МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ЛЯСКОВЕЦЬ Мирослав Сергійович

Програма "Електронний журнал" для ОС "Android"/ Software "e-register" for OS Android

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
М. С. Ляскoвець

_____

Науковий керівник:
к.т.н., доцент МАРЦЕНЮК Є.О.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 70 сторінок, 9 таблиць, 28 рисунків, список використаних джерел із 18 найменувань.

Метою дипломної роботи є розробка мобільного програмного продукту «Електронний журнал» для ОС Android.

Об'єктом дослідження є процес внесення інформації про проведені заняття, оцінювання знань студентів та дисципліни.

Предметом дослідження є програмний продукт «Електронний журнал» для ОС Android.

Методи розробки базуються на технології Java.

Одержані результати полягають в розробці мобільного програмного забезпечення «Електронний журнал» для ОС Android, яке може використовуватися на мобільних операційних системах.

Ключові слова: електронний журнал, академічна група, тема заняття, оцінка.

RESUME

Thesis contains 70 pages, 9 tables, 28 figures, list of references with 18 titles.

The aim of the thesis is to develop mobile software "Electronic Journal" for Android.

The object of research is the process of making information about ongoing classes, assessment of student learning and discipline.

The subject of the study is the software "Electronic Journal" for Android.

The methods of making technology based on Java.

The results are in developing mobile software "Electronic Journal" for Android, which can be used on mobile operating systems.

Keywords: e-magazine, an academic group, topic sessions, evaluation.

ЗМІСТ

ВСТУП

Актуальність теми

Електронний журнал обліку навчальної роботи студентів академічної групи вводиться з метою здійснення моніторингу навчального процесу та підготовки інформації для прийняття рішень щодо його удосконалення.

Журнал може бути складовою системи автоматизації управління університетом в сфері організації академічної діяльності та являє собою сучасну і удосконалену форму обліку навчальної діяльності студентів, передбачену інструктивними матеріалами

Введення журналу спрямовано на:

- забезпечення відкритості та прозорості навчального процесу ;
- формування зворотного зв’язку між студентами та викладачами і адміністрацією університету;
- відображення особистісного та професійного зростання кожного студента;
- забезпечення контролю за виконанням своїх обов’язків усіма сторонами навчального процесу.

Мета і задачі розробки

Метою розробки є створення мобільного програмного забезпечення для ведення обліку відвідування студентами академічної групи занять. Програма повинна відповідати наступним критеріям:

- мати зручну та зрозумілу форму для введення інформації про відвідування студентами занять;
- мати можливість генерувати окремі складові інформаційного простору в електронному вигляді;
- мати можливість генерувати звіти в контексті групи, дисципліни, або окремого студента;

В процесі розробки необхідно вирішити наступні задачі:

- зробити огляд існуючих мобільних програм для обліку відвідуваності в академічній групі;
- визначити необхідні функції для реалізації своєї програми «Електронний журнал»;
- провести тестування та відлагодження програмного засобу;

Методи, засоби та технології розробки

У даному випадку було вибрано методи аналізу та синтезу. Аналіз — це метод пізнання, який дає змогу поділити предмет на частини. Синтез, навпаки, є наслідком з'єднання окремих частин чи рис предмета в єдине ціле.

Аналіз та синтез взаємопов'язані, вони являють собою єдність протилежностей. Залежно від рівня пізнання об'єкта та глибини проникнення в його сутність застосовуються аналіз і синтез різного роду.

Практичне значення одержаних результатів

Практичне значення розробки полягає у створенні мобільної програми для операційної системи Андроїд, що здійснює облік відвідуваності в академічній групі. Основною перевагою такої програми є її мобільність.

РОЗДІЛ І
РОЗДІЛ АНАЛІЗУ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СПЕЦИФІКАЦІЇ ВИМОГ ПРОГРАМИ «ЕЛЕКТРОННИЙ ЖУРНАЛ»

1.1. Коротка характеристика об’єкту управління

Журнал обліку роботи академічної групи та викладачів (далі - журнал) - обов'язковий документ, в якому фіксуються результати навчальних досягнень студентів, відвідування ними занять, виконання навчальних програм тощо.

Наявність журналу на занятті обов'язкова. У позанавчальний час журнал зберігається в навчальній частині (відділенні).

Журнал складається з розділів: "Зміст", "Облік проведення занять, їх відвідування та успішність студентів", "Виконання курсових робіт (проектів), лабораторно-практичних та семінарських робіт", "Зведена відомість обліку успішності студентів та відвідування занять", "Зауваження до ведення журналу".

Записи в журналі ведуться українською мовою. Частково допускається запис змісту заняття та самостійної роботи студентів мовою вивчення предмета (навчальної дисципліни).

Записи проводяться чорнилом (стержнем) однакового кольору, чітко й охайно. На сторінках журналу не допускаються будь-які виправлення. У разі помилкового або неправильного запису виправлення погоджуються із завідувачем кафедри.

Оцінки успішності студентів проставляються за 100-бальною шкалою з предметів загальноосвітньої підготовки, з дисциплін освітньо-професійної програми - за національною шкалою (4 бальною) або шкалою ЄКТС.

У разі неатестації студента робиться відповідний запис: н. а. (не атестований(а)).

У випадках, коли студенти звільнені за станом здоров'я від занять із фізичної культури (фізичного виховання), під час виставляння підсумкових оцінок робиться відповідний запис: зв. (звільнений(а)).

У графі "зміст заняття" відповідно до робочої навчальної програми стисло записується тема заняття, практичної, лабораторної робіт тощо.

У графі "Самостійна робота" стисло записується зміст домашнього завдання (прочитати, вивчити напам'ять, повторити тощо), параграфи (сторінки) підручника, номери завдань, вправ тощо.

У розділі "Облік виконання курсових проектів, лабораторно-практичних та графічних робіт" ведеться облік виконання студентами передбачених навчальним планом і чинними програмами лабораторних, практичних, графічних робіт та курсових проектів.

Лівий бік сторінок цього розділу:

ведеться облік виконання робіт студентами;

після завершення курсу всіх передбачених робіт викладач пише слово "зараховано".

Правий бік сторінок цього розділу:

ведеться запис робіт із зазначенням дати видачі і фактичного їх виконання.

Оцінка за лабораторно-практичні роботи переноситься на основну сторінку предмета (дисципліни) (розділ "Облік проведення занять, їх відвідування та успішності студентів").

Бали за державну підсумкову атестацію виставляються у колонку з надписом ДПА без зазначення дати після підсумкової оцінки.

1.2. Огляд і аналіз існуючих аналогів, що реалізують функції «Електронного журналу» для ОС Андроїд

1.2.1. Програма «Журнал преподавателя (PRO)»

Програмний продукт розташований на електронному ресурсі https://play.google.com/store/apps/details?id=com.drprog.sjournal.full.

Ця програма має дуже багато різноманітних функцій, має багатий інтерфейс та велику кількість інструментів для заповнення різних довідників, які потім використовуються для формування записів.

Автор програмного продукту «d_romka» вказує наступну інформацію про продукт:

Програма призначена для викладачів університетів. Надає можливість вести журнали відвідування та успішності учнів.

Додаткові можливості:

- настроюється зведена відомість;
- студенти можуть бути зараховані в кілька груп / підгруп одночасно;
- настройка графічного представлення журналу (таблиці);
- імпорт записів про студентів / групах з файлів;
- збереження / відновлення бази даних (SD-card).
- графа "Середнє значення";
- синхронізація з календарем (подія календаря повинно містити назву групи, а також абревіатури дисципліни і виду заняття, наприклад, на 4 парі Лекція по Вищій математиці (Вм) в групі ЕТ-01-1 => "4п. Вм, ЕТ-01-1, Лк").

Зовнішній вигляд основного вікна програми показано на рисунку1.1

№	Ф.И.О.	Лб 1 24.09.2013	Лб 2 01.10.2013	Захист Лб 1 08.10.2013	Захист Лб 2 15.10.2013	Лб 3 22.10.2013	Лб 4 29.10.2013	Лб 5.1 05.11.2013	Лб 5.2 12.11.20
1	Белый Л.Г.			4	4				
2	Богданов С.С.			4	4	н	н	н	
3	Буц В.В.	н	н	4	4	н	н	н	н
4	Волошин А.Ю.		н	4	4		н		н
5	Иванов И.И.		н	4	4				н
6	Крамарев В.С.	н	н	4	4	н	н	н	н
7	Кулинич А.Ю.			4	4				
8	Петров Д.П.			4	4	6	н		н
9	Потапов Ю.В.			4	4	н	н	н	
10	Приморский В.А.								н
11	Романов С.М.	н	н	4	4	н	н	н	н
12	Степанов В.В.			4	4	н	н		н
	Отсутствуют	3	5	0	0	6	8	5	8

Рисунок 1.1- Основне вікно програми «Журнал преподавателя (PRO)»

На рисунку 1.2 наведено приклад проведення перекличкі де зазначено дату, номер лекції, та відзначено відсутні студенти.

Рисунок 1.2- Процес проведення перекличкі в програмі «Журнал преподавателя (PRO)»

1.2.2. Програма для обліку в академічній групі «Журнал преподавателя» ProSoft-Design

Розробник цієї програми розмістив свій продукт за електронною адресою https://play.google.com/store/apps/details?id=com.teacher_journal_pro

Приклад сновиго робочого вікна показаний на рисунку 1.3. Тут наведено приклад відомості про роботу академічної групи.

До даного програмного продукту автор наводить наступну інформацію:

Цей додаток призначений на допомогу викладачеві для ведення обліку відвідування та результативності роботи студентів на заняттях.

У розширеній версії доступні функції:

- необмежена кількість груп, підгруп і студентів;
- імпорт списку груп, підгруп і студентів з CSV-файлу;
- експорт журналу в формат Excel (XLS).

Рисунок 1.3- Основна сторінка програми «Журнал преподавателя» ProSoft-Design

1.2.3. Програма «Мобильный журнал» Dnevnik.ru

Розробник цієї програми розмістив свій продукт за електронною адресою https://play.google.com/store/apps/details?id=com.teacher_journal_pro

Приклад сновиго робочого вікна показаний на рисунку 1.4. Тут наведено приклад відомості про роботу академічної групи.

В тексті пояснення до програми автори зазначають наступне:

Додаток «Мобільний журнал» може бути використано на планшетних пристроях, що працюють під операційною системою Android.

Завдяки додатку «Мобільний журнал» викладачі можуть в режимі реального часу:

• виставляти оцінки;

• відзначати присутніх на занятті;

• залишати коментарі про роботу студентів;

• створювати і видавати домашні завдання;

• записувати і редагувати теми занять;

• створювати, редагувати і видаляти роботи на занятті.

«Мобільний журнал» містить вбудовану інструкцію, яка допоможе познайомитися з його інтерфейсом. При першому вході викладач буде запропоновано запустити її і вивчити основні елементи управління журналом. Також для полегшення роботи з журналом в додатку реалізований підрахунок середнього бала студентів, а також зручний перехід між режимами виставлення оцінок учням та за уроками.

Перед початком роботи з додатком «Мобільний журнал» ми рекомендуємо ознайомитися з наступними умовами експлуатації:

УМОВА 1: Додаток має бути встановлено на мобільний пристрій. УМОВА 2: Додаток призначений для роботи на планшетних комп'ютерах (або інших мобільних пристроях, що відповідають мінімальним вимогам,

зазначеним нижче) з діагоналлю екрана 7 "і більш, що працюють під операційною системою Android v. 4.0 і вище.

УМОВА 3: Робота з додатком можлива тільки в онлайн режимі, тобто планшетний пристрій в момент роботи з журналом має бути підключений до мережі Інтернет.

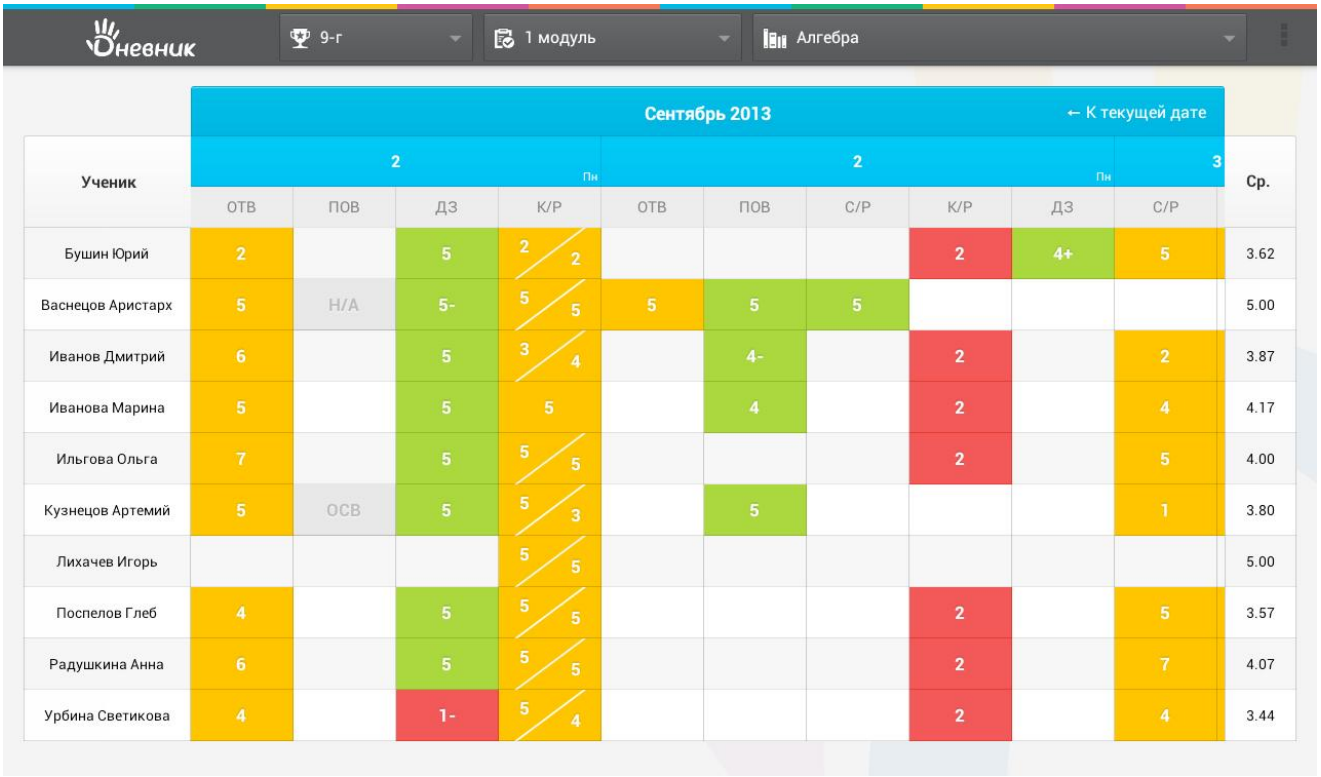


Рисунок 1.4- Основна сторінка програми «Мобильный журнал» Dnevnik.ru

1.3. Специфікація вимог до системи «Електронний журнал»

Метою створення специфікації вимог до системи є визначення функцій системи та її не функціональних вимог. Першим кроком створення специфікації є опис глосарію проекту, який відображений у таблиці 1.1

Таблиця 1.1

Глосарій проекту

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Журнал обліку роботи академічної групи	Документ де вноситься інформація про проведені заняття (дата, місце, відвідуваність, оцінки)
Електронний Журнал	Журнал в якому вся інформація про проведені заняття записуються в електронному вигляді
Тема заняття	Тематика за якою проводяться заняття відповідно до навчального плану та робочої програми
Система оцінок	Розрядність системи балів за якою виставляються оцінки
Академічна група	Перелік слухачів (студентів), які відвідують заняття разом
Оцінка	Виставлене символьне значення с контексті «системи оцінок», яке показує рівень знань
Вид заняття	Встановлений порядок взаємодії викладач ата студентів. Бувають (лекції, практичні, лабораторні, семінари)
2. Користувачі системи	
Викладач	Працівник вищої, середньої спеціальної, професійно-технічної або загальноосвітньої школи, ведучий який-небудь учбовий предмет
3. Вхідні та вихідні документи	
Панель керування	Електронна форма для переходу до основних функцій
Список групи	Документ в якому наведено повний список

	групи студентів
Перелік дисциплін	Частина навчального плану в якій навдено список навчальних дисциплін
Зміст заняття	Методичні матеріали необхідні для проведення заняття
Звіт	Звіт з існуючої бази даних в певних розрізах та об’ємах

Опрацювавши джерела, що наведені вище приймаємо наступну специфіка вимог до системи електронного журналу. Вимоги будемо специфікувати за допомогою діаграми варіантів використання.

Діаграми варіантів використання описують взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі. Важливо розуміти, що діаграми варіантів використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи. Діаграми варіантів використання призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами, і особливо знадобляться для визначення необхідних характеристик системи. Іншими словами, діаграми варіантів використання говорять про те, що система повинна робити, не вказуючи самі застосовувані методи.

Варіант використання описує, з точки зору чинного особи, групу дій в системі, які призводять до конкретного результату.

Варіанти використання є описами типових взаємодій між користувачами системи і самою системою. Вони відображають зовнішній інтерфейс системи і вказують форму того, що система повинна зробити (саме що, а не як).

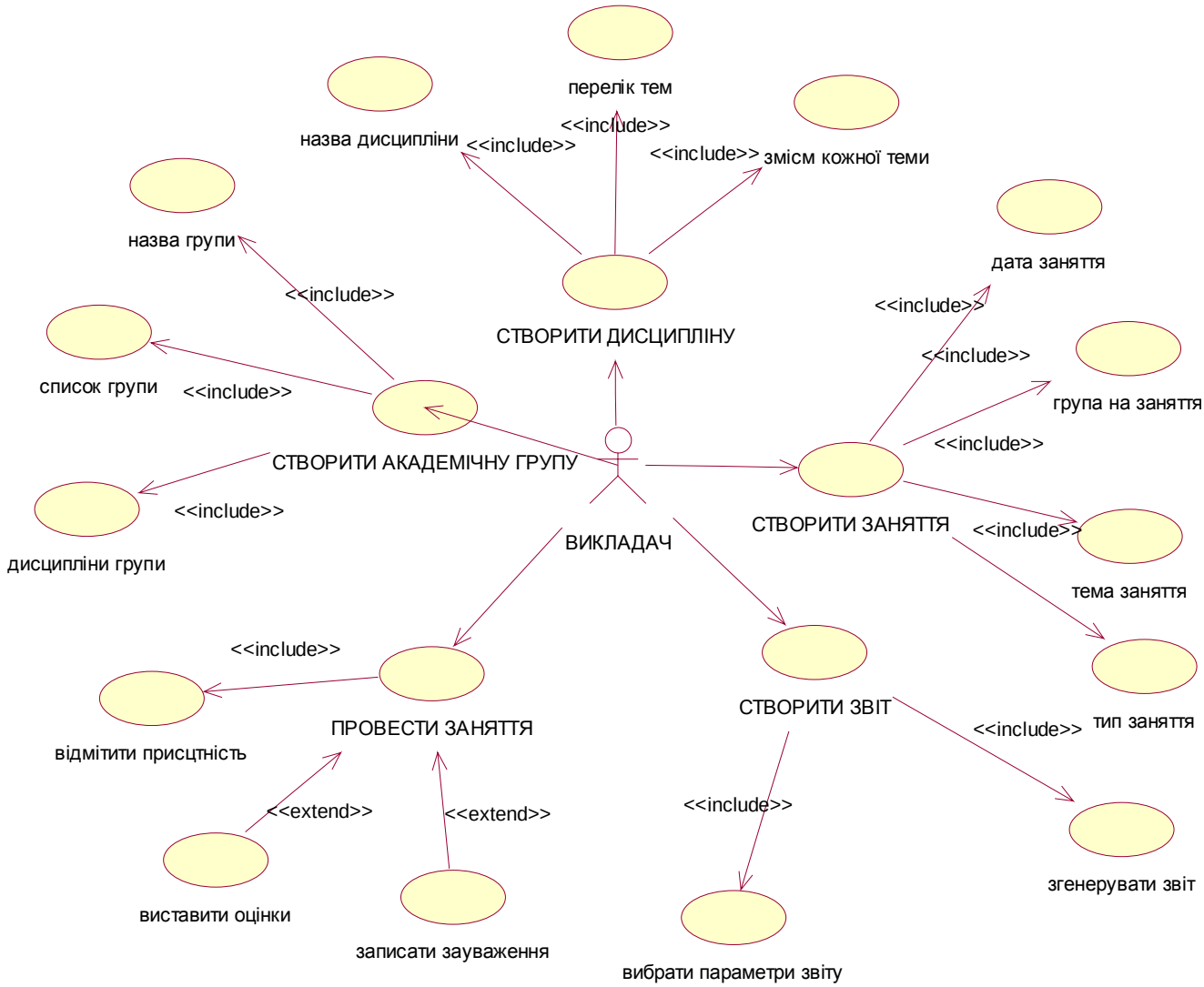


Рисунок 1.5.- Загальна діаграма варіантів використання «Електронного журналу»

При роботі з варіантами використання важливо пам'ятати декілька простих правил:

- кожен варіант використання відноситься як мінімум до одного дійової особи,
- кожен варіант використання має ініціатора,
- кожен варіант використання призводить до відповідного результату (результату з «бізнес-значенням»).

Варіанти використання також можуть взаємодіяти з іншими варіантами використання.

Єдиний актор проектованої системи, буде взаємодіяти із всіма варіантами використання. У модель включено асоціації, що відповідають за напрямки передачі інформації між актором і варіантами використання.

На рисунку 1.5 зображено загальну діаграму варіантів використання проектованого програмного забезпечення.

Далі наводимо опис варіантів використання, що реалізують основну функціональність у вигляді таблиць.

Таблиця 1.2

Варіант використання «СТВОРИТИ ДИСЦИПЛІНУ»

Контекст використання	Необхідність проведення занять за даною дисципліною
Дійові особи	Користувач системи (викладач)
Передумови	Початок роботи з системою
Триггер	Запуск програми на виконання
Сценарій	Внести назву дисципліни Внести перелік тем Внести зміст тем Зберегти дисципліну
Постумова	В системі створено нову дисципліну

Таблиця 1.3

Варіант використання «СТВОРИТИ АКАДЕМІЧНУ ГРУПУ»

Контекст використання	Необхідність мати інформацію про академічну групу
Дійові особи	Користувач системи (викладач)
Передумови	Необхідність створити документ з переліком академічної групи
Триггер	Робота з академічною групою
Сценарій	Ввести назву групи

	Ввести студентів групи Зберегти інформацію про групу
Постумова	Отримання категорії «Група» для подальшого використання

Таблиця 1.4

Варіант використання «СТВОРИТИ ЗАНЯТТЯ»

Контекст використання	Необхідність мати інформацію про проведене заняття
Дійові особи	Користувач системи (викладач)
Передумови	Необхідність створити документ з переліком проведених занять
Триггер	Робота з академічною групою
Сценарій	Вибрати дату проведення Вибрати групу на заняттях Вибрати тему заняття Вибрати тип заняття
Постумова	Отримання категорії «Заняття» для подальшого використання

Таблиця 1.5

Варіант використання «ПРОВЕСТИ ЗАНЯТТЯ»

Контекст використання	Необхідність внести інформацію про проведене заняття
Дійові особи	Користувач системи (викладач)
Передумови	Необхідність доповнити інформацію в документ з переліком проведених занять
Триггер	Робота з академічною групою
Сценарій	Вибрати студента

	Записати інформацію про студента
Постумова	Доповнення категорії «Заняття» для подальшого використання

Таблиця 1.6

Варіант використання «СТВОРИТИ ЗВІТ»

Контекст використання	Створення звіту за певний період та і певному вигляді
Дійові особи	Користувач системи (викладач)
Передумови	Перейти в режим створення звіту
Триггер	Завдання параметрів звіту
Сценарій	Вибрати період звіту Вибрати вид звіту Вибрати форму відображення звіту
Постумова	Отримати візуалізацію звіту за вибраними параметрами

Далі виконаємо розкадровку варіантів використання для опису необхідних екранних форм, що використовуються для реалізації функцій.

Розкадровка варіанту використання «СТВОРИТИ ДИСЦИПЛІНУ» показана на рисунку 1.6.

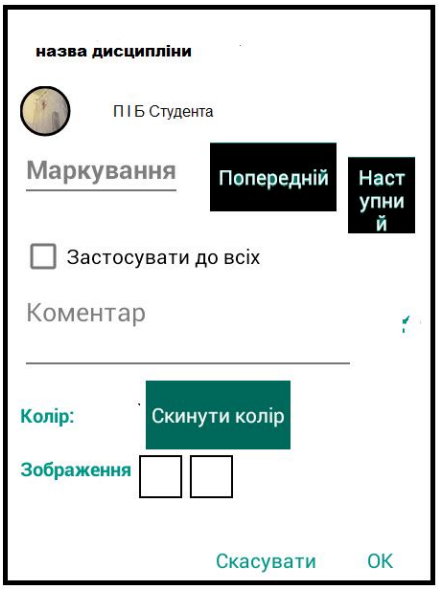


Рисунок 1.6.- Розкадровка варіанту використання «СТВОРИТИ ДИСЦИПЛІНУ»

Розкадровка варіанту використання «СТВОРИТИ АКАДЕМІЧНУ ГРУПУ» показана на рисунку 1.7.

Рисунок 1.7.- Розкадровка варіанту використання «СТВОРИТИ АКАДЕМІЧНУ ГРУПУ»

Розкадровка варіанту використання «СТВОРИТИ ЗАНЯТТЯ» показана на рисунку 1.8.

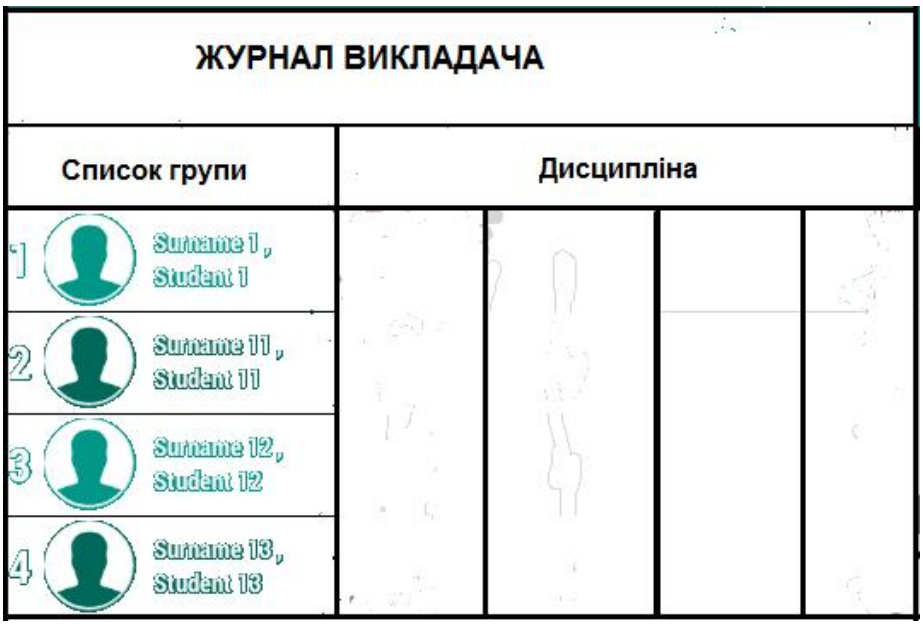


Рисунок 1.8.- Розкадровка варіанту використання «СТВОРИТИ ЗАНЯТТЯ»

Розкадровка варіанту використання «ПРОВЕСТИ ЗАНЯТТЯ» показана на рисунку 1.9.

Рисунок 1.9.- Розкадровка варіанту використання «ПРОВЕСТИ ЗАНЯТТЯ»

Розкадровка варіанту використання «СТВОРИТИ ЗВІТ» показана на рисунку 1.10.

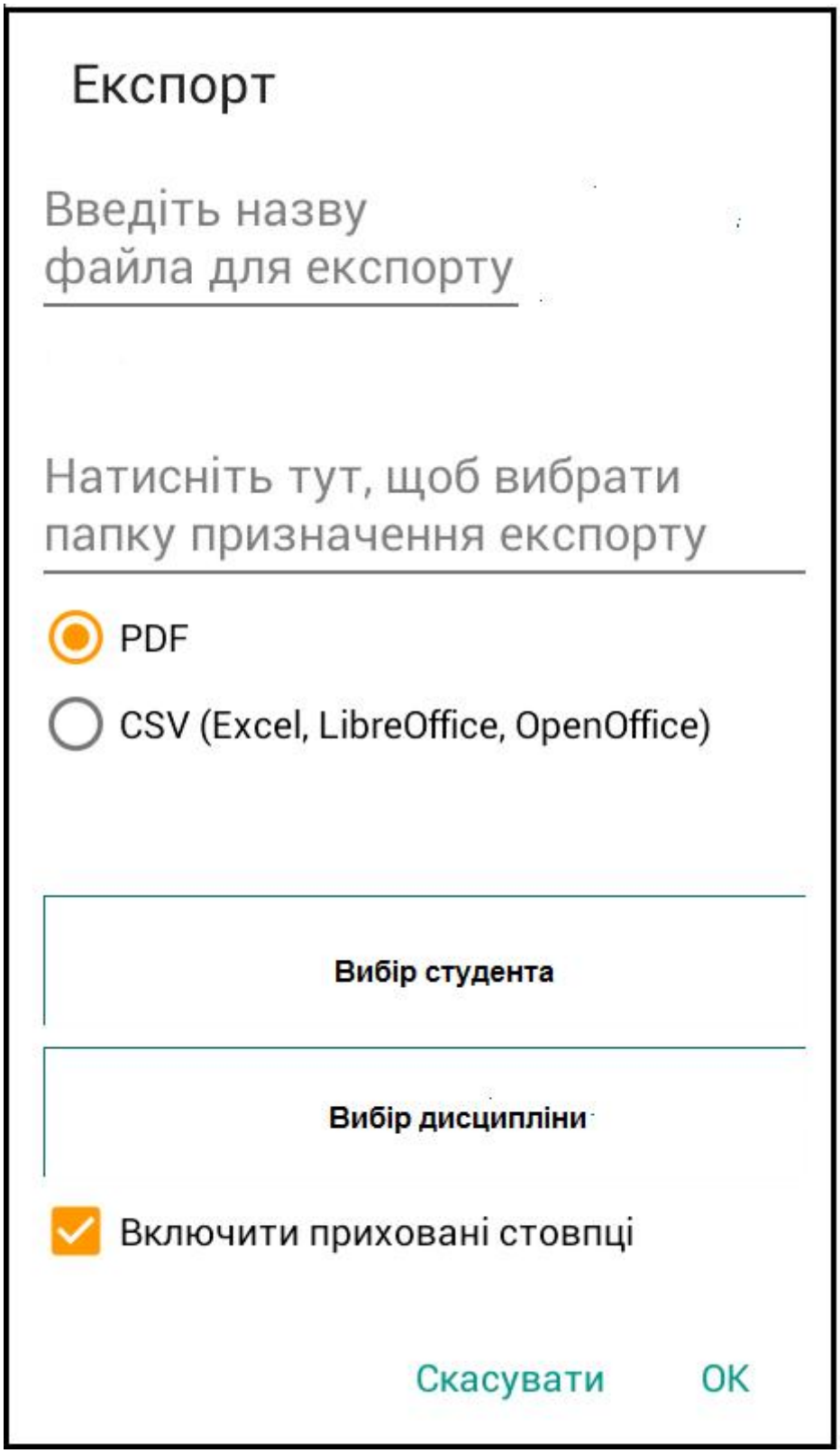


Рисунок 1.10.- Розкадровка варіанту використання «СТВОРИТИ ЗВІТ»

В результаті опису отримуємо специфікацію функціональних та нефункціональних вимог відповідно таблиця 1.9 та таблиця 1.10.

Таблиця 1.9

Специфікацію функціональних вимог

Іденти-фікатор вимог	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	СТВОРИТИ ДИСЦИПЛІНУ	Обов’язкова	Середня	
2	СТВОРИТИ АКАДЕМІЧНУ ГРУПУ	Обов’язкова	Середня	
3	СТВОРИТИ ЗАНЯТТЯ	Обов’язкова	Висока	
4	ПРОВЕСТИ ЗАНЯТТЯ	Обов’язкова	Середня	
5	СТВОРИТИ ЗВІТ	Обов’язкова	Середня	

Таблиця 1.10

Специфікацію нефункціональних вимог

Іденти-фікатор вимог	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
Застосовність				
1	СТВОРИТИ ДИСЦИПЛІНУ	Обов’язкова	5	
2	СТВОРИТИ АКАДЕМІЧНУ ГРУПУ	Обов’язкова	20	
3	СТВОРИТИ ЗАНЯТТЯ	Обов’язкова	30	
4	ПРОВЕСТИ ЗАНЯТТЯ	Обов’язкова	15	
5	СТВОРИТИ ЗВІТ	Обов’яз кова	15	
Надійність				
1	СТВОРИТИ ДИСЦИПЛІНУ	Обов’язкова	99	
2	СТВОРИТИ АКАДЕМІЧНУ ГРУПУ	Обов’язкова	98	

3	СТВОРИТИ ЗАНЯТТЯ	Обов’язкова	98	
4	ПРОВЕСТИ ЗАНЯТТЯ	Обов’язкова	98	
5	СТВОРИТИ ЗВІТ	Обов’язкова	98	
Експлуатаційна придатність				
1	СТВОРИТИ ДИСЦИПЛІНУ	Обов’язкова	100	
2	СТВОРИТИ АКАДЕМІЧНУ ГРУПУ	Обов’язкова	100	
3	СТВОРИТИ ЗАНЯТТЯ	Обов’язкова	100	
4	ПРОВЕСТИ ЗАНЯТТЯ	Обов’язкова	100	
5	СТВОРИТИ ЗВІТ	Обов’язкова	100	

Висновки до першого розділу

Проведено огляд існуючих програмних засобів для вирішення функцій «Електронного журналу». Огляд виявив, що існуючі програмні засоби не вирішують відповідним чином функціональні вимоги. Базуючись на цьому було поставлено завдання для створення власного програмного засобу - «Електронного журналу», визначено варіанти використання мобільної програми та висунуто функціональні та нефункціональні вимого.

РОЗДІЛ ІІ
ПРОЕКТУВАННЯ ПРОГРАМИ «ЕЛЕКТРОННИЙ ЖУРНАЛ» ДЛЯ ОС ANDROID»

2.1. Розроблення архітектури програмної системи

При проектуванні архітектури програмного забезпечення необхідно дотримуватися наступних основних принципів, які допоможуть створити архітектуру ефективну та звезти до мінімуму витрати і вимоги до технічного обслуговування, а також сприяє зручність використання:

- поділ задачі - поділити додаток на певні модулі, з невеликим перекриттям в функціональності. Важливим фактором є мінімізація точок взаємодії для досягнення високої згуртованості і низького зчеплення. Тим не менше, поділ функціональних можливостей в неправильних межах може привести до сильного зв'язку і складності між функціями , навіть якщо функціональні можливості, що містяться в функції не призводить до істотного перекриття;
- одиничний принцип відповідальності - кожен компонент або модуль повинен відповідати тільки за одну конкретну функцію або функціональність, або агрегації згуртованої функціональності.
- принцип найменшого знання - компонент або об'єкт не повинен знати про внутрішні деталей інших компонентів або об'єктів.
- принцип неповторюваності - необхідність вказати намір в одному місці. Наприклад, з точки зору розробки додатків, конкретні функціональні можливості повинні бути реалізовані тільки в одному компоненті; функціональні можливості не повинні бути продубльовані в будь-якому іншому компоненті;
- принцип зведення до мінімуму авансової розробки - розробляти тільки необхідні функціональні можливості. У деяких випадках може знадобитися авансовий комплексне проектування і тестування, якщо вартість

розробки або провал в розробці дуже високий. В інших випадках, особливо для швидкої розробки, необхідно уникати великої авансової розробки. Якщо ваші вимоги програми не ясні, або якщо є можливість розвитку з часом, слід уникати великих передчасних конструкторських робіт.

При розробці програми або системи, метою створення архітектури програмного забезпечення є зведення до мінімуму складності шляхом відділення структури в різних проблемних областях. Наприклад, призначений для користувача інтерфейс, обробка бізнес логіки, і доступ до даних все це представлення різних проблемних областей. У кожній області, компоненти дизайну повинні зосередитися на конкретній області, і не слід змішувати код з інших областей. Наприклад, компоненти обробки не повинен містити код, який безпосередньо отримує доступ до джерела даних, але замість цього слід використовувати або бізнес-компоненти або компоненти доступу до даних для отримання даних.

Тим де необхідно зробити визначення вартості та інвестиції для додатка не є важливою можливо, буде потрібно спростити структуру, щоб дозволити наприклад, прив'язку даних до інтерфейсу користувача для результуючого проекту. Загалом, необхідно розглядати функціональні можливості з комерційної точки зору.

Провівши ієрархічний аналіз вимог до програмного забезпечення, було виділено пакети, що відповідають підсистемам, які будуть використовуватися для представлення користувацького інтерфейсу, зв’язку з базою даних, реалізації класів предметної області.

На рисунку 2.1 показано діаграму, на якій зображено архітектуру програмного засобу для планування справ.

Рисунок-2.1. Архітектура програмної системи «Електронний журнал»

Для виявлення акторів, шукаємо безпосереднього користувача журналу. Проаналізувавши вимоги до програмного забезпечення, встановлюємо, що єдиним актором є «Викладач». Отже, єдиним актором у нашій функціональній моделі буде користувач(“teacher”).

Виходячи з функції системи, визначених у попередньому розділі, встановлюємо наступні функціональні блоки системи, які співпадають з варіантами використання:

СТВОРИТИ ДИСЦИПЛІНУ- CREATE DISCIPLINE

СТВОРИТИ АКАДЕМІЧНУ ГРУПУ - CREATE ACADEMIC GROUPS

СТВОРИТИ ЗАНЯТТЯ - CREATE EMPLOYMENT

ПРОВЕСТИ ЗАНЯТТЯ - CONDUCT LESSON

СТВОРИТИ ЗВІТ - CREATE REPORT

Для зв’язування в єдину систему створюємо додатковий модуль, який буде керувати функціональними модулями

МОДУЛЬ КЕРУВАННЯ - CONTROL MODULE.

Деталізуємо обрані варіантів використання і модулі системи з використанням діаграми модулів системи рисунок 2.2.

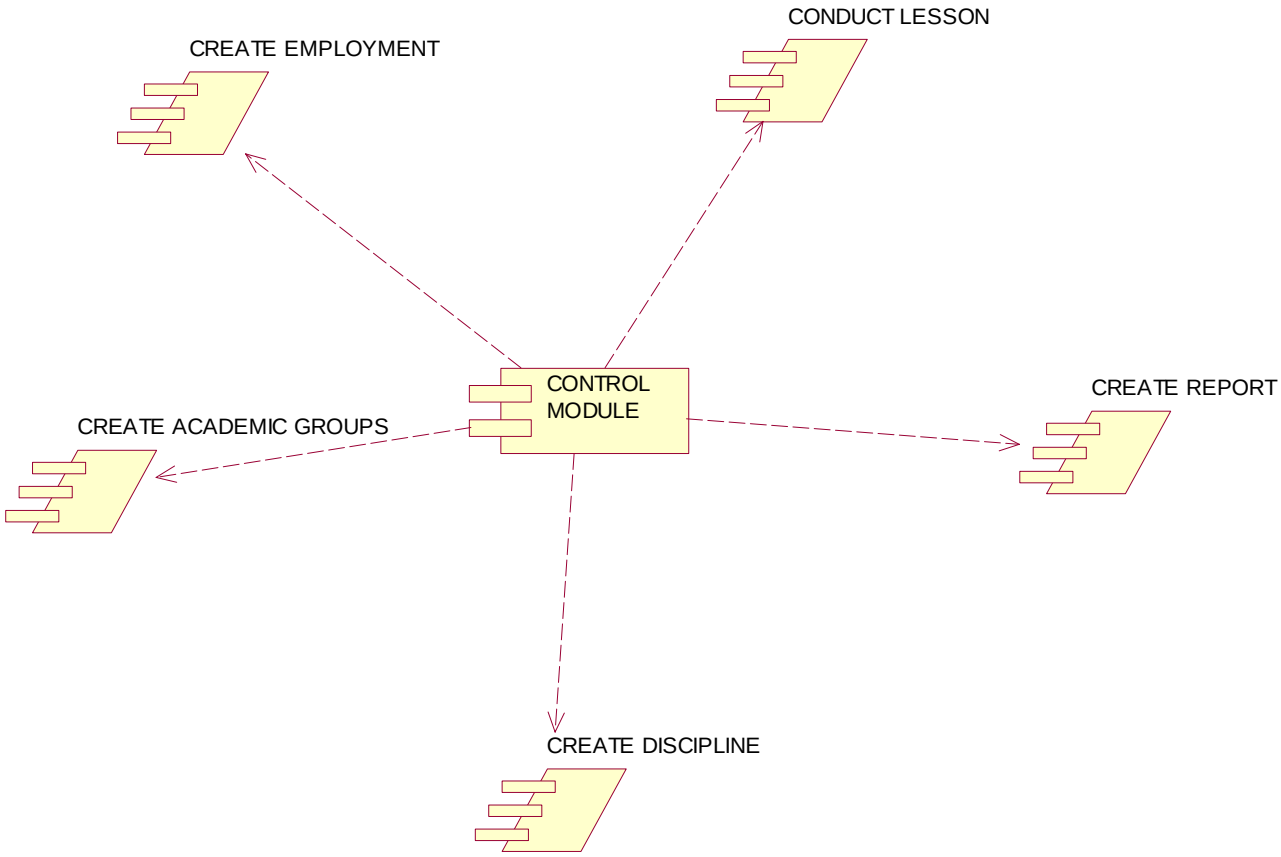


Рисунок-2.2 Структура схеми програмної системи

Для реалізації вибраної системи вибираємо об’єктно-орієнтований підхід до проектування та програмування. Тобто наступним кроком є визначення діаграми класів (рисунок 2.3)

Виходимо з того принципу, що забезпечення однієї функції програмного засобу буде реалізовано в одному модулі.

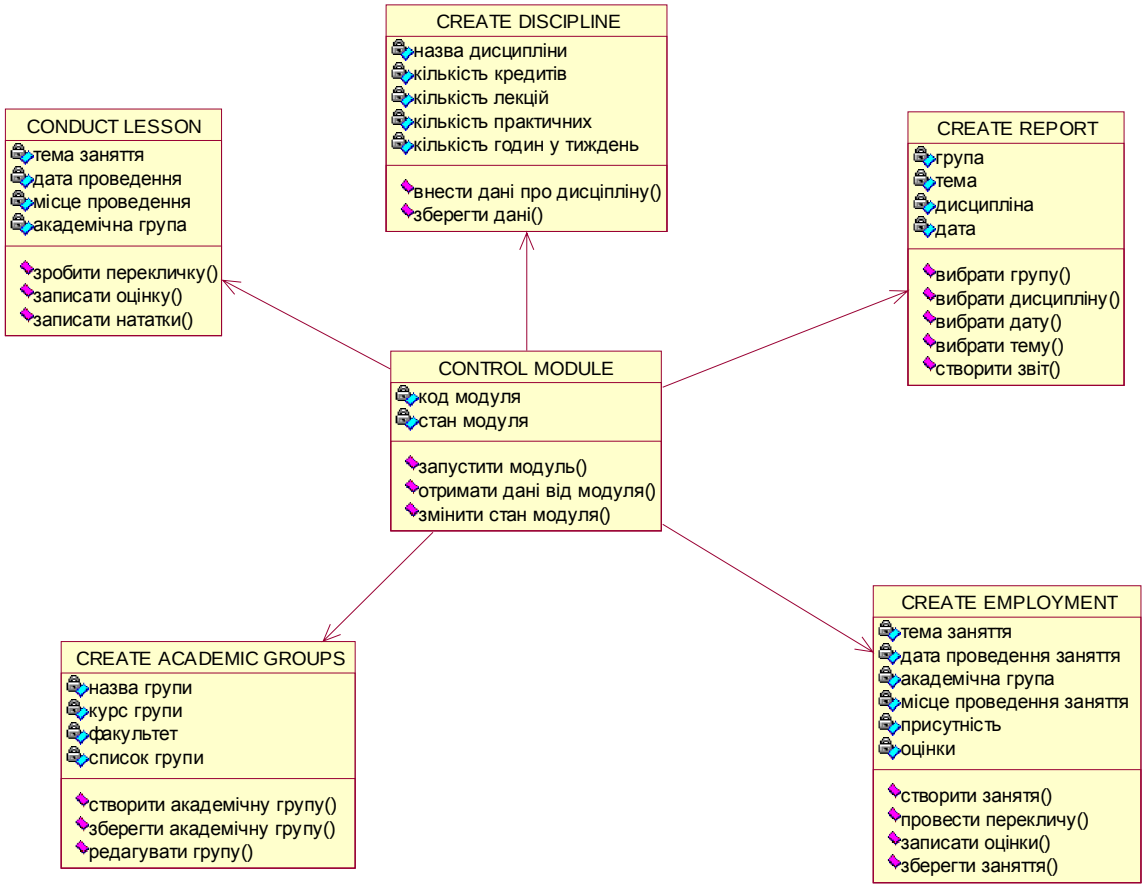


Рисунок-2.3 Діаграма класів програмної системи «Електронний журнал»

2.2. Проектування структури бази даних

Процес проектування складається з наступних етапів:

Визначення мети створення бази даних, це допомагає підготувати вирішення решти кроків.

Знайти і організовувати необхідну інформацію, на цьому етапі необхідно зібрати всі види інформації, яку необхідно записати в базі даних.

Розділити цю інформацію в таблицях. На цьому етапі необхідно розподілити ваші інформаційні елементи в великі суб'єкти тобто кожен предмет стає таблицею.

Вставте інформаційні елементи в стовпці, тобто необхідно вирішити, яку інформацію необхідно зберігати в кожній таблиці. Кожен елемент стає полем, і відображається у вигляді стовпчика в таблиці.

Вказати первинні ключі. Необхідно вибрати первинний ключ кожної таблиці. Первинний ключ являє собою стовпець, який використовується для однозначної ідентифікації кожного рядка.

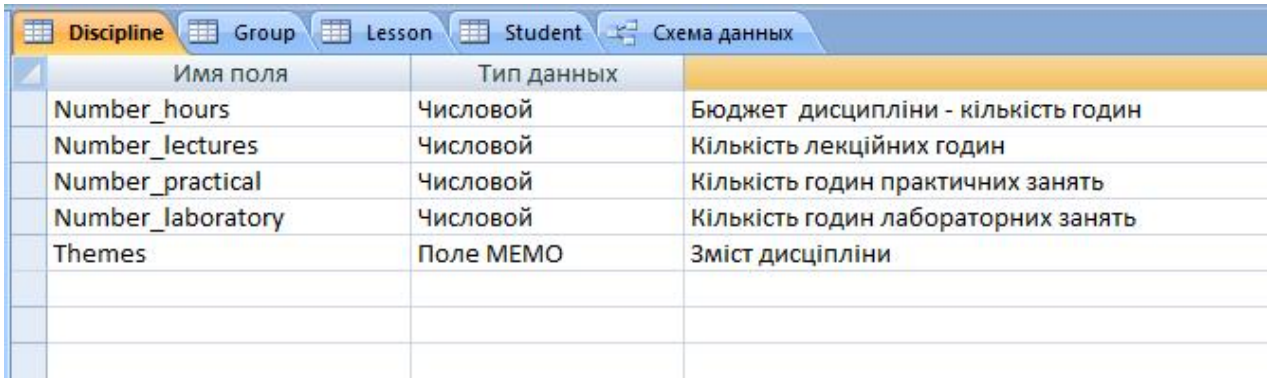
Встановлення відношень між таблицями, тут необхідно виявити в кожній таблиці певні поля і вирішити, як дані в одній таблиці пов'язана з даними в інших таблицях. Додавання полів у таблиці або створити нові таблиці для з'ясування відносин, у міру необхідності.

Уточнити конструкцію, тут необхідно проаналізувати існуючий дизайн на наявність помилок. Створити таблиці і додати кілька записів вибіркових даних. Подивитися, якщо можна отримати результати, які будуть адекватними до вирішеної задачі. По можливості внести зміни в конструкції таблиць даних, у міру необхідності.

Застосувати правила нормалізації, тут необхідно подивитися, чи правильно побудовані таблиці, якщо ні то необхідно внести зміни в таблиці, в міру необхідності.

Проектуємо архітектуру баз даних (БД). Всі архітектури представляються нотацією UML і при потребі, доповнюються текстовими описами.

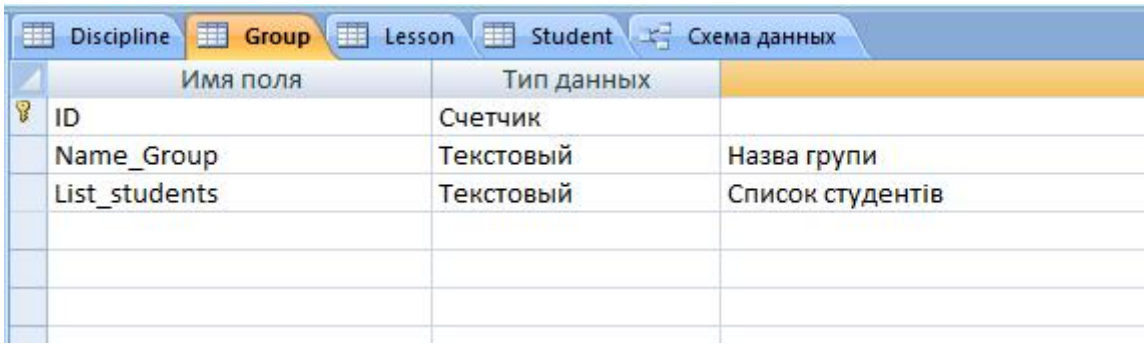
В ході проектування архітектури БД, було обрано для її відтворення реляційну модель даних та виділено наступні відношення, що відображені на наступних рисунках рисунки 2.4-2.7.



Discipline | Group | Lesson | Student | Схема данных

Имя поля	Тип данных	
Number_hours	Числовой	Бюджет дисципліни - кількість годин
Number_lectures	Числовой	Кількість лекційних годин
Number_practical	Числовой	Кількість годин практичних занять
Number_laboratory	Числовой	Кількість годин лабораторних занять
Themes	Поле MEMO	Зміст дисципліни

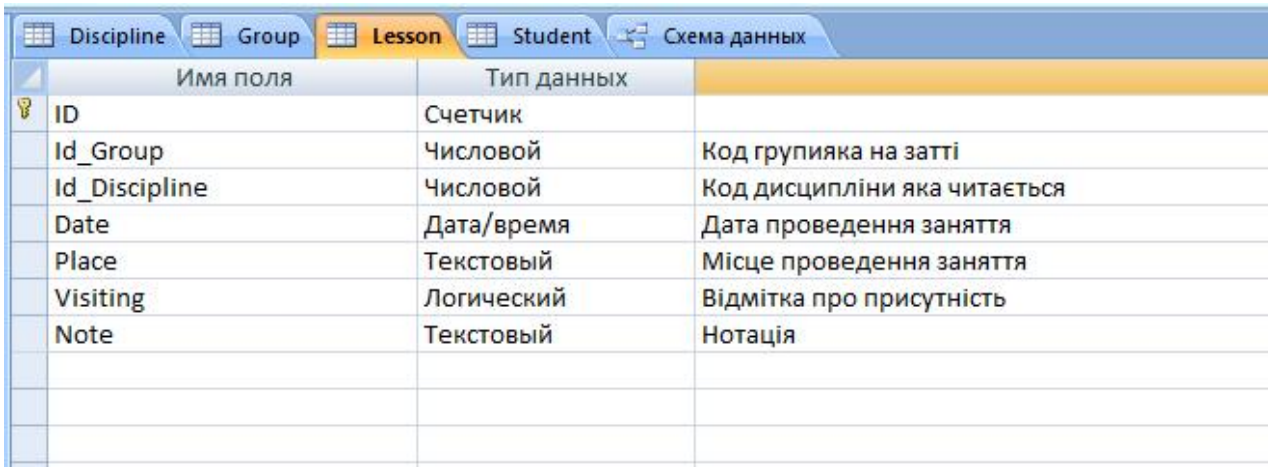
Рисунок 2.4- Таблиця бази даних «Discipline»



Discipline | Group | Lesson | Student | Схема данных

Имя поля	Тип данных	
ID	Счетчик	
Name_Group	Текстовый	Назва групи
List_students	Текстовый	Список студентів

Рисунок 2.5- Таблиця бази даних «Group»



Discipline | Group | Lesson | Student | Схема данных

Имя поля	Тип данных	
ID	Счетчик	
Id_Group	Числовой	Код групияка на затті
Id_Discipline	Числовой	Код дисципліни яка читається
Date	Дата/время	Дата проведення заняття
Place	Текстовый	Місце проведення заняття
Visiting	Логический	Відмітка про присутність
Note	Текстовый	Нотація

Рисунок 2.6- Таблиця бази даних «Lesson»

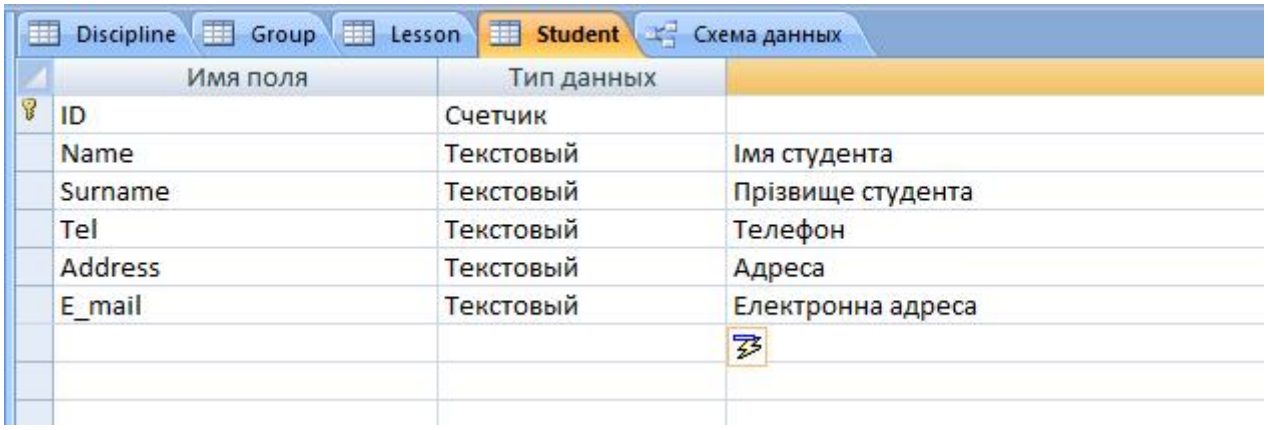


Рисунок 2.7- Таблиця бази даних «Student»

Діаграма реляційної моделі даних зображена на рисунку 2.8.

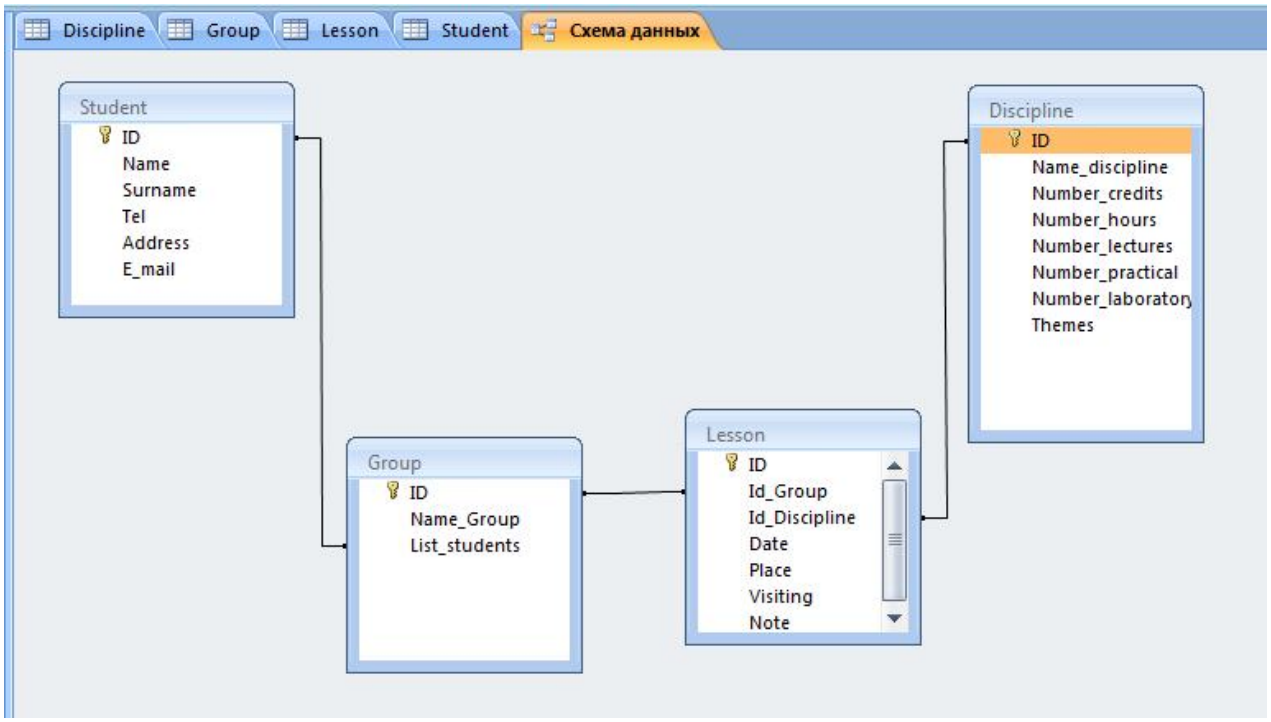


Рисунок 2.8- ER діаграма бази даних

Висновки до другого розділу

В другому розділі запроектовано програму «Електронний журнал», також запроектовано до неї база даних. Проектування здійснювалося з використанням об’єктно-орієнтованого підходу до проектування, а також використовувалася UML діаграми для нотацій.

РОЗДІЛ III
РОЗДІЛ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ «ЕЛЕКТРОННИЙ ЖУРНАЛ»

3.1. Програмна реалізація проекту

3.1.1. Вибір засобу створення системи

Велику популярність серед мобільних пристроїв здобули пристрої на базі операційної системи «Android», тому засобом створення програмного забезпечення буде Android Studio.

Android Studio - це інтегроване середовище розробки (Integrated Development Environment IDE) для роботи з платформою Android , анонсована 16 травня 2013 року на конференції Google I / O .

IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse .

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains , офіційне засіб розробки Android додатків. Дане середовище розробки доступна для Windows , OS X і Linux.

Нові функції з'являються з кожною новою версією Android Studio. На даний момент доступні наступні функції:

- Розширений редактор макетів: WYSIWYG , здатність працювати з UI компонентами за допомогою Drag-and-Drop , функція попереднього макета на декількох конфігураціях екрану.
- Збірка програм, заснована на Gradle .
- Різні види збірок і генерація кількох .apk файлів
- рефакторинг коду

- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій і інше.
- Вбудований ProGuard і утиліта для підписки додатків.
- Шаблони основних макетів і компонентів Android.
- Підтримка розробки додатків для Android Wear і Android TV [6] .
- Вбудована підтримка Google Cloud Platform, яка включає в себе інтеграцію з сервісами Google Cloud Messaging і App Engine.
- Android Studio 2.1 підтримує Android N Preview SDK, а це означає, що розробники зможуть почати роботу зі створення програми для нової програмної платформи.
- Нова версія Android Studio 2.1 здатна працювати з оновленим компілятором Jack, а також отримала покращену підтримку Java 8 і вдосконалену функцію Instant Run [7] .

Системні вимоги:

Windows- версія OS Microsoft Windows 10/8/7 / Vista / 2003 (32 або 64-bit).

OS X- Mac® OS X® 10.8.5 або вище, до 10.9 (Mavericks).

Linux- GNOME або KDE.

Оперативна пам'ять- 2 ГБ (мінімум), 4 ГБ (рекомендується).

Вільне місце на диску- 400 МБ.

Вільне місце для Android SDK- 1 ГБ (мінімум).

Версія JDK- Java Development Kit 7

Роздільна здатність екрану - 1280 x 800 (мінімум).

На рисунку 3.1 наведено зовнішній вигляд середовища розробки Android Studio.

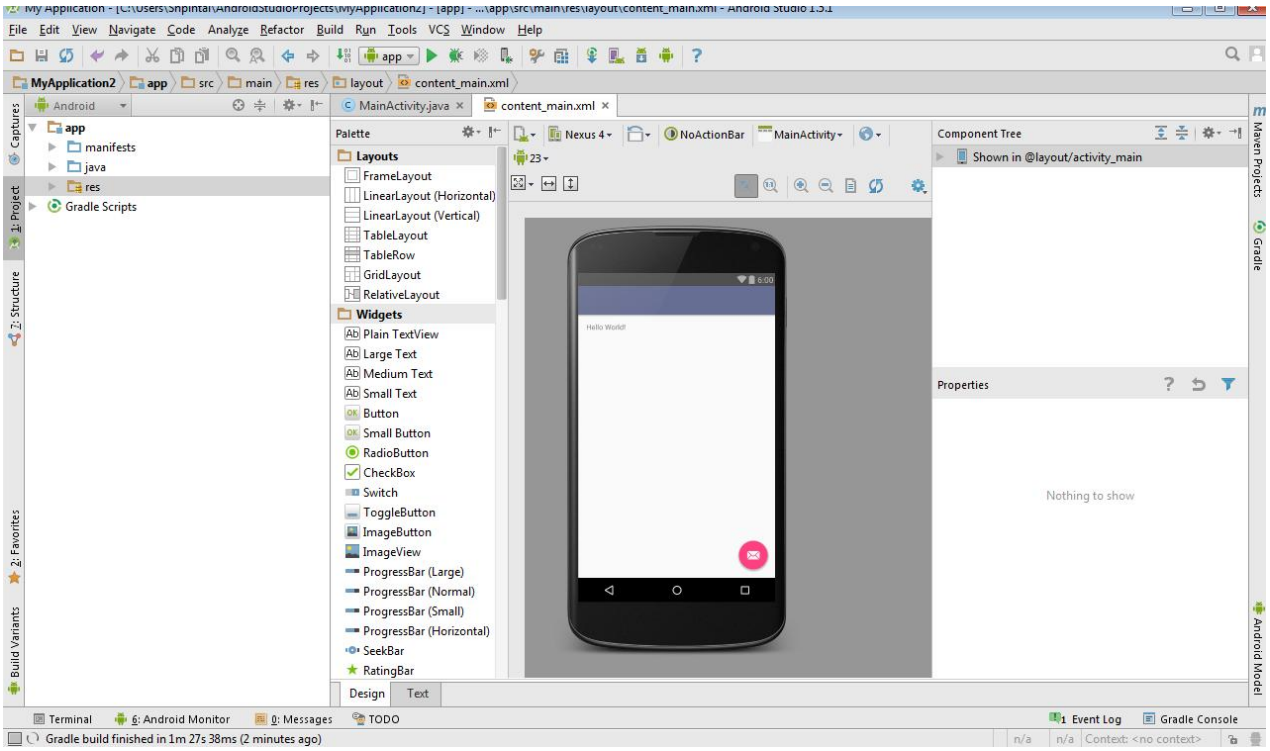


Рисунок 3.1- Зовнішній вигляд середовища розробки Android Studio

Для створення нового проекту необхідно виконати наступні дії:

Запускаємо Android Studio і вибираємо File | New | New Project. З'явиться діалогове вікно майстра .

Поле Application name: - зрозуміле ім'я для програми, яка буде відображатися в заголовку програми. За замовчуванням вже є My Application .

Поле Company Domain: служить для вказівки вашого сайту. За замовчуванням там може з'явитися ваше ім'я як користувача комп'ютера. Якщо сайт у вас є, то можете ввести його адресу, або придумайте яку-небудь назву. Введене ім'я запам'ятовується і буде автоматично підставлятися в наступних нових проектах.

Поле Package name: формує спеціальний Java-пакет на основі вашого імені з попереднього поля. У Java використовується перевернутий варіант для найменування пакетів, тому спочатку йде домен, а потім вже назва сайту. Пакет служить для унікальної ідентифікації вашої програми, коли ви будете його поширювати. Кнопка Edit дозволяє відредагувати підготовлений варіант.

Третє поле Project location: дозволяє вибрати місце на диску для створюваного проекту рисунок 3.2.

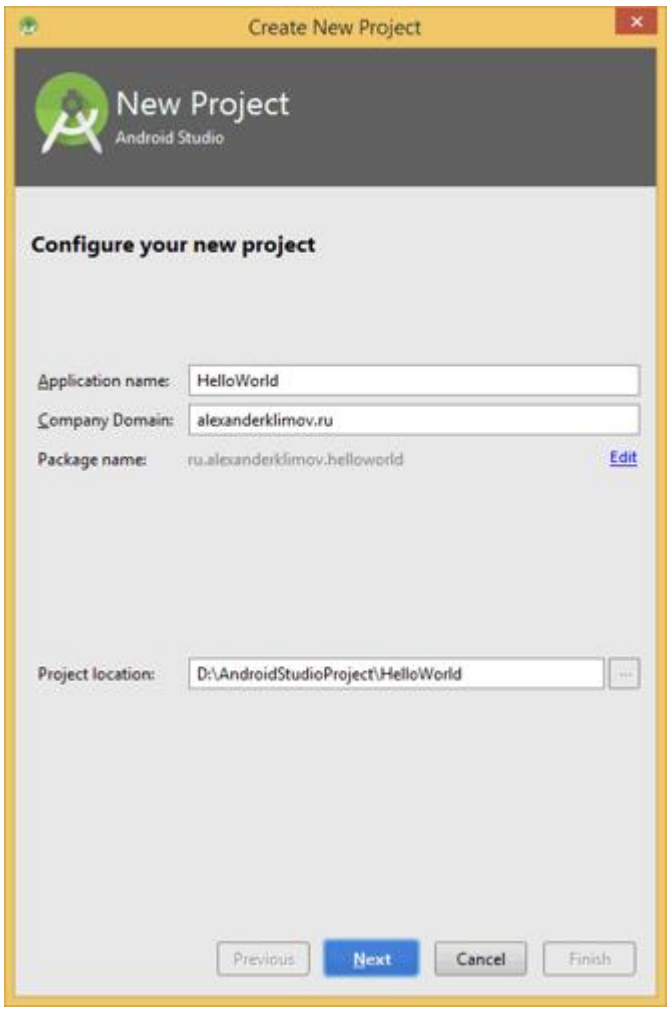


Рисунок 3.2- Діалогове вікно майстра розробки Android Studio

Натискаємо на кнопку Next і переходимо до наступного вікна рисунок 3.3. Тут ми вибираємо типи пристроїв, під які будемо розробляти свій додаток. Вибираємо для смартфонів і планшетів, тобто залишаємо прапорець у першого пункту.

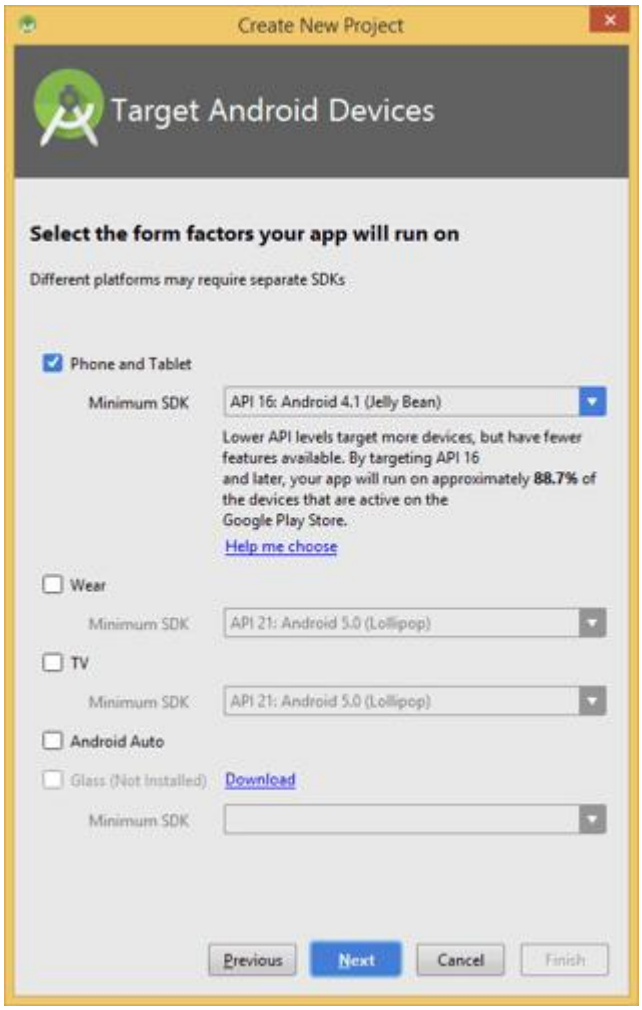


Рисунок 3.3- Діалогове вікно вибору типу пристроїв Android Studio

Крім вибору типу пристроїв, вибраємо мінімальну версію системи, під якою буде працювати додаток. Вибераємо свій варіант. На даний момент Гугл підтримує версії, починаючи з API 7, випускаючи спеціальні бібліотеки сумісності для старих пристроїв.

Далі знову натискаємо кнопку Next . У вікні, що появилося рисунок 3.4 слід вибрати зовнішній вигляд екрану програми.

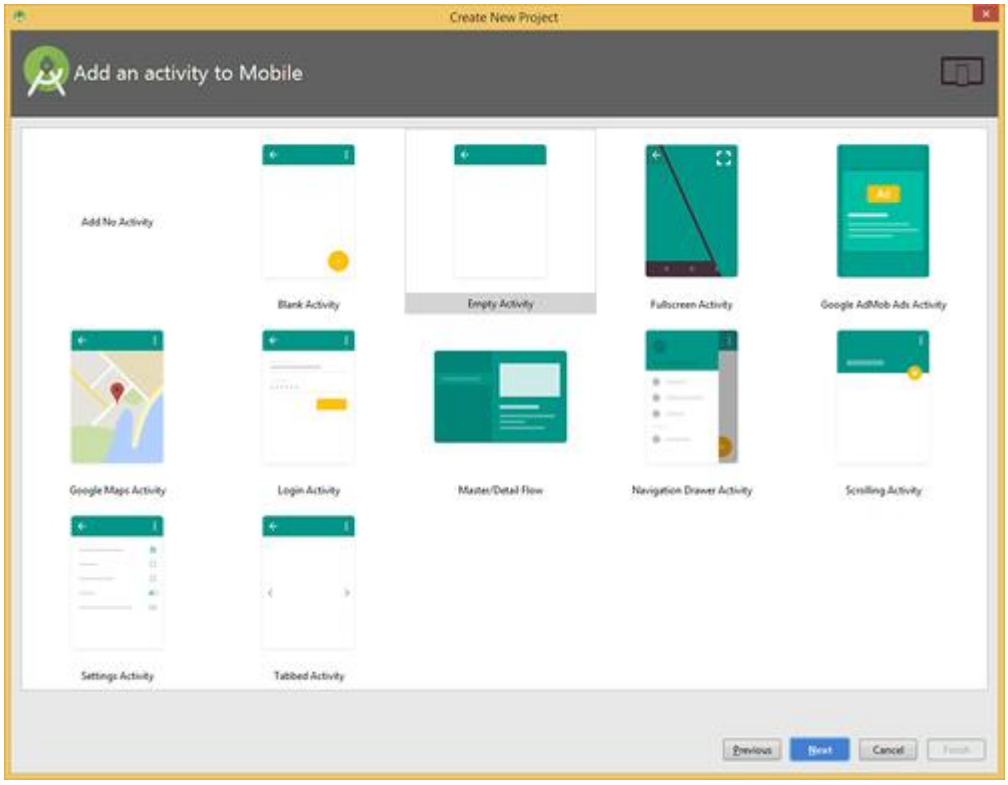


Рисунок 3.4- Діалогове вікно вибору зовнішнього вигляду екрану програми Android Studio

Запропоновані шаблони дозволяють заощадити час на написання стандартного коду для типових ситуацій.

Пару років назад був тільки один шаблон. Список шаблонів постійно поповнюється і тепер їх більше. Перерахую частину з них.

- Blank Activity
- Empty Activity
- Fullscreen Activity
- Google Maps Activity
- Google AdMod Ads Activity
- Login Activity
- Master / Detail Flow
- Navigation Drawer Activity (новинка)
- Scrolling Activity (новинка)

Шаблон Empty Activity призначений для звичайних телефонів. На зображенні над назвою видо приблизний вигляд програми з використанням даної заготовки.

Шаблон Master / Detail Flow призначений для планшетів з реалізацією двохпанельний режиму.

Шаблон Fullscreen Activity можна використовувати для ігор, коли потрібно додатковий простір без зайвих деталей.

Інші шаблони потрібні для створення додатків з гуглокартамі або сервісами Google Play.

Отже, вибраємо варіант Empty Activity і переходимо до наступного вікна рисунок 3.5.

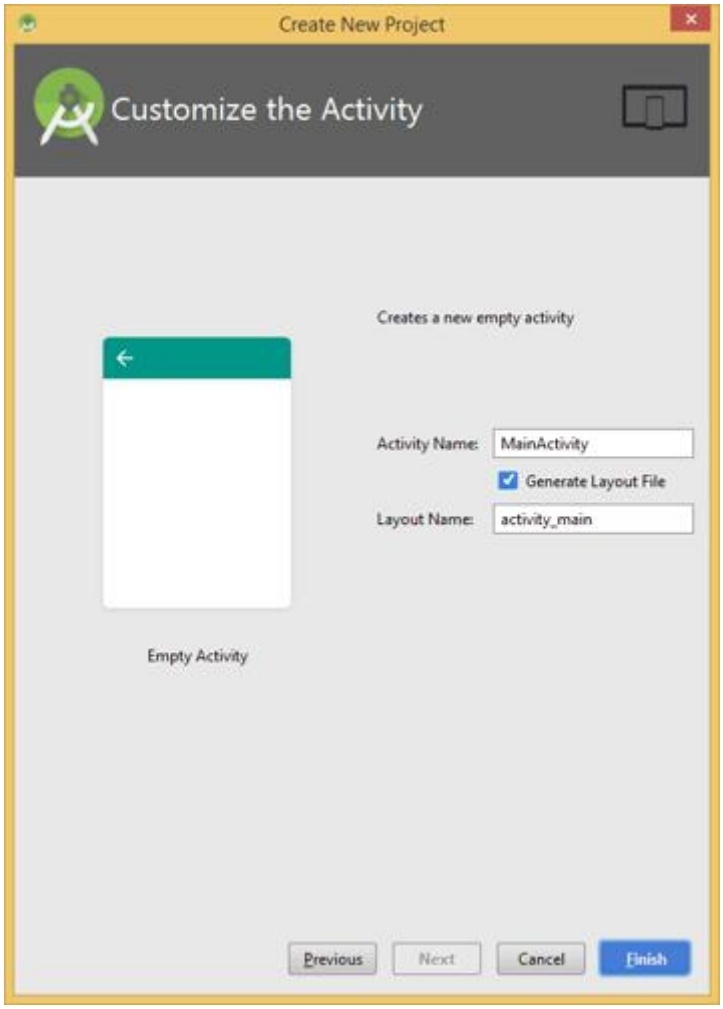


Рисунок 3.5- Діалогове вікно завершення створення проекту

Натискаємо кнопку Finish і дивимося, що далі буде.

А далі студія формує проект і створює необхідну структуру з різних файлів і папок.

3.1.2. Вибір мови програмування

В якості мови програмування для Android використовується Java. Для створення призначеного для користувача інтерфейсу використовується XML.

Java - об'єктно-орієнтована мова програмування , розроблена компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-який віртуальної Java-машині незалежно від комп'ютерної архітектури. Дата офіційного випуску - 23 травня 1995 року.

XML (EXtensible Markup Language - розширювана мова розмітки). Рекомендований Консорціумом Всесвітньої павутини (W3C). Специфікація XML описує XML-документи і частково описує поведінку XML-процесорів (програм, які читають XML-документи і забезпечують доступ до їх вмісту). XML розроблявся як мова з простим формальним синтаксисом, зручний для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті. Мова називається розширюваною, оскільки нею не фіксується розмітку, яка використовується в документах: розробник може створити розмітку відповідно до потреб конкретної області, будучи обмеженим лише синтаксичними правилами мови. Поєднання простого формального синтаксису, зручності для людини, розширюваності, а також базування на кодуваннях Юнікод для подання змісту документів призвело до широкого використання як власне XML, так і множини похідних спеціалізованих мов на базі XML в найрізноманітніших програмних засобах.

3.1.3. Організація інтерфейсу з користувачем

У першому розділі було запроектовано інтерфейси програмного засобу з користувачем дивись рисунки 1.6-1.10) реалізовуємо відповідні форми у вибраному середовищі програмування.

На рисунку 3.5 діалогове вікно дисципліни, тобто внесення в систему запису про кількісні характеристики дисципліни та її зміст

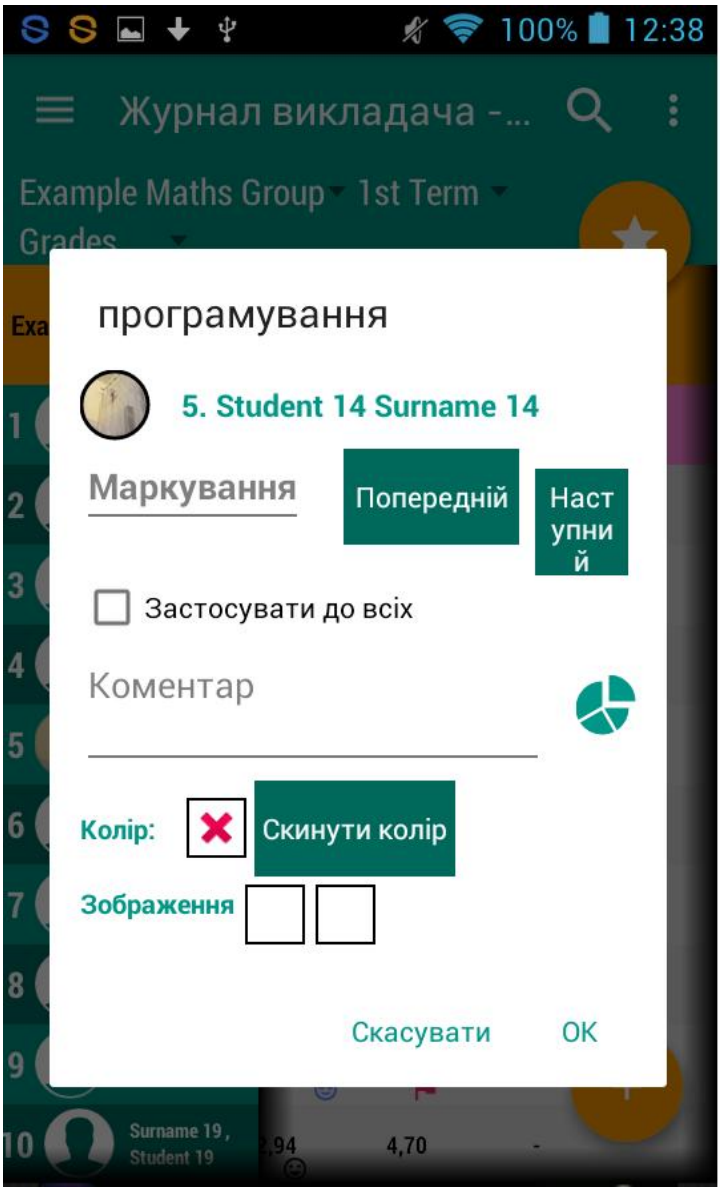


Рисунок 3.5- Діалогове вікно для реалізації варіанту використання «СТВОРИТИ ДИСЦИПЛІНУ»

На рисунку 3.6 діалогове вікно для внесення інформації про студентів та занести інформацію про академічну групу

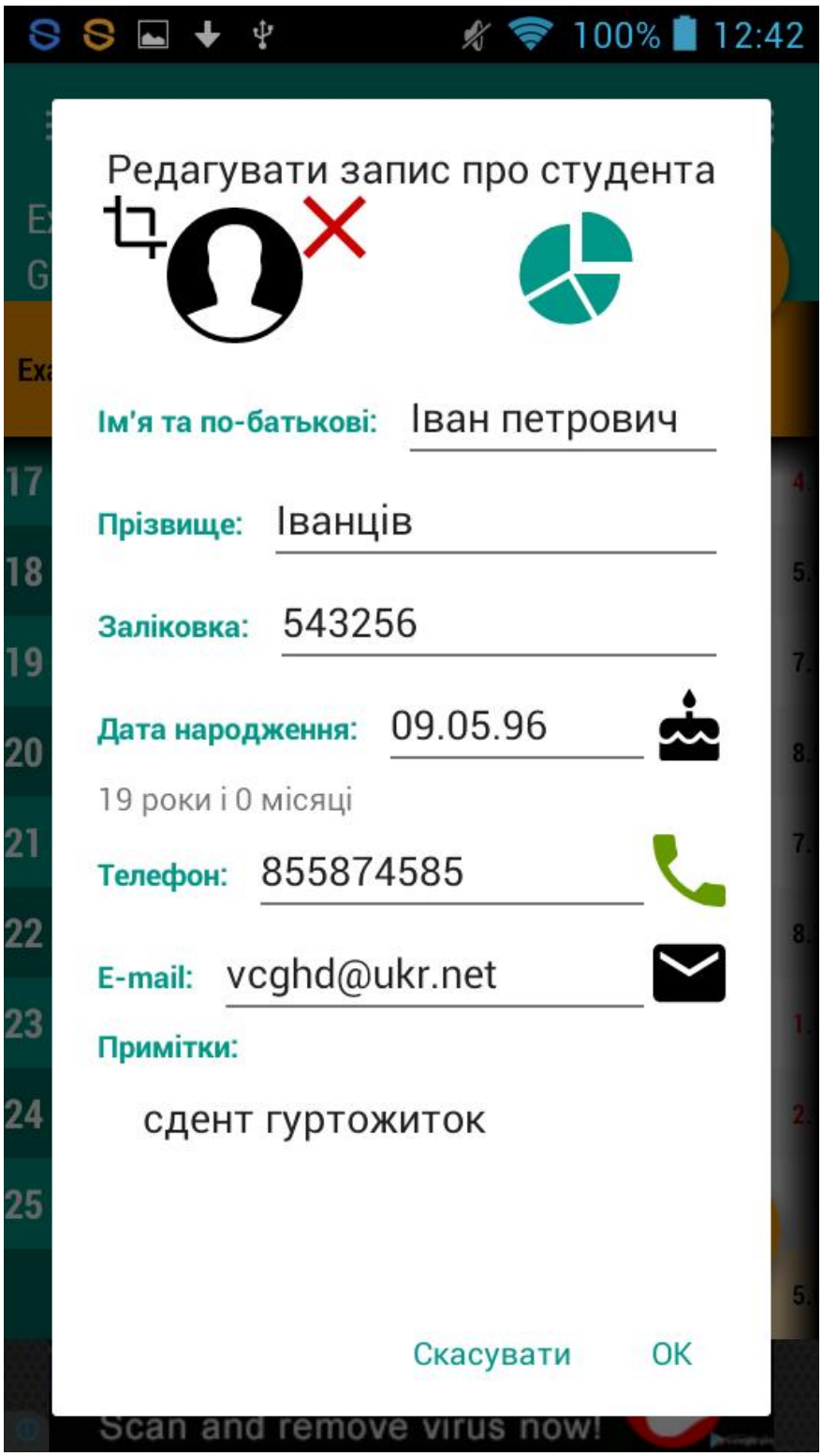


Рисунок 3.6- Діалогове вікно для реалізації варіанту використання «СТВОРИТИ АКАДЕМІЧНУ ГРУПУ»

На рисунку 3.7 діалогове вікно створення «заняття», тобто описати заняття в конкретному місці, за конкретною темою та з конкретно югрупою

Рисунок 3.7- Діалогове вікно для реалізації варіанту використання «СТВОРИТИ ЗАНЯТТЯ»

На рисунку 3.8 показано діалогове вікно для внесення інформації про активність студентів на занятті

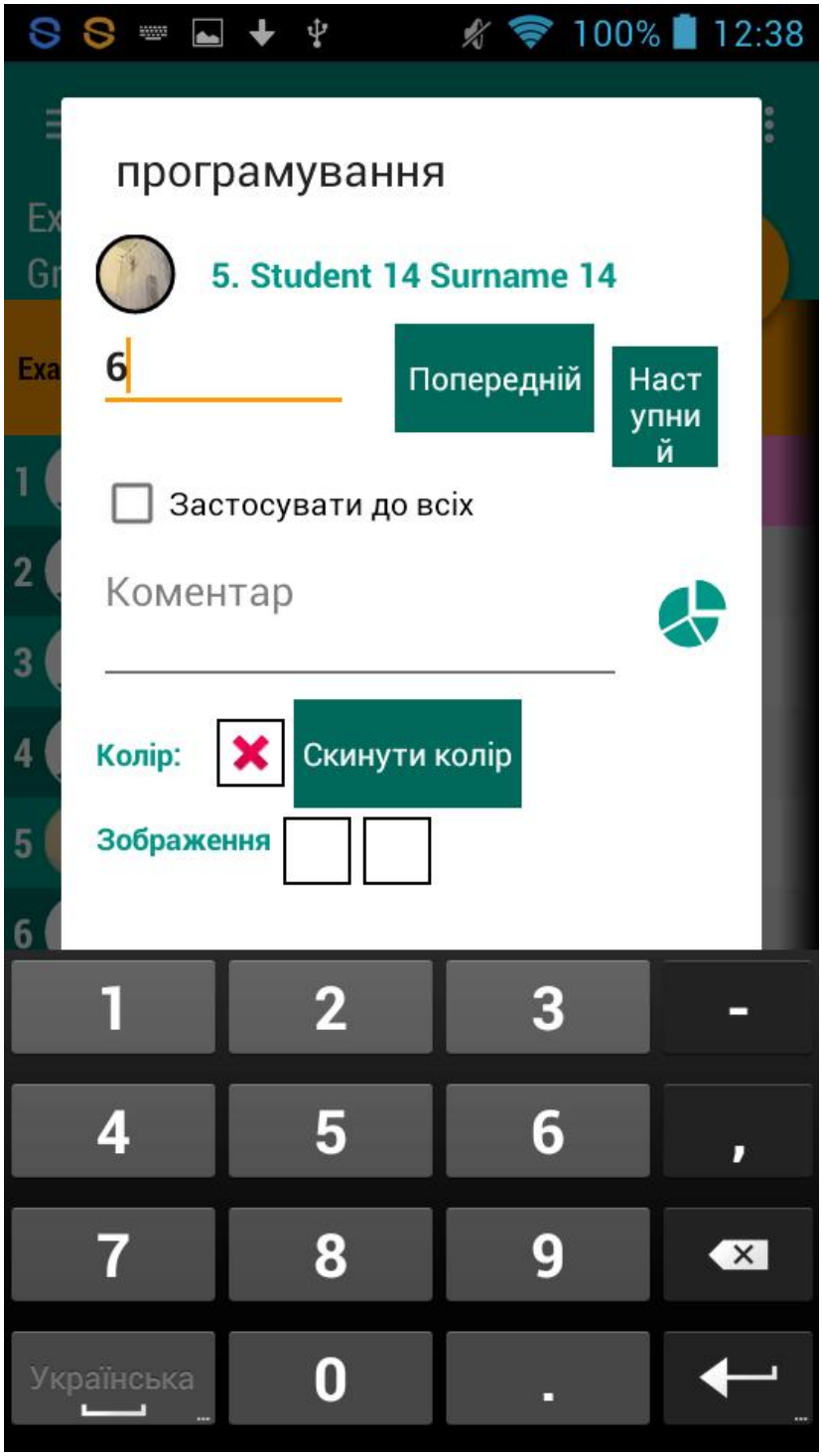


Рисунок 3.8- Діалогове вікно для реалізації варіанту використання «ПРОВЕСТИ ЗАНЯТТЯ»

На рисунку 3.9 показано діалогове вікно створення звіту

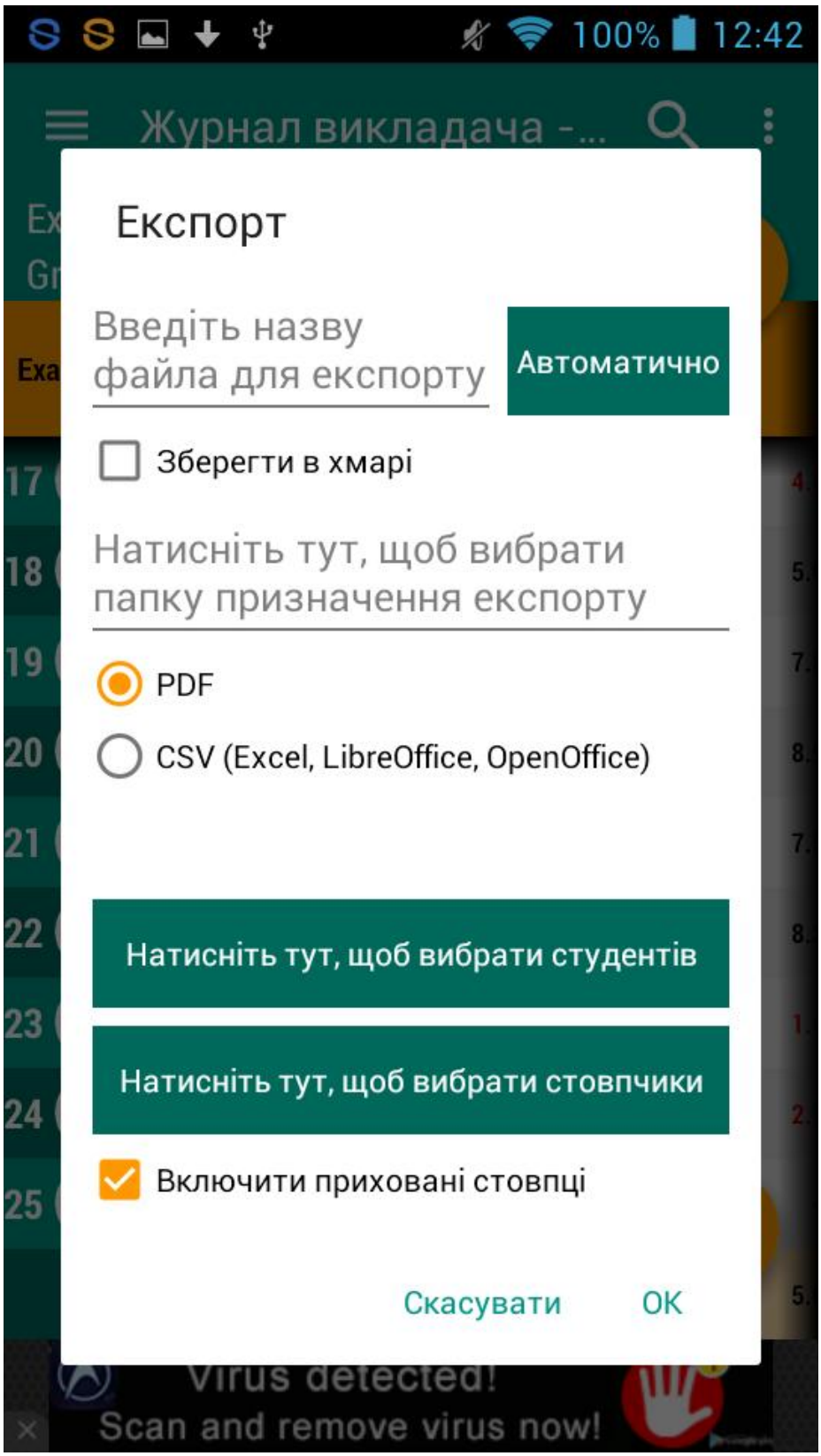


Рисунок 3.9- Діалогове вікно для реалізації варіанту використання «СТВОРИТИ ЗВІТ»

3.2. . Програмна реалізація бази даних

3.2.1. Вибір бази даних SQLite для додатку андроїд

Android використовує для роботи з базами даних відому бібліотеку SQLite. SQLite вона зарекомендувала себе як надзвичайно надійної системи баз даних, яка використовується в багатьох побутових електронних пристроях і програмах, включаючи деякі MP3-програвачі, iPhone, iPod Touch, Mozilla Firefox і ін.

За допомогою SQLite можна створювати незалежні реляційні бази даних. Android зберігає бази даних в каталозі / data / data / <ім'я_вашого_пакета> / databases на емуляторі, на пристрої шлях може відрізнятися. За замовчуванням всі бази даних закриті, доступ до них можуть отримати лише ті додатки, які їх створили.

Кожна база даних складається з двох файлів. Ім'я першого файлу бази даних відповідає імені бази даних. Це основний файл баз даних SQLite, в ньому зберігаються всі дані. Другий файл - файл журналу. Його ім'я складається з імені бази даних і суфікса "-journal". У файлі журналу зберігається інформація про всі зміни, внесені в базу даних. Якщо в роботі з даними виникне проблема, Android використовує дані журналу для скасування (або відкату) останніх змін.

Перш за все потрібно запам'ятати, що в Android вже є готовий клас SQLiteOpenHelper , від якого потрібно успадковуватися.

3.2.2. Створення бази даних

Для роботи з базою даних необхідно створити новий проект як зазвичай. Для взаємодії з базою даних в місце запуску модуля необхідно вести результат зчитування даних.

```
<LinearLayout xmlns: android = "http://schemas.android.com/apk/res/android"
    xmlns: tools = "http://schemas.android.com/tools"
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"
    android: orientation = "vertical"
    android: paddingBottom = "@ dimen / activity_vertical_margin"
    android: paddingLeft = "@ dimen / activity_horizontal_margin"
    android: paddingRight = "@ dimen / activity_horizontal_margin"
    android: paddingTop = "@ dimen / activity_vertical_margin"
    tools: context = ". MainActivity">

    <Button
        android: id = "@ + id / button"
        android: layout_width = "match_parent"
        android: layout_height = "wrap_content"
        android: onClick = "onClick"
        android: text = "Запит до бази даних" />

    <TextView
        android: id = "@ + id / textView"
        android: layout_width = "match_parent"
        android: layout_height = "wrap_content"
        android: textAppearance = "? android: attr / textAppearanceLarge" />
```

```
</ LinearLayout>
```

При роботі з базою даних необхідно створити окремий клас. Зазвичай, в проекті у нас вже є один клас MainActivity , який відноситься до активності. А тепер буде два класи. Тільки другий клас буде успадковуватися немає від класу Activity , а від класу SQLiteOpenHelper .

Створений клас буде працювати з базою даних - додавати, вибирати, видаляти та інші операції.

Для створення лацаємо правою кнопкою миші на імені пакета в лівій частині студії і вибираємо в меню New | Java Class і в діалоговому вікні вибираємо ім'я для нового класу, наприклад, DatabaseHelper . Слово Helper зазвичай використовують, щоб показати, що клас є обгорткою (допоміжним класом) якогось абстрактного класу.

У нас з'явиться заготовка. Наслідуючи від SQLiteOpenHelper . Студія запропонує створити два обов'язкових методуonCreate () і onUpgrade ().

```
package ru.alexanderklimov.database;


import android.content.Context;

import android.database.DatabaseErrorHandler;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

   public DatabaseHelper (Context context, String name,
SQLiteDatabase.CursorFactory factory,

                          int version) {

      super (context, name, factory, version);

   }
```

```
public DatabaseHelper (Context context, String name, SQLiteDatabase.CursorFactory factory,
                       int version, DatabaseErrorHandler errorHandler) {
    super (context, name, factory, version, errorHandler);
  }
  @Override
  public void onCreate (SQLiteDatabase db) {
 }
  @Override
  public void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion) {
  }}
```

Ту, константа DATABASE_NAME відповідає за ім'я файлу, в якому буде зберігатися база даних програми. Можна придумати будь-яке ім'я.

Друга константа DATABASE_VERSION відповідає за номер версії бази. Принцип її роботи схожий з номером версій самого додатка. Коли ми бачимо, що вийшла нова версія Chrome, то розуміємо, що час оновлюватися. Аналогічно надходить і сам додаток, коли помічає, що номер версії бази став іншим. Як тільки програма помітила оновлення номера бази, вона запускає метод onUpgrade () , який у нас сформувався автоматично. У цьому методі необхідно розмістити код, який повинен спрацювати при оновленні бази.

Метод onCreate () тут створює базу даних з необхідними даними для роботи.

Робота з базою даних

Переходимо до головної активності і напишемо наступний код:

```
private DatabaseHelper mDatabaseHelper;
```

```
public void onClick (View v) {
        mDatabaseHelper = new DatabaseHelper (this, "mydatabase.db", null, 1);
        SQLiteDatabase sdb;
        sdb = mDatabaseHelper.getReadableDatabase ();
}
```

Ми використовували другий конструктор, але можна було написати і new DatabaseHelper (this) з використанням спрощеного конструктора. Таким чином, ви можете перенести логіку створення бази в окремий клас.

Запустивши проект натисканням на кнопку, буде створено на пристрої базу даних. Переконатися в цьому можна, якщо запустити Android Device Monitor . Вибераємо вкладку File Explorer і знаходимо свій додаток. З'явилася папка data / data / Имя_Пакета / databases з файлом mydatabase.db . Метод getReadableDatabase створює або відкриває базу даних.

Висновки до третього розділу

В третьому розділі описано процес кодування (створення програмного коду) програмного засобу «Електронний журнал» з використанням мови програмування Java. В цьому розділі також показані діалогові вікна для вирішення конкретних необхідних задач.

РОЗДІЛ IV
РОЗДІЛ ТЕСТУВАННЯ ТА ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ

4.1. Тестування

Тестування модулів (або блоків) являє собою процес тестування окремих підпрограм або процедур програми. Тут мається на увазі, що, перш ніж починати тестування програми в цілому, слід протестувати окремі невеликі модулі, що утворюють цю програму. Такий підхід мотивується трьома причинами.

По-перше, з'являється можливість управляти комбінаторикою тестування, оскільки спочатку увага концентрується на невеликих модулях програми.

По-друге, полегшується завдання налагодження програми, тобто виявлення місця помилки і виправлення тексту програми.

По-третє, допускається паралелізм, що дозволяє одночасно тестувати кілька модулів. Мета тестування модулів - порівняння функцій, що реалізуються модулем, зі специфікаціями його функцій або інтерфейсу. Тестування модулів в основному орієнтовано на принцип «білого ящика». Це пояснюється перш за все тим, що принцип «білого ящика» важче реалізувати при переході в подальшому до тестування більших одиниць, наприклад, програм в цілому.

Крім того, подальші етапи тестування орієнтовані на виявлення помилок різного типу, тобто помилок, не обов'язково пов'язаних з логікою програми, а виникають, наприклад, через невідповідність програми вимогам користувача. Є великий вибір можливих підходів, які можуть бути використані для злиття модулів у більші одиниці.

Покрокове тестування.

Реалізація процесу тестування модулів спирається на два ключових положення: побудова ефективного набору тестів і вибір способу, за допомогою якого модулі комбінуються при побудові з них робочої програми. Друге положення є важливим, так як воно задає форму написання тестів модуля, типи засобів, використовуваних при тестуванні, порядок кодування і тестування модулів, вартість генерації тестів і вартість налагодження. Розглянемо два підходи до комбінування модулів: покрокове і монолітне тестування.

Виникає питання: «Що краще - по виконати окремо тестування кожного модуля, а потім, комбінуючи їх, сформувати робочу програму або ж кожен модуль для тестування підключати до набору раніше протестованих модулів?».

Перший підхід зазвичай називають монолітним методом, або методом «великого удару», при тестуванні і складанні програми;

Другий підхід відомий як покроковий метод тестування або збірки.

Метод покрокового тестування передбачає, що модулі тестуються не ізольовано один від одного, а підключаються по черзі для виконання тесту до набору вже раніше протестованих модулів. Покроковий процес триває до тих пір, поки до набору протестованих модулів не буде підключений останній модуль.

Детального розбору обох методів ми робити не будемо, наведемо лише деякі загальні висновки.

1. Монолітне тестування вимагає великих витрат праці. При покроковому ж тестуванні «знизу-вгору» витрати праці скорочуються.

2. Витрата машинного часу при монолітному тестуванні менше.

3. Використання монолітного методу надає великі можливості для паралельної організації роботи на початковій фазі тестування (тестування всіх модулів одночасно). Це положення може мати важливе значення при виконанні великих проектів, в яких багато модулів і багато виконавців,

оскільки чисельність персоналу, який бере участь в проекті, максимальна на початковій фазі.

4. При покроковому тестуванні раніше виявляються помилки в інтерфейсах між модулями, оскільки раніше починається збірка програми. На противагу цьому при монолітному тестуванні модулі «не бачать один одного» до останньої фази процесу тестування.

5. Налагодження програм при покроковому тестуванні легше. Якщо є помилки в міжмодульних інтерфейси, а зазвичай так і буває, то при монолітному тестуванні вони можуть бути виявлені лише тоді, коли зібрана вся програма. У цей момент локалізувати помилку досить важко, оскільки вона може перебувати в будь-якому місці програми. Навпаки, при покроковому тестуванні помилки такого типу в основному пов'язані з тим модулем, який підключається останнім.

6. Результати покрокового тестування більш досконалі. На закінчення відзначимо, що п. 1, 4, 5, 6 демонструють переваги покрокового тестування, а п. 2 і 3 - його недоліки. Оскільки для сучасного етапу розвитку обчислювальної техніки характерні тенденції до зменшення вартості апаратури і збільшення вартості праці, наслідки помилок в математичному забезпеченні досить серйозні, а вартість усунення помилки тим менше, ніж раніше вона виявлена; переваги, зазначені в п. 1, 4, 5, 6, виступають на перший план. У той же час збиток, що наноситься недоліками (п. 2 і 3), невеликий. Все це дозволяє нам зробити висновок, що покрокове тестування найбільш прийнятний.

Переконавшись у перевагах покрокового тестування перед монолітним, досліджуємо дві можливі стратегії тестування: спадний і висхідний. Перш за все з'ясуємо в термінологію.

По-перше, терміни «спадний тестування», «спадна розробка», «спадний проектування» часто використовуються як синоніми. Дійсно, терміни «спадний тестування» і «спадна розробка» є синонімами (в тому сенсі, що

вони мають на увазі певну стратегію при тестуванні і створенні текстів модулів), але спадний проектування - це зовсім інший і незалежний процес. Програма, спроектована низхідним методом, може тестуватися і низхідним, і висхідним методами.

По-друге, висхідна розробка, або тестування, часто ототожнюється з монолітним тестуванням. Це непорозуміння виникає через те, що початок висхідного тестування ідентично монолітному при тестуванні нижніх або термінальних модулів. Але вище ми показали, що висхідне тестування насправді являє собою покрокову стратегію.

Висхідне тестування При висхідному підході програма збирається і тестується «знизу вгору». Тільки модулі самого нижнього рівня («термінальні» модулі, модулі, які не викликають інших модулів) тестуються ізольовано, автономно. Після того як тестування цих модулів завершено, виклик їх повинен бути так само надійний, як виклик вбудованої функції мови або оператор присвоювання. Потім тестуються модулі, безпосередньо викликають вже перевірені. Ці модулі більш високого рівня тестуються не автономно, а разом з уже перевіреними модулями більш низького рівня. Процес повторюється до тих пір, поки не буде досягнута вершина. Тут завершується і тестування модулів, і тестування сполучень програми. При висхідному тестуванні для кожного модуля необхідний драйвер: потрібно подавати тести відповідно до сполучення модуля, що тестується. Одне з можливих рішень - написати для кожного модуля невелику провідну програму. Тестові дані представляються як «вбудовані» безпосередньо в цю програму змінні і структури даних, і вона багаторазово викликає тестовий модуль, з кожним викликом передаючи йому нові тестові дані. Є і краще рішення: скористатися програмою тестування модулів - це інструмент тестування, що дозволяє описувати тести на спеціальній мові і позбавляє від необхідності писати драйвери. Тут відсутні проблеми, пов'язані з неможливістю або складністю створення всіх тестових ситуацій, характерні

для низхідного тестування. Драйвер як засіб тестування застосовується безпосередньо до того модулю, який тестується, де немає проміжних модулів, які слід брати до уваги. Аналізуючи інші проблеми, що виникають при низхідному тестуванні, можна помітити, що при висхідному тестуванні неможливо прийняти нерозумне рішення про суміщення тестування з проектуванням програми, оскільки не можна почати тестування до тих пір, поки не спроектовані модулі нижнього рівня. Не існує також і труднощів з незавершеністю тестування одного модуля при переході до тестування іншого, тому що при висхідному тестуванні з застосуванням декількох версій заглушки немає складнощів з поданням тестових даних.

Спадне тестування. Спадне тестування не є повною протилежністю висхідному, але в першому наближенні може розглядатися як таке. При низхідному підході програма збирається і тестується «зверху вниз». Ізольовано тестується тільки головний модуль. Після того як тестування цього модуля завершено, з ним з'єднуються (наприклад, редактором зв'язків) один за іншим модулі, безпосередньо викликані їм, і тестується отримана комбінація. Процес повторюється до тих пір, поки не будуть зібрані і перевірені всі модулі. При цьому підході виникають два питання:

1. Що робити, коли тестовий модуль викликає модуль нижчого рівня (якого в даний момент ще не існує)?

2. Як подаються тестові дані?

Відповідь на перше питання полягає в тому , що для імітації функцій відсутніх модулів програмуються модулі - заглушки, які моделюють функції відсутніх модулів. Цікавий і друге питання: в якій формі готуються тестові дані і як вони передаються програмі? Якби головний модуль містив всі необхідні операції введення і виведення, відповідь була б проста: тести пишуться у вигляді звичайних для користувачів зовнішніх даних і передаються програмі через виділені їй пристрої введення. Так, проте, трапляється рідко. У добре спроектованої програмі фізичні операції введення-

виведення виконуються на нижніх рівнях структури, оскільки фізичний введення-виведення - абстракція досить низького рівня. Тому для того, щоб вирішити проблему економічно ефективно, модулі додаються не в строго низхідній послідовності (всі модулі одного горизонтального рівня, потім модулі наступного рівня), а таким чином, щоб забезпечити функціонування операцій фізичного введення-виведення якомога швидше. Коли ця мета досягнута, спадний тестування отримує значну перевагу: всі подальші тести готуються в тій же формі, яка розрахована на користувача. Спадний метод має як переваги, так і недоліки в порівнянні з висхідним. Найзначніше перевага - то, що цей метод поєднує тестування модуля, тестування сполучень і частково тестування зовнішніх функцій. З цим же пов'язана інша його перевага: коли модулі введення-виведення вже підключені, тести можна готувати в зручному вигляді. Спадний підхід вигідний також в тому випадку, коли є сумніви щодо здійсненності програми в цілому або, коли в проекті програми можуть виявитися серйозні дефекти. Перевагою спадного підходу дуже часто вважають відсутність необхідності в драйверах; замість драйверів вам просто слід написати «заглушки». Спадний метод тестування має, на жаль, деякі недоліки. Основним з них є те, що модуль рідко тестується досконально відразу після його підключення. Справа в тому, що дуже докладне тестування деяких модулів може зажадати вкрай витончених заглушок. Програміст часто вирішує не витрачати багато часу на їх програмування, а замість цього пише прості заглушки і перевіряє лише частину умов в модулі. Він, звичайно, збирається повернутися і закінчити тестування розглянутого модуля пізніше, коли прибере заглушки. Такий план тестування - безумовно не краще рішення, оскільки про відкладення умовах часто забувають. Другий тонкий недолік спадного підходу полягає в тому, що він може породити віру в можливість почати програмування і тестування верхнього рівня програми до того, як вся програма буде повністю спроектована. Ця ідея на перший погляд здається економічною, але зазвичай справа йде зовсім навпаки. Більшість

опитаних проектувальників визнає, що проектування програми - процес інтерактивний. Рідко перший проект виявляється досконалим. Нормальний стиль проектування структури програми передбачає після закінчення проектування нижніх рівнів повернутися назад і підправити верхній рівень, внісши в нього деякі удосконалення або виправляючи помилки, або іноді навіть викинути проект і почати все спочатку, тому що розробник раптово побачив кращий підхід. Якщо ж головна частина програми вже запрограмована і відтестована, то виникає серйозний опір будь-яким поліпшенням її структури. В кінцевому підсумку за рахунок таких поліпшень зазвичай можна заощадити більше, ніж ті кілька днів або тижнів, які розраховує виграти проектувальник, приступаючи до програмування занадто рано.

4.1.1. Ручне тестування

Ручне тестування призначеного для користувача інтерфейсу проводиться тестувальником-оператором, який керується в своїй роботі описом тестових прикладів у вигляді набору сценаріїв. Кожен сценарій включає перерахування послідовності дій, які повинні виконати оператор, і опис важливих для аналізу результатів тестування у відповідь реакцій системи, відбиваних в призначеному для користувача інтерфейсі.

Форма запису сценарію для проведення ручного тестування - таблиця, в якій в одній колонці описані дії (кроки сценарію), в іншій - очікувана реакція системи, а третя призначена для запису того, чи співпала очікувана реакція системи з реальною і перерахування неспівпадань.

Результати проведення тестування наведені в таблиці 4.1.

Таблиця 4.1.

Результати проведення ручного тестування

№ п/п	Дія	Реакція системи	Результат
1.	Запустіть програму	Появляється стартове вікно програми рисунок	Вірно
3.	Вибрати параметри заняття та створити дисципліну	Діалогове вікно рисунок 3.5 в необхідні поля вносимо інформацію	Вірно
4.	Внести параметри кожного студента академічної групи та створити академічну групу	Діалогове вікно рисунок 3.6 в необхідні поля вносимо інформацію	Вірно
5.	Вибрати академічну групу, дисципліну, дату, та тему і створити заняття	У формі рисунок 3.7 створено заняття з «Програмуванн»	Вірно
6	Внести інформацію про активність студента	За формою рисунок 3.8 можна внести інформацію про вибраного студента	Вірно
7	Створити звіт	Форма рисунок 3.9 дозволяє створити необхідний звіт	Вірно

4.2. Розгортання програмного продукту

Для розгортання програмного продукту необхідно отримати інсталяційний пакет, зберегти його на постійному носії та запустити. В

результаті запуску такого файлу в режимі діалогу на комп’ютер буде скопійовано необхідні для роботи файли у вказані директорії.

Висновки до четвертого розділу

В четвертому розділі проведено тестування мобільної програми «Електронний журнал» , тут показано її придатність до використання, а також надано інформацію, що до його розгортання. Також є інструкція користувачу, яка охоплює всі сценарії роботи з засобом.

ВИСНОВКИ

В результаті проведеного вивчення питання стану програмного забезпечення для програми «Електронний журнал» для ОС Android було поставлена задача створити своє аналогічне програмне забезпечення. Для створення такого програмного забезпечення використовували засіб розробки Android Studio, мова програмування Java, та технологія XML для зберігання даних. Це дало можливість створити програмне забезпечення мобільного типу. Тестування показало коректну роботу додатку. Інструкція до використання програмного забезпечення не є обов’язковою, так як програмний засіб є інтуїтивно зрозумілим та легким у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вигерс Карл Разработка требований к программному обеспечению. Пер, с англ. - М.:Издательско-торговый дом «Русская Редакция», 2004. - 576с.

2. Лешек А. Мацяшек Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ. - М.: Издательский дом "Вильямс". - 2002. - 432 с.

3. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПЗ Программная инженерия. Программные требования. Режим доступу: http://www.sorlik.ru/swebok/3-1-software_engineering_requirements.pdf

4. Фаулер Скотт До. UML в короткому викладі. Застосування стандартної мови об'єктного моделювання: Пер. з англ. – М.:Мир, 1999. – 191 с.

5. Нотон П. JAVA:Справ. руководство:Пер.с англ./Под ред.А.Тихонова.-.:БИНОМ:Восточ.Кн.Компания,1996:Восточ.Кн.Компания.-447с..-(Club Computer)

6. Патрик Нотон, Герберт Шилдт Полный справочник по Java.- McGraw-Hill,1997, Издательство "Диалектика",1997

7. Дэвид Флэнэген Java in a Nutshell.- O'Reilly & Associates, Inc., 1997, Издательская группа BHV, Киев, 1998

8. Чен М.С. и др.Программирование на JAVA:1001 совет:Наиболее полное руководство по Java и Visual J++:Пер.с англ./Чен М.С.,Грифис С.В.,Изи Э.Ф.-Минск:Попурри,1997.-640с.ил.+ Прил.(1диск).

9. Майкл Эферган Java: справочник.- QUE Corporation, 1997, Издательство "Питер Ком", 1998

10. Ренеган Э.Дж.(мл.)1001 адрес WEB для программистов:Новейший путеводитель программиста по ресурсам World Wide Web:Пер.с англ..-Минск:Попурри,1997.-512с.ил.

11. Сокольский М.В.Все об Intranet и Internet.-М.:Элиот,1998.-254с.ил.

12. Джо Вебер Технология Java в подлиннике.- QUE Corporation, 1996, "BHV-Санкт-Петербург",1997

13. Джейсон Мейнджер Java: Основы программирования.- McGraw-Hill,Inc.,1996, Издательская группа BHV, Киев,1997

14. И.Ю.Баженова Язык программирования Java.- АО "Диалог-МИФИ", 1997

15. Джон Родли Создание Java-апплетов.- The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996 5

16. Майкл Томас, Пратик Пател, Алан Хадсон, Доналд Болл(мл.) Секреты программирования для Internet на Java.- Ventana Press, Ventana Communications Group, U.S.A.,1996, Издательство "Питер Пресс", 1997

17. Аарон И.Волш Основы программирования на Java для World Wide Web.- IDG Books Worldwide,Inc.,1996, Издательство "Диалектика",1996

18. Кен Арнольд, Джеймс Гослинг Язык программирования Java.-Addison-Wesley Longman,U.S.A.,1996, Издательство "Питер-Пресс", 1997

ДОДАТКИ

Додаток А

Код програми

```
<?xml  version = "1.0"  encoding = "utf-8" ?>
<LinearLayout
 xmlns:android =
"http://schemas.android.com/apk/res/android"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 android:orientation = "vertical" >
<LinearLayout
 android:id = "@+id/linearLayout1"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<TextView
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Name"
 android:layout_marginLeft = "5dp"
 android:layout_marginRight = "5dp" >
</TextView>
<EditText
 android:id = "@+id/etName"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:layout_weight = "1" >
<requestFocus >
```

```
</requestFocus>
</EditText>
</LinearLayout>
<LinearLayout
 android:id = "@+id/linearLayout3"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<TextView
 android:id = "@+id/textView2"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Email"
 android:layout_marginLeft = "5dp"
 android:layout_marginRight = "5dp" >
</TextView>
<EditText
 android:id = "@+id/etEmail"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:layout_weight = "1" >
</EditText>
</LinearLayout>
<LinearLayout
 android:id = "@+id/linearLayout2"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<Button
 android:id = "@+id/btnAdd"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
```

```
 android:text = "Add" >
</Button>
<Button
 android:id = "@+id/btnRead"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Read" >
</Button>
<Button
 android:id = "@+id/btnClear"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Clear" >
</Button>
</LinearLayout>
</LinearLayout>
class MainActivity extends Activity implements
OnClickListener {

  final String LOG_TAG = "myLogs" ;   Button btnAdd,
btnRead, btnClear;   EditText etName, etEmail;
DBHelper dbHelper;
  / ** Called when the activity is first created. * /
  @Override
  Public void onCreate ( Bundle savedInstanceState ) {
    super .onCreate ( savedInstanceState ) ;
    setContentView ( R.layout.main ) ;     btnAdd =

( Button ) findViewById ( R.id.btnAdd ) ;
    btnAdd.setOnClickListener ( this ) ;     btnRead =
```

```
( Button ) findViewById ( R.id.btnRead ) ;
    btnRead.setOnClickListener ( this ) ;     btnClear =

( Button ) findViewById ( R.id.btnClear ) ;
    btnClear.setOnClickListener ( this ) ;     etName =

( EditText ) findViewById ( R. id.etName ) ;
    etEmail = ( EditText ) findViewById ( R.id.etEmail )
;

    // створюємо об'єкт для створення і управління
версіями БД
    dbHelper = new DBHelper ( this ) ;
  }


  @Override
  public void onClick ( View v ) {

    // створюємо об'єкт для даних
    ContentValues cv = new ContentValues () ;

    // отримуємо дані з полів введення
    String name = etName.getText () .toString () ;
    String email = etEmail.getText () .toString () ;

    // підключаємося до БД
    SQLiteDatabase db = dbHelper.getWritableDatabase ()
;
```

```
    switch ( v.getId ()) {
    case R.id.btnAdd:
      Log.d ( LOG_TAG, "--- Insert in mytable: ---" ) ;
      // підготуємо дані для вставки у вигляді пар:
найменування стовпця - значення

      cv.put ( "name" , name ) ;
      cv.put ( "email" , email ) ;
      // вставляємо запис і отримуємо її ID
      long rowID = db.insert ( "mytable" , null, cv ) ;
      Log.d ( LOG_TAG, "row inserted, ID =" + rowID ) ;
      break ;
    case R.id.btnRead:
      Log.d ( LOG_TAG, "--- Rows in mytable: ---" ) ;
      // робимо запит всіх даних з таблиці mytable,
отримуємо Cursor
      Cursor c = db.query ( "mytable" , null, null,
null, null, null, null ) ;

      // ставимо позицію курсора на перший рядок вибірки
      // якщо в вибірці немає рядків, повернеться false
      if ( c.moveToFirst ()) {

        // визначаємо номери стовпців на ім'я в вибірці
        int idColIndex = c.getColumnIndex ( "id" ) ;
        int nameColIndex = c.getColumnIndex ( "name" ) ;
        int emailColIndex = c.getColumnIndex ( "email" )
;
```

```
        do {
          // отримуємо значення за номерами стовпців і
пишемо все в лог
          Log.d ( LOG_TAG,
              "ID =" + c.getInt ( idColIndex ) +
              ", name =" + c. getString ( nameColIndex )
+
              ", email =" + c.getString ( emailColIndex
)) ;
          // перехід на наступний рядок
          // а якщо наступної немає (поточна - остання),
то false - виходимо з циклу
        } while ( c.moveToNext ( )) ;
      } else
        Log.d ( LOG_TAG, "0 rows" ) ;
      c.close () ;
      break ;
    case R.id.btnClear:
      Log.d ( LOG_TAG, "--- Clear mytable: ---" ) ;
      // видаляємо все записи
      int clearCount = db.delete ( "mytable" , null,
null ) ;
      Log.d ( LOG_TAG, "deleted rows count =" +
clearCount ) ;
      break ;
    }
    // закриваємо підключення до БД
    dbHelper.close () ;
  }
```

```
  class DBHelper extends SQLiteOpenHelper {

    public DBHelper ( Context context ) {
      // конструктор суперкласу
      super ( context, "myDB" , null, 1 ) ;
    }

    @Override
    public void onCreate ( SQLiteDatabase db ) {
      Log.d ( LOG_TAG, "--- onCreate database ---" ) ;
      // створюємо таблицю з полями
      db.execSQL ( "create table mytable ("
          + "id integer primary key autoincrement,"
          + "name text,"
          + "email text" + " ); " ) ;
    }

    @Override
    public void onUpgrade ( SQLiteDatabase db, int
oldVersion, int newVersion ) {     }   } }

private DatabaseHelper mDatabaseHelper;
private SQLiteDatabase mSqLiteDatabase;

public void onCreate (Bundle savedInstanceState) {
    super.onCreate (savedInstanceState);
```

```
    setContentView (R.layout.activity_main);

    mDatabaseHelper = new DatabaseHelper (this,
"mydatabase.db", null, 1);

    mSqLiteDatabase =
mDatabaseHelper.getWritableDatabase ();

    ContentValues values = new ContentValues ();
    // Задайте значення кожному за шпальти
    values.put (DatabaseHelper.CAT_NAME_COLUMN,
"Рижик");
    values.put (DatabaseHelper.PHONE_COLUMN,
"4954553443");
    values.put (DatabaseHelper.AGE_COLUMN, "5");
    // Вставляємо дані в таблицю
    mSqLiteDatabase.insert ( "cats", null, values);
}

public void onClick (View view) {
    Cursor cursor = mSqLiteDatabase.query ( "cats", new
String [] {DatabaseHelper.CAT_NAME_COLUMN,
                    DatabaseHelper.PHONE_COLUMN,
DatabaseHelper.AGE_COLUMN},
            null, null,
            null, null, null);

    cursor.moveToFirst ();
```

```
    String catName = cursor.getString
(cursor.getColumnIndex
(DatabaseHelper.CAT_NAME_COLUMN));
    int phoneNumber = cursor.getInt
(cursor.getColumnIndex (DatabaseHelper.PHONE_COLUMN));
    int age = cursor.getInt (cursor.getColumnIndex
(DatabaseHelper.AGE_COLUMN));

    TextView infoTextView = (TextView) findViewById
(R.id.textView);
    infoTextView.setText ( "Кот" + catName + "має
телефон" + phoneNumber + "і йому" +
            age + "років");

    // Не забуваємо закривати курсор
    cursor.close ();
}
```