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: «Інформаційна система ОСББ «Оболоня-парк»»

Студент групи КН-41 Бугера Н.В. _____
(підпис)

Керівник проєкту Кузик В. М. _____
(підпис)

Консультанти:

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

нормоконтролер Гавришків Н. Г. _____
(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 проєктування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»

студенту Бугері Назарію Володимировичу

1. Тема проєкту «Інформаційна система ОСББ «Оболоня-парк»
затверджена наказом по коледжу від “01” жовтня 2021р., № 178-н
2. Термін здачі студентом завершеного проєкту “___” _____2022 р.
3. Вихідні дані до проєкту _____
_____

4. Перелік питань, які повинні бути розроблені в проєкті:
а) основна частина _____
_____

б) техніко-економічне обґрунтування _____
_____

5. Перелік графічного матеріалу _____
_____

6. Консультанти проєкту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступень, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН

дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування	20.09.21 р.	01.10.21 р.
2.	Огляд типових рішень та написання відповідного розділу ПЗ	20.11.21 р.	27.01.22 р.
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	27.01.22 р.	14.02.22 р.
4.	Розробка функціональних вимог до проєкту та робота надструктурою програмного продукту. Написання відповідного розділу ПЗ.	17.02.22 р	02.03.22 р.
5.	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ.	02.03.22 р.	16.03.22 р.
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.22 р.	17.04.22 р.
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.22 р.	04.05.22 р.
8.	Доопрацювання модулів	05.05.22 р.	18.05.22 р.
9.	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.22 р.	19.05.22 р.
10.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	19.05.22 р.	05.06.22 р.
11.	Робота над оформленням пояснювальної записки	05.06.22 р.	14.06.22 р.
12.	Оформлення пояснювальної записки	13.06.22 р.	14.06.22 р.
13.	Попередній захист дипломного проєкту	15.06.22 р.	16.06.22 р.
14.	Підготовка до захисту дипломного проєкту	17.06.22 р.	24.06.22 р.
15.	Захист дипломного проєкту	25.06.22 р.	26.06.22 р.

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

Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Дипломний проєкт. «Інформаційна система ОСББ «Оболоня-парк»». Сторінок – 70, рисунків – 25, додатків – 1.

Для належного виконання запланованого проєкту буде проведено аналіз подібних інформаційних ресурсів у вигляді веб-сайтів, аналогів мобільних додатків для Android обраної предметної області у публічному користуванні не виявлено. Було обрано дві окремі інформаційні системи ОСББ:

– ОСББ «ПРИДНІПРОВСЬКЕ» (prydniprovske.aosbb.kiev.ua);

– ОСББ «Зоряне» (http://zoriane.kiev.ua/);

Мета проєкту – інформаційну систему ОСББ «Оболоня-парк» для учасників ОСББ та його управління під операційну систему Android.

Завданням проєкту є розробка мобільного додатка інформаційної системи актуальними засобами, який буде відповідати створеному технічному завданню та сучасним вимогам подібних додатків.

Результатом буде додаток, що можна вільно встановлювати та використовувати. Наявність потрібної інфраструктури користувацького інтерфейсу для легкої координації при пошуку потрібної інформації.

Щоб розробити такий мобільний додаток при огляді наявних середовищ було обрано середовище розробки Android Studio. Для реалізації необхідних збереження, відображення інформації використано SQLite спільно з технологією Room, що реалізована сучасною мовою програмування Kotlin.

МОБІЛЬНИЙ ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, KOTLIN, ROOM, SQLITE, ANDROID STUDIO, ОПЕРАЦІЙНА СИСТЕМА ANDROID, DAO

Abstract

Diploma project. ««Obolonya-Park» OSBB information system». Pages – 70, figures – 25, appendices – 1. For the proper implementation of the planned project, an analysis of such information resources in the form of websites will be conducted, analogues of mobile applications for Android in the selected subject area in public use have not been identified. Two separate information systems of OSBB information system were chosen:

– «PRYDNIPROVSKE» OSBB (prydniprovske.aosbb.kiev.ua);

– «ZORIANE» OSBB (http://zoriane.kiev.ua/);

The purpose of the project is the «Obolonya-Park» OSBB information system for OSBB participants and its management for the Android operating system.

The task of the project is to develop a mobile application of the information system with relevant tools that will meet the established technical task and modern requirements of such applications

The result is an application that can be freely installed and used. Having the right user interface infrastructure for easy coordination when searching for the right information.

To develop such a mobile application, when reviewing existing environments, the Android Studio development environment was chosen. SQLite is used in conjunction with Room technology, implemented in the modern Kotlin programming language, to implement the necessary storage and display of information.

MOBILE APPLICATION, INFORMATION SYSTEM, KOTLIN, ROOM, SQLITE, ANDROID STUDIO, ANDROID OPERATING SYSTEM, DAO

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

PHP – об'єктно-орієнтована серверна мова програмування.

SQLite – компактна система управління базами даних, яка є спрощеною та полегшеною версією повноцінних варіантів СУБД, що дозволяє використовувати її для локального зберігання даних для Android додатків і не тільки.

ОС – операційна система.

ОСББ – об'єднання співвласників багатоквартирного будинку.

Room – бібліотека, що надає зручну обгортку для роботи з базою даних SQLite.

DAO – Data Access Object, об’єкти доступу до даних, основні класи, де визначається взаємодія з базою даних.

SDK – Android Software Development Kit, універсальний набір інструментів, що утворює повноцінне середовище розробки мобільних додатків для ОС Android.

ERP – Enterprise Resource Planning, це системи, які підтримують усі бізнес-процеси підприємства: управління фінансами, продажами, виробництвом, логістикою, операційною діяльністю, відносинами з клієнтами, звітність та інші.

UI – User Interface, інтерфейс користувача, що представляє структуру програми для користувача, з яким їм взаємодіє на пряму.

ВСТУП

Розробка даної дипломної роботи буде зосереджена навколо декількох основних її складових. Деякі будуть мати відношення до ідентифікації проєкту, як інформаційна система, що являється величезною окремою темою у теперішньому сучасному цифровому просторі та мобільних додатків зокрема. Інші будуть мати відношення до безпосередніх проєктних рішень та архітектури додатка, що буде розроблено.

Декілька тверджень, чому було обрано саме ОС Android:

– легка настройка – цю характеристику визначає універсальність платформи з високою гнучкістю та простими параметрами налаштування. Також Android це надійна ОС, яка може інтегрувати всі типи модифікацій, від простих до складних;

– технологія носіння, що стає новою нормою в корпоративному секторі. Пристрої, що носяться, стають ефективним інструментом спілкування та взаємодії з різноманітними інформаційними системами, зокрема і такою, що буде розроблена в результаті виконання дипломної роботи;

– платформа, що розвивається. Android все ще розвивається. Смартфони Android створені кількома відомими компаніями, такими як Samsung. Вони продовжують створювати нові функції, щоб залишатися стійкими на тлі зростаючої конкуренції, і спільнота розробників Android швидко адаптується до них.

Інформаційні системи в будь-якому її вигляді в сучасних цифрових технологіях потрібні кожній предметній області. Розроблена інформаційна система для ОСББ насамперед штовхає цю предметну область до прогресування у світі цифрових технологій, коли така система стане доступною велика кількість учасників ОСББ залюбки використовуватиме її для довідки по інформації відповідного об’єднання співвласників. Таким чином створення такої локальної системи оправдує її розробку.

1 АНАЛІТИЧНИЙ РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд об’єкту інформатизації

В основу ідеї дипломного проєкту входить суть створення та функціонування ОСББ, як об’єднання з значними перевагами, якщо в основі будуть відповідальні та заохочені люди. До явних переваг можна віднести:

– співвласники мають можливість приймати швидкі та ефективні рішення щодо управлінням власним будинком. В ОСББ зрозуміло та строго визначена структура основних органів та їх обов’язки. Загальні збори ОСББ можуть приймати свої рішення, а управління ОСББ, що є його виконавчим органом, будуть вести контроль за виконанням висунутих рішень;

– співвласники повноцінно керують спільним майном свого багатоквартирного будинку, який є їхньою спільною сумісною власністю відповідно до статті 5 Закону № 417. Це дасть можливість власними силами формувати бюджет, штатний розклад, встановлювати внески на обслуговування та утримання будинку, обирати лише якісних постачальників комунальних послуг для будинку;

– співвласники можуть заробляти гроші на спільній власності – наприклад, віддавати фасад будинку в оренду під рекламні площі, або облаштувати підвал складським приміщенням. Кошти від оренди підвалів, прибудинкової території, магазинів, що розташовані в будинку будуть відповідно витрачатися на обслуговування багатоквартирного будинку;

– співвласники повністю контролюють використання своїх бюджетних коштів. Поступлення коштів для виконання ОСББ своїх функцій співвласники здійснюють сплачуванням внесків на утримання й ремонт спільного майна, розмір та склад яких узгоджується та приймається рішенням загальних зборів об’єднання.

– співвласники мають повне право власноруч проводити модернізацію свого будинку, що дозволить при правильному підході зменшити суми оплати за житлово-комунальні послуги. Як результат співвласники покращуватимуть стан будинку і умови проживання у ньому;

– співвласники можуть виконувати необхідні та термінові ремонтні роботи в будинку, визначаючи їх першочерговість, а керівництво ОСББ буде швидше реагувати на наявні проблеми;

– ОСББ може проводити укладання будь-яких угод, замовляти різні послуги, необхідні для утримання будинку та реалізації рішень загальних зборів. Об’єднання зможе укладати договори про обслуговування будинку з компаніями, які пропонують кращі послуги, а також розірвати договір, якщо компанія неякісно виконуватиме роботу. Це влаштує контроль за якістю надання житлово-комунальних послуг;

– ОСББ це статус юридичної особи, що дає можливість відкривати рахунки в банках, підписувати договори, виконувати наймання персоналу та повноцінно розпоряджатися власними коштами для забезпечення управління багатоквартирним будинком.

Також ОСББ має деякі відносні та фактичні недоліки, а тому разом з масою позитивних моментів, що надає організація ОСББ, є речі, які слід взяти до уваги:

– вибір голови об'єднання. Важливий момент, якому потрібно приділити багато часу та обговорень. Саме від вибору людини на цю посаду в ОСББ і його організаційних здібностей буде залежати ефективність володіння будинком і якість життя мешканців;

– утримання території. При створенні ОСББ у його власність переходить і територія, що вважається прибудинковою, і тепер вирішувати проблеми зв’язані з нею(прибирання, догляд) доведеться учасникам ОСББ;

– санкції неплатникам. У статуті ОСББ, зазвичай, прописуються достатньо жорсткі санкції неплатникам, які дозволяють легко та повноправно відключити невідповідального мешканця від комунальних послуг навіть без проведення будь-якого суду;

– необхідний великий активний вклад кожного учасника ОСББ та постійна взаємодія з управлінням та іншими учасниками.

Це все що стосувалось об’єкту, який буде інтегровано в додаток при виконанні дипломного проєкту.

1.2 Опис предметної області

В цьому розділі буде проведено опис шаблону предметної області, а саме інформаційну систему, що є важливим елементом для всього інформаційного простору. Буде викладено поточні та можливі тенденції розвитку інформаційних систем. Представлено цілі, завдання та метод дослідження. Також буде коротко описано про деякі пов’язані дослідження.

Інформаційні системи постійно розвиваються та будуть все більш і більш інтерактивними, і все більше сфер медіа будуть інтегровані в ці системи. Оскільки різні IT-системи стають легшими та «природнішими» для взаємодії, все більше роботи виконується різними IT-рішеннями або за їх підтримки. Мова вже не йде лише про побудову інформаційних систем з метою автоматизації та раціоналізації певних робочих процесів. Використання та розвиток різних IT-додатків та інформаційних систем стає все більше питанням підтримки пошуку та надання інформації, а також розваг та освіти. Інформаційні теперішнього покоління – це інтеграція газет, телебачення, радіо та «традиційних» інформаційних систем, які вже існували в кількох сервісах і порталах в Інтернеті. Електронний бізнес це яскравий приклад сфери, яка повністю поклалася на це розвиток інформаційних систем, щоб досягти успіху. Людям потрібні легкі та насичені інтерфейси та прямий зворотній зв’язок. Якщо покупки в мережі мають конкурувати зі звичайними покупками, вони повинні, наскільки це можливо, пропонувати можливості відчути, випробувати та вивчити продукти, які продаються. Також має бути доступним легке та інтерактивне спілкування та підтримка, що в просунутих системах завжди присутнє в зв’язку з конкуренцією на ринку інформаційних систем.

Сучасні інформаційні системи загалом базуються на розповсюджених та усім доступних платформах. Тенденція спрямована на спільність платформи та вимоги до легкого доступу та доступу незалежно від місця розташування чи конкретних стандартів технічного застосування. Доступ до Інтернету можливий через набагато більше каналів, ніж через ПК та звичайні мережеві термінали. Мобільні телефони, цифрове телебачення та інші електронні аксесуари.

Розробка на основі компонентного підходу стала поширеною практикою та прийняла більш сучасні та просунуті методології. Це все впливає на стрімку швидкість розробки та стандартизації програмних додатків. Тенденція до спільності платформ, підкреслює цей прогрес, оскільки набуває піку поширеності, а це значить що окремі програмні компоненти, послуги і навіть загальні класи програмування можна купити та інтегрувати з різними додатками.

Постійний приріст трафіку таких систем та нові мобільні технології роблять використання різноманітних інформаційних систем більш поширеними та необхідними, як інтегровані частини різних продуктів для цілей надання інформації. Прикладом цього є, наприклад, підключення до Інтернету в автомобілях, які збільшують можливості для водіїв отримати різноманітні послуги. Загалом це робить інформаційні системи більш складними та багато дисциплінарними, ніж будь-коли раніше. Це означає, що розробка та обслуговування інформаційних систем відповідатиме новим викликам, які вимагають нових методів і прийомів, що буде сприяти актуальності та розвитку сфери інформаційних систем і надалі.

Зараз відбувається постійне об’єднання традиційних методів розробки інформаційних систем з новими способами роботи, розробленими з інших дисциплін, щоб знайти спосіб інтеграції всіх компетенцій, які необхідні для створення майбутніх мультимедійних інформаційних систем ще вищою якості.

Фундаментальні проблеми розвитку, які впливали в минулому і частково продовжують це робити, інформаційних систем. На просторах інтернету відносно різних досліджень існує шість фундаментальних проблем розвитку систем, які справедливі щодо всіх нетривіальних інформаційних систем. Нижче представлені ці фундаментальні та пов’язані проблеми.

– Люди мають вузькі когнітивні обмеження. Оскільки всі нетривіальні інформаційні системи перевищують когнітивні межі людини, існує потреба в інструментах і методах подолання цих обмежень. Прикладами цього є моделі та методи розробки інформаційних систем.

– Інформаційні системи складні. Інформаційні системи майже за визначенням було важко оглянути та зрозуміти, оскільки вони перевершують когнітивні обмеження більшості людей, що не втягнуті в безпосередню розробку цих систем. Інформаційні системи самі по собі є складними, а також розробка та побудова інформаційних систем є складним завданням. Для того, щоб мати можливість оглядати складні системи, їх поділяють на набір визначених взаємопов'язаних частин (можна розглядати як підсистеми та компоненти). Робота по створенню інформаційних систем повинна підтримуватися, контролюватися та аналізуватися певною науковою методологією, щоб створити гарну систему. У такій роботі, як розробка інформаційних систем, це не може бути здійснено за допомогою інтуїції та припущень, оскільки ці припущення, як правило, суперечливі. Є кілька прикладів нових Інтернет-компаній, які почали створювати веб-сторінки, які не вимагали використання будь-яких комплексних методологій розробки систем. Прості веб-сторінки розвиваються, додаються нові функції, а сторінки підключаються до інших систем. Це спричинювало деякі проблеми, тому що з підвищеною складністю неможливо впоратися без методичного забезпечення.

– Інформаційні системи багатопрофільні. Існує кілька зацікавлених сторін інформаційної системи. Ці зацікавлені сторони дійсно мають різні інтереси, потреби та навички, пов'язані з системами. Ця проблема може бути пов'язана як із використанням, так і з розвитком системи. Користувачі системи можуть мати різні потреби, тому важливо, щоб система могла бути адаптована до різних профілів користувачів. Поки система розробляється, різні зацікавлені сторони можуть мати проблеми зі спілкуванням через відсутність спільної мови та зосередженості на наявних проблемах. При розробці мультимедійних інформаційних систем ця проблема повинна викликати великий інтерес, оскільки в такому проєкті поєднуються інженери та актори з різних дисциплін.

– Інформаційні системи динамічні. Якщо інформаційні системи мають бути підтримкою для користувачів, їх необхідно швидко перепроєктувати.

Зміни в середовищі безперервно висувають нові вимоги до користувачів, а отже, і потребу в системній підтримці змінюються. Величезні та складні системи, як правило, важко змінити. Ця проблема, ймовірно, стає ще більш серйозною, оскільки кількість медіа, інтегрованих у системи, зростає.

– Інформаційні системи інфологічні. Актуальність інформаційної системи залежить від погляду кожного окремого користувача на систему та того, що вона містить, що робить участь користувача необхідним у процесі проєктування системи.

– Існують людські та соціальні аспекти інформаційних систем. Інформаційні системи впливають як на людські, так і на соціальні системи в навколишньому середовищі, де вони діють. Людські та соціальні системи підлягають обслуговуванню та підтримці з боку інформаційних систем. Не буде перебільшенням стверджувати, що розробники систем часто ігнорують цей факт. Оскільки характер інформаційних систем стає все менше питанням автоматизації та раціоналізації, людські та соціальні аспекти стають все більш важливими, які необхідно враховувати при розробці інформаційних систем.

Розробити інформаційну систему і в той же час подбати про поточні зміни в організації, якій має обслуговувати інформаційна система, є викликом. Це стає особливо складним, якщо розробникам доводиться брати до уваги всі фундаментальні проблеми, як описано вище. Останнім часом додатки інформаційних систем стали більш виразними завдяки впровадженню мультимедіа, інтерактивних медіа та Інтернету. Гіпотеза полягає в тому, що майбутні інформаційні системи будуть ще більш розподіленими, мережевими та більш інтерактивними та мультимедійними в порівнянні з традиційними інформаційними системами. Інтернет відкриває нові можливості щодо інтеграції організацій на географічних відстанях, а також розвитку нових комунікаційних інструментів, де зливаються IT, телекомунікації та медіа. Це має дати нам можливість створювати інформаційні системи, адаптовані (інфологічні) до потреб кожного користувача.

Завдяки цьому ми також отримаємо інформаційні системи, які є більш складними, динамічними та мультидисциплінарними, ніж будь-коли раніше.

Поняття інформації є складним, і можна знайти різноманітні визначення та тлумачення. Крім того, існують різні погляди на те, як поняття інформації пов’язане з поняттями даних і знання. У цій роботі інформація розглядається як явище, що залежить від інтерпретації даних. У цьому поданні дані є лише символами та сигналами, які виражають і охоплюють інформацію. Це досить поширене тлумачення поняття в суспільстві інформаційних систем. Рисунок 1.1 ілюструє погляд на зв’язок між цими поняттями.

Рисунок 1.1 – Зв’язок між поняттями інформаційної системи

Дані – це вираз у вигляді символів і певних сигналів. Прикладом даних є послідовність букв на цій сторінці. Дані в цьому контексті знаходяться на синтаксичному рівні.

Інформація – це інтерпретація даних одержувачем. Інтерпретація залежить від представлення даних, а також від системи приймача. Інформація має певний зміст і семантику в залежності від представлення, а також інтерпретації одержувачем.

Знання залежать від особистого досвіду і знаходяться на прагматичному рівні. Знання генеруються під час застосування інформації та пов’язаної з іншою інформацією, а знання використовуються для адаптації та коригування поведінки до нових ситуацій.

Інформаційну систему можна коротко визначити як систему, яка керує інформацією шляхом: збору, маніпуляції, зберігання, передачі та відображення інформації. Оскільки більшість інформаційних систем використовується для підтримки певної організації, необхідно коротко обговорити зв’язок між цими двома поняттями. Організації як системи впорядковані на різних рівнях ієрархії залежно від типу роботи, що виконується на кожному рівні. Рисунок 1.2 ілюструє основні ієрархічні рівні звичайної бізнес-організації.

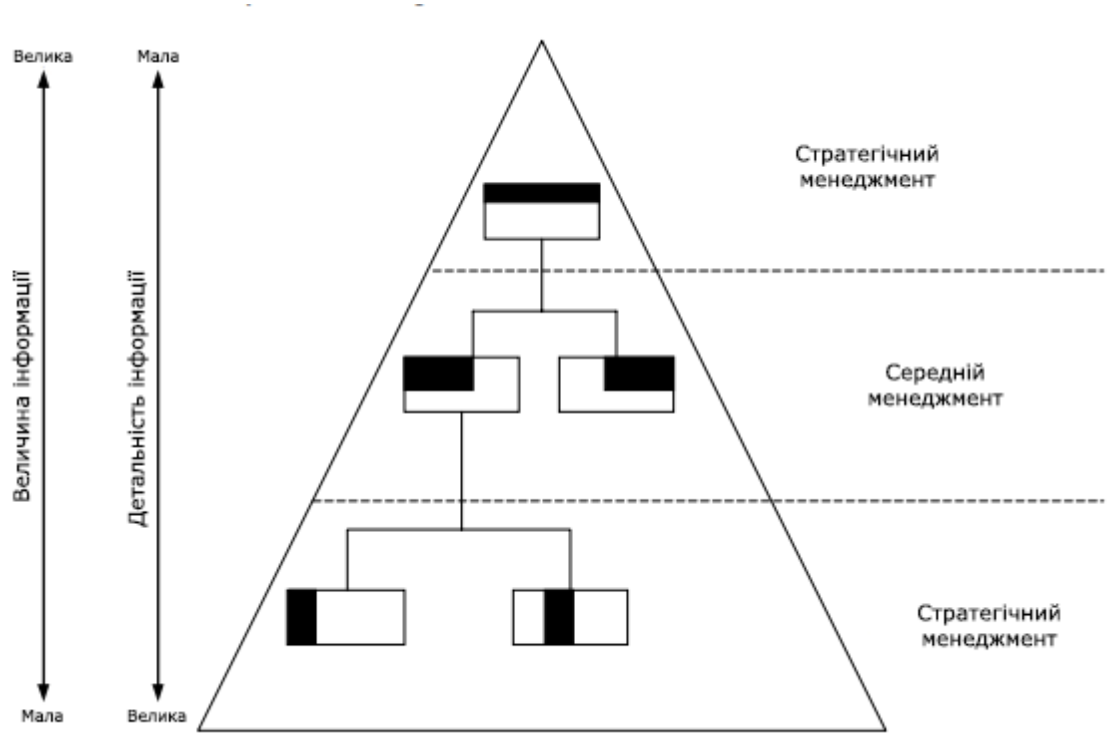


Рисунок 1.2 – Основні ієрархічні рівні інформаційної системи

Кожен з рівнів нижче керівництва організації впорядкований за різними функціями, спеціалізованими для виконання конкретних завдань. Рівень стратегічного менеджменту – це той, на який спрямована організація в цілому. На середньому управлінському рівні здійснюється тактичне управління, таке як планування та координація. Безпосередня робота організації з додавання вартості здійснюється на рівні оперативного управління. Інформаційні потреби кожного рівня сильно відрізняються. Рішення в першу чергу приймаються на рівні стратегічного менеджменту. Для прийняття рішень на цьому рівні необхідна широка і вичерпна інформація від усієї організації.

На нижчих організаційних рівнях інформаційні потреби стають більш спеціалізованими та детальними. Інформаційні системи бізнесу створюються для підтримки організації, до якої вони належать. Підтримка, яку інформаційна система повинна надати організації, може бути у формі ефективної діяльності, підтримки бізнес-аналізу та моніторингу цілей і досягнення цілей.

Інформаційні системи включають як формальні, так і неформальні аспекти. Неформальні аспекти насамперед орієнтовані на явища та види діяльності, які важко формалізувати, наприклад, соціальні навички, досвід і знання про реакції різних людей у різних ситуаціях. На додаток до цієї категоризації інформаційну систему можна розділити на комп’ютеризовану та некомп’ютеризовану частину. Комп'ютеризована частина інформаційної системи складається в основному з тих аспектів системи, які можна легко формалізувати і тим самим автоматизувати. Це стосується переважно повторюваних операцій (процедур), які можна раціоналізувати за допомогою автоматизації. Це означає, що інформаційна система не є єдиним і простим об’єктом, а має кілька компонентів і зацікавлених сторін. Відношення зацікавленої сторони до інформаційної системи кожного разу залежить від погляду, який цікавить. У цій роботі ми будемо стосуватися концепції інформаційної системи як комп’ютеризованої частини інформаційної системи.

Рисунок 1.3 описує погляд на інформаційні системи, як підмножину робочої системи, а також контекст інформаційних систем щодо організації, інформаційних технологій (тобто апаратного та програмного забезпечення) та бізнес-середовища.

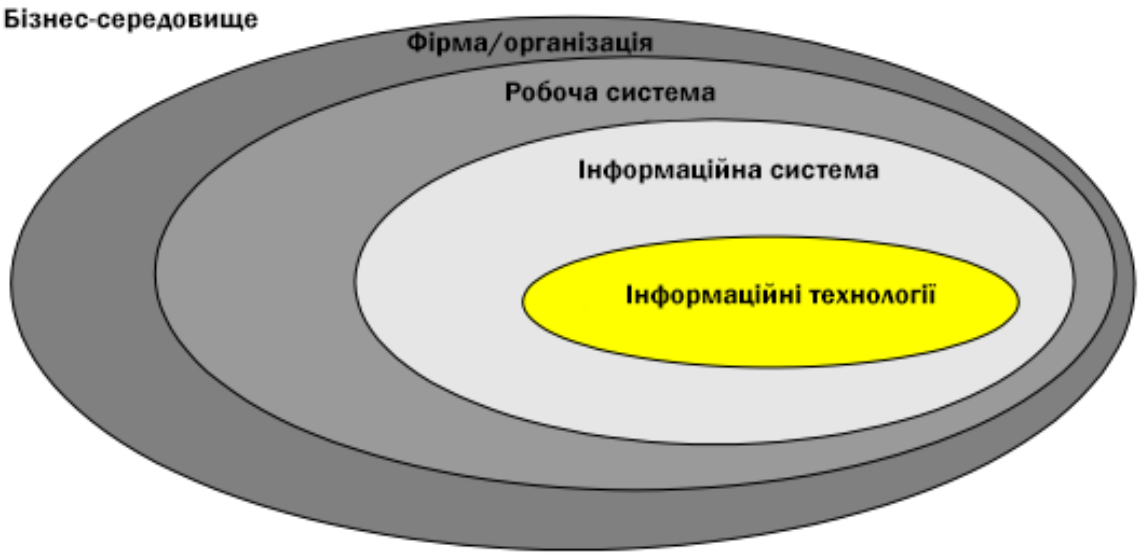


Рисунок 1.3 – Бізнес-середовище інформаційного середовища

Різні рівні організації вимагають різних типів підтримки інформаційних систем і представлення інформації. Рисунок 1.4 є спробою дати огляд того, як різні типи бізнес-інформаційних систем приблизно пов'язані з різними організаційними рівнями, представленими на рисунку 1.2.

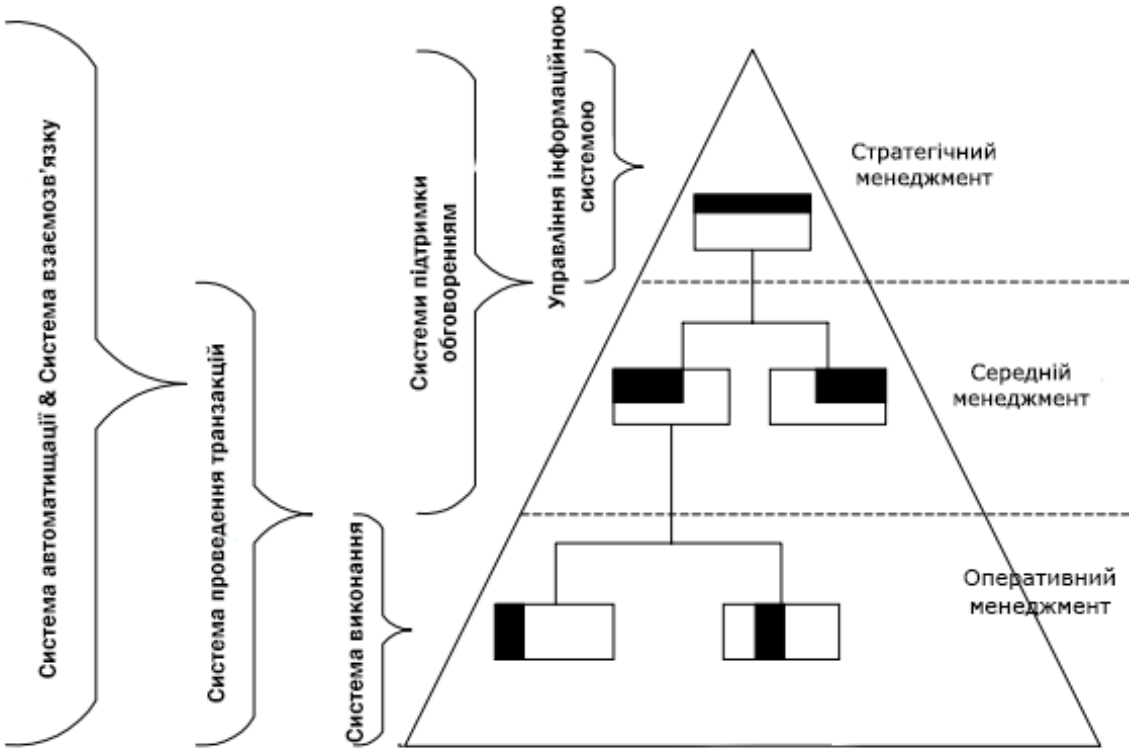


Рисунок 1.4 – Бізнес-середовище інформаційного середовища

Можна розділити інформаційні системи на три основні категорії: бізнес-інформаційні системи, комп’ютерні засоби навчання та розважальні засоби.

– Системи ділової інформації розробляються для обслуговування та підтримки організації. Ці системи в деякій літературі називаються інформаційними системами управління.

– Комп'ютерні засоби навчання (можна замінити терміном електронне навчання) – це мультимедійні системи, які можна використовувати як доповнення до звичайної освіти, а також для навчання та керівництва співробітникам різних організацій, а також приватні особи. Комп’ютерні засоби навчання можуть бути доступними як поза, так і в Інтернеті. Основною перевагою цих систем є різноманітні можливості, які пропонує мультимедіа відповідно до керівництва та представлення різноманітних матеріалів. Користувач також може вільно використовувати системи, коли захоче, вибрати рівень складності та повторити певні завдання.

– Розважальні медіа стають все більш важливими, оскільки їх комерційний потенціал зростає. Комп’ютерні ігри є прикладами засобу, який так само, як фільми та романи, можна використовувати для пояснення та вираження різних аспектів навколишнього світу.

В цьому розділі було описано головні та важливі концептуальні моменти природи інформаційних систем. В розділі 1.4 буде описано загальну концепцію інформаційних систем зі сторони програмного інженера та того, як відбувається їх конструювання, впровадження, реалізація.

1.3 Огляд обраних аналогів

Буде проведено опис того, що зможе побачити будь-який користувач при відвідуванні веб-сайтів (функціональність, графічний контент да представлені дані). Першими будуть оглянуті два веб-сайти конкретних ОСББ.

Першим веб-сайтом буде проведено огляд власного сайту ОСББ «Зоряне», яке доступне за посиланням http://zoriane.kiev.ua/, з’єднання є незахищеним, що видає достатню простоту сайту. Якість виконання інформаційних систем

напряму залежить від потреб замовників та майбутніх користувачів, що обговорюються перед початком проєктування.

При переході за посиланням користувача зустрічатиме головна сторінка, де знаходиться повідомлення, що знайомить користувача з об'єктом інформаційної системи – ОСББ «Зоряне». На головній сторінці також знаходиться невеликий блок новин, з якими користувач може ознайомитись. Для навігації по сайту користувачеві пропонується цілих два меню, верхнє та бокове. Вигляд головної сторінки зображено на рисунку 1.5.

Рисунок 1.5 – Вигляд головної сторінки ОСББ «Зоряне»

З присутності двох меню та деяких підменю окремих параметрів можна сказати, що є деяке перенасичення інформації та відсутність належної її уніфікації. Но це не завжди рахується, як мінус, адже, потрібну інформацію і в такому випадку не важко знайти для користувача. Відсутність уніфікації це лише максимальна простота розробки. Далі буде розглянуто основні для розробки дипломного проєкту функції да сторінки з інформацією. Ще на головній сторінці є елемент мультимедіа – слайдер з зображеннями житлового об'єкту ОСББ.

В більшості усі пункти меню ведуть до невеликих блоків з відповідною інформацією про деякі інформаційні елементи ОСББ. На рисунку 1.6 вигляд одного з інформаційних блоків під назвою Сервісні служби.

Рисунок 1.6 – Огляд інформаційного блоку Сервісні служби

Інформація по всіх блоках відображені у вигляді простих таблиць. Відрізняється від інформаційних сторінок лише сторінка контактів, що дозволяє зворотній зв'язок між користувачами, потенційними чи фактичними учасниками ОСББ та управлінням. Сторінка контактів виглядає максимально просто, на ній присутні елементи для введення потрібної інформації та кнопки, що буде надсилати створене повідомлення на електронну пошту ОСББ чи одного члена з управління.

Сторінка Контакти зображена на рисунку 1.7.

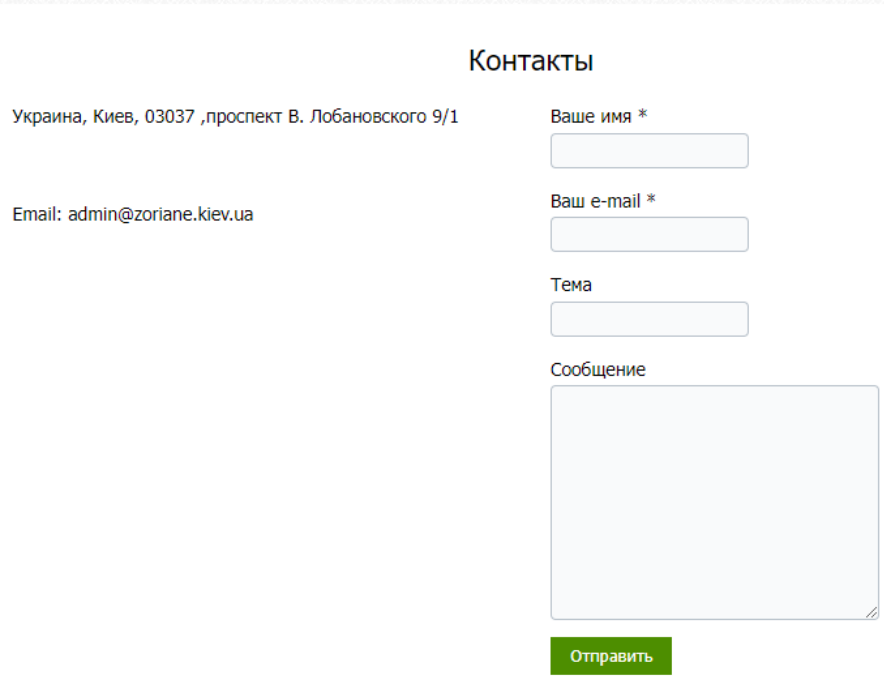


Рисунок 1.7 – Сторінка Контакти для зворотного зв’язку

Більше ніяких особливостей сайту не виявлено. В цілому такі інформаційні системи успішно виконують свою функцію та не потребують надзвичайних модифікацій. Усі пункти меню ведуть до відповідної інформації та несправностей в роботі сайту не помічено.

Наступним кроком розділу є огляд подібного веб-сайту ОСББ «Придніпровське». При переході за посиланням https://prydniprovske.aosbb.kiev.ua/ відображається головна сторінка ОСББ. На головній сторінці присутнє привітання користувача та зображення будинку об’єднання. Прокрутивши вниз також бачимо, що на головній сторінці присутній і блок актуальних новин, коротка інформація, блок оголошень, перелік документів, що може знадобитись користувачам та блок контактів в кінці головної сторінки. Відповідно і відображене головне меню, що присутнє на кожній з сторінок для переходу між різними сторінками з потрібною інформацією.

Головна сторінка зображена на рисунку 1.8.

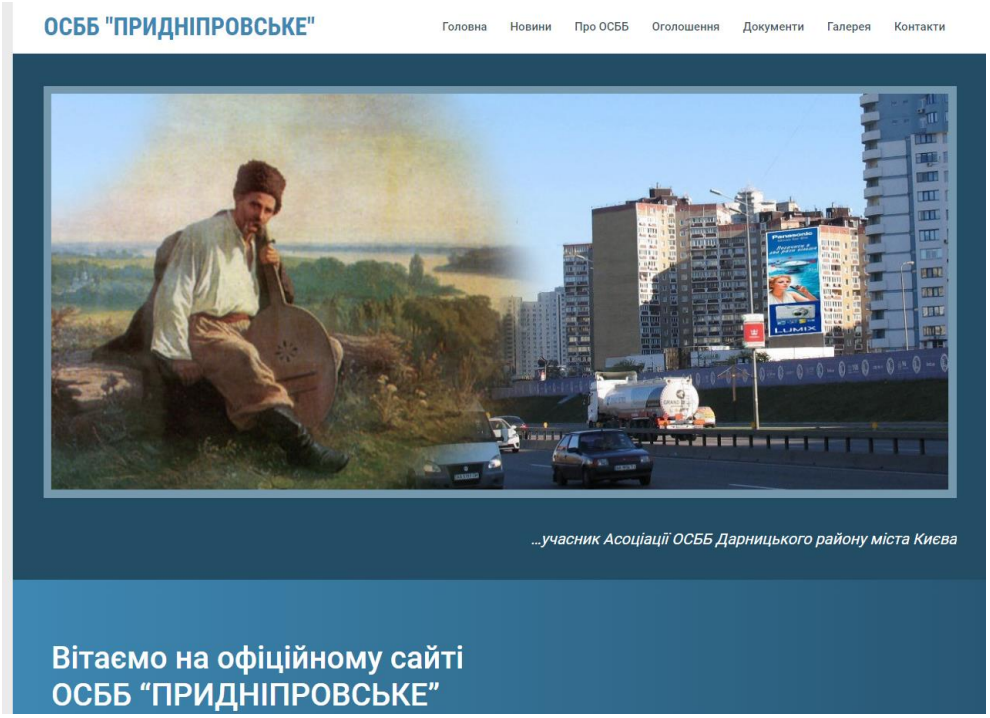


Рисунок 1.8 – Головна сторінка ОСББ «Придніпровське»

Так само, як і в минулому розглянутому аналогу в головну сторінку інтегровано блок з новинами, що є вибором замовника. Що важливіше є також блок оголошень, який є важливим рішенням для інтегрування на головну сторінку. Це дозволить швидко та без лишніх переходів на сайті користувачам отримувати найактуальнішу інформацію свого житлового будинку. Вигляд імітованої цифрової дошки оголошень зображено на рисунку 1.9.

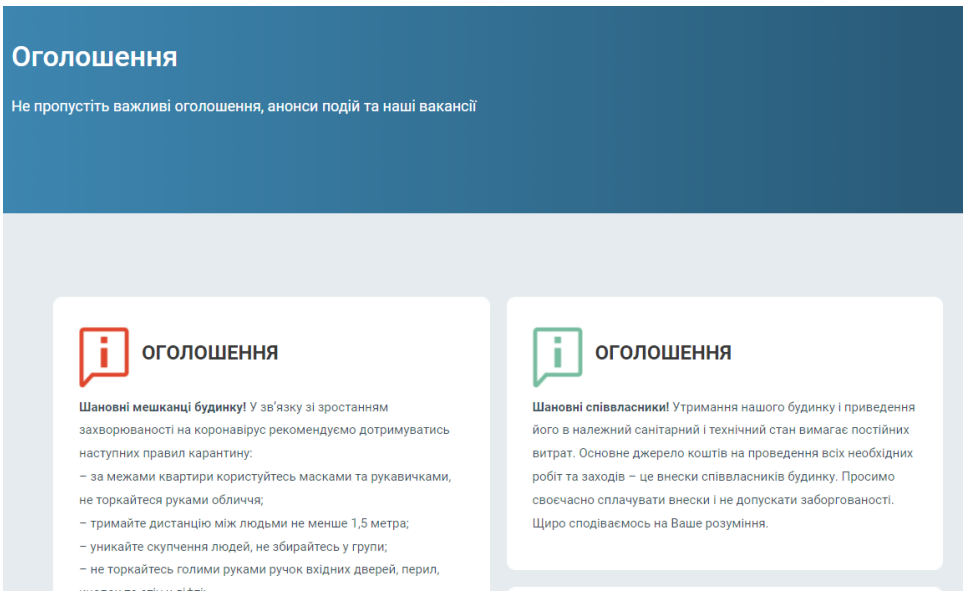


Рисунок 1.9 – Дошка оголошень на головній сторінці

Останнім важливим блоком на сторінці для розробки дипломного проєкту є Контакти. Усі подібні функції зворотного зв’язку в більшості інформаційних системах виконані подібним чином. Блоки вводу даних та кнопка відправлення повідомлення на сторону ОСББ зображено на рисунку 1.10.

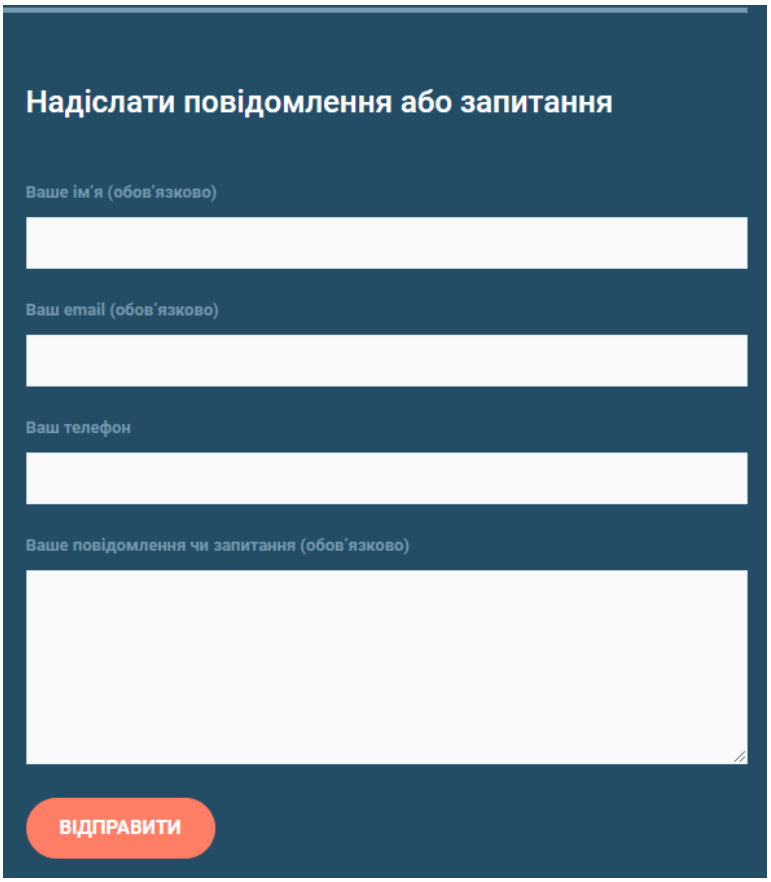


Рисунок 1.10 – Блок зворотного зв’язку

Сайт інформаційної системи ОСББ «Придніпровське» виконаний популярним виглядом одностороннього сайту, коли навігація (головне меню) лише перенаправляє на потрібне місце з відповідною інформацією. Такий метод дуже доречно підходить звичайному інформаційному сайту ОСББ, що зменшує плутанину при навігації до потрібної користувачеві інформації.

1.4 Інженерна розробка інформаційної системи

У цьому розділі подано огляд історії розробки інформаційних систем та тенденцій розвитку інформаційних систем. Також обговорюється природа методологій розробки інформаційних систем. Розділ також містить обговорення розробки інформаційних систем у динамічному середовищі.

Розвиток інформаційних систем – історія та методологія. Комп'ютеризовані інформаційні системи розробляються в різних формах понад 50 років. Розробка охоплює все: від простих додатків, розроблених для підтримки окремого процесу або функції, до сучасних додатків ERP (планування ресурсів підприємства) для всієї компанії. Ранні інформаційні системи зазвичай мали науковий або військовий характер. Коли розробка інформаційних систем стала більш регулярною ознакою в бізнес-середовищі, виникла потреба в практичних рекомендаціях. На самому початку становлення інформаційних систем вони були погано задокументовані, і програмісти, які їх створили, були єдиними, хто мав знання про те, як створювалися системи та як вони працювали. Така ситуація стала неспроможною, коли системи потрібно було підтримувати, розвивати та поєднувати з іншими системами. У міру вдосконалення технологій розробки збільшувалися можливості для розробки більш досконалих і складних систем. Потреба в структурованих методах роботи ставала все більш гострою.

Починаючи з шістдесятих років, почали шукати остаточну методологію розробки інформаційних систем і запровадити кілька сотень методологій, заснованих на дослідженнях. Методології дуже відрізняються, але в цілому вони дають рекомендації щодо контролю та координації процесу розробки та впровадження інформаційних систем. Усі бізнес-інформаційні системи мають життєвий цикл, який тягнеться від бізнес-техніко-економічного обґрунтування через розробку до використання, обслуговування та ліквідації. Методології розробки інформаційних систем мають орієнтир у так званій моделі життєвого циклу. Зазвичай основну увагу та засоби приділяють фазі розробки.

Розробка інформаційних систем, що призводить до створення нової програми, тобто експлуатація та обслуговування більш-менш розглядаються як

«щоденна робота», яку виконують не спеціалісти з розробки систем. Технічне обслуговування та різноманітні коригування можуть бути такими ж важливими для системи в довгостроковій перспективі, як і фаза розробки, але, на жаль, вона не досягла такого ж статусу загалом. Ілюстрація життєвого циклу системи наведена нижче на малюнку 1.11.

Рисунок 1.11 – Життєвий цикл системи

Етап розробки інформаційних систем загалом розглядається як складне завдання, що вимагає багато роботи та методичного забезпечення. Загальноприйнято говорити про життєвий цикл розробки системи, який часто називають каскадною моделлю. Філософія підходу Waterfall походить від інших інженерних процесів і має на меті розділити розробку системи на різні етапи, щоб полегшити процес в цілому. Кількість та імена кроків відрізняються від випадку до випадку, але загалом вони включають: техніко-економічне обґрунтування, системний аналіз, розробку систем, впровадження, перевірку, а іноді також експлуатацію та обслуговування [1]. Цей підхід і його вдосконалення, які часто носять більш ітеративний характер, стали основою для багатьох методологій розробки інформаційних систем. Ці методології можуть бути придатними для програмування та розробки програмного забезпечення, але розробка інформаційних систем не є чистою інженерною проблемою, а радше багато дисциплінарною проблемою.

Теми, що охоплюються, варіюються від соціальних наук і ділового адміністрування до інженерії. Якщо цей факт не враховувати при розробці інформаційних систем, то гарантуються різні проблеми. Як наслідок інженерної парадигми, багато інформаційних систем було «виготовлено», не задовольнивши організації, які вони мають підтримувати. Однією з класичних проблем є те, що в інформаційних системах не враховуються потреби та вимоги кількох учасників організації, що цікавить. У центрі уваги часто є види діяльності, які легко можна автоматизувати; замість того, що слід автоматизувати.

У міру розвитку методології час від часу змінювалася стратегія розробки та побудови інформаційної системи. З самого початку розвиток інформаційних систем був головним чином питанням автоматизації та раціоналізації рутин і процесів. Інформаційна система з часом стає все більш критичною проблемою в кожній організації. Оскільки інформаційна система повинна завжди підтримувати організацію, потрібно знати, як організація буде розвиватися. Це важливо для того, щоб уможливити діяльність з розвитку системи, яка дійсно сприяє організації та її прогресу. Прикладами методологій, які розглядають розробку інформаційних систем як питання процесу розвитку бізнесу, є Direct і SIM Method. Деякі методології не враховують аспекти зручності використання, а деякі методики погано підтримують розробку вимог. Спостерігається тенденція до розробки та використання окремих методів висвітлення цих аспектів. Досить звичайно також те, що використана методологія не відповідає цій проблемі. Існує три різні стратегії для вирішення цієї проблеми.

– Багатомасштабна методологія. Мова може йти про розробку базової методології для комплексних ситуацій і розробку міні-версії для менших проблем і швидкої роботи систем. Також може йти мова про розробку простої та загальної методології для «нормального» випадку. Таку методологію потім можна адаптувати до різних особливих випадків.

– Комбінація різних методологій У наборі інструментів включено декілька методологій. З цього набору інструментів вибираються різні методології з різних фаз життєвого циклу та об’єднуються в так званий ланцюжок методологій. Методології також можна об’єднати в методологічний альянс шляхом з’єднання різних методологій з однієї фази моделі життєвого циклу.

– Методології на основі компонентів Методологія складається з набору визначених компонентів. Компоненти повинні бути загальними та гнучкими, щоб їх можна було обмінювати та адаптувати до різних методологічних контекстів.

Цей розділ виконує роль підведення до безпосередньої розробки системи, тому і містить деякі деталі технічного створення інформаційних систем в цілому.

2 ЕТАП ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Створення UML діаграм

В цьому підпункті будуть представлені дві популярні UML діаграми, що створюються майже для будь-якого програмного рішення для логічного уточнення майбутнього функціоналу та структури проєкту.

UML діаграми представляють собою графічне зображення бізнес-логіки розроблюваного програмного продукту, елементи яких відрізняються відносно типу діаграми, що будується. Конкретно для поточного проєкту буде розроблено та описано діаграма використання (Use Case Diagram) для представлення логіки та діаграма класів (Class Diagram) для відображення майбутньої програмної структури [2]. Етап проєктування важливий для просування в ланцюжку повної розробки програмного продукту. Вирішено створювати лише дві діаграми, тому що по своїй логічній складовій проєкт буде досить примітивним, а велика кількість UML діаграм представляє саме цю частину. Для програмної частини достатньо однієї основної діаграми, а детальніший опис роботи алгоритмів всередині класів та їх взаємодія буде описана на етапі реалізації.

Діаграма використання зображена на рисунку 2.1.

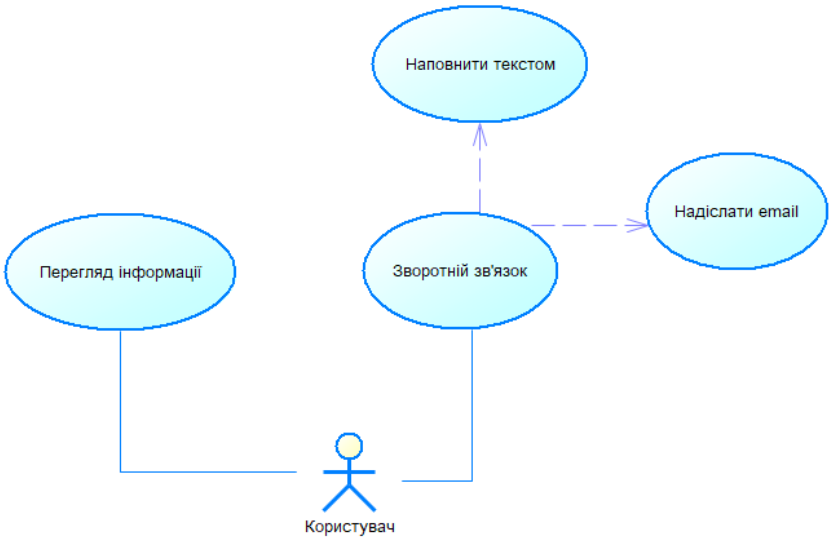


Рисунок 2.1 – Діаграма використання

В системі буде ідентифікуватись лише один дійовий персонаж – звичайний користувач, що представлений на діаграмі Користувач. Система буде нести повністю інформаційних характер без особливого інтерактивного втручання, за виключенням функції зворотного зв’язку.

Опис кейсів діаграми:

– перегляд інформації – перехід по усіх інформаційних активностях додатка та перегляд присутньої на них інформації(елементи прокрутки екрану, статична інформації, зображення і т. д.);

– зворотній зв’язок – представляє собою можливість користувача надіслати електронний лист на електронну скриньку ОСББ від свого імені вказавши тематику повідомлення та сам текст повідомлення;

– наповнити текстом – кейс залежний від варіанту Зворотній зв’язок, що передбачає собою наповнення пустих полів тексту для відправлення електронного листа для ОСББ;

– надіслати email – дія, після виконання якої відбудеться системне надсилання написаного тексту в новий відкритий додаток обраний користувачем, що підходить для надсилання електронних листів.

Діаграма класів проєкту має широку структуру та велику кількість класів по різному між собою зв’язаних. Саме так виглядають проєкти з чистою та зрозумілою для програмного інженера структурою. Розробка цього проєкту в першу чергу полягала у створені зрозумілої та стандартизованої структури відносно додатків для Android на мові програмування Kotlin з використанням архітектурної методології MVVM (Model-View-ViewModel).

Діаграма класів зображена на рисунку 2.2.

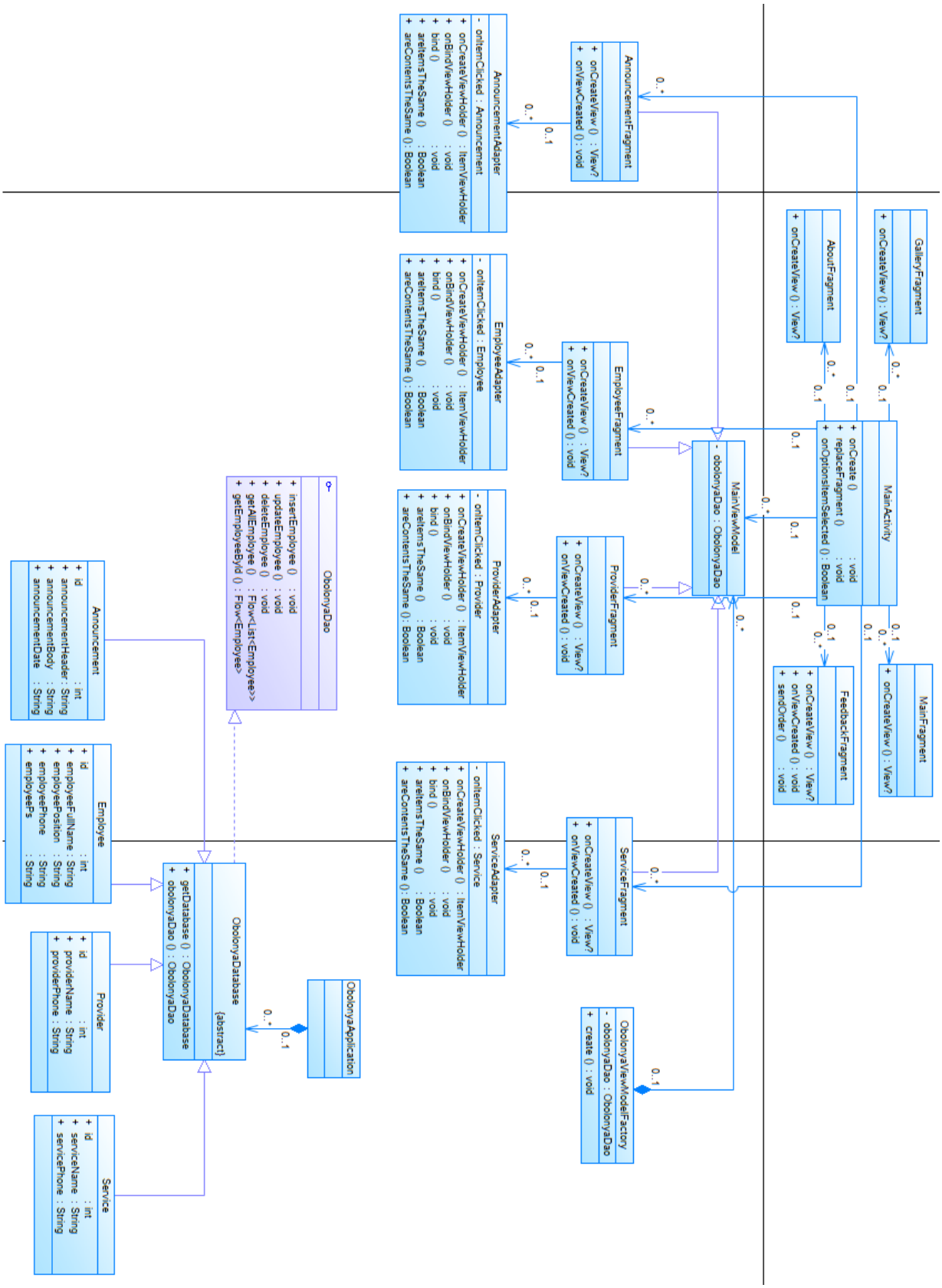


Рисунок 2.2 – Діаграма класів

Архітектура MVVM сприяє поділу розробки графічного інтерфейсу користувача за допомогою мови розмітки або коду GUI. Повною формою MVVM є Model–View–ViewModel.

Модель представлення MVVM – це перетворювач значень, що значить, що така модель перегляду відповідає за надання об’єктів даних з моделі таким чином, щоб об’єкти легко керувалися та представлялися [3].

Далі буде описана теоретична схема побудови системи на основі MVVM моделі та вказано на переваги перед подібною схемою MVC (Model View Controller). Це архітектурний шаблон, який розділяє програми на три основні логічні компоненти: Модель, Подання та Контролер. Графічна схема MVVM зображена на рисунку 2.3.

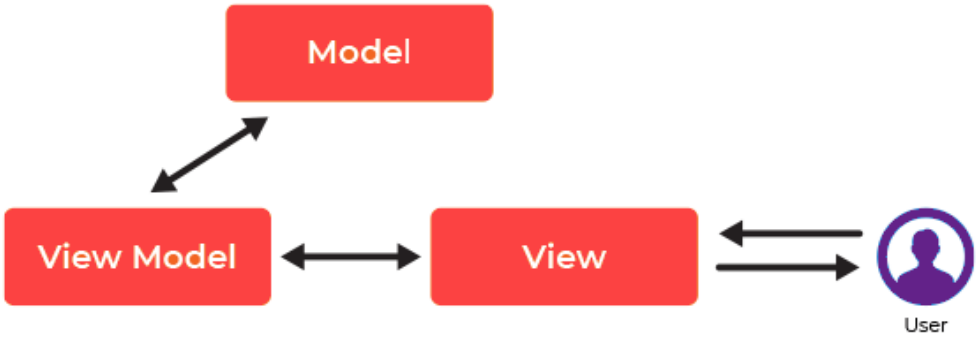


Рисунок 2.3 – Схема MVVM

Детальний опис компонентів:

– Model. Модель зберігає дані та пов’язану логіку. Вона представляє дані, які передаються між компонентами контролера або будь-якою іншою пов’язаною бізнес-логікою.

Наприклад, об’єкт отримає інформацію про учня із шкільної бази даних. Він маніпулює даними і відправляє їх назад до бази даних або використовує для відтворення тих самих даних.

– View. Представлення означає компоненти інтерфейсу користувача, такі як HTML, CSS, jQuery тощо. У MVVC подання шаблону відповідає за

відображення даних, отриманих від контролера як результат. Цей перегляд також перетворюється на модель(і) в інтерфейс користувача (UI).

– View Model. Модель представлення відповідає за представлення функцій, команд, методів, для підтримки стану View. Вона також відповідає за роботу з моделлю та активацію подій у поданні.

Перелік відмінностей від схеми MVC:

– подання є точкою входу до програми;

– відношення один до багатьох між моделлю перегляду та перегляду;

– у view є посилання на view-model;

– mvvm є відносно новою моделлю;

– процес налагодження буде складним, якщо у нас є складні прив’язки даних;

– легко для окремого модульного тестування, а код керується подіями.

Переваги MVVC:

– бізнес-логіка відокремлена від ui;

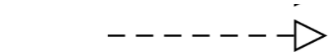
– легкий в обслуговуванні та тестуванні;

– легкі для повторного використання компонентів;

– архітектура слабкого зв’язку: mvvm робить архітектуру програми настільки ж слабо зв’язаною;

– можливість писати блокові тести, як для моделі представлення, так і для шару моделі без необхідності посилатися на view.

Зв’язки між класами застосовані при побудові UML діаграми.

 : зв'язок реалізації, на схемі відображає реалізацію іншим класом інтерфейсу;

 : зв'язок асоціації, найбільша кількість на схемі і застосовуються для відображення самого примітивного взаємозв’язку між класами;

 : зв'язок наслідування, показує, що один(підтип) з класів є приватною формою іншого класу(надтип), який називається узагальненням першого;

 : зв'язок композиції, строгий вигляд асоціації, що значить, коли клас контейнер буде видалено, то усі класи екземпляри будуть видалені разом з ним.

Опис майбутнього призначення класів проєкту:

– MainActivity – клас, який відповідає за внутрішню маршрутизацію між усіма іншими активностями програми та класами.

– GalleryFragment, MainFragment, AboutFragment, FeedbackFragment – не ведуть за собою послідовний виклик інших класів, виконують лише власне наповнення для відображення сталої текстової інформації.

– AnnouncementFragment, EmployeeFragment, ProviderFragment, ServiceFragment – подібні структурно до класів з минулого підпункту списку, проте своєю структурою посилаються на використання іншого ряду класів для відображення не сталої інформації, а інформації з бази даних.

– MainViewModel – єдиний клас типу View Model виконує структурну роль для обробки вхідної інформації та виклику потрібних класів для роботи усього додатка. Таких класів може бути і більше, це залежить від масштабів програмного продукту, що розробляється.

– ObolonyaViewModelFactory – клас на основі методу патерну Фабрика для створення копій інтерфейсу бази даних для різних класів, що будуть її використовувати.

Фабричний метод – це породжувальний патерн розробки, який визначає основний центральний інтерфейс для створення об'єктів у суперкласі, що дає змогу підкласам змінювати тип об'єктів, що будуть створені.

– AnnouncementAdapter, EmployeeAdapter, ProviderAdapter, ServiceAdapter – класи, що будуть виступати мостом між відображенням інформації для графічного інтерфейсу користувача та програмними рішеннями, що отримують та містять цю інформацію.

– ObolonyaDao – інтерфейс для роботи з базою даних програмного додатка.

– ObolonyaDatabase – клас, що структурно міститиме програмне рішення створення або отримання бази даних для додатка.

– ObolonyaApplicationi – строго залежний клас від ObolonyaDatabase та делегує цей клас для ініціалізації створення чи отримання бази даних додатка.

– Announcement, Employee, Provider, Service – ця сукупність класів виду Entity для відображення та утримання інформації з бази даних. Кожен клас відповідає за окрему таблицю бази.

2.2 Опис загальної архітектури додатка

В даному підрозділі буде детальніше описано принцип архітектурної побудови програмного рішення. Також буде наведено деякі шаблонні фрагменти коду, що в наступному розділі безпосередньої розробки адаптуються під конкретний код додатка. На самій високій точці стоїть архітектура MVVM, зараз буде фактична її реалізація програмною архітектурою [4].

Початком створення складної архітектури є створення Сутностей (Entity) для утримання інформації та використання Room (технологія управління та обробки інформації бази даних).

Створюємо новий файл класу Kotlin, який містить клас даних . Цей клас описуватиме Entity (який представляє таблицю SQLite). Кожна властивість у класі представляє стовпець у таблиці. Зрештою, Room використовуватиме ці властивості як для створення таблиці, так і для створення екземплярів об’єктів із рядків у базі даних: data class Word (val word: String).

Щоб зробити клас значущим для бази даних Room, потрібно створити асоціацію між класом і базою даних за допомогою анотацій Kotlin:

@Entity(tableName = "word_table")

class Word(@PrimaryKey @ColumnInfo(name = "word") val word: String)

Опис ролі анотацій:

– Кожен @Entity клас представляє таблицю SQLite. Додається анотація до декларації класу, щоб вказати, що це сутність.

– @PrimaryKey – кожній сутності потрібен первинний ключ. Щоб все було просто, кожне слово діє як власний первинний ключ.

– @ColumnInfo вказує ім’я стовпця в таблиці, якщо потрібно, щоб воно відрізнялося від імені змінної-члена. Це дає назву стовпцю.

Наступний етап створення правильної чистої архітектури – створення DAO (об’єкт доступу до даних). DAO має бути інтерфейсом або абстрактним класом. Room використовує DAO для створення чистого API для вашого коду. Room підтримує coroutines (сопрограми/співпрограми/корутини) Kotlin [5]. Це дозволяє анотувати запити suspend модифікатором, а потім викликати їх із співпрограми або з іншої функції призупинення. Корутини це інструмент створення кількох потоків або багатопотоковість Kotlin.

```
@Dao
interface WordDao {
  @Query("SELECT * FROM word_table ORDER BY word ASC")
  fun getAlphabetizedWords(): List<Word>
```

Далі потрібно виконати спостереження за змінами бази даних. Щоб спостерігати за змінами даних, використовується Flow. Flow Використовуємо в описі методу значення типу, що повертається, і Room генерує весь необхідний код для оновлення Flow під час оновлення бази даних.

Потік (Flow) – це асинхронна послідовність значень. Створює значення по одному (замість усіх відразу), які можуть генерувати значення з асинхронних операцій, таких як мережеві запити, виклики бази даних або інший асинхронний код.

```
@Query("SELECT * FROM word_table ORDER BY word ASC")
fun getAlphabetizedWords(): Flow<List<Word>>
```

Клас бази даних Room має бути абстрактним і розширеним RoomDatabase. Зазвичай потрібен лише один екземпляр бази даних Room для всієї програми [6]. Тому створюється RoomDatabase для створення бази даних та застосування попередніх створених класів.

Створення ViewModel. Роль ViewModel's полягає в тому, щоб надати дані інтерфейсу користувача та пережити зміни конфігурації. А ViewModel діє як комунікаційний центр між базою даних та UI. Також можна використовувати ViewModel для обміну даними між фрагментами. ViewModel є частиною бібліотеки життєвого циклу додатка.

На цьому етапі при створені ViewModel використовується інструмент LiveData – це доступний для спостереження утримувач даних, можна отримувати сповіщення щоразу, коли дані змінюються. На відміну від Flow, LiveData підтримує життєвий цикл, що означає, що він поважатиме життєвий цикл інших компонентів, таких як Activity або Fragment. LiveData автоматично зупиняє або відновлює спостереження залежно від життєвого циклу компонента, який прослуховує зміни. Це робить LiveData ідеальним компонентом для змінних даних, які буде використовувати або відображати інтерфейс користувача.

```
val allWords: LiveData<List<Word>> = DAO.allWords.asLiveData()
```

Далі потрібно додати макет XML для списку та елементів. XML макети в структурі програмного додатка Kotlin призначені для відображення графічного інтерфейсу користувача з усіма даними та косметичними елементами.

Дані будуть відображатись у форматі RecyclerView (програмний елемент списку), що трохи приємніше, ніж просто кидати дані в файл TextView (програмний елемент відображення звичайного тексту).

На цьому етапі потрібно буде створити: ListAdapter – клас для передачі інформації з бази даних у графічний інтерфейс).

Потрібно лише один екземпляр бази даних і сховища у програмі, а не багаторазове його створення при необхідності отримання даних з бази даних. Простий спосіб досягти цього – створити їх як членів Application класу. Тоді вони будуть просто витягуватися з програми, коли вони потрібні, а не створюватися щоразу.

```
class WordsApplication : Application() {
  val database by lazy { RoomDatabase.getDatabase(this) }
}
```

Далі ці всі класи та взаємозв’язки потрібно привести в дію та використати [7]. Для цього створюється Activity або активність, що буде використовувати ViewModel та ListAdapter, а ці два класи в свою чергу застосують і інші.

Таким чином закінчується побудова архітектури. В результаті застосовані усі складові (база даних, програмні обробки даних, відображення у графічному інтерфейсі користувача).

2.3 Конструювання та опис бази даних

База даних для цього проєкту матиме зрозумілу та логічну назву obolonya_database. Вона буде представляти собою просте сховище даних для майбутнього її зв’язування з серверною частиною, якщо це буде потрібно.

Для попереднього наповнення та створення бази даних SQLite (спрощена версія реляційної бази даних для мобільної платформи Android). Хоч сама база даних є реляційною побудова зав'язків між таблицями не є потрібним для реалізації програмного продукту. Но застосовано принцип унікального ключа кожної таблиці PrimaryKey та деякі інші деталі табличної структури. SQLite неможливо відкривати всередині середовища для повноцінного її перетворення, тому було використано популярну для цієї цілі програму DB Browser for SQLite (рисунок 2.4).

Ім'я	Тип	Схема
Таблиці (5)		
announcement		CREATE TABLE "announcement" ("id" INTEGER NOT NULL UNIQUE, "header" T
id	INTEGER	"id" INTEGER NOT NULL UNIQUE
header	TEXT	"header" TEXT NOT NULL
body	TEXT	"body" TEXT NOT NULL
date	TEXT	"date" TEXT NOT NULL
employee		CREATE TABLE "employee" ("id" INTEGER NOT NULL UNIQUE, "fullName" TEX
id	INTEGER	"id" INTEGER NOT NULL UNIQUE
fullName	TEXT	"fullName" TEXT NOT NULL
position	TEXT	"position" TEXT NOT NULL
phone	TEXT	"phone" TEXT NOT NULL
ps	TEXT	"ps" TEXT NOT NULL
provider		CREATE TABLE "provider" ("id" INTEGER NOT NULL UNIQUE, "name" TEXT NO
id	INTEGER	"id" INTEGER NOT NULL UNIQUE
name	TEXT	"name" TEXT NOT NULL
phone	TEXT	"phone" TEXT NOT NULL
service		CREATE TABLE "service" ("id" INTEGER NOT NULL UNIQUE, "name" TEXT NOT
id	INTEGER	"id" INTEGER NOT NULL UNIQUE
name	TEXT	"name" TEXT NOT NULL
phone	TEXT	"phone" TEXT NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

Рисунок 2.4 – Вигляд інтерфейсу DB Browser for SQLite

Спроєктовані таблиці бази даних obolonya_database за допомогою зручного інтерфейсу автоматично генерують SQL код для зручності та повторного використання в інших подібних програмах.

Було спроєктовано 3 таблиці:

– employee – таблиця для збереження інформації про працівників та персонал ОСББ;

– provider – таблиця для збереження інформації про провайдерів, що обслуговують будинок ОСББ та надають послуги проведення та налаштування кабельного інтернету та телебачення;

– service – таблиця для збереження інформації про різні аварійні служби, служби обслуговування різних систем, що може пригодитись жителям ОСББ.

Структура таблиць, назви полів, типи даних для полів та деяка інша інформація про структуру таблиць відображена на рисунку 2.4 та не потребує додаткового опису.

3 СТВОРЕННЯ ПРОГРАМНОГО КОДУ ДОДАТКА

3.1 Програмні рішення бізнес-логіки

В цьому підрозділі будуть представленні більш розгорнуто окремі важливі елементи архітектури та бізнес-логіки внутрішньої роботи програмного додатка. Розберемо класи та їх особливості з сторони програмного коду.

Для розробки проєкту обрано відносно нову та сучасну об’єктивно-орієнтовану мову програмування для мобільних додатків Android Kotlin. Детальний опис завжди можна знайти та ознайомитись на просторах інтернету. Для проєкту важливо, що це досить перспективна мова програмування для мобільних додатків та використовується провідними компаніями для розробки власних продуктів.

В якості середовища розробки обрано Android Studio найновішої версії Bumblebee. Це середовище є найпопулярнішим для розробки додатків на Android [8]. Пропонує широкий вибір мов для розробки, велику кількість готових шаблонів для графічного інтерфейсу та модулів, що можна вмонтувати в додаток. Також важливим аспектом є зручний та якісний емулятор смартфону, в який можна встановити велику кількість систем на вибір. Функціональність такого середовища вражає розробників і високо ціниться.

При розробці такої великої кількості класів, файлів та інтерфейсів важливим є правильна та зрозуміла структура файлів всередині проєкту, вигляд файлової структури додатка для ОСББ зображено на рисунку 3.1.

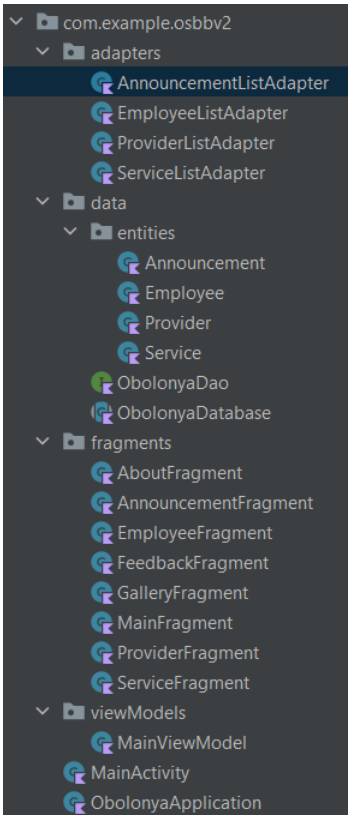


Рисунок 3.1 – Файлова структура додатка

Усі класи поділені по відповідних папках, що логічно відображає їх роль в проєкті. Опишемо класи, які входять у відповідні групи (папки), що дасть розуміння їх значень в проєкті.

Перша група це adapters містять адаптери, що передають інформацію між базою даних та фрагментами, які в свою чергу передають ці дані на графічний інтерфейс користувача. Кожен з чотирьох класів має подібну структуру з однаковими функціями:

– onCreateViewHolder – при створені утримувача даних ViewHolder повертає елемент, що ці дані містить (лістинг 3.1).

Лістинг 3.1 – Код функції onCreateViewHolder

```
return ItemViewHolder(
    ServiceListItemBinding.inflate(
        LayoutInflater.from(
            parent.context
        )
    )
)
```

– onBindViewHolder – при визначенні елемента та його даних закріпляє ці дані за утримувачем, який буде передавати їх в інші функції (лістинг 3.2).

Лістинг 3.2 – Код функції onBindViewHolder

```
val current = getItem(position)
holder.itemView.setOnClickListener {
    onItemClicked(current)
}
holder.bind(current)
```

– ItemViewHolder – клас всередині класу для отримання передачі даних утримувача у фрагмент, що буде вже відображати їх на графічний інтерфейс.

Лістинг 3.3 – Код функції ItemViewHolder

```
RecyclerView.ViewHolder(binding.root) {

fun bind(employee: Service) {
    binding.itemServiceName.text = employee.serviceName
    binding.itemServicePhone.text = employee.servicePhone
}
```

Для усіх адаптерів відрізняється ділянка коду функція bind, для різних таблиць різна кількість елементів та різна їх назва відповідно.

Цілісна структура одного з адаптерів буде подана в Додатка А.

Наступна група entities – сутності, що будуть виступати контейнерами для кожного запису таблиці з бази даних для оперування ним у додатка [9]. Для кожного запису створюється окремий екземпляр відповідної сутності. Усі сутності побудовані за одною структурою, буде наведено для прикладу сутність announcement (лістинг 3.4).

Лістинг 3.4 – Код сутності announcement

```
@Entity(tableName = "announcement")
data class Announcement (
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    @NonNull @ColumnInfo(name = "header")
    val announcementHeader: String,
    @NonNull @ColumnInfo(name = "body")
    val announcementBody: String,
    @NonNull @ColumnInfo(name = "date")
    val announcementDate: String,
)
```

У усіх інший сутностей буде змінюватись параметр tableName на відповідну таблицю та кількість і вид змінних, що оголошуються за допомогою ключового слова val.

В групі database відповідно міститься вищеописана підгрупа entities та дві файли, що відповідають за частину функціонування прямого контакту з базою даних.

Інтерфейс ObolonyaDao з групи database виконує прямі звернення до бази даних у вигляді готових функцій та SQL-запитів. Для кожної таблиці передбачено необхідна кількість запитів та функцій для роботи додатка. Лістинг 3.5 містить код цих елементів для таблиці employee. Для інших таблиць точно такі ж елементи інтерфейсу з відповідно зміненими назвами таблиці, яку вони будуть обслуговувати.

Лістинг 3.5 – Код частини елементів ObolonyaDao

```
@Insert(onConflict = OnConflictStrategy.IGNORE)
suspend fun insertEmployee(employee: Employee)

@Update
suspend fun updateEmployee(employee: Employee)

@Delete
suspend fun deleteEmployee(employee: Employee)

@Query("SELECT * from employee")
fun getAllEmployee(): Flow<List<Employee>>

@Query("SELECT * from employee WHERE id = :id")
fun getEmployeeById(id: Int): Flow<Employee>
```

Ключові слова позначені символом @ називаються анотацією. Кожна анотація відповідає певній операції над базою даних.

Файл ObolonyaDatabase виступає прямою реалізаціє RoomDatabase та виконує роль створення бази даних з файлу obolonya_database, що був попередньо конструйований у програмі DB Browser for SQLite. Виконує лише одну функцію getDatabase – отримання бази даних з файлу, якщо він існує або створення нового файлу бази даних.

Група fragments містить частини класів бізнес-логіки для приймання та передавання даних до графічного інтерфейсу. Використовує адаптер та сутності відповідної таблиці для роботи. Містять наступні функції:

– onCreateView – при створенні екземпляру класу відбувається прив’язка елементів відображення інформації з графічного інтерфейсу та програмних змінних з сторони бізнес-логіки(лістинг 3.6).

Лістинг 3.6 – Код функції onCreateView фрагменту EmployeeFragment

```
inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    _binding = FragmentEmployeeBinding.inflate(inflater, container, false)
    return binding.root
```

– onViewCreated – по закінчені створення екземпляру класу виконує передачу інформації на графічний інтерфейс. Використовує для роботи адаптер та ViewModel (лістинг 3.7).

Лістинг 3.7 – Код функції onViewCreated фрагменту EmployeeFragment

```
super.onViewCreated(view, savedInstanceState)

val adapter = EmployeeListAdapter {
}

binding.employeeRecyclerView.layoutManager =
LinearLayoutManager(this.context)
binding.employeeRecyclerView.adapter = adapter

viewModel.allEmployee.observe(this.viewLifecycleOwner) { items ->
    items.let {
        adapter.submitList(it)
    }
}
```

Повністю код фрагменту буде представлено в додатка А.

Остання група viewModels виступає основним зв’язуючим класом класів по отриманню та оброці інформації з класами, що будуть її відображати на графічному інтерфейсі. Елемент, що виконує передачу інформації (лістинг 3.8)

Лістинг 3.8 – Код елементу передачі інформації класу MainViewModels

```
val allAnnouncement: LiveData<List<Announcement>> =
obolonyaDao.getAllAnnouncement().asLiveData()
```

Також містить успішно реалізований патерн Фабрика (лістинг 3.9).

Лістинг 3.9 – Код патерну Фабрика (Factory)

```
class ObolonyaViewModelFactory(private val obolonyaDao: ObolonyaDao) :
ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(MainViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return MainViewModel(obolonyaDao) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }}
```

Також є два класи, які не належать до створених груп, а знаходять в головній папці проєкту та є глобальними та основними в роботі всього додатка – MainActivity і ObolonyaApplication.

MainActivity виконує роль маршрутизації та запуску кожного можливого екрану в додатка та кожної програмної активності(лістинг 3.10). В цьому класі реалізоване бічне меню [10]. Повний код цього класу буде наведено в Додатка А.

Лістинг 3.10 – Код маршрутизації по активностям додатка

```
when(it.itemId){
    R.id.nav_main -> replaceFragment(MainFragment(),
it.title.toString())
    R.id.nav_about -> replaceFragment(AboutFragment(),
it.title.toString())
    R.id.nav_employee -> replaceFragment(EmployeeFragment(),
it.title.toString())
    R.id.nav_providers -> replaceFragment(ProviderFragment(),
it.title.toString())
    R.id.nav_announcement -> replaceFragment(AnnouncementFragment(),
it.title.toString())
    R.id.nav_gallery -> replaceFragment(GalleryFragment(),
it.title.toString())
    R.id.nav_service -> replaceFragment(ServiceFragment(),
it.title.toString())
    R.id.nav_feedback -> replaceFragment(FeedbackFragment(),
it.title.toString())
}
```

ObolonyaApplication відповідає за одноразове створення бази даних (лістинг 3.11), що запобігатиме повторному її повторному створенню, яке призведе до фатальної помилки додатка та припинення його роботи.

Лістинг 3.11 – Єдине створення бази даних для додатка

```
val database: ObolonyaDatabase by lazy {
ObolonyaDatabase.getDatabase(this) }
```

На цьому закінчуються класи програмної бізнес-логіки.

3.2 Програмні рішення графічного інтерфейсу користувача

Відображення графіки та будь-чого на екрані смартфону відбувається за допомогою XML файлів, які містять різні структури (теги) для передавання інформації користувачу.

XML означає Extensible Markup Language. XML – це мова розмітки, схожа на HTML, що використовується для опису даних. Xml сам по собі добре читаються як людиною, так і машиною. Крім того, він масштабований і простий у розробці. В Android використовується xml для розробки макетів різних екранів додатка.

Вся концепція інтерфейсу користувача Android визначається за допомогою ієрархії об’єктів View і ViewGroup. ViewGroup – це невидимий контейнер, який організовує дочірні елементи [11]. Ці дочірні елементи є іншими тегами, які використовуються для створення різних частин інтерфейсу користувача. Одна ViewGroup може мати іншу ViewGroup як дочірній елемент, як показано на рисунку 3.2.

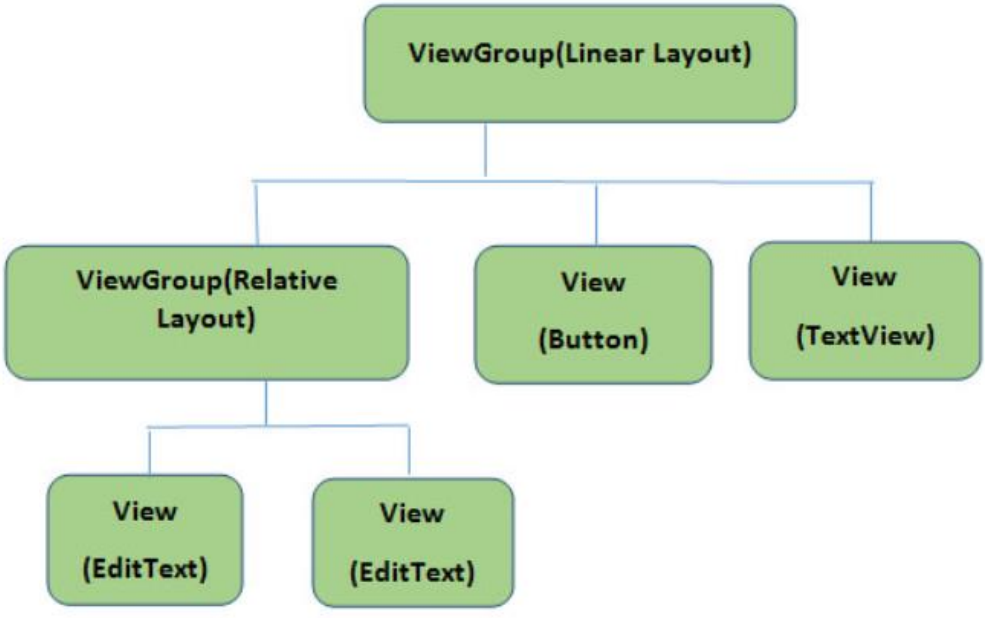


Рисунок 3.2 – Часто використовувана ієрархія макету XML

Для даного проєкту було створено в загальній кількості 32 XML файлів створених з нуля для відображення інформації та необхідних структур додатка. Також в проєкті по замовчуванню присутні елементи, що містять програмні інструкції глобального використання стрічок, кольорів, стилі, об’єднанні конфігурації окремих тегів файлів, теми та ще немало різних конфігурацій.

Важливою структурою є бічне меню, що складається з двох xml файлів nav_munu та nav_header. Перша частина це основне тіло меню, на якому розташовані всі елементи маршрутизації (лістинг 3.1).

Лістинг 3.1 – Код макету частини меню з елементами маршрутизації

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

    <group android:checkableBehavior="single">
    <item
        android:id="@+id/nav_main"
        android:title="@string/nav_menu_main"
        android:icon="@drawable/ic_main"/>
    <item
        android:id="@+id/nav_about"
        android:title="@string/nav_menu_about"
        android:icon="@drawable/ic_about"/>
    <item
        android:id="@+id/nav_employee"
        android:title="@string/nav_menu_employee"
        android:icon="@drawable/ic_employees"/>
<item android:title="Зворотній зв'язок">
        <menu>
            <item
                android:id="@+id/nav_feedback"
                android:icon="@drawable/ic_feedback"
                android:title="Надіслати Email" />
        </menu>
    </item>
</menu>
```

Інша частина це заголовок меню, на якому розташований фон цього заголовку та логотип додатка. Вигляд усіх можливих варіантів відображення макетів на екрані буде відображено у розділі тестування.

Макет fragment_main містить назву ОСББ його логотип та вступна інформація. У створенні більшості макетів використано розділювач елементів у вигляді тегу TextView (відображення тексту), в якого залитий задній план відповідним кольором та заокруглені кути. Цей фрагмент містить лише теги TextView. Подібні прості макети це – fragment_about та fragment_gallery (додатково відображає ще і картинки за допомогою тегу ImageView)

Є дві категорії, на які можна розділити присутні розроблені макети додатка – макет відображення списку елементів(записів якоїсь однієї з таблиць) та макет єдиного елемента з цього списку.

До першої категорії відносяться: fragment_employee, fragment_provider, fragment_service, fragment_announcement. Кожен це макет використовує тег RecyclerView(представляє собою список відображення елементів).

До другої групи відносяться: employee_list_item, provider_list_item, service _list_item, announcement _list_item. Такі елементи будуть відображати інформації окремого запису певної таблиці [12]. Використовуються теги тексту, відображення картинки та елемент розділювач, що описаний вище. Групуючим елементом виступає CardView (зручний елемент для відображення окремих розділених елементів). Код provider_list_item представлено у лістингу 3.2.

Лістинг 3.2 – Структура макету відображення запису таблиці provider

```
<androidx.constraintlayout.widget.ConstraintLayout
    <ImageView
    <TextView/>
    <TextView/>
    <TextView/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

4 ПЕРЕВІРКА РОБОТИ ДОДАТКА

4.1 Розгортання програмного продукту

Для функціонування додатка на мобільний пристрій потрібно завантажити та встановити Android apk файл надавши йому потрібні дозволи. Встановлення Android-додатка є стандартною процедурою для користувачів мобільних пристроїв із операційною системою Android та не потребує спеціальних знань чи вмінь.

Розроблений додаток буде повністю сумісний із усіма пристроями починаючи із SDK 16, що відповідає версії 4.1.2 операційної системи Android. На пристроях із більш ранім SDK додаток не працюватиме.

4.2 Ручне тестування додатка за допомогою емулятора

Тестування Android-додатка проводилось вручну на емуляторі пристрою Pixel 2 XL API 29 із версією Android 10. Під час тестування було перевірено коректність розміщення усіх елементів інтерфейсу та їх відповідність заданим стилям й призначеному функціоналу.

Створення емулятора зображено на рисунку 4.1.

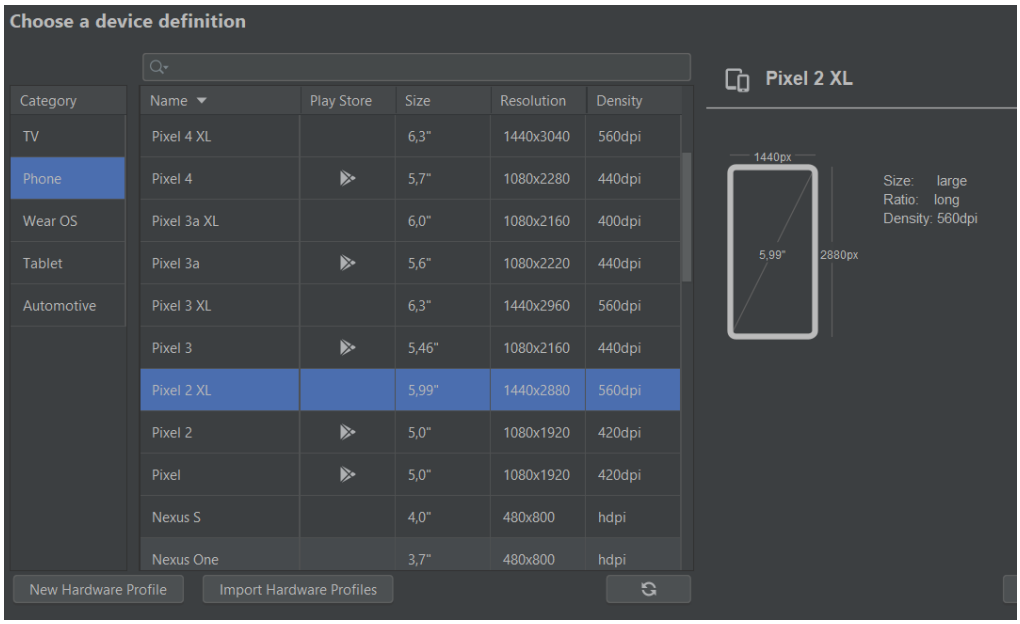


Рисунок 4.1 – Процес створення емулятора для тестування

Після встановлення додатка на емулятор запускаємо його. При відкритті відображається головна і одночасно початкова активність з ознайомчою інформацією (рисунок 4.2).

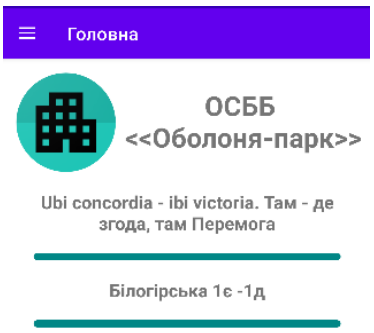


Рисунок 4.2 – Головна сторінка додатка

При запуску перебуваючи на головній активності користувач одразу скористається відкриттям бічного меню для ознайомлення з доступними пунктами маршрутизації. Вигляд відкритого бічного меню зображено на рисунку 4.3.

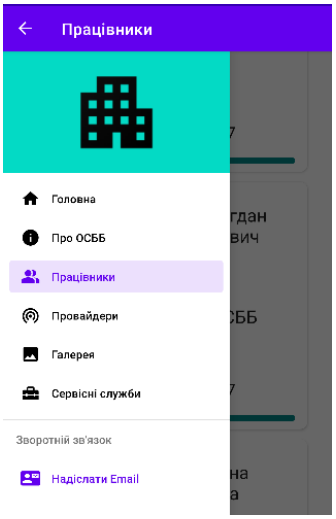


Рисунок 4.3 – Бічне меню додатка

Далі буде перевірка роботи та коректність відображення кожного пункту маршрутизації.

Переходимо на екран Про ОСББ (рисунок 4.4)

Рисунок 4.4 – Активність Про ОСББ

По першому переходу одразу стає зрозумілим правильність роботи декількох функцій програмного коду. Бічне меню після переходу автоматично ховається, щоб не закривати контент обраної активності. Також заголовок активності успішно змінюється відображаючи свою назву. Переходимо до наступного пункту маршрутизації – Працівники (рисунок 4.5).

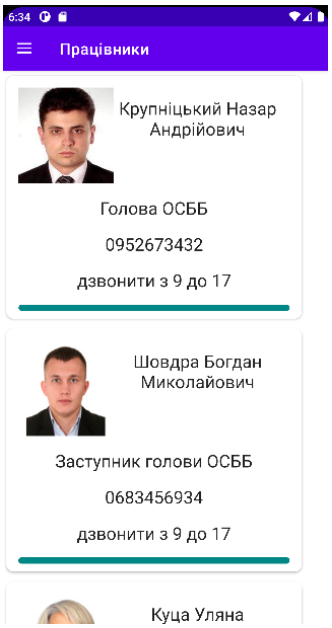


Рисунок 4.5 – Активність Працівники

Відкрита активність працює правильно, записи бази даних відповідної таблиці відображаються в окремих елементах списку, що в свою чергу коректно скролиться. Наступна активність для тестування – Провайдери (рисунок 4.6).

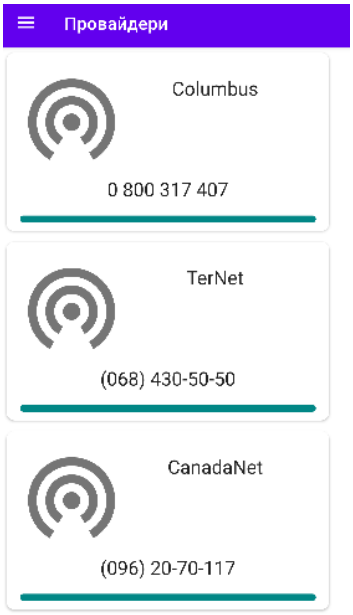


Рисунок 4.6 – Активність Провайдери

Записи таблиці провайдерів з бази даних відображаються правильним чином, також активність Сервісні служби та Галерея працює так само добре. Заголовок поточної активності змінюється на поточну. Перевіримо важливу функцію надсилання електронного листа та протестуємо коректність роботи перенесення введених даних на інший додаток для надсилання повідомлення. Вигляд активності Надіслати Email зображено на рисунку 4.7.

Тестування виявило несправність неправильного перенесення текстової інформацію з введених полів для надсилання електронного листа у додаток для обробки цієї інформації. Цю несправність успішно усунуто. В інших місцях додаток працює, як і було заплановано.

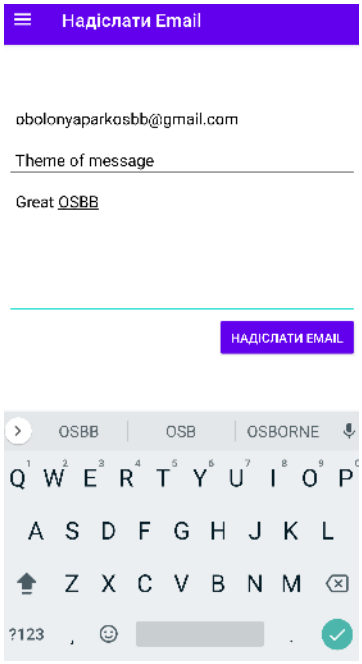


Рисунок 4.7 – Активність Надіслати Email

Після внесення теми листа та його тексту (електронна адреса отримувача встановлена по замовчуванню) натискаємо на кнопку для початку надсилання. Буде запропоновано усі можливі додатки для надсилання електронного листа. Для тестування обрано Gmail. Вигляд листа в Gmail зображено на рисунку 4.8.

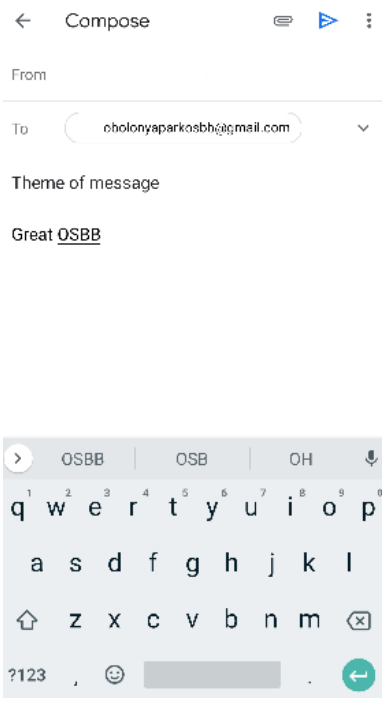


Рисунок 4.8 – Перенесений текст з додатку ОСББ

5 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

5.1 Аналіз ринку

Метою техніко-економічного розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності програмного засобу оптимізації роботи куратора групи та прийняття рішення про його подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки. Для проведення даного дослідження необхідно провести ряд розрахунків.

Характеристика програмного продукту:

Програмний продукт розроблений на найновішій постійно набираючій популярності та об’єктно-орієнтованій мові програмування Kotlin. Додаток розроблено під мобільну операційну систему Android, яка є найпопулярнішою в світі серед мобільних операційних систем. Android додатки пишуться на Kotlin/Kava/C# та декількох інших мовах. В Android інструменти SDK компілюють всі коди програми з будь-якими даними та ресурсами в файли APK: Android-пакет, який є архівним файлом з .apk суфіксом. Один АПК файл містить весь Android додаток і файл для декларацій пристроїв, які потребує програма.

Ринок збуту такого спеціалізованого програмного продукту не є великим та визначений наперед. Тобто проєкт розроблявся спеціально для конкретного ОСББ, що не вимагає власноруч виконувати пошук споживача. Якщо розглядати тип продукту – інформаційна система, то попит на такі мобільні додатки був завжди і є тепер. Тип додатку інформаційна система є фактично універсальним для будь-якої сфери діяльності, тому проблем з ринком збуту не передбачається.

Далі перераховано характеристики експлуатації програмного продукту:

– для користування продуктом потрібний мінімум: смартфон мінімальної продуктивності, мобільна операційна система Android, встановлений додаток ОСББ «Оболоня-парк»;

– розроблений продукт є інформаційною системою з можливістю зворотного зв’язку;

– мінімальні можливості для клієнта продукту є можливість ознайомитись з інформацією про ОСББ;

– потенційним замовником продукту вважається користувач смартфону на основі операційної системи Android з підключенням до інтернету;

– основним ринком реалізації продукту може виступати PlayMarket. В якості додаткових ринків реалізації продукту можуть виступати різноманітні тематичні форуми та веб-сайти. Фактично додаток буде розміщеним на PlayMarket для вільного завантаження;

– продукт реалізовано на некомерційній основі і буде знаходитись у вільному доступі абсолютно безкоштовно;

– сервісне обслуговування буде організовано для початку єдиним адміністратором інформаційного простору продукту, який буде мати усі доступні права доступу.

5.2 Розрахунок витрат на проєктування

Зведені розрахунки витрат наведено у таблиці 5.1.

Таблиця 5.1 – Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Факт. витрачено матеріалів	Ціна за одиницю, грн.	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1.	Допоміжна література	1	600	600	60	660
2.	Папір (формат А4)	2	80	160	16	176
3.	Ручка кулькова	2	10	20	2	22
4.	Олівець простий	2	10	20	2	22
5.	Диски CD-R	2	15	30	3	33

Продовження таблиці 5.1

6.	Зошит, 96 арк.	1	50	50	5	55
7.	Тонер для принтера	1	90	90	9	99
8.	Канцелярські маркери (червоний, зелений)	2	20	40	4	44
Разом						1111,00

З кожним роком сучасні технології розвиваються, розробка мобільних додатків також не стоїть на місці. Їх реалізація сприяє покращенню комунікабельності, якості спілкування, вирішення різноманітних складних завдань. Хорошим професіоналам це приносить досить високий прибуток. Заробітна плата учасників проєктування проєкту наведено у таблиці 5.2.

Таблиця 5.2 – Розрахунок заробітної плати проєктувальників

№ п/п	Посада виконавця	Оклад, грн/міс	Відрахування грн/міс	Кількість		Сума з/п, грн.
				чол.	місяців	
1	Технік-програміст	5200	950,5	2	6	4249,5
2	Тестувальник	4600	910	1	6	3690
Усього зарплати:						175122

Загальний кошторис витрат протягом усього проєктування та по його завершенні наведено у таблиці 5.3.

Таблиця 5.3 – Кошторис витрат

Найменування статей витрат	Сума, грн	Обґрунтування
1. Зарплата проєктувальників.	175122	При розробці мобільного додатку потрібно розробити великий функціонал і відповідно протестувати його
2. Відрахування на соціальні потреби.	38526	22% від суми зарплат
3. Контрагентські роботи і послуги.	480	Прибирання приміщення
4. Витрати на відрядження.	870	Похід усієї команди в кінотеатр
5. Інші прямі витрати.	1111	Придбання канцелярських товарів, паперу, маркерів.

Продовження таблиці 5.3

6. Усього прямих витрат.	216109	175122+38526+480+870+1111= 216109
7. Накладні витрати.	4400	25% - витрати на опалення, електроенергію, купівлю комплектуючих для ремонту комп'ютерів
8. Планові накопичення.	3670	Преміювання працівників за хорошу роботу та якісне виконання поставлених задач (20%).
9. Усього, кошторисна вартість проєкту.	224179	216109+4400+3670= 226768
10. Податок на додану вартість.	45353,6	20% від усього кошторису
11. Загалом, договірна ціна розробки З/П.	269532,6	224179+45353,6= 269532,6

5.3 Обґрунтування необхідності розробки

Попит на мобільні телефони неухильно зростає. І це дає підставу стверджувати про актуальність і доцільність процесу розробки мобільних додатків. Головне завдання творців продукту – оцінити свою цільову аудиторію, для кого він створюється, так як лише корисна розробка зможе отримати справжнє визнання з боку користувачів.

Розробка мобільного додатку, який фактично існуватиме як просто інформаційна система, не є особливо важливою для просування на ринку мобільних додатків.

Подібних програмних додатків для мобільних пристроїв в українському сегменті майже не зустрічається, що дає шанс для розробленого додатку на успіх та надає деяку важливість. На сьогоднішній день більшість учасників ОСББ будуть приємно здивовані можливості отримання інформації за допомогою мобільного додатку.

Таким чином, варто визнати, що майбутнє ПК це портативна, легка та функціональна техніка – планшети, електронні книжки, нетбуки і смартфони – і все це в основному працює саме на операційній системі Android.

Згідно зі статистикою, саме Android лідирує зараз на ринку смартфонів, займаючи на ньому більше половини всього обсягу продажів. Таким чином навчання розробці додатків під Android на сьогоднішній день украй актуально, так як кількість спеціалістів менша ніж попит на них.

В межах економічного обґрунтування, важливо описати можливості та значення в цілому мобільної розробки для ведення бізнесу. Що важливо враховувати при використанні мобільних додатків для підтримування конкуренція на власному ринку.

Завдяки поширенню та ескалації технологій ринок мобільних додатків зріс по днях, особливо за останнє десятиліття. Подальші оцінки досліджень показують, що розмір ринку мобільних додатків досягне 407,31 мільярда доларів до 2026 року. Таким чином, якщо є деякі популярні ідеї додатків, то це найкращий час для інвестування в розробку мобільних додатків.

Оскільки ринок мобільних додатків дуже динамічний, йти в ногу з тенденціями, що змінюються, стає дуже важливим. Для цього дуже важливим фактором є глибоке дослідження ринку перед розробкою мобільного додатка. Важливість дослідження ринку мобільних додатків можна зрозуміти з того факту, що воно дає компанії цінну інформацію про своїх конкурентів. Крім того, це дає вам чітке уявлення про ваші власні сильні та слабкі сторони.

Ефективність мобільних додатків у сучасну епоху значною мірою залежить від закулісної дослідницької роботи. Багато досліджень показали, що мобільні додатки, розроблені без відповідних досліджень, вмирають передчасною смертю в магазині додатків. Статично, близько 72% майбутніх мобільних додатків не можуть залишити свій слід на ринку через погану дослідницьку роботу.

Далі буде обговорено, чому необхідні дослідження ринку для мобільного додатка та різницю, яку це може внести в загальну рентабельність інвестицій вашого бізнесу.

Бізнес розробки мобільних додатків дуже динамічний і залежить виключно від мінливих потреб і побажань клієнтів. Дослідження ринку є дуже важливим для створення мобільного додатка . Таким чином, підприємства повинні залишатися на вершині своєї гри, оскільки знають останні тенденції ринку.

Через його постійно змінюваний характер розробка мобільного додатка є дуже міцним горіхом. А без належного дослідження ринку весь процес може стати безнадійним за мить. Компанії не знають, хто є їх цільовою аудиторією, який ринок найкраще підходить для їхнього мобільного додатка, і, головне, набір функцій, які необхідно розділити, а які функції будуть додані пізніше як додаткові .

Крім того, повне дослідження ринку також допомагає утримувати проєкт розробки програми в межах бюджету. Це також допомагає маркетинговій команді придумати унікальні ідеї та підвищити популярність мобільного додатка.

Детальний аналіз ринку для вашого бізнес-додатка дасть цінну інформацію та убереже від жахливих помилок. Більше того, якщо можна зрозуміти болючі точки клієнтів, не буде заповнюватись мобільний додаток непотрібними функціями.

Зі збільшенням кількості доступних опцій рівень терпіння користувачів швидко падає. Вони миттєво відкинуть мобільний додаток, якщо він змушує їх чекати на виконання важливих завдань.

Глибоке дослідження ринку також допоможе стати піонером у своїй галузі, використовуючи мобільні додатки як платформу, щоб допомогти бізнесу досягти ще більших висот.

Переваги дослідження ринку.

Переваги, які підприємства отримують від дослідження ринку для розробки передового мобільного додатка, величезні. Нижче перераховано деякі з основних переваг:

– Швидший збір даних. Дослідження ринку мобільних додатків дозволяє швидше збирати дані. Це одна з важливих відповідей на запитання:

«чому дослідження ринку додатків рекомендується як обов'язкова стратегія?» Оскільки клієнти нового віку використовують свої смартфони частіше, ніж будь-який інший пристрій, отримати швидші відповіді через додаток легше.

– Кращі інсайди. Під час поглибленого дослідження знання не обмежуються лише текстовими запитаннями. Ви можете оцінити поведінку клієнта з різних точок, таких як платформа соціальних медіа, фотографії, аудіо, відео тощо. Дослідження ринку дає вам більш різноманітний набір даних для подальшого аналізу.

– Підвищує цінність бренду. Потреби та бажання споживачів постійно змінюються. Таким чином, підприємствам стає важко знайти свою основну аудиторію, на яку націлюватимуться, або які сегменти ринку будуть більше зацікавлені в їхньому мобільному додатку. Дослідження ринку допомагає зміцнити відносини між клієнтом і брендом і в кінцевому підсумку підвищити цінність бренду.

– Кращі результати. Завдяки стоїчному дослідженню ринку мобільних додатків ви можете підвищити зручність використання свого мобільного додатка. Різні дослідження ринку показують, що 80% користувачів смартфонів перевіряють свій телефон протягом 15 хвилин після пробудження. Таким чином, чим більше ви взаємодієте зі своїми клієнтами в мобільному додатку, тим більше шансів на розвиток бізнесу.

Процес проведення дослідження ринку мобільних додатків.

Для бізнес-центрів неминуче мати чітке уявлення про проведення свого маркетингового дослідження. Більше того, немає чіткого шаблону, який визначає способи проведення маркетингових досліджень. Прагнучи виділитися серед своїх конкурентів, кожен бізнес проводить дослідження по-своєму.

Загалом процес дослідження ринку поділяється на дві великі категорії:

– Первинне дослідження. У первинних дослідженнях компанія повинна визначити фактичну потребу додатка на ринку. Після цього вони повинні розробити оптимальну бізнес-модель, щоб програма залишалася

актуальною на ринку. Після того, як бізнес-модель завершена, наступним кроком стає оптимізація маркетингової стратегії для програми. Маркетингова стратегія має набагато більше значення на сучасному ринку, оскільки клієнти більше схильні до персоналізованих послуг.

– Вторинне дослідження. Вторинні дослідження в основному зосереджені на головній силі мобільного додатка. Коли ви визначаєте основну силу мобільного додатка, розділити цільову аудиторію стає легко. Крім того, компанія може оптимізувати свою стратегію в соціальних мережах і задовольнити кожного зі своїх цільових клієнтів окремо.

Проведення дослідження ринку для мобільного додатка.

У сучасному діловому світі термін придатності мобільних додатків дуже нестабільний. Таким чином, стартапи повинні переконатися, що вони роблять все правильно, коли справа стосується розуміння потреб клієнтів. Маркетинг мобільних додатків допомагає компаніям залишатися на місці і приймати кращі рішення відповідно до ситуації.

Існує кілька способів проведення маркетингових досліджень. Далі перерахування деяких з топів для зручності.

Порівняння ідей додатків. Тут можуть стати в нагоді пошукові системи та різні інструменти дослідження. Це може допомогти підприємствам дізнатися більше про ринковий попит. Компанії можуть проаналізувати, чи є реальний попит на мобільні програми з вибраними ними функціями. Можна використовувати такі інструменти, як Google Trends та пошук щодо термінів, щоб зробити свою програму популярною.

Створення бізнес-моделі. Це неписане правило мати стоїчну бізнес- модель, і це буде важливою частиною стратегій дослідження ринку мобільних додатків. Кожна галузь та її ніша будуть різними та матиме інший набір принципів, яким слід дотримуватися. Отже, складання надійної бізнес-моделі важливо, щоб подолати дуже конкурентний жанр додатків. Бізнес- модель також може діяти як керівництво під час процесу запуску програми.

Все буде залежати від типу мобільного додатка, який створюватиметься, і від цільової аудиторії, яку потрібно задовольнити.

Визначення цільової аудиторії. Поряд із знанням проблем користувачів, потрібно визначити, хто є реальними користувачами. Крім того, визначення цільової аудиторії також допоможе змінити модель доходу.

Коли буде зрозуміло, чого хоче цільова аудиторія, буде включено лише ті функції, які найкраще підходять для програми. Тому що в сучасному сценарії високої конкуренції бути самовдоволеним чимось не вихід.

Потрібно зрозуміти, що роблять конкуренти. Після того, як зрозуміло бізнес-модель і цільову аудиторію, не потрібно забувати стежити за тим, що роблять конкуренти.

Розуміючи вищезгадані речі, можна внести необхідні зміни у бізнес-процес.

Слідкування за тенденціями Play Market. Важливе значення має належний аналіз Play Market, оскільки він допомагає розшифрувати моделі купівлі та зацікавленості клієнтів. Розробники додатків повинні завжди перевіряти діаграми популярності, щоб зрозуміти, що працює у світі додатків, і відповідно розробляти свій.

Важливо уважно стежити за тенденціями як безкоштовних, так і платних програм. Тенденції безкоштовних програм дадуть уявлення про те, що люди роблять у своїх мобільних додатках. З іншого боку, тенденції платних додатків змусять зрозуміти, куди варто витрачати гроші.

Критичність до власних ідей. Не потрібно уникати погляду на ідеї під критичним кутом. Це допоможе дізнатися його реальний потенціал. Більше того, це ще більше прояснить довіру до ідеї в магазині додатків. Важливість дослідження ринку мобільних додатків можна зрозуміти з того, що воно допоможе вирішити проблеми, пов’язані з достовірністю ідеї.

Процес розробки мобільних додатків і маркетингу мобільних додатків був би безнадійним без адекватних стратегій дослідження ринку мобільних додатків.

Тому підприємствам слід розробити конкретну стратегію дослідження ринку, оскільки це допоможе створити міцну базу для створення мобільного додатка для певної аудиторії. Крім того, завдяки стратегіям дослідження ринку додатків якість кінцевого продукту значно підвищується.

ВИСНОВКИ

В результаті виконання усіх частин проєкту дипломної роботи було виконано багато дослідницької та практичної роботи. Завдання щодо структури та інформаційної бази були виконані в потрібному вигляді. Додаток повністю функціонує та готовий до експлуатації, потрібно лише відредагувати деяку інформації головної активності додатка.

В ході підготовки до проєктування додатку досліджено та проаналізовано велику кількість матеріалів про формування та створення інформаційної системи обраної предметної області. Для цього відповідно було виконано роботу над аналізом обраної предметної області, що дало потрібну інформацію для початку проєктування проєкту.

В наступному етапі проєктування використано здобуті навики проєктування додатків. Безпосередньо застосовано на практиці знання роботи з середовищами створення UML діаграм та їх успішна візуалізація обраними інструментами.

Реалізовано спроєктовану інформаційну систему у вигляді мобільного додатку методами та інструментами описаними під час етапу проєктування. Код додатку виконано об’єктно-орієнтованою популярною мовою програмування Kotlin.

Після повного завершення розробки програмного продукту виконано ручне тестування, що допомогло покращити якість та виправити деякі помилки створеного додатку. Тестування проведено на емуляторі середовища розробки Android Studio. Після завершення усіх етапів проєктування та розробки отримано запланований продукт – інформаційну систему ОСББ «Оболоня-парк» у вигляді мобільного додатку для операційної системи Android.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дэвид Гриффитс. Head First. Программирование для Android. М.: Питер, 2016. 704 с.
2. Пол Дейтел. Android для разработчиков. М.: Питер, 2016. 512 с.
3. Рето Майер. Android 4. Программирование приложений для планшетных компьютеров и смартфонов. М.: ЄКСМО, 2013. 816 с.
4. Чад Фаулер. Программист-фанатик. М.:Питер, 2016. 208 с.
5. Android. *Вікіпедія - вільна енциклопедія:* веб-сайт. URL: https://ru.wikipedia.org/wiki/Android (дата звернення: 14.03.2022).
6. Android_Studio. *Вікіпедія - вільна енциклопедія:* веб-сайт. URL: https://ru.wikipedia.org/wiki/Android_Studio (дата звернення: 05.03.2022).
7. Dagger 2. Часть первая. *Habrahabr:* веб-сайт. URL: https://habrahabr.ru/post/279125/ (дата звернення: 25.03.2022).
8. Kotlin. *Вікіпедія - вільна енциклопедія:* веб-сайт. URL: https://ru.wikipedia.org/wiki/Kotlin (дата звернення: 17.03.2022).
9. Документація Git. *Git:* веб-сайт. URL: https://gitscm.com (дата звернення: 03.03.2022).
10. Документація Realm. *Realm:* веб-сайт. URL: https://realm.io (дата звернення: 03.03.2022).
11. Документація. Dagger *Dagger:* веб-сайт. URL: https://google.github.io/dagger (дата звернення: 07.03.2022).
12. Структура приложения для Android *Habrahabr:* веб-сайт. URL: https://habrahabr.ru/company/ncloudtech/blog/274025 (дата звернення: 25.03.2022).

ДОДАТКИ

Додаток А

Програмний код

Лістинг А1 – Код одного з адаптерів AnnouncementListAdapter

```
class AnnouncementListAdapter(private val onItemClicked: (Announcement)
-> Unit) :
    ListAdapter<Announcement,
AnnouncementListAdapter.ItemViewHolder>(DiffCallback) {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ItemViewHolder {
        return ItemViewHolder(
            AnnouncementListItemBinding.inflate(
                LayoutInflater.from(
                    parent.context
                )
            )
        )
    }

    override fun onBindViewHolder(holder: ItemViewHolder, position: Int)
{
        val current = getItem(position)
        holder.itemView.setOnClickListener {
            onItemClicked(current)
        }
        holder.bind(current)
    }
    class ItemViewHolder(private var binding:
AnnouncementListItemBinding) :
        RecyclerView.ViewHolder(binding.root) {

        fun bind(employee: Announcement) {
            binding.itemAnnouncementHeader.text =
employee.announcementHeader
            binding.itemAnnouncementBody.text =
employee.announcementBody
            binding.itemAnnouncementDate.text =
employee.announcementDate
        }}
```

```
    companion object {
        private val DiffCallback = object :
DiffUtil.ItemCallback<Announcement>() {
            override fun areItemsTheSame(oldItem: Announcement, newItem:
Announcement): Boolean {
                return oldItem === newItem
            }

            override fun areContentsTheSame(oldItem: Announcement,
newItem: Announcement): Boolean {
                return oldItem.announcementHeader ==
newItem.announcementHeader
            }
        }
    }
}
```

Лістинг А2 – Код сутності представлення інформації Announcement

```
@Entity(tableName = "announcement")
data class Announcement (
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    @NonNull @ColumnInfo(name = "header")
    val announcementHeader: String,
    @NonNull @ColumnInfo(name = "body")
    val announcementBody: String,
    @NonNull @ColumnInfo(name = "date")
    val announcementDate: String,)
```

Лістинг А3 – Код функції get_database()

```
fun getDatabase(context: Context): ObolonyaDatabase {
    // if the INSTANCE is not null, then return it,
    // if it is, then create the database
    return INSTANCE ?: synchronized(this) {
        val instance = Room.databaseBuilder(
            context,
            ObolonyaDatabase::class.java,
            "app_database")
            .createFromAsset("database/obolonya_database.db")
            .fallbackToDestructiveMigration()
            .build()
```

```
            INSTANCE = instance
            // return instance
            instance
        }
    }
```

Лістинг А4 – Код програмної частини фрагменту AnnouncementFragment

```
class AnnouncementFragment : Fragment() {
    private val viewModel: MainViewModel by activityViewModels {
        MainViewModel.ObolonyaViewModelFactory(
            (activity?.application as
ObolonyaApplication).database.obolonyaDao()
        )
    }

    private var _binding: FragmentAnnouncementBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        _binding = FragmentAnnouncementBinding.inflate(inflater,
container, false)
        return binding.root
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?)
{
        super.onViewCreated(view, savedInstanceState)

        val adapter = AnnouncementListAdapter {
        }

        binding.announcementRecyclerView.layoutManager =
LinearLayoutManager(this.context)
        binding.announcementRecyclerView.adapter = adapter
```

```
        // Attach an observer on the allItems list to update the UI
automatically when the data
        // changes.
        viewModel.allAnnouncement.observe(this.viewLifecycleOwner) {
items ->
            items.let {
                adapter.submitList(it)
            }
        }
    }
}
```

Лістинг А5 – Код класу MainViewModel

```
class MainViewModel(private val obolonyaDao: ObolonyaDao) : ViewModel()
{
    // Cache all items form the database using LiveData.
    val allEmployee: LiveData<List<Employee>> =
obolonyaDao.getAllEmployee().asLiveData()

    val allAnnouncement: LiveData<List<Announcement>> =
obolonyaDao.getAllAnnouncement().asLiveData()

    val allProvider: LiveData<List<Provider>> =
obolonyaDao.getAllProvider().asLiveData()

    val allService: LiveData<List<Service>> =
obolonyaDao.getAllService().asLiveData()
    /**
     * Factory class to instantiate the [ViewModel] instance.
     */
    class ObolonyaViewModelFactory(private val obolonyaDao: ObolonyaDao)
: ViewModelProvider.Factory {
        override fun <T : ViewModel> create(modelClass: Class<T>): T {
            if (modelClass.isAssignableFrom(MainViewModel::class.java))
{
                @Suppress("UNCHECKED_CAST")
                return MainViewModel(obolonyaDao) as T
            }
            throw IllegalArgumentException("Unknown ViewModel class")
        }
    }
}
```