Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням
комп’ютерних та видавничих технологій
Чубей О.О. /_____/
підпис
«___» _____ 2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

освітньо-кваліфікаційного рівня «молодший спеціаліст»

зі спеціальності 122 «Комп’ютерні науки»

на тему: «Мобільний органайзер керівника гуртка»

Студент групи Кн-41 Солонинка А.В. _____
(підпис)

Керівник проєкту Посвятовська О.Б. _____
(підпис)

Консультанти:

з техніко-економічного обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Кульчинська Н.З. _____
(підпис)

Тернопіль – 2022

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням комп’ютерних та видавничих технологій
Чубей О.О. /_____/
підпис
«___» _____ 2021 р.

ЗАВДАННЯ

на дипломне проектування
на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»
студенту Солонинці Андрію Володимировичу
(прізвище, ім’я та по-батькові студента)

1. Тема проекту «Мобільний органайзер керівника гуртка »
затверджена наказом по коледжу від “___”_____202_ р., №___
2. Термін здачі студентом завершеного проекту “___” _____202_ р.
3 Вихідні дані до проекту: Мобільні застосунки – органайзери, їхні функціональні можливості, вимоги до додатків такого типу.
4. Перелік питань, які повинні бути розроблені в проекті:
а) основна частина: огляд існуючих рішень і постановка завдання, проєктування, реалізація та тестування системи.
б) техніко-економічне обгрунтування: аналіз ринку, розрахунок витрат на проєктування, обґрунтування необхідності розробки.
5. Перелік графічного матеріалу _____
_____
6. Консультанти проекту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
З техніко-економічного обгрунтування	Меленчук Л.І. (вчена ступінь,звання П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проектування

№	Найменування етапу	Терміни	
		початку	завершення
1	Вибір теми, ознайомлення з вимогами до дипломного проєктування	20.09.2021	01.10.2021
2	Огляд типових рішень та написання відповідного розділу ПЗ	02.12.2021	28.01.2021
3	Дослідження технологій реалізації та написання відповідного розділу ПЗ	28.01.2022	17.02.2022
4	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написання відповідного розділу ПЗ	16.02.2022	03.03.2022
5	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ	03.03.2022	15.03.2022
6	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	17.03.2022	14.04.2022
7	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	15.04.2022	03.05.2022
8	Доопрацювання модулів	04.05.2022	18.05.2022
9	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	18.05.2022	25.05.2022
10	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	26.05.2022	04.06.2022
11	Робота над оформленням пояснювальної записки	04.06.2022	12.06.2022
12	Попередній захист дипломного проєкту, доопрацювання	15.06.2022	15.06.2022
13	Підготовка до захисту дипломного проєкту	16.06.2022	24.06.2022
14	Захист дипломного проекту	25.06.2022	26.06.2022

7. Дата видачі завдання “___” _____ 2021 р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Мобільний додаток «Органайзер» для смартфонів на базі Android. Дипломний проєкт. Солонинка Андрій Володимирович. Галицький фаховий коледж імені В’ячеслава Чорновола, відділення комп’ютерних та видавничих технологій, спеціальність 122 «Комп’ютерні науки», ГФК, 2022.Сторінок – 92, рисунків – 32, додатків – 2.

Об’єкт дослідження – мобільні застосунки «органайзери», технології та інструментарій створення додатків для мобільних пристроїв.

Метою дипломного проєкту є створення програмного продукту для смартфонів, які використовують операційну систему Android. Програма дозволить користувачу зберігати дані про студента та майбутні лекції. Вести облік даних, відмічати присутність та виставляти оцінки.

Для реалізації поставленої задачі було використано велику кількість різноманітних інструментів, шаблонів доступних в середовищі програмування та мову Java.

Результатом розробки стала завершена робота, яка готова до використання.

СИСТЕМА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ JAVA, ANDROID STUDIO, ANDROID, ФУНКЦІЯ, БАЗА ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ER-ДІАГРАМА.

Abstract

Mobile application "Organizer" for smartphones based on Android. Diploma project. Solonynka Andriy Volodymyrovych. Galytsky Vocational College named after Vyacheslav Chornovil, Department of Computer and Publishing Technologies. Specialty 122 "Computer Science". GVC, 2022. Pages – 92, Figures – 32, Appendices – 2.

The object of research is mobile applications "organizers", technologies and tools for creating applications for mobile devices.

The aim of the diploma project is to create a software product for smartphones that use the Android operating system. The program will allow the user to save data about the student and future lectures. Keep records of data, mark attendance and make assessments.

To implement this task, a large number of different tools, templates available in the programming environment and the Java language were used.

The result of the development was a completed work that is ready for use.

SYSTEM, JAVA SOFTWARE, ANDROID STUDIO, ANDROID, FUNCTION, DATABASE, USER INTERFACE, ER-DIAGRAM.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення.

GUI – graphical user interface.

MBaaS – mobile backend as a service.

SOA – service-oriented architecture.

IDE – integrated drive electronics.

UML – unified modeling language.

ПК – персональний комп’ютер.

ОС – операційна система.

VM – virtual machine.

WORA – write once, run anywhere.

JVM – java virtual machine.

ВСТУП

Технологічний прогрес стрімко рухається вперед та з кожним роком приносить у наше життя стільки нового, що ми вже не можемо уявити своє існування без них. Мета полягає в тому, щоб зробити наше життя комфортним та зручним.

Попереднє планування допомагає покращити результат будь-якої події, як особистої, так і професійної.

Спочатку всі користувались простими записниками. З розвитком інформаційних технологій на зміну блокнотам спочатку прийшли електронні записники, а потім кишенькові комп'ютери, комп'ютерні програми та онлайн-нотатки, які можуть мати допоміжні функції: нагадування про заплановані події, захист та синхронізацію інформації.

Перехід від звичайного планування до планування за допомогою мобільних програм «органайзерів» надає ряд переваг:

– на стан навколишнього середовища позитивно вплине зменшення використання паперу;

– можливість використання електронного записника в будь який момент часу;

– нагадування про заплановані події.

Використання органайзера у вигляді мобільного додатку значно збільшує його ефективність та відповідає сучасним вимогам організації діяльності.

Отже, виходячи з високої популярності мобільних органайзерів в сучасному світі, основною метою даної роботи є створення власної реалізації мобільного органайзера на базі системи Андроїд.

З огляду на це, метою даного проекту є реалізація мобільного органайзера керівника гуртка.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз діяльності керівника музичного гуртка

Основна діяльність керівника музичного гуртка полягає в забезпеченні якісного освітнього процесу учасників.

Важливою складовою цього є проведення занять, що зазвичай відбувається у форматі звичайних уроків (або пар) зі спеціалізованих дисциплін.

Фактично, діяльність керівника музичного гуртка проходить по принципу діяльності вчителя в школі або вищому навчальному закладі, тобто, учасники гуртка приходять на уроки (пари), набувають відповідних знань, вмінь та навичок, отримують оцінювання результатів своєї навчальної діяльності, також керівник контролює відвідуваність, тощо.

1.2 Актуальність використання мобільних додатків

Мобільний додаток – програмне забезпечення, яке призначене для роботи з мобільними пристроями. Мобільні програми зазвичай відрізняються від програм, призначених для запуску на настільних комп'ютерах, також веб-програми, які працюють в мобільному браузері, а не на самому мобільному пристрої.

Спочатку програми були призначені для підвищення продуктивності, таких як електронна пошта, календар і бази даних контактів, але загальний попит на програми призвів до стрімкого розширення в інших сферах, таких як мобільні ігри, автоматизація виробництва, GPS і послуги на основі визначення місцезнаходження, відстеження замовлень і квитків покупки, тож тепер доступні мільйони програм. Спочатку для підвищення продуктивності розроблялися програми, такі як електронна пошта, календар та бази даних контактів, але суспільний попит на програми призвів до швидкого розширення в інших областях, таких як мобільні ігри, автоматизація виробництва, GPS та

служби позиціонування. Відстеження замовлень та квитків. Тепер доступні покупки та мільйони програм. Багатьом додаткам потрібен доступ до Інтернету. Додатки зазвичай завантажуються з магазинів додатків, які є типом платформ цифрового розповсюдження.

Програми в цілому поділяються на три типи: нативні програми, гібридні та веб-програми. Нативні програми розроблені спеціально для мобільної операційної системи, як правило, iOS або Android. Веб-програми написані на HTML5 або CSS і зазвичай запускаються через браузер. Гібридні програми створюються використовуючи такі веб-технології, як JavaScript, CSS і HTML5, та функціонують як веб-програми, замасковані в нативному контейнері [1].

Типи

Мобільні програми можна класифікувати за різними методами. Поширеною схемою є розрізнення нативних, веб-додатків та гібридних програм.

Нативні додатки

Усі програми, які орієнтовані на певну мобільну платформу, називаються нативними програмами. Додаток, призначений для пристроїв Android, не працюватиме на пристроях з операційною системою IOS. Тому компанії більшість програм розробляють для кількох платформ.

Під час розробки нативних програм професіонали використовують найкращі в своєму класі модулі інтерфейсу користувача. Цим можна пояснити гарну продуктивність, послідовність і великий досвід роботи з користувачем.

Основною метою створення таких програм є забезпечення найкращої продуктивності для конкретної мобільної операційної системи.

Веб-додаток

Веб-додаток реалізовано за допомогою стандартних веб-технологій HTML, CSS і JavaScript. Доступ до Інтернету зазвичай потрібен для належної поведінки або можливості використання всіх функцій порівняно з

використанням в автономному режимі. Більшість, якщо не всі дані користувача зберігаються в хмарі.

Продуктивність цих програм подібна до веб-програми, що працює у браузері, яка може бути помітно повільнішою, ніж еквівалентна рідна програма. Він також може не мати такого рівня функцій, як рідний додаток.

Гібридний додаток

Концепція гібридного додатка – це поєднання нативних і веб-додатків. В цю категорію входять програми, розроблені за допомогою Apache Cordova, React Native, Flutter, Sencha Touch, Xamarin та інших.

Вони створені для підтримки веб та нативних технологій на кількох платформах. Крім того, ці програми легше та швидше розробляються. Це передбачає використання єдиної кодової бази, яка працює в кількох мобільних операційних системах.

Незважаючи на ці переваги, гібридні програми показують найгіршу продуктивність. Програми часто відрізняються за зовнішнім виглядом та функціональністю в різних мобільних операційних системах.

Професійне керування мобільними додатками допомагає захищати, дані що використовуються ними.

Розробка мобільного додатка – це процес, під час якого створюється додаток для мобільних пристроїв. Ці програми можуть бути попередньо встановлені на телефонах під час виробничих платформ або розгорнуті як веб-застосунки з використанням обробки на стороні сервера або на стороні клієнта (наприклад, JavaScript), щоб забезпечити «програмний» досвід роботи у веб-браузері. Розробникам прикладного програмного забезпечення необхідно враховувати різні розміри екранів, апаратні функції та конфігурації через гостру конкуренцію у мобільному програмному забезпеченні та змін на кожній платформі. Розробка мобільних програм постійно зростає, приносячи дохід і робочі місця. За оцінками аналітиків за 2018 рік, у ЄС є 529 000 робочих

місць у сфері прямої економіки додатків, тоді як 28 членів (включаючи Великобританію), 60 відсотків з яких є розробниками мобільних додатків [2].

Як частина процесу розробки проектування мобільного інтерфейсу (GUI) також є важливим при розробці мобільних додатків. Мобільний інтерфейс користувача розглядає обмеження, контексти, екран, введення і мобільність як структуру дизайну. Користувач часто є інтерфейсом зі своїм пристроєм, інтерфейс містить як апаратні, так і програмні компоненти. Введення користувача дозволяє їм маніпулювати системою, а виведення пристрою дозволяє системі вказувати на наслідки користувацьких маніпуляцій. Обмеження дизайну мобільного інтерфейсу користувача включають обмежений фокус і форм-фактори, такі як розмір екрану мобільного пристрою для рук користувача. Контексти мобільного інтерфейсу повідомляють про дії користувача, такі як розташування та розклад, які можна переглянути, коли користувач взаємодіє з мобільним додатком. Загалом, мета дизайну мобільного інтерфейсу полягає в тому, щоб в основному мати зрозумілий і інтуїтивно зрозумілий інтерфейс. Функціональність підтримується платформами корпоративних мобільних програм або вбудованими середовищами розробки (IDE).

Інтерфейси мобільних програм призначаються мобільним серверним системам для підтримки доступу до корпоративних систем. Мобільний сервер полегшує маршрутизацію даних, безпеку, аутентифікацію, авторизацію, роботу в автономному режимі та оркестрацію послуг. Ця функціональність підтримується комбінацією компонентів проміжного програмного забезпечення, включаючи сервер мобільних програм, мобільний сервер як послугу (MBaaS) та сервісно-орієнтовану інфраструктуру (SOA) [3].

1.3 Актуальність використання персонально органайзера

Персональний органайзер, щоденник, журнал дат, щоденник, планувальник дня, персональний аналоговий помічник, планувальник книг,

планувальник на рік або розпорядок дня (від лат. agenda – речі, які потрібно робити) – це невелика книжка або папка, призначена для портативного використання. Зазвичай він містить щоденник, календар, адресну книгу, чистий папір та інші розділи [4]. Організатор є персональним інструментом і може також містити сторінки з корисною інформацією, наприклад, карти та телефонні коди. Це пов'язано з окремими настільними канцелярськими товарами, які мають одну або кілька однакових функцій, наприклад, календарі зустрічей, ролодекси, блокноти та альманахи.

Їх іноді називають філофаксом на честь британської компанії Filofax, яка виробляє популярний асортимент персональних гаманців-органайзерів.

Наприкінці 20-го століття персональні органайзери з паперу та папки почали замінюватися електронними пристроями, такими як персональні цифрові помічники, програмне забезпечення для керування персональною інформацією та онлайн-органайзери. Цей процес прискорився на початку 21 століття з появою смартфонів, планшетних комп'ютерів, розумних годинників і різноманітних мобільних додатків.

Електронний органайзер (або електричний органайзер) – це невеликий комп'ютер розміром із калькулятором, часто з вбудованою програмою щоденника та іншими функціями, такими як адресна книга та календар. Зазвичай він має невелику буквено-цифрову клавіатуру та РК-екран з одним, двома або трьома рядками. Електронний щоденник або органайзер був винайдений індійським бізнесменом Сатьяном Пітрода в 1975 році. Його вважають одним із перших піонерів портативних обчислень завдяки винаходу електронного щоденника в 1975 році.

Вони були дуже популярні, особливо серед бізнесменів у 1990-х, але через появу персональних цифрових помічників, а пізніше смартфонів у 2000-х і 2010-х роках відповідно, Обидва мають великий набір функцій, електронні органайзери сьогодні здебільшого використовуються для дослідницьких цілей. Однією з основних тем досліджень є вивчення того, як електроніка

зможе допомагати людям з обмеженими інтелектуальними можливостями використати цей тип обладнання у повсякденному житті. Нещодавно електронні органайзери використовувалися для підтримки людей з хворобою Альцгеймера, щоб мати візуальне уявлення про графік.

1.4 Огляд існуючих рішень

Станом на 2022 рік на просторах інтернету можна знайти багато схожих за функціоналом програм. Перед постановкою задачі на розробку слід оглянути обрану нішу ринку програмного забезпечення. Порівняльний аналіз проводиться на прикладі мобільних додатків схожої тематики.

Для аналізу було обрано такі три проекти: aCalendar, Google Calenda, TickTick.

На рисунку 1.1 подано зображення інтерфейсу програми aCalendar.

Він має дуже незвичайний дизайн і унікальну навігацію, якою мало хто може похвалитися. Пересування додатком здійснюється за допомогою свайпів по горизонталі і вертикалі, що дуже зручно.

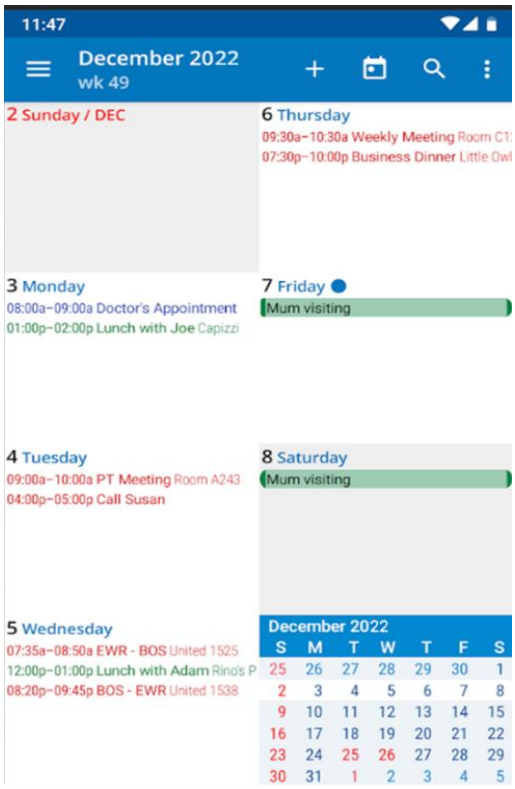

Рисунок 1.1 – Інтерфейс програми «aCalendar»

Розробник даного програмного продукту компанія Tapir Apps.

Особливості додатку aCalendar, що можна віднести до переваг:

– інтуїтивна навігація з плавними переходами;

– день, тиждень та місяць календаря;

– віджет на повний екран;

– дні народження з фотографіями із записника та редагування;

– використовує рідний календар android та синхронізацію з ним;

– не розряджає акумулятор;

– QR штрих-код для обміну подіями;

– 12/24-годинний формат у налаштуваннях системи годин;

– оптимізація відображення тексту та покращення розриву рядка;

– безкоштовний;

– відсутня реклама;

– персоналізація зовнішнього вигляду програми, наприклад, можливість вибору різних тем або кольорів;

– можливість використання великої кількості мов. Це дозволяє використовувати його будь-якою мовою, на ваш вибір і робить цей додаток зручним для великої кількості користувачів по всьому світу;

– оптимізований під різні види пристроїв;

– додаток потребує дуже малий об’єм пам’яті. Менший розмір означає, що є набагато більше місця на пристрої для інших додатків, а також інших даних, таких як фотографії та музика.

До недоліки даної програми можна віднести:

– переповнення інтерфейсу надлишковими функціями; мінімалістичний дизайн має більш естетичний вигляд, в результаті, інтерфейс стає простим та зручним у використанні;

– в додатку немає можливості визначити стан задачі, такі як «виконане», «в процесі виконання», «відкладені»;

– немає можливості розраховувати час до події та виводити його на календарі, оскільки нагадування про те скільки часу залишилось до певної події досить корисна функція;

– відсутня можливість керування списком присутніх на власних подіях. В додатку немає функцій додавання чи видалення людей, а також перевірка статусу гостей.

Проведений аналіз програмного продукту aCalendar свідчить про його актуальність в наслідок великої кількості переваг, проте він має свої недоліки, які відштовхують частину користувачів.

Основним по популярності можна вважати Google Calendar.

На рисунку 1.2 подано зображення інтерфейсу програма Google Calendar.

Цей органайзер залучив багатьох користувачів завдяки простоті використання, особливо для командної роботи, оскільки він дозволяє завжди знати про всі важливі заплановані корпоративні заходи, незалежно від того, де і як далеко знаходяться ваші співробітники один від одного.

Тож для вас працює не тільки ваш особистий календар, а й календарі всіх ваших співробітників. Просто накладаючи ці календарі один на одного, дуже легко організовувати спільні зустрічі, конференції тощо.

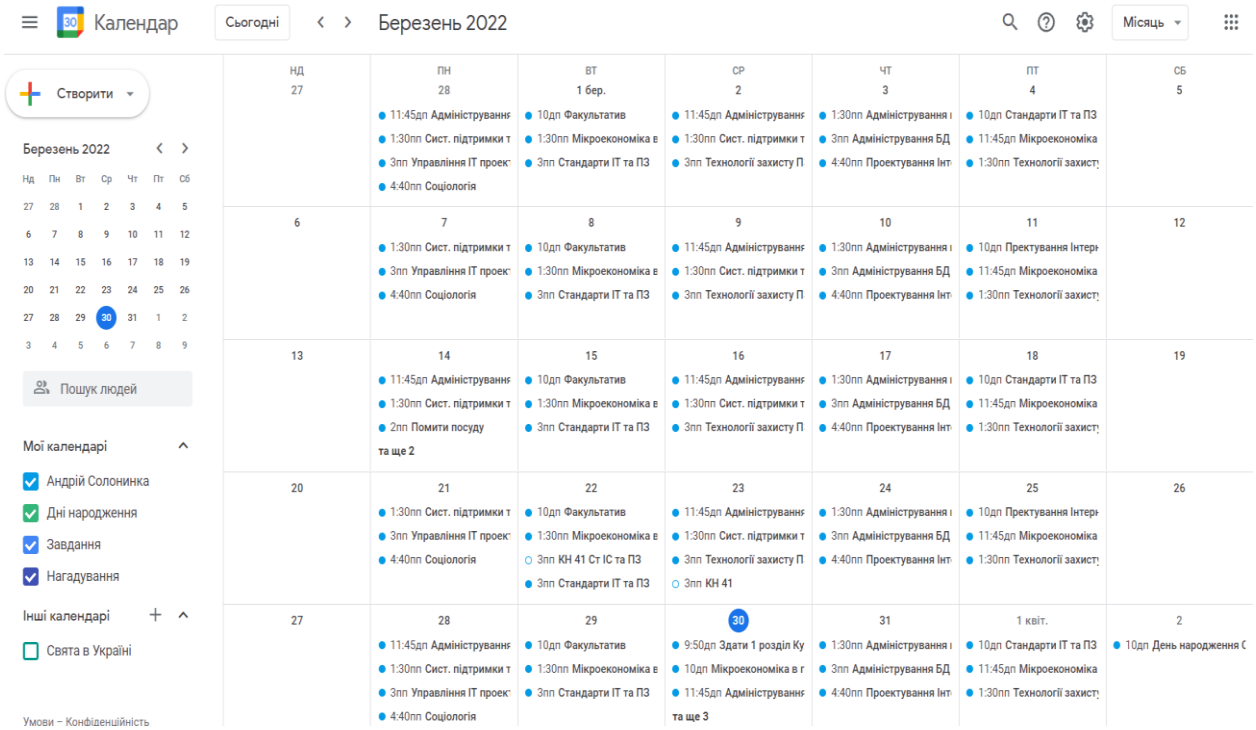

Рисунок 1.2 – Інтерфейс програми «Google Calendar»

Розробником даного програмного продукту є компанія Google.

Особливості додатку Google Calendar, що можна віднести до переваг:

– відсутність реклами (вона в додатку може бути надто відволікаючою та нав'язливою), додатки та блоги без реклами краще сприймаються користувачем, є більш естетичними та приємними при використанні, що сприяє тому щоб контент краще виділявся;

– додаток є повністю безкоштовним;

– присутня функція розрахунку часу до подій та виведення його в сповіщеннях;

– можливість персоналізувати зовнішній вигляд, що буде сприяти зручному користування додатком;

– здатність працювати на різних типах пристроїв;

– наявність пошуку. В тому разі коли потрібно знайти щось конкретне, є можливість пошуку по ваших даних за допомогою ключових слів;

– експорт даних. Можливість експорту даних з додатку на особисту електронну пошту;

– нагадування про події та інші види нагадувань;

– приємний та зручний інтерфейс.

До недоліки даної програми можна віднести:

– Відсутність віджетів.

– Заходи не можна копіювати.

– Відсутні підзадачі.

– Відсутність швидкого переходу.

– Відсутній табличний формат.

Google Calendar є основним програмним продуктом на ринку електронних органайзерів, проте він не має спеціалізованого функціоналу і не має можливості кастомізації.

Наступним для аналізу обрано застосунок TickTick.

На рисунку 1.3 подано зображення інтерфейсу програма TickTick.

TickTick – це програма з наявністю списку справ, що швидко розвивається, яка пропонує широку функціональність практично на кожній платформі, яку ви можете собі уявити. Це потужний та гнучкий інструмент для планування поточних справ, що дозволяє складати графік робіт для себе та для всього колективу. Завдяки обробці природної мови завдання можна швидко додавати. Настільна версія також пропонує універсальні комбінації клавіш, а мобільні пристрої мають закріплені сповіщення та віджети, які дозволяють швидко додавати завдання, перш ніж повернутися до того, що ви робите. Завдання можна організувати за списками, тегами та термінами виконання, і ви навіть можете додати підзавдання до будь-якого завдання.

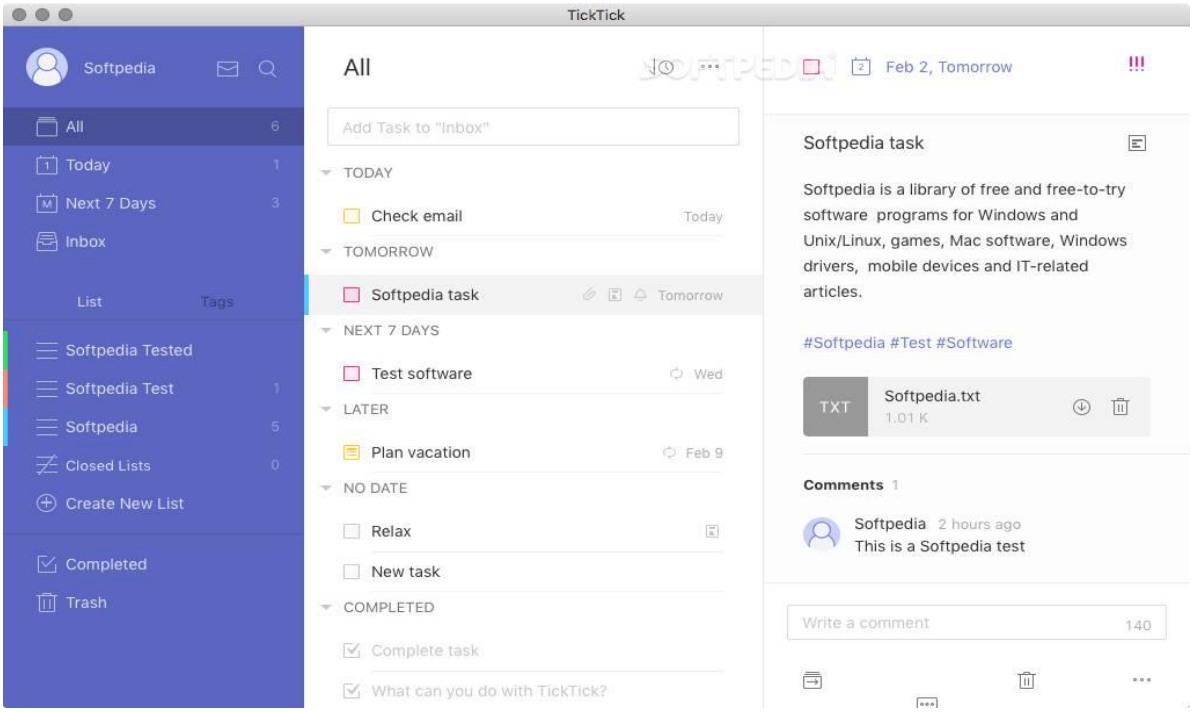

Рисунок 1.3 – Інтерфейс програми «TickTick»

Розробником даного програмного продукту є компанія Appest Limited.

Особливості додатку TickTick, що можна віднести до переваг:

– створення та відстеження завдань;
– приємний дизайн застосунку;
– перегляд завдань у календарі або списках;
– пошук та система тегів;
– коментування задач;
– управління задачами свайпами;
– спільна робота;
– трекер звичок;
– встановлення дедлайнів та пріоритетності завдань;
– створення регулярних завдань;
– вкладення pdf та фото до списків завдань;
– перегляд статистики;
– створення завдань із електронних листів;
– перегляд статистики продуктивності;

– кастомізований лист завдань та підзадач, чек лист, швидкий блокнот, який можна фільтрувати та об'єднувати у тематичні папки;

– інтеграція з популярними сервісами: Gmail, Outlook, Slack та іншими.

До недоліків даної програми можна віднести:

– відсутня українська мова;

– обмежений функціонал в безкоштовній версії;

– маленький вибір мов для користування;

– відсутність можливості встановлення цілей.

TickTick досить маловідомий, але він швидко розвивається і знаходить свою аудиторію. Проте, не дивлячись на всі його переваги, це скоріше «календар цілей», аніж «електронний органайзер».

1.5 Постановка задачі

Задачею даного проекту є проектування, розробка мобільного застосунку, призначенням якого є зручний облік даних вчителя. Проаналізувавши готові проекти схожої тематики, відмітивши основні плюси та мінуси, остаточно були сформульовані вимоги щодо успішного мобільного застосунку.

Завданнями при створенні мобільного органайзера має бути:

Перед розробкою даного проекту, були поставлені такі завдання:

– ознайомитися з основними правилами і рекомендаціями з розробки й створення мобільних застосунків і беззастережно дотримуватися їх у своїй практиці;

– реалізувати зручний інтерфейс;

– не перевантажувати застосунок не потрібним функціоналом;

– надавати можливість для обліку;

– реалізувати наявність розкладу занять;

– реалізувати можливість виставлення оцінок;

– реалізувати наявність в кожного студента статусу;

– реалізувати можливість зберігання та редагування даних;

– реалізувати можливість перегляду списку студентів.

Реалізований програмний продукт має бути інтуїтивно зрозумілим в користуванні та забезпечувати функціонування достатньої кількості функцій.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Аналіз варіантів використання органайзера

Діаграма варіантів використання – це представлення взаємодії користувача з системою, що демонструє зв’язок між користувачем та різними варіантами використання, у яких бере участь користувач.

Діаграма варіантів використання здатна розрізняти різні типи користувачів системи та різні варіанти використання та часто супроводжується іншими типами діаграм.

У той час як сам варіант використання може детально вивчити будь-яку можливість, діаграма варіанта використання здатна допомогти надати огляд системи більш високому рівні. Вже було сказано, що «корисні моделі це принципи вашої системи».

Завдяки спрощеному характеру, схеми використання є гарним засобом комунікації для зацікавлених сторін.

Мета діаграми використання – показати динамічний аспект системи. Додаткові схеми та документація можуть використовуватись для забезпечення повного функціонального та технічного розуміння системи. Вони забезпечують спрощене та графічне уявлення того, що насправді має робити система [5].

Елементи схеми є:

а) рамки системи – прямокутник з назвою на верху та еліпсами (прецедентами) усередині;

б) актор – стилізований людський персонаж, що позначає набір користувацьких ролей (людина, зовнішня сутність, клас, інша система), взаємодіє з будь-якої сутністю. Актори не можуть взаємодіяти між собою (через відключення обробки/дослідження звітів);

в) прецедент – позначений еліпс, що представляє систематичне виконувану дію (може включати можливі варіанти), що призводить до

результатів, що спостерігаються дійовими особами. Заголовок може бути або ім'ям, або описом (з точки зору дійової особи) того, що система робить (а не як). Під час сценарію актори систематично обмінюються повідомленнями. Декілька різних сценаріїв можуть бути зв’язані з одним прецендентом.

Рисунок 2.1 є діаграмою варіантів використання, що описує можливі дії користувача в системі.

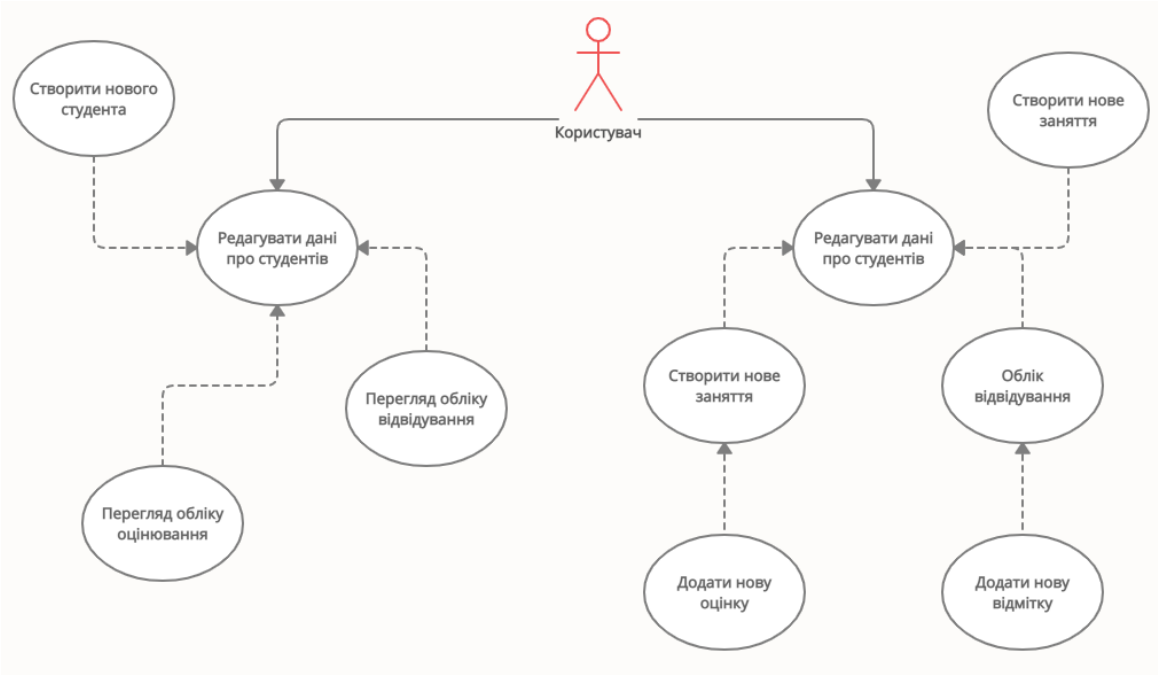

Рисунок 2.1 – Діаграма варіантів використання

Дана діаграма в повній мірі відображає можливості використання функцій органайзера.

2.2 Проектування пакетної будови системи

Уся система розділена на окремі шари, кожен з яких несе свою логічну мету. Діаграма пакетів зображена на рисунку 2.2.

Діаграма пакетів це фактично предок діаграми класів. В джава-розробці для спрощення читабельності архітектури всі файли розбиваються на окремі пакети, наприклад, пакет активностей, пакет контролерів, тощо.

Всередині пакетів знаходяться класи.

Діаграма пакетів спроектована в додатку Code Iris.

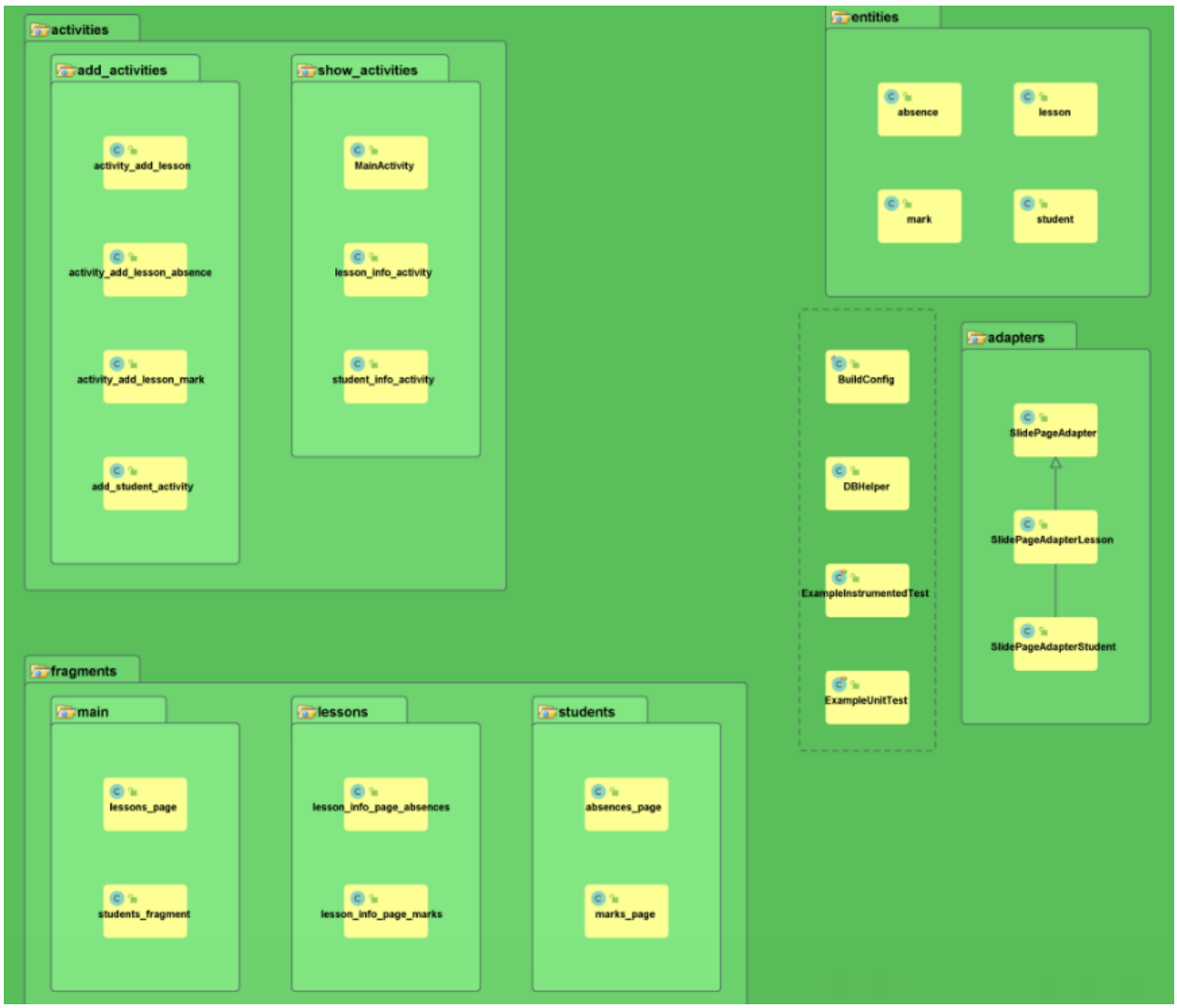

Рисунок 2.2 – Діаграма пакетів

З діаграми пакетів видно, що додаток містить в собі 4 основні пакети, які вміщають в собі активності, сутності, адаптери і фрагменти.

2.3 Проектування внутрішньої будови системи

У розробці програмного забезпечення діаграма класів уніфікованої мови моделювання (UML) є тип діаграми статичної структури, яка описує структуру системи, зображаючи класи системи, їх атрибути, операції та відносини між об'єктами [6].

Основним будівельним елементом об’єктно-орієнтованого моделювання є діаграма класів. Використовується вона для загального

концептуального моделювання структури програми та детального моделювання для перетворення моделей у програмний код. Діаграми класів також можна використовувати для моделювання даних. Класи на діаграмі класів представляють як базові елементи, взаємодії у програмі, і програмовані класи.

На схемі класи подаються вікнами, що містять три відсіки:

– Верхній відсік містить назву класу. Надруковано жирним шрифтом і по центру, перша заголовна літера.

– Середній відсік містить атрибути класу. Вирівнюються вони по лівому краю, перша буква була.

– У нижньому відсіку містяться операції, які клас може виконувати. Вони також вирівняні по лівому краю, а початкова літера мала.

При проектуванні системи виділяють певні класи, групуючи їх у діаграмі класів, що допомагає знайти з-поміж них статичні зв'язку. Класи концептуального проектування при детальному моделюванні, як правило поділяються на кілька підкласів.

Залежність – це семантичний зв'язок між вільними та підпорядкованими елементами моделі. Він існує між двома елементами, коли зміна визначення одного елемента (сервера або цільового елемента) може змінити інший елемент (клієнт або джерело). Ця асоціація є односторонньою. Залежність показана пунктирною лінією із відкритою стрілкою, яка вказує від клієнта до постачальника.

Для більш докладного опису поведінки систем ці діаграми класів можна розширити діаграмою станів або кінцевим автоматом UML.

Асоціація є сімейством посилань. Бінарна асоціація (з двома кінцями) зазвичай представлена рядком. Асоціація може зв'язати набір. Асоціація, яка має три ланки називається потрійною асоціацією.

Асоціації поділяються на чотири типи: односпрямована, двонаправлена, агрегаційна та рефлексивна. Найбільш поширеними є двонаправлені та односпрямовані асоціації.

Агрегація може відбуватися в тому випадку, коли клас є колекцією або контейнером інших класів, але класи, що містяться в ньому, не мають сильної залежності життєвого циклу від контейнера. В тому випадку коли контейнер буде знищено, його вміст буде існувати.

У UML він графічно представлений порожнистим ромбом на вміщуючому класі, що пов’язує з вміщеним класом. Сукупність – це семантично розширений об'єкт, який сприймається як єдине ціле у багатьох операціях, хоча фізично складається з кількох менших об'єктів.

Графічне представлення UML узагальнення – це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації – це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації – це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з

користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація – це зв'язок між класами, інтерфейсами, компонентами та пакетами, що сполучає елемент клієнта з елементом постачальника. Відношення реалізації між класами/компонентами та інтерфейсами показує, що клас/компонент реалізує операції, пропоновані інтерфейсом.

Залежність – це слабкіша форма асоціації, що вказує на те, що один клас залежить від іншого, тому що він використовує його в конкретний момент часу. Клас залежить від іншого, якщо незалежний клас є змінною параметра або локальної змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді зв'язок між двома класами дуже слабкий. Вони взагалі не реалізовані зі змінними членами. Зазвичай вони реалізовуються, як аргументи функцій-членів.

UML-представлення асоціації – це лінія, що поєднує два зв'язані класи. На кожному кінці рядка є додаткові символи.

Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутностей моделюють інформацію, якою обробляє система. Їх не можна ідентифікувати як таблиці у базах даних чи інших сховищах даних.

Вони намальовані у вигляді кіл із короткою лінією, прикріпленої до нижньої частини кола. Як альтернатива їх можна намалювати як звичайні класи зі стереотипним позначенням «сутність» над ім'ям класу.

На рисунку 2.3 зображено діаграму класів системи, спроектована в додатку Code Irirs.

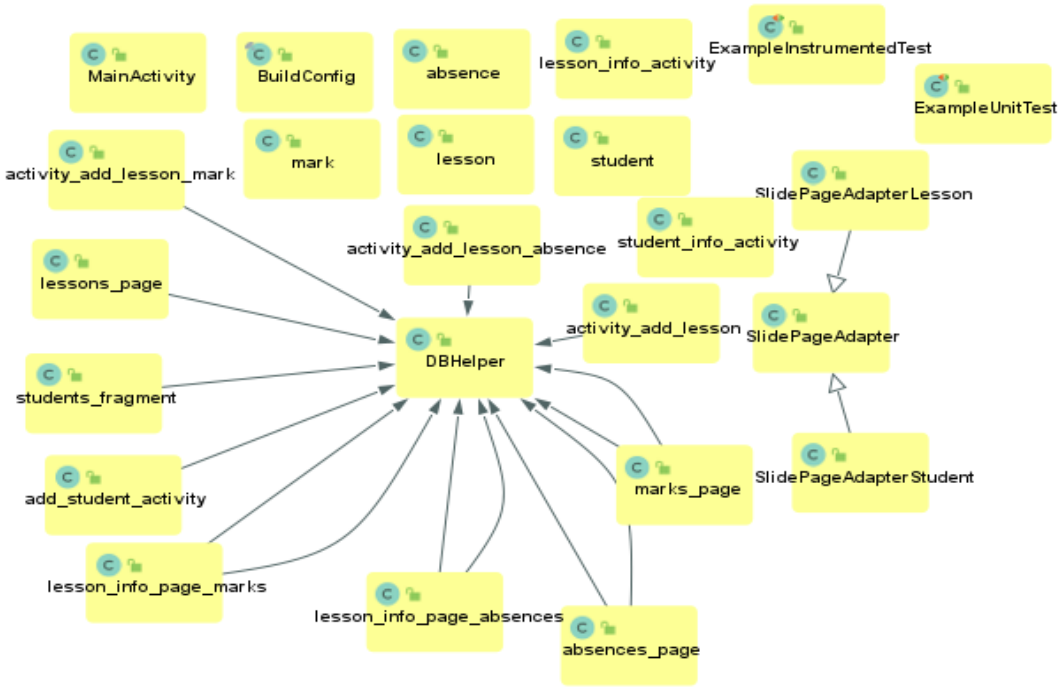

Рисунок 2.3 – Діаграма класів

З діаграми видно, що система складається з 25-ти класів, 4 з яких відповідають за сутності, 1 за роботу за базами даних, а інші є класами активностей, фрагментів і реалізаціями SlideAdapter, які допомагають створити екрани, які змінюються рухом в сторону.

2.4 Проектування інтерфейсу

Кожна програма створюється з розрахунком на велику кількість користувачів, яким необхідно, щоб у додатку був зручний та зрозумілий спосіб комунікації з ними. Такими інструментами є інтерфейс користувача. За допомогою інтерфейсу користувач керує додатком, отримує від неї повідомлення та відповідає на її запити тощо.

При проектуванні інтерфейсу початковим рішенням є вибір базових стандартів типів інтерфейсних елементів управління, які мають враховувати специфіку відповідної предметної області. При розробці інтерфейсу користувача необхідно враховувати характеристики планованих кінцевих користувачів програмного забезпечення, що розробляється. Специфікація типу інтерфейсу користувача визначає його синтаксис.

Зрозуміло, що одним з факторів, що впливає на ефективність використання програмного забезпечення, є простота використання інтерфейсу користувача.

Графічний користувацький інтерфейс – це тип інтерфейсу користувача, в якому елементи представлені користувачеві на екрані у вигляді зображень.

Дизайн графічного інтерфейсу є дуже важливою частиною розробки програмного забезпечення. На цьому етапі проектується простота та зручність у використанні програми, від цього також залежить майбутня взаємодія користувача та програми. Якщо цьому етапу не надати належного значення, репутація програми може почати знижуватися відразу після першого знайомства з нею користувача, навіть при відмінному функціоналі.

Android Studio надає програмісту набір візуальних компонентів для створення графічного інтерфейсу користувача. Вони практично не відрізняються від використовуваних в інших програмах, що

використовують графічний інтерфейс. Тому розробнику, навіть новачкові, нескладно приступити до створення інтерфейсу користувача.

Лише один робочий екран, як правило, використовується лише в простих програмах, в яких не потрібно виконувати безліч функцій. Однак у нашому випадку, неможливо обійтись лише одною формою, залишається створення кількох екранних форм. Це дасть право згрупувати та структурувати функції програми, що дозволить користувачеві без проблем орієнтуватися у програмі та виконувати необхідні функції.

Вибір кольорів – один із найважливіших критеріїв при розробці графічного інтерфейсу. Кольори мають дивовижну властивість: вони можуть зробити просте неповторним, а прекрасне перетворити на потворне.

Вибір шрифту також відіграє важливу роль у проектуванні та розробці інтерфейсу користувача. Підібравши гарний шрифт, можна досягти високої читабельності тексту та легкості сприйняття.

Після детального вибору шрифтів та кольорової гамми залишається ще один важливий крок – розміщення та угруповання елементів на екрані. Необхідно продумати, яка повинна бути структура кожної форми, що вона має виконувати та які елементи вона буде в себе вміщати.

Гарно структуровані екранні форми визначають простоту та зручність використання функціоналу.

Структуру додатку можна відобразити у вигляді моделей, максимально наближених до виду на мобільному пристрої. На рисунках 2.4 та 2.5 показані макети списку студентів та списку занять.

Рисунок 2.4 – Макет списку студентів

Рисунок 2.5 – Макет списку занять

Вікна із відображення списку всіх студентів та списку всіх занять, це одні з головних вікон додатку, інформація показана на них не є

надлишковою, цього легко досягти, відображаючи тільки коротку інформацію про те чи інше заняття, про того чи іншого студента, щоб дізнатись більш детальну інформацію потрібно натиснути на того учасника гуртка чи на ту лекція, що найбільше зацікавила. Перехід між цими вікнами здійснюється за допомогою свайпів, які вказані на макетах.

Далі рисунках 2.6, 2.7, 2.8, 2.9, 2.10, 2.11 показані підпорядковані макети, при взаємодії з якими інформація на макеті 2.4 та 2.5 буде ставати більш обширною.

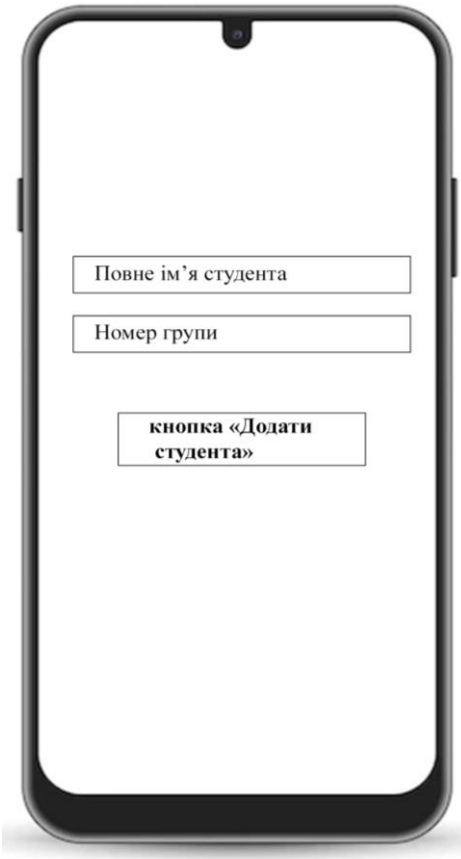

Рисунок 2.6 – Макет додавання студента

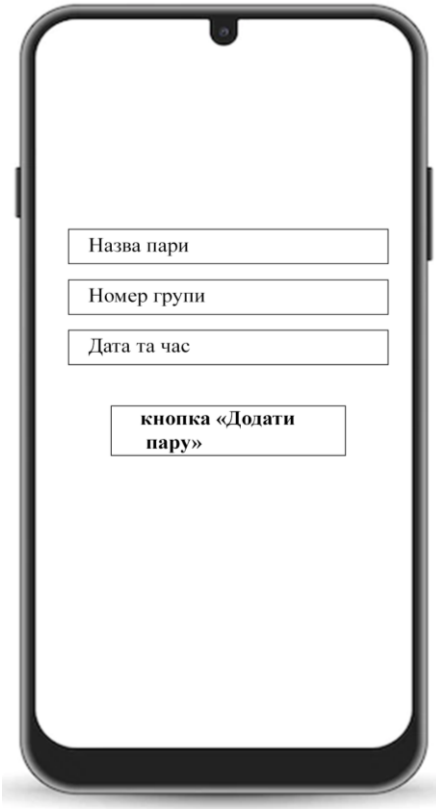

Рисунок 2.7 – Макет додавання заняття

Рисунок 2.8 – Макет перегляду оцінок на занятті

Рисунок 2.9 – Макет перегляду відвідування заняття

Рисунок 2.10 – Макет перегляду оцінок студента

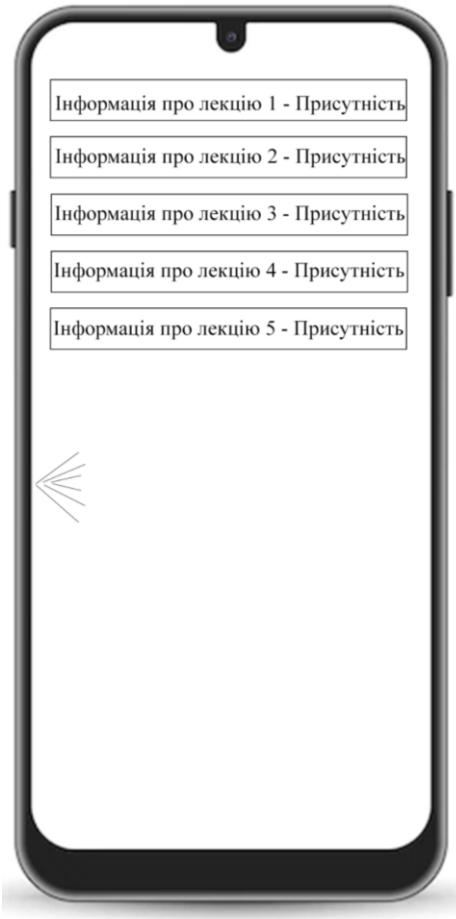

Рисунок 2.11 – Макет перегляду відвідуваності студента

Гарним варіантом було б зробити можливим користувачеві виконати потрібну дію не більше декількох дотиків, але великий обсяг інформації не дозволяє зменшити кількість кроків для цієї функції.

Через це, реалізовано дуже простий і швидкий інтерфейс, спрямований на виконання потрібної функції з найменшою кількістю дотиків.

Тому, користуючись блок-схемами та моделями віконних форм, можна досягти спрощення процесу реалізації.

Створювати форми з їхньою допомогою нескладно, чи конструктором, чи вручну.

Цей підхід завоював популярність серед розробників та дизайнерів GUI.

3 РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ

3.1 Огляд інструментальних засобів розробки

3.1.1 Мова програмування

Java – це об'єктно-орієнтована мова програмування спроектована так, щоб мати якнайменше залежностей від реалізації. Це мова програмування загального призначення, яка дозволяє програмістам написати один раз і запустити його будь-де (WORA). Це означає, що скомпільований код Java може працювати на всіх платформах, що підтримують Java без необхідності перекомпіляції [7].

Програми Java компілюються на байт-код, який можна запускати на будь-якій віртуальній машині Java (JVM) незалежно від базової комп'ютерної архітектури. Синтаксис Java подібний до C і C++, але має менше засобів низького рівня, ніж будь-який з них. Java Runtime Environment надає динамічні функції (такі як перегляд і редагування коду під час виконання), зазвичай недоступні у традиційних мовах, що компілюються.

В 2019 році Java була визнаною однією з найпопулярніших мов програмування, що використовуються відповідно до GitHub, особливо для веб-додатків клієнт-сервер, із 9 мільйонами розробників.

Однією з цілей розробки Java є переносність, а це означає, що програми, написані під платформу Java, повинні однаково працювати на будь-якому поєднанні апаратного забезпечення та операційної системи з адекватною підтримкою середовища виконання.

Можна скомпілювати код Java наполовину, званий байт-код Java, щоб замінити непроміжний машинний код, який повинен лежати в архітектурі. Інструкції байт-коду Java аналогічні машинному коду, але розпізнаються під час запуску віртуальної машини (VM), написаної спеціально апаратного забезпечення хоста.

Стандартні бібліотеки надають загальний спосіб доступу до функцій, невідомих хосту, таких як графіка, багатопоточність та робота в мережі.

Java сама по собі не залежить від платформи і не адаптована до конкретної платформи, для якої вона призначена, оскільки віртуальна машина Java (JVM) перекладає байт-код Java машинною мовою платформи.

OpenJDK – ще одна помітна реалізація Java SE, яка розповсюджується під ліцензією GNU GPL. Впровадження розпочалося, коли Sun почала випускати вихідний код Java з ліцензією GPL. Починаючи з Java SE 7, OpenJDK є офіційною еталонною реалізацією Java.

Java не залежить від платформи, вона є важливою для Java EE і для сертифікації реалізації потребує доопрацювання. Цей носій дозволяє надсилати додатки на сторону сервера.

Android

Мова Java є фундаментальною опорою Android, мобільної операційної системи з відкритим вихідним кодом. Хоча Android побудований на ядрі Linux і написаний в основному на C, Android SDK користується мовою Java як основою для програм Android, але не використовує будь-які стандартні графічні інтерфейси користувача, SE, ME або інші стандарти Java. Мова байт-коду, яку підтримує Android SDK, несумісна з байт-кодом Java і використовує свою віртуальну машині, оптимізовану для пристроїв із низьким рівнем пам’яті, таких як смартфони та планшетні комп’ютери. Залежно від версії Android, байт-код або інтерпретується віртуальною машиною Dalvik, або компілюється в рідний код за допомогою середовища виконання Android.

Android не надає повну стандартну бібліотеку Java SE, але Android SDK включає незалежну реалізацію великої її підмножини. Він підтримує Java 6 та деякі функції Java 7 та забезпечує реалізацію, сумісну зі стандартною бібліотекою (Apache Harmony).

3.1.2 Середовище розробки

В процесі вибору середовища розробки, необхідно проаналізувати його щодо підтримки обраного мови програмування, надання засобів розробки та наявності графічного віконного редактора. Все це підтримується платформою Android Studio.

Android Studio – це офіційне інтегроване середовище розробки (IDE) для операційної системи Google Android, створене на основі програмного забезпечення JetBrains IntelliJ IDEA та розроблене спеціально для розробки під Android [8].

Android Studio пропонує безліч функцій, що підвищують продуктивність під час створення програм для Android, наприклад:

– підтримка збірки на основі Gradle;

– зручний та багатофункціональний емулятор;

– інтеграція ProGuard та можливість підпису додатків;

– єдине середовище, в якому можна розробляти для всіх Android-пристроїв;

– миттєвий запуск для застосування змін до запущених програм без створення нового APK;

– вбудована підтримка Google Cloud Platform для інтеграції з Firebase Cloud Messaging (раніше Google Cloud Messaging) та Google App Engine.

За допомогою Android Studio скомпільовану програму можна опублікувати в Google Play Store. Програма повинна відповідати політиці змісту Google Play Store.

3.2 Реалізація функціоналу мобільному застосунку

Для реалізації програмного продукту було здійснено такі основні кроки:

а) завантажено інструменти для розробки;

б) встановлено завантажені пакети;

в) підключено бібліотеки;

г) налаштовано середовище розробки;

д) безпосередня розробка програми.

3.2.1 Підготовка до розробки програмного продукту

Для виконання завдання необхідно встановити середовище розробки Android-програм «Android Studio». Як було зазначено вище, ця програма є безкоштовною, тому її можна легко завантажити з офіційного сайту (рисунок 3.1).

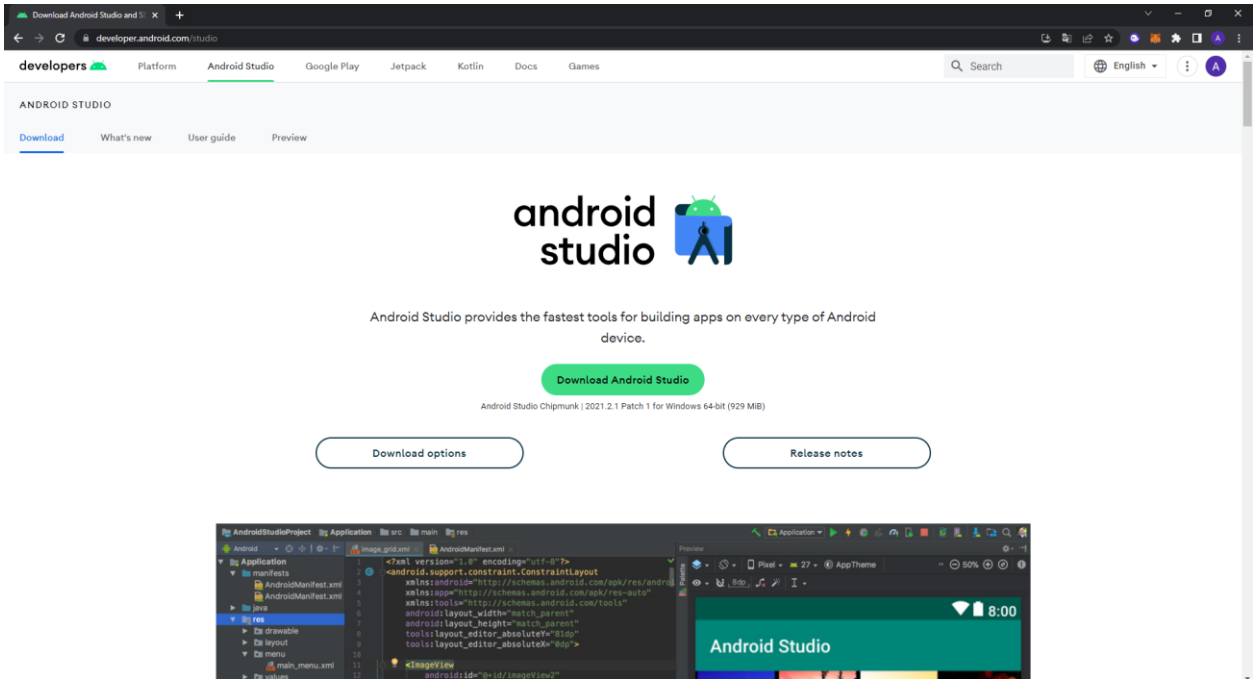

Рисунок 3.1 – Сторінка з завантаженням додатку “Android Studio”

Після початку установки інсталятор самостійно завантажує потрібні SDK та інші матеріали, які підходять для версії вашої системи на ПК.

При розробці мобільного додатку необхідно врахувати версію операційної системи, під яку розробляється продукт, та мінімальну версію операційної системи, яка підтримує запуск цієї програми.

Щоб спростити вибір потрібної мінімальної ОС, Android Studio надає миттєву статистику за кількістю пристроїв із різними версіями ОС. Вам також необхідно вибрати, для якого саме пристрою ви хочете створити

програму: SmartWatch, Смартфон, ТБ, Android Auto або Окуляри (рисунок 3.2).

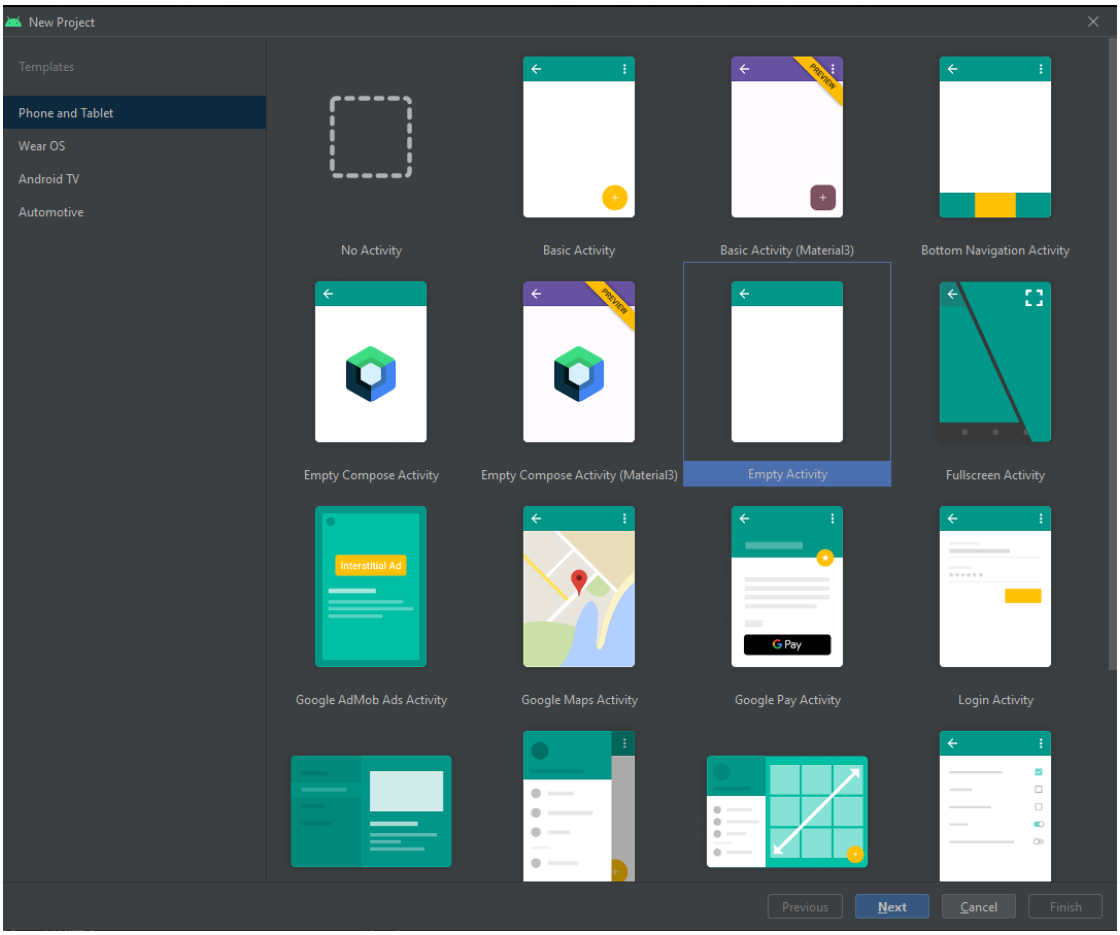

Рисунок 3.2 – Вибір пристрою для розробки додатку

Після того як було обрано та для якого пристрою буде розроблятись програма, та з яким шаблоном, пересуваємось далі (рисунок 3.3).

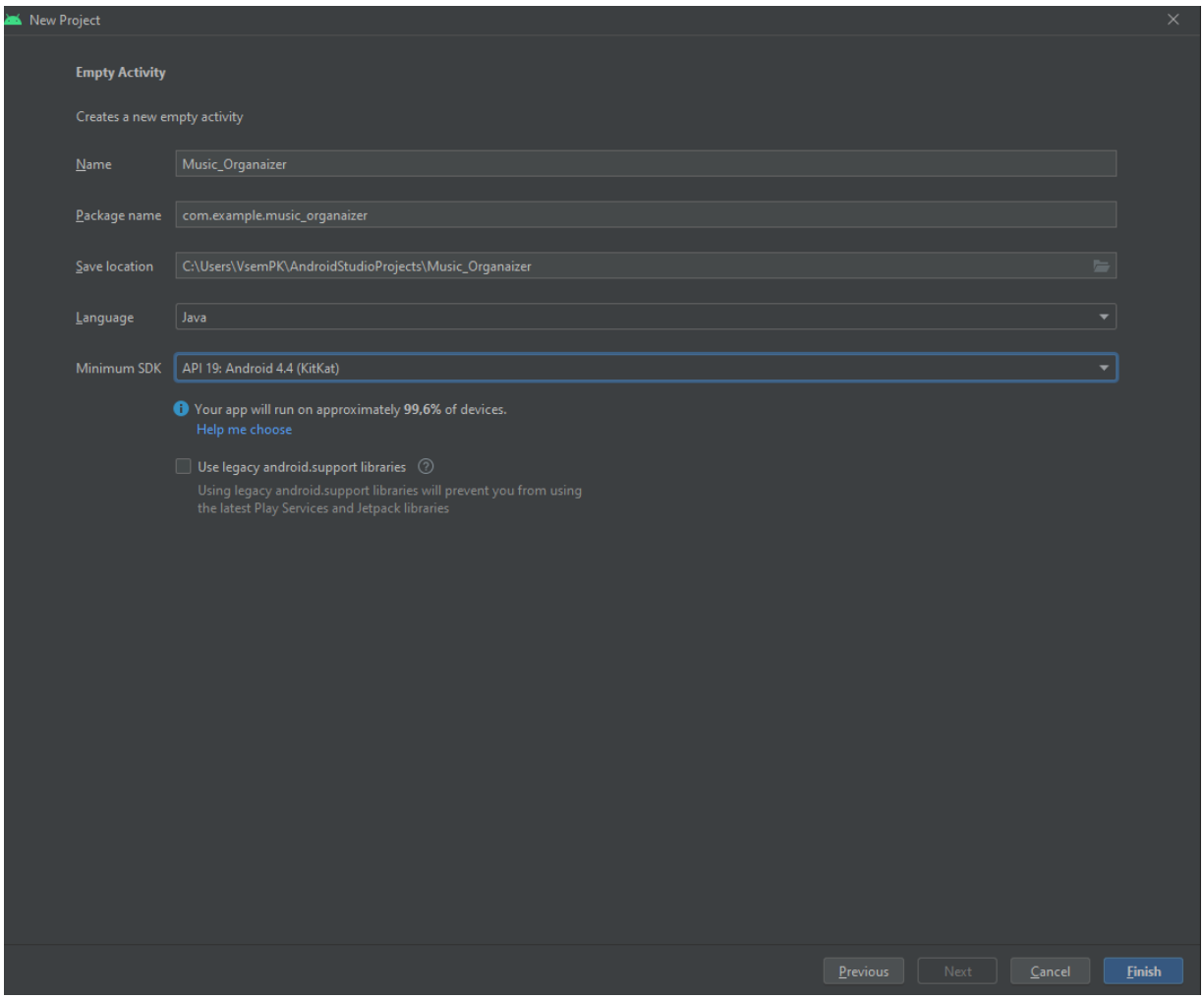

Рисунок 3.3 – Створення проекту

В цьому вікні потрібно вказати ім'я проекту, місце де він буде зберігатись, вибрати мову на якій буде написаний код та мінімальну операційну систему на якій працюватиме додаток. Додаток відмовиться працювати, якщо версія операційної системи буде нижче за мінімальну вказану. На версіях вище мінімальної додаток без проблем буде працювати.

3.2.2 Реалізація інтерфейсу користувача

При написанні програмного коду завжди велика увага приділяється його стандартизації – написання коду відповідно до певних правил та інструкцій з оформлення тих чи інших блоків, класів, змінних тощо.

Таким чином, створення програми – трудомісткий процес, який

можна стандартизувати, щоб спростити роботу з ним у майбутньому.

Початком роботи є розміщення функціональних елементів на формах відповідно до заздалегідь розроблених макетів. Існує багато об'єктів, які можна розмістити на формі. Розглянемо лише ті, які потрібні для реалізації програми:

а) TextView – це найчастіше використовуваний елемент у будь-якій програмі. Він може відображати будь-яку інформацію.

б) Button – елемент кнопка, містить в собі великий набір функцій. Кнопки дозволяють реалізовувати складні користувацькі інтерфейси. Кнопки дають змогу виконувати певні функції залежно від потреб користувача.

в) ImageView – вся інформація, представлена у вигляді графіки, реалізована за допомогою цього компонента.

г) ListViev – це елемент, який вражає своїми можливостями – зміна властивості призводить до суттєвих змін можливостей елемента. З його допомогою можна не лише відображати задані елементи, а й створювати комбо-списки зі своїми властивостями.

Після того, як екранні форми було сформовано, подальшою роботу буде програмування кнопок та основного функціоналу.

Основними функціями в цьому додатку є обробка виконання певних дій. Розглянемо функції додавання дій в нашому додатку (лістинг 3.1, 3.2, 3.3 та 3.4).

Лістинг 3.1 – Функція додавання інформації про нове заняття

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_lesson);
    dbHelper = new DBHelper(this);
    name = findViewById(R.id.lesson_name_edit);
    group = findViewById(R.id.lesson_group_edit);
    datetime = findViewById(R.id.lesson_dattime_edit);
    new_lesson = findViewById(R.id.new_lesson_btn);


    new_lesson.setOnClickListener(new
```

```
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                dbHelper.addLesson(name.getText().toString(),
group.getText().toString(), datetime.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
             }
        });
```

В цьому мобільному додатку була розроблена досить цікава та не звична функція “статус учасника музичного гуртка”.

Для того, щоб розуміти добросовісний учень чи ні, потрібно тільки подивитись на його статус. Статус – певна кількість очок, яка присутня в кожного, хто відвідує гурток. Працює це таким чином, керівник гуртка під час заняття перевіряє присутніх та заносить ці дані в додаток. Програма аналізує дані та тим хто присутній на занятті додає бали, в того студента, який відсутній бали знімаються. Після декількох занять, глянувши на список студентів можна зрозуміти, хто сумлінно відвідував кожне заняття, а хто лінувався.

Лістинг 3.2 – Функція статусу учасника музичного гуртка

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_lesson_absence);
        dbHelper = new DBHelper(this);
        student = findViewById(R.id.student_id_edit);
        status = findViewById(R.id.absence_status_edit);
        new_absence = findViewById(R.id.new_absence_btn);
        Intent intent = getIntent();
        Bundle bndl = intent.getBundleExtra("id");
        lesson_id = bndl.getString("id");

        new_absence.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                List<student> stdns = dbHelper.getStudents();
                int rate = 0;
                for(int i = 0; i < stdns.size();i++)
                {
                    if(stdns.get(i).student_id              ==
Integer.parseInt(student.getText().toString()))
```

```
                    {
                        rate                                    =
Integer.parseInt(stdns.get(i).student_status);
                    }
                }
                dbHelper.addAbsence(lesson_id,
student.getText().toString(), status.getText().toString());
                if(status.getText().toString().equals("+"))
                {
                    rate+=10;
                }
                else  if(status.getText().toString().equals("-
"))
                {
                    rate-=10;
                }
                dbHelper.updateStudent(String.valueOf(rate),
student.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
```

Лістинг 3.3 – Функція додавання оцінки за конкретне заняття

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_lessons_mark);
        dbHelper = new DBHelper(this);
        student = findViewById(R.id.student_id_mark_edit);
        status = findViewById(R.id.mark_value_edit);
        new_mark = findViewById(R.id.new_mark_btn);
        Intent intent = getIntent();
        Bundle bndl = intent.getBundleExtra("id");
        lesson_id = bndl.getString("id");

        new_mark.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                dbHelper.addMark(lesson_id,
student.getText().toString(), status.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }
```

Лістинг 3.4 – Функція додавання нового студента

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_add_student);
   dbHelper = new DBHelper(this);
   name = findViewById(R.id.studnt_name_edit);
   group = findViewById(R.id.studnt_group_edit);
   new_student = findViewById(R.id.new_student_btn);

   new_student.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            dbHelper.addStudent(name.getText().toString(),
group.getText().toString());
            dbHelper.close();
            Intent intent = new Intent();
            setResult(RESULT_OK, intent);
            finish();
        }
   });
```

Між вікнами, які знаходяться на одному рівні здійснена навігація свайпами (лістинг 3.5).

Лістинг 3.5 – Навігації між однорівневими вікнами

```
public class SlidePageAdapterLesson extends
SlidePageAdapter {
   String curr_lesson;
   public SlidePageAdapterLesson(@NotNull FragmentManager
fm, int lsn_id) {
        super(fm);
        curr_lesson = String.valueOf(lsn_id);
    }
    @NonNull
    @NotNull
    @Override
    public Fragment getItem(int position) {
        Bundle bundle = new Bundle();
        bundle.putString("id", curr_lesson);

        lesson_info_page_marks marks = new
lesson_info_page_marks();
        marks.setArguments(bundle);

        lesson_info_page_absences absences = new
lesson_info_page_absences();
        absences.setArguments(bundle);

        switch (position) {
```

```
            case 0:
                return marks;
            case 1:
                return absences;
            default:
                return marks;
        }
    }

    @Override
    public int getCount() {
        return 2;
    }
```

Програмний код усіх класів та функцій мобільного органайзера подано в додатку А.

3.3 Розробка графічного інтерфейсу користувача

Розробка мобільних додатків включає в себе не лише написання коду, але й розробку інтерфейсу користувача за схемами і шаблонами екранних форм. Графічний інтерфейс користувача органайзера керівника музичного гуртка складається з кількох основних вікон, а саме:

- вікно списку студентів (рисунок 3.4);
- вікно створення студента (рисунок 3.5);
- вікно списку занять (рисунок 3.6);
- вікно створення заняття (рисунок 3.7);
- вікно перегляду оцінок за заняття (рисунок 3.8);
- вікно додавання оцінки (рисунок 3.9);
- вікно перегляду відвідування заняття (рисунок 3.10);
- вікно додавання нової відмітки відвідуваності (рисунок 3.11);
- вікно перегляду оцінок студента (рисунок 3.12);
- вікно перегляду відвідуваності студента (рисунок 3.13).

Для початку розглянемо вікно списку студентів (рисунок 3.4), на ньому присутні: ID, ПІП, група та статус студента.

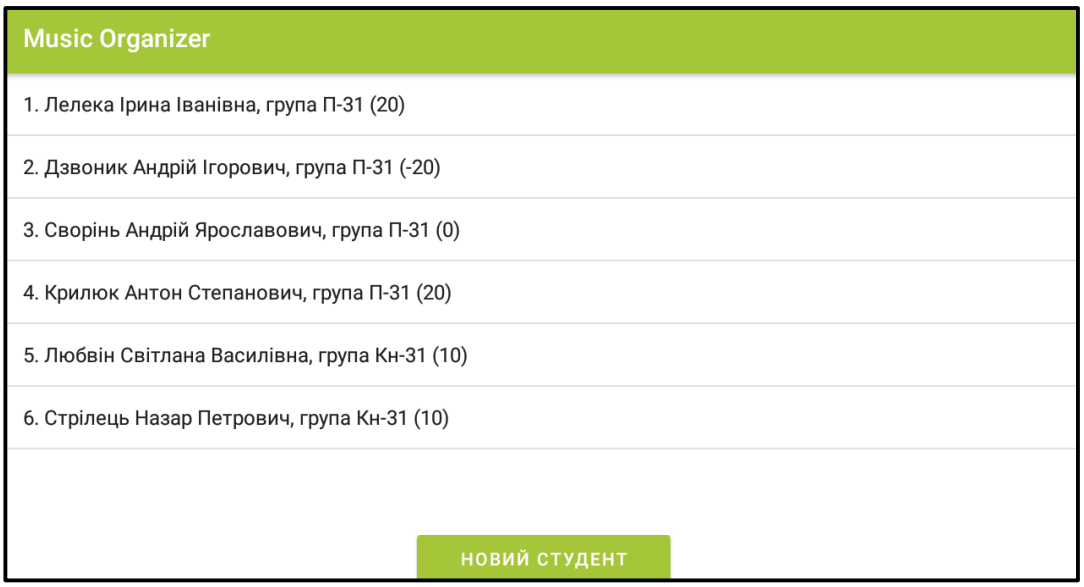

Рисунок 3.4 – Вікно списку студентів

В цьому вікні можна додати нового або ж переглянути інформацію про певного студента. Після взаємодії з кнопкою “Новий студент” відкриється нове вікно (рисунок 3.5) з допомогою якого можна поповнити список студентів.

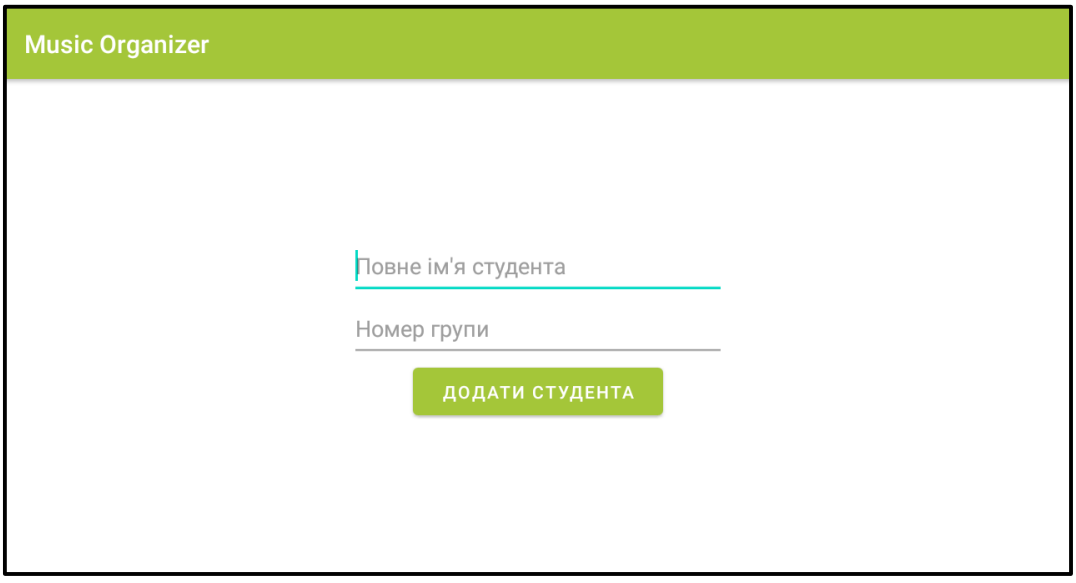

Рисунок 3.5 – Вікно створення студента

В цьому вікні потрібно заповнити такі поля як “Повне ім’я студента” та “Номер групи”, далі потрібно натиснути на кнопку “Додати студента”. Після виконання цих дій нас поверне на сторінку з оновленим списком студентів.

За допомогою свайпа вправо ми можемо перейти до вікна списку занять (рисунок 3.6). На цій сторінці подано детальну інформацію про заплановані лекції.

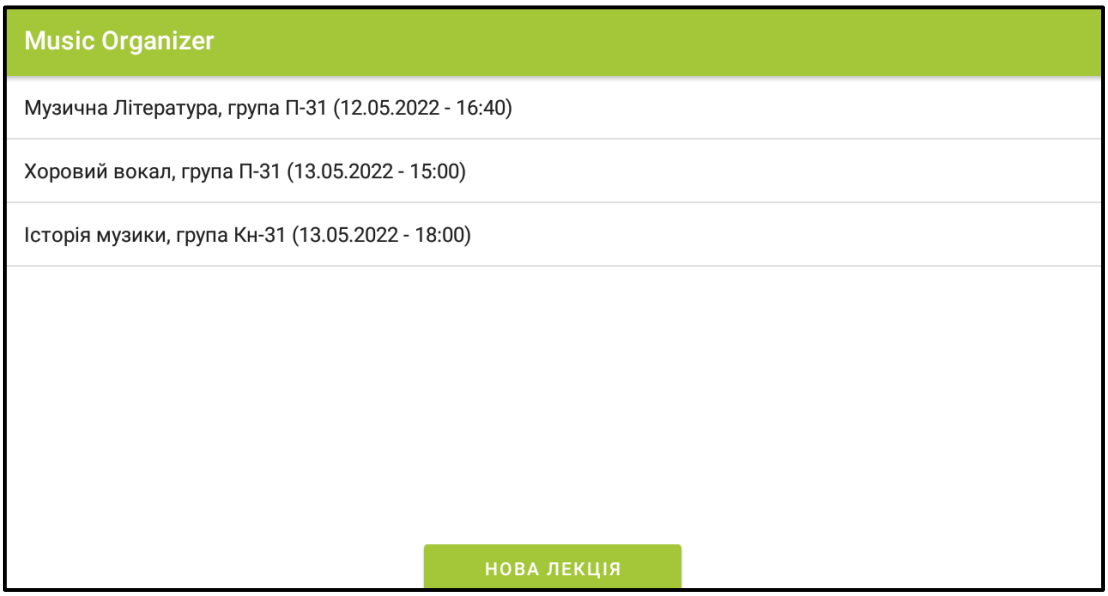

Рисунок 3.6 – Вікно списку занять

Тут можна додати нову інформацію про майбутні лекції, або ж виставити чи переглянути оцінки, натиснувши на конкретну лекцію.

Після взаємодії з кнопкою “Нова лекція”, відкриється нове вікно (рисунок 3.7), яке призначене для додавання нових лекцій.

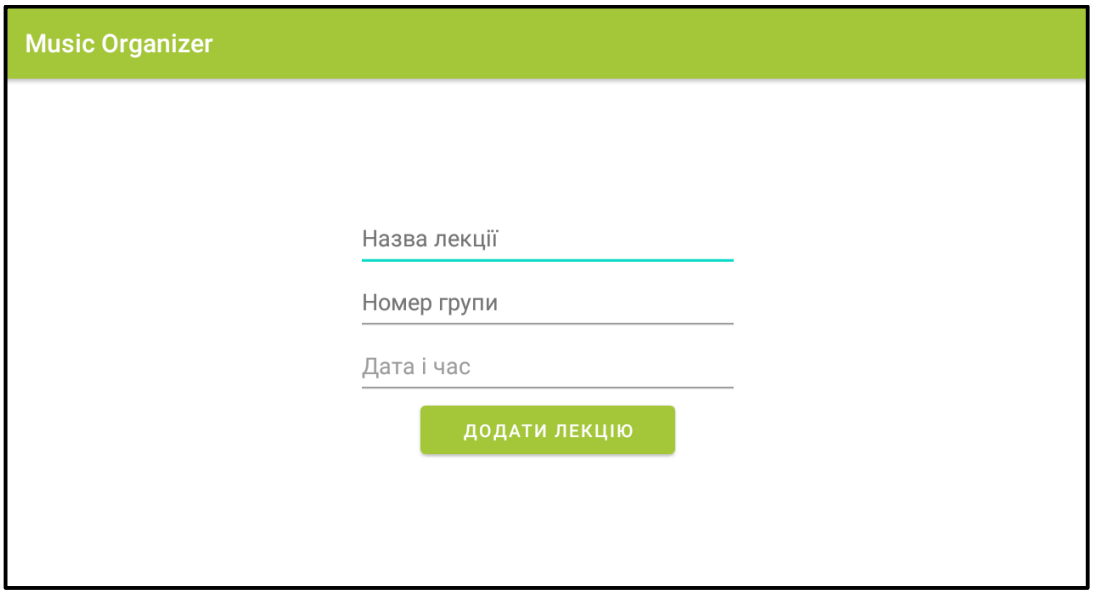

Рисунок 3.7 – Вікно створення заняття

Заповнивши інформації в поданих полях, потрібно натиснути на кнопку “Додати пару”, після цього додаток відкриє вікно з усіма заняттями.

Після натиску на конкретну пару, відкриється вікно з оцінками (рисунок 3.8), де зліва ID студента, а справа оцінка за пару.

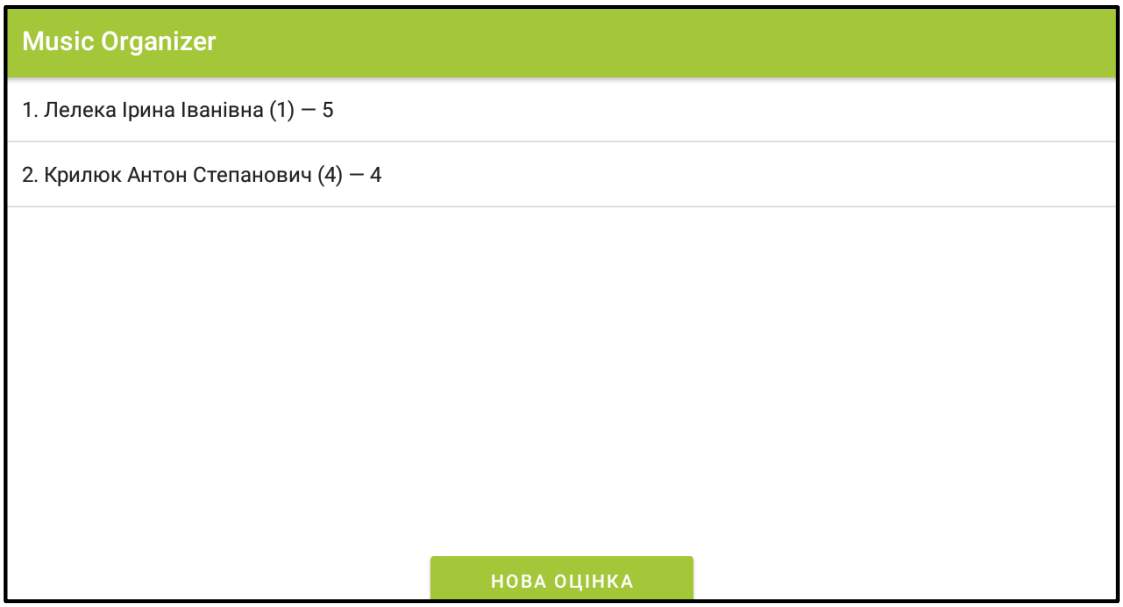

Рисунок 3.8 – Вікно перегляду оцінок за заняття

Після взаємодії з кнопкою “Нова оцінка” відкриється сторінка “додавання оцінок” (рисунок 3.9).

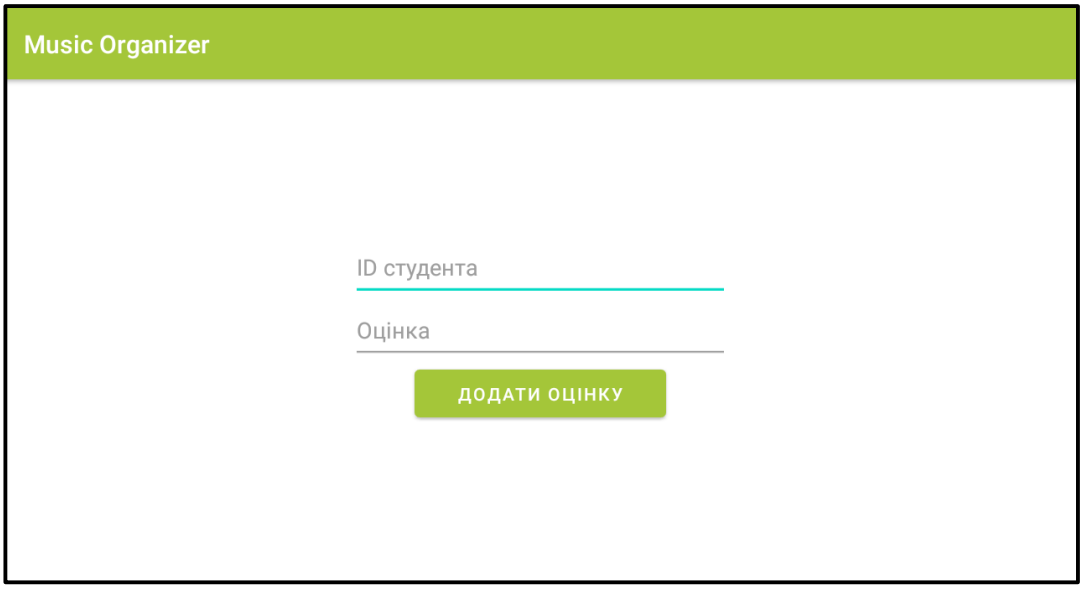

Рисунок 3.9 – Вікно додавання оцінки

Для того щоб додати оцінку потрібно заповнити поле “ID студента” та “Оцінка”, далі натиснути на кнопку “Додати оцінку”. Після цих дій додаток повернеться на сторінку з оцінками, при свайпі вправо для перегляду стане доступне вікно відвідувань занять (рисунок 3.10).

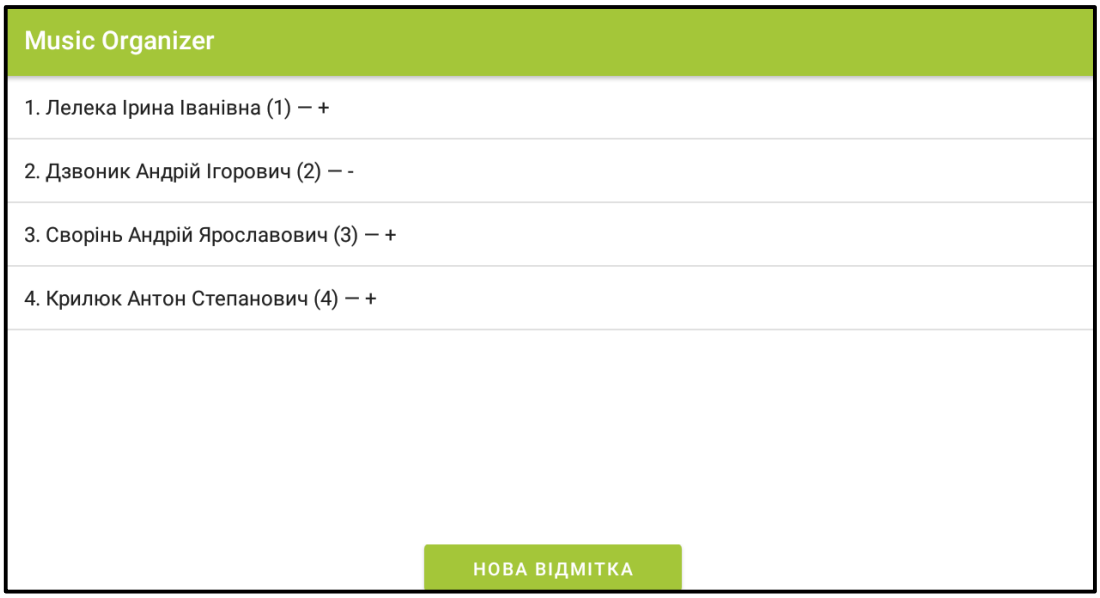

Рисунок 3.10 – Вікно перегляду відвідування заняття

В цьому вікні подана інформація про відвідування занять студентами, зліва вказане ID студента, справа символ “+” (присутній на занятті) чи “-” (відсутній на занятті). При взаємодії з кнопкою “Нова відмітка” відкриється вікно додавання нової відмітки відвідуваності (рисунок 3.11).

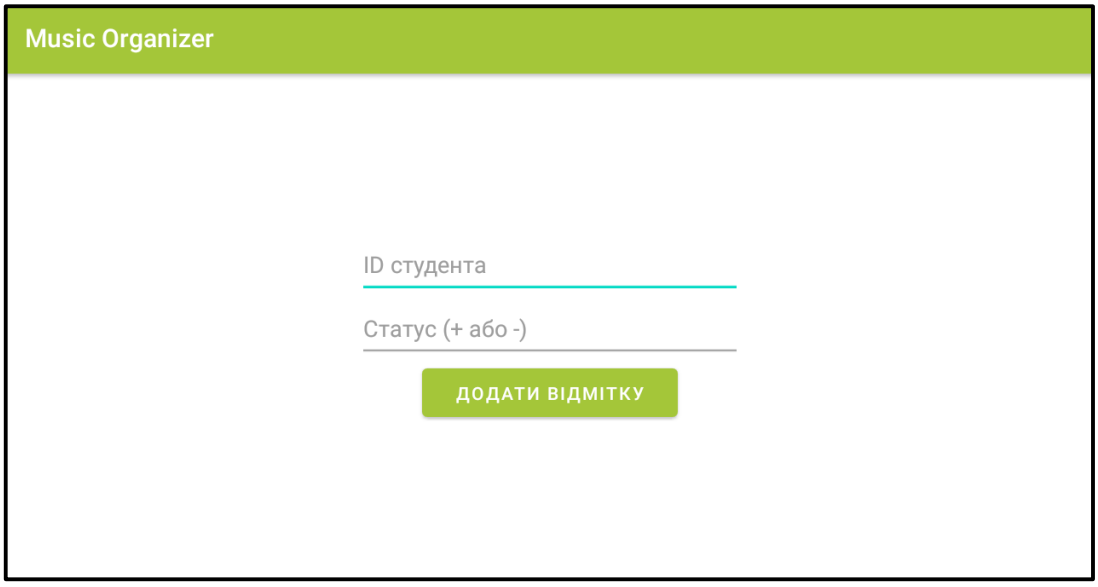

Рисунок 3.11 – Вікно додавання нової відмітки відвідуваності

На цій сторінці можна побачити поля “ID студента”, “Статус (+ або -)”, які потрібно заповнити. Коли все готово можна натиснути на кнопку “Додати відмітку”, тоді програма повернеться до оновленого списку з відмітками.

Повернемось до вікна списку студентів (рисунок 3.4) та натиснемо на конкретного учасника гуртка. Після взаємодії з певним студентом відкриється вікно перегляду оцінок цього студента (рисунок 3.12).

Рисунок 3.12 – Вікно перегляду оцінок студента

На цій сторінці розписано: назва предмету, дата оцінення та саму оцінку. Свайнувши на цій сторінці вправо, відкриється вікно перегляду відвідуваності цього студента (рисунок 3.13).

Рисунок 3.13 – Вікно перегляду відвідуваності студента

На сторінці вказано назву лекція та знак “+” чи “-”, що вказує на те, чи присутній студент або ж його немає.

3.4 Тестування

Тестування ПЗ – процес перевірки відповідності встановленим вимогам до продукту та реалізованої функціональності, що здійснюється методом спостереження за роботою додатка у штучно створених ситуаціях та в обмеженому наборі тестів, вибраних певним способом.

Тестування програмного забезпечення може надати об’єктивне, незалежне уявлення про програмне забезпечення, що дає змогу організаціям оцінити та зрозуміти ризики, пов’язані з впровадженням програмного забезпечення. Методи тестувань включають, але не обмежуються:

– аналіз вимог до продукту на предмет повноти та правильності в будь яких контекстах, таких як перспективи галузі та бізнесу, здійсненність та прибутковість впровадження, простота в використанні, продуктивність, безпека, обґрунтування інфраструктури тощо;

– аналіз архітектури та загального дизайну продукту;

– робота з розробниками продуктів для покращення методів кодування, шаблонів проектування, тестів, які можуть бути написані частиною коду на базі різних методів, таких як граничні умови тощо;

– запуск програми для перевірки поведінки є;

– перевірка інфраструктури розгортання та зв’язаних з нею сценаріїв та засобів автоматизації;

– приймати участь у виробничій діяльності з використанням методів моніторингу та спостереження.

Тестування ПЗ може надати користувачам об'єктивну, незалежну інформацію про якість та ризик відмови програмного забезпечення.

Помилки програмного забезпечення виникають у результаті наступного процесу: програміст допускає помилку, що призводить до помилки (дефекту,

збою) у вихідному коді програмного забезпечення. Якщо припуститися цієї помилки, в деяких ситуаціях система даватиме невірні результати, що призведе до збою.

Не всі програмні помилки викликані не правильним кодуванням. Одним з найпоширеніших джерел дорогих дефектів є упущення в вимогах, тобто невизнані вимоги, що призводять до помилок, які пропустив розробник. Прогалини у вимогах часто бувають нефункціональними вимогами, такими як тестування, продуктивність, масштабованість, ремонтопридатність, та інші вимоги безпеки.

Існує багато підходів до тестування програмного забезпечення. Статичними тестами називаються огляди, покрокові інструкції та перевірки, а виконання запрограмованого коду на певному наборі тестових випадків називається динамічним тестуванням.

Статичне тестування розпочинається на початковій стадії життєвого циклу програмного забезпечення і тому є частиною процесу перевірки. Для цього типу тесту іноді навіть не потрібен комп'ютер, наприклад, для перевірки вимог.

Динамічне тестування виконується при запуску програми. Динамічне тестування може розпочатися до 100% завершення програми для перевірки окремих ділянок коду та застосування до окремих функцій чи модулів. Поширеними методами є використання заглушок/драйверів або виконання серед налагодження.

Статичне тестування включає перевірку, тоді коли динамічне тестування також передбачає перевірку.

Пасивне тестування допомагає змінити поведінку системи без взаємодії з програмним продуктом.

На відміну від активних тестів, тестувальники не надають жодних тестових даних, а вивчають системні журнали та трасування. Вони шукають

певні моделі та специфічну поведінку, щоб приймати рішення. Це пов'язано з автономною перевіркою під час виконання та аналізом журналу.

Дослідницьке тестування – це такий підхід тестування ПЗ, який коротко описується як паралельне навчання, розробка та виконання тестів.

«Коробковий» підхід

Методи тестування ПЗ звично поділяються на тестування «білої скриньки» та «чорної скриньки». Ці підходи використовують для опису погляду тестувальника розробки тестових випадків. Гібридний підхід, званий тестуванням сірої скриньки, може використовуватись у методології тестування програмного забезпечення.

У міру того, як концепція тестування сірої скриньки, яка розробляє тести на основі певних елементів дизайну, стає все більш популярною, це «довільна відмінність» між тестуванням чорної скриньки та тестуванням білої скриньки дещо зникло (таблиця 3.1).

Таблиця 3.1 – Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Невірні дані при заповненні оцінки	При введенні невірних даних система повідомляє користувача про це.	При введенні невірних даних система повідомляє користувача про це.
2	Пусті поля при вводі даних	При спробі створення запису з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення запису з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
4	Створення відмітки з пустими полями	При спробі створення відмітки з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення відмітки з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

Для повної перевірки потрібно протестувати застосунок на іншому пристрої.

Для цього буде пройдена перевірка на слабкому телефоні, в якому операційна система андроїд 6.0.

Додаток без проблем відкрився далі буде здійснена перевірка функціоналу (рисунок 3.15, 3.16, 3.17, 3.18).

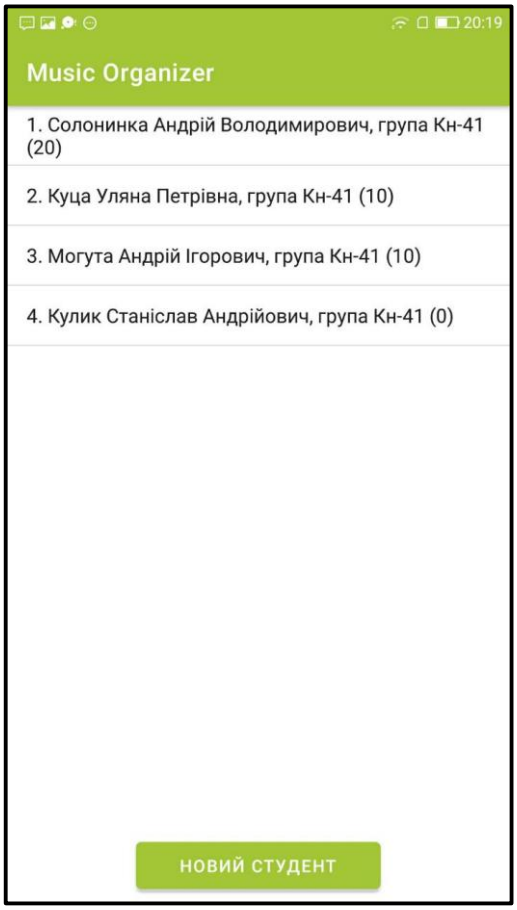

Рисунок 3.15 – Вікно списку учасників гуртка

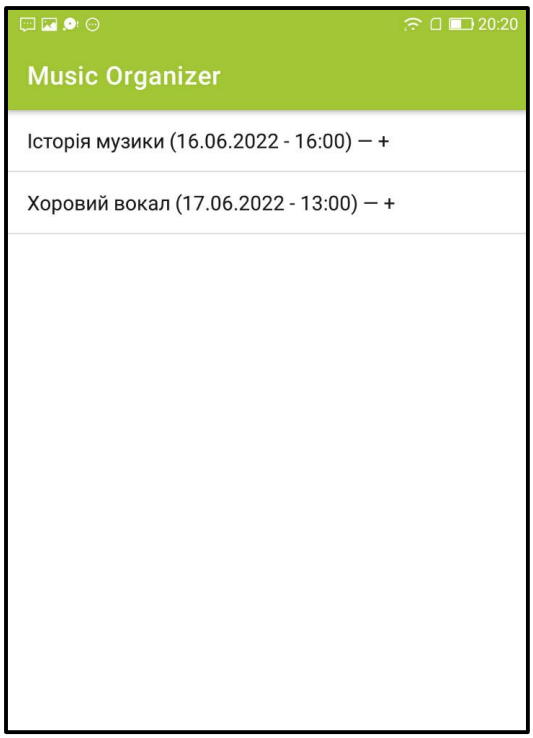

Рисунок 3.16 – Вікно перегляду відвідуваності студента

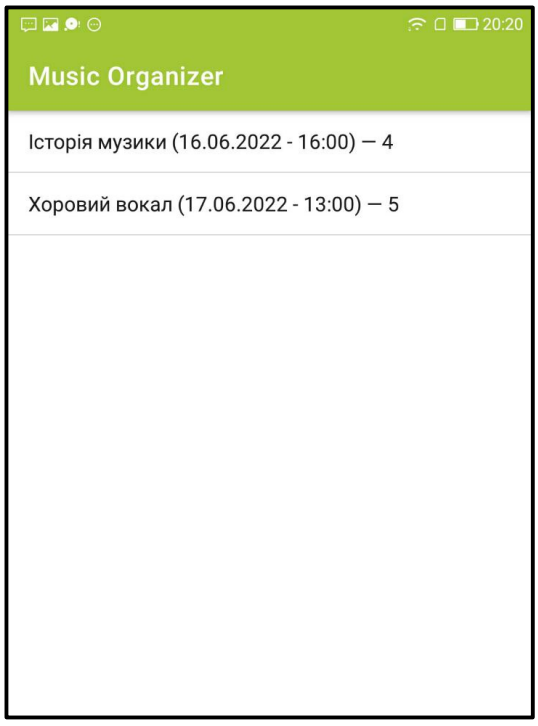

Рисунок 3.17 – Вікно перегляду оцінок певного студента

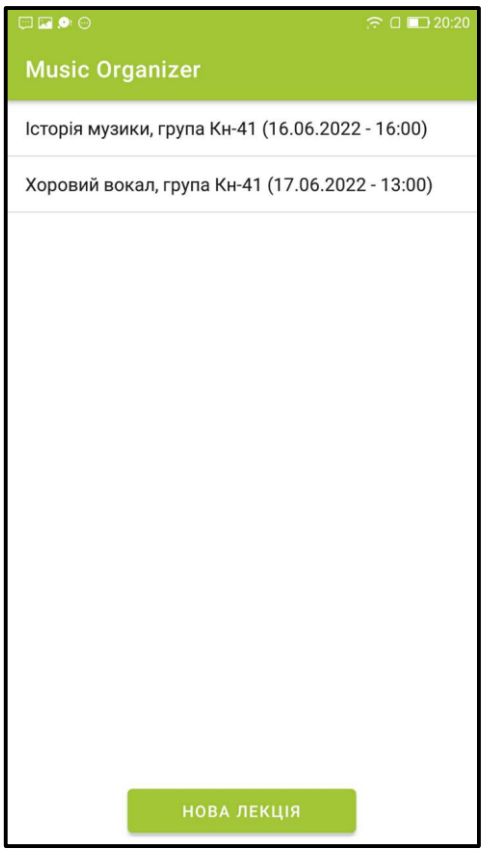

Рисунок 3.18 – Вікно перегляду лекцій

Додаток успішно завершив перевірку, вікна на різних пристроях виглядають однаково, це можна побачити на рисунку. 3.4 та 3.15, тому можна бути впевненим в справній роботі застосунку. Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку

Результатом дипломного проекту є програма обсягом 3.8 МБ. Сьогодні на ринку присутні такі продукти з програмною та технічною реалізацією. Ця програма є їхньою модифікацією, реалізованою лише програмно. Замість технічної частини продуктів на ринку ця програма використовує пристрої, які є у користувачів. Для роботи програми потрібен телефон із працюючою версією андроїд. Версія Android не має значення для використання мобільного додатка, але вона повинна бути не нижче за Android 5.0. При нижчій версії, програма скоріш за все не запуститься, оскільки вона не адаптована під старі версії.

Екран пристрою може мати будь-яку діагональ.

Найважливішими потребами покупців є оптимальна ціна та гарна якість продукту. Він також повинен бути простий у встановленні та використанні. Люди, які навчають інших, є потенційними клієнтами. Цей продукт може спростити вчителям роботу, та зробити її облік цікавим.

Для більшої доступності доцільно реалізувати програмне забезпечення в мережі Інтернет.

Для кращої доступності доцільно реалізувати програмне забезпечення в Інтернеті.

На українському ринку доволі мало конкурентів. Проте іноземні компанії випускають досить гарні застосунки, основна відмінність створеного мобільного додатка – українська мова в програмі, на даний час це дуже важливо, можна виділити програму todolist, яка зараз дуже популярна на ринку в Україні. Сама програма дуже адаптивна, але української мови в ній немає, а деякі функції платні.

4.2 Розрахунок витрат на проєктування

Усі необхідні дані подано в додатку Б.

Зарплата проектувальникам розраховується на двох осіб.

Потрібен розрахунок заробітної платні для двох працівників: програміст та тестувальник (таблиця 4.2).

Середня заробітна плата молодшого програміста становить – 16500 грн в місяць.

Податок фізичних осіб: 2970 грн.

Військовий збір: 495 грн.

Єдиний соціальний внесок.

Відрахування: 2970 грн + 495 грн = 3465 грн.

Від повної суми програміст отримає – 13035 грн (16500 – 3465).

Середня заробітна плата тестувальника становить 15000 грн.

Податок фізичної особи: 15000 * 18% = 2700 грн.

Військовий збір: 15000 * 3% = 450 грн.

Відрахування: 2700 + 450 = 3150 грн.

Від повної суми тестувальник отримає – 11850 грн (15000 - 3150).

Таблиця 4.2 – Розрахунок заробітної плати

№	Посада	Оклад, грн/міс	Відрахування грн/міс	Кількість		Сума з/п, грн.
				чол.	місяців	
1	Програміст	16500	3465	1	4	52140
2	Тестувальник	15000	3150	1	1	11850
Усього заробітної плати:						63990

Контрагентські роботи – 63990 * 12% = 7 678,8 грн.

Витрат на відрядження немає.

Інші прямі витрати становлять 63990 * 42% = 26875,8 грн.

Загальна сума витрат за чотири місяці виконання проекту становить 63990 + 7678,8 + 26875,8 = 98544,6 грн.

Накладні витрати за чотири місяці становлять 98544,5 * 30% = 29563,38 грн.

Планові накопичення протягом чотирьох місяців: (98544,5 + 29563,38) * 20% = 25621,6 грн.

Кошторисна вартість становить: 98544,5 + 29563,38 + 25621,6 = 153729,45 грн.

Податок на додану вартість 153729.45 * 20% = 30745.9 грн.

Договірна ціна проєкту становить: 153729,45 грн + 30745,9 грн = 184475,1 грн.

Кошторис проекту поданий у таблиці 4.2.

Таблиця 4.2 – Кошторис проекту

Найменування статей витрат	Сума, грн
Заробітна плата працівників	63990
Відрахування на соціальні потреби	6615
Контрагентські роботи	7678,8
Витрати на відрядження	0
Інші прямі витрати	26875,8
Усього прямих витрат	98544,6
Накладні витрати	29563,38
Планові накопичення	25621,6
Усього, кошторисна вартість проєкту	153729,45
Податок на додану вартість	30745,9
Договірна ціна проєкту	184475,34

4.3 Обґрунтування необхідності розробки

Будь-хто та будь-якого віку може використати готовий продукт цього проекту. Проте головною метою створення було полегшення та збільшення

ефективності роботи вчителям, це допоможе їм краще організувати свій час. Надасть можливість спростити навчання та модернізувати його.

Люди, які використовують застосунок зможуть тим самим допомогти екології, відмовившись від паперових засобів, цим самим автоматично зменшать вирубку лісів.

Правильно організувати свій час, це надважлива проблема в житті кожного з нас. В більшості відомих та заможних людей, кожен день розписаний на перед, дія за дією. Такий підхід допомагає людині зосередитись на завданні, правильно розприділити час між кожним завдання та зробити їх виконання максимально приємним та продуктивним.

Тому при грамотному управлінні, ця програма буде дуже перспективною, вона зможе приносити прибуток, крім того, вона буде корисною для потенційних користувачів.

ВИСНОВКИ

В ході дипломного проектування було створено мобільний додаток, що виконує функції органайзера керівника музичного гуртка.

Після ознайомлення з основними правилами і рекомендаціями з розробки й створення мобільних застосунків та огляду існуючих на даний час подібних рішень спроектовано та реалізовано органайзер, що відвідає вимогам замовника.

Зокрема, було реалізовано наступні функції:

- можливість для обліку відвідувань учасниками гуртка занять;
- відображення розкладу занять;
- можливість виставлення оцінок;
- надання кожному студенту визначеного статусу відповідно до його відвідування;
- наявність нагадування про наступні події;
- можливість перегляду, зберігання та редагування даних.

В якості інструментарії для реалізації мобільного додатку було використано мову програмування Java і середовище розробки Android Studio.

До переваг створеного мобільного органайзеру для керівника музичного гуртка можна віднести:

- не перевантаження зайвим функціоналом;
- реалізація функції надання статусу;
- не вимогливість до ресурсів системи;
- простий та інтуїтивно зрозумілий інтерфейс.

Тестування мобільного додатку підтвердило його адаптивність під різні технічні характеристики пристроїв.

За необхідності в додаток можуть бути включені додаткові функції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Типи мобільних додатків. *Smile-ukraine.com*: вебсайт. URL: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types (дата звернення: 23.02.2022).
2. Розробка мобільних додатків. *Webcase.com.ua*: вебсайт. URL: https://webcase.com.ua/uk/razrabotka-mobilnyh-prilozhenij/ (дата звернення: 27.02.2022).
3. Мобільний інтерфейс. *Wezom.com.ua*: вебсайт. URL: https://wezom.com.ua/blog/dizajn-interfejsov-mobilnyh-prilozhenij (дата звернення: 01.03.2022).
4. Мобільні додатки органайзери. *Zhyvyaktyvno.org*: вебсайт. URL: https://zhyvyaktyvno.org/news/mobln-dodatki-organajzeri (дата звернення: 05.03.2022).
5. Діаграма варіантів використання. *Studopedia.ru*: вебсайт. URL: https://studopedia.ru/19_284009_diagrama-variantiv-vikoristannya-USE-case-diagram.html (дата звернення: 15.15.1515).
6. Розробка UML діаграм. *Studfile.net*: вебсайт. URL: https://studfile.net/preview/5200239/page:6/ (дата звернення: 20.03.2022).
7. Керівництво по мові програмування Java. *Habr.com*: вебсайт. URL: https://habr.com/ru/hub/java/ (дата звернення: 03.04.2022).
8. Android Studio. *Metanit.com*: вебсайт. URL: https://metanit.com/java/android/ (дата звернення: 06.04.2022).

ДОДАТКИ

Додаток А

Програмний код

Лістинг А1 – Функція додавання лекцій

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_lesson);
        dbHelper = new DBHelper(this);
        name = findViewById(R.id.lesson_name_edit);
        group = findViewById(R.id.lesson_group_edit);
        datetime = findViewById(R.id.lesson_dattime_edit);
        new_lesson = findViewById(R.id.new_lesson_btn);

        new_lesson.setOnClickListener(new  View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                dbHelper.addLesson(name.getText().toString(),
group.getText().toString(), datetime.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }
```

Лістинг А2 – Функція додавання статусу

```
new_absence.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                List<student> stdns = dbHelper.getStudents();
                int rate = 0;
                for(int i = 0; i < stdns.size();i++)
                {
                    if(stdns.get(i).student_id                  ==
Integer.parseInt(student.getText().toString()))
                    {
                        rate                                     =
Integer.parseInt(stdns.get(i).student_status);
                    }
                }
                dbHelper.addAbsence(lesson_id,
student.getText().toString(), status.getText().toString());
                if(status.getText().toString().equals("+"))
                {
                    rate+=10;
                }
                else if(status.getText().toString().equals("-")){
```

Продовження лістингу А2

```
                    rate-=10;
                }
                dbHelper.updateStudent(String.valueOf(rate),
student.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
```

Лістинг А3 – Функція додавання оцінок

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_lessons_mark);
        dbHelper = new DBHelper(this);
        student = findViewById(R.id.student_id_mark_edit);
        status = findViewById(R.id.mark_value_edit);
        new_mark = findViewById(R.id.new_mark_btn);
        Intent intent = getIntent();
        Bundle bndl = intent.getBundleExtra("id");
        lesson_id = bndl.getString("id");

        new_mark.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                dbHelper.addMark(lesson_id,
student.getText().toString(), status.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }
```

Лістинг А4 – Функція додавання студентів

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_student);
        dbHelper = new DBHelper(this);
        name = findViewById(R.id.studnt_name_edit);
        group = findViewById(R.id.studnt_group_edit);
        new_student = findViewById(R.id.new_student_btn);

        new_student.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
```

Продовження лістингу А4

```
                dbHelper.addStudent(name.getText().toString(),
group.getText().toString());
                dbHelper.close();
                Intent intent = new Intent();
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }
```

Лістинг А5 – Клас SlidePageAdapter

```
public class SlidePageAdapter extends FragmentPagerAdapter {

    public SlidePageAdapter(@NotNull FragmentManager fm)
    {
        super(fm);
    }
    @NonNull
    @NotNull
    @Override
    public Fragment getItem(int position) {
        switch (position) {
            case 0:
                return new students_fragment();
            case 1:
                return new lessons_page();
            default:
                return new students_fragment();
        }

    }

    @Override
    public int getCount() {
        return 2;
    }
}
```

Лістинг А6 – Клас SlidePageAdapterLesson

```
public class SlidePageAdapterLesson extends SlidePageAdapter {
    String curr_lesson;
    public  SlidePageAdapterLesson(@NotNull  FragmentManager  fm,
int lsn_id) {
            super(fm);
            curr_lesson = String.valueOf(lsn_id);
        }

        @NonNull
        @NotNull
```

Продовження лістингу А6

```
        @Override
        public Fragment getItem(int position) {
            Bundle bundle = new Bundle();
            bundle.putString("id", curr_lesson);

            lesson_info_page_marks       marks       =       new
lesson_info_page_marks();
            marks.setArguments(bundle);

            lesson_info_page_absences     absences     =     new
lesson_info_page_absences();
            absences.setArguments(bundle);

            switch (position) {
                case 0:
                    return marks;
                case 1:
                    return absences;
                default:
                    return marks;
            }
        }

        @Override
        public int getCount() {
            return 2;
        }
}
```

Лістинг А7 – Клас SlidePageAdapterStudent

```
public class SlidePageAdapterStudent extends SlidePageAdapter {
    String curr_student;
    public  SlidePageAdapterStudent(@NotNull  FragmentManager  fm,
int lsn_id) {
            super(fm);
        curr_student = String.valueOf(lsn_id);
        }

        @NonNull
        @NotNull
        @Override
        public Fragment getItem(int position) {
            Bundle bundle = new Bundle();
            bundle.putString("id", curr_student);

            absences_page absences_page = new absences_page();
            absences_page.setArguments(bundle);

            marks_page marks_page = new marks_page();
            marks_page.setArguments(bundle);
```

Продовження лістингу А7

```
            switch (position) {
                case 0:
                    return marks_page;
                case 1:
                    return absences_page;
                default:
                    return marks_page;
            }
        }

        @Override
        public int getCount() {
            return 2;
        }
}
```

Лістинг А8 – Клас lesson_info_activity

```
public class lesson_info_activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lesson_info);
        Intent intent = getIntent();
        int id = Integer.parseInt(intent.getStringExtra("id"));
        SlidePageAdapter          adapter          =          new
SlidePageAdapterLesson(getSupportFragmentManager(), id);

        ViewPager viewPager = findViewById(R.id.pager_lesson);
        viewPager.setAdapter(adapter);
    }
}
```

Лістинг А9 – Клас student_info_activity

```
public class student_info_activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_student_info);
        Intent intent = getIntent();
        int id = Integer.parseInt(intent.getStringExtra("id"));
        SlidePageAdapterStudent       adapter       =       new
SlidePageAdapterStudent(getSupportFragmentManager(), id);

        ViewPager viewPager = findViewById(R.id.pager_students);
        viewPager.setAdapter(adapter); // устанавливаем адаптер
    }
}
```

Лістинг А10 – Клас MainActivity

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SlidePageAdapter adapter = new
SlidePageAdapter(getSupportFragmentManager());

        ViewPager viewPager = findViewById(R.id.pager);
        viewPager.setAdapter(adapter); // устанавливаем адаптер
    }
}
```

Лістинг А11 – Клас absence

```
package com.joreikarr.musicorganizer.entities;

public class absence {
    public int absence_id;
    public String lesson_id;
    public String student_id;
    public String absence_status;

    public absence(int absence_id, String lesson_id, String
student_id, String absence_status) {
        this.absence_id = absence_id;
        this.lesson_id = lesson_id;
        this.student_id = student_id;
        this.absence_status = absence_status;
    }

    public int getAbsence_id() {
        return absence_id;
    }
    public void setAbsence_id(int absence_id) {
        this.absence_id = absence_id;
    }
    public String getLesson_id() {
        return lesson_id;
    }
    public void setLesson_id(String lesson_id) {
        this.lesson_id = lesson_id;
    }
    public String getStudent_id() {
        return student_id;
    }
    public void setStudent_id(String student_id) {
        this.student_id = student_id;
    }
    public String getAbsence_status() {
```

Продовження лістингу А11

```
        return absence_status;
    }
    public void setAbsence_status(String absence_status) {
        this.absence_status = absence_status;
    }
}
```

Лістинг А12 – Клас lesson

```
package com.joreikarr.musicorganizer.entities;

public class lesson {
    public int lesson_id;
    public String lesson_name;
    public String lesson_group;
    public String lesson_date_time;

    public lesson(int lesson_id, String lesson_name, String
lesson_group, String lesson_date_time) {
        this.lesson_id = lesson_id;
        this.lesson_name = lesson_name;
        this.lesson_group = lesson_group;
        this.lesson_date_time = lesson_date_time;
    }

    public int getLesson_id() {
        return lesson_id;
    }
    public void setLesson_id(int lesson_id) {
        this.lesson_id = lesson_id;
    }
    public String getLesson_name() {
        return lesson_name;
    }
    public void setLesson_name(String lesson_name) {
        this.lesson_name = lesson_name;
    }
    public String getLesson_group() {
        return lesson_group;
    }
    public void setLesson_group(String lesson_group) {
        this.lesson_group = lesson_group;
    }
    public String getLesson_date_time() {
        return lesson_date_time;
    }
    public void setLesson_date_time(String lesson_date_time) {
        this.lesson_date_time = lesson_date_time;
    }
}
```

Лістинг А13 – Клас mark

```
package com.joreikarr.musicorganizer.entities;

public class mark {
    public int mark_id;
    public String lesson_id;
    public String student_id;
    public String mark_value;

    public mark(int mark_id, String lesson_id, String student_id,
String mark_value) {
        this.mark_id = mark_id;
        this.lesson_id = lesson_id;
        this.student_id = student_id;
        this.mark_value = mark_value;
    }


    public int getMark_id() {
        return mark_id;
    }
    public void setMark_id(int mark_id) {
        this.mark_id = mark_id;
    }
    public String getLesson_id() {
        return lesson_id;
    }
    public void setLesson_id(String lesson_id) {
        this.lesson_id = lesson_id;
    }
    public String getStudent_id() {
        return student_id;
    }
    public void setStudent_id(String student_id) {
        this.student_id = student_id;
    }
    public String getMark_value() {
        return mark_value;
    }
    public void setMark_value(String mark_value) {
        this.mark_value = mark_value;
    }
}
```

Лістинг А14 – Клас student

```
package com.joreikarr.musicorganizer.entities;

public class student {
    public int student_id;
    public String student_name;
    public String student_group;
```

Продовження лістингу А14

```
    public String student_status;

    public student(int student_id, String student_name, String
student_group, String student_status) {
        this.student_id = student_id;
        this.student_name = student_name;
        this.student_group = student_group;
        this.student_status = student_status;
    }

    public int getStudent_id() {
        return student_id;
    }
    public void setStudent_id(int student_id) {
        this.student_id = student_id;
    }
    public String getStudent_name() {
        return student_name;
    }
    public void setStudent_name(String student_name) {
        this.student_name = student_name;
    }
    public String getStudent_group() {
        return student_group;
    }
    public void setStudent_group(String student_group) {
        this.student_group = student_group;
    }
    public String getStudent_status() {
        return student_status;
    }
    public void setStudent_status(String student_status) {
        this.student_status = student_status;
    }
}
```

Лістинг А15 – Клас lesson_info_page_absences

```
package com.joreikarr.musicorganizer.fragments.lessons;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
```

Продовження лістингу А15

```
import androidx.fragment.app.Fragment;
import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
import
com.joreikarr.musicorganizer.activities.add_activities.activity_
add_lesson_absence;
import com.joreikarr.musicorganizer.entities.absence;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;
import java.util.List;

public class lesson_info_page_absences extends Fragment {

    ListView absences;
    Button add_btn;
    List<absence> absences_list;
    List<String> absences_strings;
    DBHelper dbHelper;
    String lesson_id;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view =
inflater.inflate(R.layout.lessons_page_absences, container,
false);
        absences = (ListView)
view.findViewById(R.id.lesson_absences_lw);
        add_btn = (Button)
view.findViewById(R.id.btn_add_lesson_absence);
        dbHelper = new DBHelper(getContext());
        lesson_id = getArguments().getString("id");
        setListView();

        add_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getActivity(),
activity_add_lesson_absence.class);
                Bundle bundle = new Bundle();
                bundle.putString("id", lesson_id);
                intent.putExtra("id", bundle);
                startActivityForResult(intent, 4);
            }
        });
        return view;
```

Продовження лістингу А15

```
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable @org.jetbrains.annotations.Nullable Intent data) {
        if(requestCode == 4)
        {
            setListView();
        }
    }

    public void setListView()
    {
        absences_strings = new ArrayList<>();
        absences_list = dbHelper.getAbsences();
        if(absences_list.size() > 0) {
            for (int i = 0; i < absences_list.size(); i++) {
                if
(absences_list.get(i).lesson_id.equals(String.valueOf(lesson_id)
)) {
                    String tmp = absences_list.get(i).student_id
+ " — " + absences_list.get(i).absence_status;
                    absences_strings.add(tmp);
                }
            }
        }
        ArrayAdapter<String>         adapter         =         new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
absences_strings);
        absences.setAdapter(adapter);
    }
}
```

Лістинг А16 – Клас lesson_info_page_marks

```
package com.joreikarr.musicorganizer.fragments.lessons;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
```

Продовження лістингу А16

```
import
com.joreikarr.musicorganizer.activities.add_activities.activity_
add_lesson_mark;
import
com.joreikarr.musicorganizer.fragments.main.students_fragment;
import com.joreikarr.musicorganizer.entities.mark;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;
import java.util.List;

public class lesson_info_page_marks extends Fragment {

    ListView marks;
    Button add_btn;
    List<mark> marks_list;
    List<String> marks_strings;
    DBHelper dbHelper;
    String lesson_id;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.lessons_page_marks,
container, false);
        marks                    =                    (ListView)
view.findViewById(R.id.lessons_marks_lw);
        add_btn                     =                     (Button)
view.findViewById(R.id.btn_add_lesson_mark);
        dbHelper = new DBHelper(getContext());
        lesson_id = getArguments().getString("id");
        setListView();

        add_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent   intent   =   new   Intent(getActivity(),
activity_add_lesson_mark.class);
                Bundle bundle = new Bundle();
                bundle.putString("id", lesson_id);
                intent.putExtra("id", bundle);
                startActivityForResult(intent, 3);
            }
        });
        return view;
    }
```

Продовження лістингу А16

```
    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable @org.jetbrains.annotations.Nullable Intent data) {
        if(requestCode == 3)
        {
            setListView();

        }
    }

    public void setListView()
    {
        marks_strings = new ArrayList<>();
        marks_list = dbHelper.getMarks();
        if(marks_list.size() > 0) {
            for (int i = 0; i < marks_list.size(); i++) {
                if
(marks_list.get(i).lesson_id.equals(String.valueOf(lesson_id))) {
                    String tmp = marks_list.get(i).student_id + "
— " + marks_list.get(i).mark_value;
                    marks_strings.add(tmp);
                }
            }
        }
        ArrayAdapter<String>         adapter         =         new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
marks_strings);
        marks.setAdapter(adapter);
    }
}
```

Лістинг А17 – Клас lessons_page

```
package com.joreikarr.musicorganizer.fragments.main;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
```

Продовження лістингу А17

```
import
com.joreikarr.musicorganizer.activities.add_activities.activity_
add_lesson;
import com.joreikarr.musicorganizer.entities.lesson;
import
com.joreikarr.musicorganizer.activities.show_activities.lesson_i
nfo_activity;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;
import java.util.List;

public class lessons_page extends Fragment {

    ListView lessons;
    Button add_btn;
    List<lesson> lessons_list;
    List<String> lessons_strings;
    DBHelper dbHelper;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.lessons_page,
container, false);
        lessons = (ListView) view.findViewById(R.id.lessons_lw);
        add_btn = (Button)
view.findViewById(R.id.btn_add_lesson);
        dbHelper = new DBHelper(getContext());
        setListView();


        add_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getActivity(),
activity_add_lesson.class);
                startActivityForResult(intent, 2);
            }
        });

        lessons.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView,
View view, int i, long l) {
```

Продовження лістингу А17

```
                Intent intent = new Intent(getContext(),
lesson_info_activity.class);
                intent.putExtra("id",
String.valueOf(lessons_list.get(i).lesson_id));
                startActivity(intent);
            }
        });
        return view;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable @org.jetbrains.annotations.Nullable Intent data) {
        if(requestCode == 2)
        {
            setListView();
        }
    }

    public void setListView()
    {
        lessons_strings = new ArrayList<>();
        lessons_list = dbHelper.getLessons();
        for(int i = 0; i < lessons_list.size(); i++)
        {
            String tmp = lessons_list.get(i).lesson_name + ",
група " + lessons_list.get(i).lesson_group +
                    " ("  + lessons_list.get(i).lesson_date_time
+ ")";
            lessons_strings.add(tmp);
        }
        ArrayAdapter<String> adapter = new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
lessons_strings);
        lessons.setAdapter(adapter);
    }
}
```

Лістинг А18 – Клас students_fragment

```
package com.joreikarr.musicorganizer.fragments.main;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
```

Продовження лістингу А18

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
import
com.joreikarr.musicorganizer.activities.add_activities.add_stude
nt_activity;
import
com.joreikarr.musicorganizer.activities.show_activities.lesson_i
nfo_activity;
import
com.joreikarr.musicorganizer.activities.show_activities.student_
info_activity;
import com.joreikarr.musicorganizer.entities.student;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;
import java.util.List;

public class students_fragment extends Fragment {
    boolean time_to_update = false;
    ListView students;
    Button add_btn;
    List<student> students_list;
    List<String> students_strings;
    DBHelper dbHelper;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.students_page,
container, false);
        students = (ListView)
view.findViewById(R.id.students_lw);
        add_btn = (Button)
view.findViewById(R.id.btn_add_student);
        dbHelper = new DBHelper(getContext());
        setListView();
        add_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getActivity(),
add_student_activity.class);
                startActivityForResult(intent, 1);
            }
        });
```

Продовження лістингу А18

```
        students.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView,
View view, int i, long l) {
                Intent intent = new Intent(getContext(),
student_info_activity.class);
                intent.putExtra("id",
String.valueOf(students_list.get(i).student_id));
                startActivity(intent);
            }
        });


        return view;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable @org.jetbrains.annotations.Nullable Intent data) {
        if(requestCode == 1 || requestCode == 2)
        {
            setListView();
        }

    }

    public void setListView()

    {
        students_strings = new ArrayList<>();
        students_list = dbHelper.getStudents();
        for(int i = 0; i < students_list.size();i++)
        {

            String tmp = students_list.get(i).student_id + ". " +
students_list.get(i).student_name + ", група " +
students_list.get(i).student_group +
             " ("  +students_list.get(i).student_status + ")";
            students_strings.add(tmp);
        }

        ArrayAdapter<String> adapter = new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
students_strings);
        students.setAdapter(adapter);
    }
}
```

Лістинг А19 – Клас absences_page

```
package com.joreikarr.musicorganizer.fragments.students;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
import com.joreikarr.musicorganizer.entities.absence;
import com.joreikarr.musicorganizer.entities.lesson;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.List;

public class absences_page extends Fragment {

    ListView absences;
    List<lesson> lessons_list;
    List<absence> absence_list;
    List<String> absence_strings;
    DBHelper dbHelper;
    String student_id;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.absences_page,
container, false);
        absences = (ListView)
view.findViewById(R.id.absences_lw);
        dbHelper = new DBHelper(getContext());
        student_id = getArguments().getString("id");
        setListView();
        return view;
    }
```

Продовження лістингу А19

```
        public void setListView()
        {
            absence_strings = new ArrayList<>();
            lessons_list = dbHelper.getLessons();
            absence_list = dbHelper.getAbsences();
            for(int i = 0; i < absence_list.size(); i++)
            {

if(absence_list.get(i).student_id.equals(student_id))
                {
                    for(int j = 0; j < lessons_list.size();j++)
                    {

if(String.valueOf(lessons_list.get(j).lesson_id).equals(absence_
list.get(i).
                                    lesson_id))
                            {
                                String              tmp             =
lessons_list.get(j).lesson_name + " (" +

lessons_list.get(j).lesson_date_time + ") — " +

absence_list.get(i).absence_status;
                                absence_strings.add(tmp);
                            }
                    }
                }
            }
            ArrayAdapter<String>       adapter       =       new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
absence_strings);
            absences.setAdapter(adapter);
        }
    }
```

Лістинг А20 – Клас marks_page

```
package com.joreikarr.musicorganizer.fragments.students;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import com.joreikarr.musicorganizer.DBHelper;
import com.joreikarr.musicorganizer.R;
```

Продовження лістингу А20

```
import com.joreikarr.musicorganizer.entities.lesson;
import com.joreikarr.musicorganizer.entities.mark;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;
import java.util.List;

public class marks_page extends Fragment {

    ListView marks;
    List<lesson> lessons_list;
    List<mark> mark_list;
    List<String> mark_strings;
    DBHelper dbHelper;
    String student_id;

    @Nullable
    @org.jetbrains.annotations.Nullable
    @Override
    public View onCreateView(@NonNull @NotNull LayoutInflater
inflater, @Nullable @org.jetbrains.annotations.Nullable ViewGroup
container, @Nullable @org.jetbrains.annotations.Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.marks_page,
container, false);
        marks = (ListView) view.findViewById(R.id.marks_lw);
        dbHelper = new DBHelper(getContext());
        student_id = getArguments().getString("id");
        setListView();
        return view;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable @org.jetbrains.annotations.Nullable Intent data) {
        if(requestCode == 2)
        {
            setListView();
        }
    }

    public void setListView()
    {
        lessons_list = dbHelper.getLessons();
        mark_list = dbHelper.getMarks();
        mark_strings = new ArrayList<>();
        for(int i = 0; i < mark_list.size(); i++)
        {
            if(mark_list.get(i).student_id.equals(student_id))
            {
                for(int j = 0; j < lessons_list.size();j++)
```

Продовження лістингу А20

```
                {

if(String.valueOf(lessons_list.get(j).lesson_id).equals(mark_lis
t.get(i).
                                lesson_id))
                    {
                        String                tmp                =
lessons_list.get(j).lesson_name + " (" +

lessons_list.get(j).lesson_date_time + ") — " +
                                    mark_list.get(i).mark_value;
                        mark_strings.add(tmp);
                    }
                }
            }
        }
        ArrayAdapter<String>         adapter         =         new
ArrayAdapter<>(getContext(), android.R.layout.simple_list_item_1,
mark_strings);
        marks.setAdapter(adapter);
    }
}
```

Лістинг А21 – Клас DBHelper

```
package com.joreikarr.musicorganizer;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.joreikarr.musicorganizer.entities.absence;
import com.joreikarr.musicorganizer.entities.lesson;
import com.joreikarr.musicorganizer.entities.mark;
import com.joreikarr.musicorganizer.entities.student;

import java.util.ArrayList;
import java.util.List;

public class DBHelper  extends SQLiteOpenHelper{

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "organizerDB";

    public static final String TABLE_STUDENTS = "students";

    public static final String STUDENT_KEY_ID = "_id";
    public static final String STUDENT_KEY_NAME = "name_student";
```

Продовження лістингу А21

```
    public static final String STUDENT_KEY_GROUP =
"group_student";
    public static final String STUDENT_KEY_STATUS =
"status_student";


    public static final String TABLE_LESSONS = "lessons";

    public static final String LESSON_KEY_ID = "_id";
    public static final String LESSON_KEY_NAME = "name_lesson";
    public static final String LESSON_KEY_GROUP = "group_lesson";
    public static final String LESSON_KEY_DATE_TIME =
"date_time_lesson";


    public static final String TABLE_ABSENCES = "absences";

    public static final String ABSENCE_KEY_ID = "_id";
    public static final String ABSENCE_KEY_LESSON =
"lesson_absences";
    public static final String ABSENCE_KEY_STUDENT =
"student_absences";
    public static final String ABSENCE_KEY_STATUS =
"status_absences";


    public static final String TABLE_MARKS = "marks";

    public static final String MARKS_KEY_ID = "_id";
    public static final String MARKS_KEY_LESSON = "lesson_marks";
    public static final String MARKS_KEY_STUDENT =
"student_marks";
    public static final String MARKS_KEY_VALUE = "value_marks";




    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(
                "create table " +
                TABLE_STUDENTS +
                "(" +
                STUDENT_KEY_ID + " integer primary key
autoincrement," +
                STUDENT_KEY_NAME + " text," +
```

Продовження лістингу А21

```
                STUDENT_KEY_GROUP + " text," +
                STUDENT_KEY_STATUS + " text" +
                ")"
        );

        db.execSQL(
                "create table " +
                        TABLE_LESSONS +
                        "(" +
                        LESSON_KEY_ID  +  "  integer  primary  key
autoincrement," +
                        LESSON_KEY_NAME + " text," +
                        LESSON_KEY_GROUP + " text," +
                        LESSON_KEY_DATE_TIME + " text" +
                        ")"
        );

        db.execSQL(
                "create table " +
                        TABLE_ABSENCES +
                        "(" +
                        ABSENCE_KEY_ID  +  "  integer  primary  key
autoincrement," +
                        ABSENCE_KEY_LESSON + " text," +
                        ABSENCE_KEY_STUDENT + " text," +
                        ABSENCE_KEY_STATUS + " text" +
                        ")"
        );


        db.execSQL(
                "create table " +
                        TABLE_MARKS +
                        "(" +
                        MARKS_KEY_ID  +  "  integer  primary  key
autoincrement," +
                        MARKS_KEY_LESSON + " text," +
                        MARKS_KEY_STUDENT + " text," +
                        MARKS_KEY_VALUE + " text" +
                        ")"
        );



    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("drop table if exists " + TABLE_STUDENTS);
        db.execSQL("drop table if exists " + TABLE_LESSONS);
```

Продовження лістингу А21

```
        db.execSQL("drop table if exists " + TABLE_ABSENCES);
        db.execSQL("drop table if exists " + TABLE_MARKS);

        onCreate(db);

    }



    public void updateStudent(String rate, String id)
    {
        SQLiteDatabase database = getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DBHelper.STUDENT_KEY_STATUS, rate);
        int updCount = database.update(DBHelper.TABLE_STUDENTS,
contentValues, DBHelper.STUDENT_KEY_ID + "= ?", new String[]
{id});
        close();
    }


    public void addStudent(String name, String group)
    {
        SQLiteDatabase database = getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DBHelper.STUDENT_KEY_NAME, name);
        contentValues.put(DBHelper.STUDENT_KEY_GROUP, group);
        contentValues.put(DBHelper.STUDENT_KEY_STATUS, "0");
        database.insert(DBHelper.TABLE_STUDENTS,            null,
contentValues);
        close();
    }

    public void addLesson(String name, String group, String
date_time)
    {
        SQLiteDatabase database = getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DBHelper.LESSON_KEY_NAME, name);
        contentValues.put(DBHelper.LESSON_KEY_GROUP, group);
        contentValues.put(DBHelper.LESSON_KEY_DATE_TIME,
date_time);
        database.insert(DBHelper.TABLE_LESSONS,             null,
contentValues);
        close();
    }
    public void addAbsence(String lesson, String student, String
status)
    {
        SQLiteDatabase database = getWritableDatabase();
```

Продовження лістингу А21

```
        ContentValues contentValues = new ContentValues();
        contentValues.put(DBHelper.ABSENCE_KEY_LESSON, lesson);
        contentValues.put(DBHelper.ABSENCE_KEY_STUDENT, student);
        contentValues.put(DBHelper.ABSENCE_KEY_STATUS, status);
        database.insert(DBHelper.TABLE_ABSENCES,            null,
contentValues);
        close();
    }


    public void addMark(String lesson, String student, String
mark)
    {
        SQLiteDatabase database = getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DBHelper.MARKS_KEY_LESSON, lesson);
        contentValues.put(DBHelper.MARKS_KEY_STUDENT, student);
        contentValues.put(DBHelper.MARKS_KEY_VALUE, mark);
        database.insert(DBHelper.TABLE_MARKS,               null,
contentValues);
        close();
    }

    public List<student> getStudents()
    {
        List<student> students = new ArrayList<>();
        SQLiteDatabase database = getWritableDatabase();
        Cursor cursor = database.query(DBHelper.TABLE_STUDENTS,
null, null, null, null, null, null);
        if (cursor.moveToFirst())
        {
            int                    idIndex                    =
cursor.getColumnIndex(DBHelper.STUDENT_KEY_ID);
            int                  nameIndex                    =
cursor.getColumnIndex(DBHelper.STUDENT_KEY_NAME);
            int                 groupIndex                    =
cursor.getColumnIndex(DBHelper.STUDENT_KEY_GROUP);
            int                statusIndex                    =
cursor.getColumnIndex(DBHelper.STUDENT_KEY_STATUS);
            do {
                student tmp = new student(cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        cursor.getString(groupIndex),
                        cursor.getString(statusIndex));
                students.add(tmp);
            }
            while (cursor.moveToNext());
        }
        cursor.close();
        return students;
```

Продовження лістингу А21

```
    }


    public List<lesson> getLessons()
    {
        List<lesson> lessons = new ArrayList<>();
        SQLiteDatabase database = getWritableDatabase();
        Cursor  cursor  =  database.query(DBHelper.TABLE_LESSONS,
null, null, null, null, null, null);
        if (cursor.moveToFirst())
        {
            int                     idIndex                     =
cursor.getColumnIndex(DBHelper.LESSON_KEY_ID);
            int                    nameIndex                    =
cursor.getColumnIndex(DBHelper.LESSON_KEY_NAME);
            int                   groupIndex                    =
cursor.getColumnIndex(DBHelper.LESSON_KEY_GROUP);
            int                 datetimeIndex                   =
cursor.getColumnIndex(DBHelper.LESSON_KEY_DATE_TIME);
            do {
                lesson tmp = new lesson(cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        cursor.getString(groupIndex),
                        cursor.getString(datetimeIndex));
                lessons.add(tmp);
            }
            while (cursor.moveToNext());
        }
        cursor.close();
        return lessons;
    }

    public List<absence> getAbsences()
    {
        List<absence> absences = new ArrayList<>();
        SQLiteDatabase database = getWritableDatabase();
        Cursor  cursor  =  database.query(DBHelper.TABLE_ABSENCES,
null, null, null, null, null, null);
        if (cursor.moveToFirst())
        {
            int                     idIndex                     =
cursor.getColumnIndex(DBHelper.ABSENCE_KEY_ID);
            int                    nameIndex                    =
cursor.getColumnIndex(DBHelper.ABSENCE_KEY_LESSON);
            int                   groupIndex                    =
cursor.getColumnIndex(DBHelper.ABSENCE_KEY_STUDENT);
            int                 datetimeIndex                   =
cursor.getColumnIndex(DBHelper.ABSENCE_KEY_STATUS);
            do {
                absence tmp = new absence(cursor.getInt(idIndex),
```

Продовження лістингу А21

```
                        cursor.getString(nameIndex),
                        cursor.getString(groupIndex),
                        cursor.getString(datetimeIndex));
                absences.add(tmp);
            }
            while (cursor.moveToNext());
        }
        cursor.close();
        return absences;
    }


    public List<mark> getMarks()
    {
        List<mark> marks = new ArrayList<>();
        SQLiteDatabase database = getWritableDatabase();
        Cursor   cursor   =   database.query(DBHelper.TABLE_MARKS,
null, null, null, null, null, null);
        if (cursor.moveToFirst())
        {
            int                      idIndex                    =
cursor.getColumnIndex(DBHelper.MARKS_KEY_ID);
            int                    nameIndex                    =
cursor.getColumnIndex(DBHelper.MARKS_KEY_LESSON);
            int                   groupIndex                    =
cursor.getColumnIndex(DBHelper.MARKS_KEY_STUDENT);
            int                  datetimeIndex                  =
cursor.getColumnIndex(DBHelper.MARKS_KEY_VALUE);
            do {
                mark tmp = new mark(cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        cursor.getString(groupIndex),
                        cursor.getString(datetimeIndex));
                marks.add(tmp);
            }
            while (cursor.moveToNext());
        }
        cursor.close();
        return marks;
    }
```

Додаток Б

Необхідні дані для обчислення

Таблиця Б.1 – Дані для обчислення

Назва	% вирахування
Податок фізичних осіб:	18%
Військовий збір:	3%
Контрагентські роботи:	12%