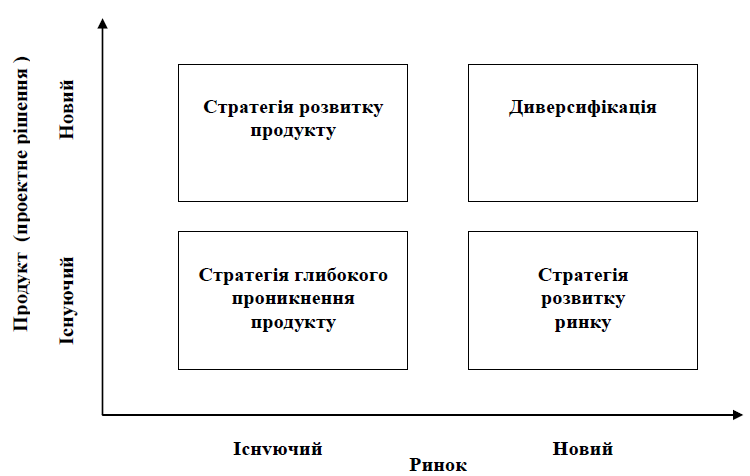


Рис. 5.2. Стратегічні альтернативи, група №2

**Стратегія глибокого проникнення на ринок (концентрація)** – полягає в пошуку та знаходженні шляхів, які збільшують збут товарів певної фірми, які продаються на освоєних ринках. Це забезпечується за допомогою більш агресивного маркетингу. Таку стратегію можна вважати ефективною, якщо ринок ще не досить насичений і розширюється. Фірма збільшує інтенсивність руху товарів, їхнього просування і використовує найбільш вигідні (конкурентоспроможні) ціни і таким чином розширює ринок збуту товарів. Фірма приваблює користувачів, які раніше не використовували їхню продукцію та збільшує попит серед споживачів, які раніше користувалися їхньою продукцією.

**Щодо стратегії розвитку ринку, то вона передбачає вихід фірми на новий ринок з уже наявним товаром. Таким чином збільшуються обсяги збуту продукції. При цьому фірма може орієнтуватися на нові сегменти ринку або ж виходити на зовсім нові географічні ринки.**

**Стратегія розвитку товару** полягає в покращенні та вдосконаленні існуючих і виготовленні нових товарів для існуючих ринків. Дана стратегія є досить ризикованою, але вона передбачає збільшення показників збуту продукції.

**Стратегія диверсифікації** – характеризується розробкою нової продукції для нових сфер бізнесу. Рівень ризику в неї вважається найвищим.

* + 1. **Вибір стратегій**

Вивчивши першу групу стратегічних альтернатив, я вибрав стратегію розроблення існуючого продукту з супутніми послугами. Програмний продукт має потребу в підтримці для збільшення аудиторії користувачів. Потрібно збільшувати кількість підтримуваних додатком видів спорту, із підтримкою турнірних таблиць та статистичної інформації, яка є різною для різних видів спорту. Також вдосконалювати графічний інтерфейс, роблячи його максимально зручним та анімований, що залежить від нових версій операційної системи Android. В майбутньому можлива підтримка більшої кількості букмекерських контор та відео трансляції матчів із додатку.

Із стратегічних альтернатив другої групи вибрано стратегію розвитку продукту. На ринку Android додатків є достатня кількість продуктів-конкурентів, але для отримання необхідного функціоналу потрібно скачувати декілька додатків одночасно, тому що жоден із них не має всього функціоналу. Тому варто виділити основу функціональності, базуючись на продуктах аналогах. Після цього добавляти новий функціонал, випускаючи нові версії продукту.

## 5.5. Бюджетування

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, які пов’язані з виготовленням та реалізацією товару. Вона дає можливість здійснити аналіз витрат та розробити заходи, завдяки яким можна підвищити рентабельність виробництва. На цьому етапі визначають собівартість товару, який виготовляється, та економічно обґрунтовується доцільність вибору однієї зі стратегій.

*Таблиця 5.2*

Бюджет витрат матеріалів та комплектуючих виробів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва матеріалів та комплектуючих** | **Марка, тип, модель** | **Фактична кількість, шт.** | **Ціна за одиницю, грн.** | **Разом, грн.** |
| Ноутбук | Lenovo ThinkPad t410 | 1 | 3900 | 3900 |
| Wifi роутер | TP-LINK TL-WR720N | 1 | 300 | 300 |
| Смартфон | Asus padfone E | 1 | 2163 | 2163 |
| Разом: | - | - | 6363 | 6363 |

*Таблиця 5.3*

Бюджет витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Посада,**  **спеціальність** | **Кількість працівників, осіб** | **Час роботи, дні** | **Денна заробітна плата працівників, грн.** | **Сума витрат на оплату праці, грн.** |
| Основна заробітна плата | | | | |
| Дизайнер | 1 | 5 | 300 | 1500 |
| Програміст | 1 | 45 | 350 | 15750 |
| Тестер | 1 | 10 | 170 | 1700 |
| Разом | 3 | 60 | 820 | 18950 |

*Таблиця 5.4*

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| **Статтi витрат** | **Сума, грн.** |
| *Змiннi загальновиробничi витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 1450 |
| - витрати на МШП; | 150 |
| - витрати на електроенергію | 300 |
| - інші змінні витрати; | 100 |
| Разом змiнних витрат: | 2000 |
| *Постiйнi загальновиробничi витрати, у т.ч.:* |  |
| - комунальнi послуги; | 750 |
| - витрати на оренду; | 2500 |
| - витрати на ремонт; | 100 |
| - iншi постiйнi витрати; | 200 |
| Разом постiйних витрат: | 3550 |
| *Разом загальновиробничих витрат:* | 5750 |

*Таблиця 5.5*

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статтi витрат** | **Сума, грн.** |
| 1 | 2 |
| *Адмiнiстративнi витрати, у т.ч.:* |  |
| - заробiтна плата адмiнiстративного персоналу; | 1500 |
| - витрати на МШП; | 150 |
| - витрати на сплату податкiв i зборiв; | 1800 |
| - знос адмiнiстративного обладнання; | 350 |
| Разом адмiнiстративних витрат: | 3800 |
| *Витрати на збут, у т.ч.:* |  |
| - заробiтна плата менеджерiв зi збуту; | 1800 |
| - витрати на гарантiйний ремонт; | 400 |
| - витрати на гарантiйне обслуговування; | 150 |
| - витрати на налагодження i експлуатацiю; | 500 |
| - витрати на паливо-мастильнi матерiали; | 0 |
| - витрати на рекламу; | 1300 |
| Разом витрат на збут: | 4150 |

*Таблиця 5.6*

**Зведений кошторис витрат на розробку проектного рiшення (продукту)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Статтi витрат** | **Одиницi вимiру** | **Фактична кiлькiсть, шт.** | **Цiна одиницi, грн.** | **Разом, грн.** |
| Сировина i матерiали | шт | 4 | (3900,300,2163) | 6363 |
| Паливо та електроенергiя на технологiчнi цiлi | кВт | 2909 | 0.99 | 287 |
| Основна заробiтна плата | грн | 3 | (1500,15750, 1700) | 18950 |
| Додаткова заробiтна плата | грн | - | - | - |
| Вiдрахування на соцiальне страхування | грн | 3 | (549,5764,622) | 6935 |
| Загальновиробничi витрати, у т.ч.: |  | | | |
| - змiннi; | грн | - | - | 2000 |
| - постiйнi; | грн | - | - | 3550 |
| *Разом виробничих витрат:* | грн | - | - | **38085** |
| Адмiнiстративнi витрати | грн | - | - | 3800 |
| Витрати на збут | грн | - | - | 4150 |
| Iншi операцiйнi витрати | грн | - | - | 1500 |
| *Разом виробничих i операцiйних витрат:* | грн | - | - | **47535** |

Для того, щоб визначити фінансові результати, необхідно розрахувати ціну (вартість) продукту (мобільного додатку), який розробляється. Ціна визначається на основі суми операційних та виробничих витрат з врахуванням рентабельності виробництва 40%. Ціна = 47535 х 0.4 = 19014 грн.

Отже, ціна мобільного додатку становить 19014 грн.

*Таблиця 5.7*

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| 1 | 2 |
| Дохiд вiд реалiзацiї продукцiї (обсяг 5) | 95070 |
| Податок на додану вартiсть (20%) | 19014 |
| Чистий дохiд вiд реалiзацiї продукцiї | 76056 |
| Собiвартiсть реалiзованої продукцiї | 19014 |
| Валовий прибуток | 57042 |

*Продовження табл. 5.7*

|  |  |
| --- | --- |
| *Операцiйнi витрати:* | |
| - адмiнiстративнi витрати | 3800 |
| - витрати на збут | 4150 |
| - iншi операцiйнi витрати; | 0 |
| Фiнансовий результат вiд операцiйної дiяльностi | 49092 |
| Податок на прибуток (25%) | 12273 |
| Чистий прибуток (збиток) | 36819 |

## 5.6. Остаточний вибір стратегії

Здійснивши економічну оцінку доцільності розробки мобільного додатку для опрацювання і відображення спортивних подій і букмекерських ставок Live2Sport на операційну систему Android можна зробити висновок, що розробка цього проектного рішення є економічно вигідним та доцільним рішенням.

Проаналізувавши продукти-аналоги було вибрано стратегію розроблення існуючого продукту із супутніми послугами із альтернативних стратегій першої групи, що передбачає постійну підтримку та супроводження програмного продукту із адаптацією під потреби користувачів. Із альтернативних стратегій з критеріями поділу на продукт та існуючий ринок, новий продукт та ринок вибрано стратегію розвитку продукту, що передбачає в розробку продукту на основі існуючих аналогів, модифікуючи та покращуючи при цьому його функціональність. Таким чином мобільний додаток поєднує кращі проектні рішення своїх конкурентів.

Також я здійснив розрахунок витрат на розробку мобільного додатку. Визначив собівартість реалізованого продукту, яка становить 19014 грн (коефіцієнт рентабельності 40%). Згідно обчислень чистий прибуток становить 36819 грн. враховуючи розробку 5 примірників мобільного додатку.

За отриманими результатами можна зробити висновок, що мобільний додаток досить вигідний у розробці та реалізації.

# Висновки

В результаті виконання бакалаврської кваліфікаційної роботи я проаналізував існуючі додатки – аналоги, вивчив та оцінив різні алгоритми для отримання опрацювання даних з мережі, зібрав та проаналізував вимоги до програмного продукту, в результаті чого сформовано специфікацію вимог до системи. Також я вивчив архітектури MVP та використав цей шаблон під час проектування системи, розробив мобільний додаток та здійснив його тестування.

Проаналізувавши продукти конкуренти я виявив, що жоден з продуктів аналогів не володіє достатньою функціональністю. Є додатки які отримують інформацію тільки про один вид спорту, звужуючи при цьому аудиторію користувачів. Є такі, які отримують інформацію про спортивні події, але в них відсутня букмекерська інформація та прогнози. Тому постає необхідність в користуванні декількома мобільними додатками, кожний з яких окремо надає певний функціонал, або ж в реалізації нового додатку, що й було здійснено.

Реалізовано функції отримання та парсингу даних з мережі, опрацювання даних спортивних подій, «живий» пошук спортивної події за пошуковою фразою, отримання даних турнірних таблиць, різноманітні фільтри, можливість підписуватись на улюблені ліги та команди, автоматичні оновлення даних та сповіщення.

Виконано тестування та відлагодження мобільного додатку. Для цього я виконав GUI тестування, модульне тестування основних модулів системи. Крім того я здійснив статичний аналіз коду за допомогою Android Studio. В результаті було виправлено помилки.

В майбутньому я планую вдосконалювати мобільний додаток, додаючи в нього нові можливості, такі як підтримку нових видів спорту, отримання інформації від нових букмекерських контор, оптимізація алгоритму опрацювання даних спортивних подій, покращення або заміна API сервера для економії інтернет трафіку.

# список Літератури

1. Букмекер [Електронний ресурс]: – Режим доступу: <https://uk.wikipedia.org/wiki/%D0%91%D1%83%D0%BA%D0%BC%D0%B5%D0%BA%D0%B5%D1%80>
2. Кращі букмекерські контори, ставки на спорт в інтернеті [Електронний ресурс]: – Режим доступу: <http://vse-prosto.весьтоп.рф/krashhi-bukmekerski-kontori-stavki-na-sport-v.html>
3. Google Play [Електронний ресурс] : – Режим доступу: <https://play.google.com>.
4. Retrofit. A type-safe HTTP client for Android and Java [Електронний ресурс] : – Режим доступу: <http://square.github.io/retrofit/>
5. Єккель Б. – Философия Джави. Библиотека програмиста. 4-е изд. – СПб.: ил. – (Серия «Библиотека програмиста»).
6. Android Studio. The Official IDE for Android [Електронний ресурс] : – Режим доступу: <https://developer.android.com/studio/index.html>
7. Android Design Principles [Електронний ресурс] : – Режим доступу: <https://developer.android.com/design/get-started/principles.html>
8. Построение Android приложений шаг за шагом, часть первая [Електронний ресурс] : – Режим доступу: <https://habrahabr.ru/company/rambler-co/blog/275943/>
9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. – П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста»).
10. Харді Б., Программирование под Android. Для профессионалов. – СПб.:Питер, 2014 – 51с.
11. Everything you ever wanted in a UX Design Platform [Електронний ресурс] : – Режим доступу <https://www.uxpin.com/tour.html>

# Додаток А. Блок-схема алгоритму опрацювання спортивних подій та букмекерських ставок

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Шліхутка Б. Т. | Підпис |  | Блок-схема алгоритму | Сторінка |
| Перевірив | Сердюк П. В. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

# Додаток Б. Діаграма варіантів використання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Шліхутка Б.Т. | Підпис |  | Діаграма варіантів використання | Сторінка |
| Перевірив | Сердюк П. В. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

# Додаток В. Діаграма діяльності

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Шліхутка Б. Т. | Підпис |  | Діаграма діяльності | Сторінка |
| Перевірив | Сердюк П. В. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

# Додаток Д. Звіт по тестуванню

Функціональність програми ми тестуємо перевіряючи чи виконуються тестові випадки.

*ID RM-1 : Вибір виду спорту*

|  |  |
| --- | --- |
| *Summary:* Перевіряється чи завантажується поточний вид спорту і чи він зберігається в налаштуваннях | |
|  | |
| *Steps:* | *Expected Results:* |
| 1. Відкрити бічну панель, просунувши пальцем по екрані зліва на право 2. Натиснути на пункт «Sport types». 3. Вибрати в списку видів спорту відповідний (“Handball”) та натиснути на нього. 4. Перезавантажити програму | 1. З’являється бокова панель. 2. Відкривається вікно з вибором виду спорту. 3. Вид спорту зберігається в налаштуваннях та відкривається вікно feed з списком його подій. 4. Після перезавантаження відкривається feed з списком подій цього ж виду спорту. |

Резолюція тестового випадку *–* **Passed**

*ID RM-2 : Додавання спортивних подій у вибране*

|  |  |
| --- | --- |
| *Summary:* Перевіряється чи додається подія у «Вибране» та чи відображається вона потім в вікні Feed | |
| *Pre-condition:* Виконано тестовий випадок RM-1. | |
| *Steps:* | *Expected Results:* |
| 1. Відкрити бічну панель та вибрати «Explore» 2. Відзначити декілька Ліг та спортивних подій (матчів для Handbal) зірочками, натиснувши на них. 3. Відкрити бічну панель та вибрати «Feed». | 1. Відкривається бокова панель. 2. Завантажується вікно Explore. 3. Біля вибраних ліг та спортивних подій з’являється біла зірочка. 4. Відкривається бокова панель. 5. Завантажується вікно Feed з подіями та лігами, які вибрав користувач. |

Резолюція тестового випадку *–* **Passed**

*ID RM-3 : Фільтрування спортивних подій та пошук*

|  |  |
| --- | --- |
| *Summary:* Коректність спільної роботи фільтрів та пошуку | |
| *Pre-condition:* Виконано тестовий випадок RM-2 (деякі ліги та спортивні події відзначено як «Вибрані», серед них є «активні» та такі, що мають прогнози). | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо вікно “Explore”. 2. Натискаємо на “Live” опцію в меню, роблячи її активною. 3. Натискаємо на “Prediction” опцію в меню, роблячи її активною. 4. Натискаємо на зірочка в меню, роблячи її активною. 5. Натискаємо на опцію пошуку в меню. 6. В поле пошуку вводимо тестову фразу «paok». | 1. Відкривається бокова панель. 2. Відкривається вікно «Explore» із усіма подіями на сьогоднішню дату. 3. Події фільтруються та виводяться тільки ті, що відбуваються в даний момент. 4. Отримані події фільтруються. З низ виводяться ті, які мають прогнози. 5. Отримані події фільтруються. З низ виводяться ті, які користувач відзначив як «Вибрані». 6. Серед тих подій, що залишилось здійснюється пошук за заданою фразою та виводяться тільки ті події які цю фразу мають (регістр не враховується). |

Резолюція тестового випадку *–* **Passed**

*ID RM-4 : Автоматичне оновлення даних спортивних подій.*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо чи відбувається автоматичне оновлення даних із встановленим інтервалом та чи відбуваються сповіщення | |
| *Pre-condition:* Виконано тестовий випадок RM-2 (деякі ліги та спортивні події відзначено як «Вибрані», серед них є «активні»). | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо бокову панель та вибираємо пункт “Settings”. 2. Встановлюємо інтервал оновлення 10 секунд. 3. Вмикаємо автоматичні оновлення та сповіщення про події. 4. Відкриваємо бокову панель та переходимо в вікно Feed. З моменту останнього перезавантаження чекаємо 10 секунд до наступного перезавантаження. | 1. Відкривається бокова панель. 2. Відкривається вікно з налаштуваннями. 3. Вмикаються оновлення, сповіщення, їхній інтервал зберігається в локальних налаштуваннях. 4. Завантажується вміст вікна feed. 5. Через 10 секунд вміст перезавантажується. |

Резолюція тестового випадку *–* **Passed**

*ID RM-5 : Перевірка роботи сповіщень*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо проходять сповіщення після настання подій, таких як гол забитий на матчі, який відзначено як «Вибране» | |
| *Pre-condition:* В налаштуваннях ввімкнені сповіщення, інтервал оновлень 10 секунд, поточний вид спорту футбол, користувач підписаний на активні події. | |
| *Steps:* | *Expected Results:* |
| 1. Закриваємо додаток. 2. Очікуємо на настання події – команда забила гол (для цього слідкуємо за прогресом на сайті tvevents.org). 3. Користувач натискає на сповіщення, яке з’явилось в телефоні. | 1. Додаток закривається. 2. При настанні події «команда забила гол» додаток формує сповіщення. 3. Серед сповіщень в телефоні з’являється з короткою інформацією хто забив гол. 4. Відкривається вікно Feed в додатку, в якому є ця подія. |

Резолюція тестового випадку *–* **Passed**

*ID RM-6 : Перевірка правильності інформації в деталях матчу*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо на відповідність інформації із інформацією на сайті tvevents.org. | |
|  | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо вікно “Explore”. 2. Натискаємо на певну подію. 3. Відкриваємо сайт tvevents.org та перевіряємо на відповідність. | 1. Завантажуються дані в вікно Explore. 2. Завантажуються вікно з деталями матчу. |

Резолюція тестового випадку *–* **Passed**

*ID RM-7 : Перевірка правильності інформації в статистиці ліги*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо на відповідність інформації із інформацією на сайті tvevents.org. | |
|  | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо вікно “Explore”. 2. Натискаємо на певну лігу. 3. Відкриваємо сайт tvevents.org та перевіряємо на відповідність із інформацією в вікні турнірної таблиці ліги | 1. Завантажуються дані в вікно Explore. 2. Завантажуються вікно з турнірною таблицею ліги. |

Резолюція тестового випадку *–* **Passed**

*ID RM-7 : Перевірка роботи функції очистки кешу*

|  |  |
| --- | --- |
| *Summary:* Чи очищаються усі «Вибрані» користувачем події | |
| *Pre-condition:* Виконано тестовий випадок RM-2 | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо вікно “Settings”. 2. Натискаємо на “Clear Cache”. 3. Переходимо у вікно «Feed». | 1. Відкривається вікно «Settings». 2. Очищуються дані користувача. 3. Відкривається вікно Feed із порожнім списком (усі «Вибрані» події очистились). |

Резолюція тестового випадку *–* **Passed**

# Додаток Е. Інструкція користувача

**Е.1. Компоненти програмного забезпечення**

Мобільний додаток написаний на мові Java. Для цього використовувалось середовище програмування Android Studio 2.1. Дана програма може працювати під управлінням ОС Android 4.0.4 і вище.

Для роботи програми потрібний доступ до інтернет мережі, частота процесора не нижче 1 GHz, оперативна пам’ять понад 512 Mb та мінімум 20 Mb вільного простору на диску.

Таблиця Е.1

Файли, що необхідні для коректної роботи мобільного додатку

|  |  |  |  |
| --- | --- | --- | --- |
| № | Файл | Призначення | Належить проекту |
| 1 | Live2Sport.apk | Виконавчий файл | Мобільний додаток для опрацювання та відображення спортивних подій і букмекерських ставок |

**E.2. Встановлення програмного забезпечення**

Для для установлення мобільного додатку потрібно скопіювати файл Livi2Sport.apk в постійну пам’ять смартфону, наприклад, на sd-карту або в пам’ять телефону. Після цього слід установити галочку «Дозволяти установку сторонніх додатків» в системних налаштуваннях смартфону. Зайти в директорію з файлом додатку та натиснути на нього. Система може спитати про дозвіл встановлення програми із стороннього джерела та дозвіл на використання інтернету. В обох випадках потрібно погодитись.

**Е.3. Базові функції ПЗ**

Мобільний додаток дозволяє виконувати такі базові функції як вибрати вид спорту за замовчуванням, отримати інформацію про спортивні події вибраного виду спорту, отримати детальну інформацію про спортивну подію із списком букмекерських коефіцієнтів та переходом на сайти букмекерських контор. Також програма отримує турнірні таблиці спортивних змагань, наприклад, футбольних ліг. Фільтрувати спортивні події за такими критеріями як активні спортивні події, спортивні події із прогнозами, «вибрані події». Наявний пошук. Також можна підписатись на «вибрані» ліги, команди (учасники спортивних змагань) та отримувати інформацію про спортивні події, які з ними пов’язані. Є можливість отримувати сповіщення. Є можливість налаштувати додаток.

Для того, щоб додати команду чи лігу у «вибране» та отримувати інформацію про них потрібно:

1. В зайти в вікно «Explore» знайти потрібну лігу чи команду. Для цього можна скористатись пошуком, ввівши потрібну пошукову фразу.
2. Відзначити потрібну команду чи лігу зірочкою.
3. При потребі можна налаштувати додаток (зайти в налаштування, ввімкнути автоматичні оновлення, сповіщення, вибрати частоту оновлення).
4. Зайти в вікно «Feed» де буде відображатись «вибрані» події.
5. Події можна відфільтрувати, наприклад, відображати тільки ті, які йдуть в даний момент.
6. Для перегляду турнірної таблиці ліги натиснути на неї.
7. Для перегляду букмекерської інформації про спортивну події натиснути на цю подію в «Feed». Відобразиться детальна інформація про цю подію.

**Е.4. Аналіз помилок та можливих проблем**

В разі неправильної роботи програми в першу чергу потрібно перевірити наявність та якість інтернет з’єднання. Якщо проблема не вирішена то перезавантажити додаток, очистити кеш та відключити оновлення та сповіщення. Якщо й це не допомогло то переустановити додаток.

# Додаток Ж. Лістинг коду

***Проектування алгоритму завантаження та парсингу даних з мережі***

TvEventsFetchr – клас, що відповідає за завантаження та парсинг даних спортивних подій і букмекерської інформації, що приходить із сервера.

Public public class TvEventsFetchr {

Нижче наведений алгоритм парсингу даних спортивних подій із json.

Void parseMatches(List<Match> items, String jsonString) {

try {

//отримуємо головний json обєкт, в якому містяться інші обєкти

JSONObject mainJsonObject = (JSONObject) new JSONTokener(jsonString.toString())

.nextValue();

//отримуємо назву виду спорту

String sportTypeKey = mainJsonObject.names().get(0).toString();

if (sportTypeKey != null) {

//формуємо об’єкт який містить масив спортивних подій

JSONArray array = mainJsonObject.getJSONArray(sportTypeKey);//sport type array

//із json масиву витягуємо окремо кожну подію

for (int i = 0; i < array.length(); i++) {

JSONObject jsonObject = array.getJSONObject(i);

добавляємо її список подій – моделі даних

items.add(new Match(jsonObject));

//Log.i(TAG, «Json object : « + jsonObject.toString());

}

}

}catch (Exception e) { Log.d(TAG, e.getMessage()); } }

Нижче наведений алгоритм парсингу букмекерських даних із json.

//на вхід отримуємо текстові дані в форматі json

List<Bookmaker> parseBookmakers(String jsonString) throws JSONException{

List<Bookmaker> bookmakers = new ArrayList<>();//сюди будуть записуватись списку букмекерів

JSONArray jsonObjectArr = new JSONArray(jsonString);

//отримуємо посилання на оголовний обєкт json

JSONObject jsonObject = new JSONObject(jsonObjectArr.get(0).toString());

//отримуємо список отрибутів json в них містяться назви букмекерів

JSONArray jsonKeys = jsonObject.names();

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());//берем поточний атрибут

if (!(key.endsWith(«\_1\_link») ||key.endsWith(«\_2\_link») ||key.endsWith(«\_x\_link») || key.equals(«event\_id»))){//берем тільки ті, що відносяться до букмекерів, фільтруєм решту атрибутів

String bookmakerName = key;//поточний атрибут – назва букмекера

//значення атрибуту – букмекерські коефіцієнти в форматі W:D:L (виграш, нічия, поразка)

String koefsStr = jsonObject.get(bookmakerName).toString();

if (koefsStr == null || koefsStr.equals(«»))

continue;

String[] koefs = koefsStr.split(«:»);//отримуємо значення коефіцієнтів

if (bookmakerName.equals(«will»))//костиль – проблема не в мене а на сервері

bookmakerName = «whill»;

bookmakers.add(new Bookmaker(bookmakerName, koefs));//додаємо букмекера

}

}

return bookmakers; }

Отримуємо json обєкт зі спортинвою подією в коструктор моделі спортивної події. Після цього модель заповнюється.

Public Match(JSONObject json) throws JSONException{

try{//записуємо в полі моделі відповіддні значення із json обєктів

eventId = json.getString(JSON\_EVENT\_ID);

teamHome = json.getString(JSON\_TEAM\_HOME);

teamAway = json.getString(JSON\_TEAM\_AWAY);

scores1 = json.getString(JSON\_SCORES1).split(« «)[0];//without penalties

scores = json.getString(JSON\_SCORES);

league = json.getString(JSON\_LEAGUE);

time = json.getString(JSON\_TIME);

coeff = json.getInt(JSON\_COEFF);

updateScore = json.getString(JSON\_UPDATE\_SCORE);

eventPrediction = json.getString(JSON\_EVENT\_PREDICTION);

liveScore = json.optString(JSON\_LIVE\_SCORE, null);

extraInfo = parseExtraInfo(json.getString(JSON\_EXTRA\_INFO));

}catch (JSONException e){

Log.e(TAG, «Error parsing item match»);

e.printStackTrace();

throw e;

}

//парсинг букмекерської інформації

JSONArray jsonKeys = json.names();

final String EVENT\_LINK = «events\_link\_»;

//тут отримуємо список букмекерів, які мають відеотрансляції спортивної події

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());

if (key.startsWith(EVENT\_LINK)){

String bookmakerName = key.replace(EVENT\_LINK, «»);

String isWatchLive = json.get(key).toString();

if (isWatchLive != null && isWatchLive.equals(«1»)){

if (bookmakerName.equals(«1xbet»))

bookmakerName = «marathon»;

mWatchLiveBookmakers.add(new Bookmaker(bookmakerName, true));

}

}

} }

Уривок коду із коструктора головної моделі спортивної події

public Match(JSONObject json) throws JSONException{

try{//записуємо в полі моделі відповіддні значення із json обєктів

eventId = json.getString(JSON\_EVENT\_ID);

teamHome = json.getString(JSON\_TEAM\_HOME);

teamAway = json.getString(JSON\_TEAM\_AWAY);

scores1 = json.getString(JSON\_SCORES1).split(« «)[0];//without penalties

scores = json.getString(JSON\_SCORES);

league = json.getString(JSON\_LEAGUE);

time = json.getString(JSON\_TIME);

coeff = json.getInt(JSON\_COEFF);

updateScore = json.getString(JSON\_UPDATE\_SCORE);

eventPrediction = json.getString(JSON\_EVENT\_PREDICTION);

liveScore = json.optString(JSON\_LIVE\_SCORE, null);

extraInfo = parseExtraInfo(json.getString(JSON\_EXTRA\_INFO));

}catch (JSONException e){

Log.e(TAG, «Error parsing item match»);

e.printStackTrace();

throw e;

}

//парсинг букмекерської інформації

JSONArray jsonKeys = json.names();

final String EVENT\_LINK = «events\_link\_»;

//тут отримуємо список букмекерів, які мають відеотрансляції спортивної події

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());

if (key.startsWith(EVENT\_LINK)){

String bookmakerName = key.replace(EVENT\_LINK, «»);

String isWatchLive = json.get(key).toString();

if (isWatchLive != null && isWatchLive.equals(«1»)){

if (bookmakerName.equals(«1xbet»))

bookmakerName = «marathon»;

mWatchLiveBookmakers.add(new Bookmaker(bookmakerName, true));

}

}

} }

***Реалізація алгоритму опрацювання спортивних подій та букмекерських ставок***

Даний алгоритм здійснює основну обробку списку спортивних подій. На вхід отримуємо модель даних із неопрацьованими спортивними подіями із попреднього кроку. Алгоритм їх фільтрує, розбиває по групах (якщо це матчі, то здійснюється розподіл по лігах). Впорядковує групи та події всередині груп.

Public class MatchLab {

public static List<League> groupMatchesByLeagues(Context context, List<Match> matches){

//перевірка опції «завантажити тільки спортивні події»

if (ResourcesSolver.loadIsActiveMatches(context)){

List<Match> tempMatches = matches;

matches = new ArrayList<>();

//перевіряється конжна подія

for (Match match : tempMatches){

if (match.isActive())//чи подія активна (відбувається в даний момент)

matches.add(match);//вставляємо в список

}

}

//перевірка опції «завантажити тільки події з прогнозами»

if (ResourcesSolver.loadIsEventPredictionMatches(context)){

List<Match> tempMatches = matches;

matches = new ArrayList<>();

for (Match match : tempMatches){

if (match.getEventPrediction() != null)

//аналогічно здійснюється перевірка чи подія має прогноз

matches.add(match);

}

}

//на цьому кроці список подій вже пофільтрований

//створюємо обєкт який вміє порівнювати події за коефіцієнтами

//потім буде використаний при сортуванні

Comparator<Match> matchComparator = new Comparator<Match>() {

@Override

public int compare(Match lhs, Match rhs) {

return lhs.getCoeff() < rhs.getCoeff()? 1: -1;

}

};

//групуємо події за лігами

List<League> groupedMatches = new ArrayList<>();

Set<String> leagues = new TreeSet<>();//формуємо множину назв ліг

for (Match match : matches){//здіснюємо прохід по всіх подіях

String league = match.getLeague();//отримуємо лігу

if (!leagues.contains(league)){

leagues.add(league);//вставляємо лігу в множину

}

}//на виході отримуємо множину із назвами ліг

for (String league : leagues){//формуємо список ліг

List<Match> matchesInLeague = new ArrayList<>();

double maxCoeff = 0;

for (Match match : matches){

//формуємо список подій для конкретної ліги

//для цього порівнюємо назву його ліги з поточною

String matchLeague = match.getLeague();

if (league.equals(matchLeague)){

matchesInLeague.add(match);

}

}

//сортуємо подій всередині ліги по спаданню коефіцієнтів

Collections.sort(matchesInLeague, matchComparator);

//визначаємо максимальний коефіцієнт подій всередині ліги. Це буде загальний коефіцієнт ліги

maxCoeff = (double)matchesInLeague.get(0).getCoeff();

String countryFlagPath = CountryFlagsLab.get(context).getCountryPath(league);

//формуємо обєкт ліги

groupedMatches.add(new League(league, matchesInLeague, countryFlagPath, maxCoeff));

}

//здійснюємо сортування списку ліг за їхнім коефіцієнтом

Collections.sort(groupedMatches, new Comparator<League>() {

@Override

public int compare(League lhs, League rhs) {

if (lhs.getCoeff() == rhs.getCoeff()){

return lhs.getName().compareTo(rhs.getName());

}

return lhs.getCoeff() < rhs.getCoeff()? 1: -1;

}

});

//для кожної з ліг здійснюємо сортування списку подій

//таким чином, що спочатку йтимуть активні події, потім подіії, які ще не розпочались

//останніми будуть завершені події

// кожна з цих трьох груп буде впорядкована за спаданням коефіцієнтів події

for (League league : groupedMatches){

Collections.sort(league.getMatches());

}

/\*тут наведений перевантажений метод класу Match

public int compareTo(@NonNull Object another) {

Match match = (Match)another;

int thisKoef = 0, anotherKoef = 0;

if (isActive())//визначається ця подія є активна

thisKoef = 1;

else if (isEnded())//чи подія завершена

thisKoef = -1;

if (match.isActive())//чи інша подія активна

anotherKoef = 1;

else if (match.isEnded())//чи інша подія завершена

anotherKoef = -1;

if (thisKoef > anotherKoef){//порівнюються коефіцієнти подій

return -1;

}else if (thisKoef < anotherKoef){

return 1;

}\*/

return groupedMatches;

}}

Інші важливі класи та методи наведені в Додатку Ж.

Клас, який відповідає за автоматичні сповіщення, які приходять із сервера.  
public class NotificationService extends IntentService {  
   
 @Override  
 protected void onHandleIntent(Intent intent) {  
 Date currentDate = new Date();  
 String sportName = ResourcesSolver.loadSportType(this);  
 modelImp.loadMatches(currentDate, sportName, new EventsLoadingListener() {  
 @Override  
 public void onSuccess(List<League> leagues) {  
 if (leagues != null) {  
 generateNotifications(leagues);  
 } }  
  
 @Override  
 public void onFailure(String message) {  
 //hide all errors  
 }  
 });  
 }  
  
 private void generateNotifications(List<League> leagues) {  
  
 for (League league : leagues) {  
 List<Match> matches = league.getMatches();  
 for (Match match : matches) {  
 if (match.isActive()) {  
 generateInfoForMatch(match);  
 }  
 } }  
 }  
 private void generateInfoForMatch(Match match) {  
 String teamHome = match.getTeamHome();  
 String teamAway = match.getTeamAway();  
 String event = match.getUpdateScore();  
 String notificationInfo = null;  
 String notificationMessage = String.format(«%s - %s», teamHome, teamAway);  
 switch (event){  
 case «1»:  
 notificationInfo = String.format(«%s scored goal», teamHome);  
 break;  
 case «2»:  
 notificationInfo = String.format(«%s scored goal», teamAway);  
 break;  
 case "3":  
 notificationInfo = String.format("Both team scoed goal");  
 break;  
 default: break;  
 }  
 if (notificationInfo != null){  
 showNotification(notificationMessage, notificationInfo);  
 } }  
 private void showNotification(String notificationMessage, String notifInfo) {  
 NotificationCompat.Builder builder =  
 new NotificationCompat.Builder(this)  
 .setSmallIcon(R.drawable.logo\_l2s)  
 .setContentTitle(getString(R.string.Live2Sport))  
 .setContentInfo(notifInfo)  
 .setContentText(notificationMessage);  
 Intent intent = new Intent(this, MainDrawerActivity.class);  
 //intent.putExtra(Main.IS\_LOAD\_FROM\_NOTIFICATION, true);  
 PendingIntent pi = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);  
 builder.setContentIntent(pi);  
 NotificationManager mNotificationManager =  
 (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
 mNotificationManager.notify(NOTIFICATION\_ID, builder.build());  
 }  
  
 public static void setServiceAlarm(Context context, boolean isOn){  
 Intent i = new Intent(context, NotificationService.class);  
 PendingIntent pi = PendingIntent.getService(context, 0, i, 0);  
 AlarmManager alarmManager = (AlarmManager)context.getSystemService(Context.ALARM\_SERVICE);  
 alarmManager.cancel(pi);  
 if (isOn){  
 alarmManager.setRepeating(AlarmManager.RTC, System.currentTimeMillis(),  
 PrefsHelper.getAutomaticRefreshInterval()\*1000,//convert to miliseconds  
 pi);  
 }else {  
 alarmManager.cancel(pi);  
 pi.cancel();  
 }  
 }}

Клас який відповідає за асинхронне завантаження та опрацювання даних спортивних подій  
public class FetchItemsTask extends AsyncTask<Object,Void,List<League>> {  
 public static final String TAG = "FetchItemsTask";  
  
 private Context context;  
 private ItemFetcherListener listener;  
 private String errorMessage = null;  
  
 public FetchItemsTask(Context context){  
 this.context = context;  
 }  
  
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 errorMessage = null;  
 }  
  
 @Override  
 protected void onPostExecute(List<League> leagues) {  
 if (errorMessage == null){  
 if (listener != null)  
 listener.onItemsLoaded(leagues);  
 }else{  
 if (listener!= null)  
 listener.onFailure(errorMessage);  
 }  
 errorMessage = null;  
 }  
  
 @Override  
 protected List<League> doInBackground(Object... params) {  
 errorMessage = null;  
 Date date = (Date)params[0];  
 String sportType = (String)params[1];  
 @SuppressWarnings("deprecation")  
 String sdate = new SimpleDateFormat("yyyy-MM-dd").format(date);  
 List<Match> matchItems = null;  
 try {  
 matchItems = new TvEventsFetchr().fetchMatches(context, sdate, sportType);  
 List<League> leagues = MatchLab.groupMatchesByLeagues(context, matchItems);  
 return (leagues != null ? leagues : new ArrayList<League>());  
 }catch (Exception ex){  
 errorMessage = ex.getMessage();  
 Log.d(TAG, ex.getMessage());  
 }  
 return null;  
 }  
  
 public void setListener(ItemFetcherListener listener) {  
 this.listener = listener;  
 }  
  
 public interface ItemFetcherListener {  
 void onItemsLoaded(List<League> leagues);  
 void onFailure(String message);  
 }  
}

Клас моделі MVP, який предаставляє роботу головного списку спортивних подій. Він отримує команди від презентера, здійснює обробку інформації. В ньому присутні методи, які виконують завантаження даних спортивних подій, відзначення команд та ліг як вибрані, здійснення пошуку.

public class MatchesListModelImp {  
 private Context context;  
 private FetchItemsTask fetchItemsTask;  
  
 private boolean isExplore;  
  
 public MatchesListModelImp(Context context, boolean isExplore){  
 this.context = context;  
 this.isExplore = isExplore;  
 }  
  
 public void loadMatches(Date date, String sportType, final EventsLoadingListener listener){  
 fetchItemsTask = new FetchItemsTask(context);  
 fetchItemsTask.setListener(new FetchItemsTask.ItemFetcherListener() {  
  
 @Override  
 public void onItemsLoaded(List<League> leagues) {  
 if (isExplore) {  
 if (PrefsHelper.getExploreStarred())  
 leagues = getChosenLeagues(leagues);//filter  
 leagues = starredChosenEvents(leagues);  
 }  
 if (!isExplore)  
 leagues = getChosenLeagues(leagues);//filter  
 if (listener != null)  
 listener.onSuccess(leagues);  
 }  
  
 @Override  
 public void onFailure(String message) {  
 if (listener != null)  
 listener.onFailure(message);  
 }  
 });  
 fetchItemsTask.execute(date, sportType);  
 }  
  
 private List<League> starredChosenEvents(List<League> leagues) {  
 if (leagues == null)  
 return leagues;  
 ChosenSports chosenSports = ChosenService.get().getChosenSports();  
 String sportType = ResourcesSolver.loadSportType(context);  
 ChosenSportLeagues sportLeagues = chosenSports.getSportLeagues(sportType);  
 if (sportLeagues == null)  
 return leagues;//empty  
 List<ChosenLeague> leagueList = sportLeagues.getLeagueList();  
 if (leagueList == null)  
 return leagues;//empty  
 for (int i = 0; i < leagues.size(); i++){  
 League league = leagues.get(i);  
 ChosenLeague currentLeague = sportLeagues.getleague(league.getName());  
 if (currentLeague == null){  
 continue;  
 }else if (!currentLeague.isChosen()&&  
 (currentLeague.getTeamsSet() == null ||  
 currentLeague.getTeamsSet().size() == 0)) {  
 continue;  
 }else {  
 league.setIsStarred(currentLeague.isChosen());  
 for (int j = 0; j < league.getMatches().size(); j++) {  
 Match match = league.getMatches().get(j);  
 boolean isTeamHomeStarred = currentLeague.isTeamStarred(match.getTeamHome());  
 boolean isTeamAwayStarred = currentLeague.isTeamStarred(match.getTeamAway());  
 match.setIsTeamHomeStarred(isTeamHomeStarred);  
 match.setIsTeamAwayStarred(isTeamAwayStarred);  
 }  
 }  
 }  
 return leagues;  
 }  
  
 private List<League> getChosenLeagues(List<League> leagues){  
 if (leagues == null)  
 return null;  
 ChosenSports chosenSports = ChosenService.get().getChosenSports();  
 String sportType = ResourcesSolver.loadSportType(context);  
 ChosenSportLeagues sportLeagues = chosenSports.getSportLeagues(sportType);  
 if (sportLeagues == null)  
 return new ArrayList<>();//empty  
 List<ChosenLeague> leagueList = sportLeagues.getLeagueList();  
 if (leagueList == null)  
 return new ArrayList<>();//empty  
 for (int i = 0; i < leagues.size(); ){  
 League league = leagues.get(i);  
 ChosenLeague currentLeague = sportLeagues.getleague(league.getName());  
 if (currentLeague == null){  
 leagues.remove(i);  
 continue;  
 }else if (!currentLeague.isChosen()&&  
 (currentLeague.getTeamsSet() == null ||  
 currentLeague.getTeamsSet().size() == 0)) {  
 leagues.remove(i);  
 continue;  
 }else {  
 if (!currentLeague.isChosen()) {  
 for (int j = 0; j < league.getMatches().size(); ) {  
 Match match = league.getMatches().get(j);  
 boolean isExists = currentLeague.getTeamsSet().contains(match.getTeamHome())  
 || currentLeague.getTeamsSet().contains(match.getTeamAway());  
 if (!isExists) {  
 league.getMatches().remove(j);  
 continue;  
 }  
 j++;  
 }  
 if (league.getMatches().size() == 0) {  
 leagues.remove(i);  
 continue;  
 }  
 }  
 }  
 i++;  
 }  
 return leagues;  
 }  
  
 public void cancellLoading(){  
 fetchItemsTask.setListener(null);  
 }  
  
 public void processSearch(List<League> originalLeagues, String query, final EventsLoadingListener listener) {  
 if (query == null) {  
 if (listener != null)  
 listener.onSuccess(originalLeagues);  
 return;  
 }  
 List<League> leagues = new ArrayList<>();  
 for (League league : originalLeagues){  
 if (league.getName().toLowerCase().contains(query.toLowerCase())){  
 leagues.add(league);  
 }else{  
 List<Match> matches = new ArrayList<>();  
 for (Match match : league.getMatches()){  
 if (match.getTeamAway().toLowerCase().contains(query.toLowerCase()) ||  
 match.getTeamHome().toLowerCase().contains(query.toLowerCase()) ){  
 matches.add(match);  
 }  
 }  
 if (matches.size() > 0){  
 leagues.add(new League(league.getName(), matches, league.getCountryFlagPath(), league.getCoeff()));  
 }  
 }  
 }  
 if (listener != null)  
 listener.onSuccess(leagues);  
 }  
}

# Додаток З. Статичний аналіз коду за допомогою Android Studio

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Шліхутка Б. Т. | Підпис |  | Статичний аналіз вихідного коду за допомогою Android Studio | Сторінка |
| Перевірив | Сердюк П. В. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

# Додаток И. Список багів та зауважень від замовника

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Опис | Bug resolution |
| 1 | 8 червня 2015 | Некілкабельна дата в списку спортивних подій | FIXED |
| 2 | 8 червня 2015 | Зробити кнопку оновити як на LiveScore, яка б підтягувала нові дані | FIXED |
| 3 | 8 червня 2015 | Підправити відображення прапорів країн | FIXED |
| 4 | 9 червня 2015 | Немає риски між спортивними подіями списку | FIXED |
| 5 | 9 червня 2015 | Замінити N/A на “? : ?” при відображенні незіграних матчів. | FIXED |
| 6 | 10 червня 2015 | Програма повинна завантажувати дані більшої кількості попередніх днів | FIXED |
| 7 | 23 червня 2015 | Відображення матчів, які не почались. змінити «? -?» на знаки питання один під одним | FIXED |
| 8 | 23 червня 2015 | Для матчу, який розпочався показувати не знаки питання, а поточний рахунок та хвилину | FIXED |
| 9 | 2 липня 2015 | Програма аварійно завершується при завантаженні даних бейсболу | FIXED |
| 10 | 6 липня 2015 | Порядок відображення спортивних подій неправильний. Спочатку повинні йти ті що не почалися і ті, які ще йдуть. | FIXED |
| 11 | 23 липня 2015 | Не відображається посилання на watch live, для подій, що йдуть в деталях. | FIXED |

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | 17 липня 2015 | Всі ліги зразу відкриті в списку | NOT A BUG |
| 13 | 17 липня 2015 | Додати американський футбол | FIXED |
| 14 | 17 липня 2015 | В детальній інформації про матч повинен відображатись детальний рахунок, як такий 4:0(1:0 3:0) | FIXED |
| 15 | 17 липня 2015 | Не оновлюються деталі спортивної події | FIXED |
| 16 | 21 липня 2015 | Програма аварійно завершується, в зв’язку з переповненням пам’яті | FIXED |
| 17 | 10 серпня 2015 | Американський футбол у списку видів спорту останній - він має бути на тому місці що й на сайті | FIXED |
| 18 | 10 серпня 2015 | Хокей в не тому місці і прапорці на хокеї Ісландії стоять | FIXED |
| 19 | 12 вересня 2015 | Деталі матчу - немає пробілу немає пробілу між закриваючою дужкою і вертикальною рискою. | FIXED |
| 20 | 12 вересня 2015 | Немає в списку MMA, eSports | FIXED |
| 21 | 20 листопада 2015 | Якщо є кінцевий результат – значить матч закінчився. А це поля scores - його і треба виводити в детальніше про матч – так і має бути по вигляду. А не писати в стовпчик результати по таймах. | FIXED |
| 22 | 14 січня 2016 | Замінити логотип для контори Марафон. | FIXED |

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | 14 січня 2016 | Змінити посилання для контори Марафон | FIXED |
| 24 | 14 січня 2016 | Не відображається посилання на трансляцію контори Марафон | FIXED |

# Додаток К. Результати тестування першої версії перед релізом

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Опис | Bug resolution |
| 1 | 31 грудня 2015 | Список подій не завжди завантажується, хоча підключення до інтернету присутнє. | FIXED |
| 2 | 31 грудня 2015 | Час матчів в додатку та на сайті відрізняються на одну годину. | NOT A BUG |
| 3 | 31 грудня 2015 | Різна кількість ставок на сайті та в додатку. | NOT A BUG |
| 4 | 31 грудня 2015 | Деталі матчу – список букмекерів. Цифри ставок з’їжджають вниз, замале поле для відображення. | FIXED |
| 5 | 31 грудня 2015 | Різні дані на сайті і в додатку, деякі матчі відсутні. | FIXED |
| 6 | 31 грудня 2015 | Немає можливості вибрати дату, матчів, потрібно гортати весь час | FIXED |
| 7 | 31 грудня 2015 | Немає іконок на всі види спорту | NOT A BUG |
| 8 | 31 грудня 2015 | Рекомендація на ставку повинна бути після завершення матчу? | NOT A BUG |
| 9 | 31 грудня 2015 | В видах спорту в яких немає рахунку, так і мають бути знаки питання? | FIXED |