ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка мобільного застосунку для пошуку рецептів по наявних інгредієнтах або обмеженому бюджеті для підтримки правильного харчування»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Артем ДУБОВИК
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Артем ДУБОВИК

Керівник: Вячеслав САВІЦЬКИЙ
викладач кафедри ІПЗ

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дубовику Артему Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка мобільного застосунку для пошуку рецептів по наявних інгредієнтах або обмеженому бюджеті для підтримки правильного харчування»
керівник кваліфікаційної роботи викладач кафедри ІПЗ Вячеслав САВІЦЬКИЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи створення мобільних застосунків, порівняльні методи дослідження предметної області

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз предметної області.
2. Аналіз існуючих аналогів.
3. Використані засоби розробки.

4. Вимоги до програмного забезпечення.
5. Моделювання та проєктування архітектури системи.
6. Опис розробки застосунку.
7. Тестування розробленої системи.
8. Висновки.

5. Перелік графічного матеріалу: *презентація*
1. Порівняльна таблиця аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Use case діаграма.
5. Діаграма розгортання.
6. Схема бази даних.
7. Екранні форми.
8. Відео представлення застосунку.
9. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02-06.03.2024	
2	Дослідження актуальності	07.03-13.03.2024	
3	Аналіз та дослідження існуючих аналогів	14.03-18.03.2024	
4	Проєктування архітектури та вибір бібліотек	18.03-.31.03.2024	
5	Створення функціоналу проєкту	01.04-25.04.2024	
6	Створення UI інтерфейсу	25.04-01.05.2024	
7	Тестування застосунку	01.05-05.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	05.05-15.05.2024	
8	Розробка демонстраційних матеріалів	15.05-23.05.2024	
9	Попередній захист роботи	13.05-24.05.2024	

Здобувач вищої освіти _____ Артем ДУБОВИК
(підпис)

Керівник
кваліфікаційної роботи _____ Вячеслав САВІЦЬКИЙ
(підпис)

РЕФЕРАТ

Текстова частина бакалаврської роботи 77с., 35 рис., 1 табл., 11 джерел.

Об'єкт дослідження – пошук та генерація рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Предмет дослідження – мобільний застосунок для пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Мета роботи – підвищення зручності користувачів при пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Методи дослідження – методи теорії інформації, методи теорії UI/UX інтерфейсу, пошук, вилучення та групування інформації.

Перед початком роботи було проаналізовано ринок подібних сервісів, такі як «KitchenStories», «CookPad», «Spillt».

Загальною перевагою розробленого застосунку над цими сервісами це індивідуальний підхід до пошуку та генерації рецептів для кожного користувача на базі його вподобань та навичок..

Для розробки було використано фреймворк Flutter та мову програмування Dart, а також Node.js для реалізації серверної частини. В якості середи розробки було обрано VSCode.

Отже, розроблено та описано застосунок, задачею якого є швидка генерація та пошук рецептів.

Галузь використання – застосунок може бути використано у будь-якій галузі життєдіяльності та не потребує професійних навичок для його використання.

Ключові слова: Flutter, Dart, Node.js, мобільний застосунок, рецепти.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

ОС – Операційна система

API – Application programming interface

URL – Universal resource locator

ВСТУП

Обґрунтування вибору теми та її актуальність. У наш час, коли доступ до інформації та послуг через мобільні застосунки став надзвичайно популярним, взаємодія з мобільним застосунком для пошуку рецептів стала вкрай важливою.

Часто, коли немає ідей, що приготувати, ми звертаємося до сайтів з пошуку рецептів, але не завжди зручно шукати підходящий рецепт, тим паче, буває, що не всі потрібні інгредієнти є в наявності. Для прискорення процесу пошуку та зручності отримання валідних рецептів був розроблений даний дипломний проект.

Використання штучного інтелекту дозволяє автоматизувати процес розпізнавання та класифікації інгредієнтів, наданих користувачем у списку, що забезпечує ефективну та точну роботу застосунку.

Ступінь вивчення проблеми. На сьогоднішній день існує багато сайтів та застосунків, які надають доступ до рецептів зі всього світу. Але в багатьох з них єдиний принцип роботи – всі вони лише посібник. Жоден з сервісів не аналізує особисті вподобання користувача, його навички та бюджет. Також більшість застосунків перевантажені рекламою, що гальмує процес пошуку рецептів та дратує користувача. Кулінарія постійно розвивається і в більшості таких сервісів інформація вже є неактуальною.

Об'єкт дослідження – пошук та генерація рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Предмет дослідження – мобільний застосунок для пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Мета роботи – підвищення зручності користувачів при пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

Методика дослідження. методи теорії інформації, методи теорії UI/UX інтерфейсу пошук, вилучення та групування інформації.

На основі проаналізованої інформації та отриманих даних, потрібно розробити комплексний підхід для вирішення задач, обрати відповідний набір інструментів, який би дозволив вирішити наявні проблеми застосовуючи мінімум ресурсів користувача.

Враховуючи всі аспекти до розробки програмного продукту, найкращим рішенням є використання штучного інтелекту, який застосовує процес збору даних з подальшим їх групуванням та виводом.

Наукова новизна полягає в створенні застосунку, який використовує простий мінімалістичний інтерфейс без рекламних модулів, та надає актуальні рецепти на основі вподобань, навичок та бюджету користувача.

Практична значущість результатів дослідження полягає у вирішенні важливої практичної задачі – розробки програмного забезпечення для пошуку рецептів на мові програмування Dart з використанням фреймворку Flutter та Node.js, що дозволяє прискорити процес пошуку рецептів, зробити процес індивідуальним, та спростити життя користувача.

Результати дослідження бакалаврської роботи апробовані на Всеукраїнській науково-технічній конференції: «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях».

1. АНАЛІЗ ТА ХАРАКТЕРИСТИКА МОБІЛЬНИХ ЗАСТОСУНКІВ

1.1. Історія створення мобільних застосунків та їх характеристика

Історія мобільних застосунків почалася у 1980-х роках з появою кишенькових комп'ютерів, які були першими пристроями, здатними виконувати програмне забезпечення. Ці перші застосунки були дуже простими та обмеженими за функціональністю, зосереджуючись на базових завданнях, таких як обробка тексту, калькуляції та планування календаря. Перші кишенькові комп'ютери, такі як Psion Organiser, з’явилися в середині 1980-х і започаткували еру мобільних обчислень.

У 1990-х роках з розвитком мобільних технологій з'явилися перші мобільні телефони, що підтримували встановлення застосунків. Одним з перших таких телефонів був IBM Simon, випущений у 1994 році. Він мав сенсорний екран і включав прості програми, такі як калькулятор, календар і адресна книга.

З появою смартфонів у 2000-х роках мобільні застосунки отримали новий імпульс розвитку. Смартфони, такі як перший iPhone (випущений у 2007 році), пропонували більш потужні процесори, розширені можливості підключення до інтернету та сенсорні екрани, що значно підвищило потенціал мобільних застосунків. Apple App Store, запущений у 2008 році, став революційним ринком, який дозволив розробникам легко розповсюджувати свої застосунки та досягати широкої аудиторії.

З ростом популярності смартфонів зросла і конкуренція серед розробників застосунків. Це призвело до появи широкого спектру програмного забезпечення для різних потреб користувачів, від розважальних ігор до складних бізнес-додатків. Застосунки стали важливим елементом екосистеми мобільних пристроїв, сприяючи їх популярності та впровадженню нових технологій.

Сьогодні мобільні застосунки стали не лише невід'ємною частиною повсякденного життя, а й ключовим інструментом для бізнесу, медіа, освіти та багатьох інших сфер. Вони надають доступ до інформації та сервісів у будь-який час і в будь-якому місці, забезпечуючи зручність та ефективність користувачам у їх повсякденних справах. Мобільні застосунки використовуються для спілкування, навчання, роботи, здоров'я, розваг та багато іншого, що робить їх універсальними інструментами для сучасної людини.

Протягом останніх років спостерігається тренд до розвитку мобільних застосунків у напрямку штучного інтелекту, віртуальної реальності та розширеної реальності. Ці технології відкривають нові можливості для інновацій та вдосконалення користувацького досвіду. Штучний інтелект дозволяє створювати більш інтелектуальні та адаптивні застосунки, що можуть аналізувати поведінку користувачів і пропонувати персоналізовані рішення. Віртуальна та розширена реальність забезпечують новий рівень взаємодії з цифровим контентом, роблячи його більш реалістичним та інтерактивним.

1.2. Переваги мобільних застосунків

Мобільні застосунки принесли безліч переваг для користувачів у всіх сферах життя. Ось декілька ключових аспектів, які роблять мобільні застосунки надзвичайно корисними та популярними:

Миттєвий доступ до інформації та сервісів: Мобільні застосунки дозволяють користувачам отримувати доступ до необхідної інформації та послуг у будь-який час і з будь-якого місця. Це підвищує продуктивність та ефективність, дозволяючи швидко знаходити відповіді на питання, робити покупки, замовляти послуги та виконувати інші завдання.

Покращення комунікації та зв'язку: Соціальні мережі та месенджери, такі як Facebook, Instagram, WhatsApp та Telegram, дозволяють користувачам легко взаємодіяти з друзями, родиною та колегами. Це сприяє покращенню комунікації та підтримці зв'язків у сучасному швидкому світі.

Персоналізація користувацького досвіду: Мобільні застосунки можуть адаптуватися до індивідуальних потреб та вподобань користувачів, пропонуючи персоналізований контент та послуги. Це робить використання застосунків більш зручним та приємним.

Спрощення планування та організації: Застосунки для керування завданнями, організації часу та створення та редагування документів, такі як Microsoft Office, Google Docs та Trello, допомагають користувачам ефективно планувати та організовувати свої справи.

Підтримка здорового способу життя: Застосунки для здоров'я та фітнесу, такі як Fitbit, MyFitnessPal та Headspace, допомагають користувачам вести здоровий спосіб життя, відстежуючи фізичну активність, харчування та сон. Це сприяє покращенню загального самопочуття та здоров'я.

Розширення можливостей бізнесу: У бізнесі мобільні застосунки стають інструментом для розширення клієнтської бази, підвищення продажів та удосконалення обслуговування клієнтів. Вони також сприяють впровадженню інноваційних технологій та підвищенню конкурентоспроможності компаній.

Інноваційні можливості: Мобільні застосунки дозволяють впроваджувати нові технології та рішення, що можуть змінити спосіб взаємодії користувачів з інформацією та сервісами. Наприклад, використання штучного інтелекту, віртуальної та розширеної реальності відкриває нові горизонти для інновацій та покращення користувацького досвіду.

Підтримка навчання та саморозвитку: Застосунки для навчання, такі як Duolingo, Coursera та Khan Academy, надають можливість отримувати нові знання та

навички у зручному форматі. Вони роблять навчання доступним для широкого кола людей, сприяючи саморозвитку та підвищенню кваліфікації.

1.3. Види мобільних застосунків

Існує безліч видів мобільних застосунків, які можна класифікувати за різними критеріями. Ось кілька основних видів:

Соціальні мережі та месенджери: Застосунки для спілкування з друзями, родиною та колегами, такі як Facebook, Instagram, WhatsApp, Telegram і т.д. Вони дозволяють користувачам обмінюватися повідомленнями, фотографіями, відео та іншими медіафайлами, підтримувати контакти та будувати соціальні мережі.

Продуктивність: Застосунки для керування завданнями, організації часу, створення та редагування документів, наприклад, Microsoft Office, Google Docs, Trello і т.д. Вони допомагають користувачам ефективно планувати свої справи, організовувати робочий процес та підвищувати продуктивність.

Здоров'я та фітнес: Застосунки для відстежування фізичної активності, здорового харчування, медитації та іншого, наприклад, Fitbit, MyFitnessPal, Headspace і т.д. Вони допомагають користувачам вести здоровий спосіб життя, відстежувати свої фізичні показники та досягати фітнес-цілей.

Ігри: Різноманітні ігри для розваг та розвитку, від простих головоломок до складних стратегічних ігор, такі як Candy Crush, PUBG Mobile, Clash of Clans і т.д. Ігри дозволяють користувачам розважатися, змагатися з іншими гравцями та розвивати свої навички.

Фінанси: Застосунки для управління фінансами, включаючи банківські застосунки, відстеження витрат, інвестування та платежі, наприклад, Revolut, Mint, PayPal і т.д. Вони допомагають користувачам керувати своїми фінансами, здійснювати платежі, відстежувати доходи та витрати.

Навчання: Застосунки для навчання та саморозвитку, включаючи онлайн-курси, мовні застосунки, застосунки для підготовки до іспитів і т.д., такі як Duolingo, Coursera, Khan Academy і т.д. Вони надають доступ до навчальних матеріалів та курсів, допомагають здобувати нові знання та навички.

Подорожі: Застосунки для планування подорожей, бронювання квитків, готелів, пошуку місць і туристичних атракцій, наприклад, Airbnb, Booking.com, Google Maps і т.д. Вони допомагають користувачам організовувати подорожі, знаходити цікаві місця та забезпечують комфорт під час подорожей.

Фотографія та відео: Застосунки для редагування, обробки та публікації фотографій і відео, такі як Instagram, Adobe Photoshop Express, TikTok і т.д. Вони дозволяють користувачам створювати та ділитися медіаконтентом, використовувати різні фільтри та ефекти для покращення якості зображень та відео.

1.4. Аналіз існуючих застосунків для пошуку рецептів

Розвиток комп’ютерних технологій та глобальна діджиталізація суттєво змінили наш підхід до готування їжі, переносячи багато аспектів цього процесу в онлайн простір. Люди все частіше вибирають шлях електронних джерел для пошуку рецептів, замість традиційних кулінарних книг. Це дозволяє їм знайти швидкі та різноманітні ідеї для приготування їжі, зручно адаптувати рецепти до власних потреб та обмінюватися враженнями з іншими користувачами. Проте, навіть у великому онлайн-просторі, вибір ідеального рецепту може стати завданням, і саме тут з'являються сервіси пошуку рецептів. Ці застосунки надають користувачам можливість швидко знайти рецепти за заданими критеріями, такими як складники, час приготування, тип страви та інші. Вони полегшують процес пошуку і дозволяють знайти ідеальну страву для будь-якої ситуації. Однак, серед недоліків таких сервісів може бути нестабільна якість рецептів, не завжди відповідна особистим смакам або

дієтичним обмеженням користувача. Також може виникати проблема з навігацією серед великої кількості варіантів, що може призвести до втрати часу та збентеження.

Застосунок «Kitchen Stories» - сервіс потребує реєстрування власного акаунту. На цій платформі, доступна англійська, німецька та китайська мова. Головна сторінка застосунку має приємний оку дизайн, темну тему (рис. 1.1), на ній знаходиться стрічка, аналогічна Instagram, з рецептами користувачів. Також ми можемо перемикатися між сторінкою зі статтями користувачів та персональною стрічкою. Але мінусом є те що вся стрічка доступна лише преміум користувачам на платній основі.

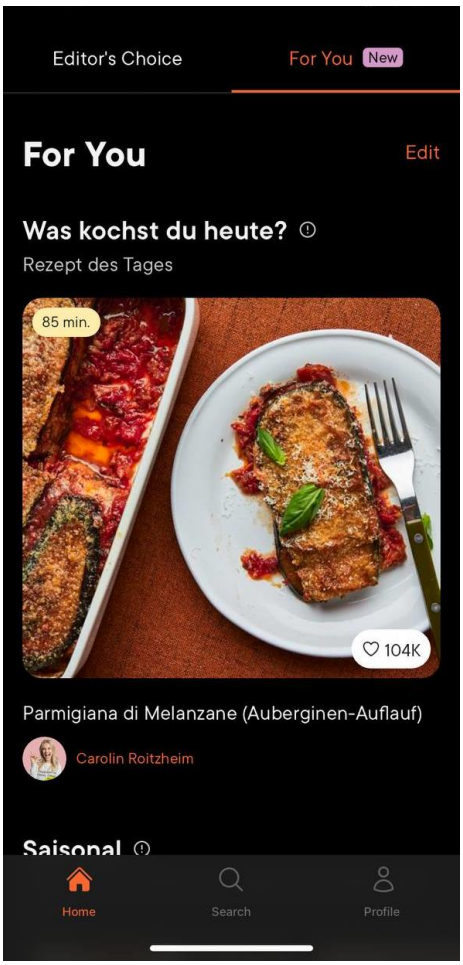


Рис.1.4.1 Головна сторінка застосунку

Також у застосунку доступний пошук рецептів за категорією (рис. 1.2). Можемо обрати, з наданих категорій, ту яка нас цікавить, відповідно до переваг користувача. Доступні категорії: страви з тіста, фастфуд, головні страви, десерти тощо. Кожна категорія описана у форматі карточки. Карточки розміщенні у два ряди і кожному ряду по три категорії. Знизу бачимо навігаційну панель, з якої можемо здійнити перехід на домашній екран, екран пошуку та профіль.

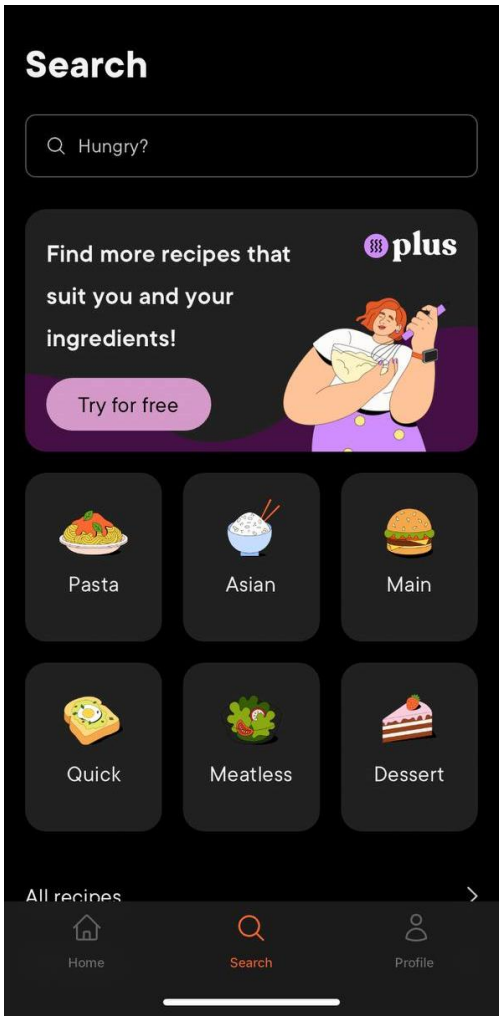


Рис.1.4.2 Сторінка пошуку рецептів

Після вибору категорії, можемо переглянути рецепт пасти (рис. 1.3). Одним з рецептів буде класична паста з соусом болоньєзе. На сторінці верхню половину екрану займає фотографія страви, в іншій половині зображений профіль користувача,

що опублікував рецепт та кількість користувачів, яким сподобався даний рецепт. Також користувач може зберегти цей рецепт, щоб в майбутньому скористатися ним і приготувати страву або ж поділитися у соціальних мережах, за допомогою кнопки «Share». В особистому кабінеті можемо переглянути рецепти, які ми зберегли.

Рис.1.4.3 Сторінка рецепту.

Застосунок «Cookpad» - сервіс потребує реєстрування власного аккаунту. На цій платформі доступна англійська та українська мова, що є великою перевагою. Дизайн, на мою думку, вже є доволі застарілим: дуже багато зайвих елементів, шрифт не читабельний, займає більшу частину фотографій страв. Головна сторінка застосунку, яка називається «Натхнення» (рис. 1.4), на ній знаходиться стрічка, аналогічна за принципом до Instagram, з рецептами користувачів.

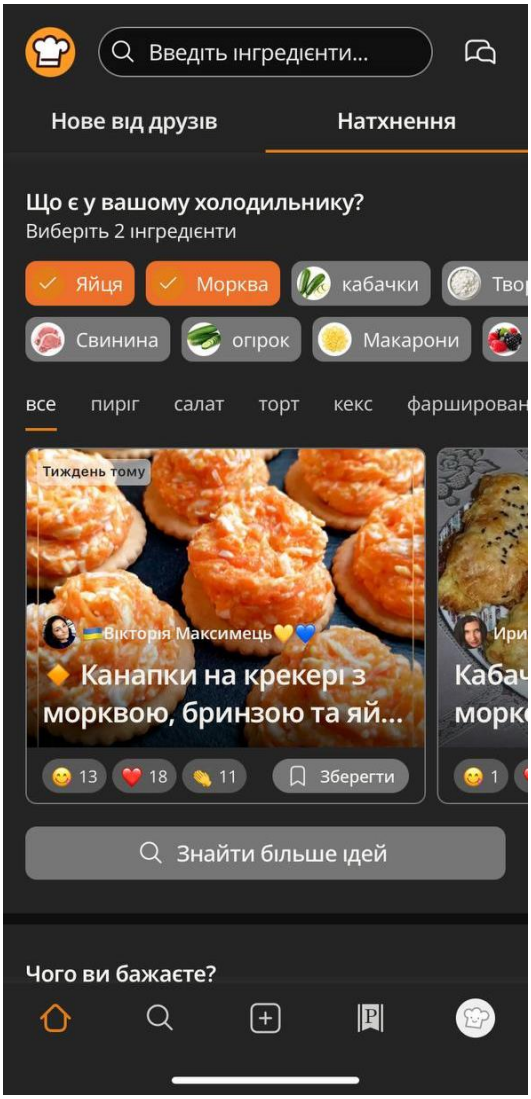


Рис.1.4.4 Головна сторінка

Є можливість перемкнутися на вкладку «Нове від друзів» та ознайомитися з рецептами, якими поділилися користувачі на яких ми підписалися. Також звідси ми можемо здійснити пошук за інгредієнтами, але мінусом є можливість обрати лише 2 інгредієнти. Якщо потрібно звузити діапазон пошуку, то з цим можуть виникнути проблеми і потрібний рецепт буде знайти набагато складніше. На сторінці пошуку присутні категорії страв, за якими можна здійснити пошук рецепту. На сторінці рецепту, отриманого після пошуку знаходиться фото готової страви, її творець та список інгредієнтів разом з кроками приготування. Кожен рецепт можна зберегти до

свого особистого кабінету. У даному застосунку ми можемо створити власний рецепт та поділитися з іншими користувачами (рис. 1.5). При процесі створення є можливість додати фото страви, час приготування їжі, інгредієнти тощо. В особистому кабінеті, схожому у дизайні до застосунку «KitchenStories» зберігаються наші рецепти та рецепти користувачів, які ми зберегли.

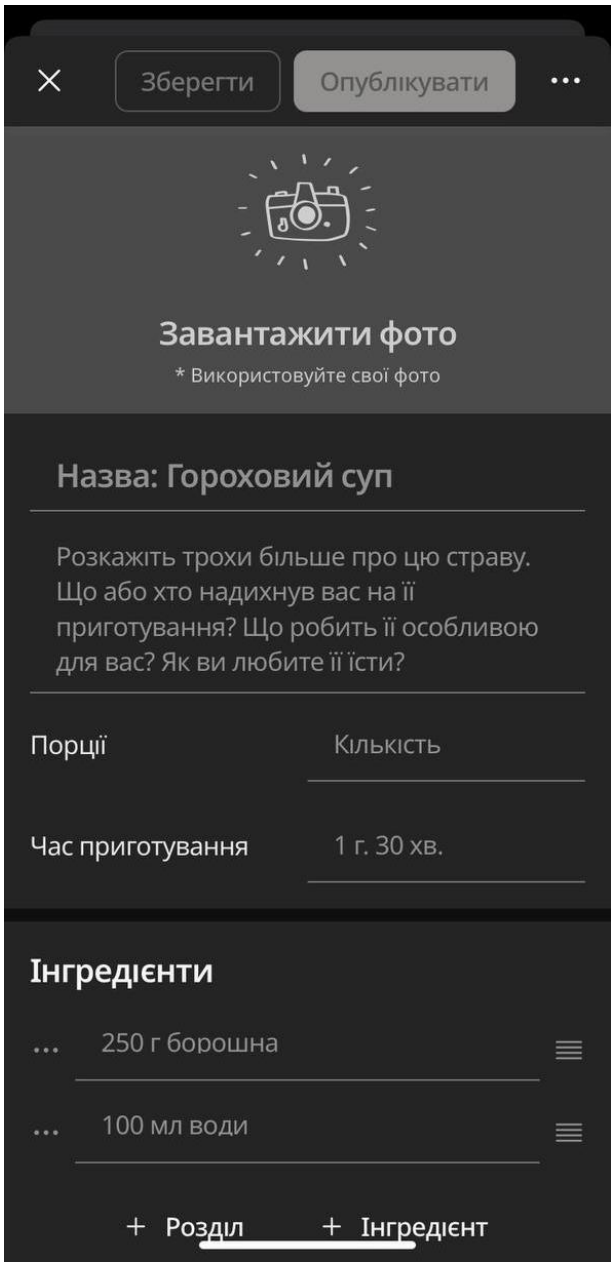


Рис.1.4.5 Сторінка створення рецепту

Додаток Spillt має мінімалістичний інтерфейс. При вході в додаток, на головній сторінці показано популярні рецепти. Окрім головної сторінки, на нижній панелі є кнопки переходу у збережені рецепти, список покупок, профіль та кнопка «+» створення рецепту. Рецепт можна імпортувати з іншого сайту, написати самому або перенести з фізичної книги рецептів. При створенні рецепту власноруч, додаток просить заповнити назву, інгредієнти, кроки приготування, додати фото. Недоліком є не дуже комфортна формастворення рецепту. Сторінку створення рецепту продемонтсровано на рис.1.6.

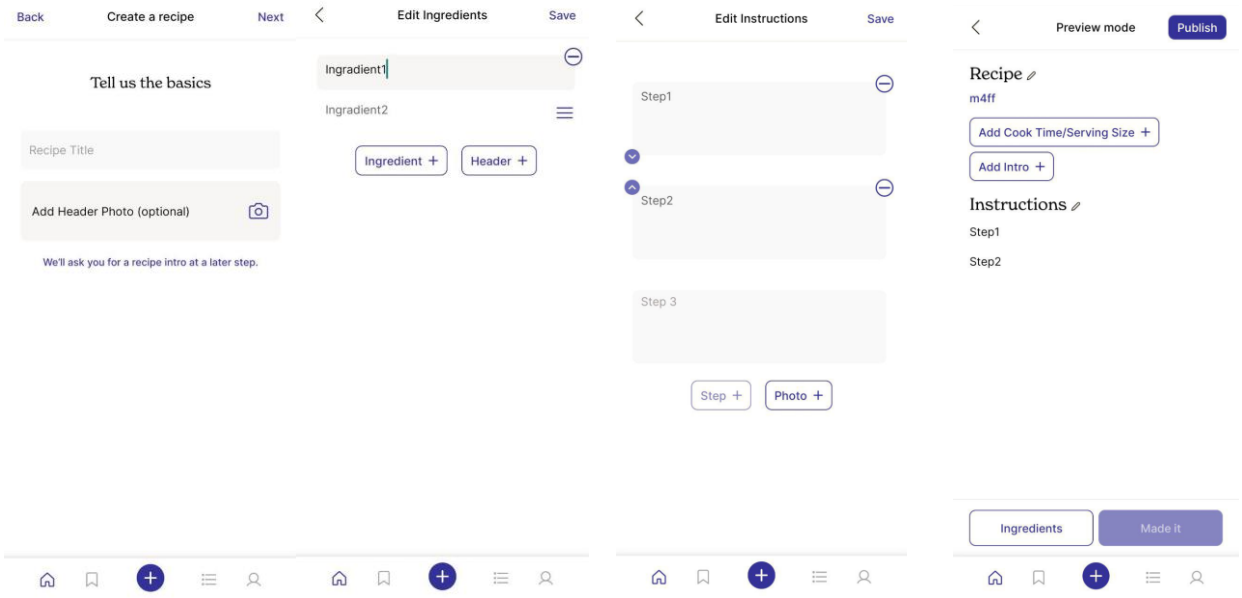


Рис.1.4.6 Сторінка створення рецепту на Spillt

Таблиця 1.4.1

Переваги та недоліки існуючих сервісів

Показник	**Kitchen Stories**	**Cookpad**	**Spillt**
Пошук рецептів за назвою	+	+	+
Пошук за тегами	+	+	-
Оцінка рецептів	Лайки	Лайки	Емоції
Можливість збереження рецептів	+	є, але в безкоштовній версії кількість обмежена	+
Створити рецепт вручну	+	+	+
Вибір мови	Тільки англійська	Англійська, українська	Тільки англійська
Пошук рецепту за інгредієнтами	-	-	+, але лише за двома
Інші особливості	Сторінка «Натхнення»	Наявність івентів та блогів	Можливість додати рецепт з фізичної книги, колекції рецептів

2. АНАЛІЗ ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Засоби розробки системи

У сучасному світі розробки мобільних та веб-застосунків, вибір правильної технології стає ключовим фактором для успіху проекту. Різноманіття доступних варіантів може ускладнити прийняття рішення, тому важливо ретельно вивчити та порівняти їх можливості. Основними технологіями, які були розглянуті мною для реалізації даного проєкту були такі мови програмування та фреймворки як: Java, ReactNative, Xamarin, Swift та Flutter. Я хотів би провести глибокий аналіз до кожного з варіантів, виділити його плюси та мінуси та обгрунтувати чому я обрав саме Flutter.

Java:

Java - це зріла та широко використовувана мова програмування, яка з'явилася в середині 90-х років і відтоді здобула величезну популярність у світі розробників програмного забезпечення. Вона має велику та активну спільноту, що надає розробникам можливість обмінюватися досвідом, отримувати підтримку в процесі створення програмних рішень, а також брати участь у численних форумах, конференціях та онлайн-курсах. Об'єктно-орієнтована природа Java сприяє інкапсуляції даних, успадкуванню та поліморфізму, що дозволяє створювати масштабовані та гнучкі програми. Завдяки модульності, Java дає змогу розбивати програми на менші, легко керовані компоненти, що полегшує їх розробку та обслуговування. Масштабованість Java робить її чудовим вибором для розробки складних мобільних застосунків, здатних обробляти велику кількість користувачів та даних.

Додатково, такі потужні інструменти розробки, як Java Studio, Eclipse та IntelliJ IDEA значно полегшують процес створення програм, надаючи зручні засоби для налагодження, тестування та розгортання програмного забезпечення. Крім того,

численні бібліотеки та фреймворки, такі як Spring, Hibernate та JavaFX, дозволяють розробникам швидко і ефективно вирішувати різноманітні задачі. Java також славиться своєю переносимістю, оскільки програми, написані на Java, можуть виконуватися на будь-якій платформі, що підтримує Java Virtual Machine (JVM). Це робить її чудовим вибором для розробки кросплатформених застосунків.

Багато компаній використовують Java для розробки корпоративних рішень, веб-застосунків, мобільних застосунків та навіть вбудованих систем. З розвитком технологій Java продовжує вдосконалюватися, пропонуючи нові можливості та інструменти для розробників. Інвестиції в навчання Java відкривають великі перспективи для кар'єрного росту, оскільки попит на кваліфікованих Java-розробників залишається стабільно високим у всьому світі.

Однак Java має певні недоліки:

- Складність для початківців: Синтаксис та концепції Java можуть здаватися складними для тих, хто не має досвіду програмування.
- Вербатість: Java потребує написання більшої кількості коду порівняно з деякими іншими мовами, що може призвести до громіздких програм.
- Не гнучкість для крос-платформного розвитку: Хоча існують фреймворки, які дозволяють використовувати Java для крос-платформного розвитку, вони не такі зрілі та зручні, як деякі альтернативи.

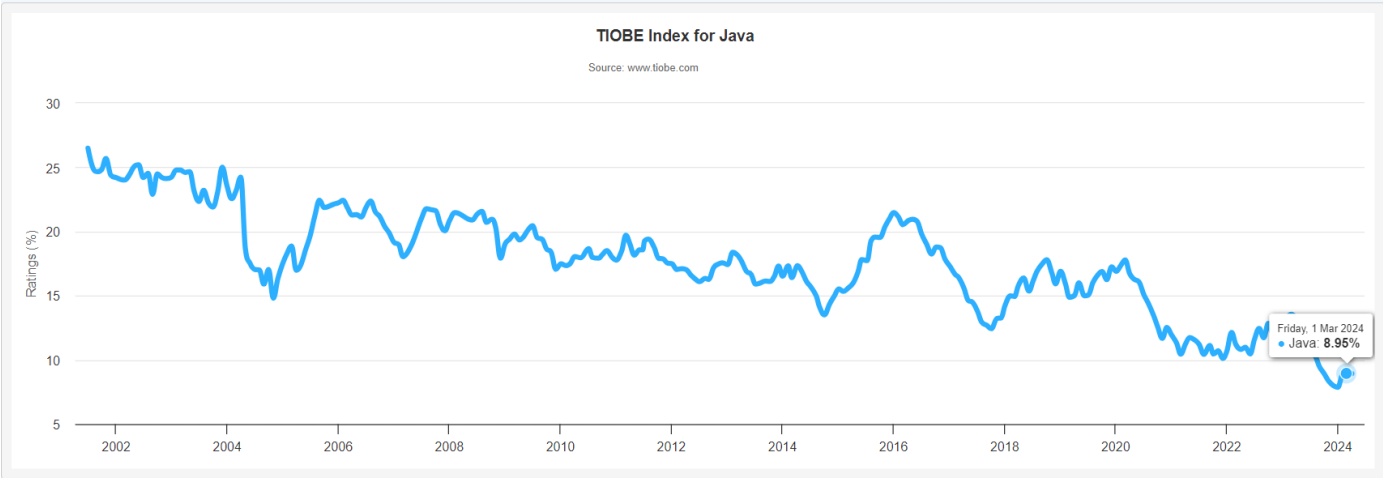


Рис.2.1.1 Рейтинг популярності мови програмування Java
з 2001 по 2024 рік

ReactNative:

ReactNative - це фреймворк, який дає можливість створювати мобільні застосунки за допомогою JavaScript, популярної та легкої для вивчення мови, що робить цей інструмент доступним навіть для початківців. Заснований на базі React, бібліотеки для побудови інтерфейсів користувача, ReactNative дозволяє розробникам використовувати знайомі концепції та інструменти для створення нативних мобільних застосунків. Він пропонує швидкий процес розробки завдяки можливості гарячого перезавантаження, що дозволяє розробникам миттєво бачити зміни у коді без необхідності перезапускати застосунок. Крім того, можливість повторного використання коду між різними платформами значно економить час та ресурси, оскільки один і той самий код можна використовувати для створення застосунків як для iOS, так і для Android.

Велика екосистема ReactNative включає численні бібліотеки та модулі, що розширюють функціональність фреймворку та дозволяють швидко додавати нові можливості до застосунків. Активна спільнота розробників забезпечує постійну підтримку та обмін знаннями, що допомагає вирішувати проблеми та впроваджувати нові ідеї. Серед інших переваг варто відзначити інтеграцію з такими популярними

інструментами, як Redux для керування станом застосунку та Axios для здійснення HTTP-запитів, що робить розробку ще більш ефективною та зручною.

ReactNative також підтримує роботу з нативними модулями, що дозволяє розширювати функціональність застосунків за допомогою мов програмування, таких як Java, Swift або Objective-C, коли це необхідно. Це забезпечує високу продуктивність та доступ до всіх можливостей нативних платформ. Багато відомих компаній, таких як Facebook, Instagram, Airbnb та Walmart, використовують ReactNative для розробки своїх мобільних застосунків, що підтверджує його надійність та ефективність.

Проте, ReactNative має певні обмеження:

- Проблеми з продуктивністю: На деяких пристроях ReactNative може демонструвати нижчу продуктивність порівняно з нативними мобільними застосунками.
- Залежність від сторонніх бібліотек: ReactNative значною мірою покладається на сторонні бібліотеки та інструменти, що може призвести до проблем з сумісністю та оновленнями.
- Не така нативна візуалізація: ReactNative не завжди може забезпечити таку ж рівень нативної візуалізації, як деякі інші мови програмування, що може призвести до неідеального користувацького досвіду.

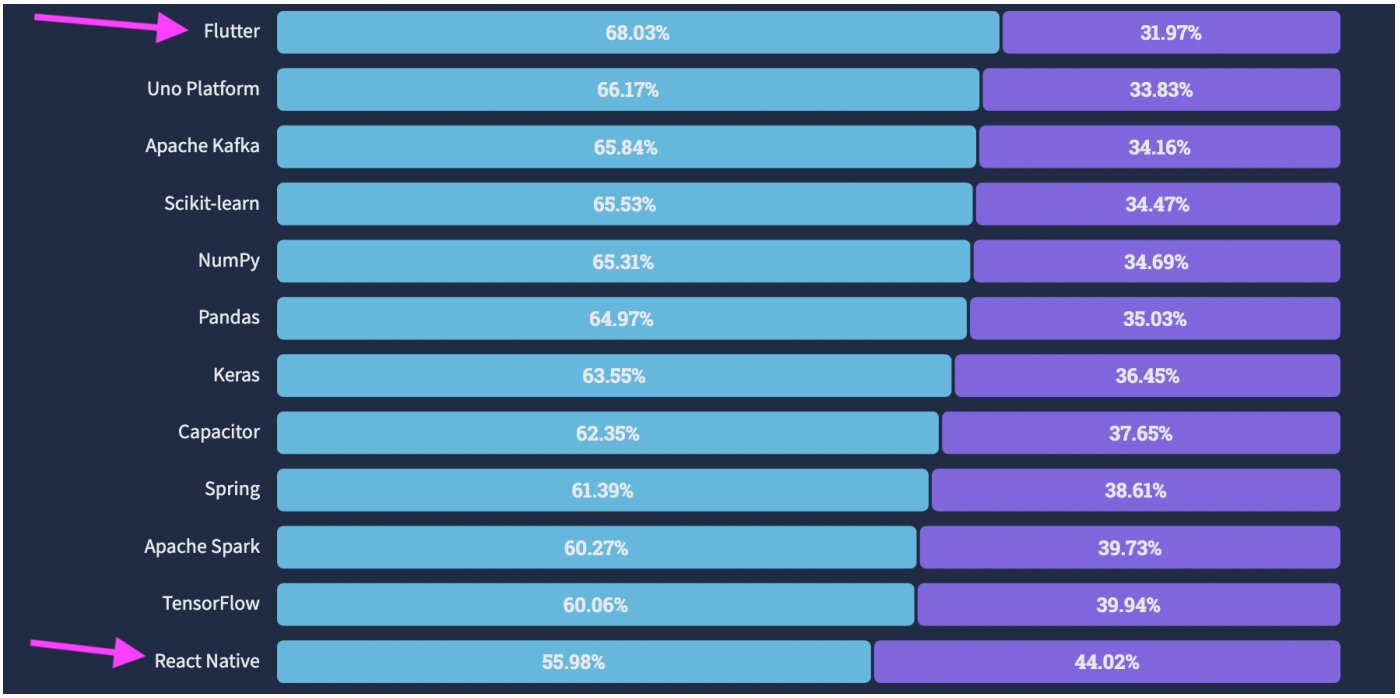


Рис.2.1.2 Результати для категорії «Найулюбленіші технології - фреймворки та бібліотеки» в опитуванні Stack Overflow 2022.

Xamarin:

Xamarin - це фреймворк, який дозволяє розробникам створювати нативні мобільні застосунки для iOS та Android за допомогою C#, знайомої мови для розробників .NET. Однією з основних переваг Xamarin є можливість написання коду лише один раз та його використання на обох платформах, що значно зменшує час та зусилля, необхідні для розробки та підтримки застосунків. Xamarin пропонує високу продуктивність, оскільки використовує нативні компоненти, що забезпечують швидку та плавну роботу застосунків.

Інтеграція з Visual Studio, потужним інструментом розробки від Microsoft, забезпечує зручне середовище для написання, налагодження та тестування коду. Крім того, Xamarin.Forms, бібліотека для створення інтерфейсів користувача, дозволяє створювати єдині інтерфейси для обох платформ, що ще більше спрощує процес розробки. Це означає, що розробники можуть створювати загальний користувацький

інтерфейс для iOS та Android, використовуючи єдиний код, що значно прискорює розробку та забезпечує послідовний вигляд і відчуття застосунків на різних платформах.

Також варто відзначити широкі можливості для тестування та аналітики, які надаються за допомогою таких інструментів, як Xamarin Test Cloud та App Center, що дозволяють забезпечити високу якість та стабільність застосунків. Xamarin Test Cloud дозволяє тестувати застосунки на реальних пристроях, що допомагає виявляти та виправляти помилки до випуску. App Center забезпечує зручний інструментарій для безперервної інтеграції та доставки, що спрощує процес розгортання оновлень та відстеження продуктивності застосунків.

Завдяки своїм можливостям Xamarin стає все більш популярним серед компаній, які прагнуть ефективно розробляти кросплатформені мобільні рішення. Впровадження Xamarin дозволяє знизити витрати на розробку та підтримку застосунків, забезпечуючи при цьому високу якість та продуктивність кінцевого продукту.

Однак Xamarin має певні недоліки:

- Мала спільнота та екосистема: Спільнота Xamarin та екосистема бібліотек та інструментів не такі зрілі та розширені, як у ReactNative або Flutter.
- Складність вивчення: C# може здатися складним для розробників, які не знайомі з екосистемою .NET.
- Не така гнучкість: Xamarin не пропонує такої ж рівня гнучкості для крос-платформного розвитку, як Flutter, що може обмежувати можливості розробників.

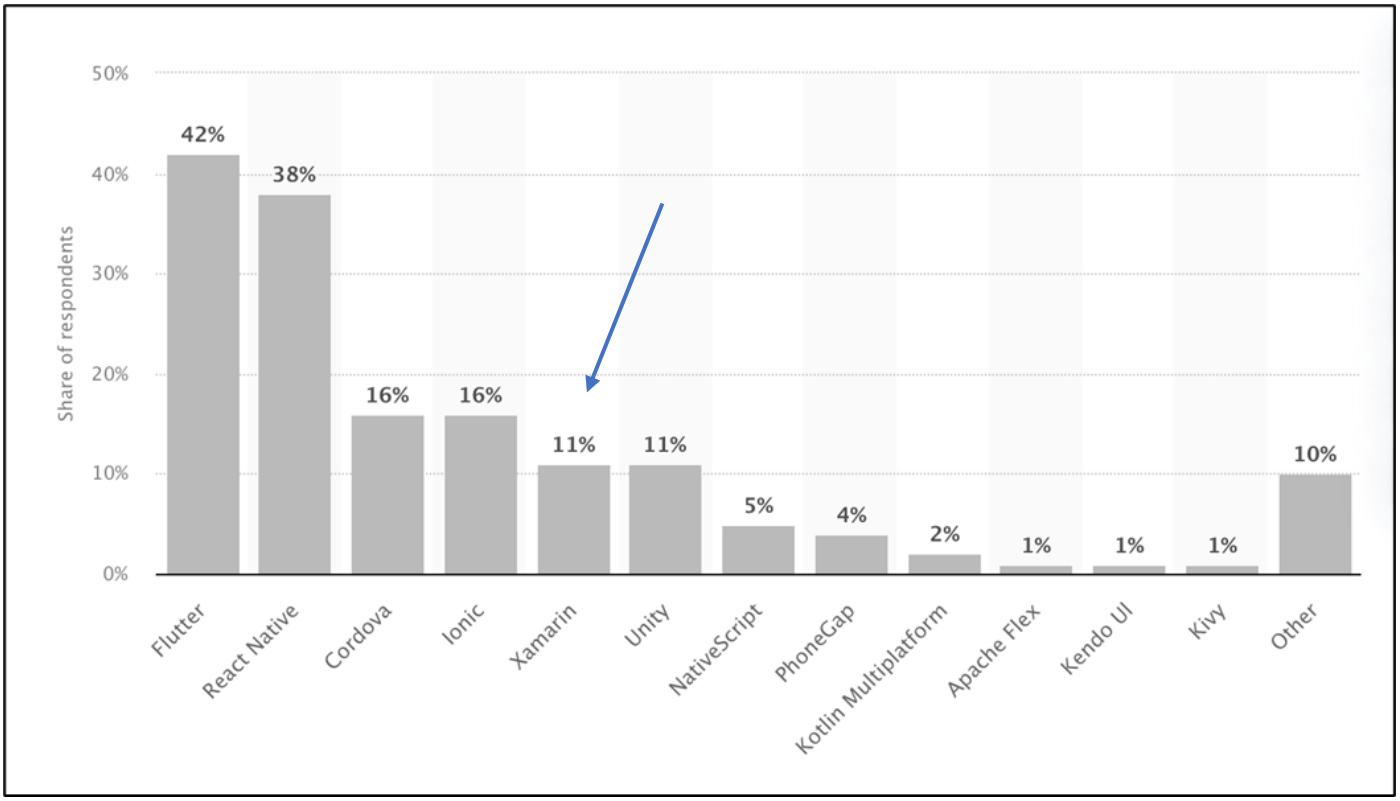


Рис.2.1.3 Статистика кількості розробників відповідно до фреймворків для розробки мобільних застосунків у відсотковому порівнянні

Swift:

Swift - це мова програмування, розроблена Apple спеціально для iOS та macOS, яка швидко завоювала популярність серед розробників завдяки своїм численним перевагам. Її сучасний та лаконічний синтаксис сприяє більш чистому та зрозумілому коду, що знижує кількість помилок та підвищує продуктивність розробників. Підтримка з боку Apple забезпечує регулярні оновлення та нові можливості, що робить Swift чудовим вибором для розробки нативних застосунків для платформ Apple.

Однією з важливих особливостей Swift є висока продуктивність, яка досягається завдяки оптимізованому компілятору та ефективному використанню ресурсів пристрою. Крім того, Swift має потужну систему управління пам'яттю та безпекою,

що допомагає уникати помилок та забезпечувати стабільну роботу застосунків. Інструменти розробки, такі як Xcode, надають зручне середовище для написання, налагодження та тестування коду, а також дозволяють швидко створювати інтерфейси користувача за допомогою Interface Builder.

Додатково, Swift активно підтримується великою спільнотою розробників, що сприяє обміну знаннями та розширенню можливостей мови. Apple регулярно проводить конференції та семінари, такі як WWDC (Worldwide Developers Conference), де розробники можуть дізнатися про нові функції Swift та обговорити найкращі практики використання мови.

Swift також підтримує сумісність з Objective-C, що дозволяє розробникам поступово переходити на нову мову, зберігаючи при цьому існуючі кодові бази. Це робить процес переходу на Swift менш стресовим та більш керованим для компаній, які вже мають значний обсяг коду на Objective-C.

Завдяки своїй зручності, потужності та підтримці з боку Apple, Swift стає все більш популярним вибором для розробників, які створюють застосунки для платформ Apple. Використання Swift дозволяє створювати швидкі, надійні та безпечні застосунки, що відповідають високим стандартам якості, встановленим Apple.

Однак Swift має певні обмеження:

- Обмежена сфера застосування: Swift використовується лише для платформ Apple (iOS, macOS, watchOS, tvOS). Це робить його непрактичним вибором для розробників, які хочуть створювати крос-платформні застосунки, доступні на Android або інших платформах.
- Складність вивчення: Swift має лаконічний синтаксис, але для розробників, не знайомих з екосистемою Apple, його вивчення може виявитися складним. Це може призвести до додаткових витрат часу та ресурсів на навчання та адаптацію.

- Залежність від Apple: Swift розроблений та підтримується Apple. Це може призвести до певних обмежень у свободі та гнучкості розробників. Apple може вносити зміни до мови або її інструментів, які не завжди відповідають потребам розробників.

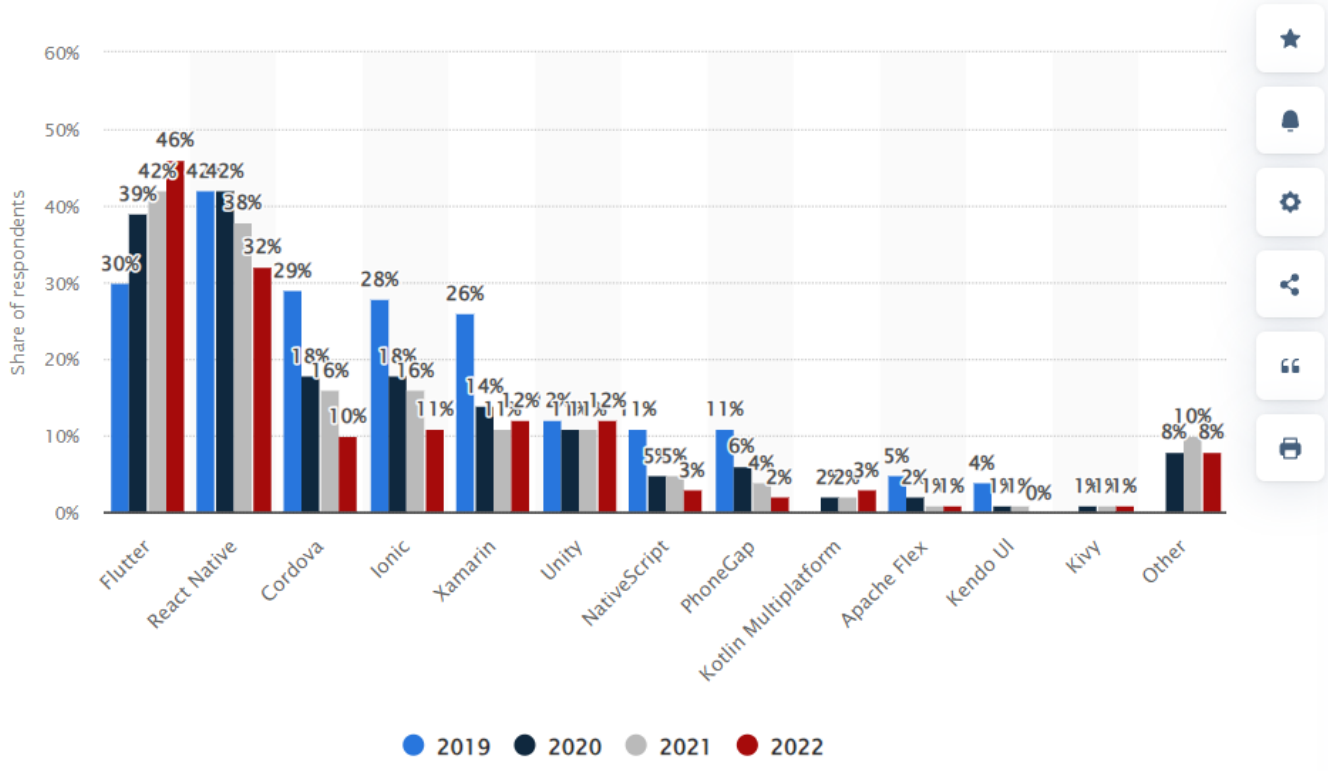


Рис.2.1.4 Статистика використання крос платформних фреймворків по всьому світу з 2019 по 2022 рік

Тож після ретельного аналізу та порівняння різних технологій, я зупинився на Flutter як оптимальному виборі для свого проекту. Ось ключові причини, які вплинули на моє рішення:

- Крос-платформна розробка: Flutter дозволяє створювати єдину кодову базу для мобільних застосунків (iOS та Android) та веб-застосунків. Це значно економить час та ресурси розробки, оскільки не потрібно писати та підтримувати окремі кодові бази для кожної платформи.
- Висока продуктивність та нативна візуалізація: Flutter використовує власний високопродуктивний движок рендерингу, який забезпечує нативну візуалізацію на всіх підтримуваних платформах. Користувацький інтерфейс застосунків, створених за допомогою Flutter, виглядатиме та працюватиме так само, як і нативні застосунки, написані на Swift або Java.
- Швидкий цикл розробки: Функція гарячого перезавантаження дозволяє миттєво відображати зміни коду в застосунку під час розробки. Це значно прискорює процес розробки та прототипування, оскільки не потрібно щоразу перезапускати застосунок для перегляду внесених змін.
- Зростаюча екосистема з підтримкою Google: Flutter - це відносно нова технологія, але вона швидко розвивається та вже має велику та активну спільноту розробників. Крім того, Google активно підтримує Flutter, що гарантує його довгострокову стабільність та розвиток.
- Гнучкість та можливості UI: Flutter пропонує багатий набір віджетів та інструментів, які дозволяють створювати складні та інтерактивні користувацькі інтерфейси. Ця гнучкість дає змогу розробникам реалізувати практично будь-який дизайн та функціональність застосунку.
- Dart - легка в освоєнні мова: Dart - це мова програмування, яку використовує Flutter. Вона має лаконічний синтаксис, схожий на JavaScript, що робить

її легкою для вивчення, навіть для розробників без попереднього досвіду роботи з Dart або іншими мовами програмування.

Висновок

Вибір технології розробки є важливим рішенням, яке може суттєво вплинути на успіх проекту. Flutter пропонує унікальне поєднання переваг: крос-платформна розробка, висока продуктивність, швидкий цикл розробки, зростаюча екосистема та гнучкість. Вважаю, що Flutter - це потужний інструмент, який ідеально підходить для створення мобільних застосунку, що відповідає моїм потребам.

2.2. Flutter

2015 рік ознаменував собою початок нової ери в мобільній розробці з появою Flutter - інструментарію Google, що кинув виклик домінуванню Java та Swift. Flutter обіцяв простоту, швидкість та нативну візуалізацію, пропонуючи альтернативу для створення мобільних застосунків з єдиною кодовою базою. Два роки інтенсивної роботи призвели до випуску Flutter 1.0 у 2017 році, який дав розробникам можливість втілювати свої ідеї в життя. Цей випуск став свідченням зрілості Flutter, готової до конкуренції з гігантами галузі. 2018 рік став роком єднання спільноти Flutter з появою Flutter Engage - щорічної конференції, де розробники ділилися досвідом, знаннями та найкращими практиками. Цей захід став рушійною силою розвитку Flutter, сприяючи його зростанню та єднанню. 2019 рік ознаменувався значним оновленням Flutter 1.0, яке значно покращило продуктивність, розширило UI-бібліотеки та наділило розробників потужними інструментами. Цей стрибок уперед підтвердив зрілість Flutter як платформи, готової до серйозних проектів. Межі розширилися у 2020 році, коли Flutter оголосив про підтримку веб-розробки, дозволяючи створювати веб-сайти та веб-застосунки з тією ж кодовою базою, що й мобільні. Цей крок зробив Flutter ще

більш універсальним та привабливим інструментом. 2021 рік став роком вражаючого прогресу з виходом Flutter 2.0, який значно покращив продуктивність, доступність та функціональність. Цей випуск зміцнив позиції Flutter як лідера в крос-платформній розробці, пропонуючи розробникам нові можливості та розширюючи горизонти. 2022 рік продовжив вражати з появою Flutter 3.0, який додав підтримку десктопних застосунків та вдосконалив мобільні та веб-можливості. Цей випуск підкреслив універсальність Flutter, роблячи його платформою не лише для мобільних пристроїв. 2023 рік став роком безперервного розвитку, де постійні оновлення та вдосконалення робили Flutter все більш потужною та гнучкою. Спільнота Flutter продовжувала динамічно розвиватися, роблячи Flutter все більш затребуваним інструментом.

2024 рік ознаменував собою домінування Flutter на ринку крос-платформної розробки, пропонуючи розробникам безпрецедентний рівень продуктивності, гнучкості та універсальності. Цей успіх - результат наполегливої роботи та відданості спільноти Flutter, яка вірить у майбутнє цієї революційної технології. Flutter - це не просто інструмент, це екосистема, яка постійно розширюється та розвивається. Її історія демонструє стрімке зростання та еволюцію, роблячи Flutter платформою майбутнього для крос-платформної розробки.

2.3. Dart

2011 рік ознаменував собою появу Dart - мови програмування, створеної Google, яка прагнула кинути виклик домінуванню JavaScript. Dart обіцяв розробникам кращу продуктивність, зручність використання та надійність. Перші роки Dart були позначені інтенсивним розвитком, але широкого впровадження не сталося. 2013 рік став поворотним моментом з випуском Dart 1.0, який офіційно вийшов з бета-тестування. Цей випуск ознаменував собою зрілість Dart, готову до використання в реальних проектах. 2014 рік став роком розширення можливостей Dart з виходом Dart

2.0, який представив суттєві покращення, включаючи AOT-компіляцію, підтримку web workers та покращену роботу з динамічними типами. Цей стрибок уперед підтвердив потенціал Dart як серйозного гравця в світі мов програмування. 2015 рік став свідком народження Flutter - інструментарію Google для крос-платформної розробки, який використовував Dart як основну мову програмування. Цей крок дав Dart нове життя, зробивши його невід'ємною частиною революційної платформи. 2016 рік ознаменувався зростанням популярності Dart завдяки випуску Dart 2.1, який приніс нові можливості, включаючи підтримку null safety та покращену роботу з async/await. Цей випуск зробив Dart ще більш зручним та потужним інструментом для розробників. 2017 рік став роком стрімкого розвитку Dart з виходом Dart 2.6, який представив значні покращення в продуктивності, візуалізації та інструментах розробки. Цей стрибок уперед зміцнив позиції Dart як мови програмування, здатної конкурувати з найкращими в галузі. 2018 рік приніс Dart 2.7, який додав нові можливості, включаючи підтримку інтерфейсів та покращену роботу з класами. Цей випуск зробив Dart ще більш гнучким та універсальним для розробки складних програм. 2019 рік ознаменувався випуском Dart 2.8, який представив покращення в продуктивності, візуалізації та роботі з веб-браузерами. Цей випуск підтвердив прихильність Google до розвитку Dart як мови програмування світового класу. 2020 рік став роком інновацій з виходом Dart 2.9, який представив революційну функцію під назвою "null safety" за замовчуванням. Цей крок зробив Dart однією з найбезпечніших та найнадійніших мов програмування на ринку. 2021 рік приніс Dart 2.12, який додав нові можливості, включаючи підтримку generics та покращену роботу з динамічними типами. Цей випуск зробив Dart ще більш потужним та гнучким для розробки складних програм. 2022 рік ознаменувався випуском Dart 2.17, який представив покращення в продуктивності, візуалізації та інструментах розробки. Цей випуск підтвердив прихильність Google до розвитку Dart як мови програмування, здатної конкурувати з найкращими в галузі. 2023 рік став роком безперервного

розвитку, де постійні оновлення та вдосконалення робили Dart все більш зручним, потужним та універсальним. Спільнота Dart продовжувала динамічно розвиватися, роблячи Dart все більш затребуваним інструментом для розробників.

2024 рік – це рік, коли Dart продовжує свій тріумфальний хід, пропонуючи розробникам безпрецедентний рівень продуктивності, зручності використання, надійності та гнучкості. Цей успіх – результат наполегливої роботи та відданості спільноти Dart, яка вірить у майбутнє цієї революції.

2.4. Node.js

Node.js – це середовище виконання JavaScript, яке дозволяє запускати код на серверній стороні. Засноване на рушії V8 від Google, Node.js було створене в 2009 році Райаном Далем і швидко здобуло популярність завдяки своїй здатності обробляти одночасно велику кількість запитів з високою продуктивністю. Це середовище виконання відкриває нові горизонти для JavaScript, дозволяючи розробникам використовувати одну мову для написання як клієнтського, так і серверного коду.

Розробка Node.js почалася з простої ідеї – забезпечити неблокуючу, асинхронну модель вводу-виводу, яка дозволяє створювати масштабовані серверні застосунки. Цей підхід став основою для побудови ефективних мережевих серверів, здатних обробляти тисячі одночасних з'єднань без перевантаження. У 2010 році, після офіційного релізу, Node.js почало активно розвиватися і впроваджувати нові можливості, які дозволили створювати високопродуктивні серверні рішення.

Однією з основних причин популярності Node.js є його екосистема, зокрема npm (Node Package Manager), який є найбільшим у світі реєстром програмних пакетів. За допомогою npm розробники можуть легко знаходити, встановлювати та використовувати численні бібліотеки та модулі, що значно прискорює процес

розробки. Від моменту свого запуску, npm розрісся до мільйонів пакетів, що робить його надзвичайно корисним інструментом для будь-якого проекту.

Node.js активно підтримується великою спільнотою розробників, що сприяє постійному вдосконаленню платформи. Регулярні оновлення та нові випуски забезпечують стабільність, безпеку та нові функціональні можливості. У 2015 році відбувся важливий етап в історії Node.js – проект об'єднався з io.js, що дало змогу прискорити розвиток та забезпечити більш активне впровадження нових стандартів та можливостей.

Завдяки своїй асинхронній природі, Node.js ідеально підходить для розробки реального часу додатків, таких як чати, ігри та інші інтерактивні сервіси. Використовуючи WebSockets, розробники можуть створювати двосторонні канали зв'язку між клієнтами та серверами, забезпечуючи миттєве оновлення даних. Це робить Node.js популярним вибором для стартапів та великих компаній, які прагнуть забезпечити високу продуктивність та масштабованість своїх веб-сервісів.

Node.js також відомий своєю модульністю. Модульна система Node.js дозволяє розробникам легко організовувати свій код та повторно використовувати компоненти у різних проектах. Ця особливість значно полегшує підтримку та розширення програмного забезпечення, що робить його привабливим вибором для розробників будь-якого рівня досвіду.

У 2016 році Node.js отримав офіційну підтримку для роботи з ECMA Script 6 (ES6), що включає сучасні функції JavaScript, такі як стрілочні функції, класи та проміси. Це забезпечило розробникам більш зручний та потужний інструментарій для написання коду, що ще більше підвищило привабливість Node.js.

З 2017 року Node.js активно розвивається у напрямку поліпшення продуктивності та безпеки. Випуск Node.js 10 у 2018 році включив підтримку нових можливостей V8, таких як покращення збору сміття та оптимізації пам'яті, що

підвищило загальну продуктивність серверних застосунків. Також було введено підтримку HTTP/2, що забезпечило швидше та ефективніше оброблення запитів.

У 2019 році Node.js отримав підтримку для Workers Threads – інструменту, який дозволяє виконувати паралельні операції у фоновому режимі, що значно покращило продуктивність багатозадачних серверних застосунків. Це стало важливим кроком у напрямку створення ще більш потужних та ефективних серверних рішень.

У 2020 році Node.js продовжив вдосконалюватися, включаючи підтримку ES Modules, що дозволило розробникам використовувати сучасні стандарти імпорту та експорту модулів, полегшуючи написання та організацію коду. Також були впроваджені численні поліпшення безпеки, що забезпечило ще більшу стабільність та надійність платформи.

Node.js став важливим інструментом для DevOps-практик завдяки своїй інтеграції з популярними інструментами автоматизації, такими як Jenkins, Docker та Kubernetes. Це дозволяє командам розробників швидко розгортати та масштабувати свої додатки, забезпечуючи високу якість та швидкість розробки.

Станом на 2021 рік, Node.js продовжує залишатися однією з найпопулярніших платформ для розробки серверних застосунків, пропонуючи розробникам потужний та гнучкий інструмент для створення масштабованих та продуктивних веб-додатків. Спільнота Node.js продовжує динамічно розвиватися, впроваджуючи нові ідеї та покращення, що робить Node.js ще більш потужною та затребуваною платформою у світі розробки програмного забезпечення.

У 2022 році Node.js випустив версію 16, яка включила нові можливості, такі як поліпшене управління пам'яттю та підтримка нових API, що зробило розробку ще більш зручною та ефективною. Також були впроваджені численні поліпшення безпеки, що забезпечило ще більшу стабільність та надійність платформи.

2023 рік став роком подальшого зростання та вдосконалення Node.js. Постійні оновлення та вдосконалення зробили платформу ще більш потужною та гнучкою,

дозволяючи розробникам створювати ще більш продуктивні та масштабовані рішення. Спільнота Node.js продовжувала динамічно розвиватися, роблячи Node.js все більш затребуваним інструментом у світі розробки програмного забезпечення.

2024 рік ознаменував собою подальше зміцнення позицій Node.js на ринку серверних рішень. Node.js пропонує розробникам безпрецедентний рівень продуктивності, гнучкості та універсальності, що робить його одним з найпопулярніших інструментів для створення серверних застосунків. Цей успіх – результат наполегливої роботи та відданості спільноти Node.js, яка вірить у майбутнє цієї революційної технології. Node.js – це не просто інструмент, це екосистема, яка постійно розширюється та розвивається. Її історія демонструє стрімке зростання та еволюцію, роблячи Node.js платформою майбутнього для розробки серверних застосунків.

Я використав Node.js для реалізації REST API, що виявилося зручним та ефективним рішенням. Завдяки асинхронній природі Node.js та його здатності обробляти численні одночасні запити, процес розробки був швидким та гнучким. Використання Express.js – популярного веб-фреймворку для Node.js – значно спростило створення маршрутизації та обробку запитів, дозволяючи зосередитися на бізнес-логіці застосунку. Модульність Node.js полегшила інтеграцію з базою даних та іншими сервісами, забезпечуючи високу продуктивність та надійність кінцевого продукту.

2.5. VSCode

Visual Studio Code (VSCode) - це безкоштовний редактор коду з відкритим кодом, розроблений компанією Microsoft. Він швидко завоював популярність серед розробників завдяки своїй простоті, гнучкості та розширюваності. VSCode підтримує

широкий спектр мов програмування, включаючи JavaScript, Python, Java, C++, C# та багато інших. Він також пропонує безліч функцій, які полегшують розробку, таких як:

- Підсвічування синтаксису: Код підсвічується різними кольорами, щоб зробити його більш читабельним і зрозумілим.
- Автозаповнення коду: VSCode може автоматично заповнювати фрагменти коду, що економить час і зменшує ймовірність помилок.
- Відлагодження: Ви можете відлагоджувати код безпосередньо в редакторі, встановивши точки зупинки та переглядаючи значення змінних.
- Інтеграція з Git: VSCode безпосередньо інтегрується з Git, що дозволяє вам легко керувати своїми репозиторіями.
- Розширення: Існує безліч розширень, які додають нові функції до VSCode, такі як теми, інструменти форматування та інтеграція з іншими інструментами.

VSCode доступний для Windows, macOS та Linux. Він також доступний у веб-версії, яка не потребує встановлення.

Переваги Visual Studio Code

- Безкоштовний та з відкритим кодом: VSCode можна використовувати безкоштовно, і його код доступний для публічного доступу.
- Простий у використанні: VSCode має зрозумілий інтерфейс, який легко освоїти.
- Гнучкий: VSCode можна налаштувати за допомогою розширень і тем, щоб він відповідав вашим потребам.
- Потужний: VSCode пропонує безліч функцій, які полегшують розробку.
- Підтримка багатьох мов програмування: VSCode підтримує широкий спектр мов програмування.

Недоліки Visual Studio Code

- Може бути повільним на великих проектах: VSCode може уповільнюватися на великих проектах з великою кількістю файлів.

- Не всі функції доступні в безкоштовній версії: Деякі функції, такі як віддалене редагування та інтеграція з Azure, доступні лише в платній версії.

Функції Visual Studio Code

- Редагування коду: VSCode пропонує потужний редактор коду з безліччю функцій, таких як підсвічування синтаксису, автозаповнення коду та складання коду.
- Відлагодження: VSCode дозволяє вам відлагоджувати код безпосередньо в редакторі, встановивши точки зупинки та переглядаючи значення змінних.
- Інтеграція з Git: VSCode безпосередньо інтегрується з Git, що дозволяє вам легко керувати своїми репозиторіями.
- Розширення: Існує безліч розширень, які додають нові функції до VSCode, такі як теми, інструменти форматування та інтеграція з іншими інструментами.
- Інтерфейс користувача: VSCode має зрозумілий інтерфейс користувача, який можна налаштувати за допомогою тем і розширень.
- Підтримка платформ: VSCode доступний для Windows, macOS та Linux. Він також доступний у веб-версії, яка не потребує встановлення.

VSCode має велику та активну спільноту розробників, які створюють розширення, пишуть документацію та допомагають іншим користувачам. Всю потрібну інформацію, допомогу та підтримку можна знайти на офіційному веб-сайті VSCode, на форумах та в соціальних мережах.

2.6. Android studio

Android Studio - це офіційне IDE, розроблене Google, що гарантує його повну сумісність з найновішими інструментами та платформами Android. Це означає, що розробник можете бути впевнений, що код завжди буде працювати без проблем на нових пристроях та версіях Android. Google постійно оновлює та вдосконалює Android Studio, гарантуючи його актуальність та надійність.

Комплексний набір інструментів:

Android Studio пропонує все необхідне для створення Android-застосунків будь-якої складності: вбудований емулятор Android дозволяє свої застосунки на різних віртуальних пристроях Android без необхідності мати фізичні девайси.

Інструменти створення інтерфейсу користувача: Drag-and-drop інструменти та візуальний редактор макетів дозволяють легко створювати красиві та зручні інтерфейси користувача.

Відлагодження та профілювання: виявляйте та виправляйте помилки в коді, а також аналізуйте продуктивність ваших застосунків за допомогою потужних інструментів.

Система побудови Gradle: унікальний доступ до процесу збірки та публікації застосунків.

Інтеграція з Google Play: можливість публікувати свої застосунки в Google Play Store безпосередньо з Android Studio.

Глибока інтеграція з платформою: Android Studio має глибоку інтеграцію з платформою Android, що дає доступ до розширених функцій та можливостей, недоступних у Visual Studio Code. Це полегшує роботу з ресурсами Android, такими як теми, стилі та API.

Велика та активна спільнота: Завдяки величезній спільноті розробників Android завжди є можливість знайти допомогу та поради на форумах, у блогах та групах в соціальних мережах. Це робить процес навчання та вирішення проблем значно простішим.

Оптимізація для Android: Android Studio оптимізований для роботи з проектами Android, що забезпечує кращу продуктивність та швидкість роботи, порівняно з Visual Studio Code, який потребує додаткових плагінів та налаштувань.

Постійні оновлення та нові функції: Google регулярно оновлює Android Studio, додаючи нові функції та вдосконалення, щоб йти в ногу з найновішими

тенденціями та потребами розробників. Це гарантує, що ви завжди будете мати доступ до найсучасніших інструментів для розробки Android-застосунки.

Досвід користувача: Інтерфейс користувача Android Studio спеціально розроблений для розробки Android, що робить його більш інтуїтивно зрозумілим та зручним для цієї мети, порівняно з Visual Studio Code, який має більш універсальний дизайн.

Повна підтримка Flutter: Android Studio пропонує офіційну підтримку розробки Flutter. Розробник отримує доступ до функцій, спеціально призначених для покращення досвіду роботи з Flutter, таких як:

- Гаряче перезавантаження (Hot reload): При зміні коду ми миттєво бачимо результат в емуляторі або на підключеному пристрої без перезапуску програми.
- Завершення коду та refactoring: Android Studio пропонує інструменти для автоматичного завершення коду Flutter, а також для його рефакторингу, що допомагає писати чистий та ефективний код.
- Відлагодження Flutter: потужні інструменти відлагодження Android Studio для виявлення та усунення помилок у коді.
- Android Studio має добре структурований інтерфейс, призначений для розробки Android. Хоча він може трохи відрізнятися від звичайної роботи з Dart, загалом він залишається інтуїтивно зрозумілим середовищем для написання коду та керування проектами Flutter.

Зважаючи на всі вищезгадані фактори, я вважаю, що Android Studio є кращим вибором для серйозної розробки мобільних застосунків. Його офіційний статус, комплексний набір інструментів, глибока інтеграція з платформою, велика спільнота, постійні оновлення, оптимізація для Android та зручний інтерфейс користувача роблять його незамінним інструментом для будь-якого розробника мобільних застосунків.

2.7. Microsoft SQL Server (MS SQL):

Microsoft SQL Server, відомий також як MS SQL, є реляційною системою управління базами даних (RDBMS), розробленою Microsoft. Вперше випущений у 1989 році, MS SQL швидко став одним з найпопулярніших виборів для зберігання та управління даними завдяки своїй надійності, масштабованості та гнучкості. Сьогодні MS SQL використовується в різних галузях, включаючи фінанси, охорону здоров'я, роздрібну торгівлю та багато інших, забезпечуючи стабільність і високу продуктивність для критично важливих додатків.

Однією з ключових особливостей MS SQL є його інтеграція з іншими продуктами Microsoft, такими як Windows Server, Azure та Visual Studio. Ця тісна інтеграція дозволяє розробникам та адміністраторам баз даних ефективно використовувати існуючу інфраструктуру та інструменти для розробки, розгортання та управління базами даних. Наприклад, можливості інтеграції з Azure забезпечують просте створення хмарних рішень, що дозволяє організаціям легко масштабувати свої бази даних та забезпечувати безперервну доступність.

MS SQL підтримує широкий спектр функцій, включаючи транзакційну обробку, аналітику та індексацію, що робить його потужним інструментом для обробки великих обсягів даних. Функції безпеки, такі як шифрування даних, управління правами доступу та аудит, забезпечують високий рівень захисту конфіденційної інформації. Крім того, вбудовані інструменти для резервного копіювання та відновлення даних допомагають зберігати цілісність даних та мінімізувати ризики втрат.

У 2012 році Microsoft випустила SQL Server 2012, який приніс значні покращення у продуктивності та функціональності. Введення функції AlwaysOn Availability Groups забезпечило високу доступність та аварійне відновлення, що дозволило організаціям досягати безперервності бізнес-процесів навіть у разі збоїв.

Інші нововведення включали поліпшене управління пам'яттю, вдосконалену індексацію та нові можливості для роботи з великими даними.

З випуском SQL Server 2016 Microsoft представила нові можливості для роботи з аналітикою та машинним навчанням. Інтеграція з R Services дозволила виконувати аналіз даних та побудову моделей машинного навчання безпосередньо в базі даних, що значно підвищило продуктивність та ефективність. Ця версія також включала функції для роботи з неструктурованими даними та поліпшені можливості для управління даними в хмарі.

SQL Server 2017 став першою версією MS SQL, доступною на платформах Linux та Docker, що забезпечило ще більшу гнучкість та масштабованість. Це розширило можливості для розгортання баз даних у різних середовищах, включаючи локальні інфраструктури, хмари та контейнери. Ця версія також включала підтримку графових баз даних, що дозволило зберігати та обробляти складні взаємозв'язки між даними.

У 2019 році вийшов SQL Server 2019, який приніс нові можливості для обробки великих даних та інтеграції з різними джерелами даних. Введення Big Data Clusters дозволило поєднувати традиційні реляційні дані з великими даними у єдиній платформі, забезпечуючи потужні інструменти для аналітики та візуалізації. Крім того, підтримка Kubernetes забезпечила ще більшу гнучкість для розгортання баз даних у контейнерних середовищах.

MS SQL також підтримує T-SQL, розширення стандартної SQL, яке включає додаткові можливості для управління транзакціями, обробки помилок та виконання складних запитів. Це дозволяє розробникам писати ефективні та масштабовані запити для роботи з великими обсягами даних, забезпечуючи високу продуктивність додатків.

Завдяки своїй надійності, масштабованості та багатофункціональності, MS SQL залишається популярним вибором для розробників, які прагнуть забезпечити стабільну роботу своїх баз даних та ефективне управління даними. Постійний

розвиток та впровадження нових технологій роблять MS SQL потужним інструментом для сучасних потреб бізнесу.

2.8. Архітектури мобільних застосунків

Архітектура програмного забезпечення — це структура системи, яка визначає компоненти програмного забезпечення, їхні відносини та взаємодії. Вона включає вибір архітектурних стилів та шаблонів, що забезпечують правильну організацію системи, її масштабованість, продуктивність, надійність та гнучкість. Для мобільних застосунків існує кілька основних архітектурних підходів, кожен з яких має свої переваги та недоліки. Основні архітектури для проектування мобільних застосунків:

MVC (Model-View-Controller) — це архітектурний патерн, який розділяє застосунок на три основні компоненти: модель, вид і контролер. Модель відповідає за управління даними та бізнес-логікою, вид (View) — за відображення інформації користувачеві, а контролер (Controller) керує взаємодією між моделлю та видом, обробляючи вхідні дані від користувача і викликаючи відповідні методи моделі. Основна перевага MVC полягає в чіткому розділенні відповідальностей, що спрощує розробку, тестування та підтримку коду. Це дозволяє різним розробникам працювати паралельно над різними компонентами застосунку, що підвищує продуктивність команди. MVC також сприяє повторному використанню коду, оскільки модель та вид можуть бути змінені незалежно один від одного. Однак, мінусом MVC є його складність, особливо для новачків, а також необхідність більш ретельного планування структури застосунку, що може збільшити початкові витрати на розробку.

Переваги:

- Чітке розділення відповідальностей між компонентами.
- Спрощення паралельної розробки та підтримки коду.
- Полегшення повторного використання компонентів.

Недоліки:

- Підвищена складність для новачків.
- Необхідність ретельного планування структури застосунку.
- Потенційні проблеми з масштабуванням у великих проектах.

MVVM (Model-View-ViewModel) — це архітектурний патерн, який покращує MVC, додавши компонент ViewModel. Модель (Model) продовжує відповідати за дані та бізнес-логіку, вид (View) — за інтерфейс користувача, а ViewModel забезпечує зв'язок між моделлю та видом. ViewModel містить логіку презентації та працює з даними моделі, надаючи їх у вигляді, зручному для відображення у виді. Основна перевага MVVM полягає в поліпшенні тестованості та повторному використанні компонентів. ViewModel можна тестувати окремо від виду, що спрощує написання тестів і підвищує якість коду. Крім того, MVVM забезпечує кращу підтримку двостороннього зв'язку (data binding), що спрощує синхронізацію даних між моделлю та видом. Однак, недоліками MVVM є складність налаштування двостороннього зв'язку та потенційні проблеми з продуктивністю у великих застосунках через збільшення обсягу обчислень у ViewModel.

Переваги:

- Покращена тестованість завдяки розділенню логіки презентації та інтерфейсу.
- Підтримка двостороннього зв'язку між моделлю та видом.
- Легка інтеграція з інструментами для роботи з даними.

Недоліки:

- Складність налаштування двостороннього зв'язку.
- Потенційні проблеми з продуктивністю у великих проектах через обсяг обчислень у ViewModel.
- Підвищена складність структури коду.

MVP (Model-View-Presenter) — це архітектурний патерн, схожий на MVC, але з деякими важливими відмінностями. У MVP модель (Model) відповідає за дані та бізнес-логіку, вид (View) — за інтерфейс користувача, а презентер (Presenter) керує взаємодією між моделлю та видом, обробляючи події від користувача та викликаючи відповідні методи моделі. Презентер також оновлює вид у відповідь на зміни в моделі. Основною перевагою MVP є те, що він забезпечує кращу модульність та розділення відповідальностей. Презентер не має прямого доступу до виду, що спрощує тестування, оскільки можна легко перевірити логіку презентера без залежності від конкретної реалізації виду. MVP також забезпечує кращу підтримку великомасштабних проектів, дозволяючи командам розробників працювати незалежно над різними частинами застосунку. Недоліками MVP є збільшення складності коду та необхідність більш ретельного управління життєвим циклом презентера, що може ускладнити розробку та підтримку застосунку.

Переваги:

- Краща модульність та розділення відповідальностей.
- Полегшене тестування презентера без залежності від виду.
- Підтримка великомасштабних проектів завдяки незалежній розробці компонентів.

Недоліки:

- Збільшення складності коду.
- Необхідність ретельного управління життєвим циклом презентера.
- Можливість виникнення проблем із підтримкою складних взаємодій між моделлю та видом.

2.9. Вимоги до програмного забезпечення

Загальні вимоги

Назва проєкту: «Drop me recipe»

Опис проекту: Мобільний застосунок призначений для пошуку та генерації рецептів на основі наявних інгредієнтів або обмеженого бюджету. Застосунок допоможе користувачам підтримувати правильне харчування, пропонуючи здорові та збалансовані рецепти.

Цільова аудиторія:

- Користувачі, які прагнуть здорового харчування
- Користувачі, які хочуть зекономити час та гроші на приготуванні їжі
- Користувачі, які мають обмежений набір інгредієнтів

Функціональні вимоги

Реєстрація та аутентифікація:

- Користувач повинен мати можливість реєструватися за допомогою електронної пошти та паролю.
- Користувач повинен мати можливість входити в систему за допомогою своїх облікових даних.

Профіль користувача:

- Користувач повинен мати можливість створити профіль, де будуть відображатися його рецепти.

Пошук рецептів:

- Користувач повинен мати можливість здійснювати пошук рецептів за наявними інгредієнтами.

Генерація рецептів:

- Застосунок повинен мати функцію генерації рецептів на основі заданих інгредієнтів.

Інформація про рецепти:

- Кожен рецепт повинен містити детальний список інгредієнтів, інструкції з приготування, час приготування та харчову цінність.
- Користувач повинен мати можливість переглядати та зберігати рецепти у своїх обраних.

Нефункціональні вимоги

Інтерфейс користувача:

- Застосунок повинен мати мінімалістичний зрозумілий інтерфейс, оптимізований для мобільних пристроїв.
- Дизайн повинен бути адаптивним, підтримуючи різні розміри екрану та орієнтації.

Продуктивність:

- Застосунок повинен швидко реагувати на дії користувача, з мінімальним часом завантаження сторінок та пошуку рецептів.
- Застосунок повинен працювати стабільно без збоїв та аварійних завершень.

Конфіденційність:

- Застосунок повинен забезпечувати безпечне зберігання даних користувачів, включаючи паролі та особисту інформацію.

Сумісність:

- Застосунок повинен підтримувати останні версії iOS та Android.

3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1. Головний екран застосунку

Головний екран застосунку – це екран у який користувач потрапляє відразу при вході у застосунок. З цього екрану починається навігація по застосунку.

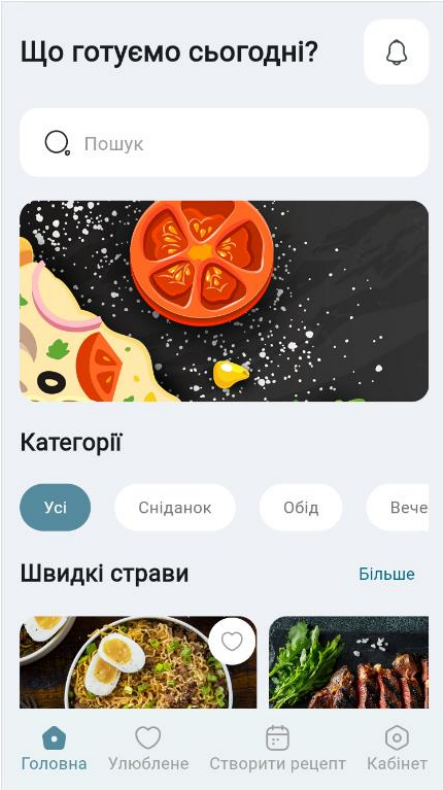


Рис.3.1.1 Головний екран застосунку

На рисунку 3.1 можна побачити головний екран, він виконаний у світлому мінімалістичному форматі, з додаванням яскравих елементів інтерфейсу, які є максимально зрозумілими та мають приємний для ока стиль. Інтерфейс побудовано на базі віджетів. Віджети є основними будівельними блоками у Flutter і можуть представляти як прості елементи, наприклад кнопки або текстові поля, так і складніші компоненти, наприклад форми, списки, макети. Основний віджет, який надає

структуру всьому застосунку це MaterialApp. Він надає структуру та дизайн у стилі Material Design. Каркасом слугує віджет Scaffold, який містить у собі такі основні елементи інтерфейсу як AppBar, Body, BottomNavigationBar. Згори рисунку зображено вітальний текст з запитання до користувача «Що готуємо сьогодні?» та іконка сповіщень. Дану частину інтерфейсу, а тобто панель, було реалізовано за допомогою кастомного віджету HomeAppbar. Двигаючись донизу можемо побачити пошукову строку, що містить текстове поле для введення ключових слів для пошуку рецепту. Реалізовано було також через кастомний віджет HomeSearchBar. Так як дані зберігаються на віддаленому сервері, пошук відбувається за допомогою REST API. Для інтеграції можливості виконання http-запитів у застосунок, було використано бібліотеку “http”.

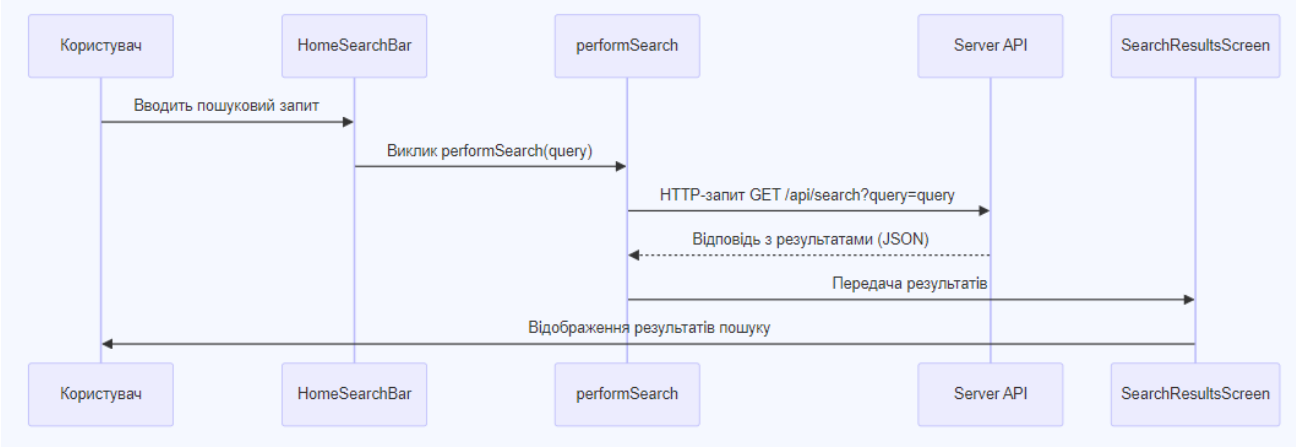


Рис.3.1.2 Послідовна діаграма відправки запиту на сервер

По центру екрану можемо бачити категорії. В залежності від категорії, отримуємо список рецептів, які більше підходять на сніданок, обід або вечерю. Вибір категорії змінює стан застосунку (StatefulWidget), що дозволяє динамічно оновлювати зміст.

Список швидких страв: кастомний виджет, який відображає список рецептів страв, приготування яких не займе багато часу, що дозволяє користувачам швидко знаходити натхнення для приготування їжі. Останнє що можна помітити на рисунку

це нижня навігаційна панель з іконками та їх описом, що надає швидкий доступ до основних розділів: головна сторінка, улюблені рецепти користувача, екран для генерації рецепту та особистий кабінет.

3.2. Екран «Швидкі страви»

Екран «Швидкі страви» - це екран, на якому подано список рецептів, кожний рецепт складається з карток, на картці зображено фото готової страви, її назва, калорійність, час приготування, та оцінки від користувачів. У правому кутку кожної картки є поле, у форматі «лайку», користувач може поставити відмітку, якщо йому сподобався і зацікавив рецепт. При цьому рецепт збережеться до списку «Улюблене».

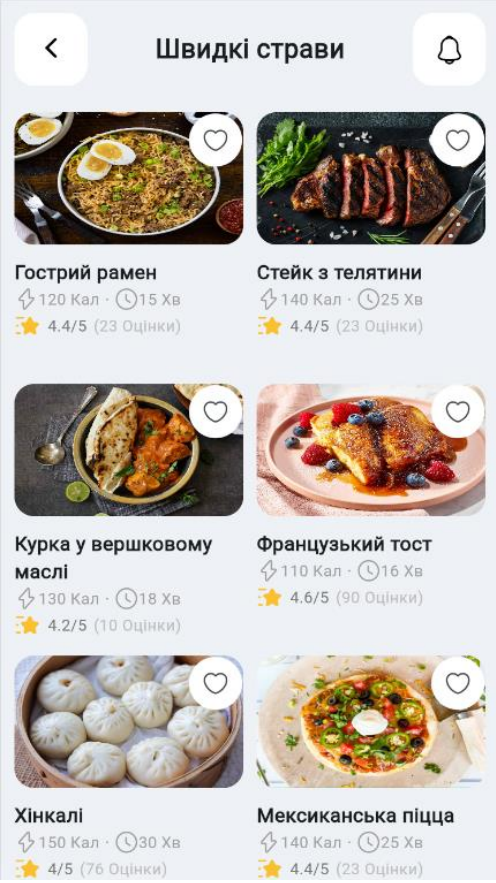


Рис.3.2.1 Екран «Швидкі страви»

Перехід на даний екран здійснюється після натискання на головному екрані (пункт 3.1) кнопки «більше», за допомогою класу для навігації Navigator. Екран, який зображений на рисунку 3.3 має схожий макет, що і екран зображений на рисунку 3.1, також нейтральний сірий фон та схожу панель AppBar, де знаходиться іконка повідомлень. Додатково на панелі у верхньому лівому кутку знаходиться кнопка у вигляді стрілочки. Ця кнопка відповідає за перехід на екран, з якого був зроблений перехід на поточний екран.

3.3.Екран рецепту

Екран рецепту дає доступ користувачу до екрану з рецептом, що він обрав. На даному екрану зображено фото готової страви у великому форматі, щоб користувач мав змогу детально ознайомитися з зовнішнім виглядом страви, яку в подальшому приготує.

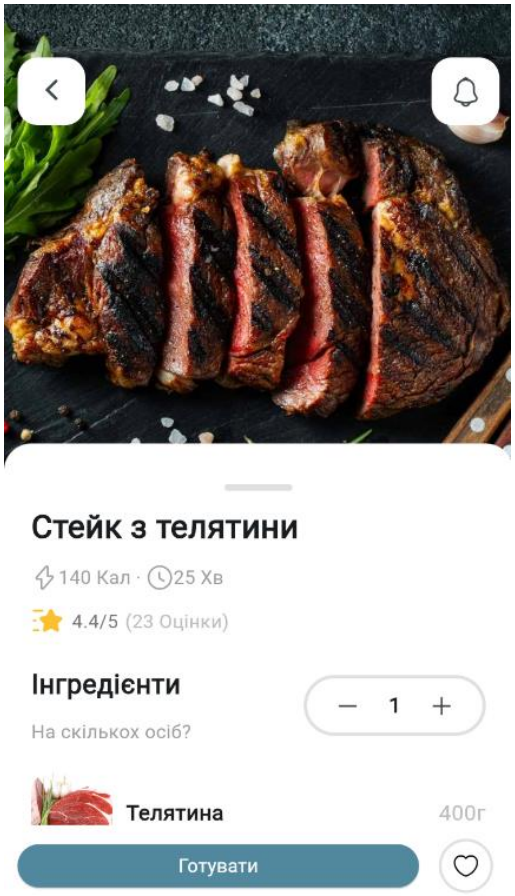


Рис.3.3.1 Екран рецепту

У самому верху екрану знаходиться та сама панель, що і на екрані, на рисунку 3.3. На нижній половині екрану знаходиться картка рецепту. На ній відображена детальна інформація про страву: її назва, калорійність, час приготування та оцінки користувачів. Трохи нижче знаходиться поле інгредієнти, зі списком продуктів, потрібних для приготування. Також є поле з лічильником порцій, в залежності від числа на лічильнику, кількість інгредієнтів, потрібних для приготування, змінюється. При натисканні кнопки «Готувати», що знаходиться у самому низу екрану, користувача перенаправляє на екран з кроками приготування.

3.4.Екран «Улюблене»

При переході на даний екран, користувач має змогу побачити список рецептів, які він уподобав, проставивши відмітки у форматі «лайку» на рецепти, що йому сподобалися. Користь цього екрану полягає у можливості не загубити рецепти і в разі чого повернутися до них, щоб мати змогу приготувати улюблену страву ще раз.

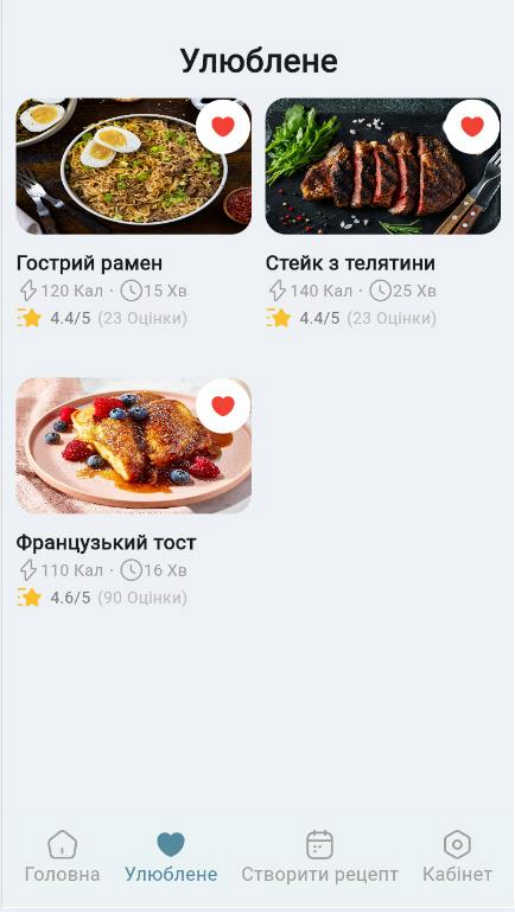


Рис.3.4.1 Екран «Улюблене»

Зверху ми бачимо заголовок екрану, як і на інших екранах це реалізовано за допомогою зручного віджету AppBar. Структуризація рецептів у таблицю відбувається за допомого віджету GridView.builder. Цей віджет також забезпечує можливість прокручувати елементи, не задіваючи інші віджети і не порушуючи їх структуру. Логіка отримання списку улюблених рецептів: при переході на даний

екран, користувач ініціює запит, мобільний застосунок надсилає http-запит до REST API у форматі JSON з логіном користувача. Кодування та декодування у JSON відбувається за допомогою методів json.encode() та json.decode(), інтегрований у пакет dart:convert. Після отримання даних REST API обробляє запит і виконує SQL-запит до бази даних, щоб отримати список рецептів для конкретного користувача. База даних виконує запит і повертає відповідні записи з таблиці FavouriteRecipe, що містить посилання на таблицю Users. REST API форматує результати у вигляді JSON і повертає їх у мобільному застосунку. Мобільний застосунок відображає отримані дані у вигляді списку.

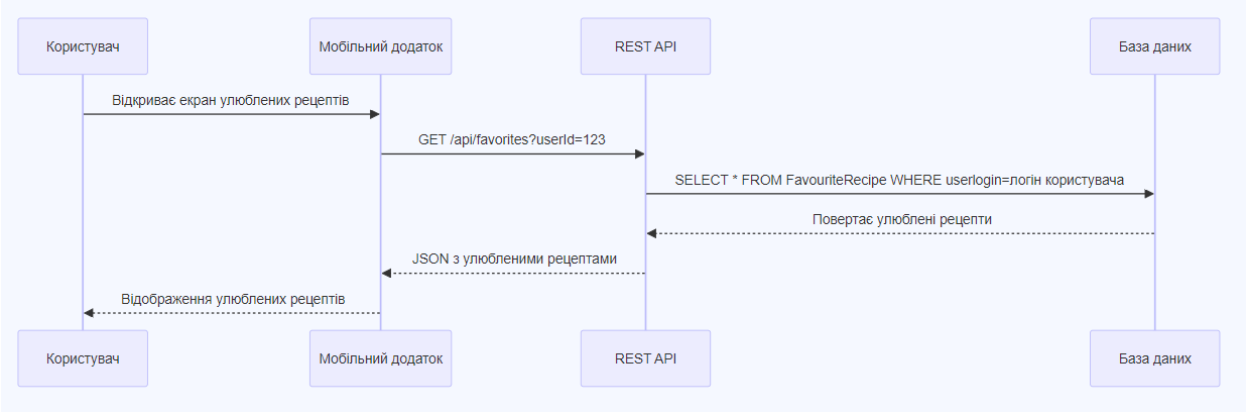


Рис.3.4.2 Діаграма послідовності отримання списку улюблених рецептів

3.5. Екран реєстрації та входу

Незважаючи на те, що основна частина функцій застосунку доступна усім користувачам, у застосунку існує особистий кабінет та можливість реєстрації та авторизації користувачів. На екрані особистого кабінету, при умові якщо користувач не увійшов до системи, видніється форма входу до системи.

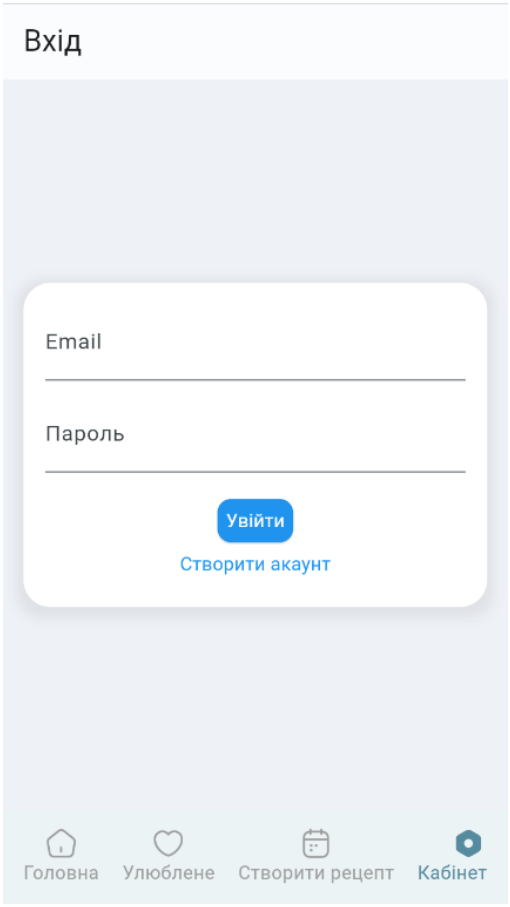


Рис.3.5.1 Екран особистого кабінету (вхід)

При умові якщо у користувача немає особистого кабінету, він може його створити на екрані реєстрації. При натисканні кнопки «створити акаунт» форма змінюється на формат реєстрації. Реєстрація, як і вхід до застосунку відбувається за допомогою електронної адреси та паролю.

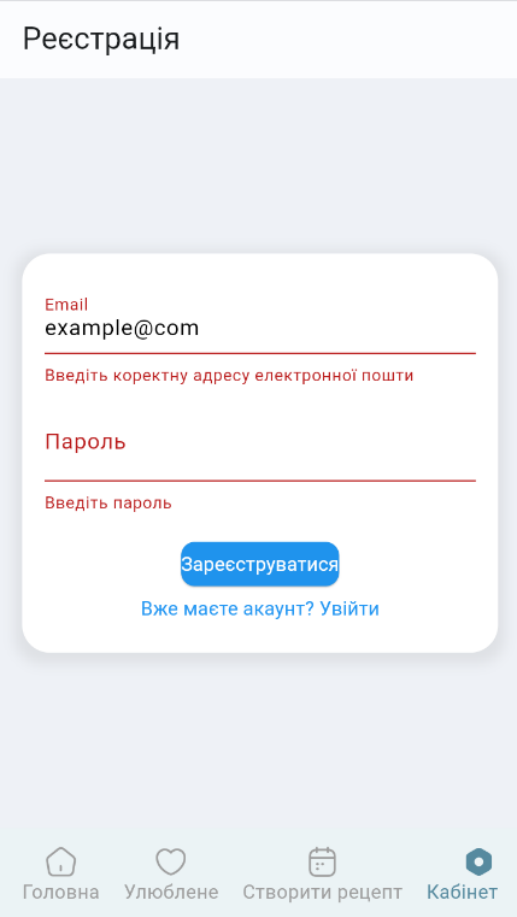


Рис.3.5.2 Екран особистого кабінету (реєстрація)

Як при вході, так і при реєстрації відбувається перевірка на валідність введених даних. Якщо пошта введена у невірному форматі, користувач отримає повідомлення що йому потрібно вказати коректну адресу електронної пошти. Перевірка відбувається за рахунок вбудованого у Flutter методу RegExp(). Також відбувається перевірка на те, щоб поле з паролем не було рівне NULL.

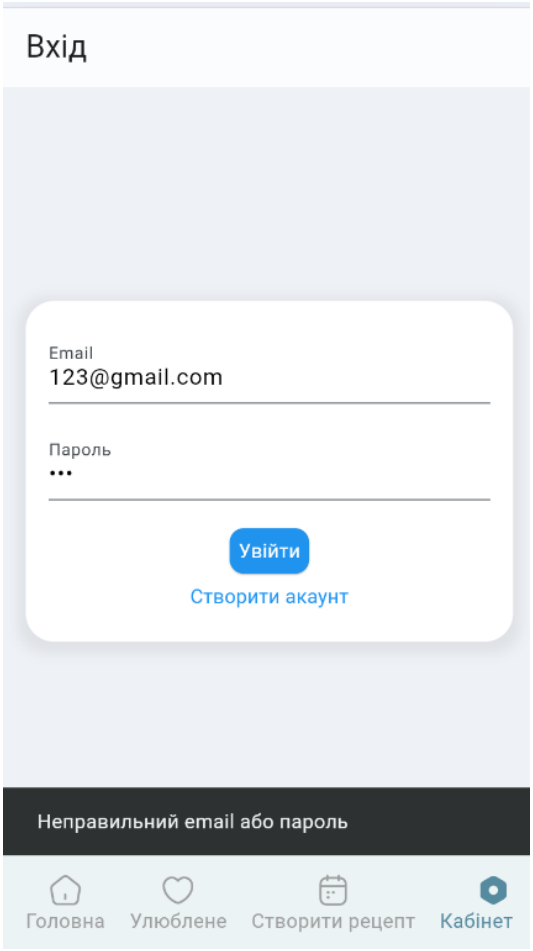


Рис.3.5.3 Інформаційне повідомлення при вході в неіснуючий акаунт.

Якщо користувач спробує увійти в акаунт, якого не знайдено. Він отримає інформаційне повідомлення, що він ввів неправильну пошту або пароль. Відображення інформаційного повідомлення відбувається за допомогою зручного вбудованого методу SnackBar(), який приймає у себе поле з текстом, а також час Duration(), протягом якого буде показане повідомлення.

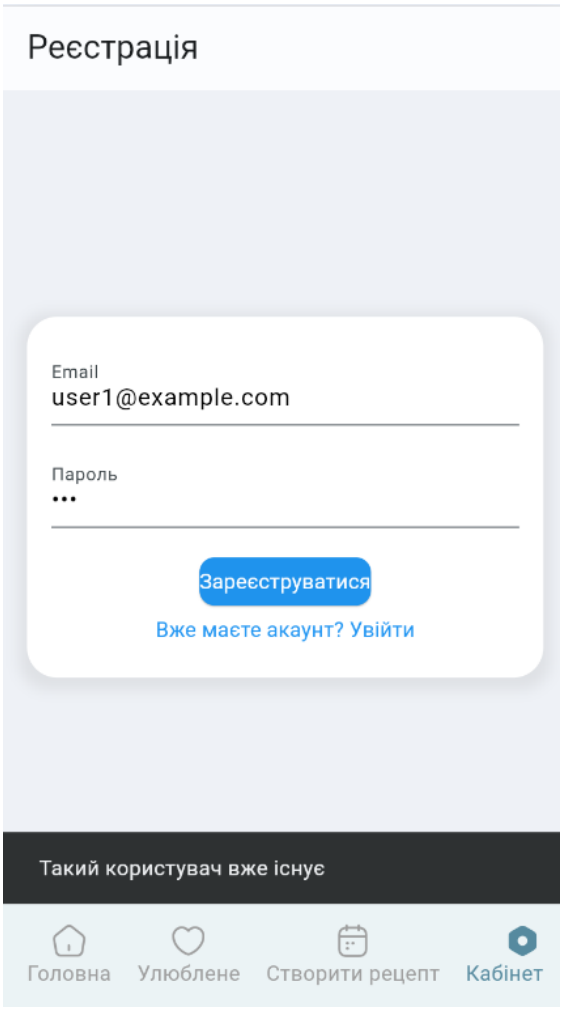


Рис.3.5.4 Інформаційне повідомлення при спробі зареєструвати вже існуючий акаунт.

При спробі створити акаунт користувача, який вже існує в базі даних, користувач отримає інформаційне повідомлення «Такий користувач вже існує». Аналогічно з повідомленням на рисунку 3.9 його відображення відбувається за допомогою методу SnackBar().

3.6. Екран особистого кабінету

Екран особистого кабінету має мінімалістичний, стандартний дизайн. На ньому знаходиться основна інформація про користувача та його рецепти.

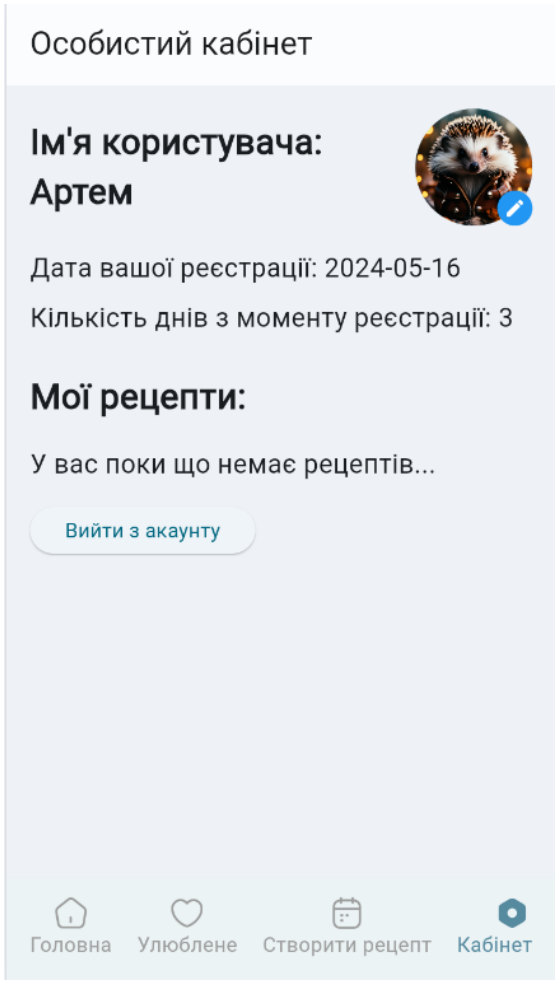


Рис.3.6.1 Особистий кабінет користувача.

Угорі екрану знаходиться заголовок екрану, як і на інших екранах це реалізовано за допомогою зручного віджету AppBar. Далі йде поле з ім’я, яке ми можемо відредагувати, натиснувши на нього. Праворуч знаходиться аватар, який користувач може змінити за бажанням, завантаживши фото. Збереження аватару відбувається за допомогою базу даних та REST API. У базі даних зберігається шлях до файлу на сервері. Нижче користувач має змогу побачити дату його реєстрації у застосунку, а також кількість днів, з моменту реєстрації. По центру екрану знаходиться список згенерованих користувачем рецептів. І останній елемент на екрані це кнопка виходу з облікового запису.

3.7. Екран генерації рецептів

Основною ідеєю застосунку була реалізація екрану застосунку для генерації рецептів, де користувач може обрати наявні інгредієнти в нього вдома, та отримати згенерований програмою рецепт.

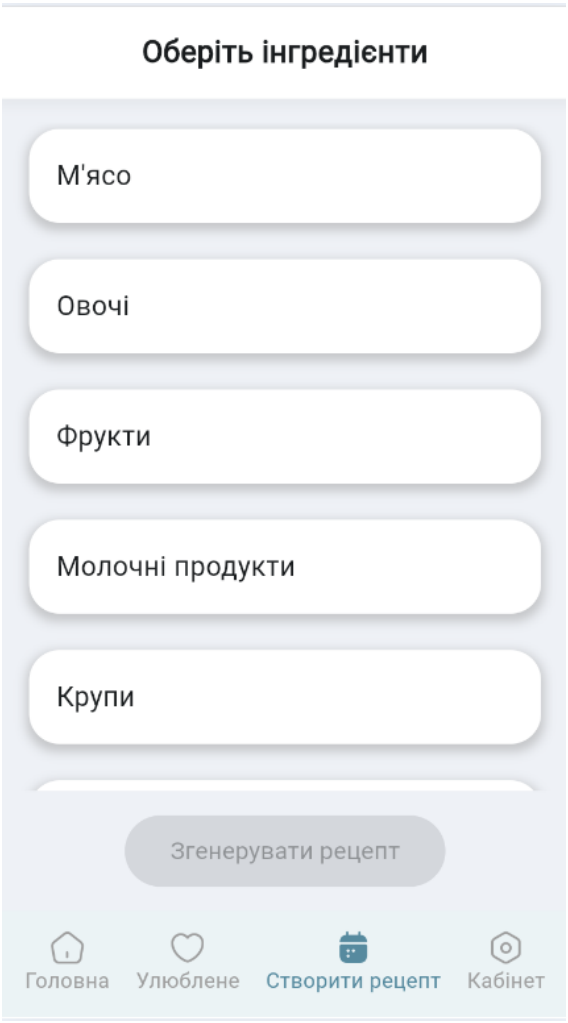


Рис.3.7.1 Екран зі списком продуктів за категоріями для генерації рецепту.

На екрані для генерації рецептів користувач бачить верхню панель з текстом де йому пропонується обрати інгредієнти. Нижче на основній частині екрану зображено категорії продуктів, розгорнувши які він може обрати наявні в нього вдома інгредієнти (Рис. 3.13). Поки користувач не обрав жодного інгредієнту, кнопка «Згенерувати рецепт» не може бути активною.

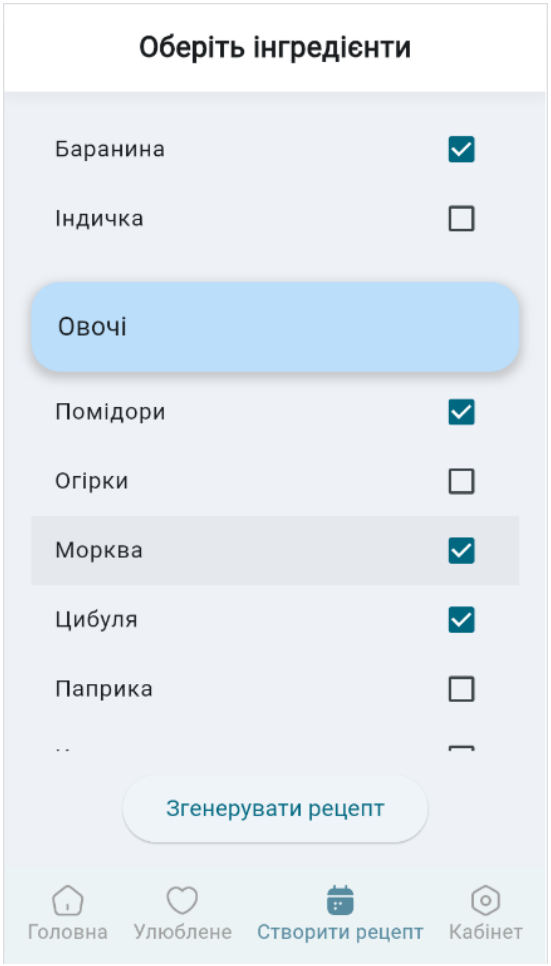


Рис.3.7.2 Список обраних користувачем інгредієнтів.

Після того як користувач обрав потрібні йому для рецепту інгредієнти, він повинен натиснути кнопку «Згенерувати рецепт», яка вже стала активною. Застосунок формує запит, враховуючи обрані інгредієнти та надсилає його до сервісу зі штучним інтелектом. Підключення до сервісу відбувається за допомогою APIKey та APIUrl. Запит надсилається у форматі JSON.

```
String apiUrl = 'https://api.openai.com/v1/chat/completions';
String apiKey = 'sk-proj-815Xl30l1985Tk4zjmeWT3BlbkFJbJaP5j0FjK7hYeXElyyu';

Map<String, String> headers = {
  'Content-Type': 'application/json',
  'Authorization': 'Bearer $apiKey',
};

Map<String, dynamic> body = {
  "model": "gpt-4o",
  "messages": [
    {
      "role": "user",
      "content": 'Згенеруй рецепт страви, з наявності у мене тільки ці продукты: ${_selectedProducts.join(', ')
    }
  ]
};

try {
  final response = await http.post(
    Uri.parse(apiUrl),
    headers: headers,
    body: json.encode(body),
  );
```

Рис.3.7.3 Запит до сервісу.

Коли рецепт згенеровано, сервіс повертає вже готовий рецепт (Рис. 3.16), який попередньо потрібно декодувати, адже відповідь надходить у JSON-форматі. Для цього використовується метод json.decode(), інтегрований у пакет dart:convert

```
if (response.statusCode == 200) {
  final jsonResponse = json.decode(utf8.decode(response.bodyBytes));
  print(response.body);

  if (jsonResponse.containsKey('choices')) {
    final responseData = jsonResponse['choices'][0]['message']['content'];
    if (responseData != null) {
      print(_selectedProducts);
      // Перехід на екран RecipeAnswer з отриманою відповіддю
      Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => RecipeAnswer(responseData)),  // MaterialPageRoute
      );
```

Рис.3.7.4 Процес декодингу отриманої відповіді з врахуванням Юнікоду UTF-8

3.8. Екран згенерованого рецепту

На екрані згенерованого рецепту знаходиться верхня панель з іконкою закладки, натиснувши яку, ми можемо назвати наш рецепт та зберегти до особистого кабінету(Рис. 3.19). В основній частині екрану знаходиться поетапно розписаний по крокам рецепт приготування, а також уточнення кількості інгредієнтів потрібних для приготування. Вкінці рецепту знаходиться інформація по часу приготування та кількості порцій. Від даної інформації користувач може відштовхуватись і корегувати кількість інгредієнтів наявних у нього вдома.

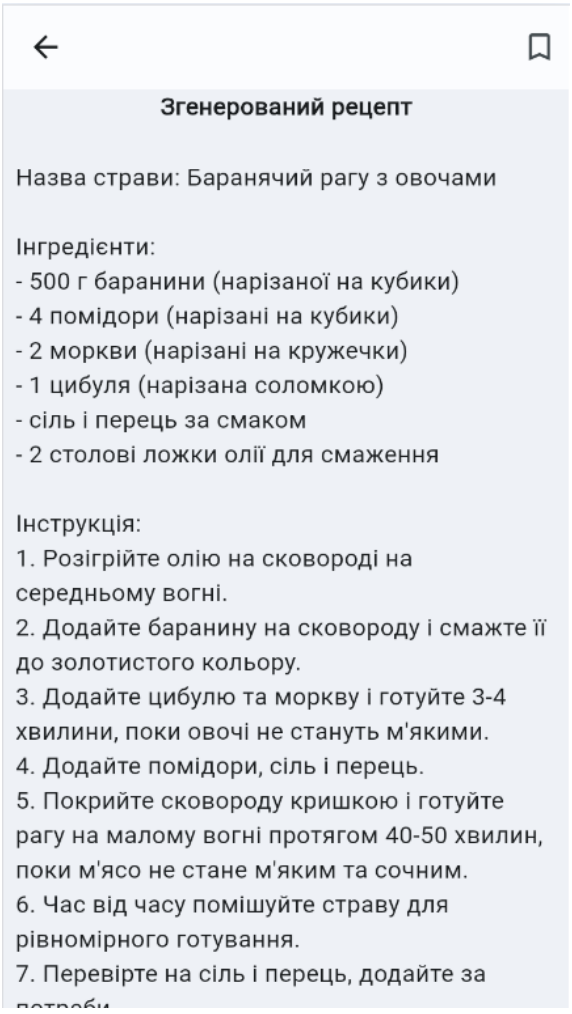


Рис.3.8.1 Згенерований рецепт

При створенні закладки користувач може ввести назву свого рецепту, яка йому до вподоби та легко запам’ятовується. Вона з’явиться у верхній частині екрану, прямо над згенерованим рецептом. Поле для вводу даних створено на базі методу у Flutter – AlertDialog().

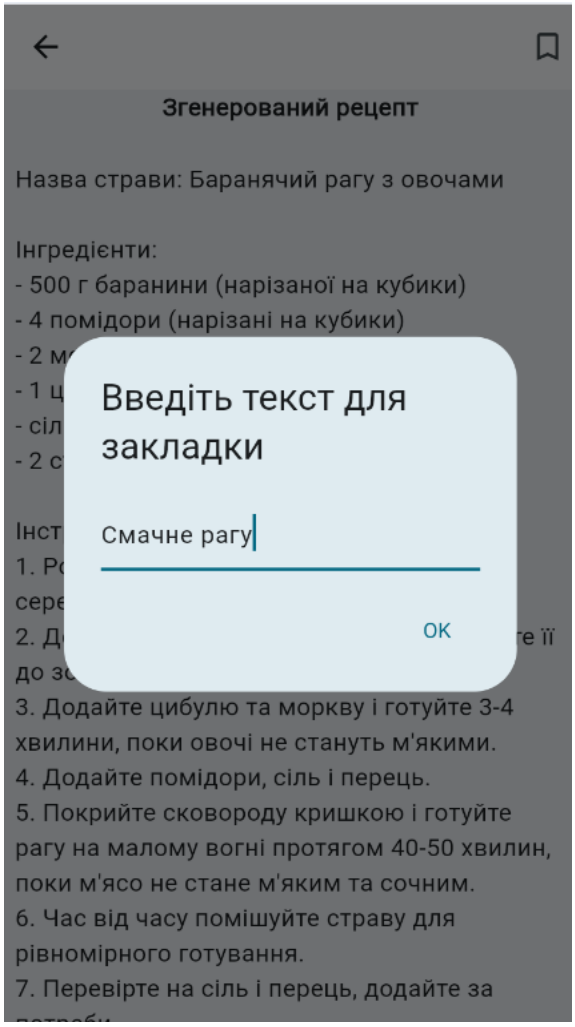


Рис.3.8.2 Поле для вводу назви закладки.

Після того, як користувач ввід текст та натиснув кнопку «ок», поле закриється, закладка створиться, а рецепту буде присвоєна назва.

Рис.3.8.3 Створена закладка.

Додатково рецепт буде додано до екрану з особистими кабінетом, що був представлений раніше.

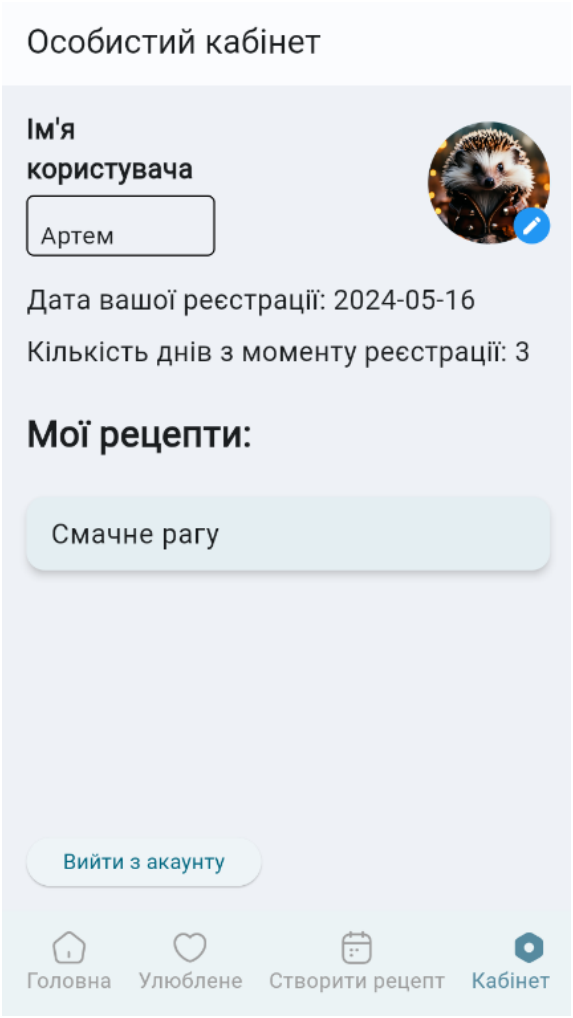


Рис.3.8.4 Рецепт додався до списку «Мої рецепти».

4. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ ЗАСТОСУНКУ

4.1. Тестування системи

Тестування є критично важливим етапом розробки програмного забезпечення, що дозволяє забезпечити його надійність, коректність та продуктивність. Для застосунку, що здійснює пошук і генерацію рецептів, важливою частиною тестування є юніт тести. Вони допомагають перевірити правильність функціонування окремих компонентів системи, таких як алгоритми пошуку, генерації рецептів та інтерфейси користувача. У цьому розділі описано основні принципи та методи тестування, а також надано приклади юніт тестів для застосунку для пошуку та генерації рецептів.

Тестування програмного забезпечення включає в себе різні методи, серед яких юніт тести, інтеграційні тести, системні тести та приймальні тести. Юніт тести фокусуються на перевірці окремих модулів або компонентів програми ізольовано від інших частин системи. Це дозволяє виявити помилки на ранніх етапах розробки та забезпечити стабільність коду.

Інтеграційні Тести

Інтеграційні тести перевіряють взаємодію між різними модулями або компонентами системи. Вони допомагають виявити дефекти, які виникають через взаємодію компонентів, навіть якщо кожен з них працює коректно окремо. Основна мета інтеграційного тестування — забезпечити, щоб всі компоненти системи працювали разом як єдине ціле.

Інтеграційне тестування зазвичай проводиться після юніт тестування. Воно може використовувати різні підходи: зверху вниз, знизу вверх або змішаний підхід. Підхід зверху вниз починається з тестування високорівневих компонентів і поступово переходить до низькорівневих, в той час як підхід знизу вверх робить протилежне.

Інтеграційні тести охоплюють ширший спектр функціональності, ніж юніт тести. Вони можуть включати перевірку взаємодії між модулями, інтерфейсів API, баз даних та інших зовнішніх систем. Наприклад, у застосунку для пошуку рецептів інтеграційні тести можуть перевіряти, чи правильно взаємодіють модулі пошуку рецептів з базою даних та інтерфейсом користувача.

Системні Тести

Системні тести перевіряють всю систему в цілому, включаючи всі її компоненти та підсистеми. Цей вид тестування спрямований на перевірку відповідності програмного забезпечення початковим вимогам. Системне тестування є комплексним і охоплює як функціональні, так і нефункціональні аспекти системи.

Основні характеристики системного тестування включають комплексність, відповідність вимогам та охоплення різних аспектів системи. Комплексність означає, що тестування охоплює всю систему, включаючи апаратне забезпечення, програмне забезпечення та мережеві компоненти. Відповідність вимогам перевіряє, чи відповідає система специфікаціям та бізнес-вимогам. Охоплення різних аспектів включає функціональне тестування, перевірку безпеки, тестування продуктивності та інші види тестування.

Системне тестування зазвичай проводиться після інтеграційного тестування. Воно включає перевірку всіх функцій системи з точки зору користувача. Наприклад, для застосунку для пошуку рецептів системні тести можуть включати перевірку процесів реєстрації користувачів, пошуку рецептів, збереження обраних рецептів та генерації нових рецептів.

Приймальні Тести

Приймальні тести (acceptance testing) проводяться для визначення, чи відповідає система вимогам і чи готова вона до здачі замовнику або кінцевим користувачам. Приймальне тестування проводиться в реальних або близьких до реальних умовах експлуатації і часто залучає замовників або кінцевих користувачів.

Основні характеристики приймального тестування включають верифікацію бізнес-вимог, залучення замовника та різні типи приймальних тестів. Верифікація бізнес-вимог перевіряє, чи відповідає програмне забезпечення бізнес-вимогам і чи виконує воно очікувані бізнес-функції. Залучення замовника означає, що замовники або кінцеві користувачі часто беруть участь у цьому виді тестування. Приймальні тести можуть включати альфа-тестування (внутрішнє тестування з залученням команди розробки) і бета-тестування (зовнішнє тестування з залученням реальних користувачів).

Приймальні тести проводяться на завершальному етапі розробки програмного забезпечення. Вони перевіряють повний функціональний спектр системи в умовах, близьких до реальних. Наприклад, для застосунку для пошуку рецептів приймальні тести можуть включати перевірку всіх функціональних можливостей, від реєстрації та входу користувача до пошуку та збереження рецептів, генерації нових рецептів та роботи з інтерфейсом користувача.

Юніт тести

Юніт тести — це метод тестування програмного забезпечення, при якому окремі модулі або компоненти перевіряються на коректність роботи. Юніт тести фокусуються на найменших частинах коду, таких як функції або методи класів. Основна мета юніт тестів полягає в тому, щоб перевірити, чи працює код правильно в ізоляції від інших частин системи.

Основні характеристики юніт тестів включають ізольованість, автоматизацію, повторюваність та незалежність. Ізольованість означає, що юніт тест перевіряє лише одну функцію або метод, використовуючи моки чи стаби для підміни залежностей. Автоматизація забезпечує можливість швидкого та регулярного виконання тестів. Повторюваність гарантує однакові результати при кожному виконанні тесту за тих самих умов. Незалежність означає, що тести не повинні залежати один від одного, що дозволяє їх виконувати в будь-якому порядку.

Юніт тести мають кілька переваг. Вони допомагають виявляти помилки на ранніх етапах розробки, що знижує витрати на їх виправлення. Тестування сприяє написанню чистого та добре структурованого коду, оскільки розробники змушені враховувати тестованість своїх модулів. Автоматизація тестів дозволяє розробникам швидше вносити зміни в код, впевнені у відсутності регресій. Юніт тести можуть також служити додатковою документацією, пояснюючи, як саме має працювати код у різних ситуаціях.

Для написання юніт тестів у Flutter використовується пакет test, який дозволяє створювати та виконувати тести. Основні принципи тестування залишаються такими ж, як і в інших мовах програмування: ізольованість, автоматизація, повторюваність та незалежність тестів.

Нижче наведено приклад коду юніт тесту.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:food_example/models/Products.dart';
import 'package:food_example/screens/recipe_generate.dart';

void main() {
  testWidgets('getProductList returns correct product widgets', (WidgetTester tester) async {
    // Створення видимого екземпляру RecipeGenerate
    final recipeGenerate = RecipeGenerate();

    // Додавання видимого екземпляру до тестового віджету
    await tester.pumpWidget(MaterialApp(home: recipeGenerate));

    // Отримання стану видимого екземпляру RecipeGenerate
    final state = tester.state<RecipeGenerateState>(find.byType(RecipeGenerate));

    List<Product> meatProducts = [
      Product('Яловичина', 'М\'ясо'),
      Product('Свинина', 'М\'ясо'),
    ];

    // Присвоєння meatProducts в стані
    state.setMeatProducts(meatProducts);

    // Виклик методу getProductList
    List<Widget> widgets = state.getProductList('М\'ясо');

    // Перевірка кількості повернених віджетів
    expect(widgets.length, 2);

    // Перевірка назв продуктів у віджетах
    expect((widgets[0] as CheckboxListTile).title.toString().contains('Яловичина'), true);
    expect((widgets[1] as CheckboxListTile).title.toString().contains('Свинина'), true);
  });
}
```

Рис.4.1.1 Приклад коду для юніт тесту

Рис.4.1.2 Результат виконання юніт тесту

4.2. Налагодження програмного коду

Android Studio є основним середовищем розробки для створення мобільних застосунків під операційну систему Android. Це потужне та функціональне середовище, яке надає розробникам широкі можливості для налагодження коду та виправлення помилок. Налагодження в Android Studio забезпечується різними інструментами та методами, що дозволяють ефективно ідентифікувати та вирішувати проблеми в коді.

Одним із основних інструментів для налагодження є лог-консоль, яка відображає повідомлення в режимі реального часу. Лог-консоль Android Studio використовує систему журналів Android (Logcat), яка дозволяє розробникам отримувати докладну інформацію про виконання програми. Вона відображає різні рівні повідомлень, включаючи помилки, попередження, інформаційні повідомлення та відлагоджувальні повідомлення. Розробники можуть додавати власні повідомлення до журналу, використовуючи методи Log.d(), Log.e(), Log.i(), Log.w() та інші.

Кожного разу, коли розробник натискає кнопку «Build» в Android Studio, проект збирається, і всі помилки та попередження в коді відображаються у вікні «Build». Android Studio автоматично компілює код і перевіряє його на наявність помилок, підсвічуючи їх у редакторі коду. Синтаксичні помилки підсвічуються червоним кольором, і часто середовище надає можливість автоматичного виправлення цих помилок за допомогою швидких дій (Quick Fix).

Якщо візуальна частина інтерфейсу створюється за допомогою XML-розмітки або інструменту Layout Editor, Android Studio також перевіряє правильність розміщення UI-елементів. Середовище може вказувати на потенційні проблеми з констрейнтами (обмеженнями) або некоректним використанням ресурсів.

Для налагодження логіки програми розробники можуть використовувати кілька методів, таких як Log, Debug, breakpoints та інші. Метод Log.d() використовується для

виведення діагностичних повідомлень у лог-консоль. Це дозволяє розробнику бачити, які значення мають змінні в певний момент часу, і які шляхи виконання вибирає програма.

Breakpoints (точки зупинки) є потужним інструментом для налагодження, який дозволяє призупинити виконання програми в певному місці і аналізувати стан змінних та виконання коду на цьому етапі. Розробник може додавати breakpoints у потрібних місцях коду і використовувати вікно Debug для перегляду значень змінних, стека викликів і багато іншого.

Також доступні різні рівні повідомлень у системі логування:

- Попереджувальні помилки — повідомлення, що вказують на некритичні проблеми, які можуть не завадити компіляції та запуску програми, але можуть впливати на її поведінку або продуктивність.
- Критичні помилки — повідомлення про серйозні помилки, які заважають компіляції програми і вимагають негайного виправлення.
- Помилки потоків — повідомлення, що вказують на проблеми з багатопотоковою розробкою, наприклад, некоректний доступ до спільних ресурсів з різних потоків.

Налагодження програмного коду в Android Studio також включає можливість використання інструментів профілювання, таких як Android Profiler, для аналізу використання ресурсів, продуктивності та поведінки застосунку під час його виконання. Ці інструменти дозволяють ідентифікувати вузькі місця у виконанні коду, проблеми з пам'яттю та інші важливі аспекти, що впливають на якість застосунку.

4.3. Оптимізація дизайну для різних видів мобільних пристроїв

Кожного року у світі з’являється безмежна кількість нових мобільних пристроїв з різними діагоналями екранів, тому розробникам мобільних застосунків та іншого

програмного забезпечення потрібно подбати щоб їхня система добре працювала та мала коректне відображення як на відносно невеликих екранах, так і на відносно великих. При розробці застосунку було оптимізовано правильне відображення UI елементів проєкту. Flutter надає потужні інструменти та методи для досягнення цієї мети. Завдяки своїй архітектурі, Flutter дозволяє створювати адаптивні інтерфейси користувача, які добре виглядають і працюють на різних розмірах екранів та типах пристроїв.

Серед основних методів та віджетів я хотів би виділити Віджет LayoutBuilder, що дозволяє створювати адаптивні макети, реагуючи на розмір свого батьківського контейнера. Це корисно для створення інтерфейсів, які автоматично підлаштовуються під різні розміри екрану. Віджет MediaQuery надає інформацію про розміри та орієнтацію екрану, а також про інші характеристики пристрою. Це дозволяє створювати адаптивні інтерфейси, які можуть змінювати своє розташування та розмір елементів залежно від контексту. Віджети Flexible та Expanded допомагають створювати макети, які можуть автоматично розтягуватися або стискатися відповідно до доступного простору. Віджет AspectRatio дозволяє зберігати пропорції елементів інтерфейсу незалежно від розміру екрану, що забезпечує консистентність дизайну. Віджети GridView та ListView дозволяють створювати адаптивні макети для списків та сіток елементів, які можуть динамічно змінюватися в залежності від розміру екрану.

Дані рішення були використані у розробленому проєкті. На рисунку 4.1 зображено вигляд застосунку на трьох різних моделях мобільних пристроїв зліва направо: Iphone SE, Iphone 12 Pro, Samsung Galaxy S8+.

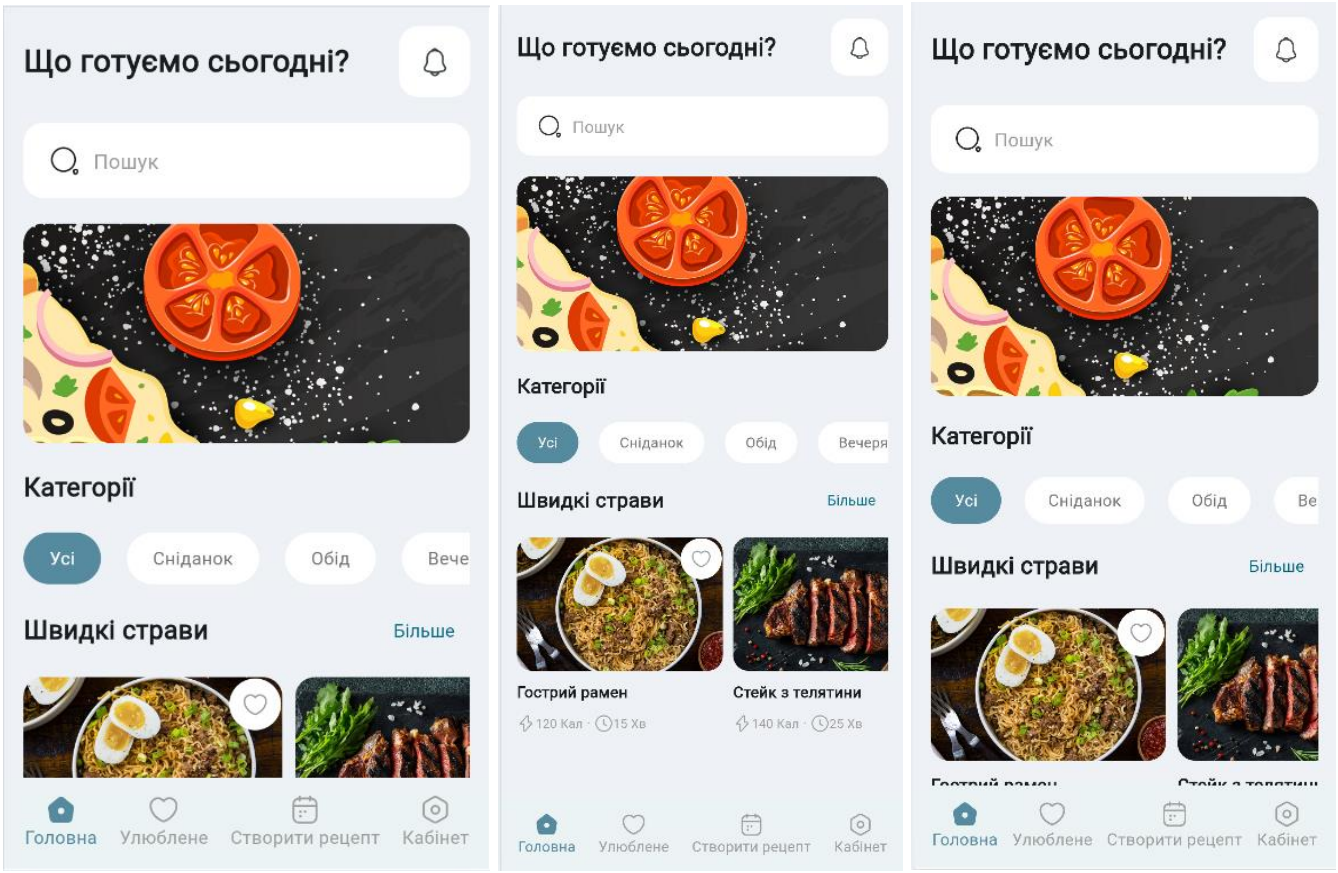


Рис.4.3.1 Огляд оптимізованого UI до різних розмірів дисплеїв

ВИСНОВКИ

Під час розробки мобільного застосунку у дипломній роботі було пройдено всі етапи, а саме:

1. Розглянуто та проаналізовано предметну область проєкту та існуючі аналоги цільового сегменту ринку.

2. На етапі проектування було розроблено архітектуру програмного забезпечення, яка включає використання фреймворку Flutter та мови програмування Dart для розробки фронтенду, а також Node.js для реалізації серверної частини. У якості середовища розробки було обрано VSCode, що забезпечило ефективну роботу з кодом та зручність у налагодженні.

3. Використовуючи інструменти Flutter, було створено мінімалістичний дизайн, який слідує правилам GUI та забезпечує просту навігацію для користувачів. Дизайн інтерфейсу було адаптовано для різних розмірів екранів та типів мобільних пристроїв, що гарантує гарний вигляд та зручність користування на будь-якому пристрої.

4. Розроблено мобільний застосунок за допомогою мови Dart та фреймворку Flutter, а також Node.js. Застосунок є самостійним завершеним продуктом та може використовуватися для серйозних задач.

5. Розроблений мобільний застосунок було піддано нефункціональному тестуванню, в ході якого було виявлено та усунено всі помилки. Застосунок працює стабільно, забезпечує швидкий доступ до інформації та дозволяє легко знаходити та генерувати рецепти.

6. Робота пройшла апробацію на наукових конференціях:

Дубовик А.Ю., Савіцький В.А. Мобільний застосунок для пошуку рецептів по наявних інгредієнтах або обмеженому бюджеті для підтримки правильного харчування. *Всеукраїнська науково-технічна конференція «Застосування*

програмного забезпечення в інформаційно-комунікаційних технологіях». 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С._331_.

Дубовик А.Ю., Савіцький В.А. Аналіз та оптимізація швидкодії мобільного застосунку для ефективного пошуку рецептів у режимі реального часу. *Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях».* 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С._230_.

ПЕРЕЛІК ПОСИЛАНЬ

1. What are the advantages and disadvantages of using Visual Studio Code [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/@ssc.ahmed.926748/what-are-the-advantages-and-disadvantages-of-using-visual-studio-code-or-atom-d3132bf1af85
2. TIOBE Index for May 2024 [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.tiobe.com/tiobe-index/
3. Flutter Vs. React Native In 2024 — Detailed Analysis [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.nomtek.com/blog/flutter-vs-react-native
4. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022 [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/
5. Flutter documentation [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://docs.flutter.dev/
6. Dart documentation [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://dart.dev/guides
7. Why use Node JS for your enterprise project? [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.miquido.com/blog/why-use-node-js
8. Benefits of Microsoft SQL Server: Company Use [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/benefits-microsoft-sql-server-company-use-team-venti-7byoe
9. Software Architecture Patterns [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/@mahmoudIbrahimAbbas/software-architecture-patterns-558486f4c3aa

10. The Pros and Cons of Android Studio and App Tools [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://pangea.ai/resources/pros-and-cons-of-android-studio-and-app-tools

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільного застосунку для пошуку рецептів по наявних інгредієнтах або обмеженому бюджеті для підтримки правильного харчування

Виконав студент 4 курсу
групи ПД-41
Дубовик Артем Юрійович
Керівник роботи
асистент кафедри ІПЗ Савіцький Вячеслав Андрійович
Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - підвищення зручності користувачів при пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.
- **Об'єкт дослідження -** пошук та генерація рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.
- **Предмет дослідження –** мобільний застосунок для пошуку та генерації рецептів за умови обмежених інгредієнтів та бюджету для підтримки правильного харчування.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати предметну галузь пошуку та генерації рецептів.
2. Провести аналіз конкурентів на ринку мобільних застосунків для пошуку та генерації рецептів.
3. Розробити функціональні та нефункціональні вимоги до мобільного застосунку для пошуку та генерації рецептів.
4. Дослідити та обрати інструменти для розробки застосунку для пошуку та генерації рецептів.
5. Розробити архітектуру та інтерфейс застосунку.
6. Провести тестування мобільного застосунку.

3

АНАЛІЗ АНАЛОГІВ

Функціонал	Kitchen Stories	Cookpad	Spilt	Drop recipe
Пошук рецептів за назвою	+	+	+	+
Оцінка рецептів	+	+	+	+
Можливість збереження рецепта	+	-	+	+
Наявність української мови	-	+	-	+
Пошук рецепту за інгредієнтами	-	-	-	+
Доступ до більшості функцій без реєстрації	-	-	-	+
Можливість генерації рецептів на основі наявних інгредієнтів	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні

1. Пошук рецепту за назвою.
2. Генерація рецепту на основі наявних інгредієнтів.
3. Збереження створеного рецепту в особистому кабінеті.
4. Можливість ставити лайки на рецепти, що сподобалися.
5. Можливість зміни фото профілю.
6. Реєстрація та авторизація акаунту.
7. Перегляд рецептів.

Нефункціональні

1. Шифрування паролю
2. Адаптивний дизайн для різних розмірів екранів
3. Кросплатформність

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Flutter Dart

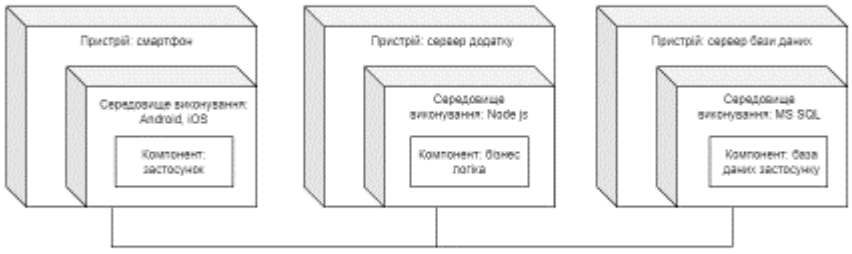
Microsoft SQL Server node android studio

6

Use case діаграма

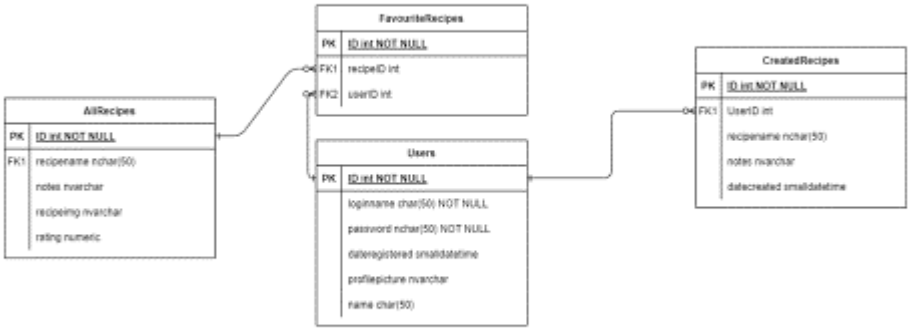
7

Діаграма розгортання



8

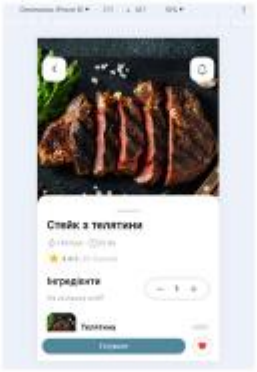
Схема бази даних



9

ЕКРАННІ ФОРМИ

Головний екран

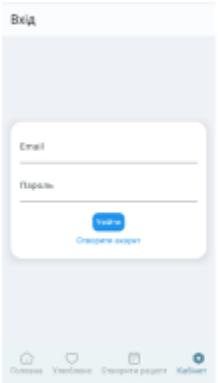


Екран з рецептом

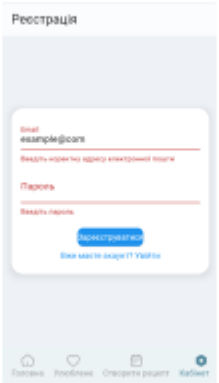
Екран «Улюблене»

10

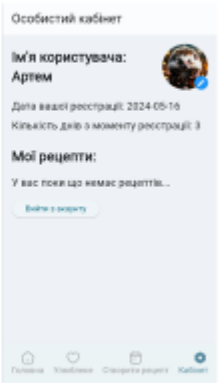
ЕКРАННІ ФОРМИ



Екран входу



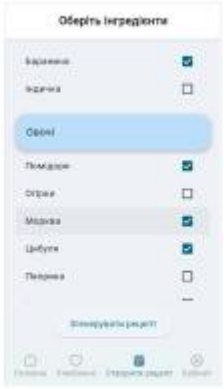
Екран реєстрації



Екран особистого кабінету

11

ЕКРАННІ ФОРМИ



Екран зі списком продуктів за категоріями для генерації рецепту

Екран створеного рецепту

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Дубовик А.Ю., Савіцький В.А. «Мобільний застосунок для пошуку рецептів по наявних інгредієнтах або обмеженому бюджеті для підтримки правильного харчування» // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» 24 квітня 2024 року; Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.331
2. Дубовик А.Ю., Савіцький В.А. «Аналіз та оптимізація швидкодії мобільного застосунку для ефективного пошуку рецептів у режимі реального часу» // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» 24 квітня 2024 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.230.

13

ВИСНОВКИ

1. Проаналізовано предметну галузь пошуку та генерації рецептів.
2. Проведено аналіз конкурентів на ринку мобільних застосунків для пошуку та генерації рецептів.
3. Розроблено функціональні та нефункціональні вимоги до мобільного застосунку для пошуку та генерації рецептів.
4. Досліджено та обрано інструменти для розробки застосунку для пошуку та генерації рецептів.
5. Розроблено архітектуру та інтерфейс застосунку.
6. Проведено тестування мобільного застосунку.

14