Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення мобільного додатку для пекарні «ГЕРЧАК»

Виконав: студент IV курсу, групи СТс-41

спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

_____ Семчишин В.В.
(підпис) (прізвище та ініціали)

Керівник _____ Михайлович Т.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Цуприк Г.Б.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_23_» _січня_ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Семчишину Віталію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення мобільного додатку для пекарні «ГЕРЧАК»

Керівник роботи Михайлович Тарас Володимирович, асистент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-134

2. Термін подання студентом завершеної роботи 14 червня 2023р.

3. Вихідні дані до роботи Відомості щодо пекарні. Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Постановка задачі на розробку мобільного додатку для пекарні «ГЕРЧАК».
2. Проектування та тестування мобільного додатку пекарні «ГЕРЧАК».
3. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Тема та мета кваліфікаційної роботи. 3. Порівняння типів мобільних додатків. 4. Формування вимог до мобільного додатку.
5. Перелік та призначення інформаційних сутностей. 6. Проектування архітектури мобільного додатку. 7. Робочий процес мобільного додатку для пекарні «ГЕРЧАК».
8. Опис інтерфейсу мобільного додатку пекарні «ГЕРЧАК». 9. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Окіпний І.Б., доцент кафедри МТ		

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про мобільні застосунки для пекарень	24.01.2023-30.01.2023	*Виконано*
3.	Переклад та опрацювання джерел про мобільні	31.01.2023-06.02.2023	*Виконано*
	додатки для пекарень		
4.	Виконання дослідження щодо мобільних додатків	07.02.2023-11.02.2023	*Виконано*
	для пекарень		
5	Розробка мобільного додатку для пекарні «ГЕРЧАК»	12.02.2023-26.04.2023	*Виконано*
6.	Оформлення розділу «Постановка задачі на розробку,	27.04.2023-30.04.2023	*Виконано*
	Мобільного додатку для пекарні «ГЕРЧАК»		
7.	Оформлення розділу «Проектування та тестуваннння	01.05.2023-22.05.2023	*Виконано*
	мобільного додатку пекарні «ГЕРЧАК»		
8.	Виконання завдання до підрозділу «Безпека	23.05.2023-02.06.2023	*Виконано*
	життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи хорони	03.06.2023-05.06.2023	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	12.06.2023-14.06.2023	*Виконано*
11.	Нормоконтроль	15.06.2023	*Виконано*
12.	Перевірка на плагіат	16.06.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	22.06.2023	*Виконано*
14.	Захист кваліфікаційної роботи	24.06.2023	

Студент Семчишин В.І.
(підпис) (прізвище та ініціали)

Керівник роботи Михайлович Т.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення мобільного додатку для пекарні «ГЕРЧАК» // Кваліфікаційна робота освітнього рівня «Бакалавр» // Семчишин Віталій Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТс-41 // Тернопіль, 2023 // С. 50, рис. – 14 , табл. – 1, кресл. – 0, додат. – 1, бібліогр. – 36.

Ключові слова: пекарня, мобільний додаток, java, android studio, бази даних, продукція, зв'язок.

Кваліфікаційна робота присвячена створенню мобільного додатку для пекарні «ГЕРЧАК».

Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення рівня поінформованості громадян та працівників пекарні щодо функціональних можливостей автоматизованого збуту, продажу та замовлення продукції.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» проведено аналіз предметної області. Проаналізовано програмне забезпечення, що використовується для розробки мобільних додатків. Описано організацію замовника, виділено основні сутності мобільного додатку та наведено перелік вимог до розробки мобільного додатку.

В другому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» спроектовано базу даних, спроектовано зв'язки мобільного додатку, описано інтерфейс мобільного додатку та проведено тестування та валідацію мобільного додатку пекарні «ГЕРЧАК».

ANNOTATION

A Mobile App Development for the Bakery «GERCHAK» // Qualification work of the educational level "Bachelor" // Semchyshyn Vitaly Ihorovych// Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering,Department of Computer Science, Group STs-41 // Ternopil, 2023 // C. 50, fig. – 14, tables – 1, chair. – 0, annexes - 1, ref. – 36.

Key words: bakery, mobile app, java, android studio, databases, products, relationship.

Qualification work is devoted to the development of a mobile application for the Bakery «GERCHAK».

The purpose of this qualification work of the educational level "Bachelor" is to increase the level of awareness of citizens and bakery workers regarding the functionality of automated marketing, sales and ordering of products.

In the first section of the qualification work of the educational level "Bachelor" the analysis of the subject area is carried out. The software used for the development of mobile applications is analyzed. The organization of the customer is described, the main essence of the mobile application is highlighted, and a list of requirements for the development of the mobile application is given.

In the second section of the qualification work of the educational level "Bachelor" the database was designed, the connections of the mobile application were designed, the interface of the mobile application was described, and the mobile application of the bakery «GERCHAK» was tested and validated.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IOS (англ. Iphone Operating System) – Власницька мобільна операційна система від Apple.

SQL (англ. Structured Query Language) – Структурована мова запитів.

БД – База даних.

ПК – Персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. Головною складовою виробничого процесу в різних видах діяльності є готові вироби, виготовлені основними відділками, цехами підприємств і призначені для продажу стороннім замовникам, а також своїм непромисловим господарствам і власному капітальному будівництву. Облік виготовленої продукції дає змогу вирішити важливе завдання – визначити розмір і склад конкретних споживчих вартостей, вироблених на підприємстві як галузі матеріального виробництва; розміри і структуру сировини для переробної промисловості, продовольчих ресурсів для споживання населення. На його підставі складають баланси продукції, які характеризують наявність і рух товарів.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення рівня поінформованості громадян та працівників пекарні «ГЕРЧАК» щодо автоматизованого збуту, продажу та замовлення продукції.

Для досягнення поставленої мети потребують вирішення ряд наступних завдань:

– Проаналізувати та порівняти типи мобільних додатків.

– Реалізувати мобільний додаток за допомогою середовища розробки мобільних додатків Android Studio [1] та за допомогою мови програмування Java [2].

– Провести валідацію та тестування мобільного додатку.

Практичне значення одержаних результатів. Розроблено мобільний додаток для автоматизації збуту, продажу та замовлення продукції з використанням мови програмування Java та сервісу для створення баз даних Firebase [3].

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПЕКАРНІ «ГЕРЧАК»

1.1 Аналіз програмного забезпечення, що використовується для розробки мобільних додатків

Першим етапом при ставленні завдання є аналіз і вибір засобів реалізації мобільного додатку через зростання числа мобільних пристроїв, що призводить до збільшення потреби в розробці додатків. Кожного дня створюються сотні мобільних додатків, які полегшують роботу з мобільними пристроями. Існує кілька типів додатків, які використовують розробники.

Нативні додатки отримали назву «нативними» через те, що вони розроблені на рідній мові програмування для певної платформи. Для Android ця мова є Java, тоді як для iOS – Objective-C або Swift. Нативні додатки розташовуються безпосередньо на пристрої і можуть бути доступні через іконку. Вони встановлюються через відповідні магазини додатків, такі як Play Market для Android та App Store для iOS.

Нативні додатки розроблені спеціально для конкретної платформи і можуть використовувати всі функції пристрою, такі як камера, GPS-датчик, акселерометр, компас, список контактів та інші. Вони також можуть розпізнавати стандартні жести, встановлені операційною системою, або навіть нові жести, які використовуються в конкретному додатку. Оскільки нативні додатки оптимізовані під конкретну операційну систему, вони працюють швидко і ефективно на будь-якому смартфоні. Нативні додатки можуть мати доступ до системи сповіщень пристрою, а в залежності від їх призначення можуть функціонувати повністю або частково без інтернет-з'єднання.

Деякі переваги нативних додатків включають:

– Швидкість та продуктивність роботи;
– Високий рівень безпеки;
– Розширений інтерфейс;
– Максимальна функціональність;
– Здатність працювати без підключення до Інтернету;
– Зручність для кінцевих користувачів.

Натомість, у нативних додатків є деякі недоліки:

– Обмеженість платформової підтримки;
– Тривалість процесу розробки;
– Висока вартість розробки;
– Необхідність випускати оновлення для косметичних змін.

Мобільні веб-додатки насправді не є стандартними додатками. Фактично, вони представляють собою веб-сайти, які адаптовані та оптимізовані для використання на мобільних пристроях. Для користування такими додатками необхідно мати браузер на пристрої, знати його URL-адресу та мати доступ до Інтернету (що дозволяє оновлювати інформацію в таких додатках).

При запуску мобільного веб-додатка користувач здійснює дії, схожі на перехід до будь-якого веб-сайту, і отримує можливість "встановити" його на свій робочий стіл, створивши закладку до веб-сайту. Одним з переваг мобільних веб-додатків є їх кросплатформеність, тобто здатність працювати на різних платформах пристроїв. Також вони не використовують програмне забезпечення пристрою, що дозволяє заощадити місце в його пам'яті.

Мобільні веб-додатки набули популярності з появою HTML5, оскільки вони надають доступ до розширеного функціоналу нативних додатків через звичайний веб-браузер. На сьогоднішній день межа між веб-додатками і звичайними веб-сторінками стає все більш нечіткою, оскільки HTML5 надає все більше можливостей, які використовуються на сайтах. Розробка веб-додатків стала швидшою завдяки використанню традиційних інструментів та фреймворків. Проте одним з обмежень мобільних веб-додатків є їхній залежний від Інтернету характер роботи.

Основний недолік мобільних веб-додатків полягає в їхній продуктивності, яка є середньою порівняно з іншими типами додатків, та залежить від доступності інтернет-з'єднання. Проте вони також мають деякі переваги, такі як повна підтримка платформ, простий та швидкий процес розробки, наявність великої кількості компетентних розробників та можливість уникнути необхідності завантаження з магазину додатків.

Нативні додатки, натомість, мають свої недоліки, такі як обов'язкове підключення до Інтернету, обмежена функціональність програмного інтерфейсу, неможливість відправки push-повідомлень, обмежена продуктивність та швидкість

роботи, а також недостатній рівень безпеки.

Гібридні додатки є поєднанням веб-додатків та нативних додатків, що забезпечує їм кросплатформеність та доступ до функціональності смартфонів. Вони можуть бути завантажені з маркетів, таких як Google Play і App Store, та мають опцію автономного оновлення інформації, але для їх роботи потрібне інтернет-підключення. Гібридні додатки поєднують переваги нативних додатків з актуальністю веб-технологій. Вони також мають нижчу вартість розробки та вищу швидкість порівняно з нативними додатками. Крім того, розробка гібридних додатків дозволяє швидко вносити зміни, не вимагаючи повторного розміщення в магазині додатків.

У підсумку, розробка гібридних додатків є перспективною, оскільки вони забезпечують підтримку двох платформ і уникнення окремої розробки для кожної операційної системи, що є важливим фактором при виборі.

Крім інших факторів, необхідно враховувати, що якість і можливості гібридних додатків в першу чергу залежать від використаного розробником фреймворку. Крім того, важливо врахувати переваги, які роблять гібридні додатки привабливими порівняно з іншими варіантами. Тому розробку гібридних додатків рекомендовано у таких випадках:

– Якщо важливо зекономити в бюджеті розробки;

– Якщо потрібно створити простий додаток з простою анімацією;

– Якщо є потреба в оперативній розробці програми для принаймні двох платформ.

Серед переваг мобільних веб-додатків можна виділити нижчеперелічені:

– Швидкість і ефективність розробки;

– Необхідність меншої кількості розробників;

– Кросплатформеність;

– Можливість автономного оновлення.

Гібридні додатки, зі свого боку, мають такі недоліки:

– Некоректна робота без інтернет-з'єднання;

– Середня швидкість роботи порівняно з нативними додатками;

– Обмежений набір візуальних елементів.

У сучасному світі мобільні пристрої практично неможливо уявити без

встановлених додатків. Мобільний додаток – це розроблена програма для планшетів і смартфонів, яка встановлюється на певну платформу і має свою функціональність. Простими словами, він виконує певні дії та вирішує певні завдання. Вибір відповідної моделі мобільного додатка є важливим етапом його розробки, і на нього впливають декілька факторів, таких як технічні можливості розробників, доступ до пристроїв, швидкість Інтернету, а також одно або багатоплатформеність додатка [4].

Оскільки, мобільний додаток для пекарні «ГЕРЧАК» повинен бути ефективним, тобто швидким та використовувати всі допустимі можливості операційної системи та апаратного забезпечення, і поки немає потреби у розробці під багато платформ, необхідно розробити додаток лише під платформу Android, найдоцільніше використовувати нативний тип розробки. Мобільний додаток для пекарні «ГЕРЧАК» буде розроблено на мові програмування Java та у середовищі розробки під ОС Android Android Studio.

1.2 Постановка завдання

Замовником є пекарня «ГЕРЧАК» [5] (див. рис. 1.1), яка займається виробництвом продукції для споживання та користується великою популярністю по всій Україні.

Рисунок 1.1 – Логотип пекарні «ГЕРЧАК»

Пекарня «ГЕРЧАК» пропонує великий вибір продукції, серед таких видів як:

– Гріссіні;

– Крутони;

– Грінки.

У зв’язку з великою кількістю клієнтів перед пекарнею постала необхідність у створенні мобільного додатку, який автоматизує роботу пекарні.

На етапі проектування можна виділити наступні сутності:

– Користувачі (клієнти, які використовують мобільний додаток; характеризується логіном та паролем);

– Адміністратор (користувач, який має повний доступ до інформації; характеризується логіном та паролем);

– Категорії продукції (зберігають інформацію про типи продукції; характеризується назвою та зображенням категорії продукції);

– Продукції (зберігають інформацію про продукцію; характеризується назвою продукції, описом продукції, ціною продукції, зображенням продукції, посиланням на сутність «Категорії продукції» та полем чи є продукція ексклюзивною);

– Замовлення продукції (зберігають інформацію про замовлення продукції; характеризується посиланням на сутності «Користувачі», «Продукції», зображенням замовленої продукції, кількістю та ціною замовлення продукції);

– Адреса доставки продукції до клієнта (зберігає інформацію про адресу доставки продукції до клієнта; характеризується посиланням на сутність «Користувачі», адресним рядком, областю, локацією, довжиною, широтою);

– Рейтинг продукції за користувачами (зберігає інформацію про рейтинг продукції за користувачами; характеризується посиланням на сутність «Продукції» та рейтингом продукції).

Кожна складених сутностей описує модель даних, що зберігається у таблицях БД.

Проаналізувавши предметну область та обговоривши функціонал з замовником, можна сформувати загальні вимоги до розробки додатку:

1. Вимоги до інтерфейсу та платформи:

– Інтерфейс повинен бути зрозумілим та ненавантаженим візуальною частиною;

– Логотип пекарні повинен відображатись у вигляді мобільного додатку;

– Інтерфейс повинен використовувати сучасні компоненти;

– Розробка мобільного додатку виконується під платформу Android, мінімальна версія SDK – 21.

2. Вимоги до доступу та захисту:

– Вхід користувача повинен бути захищеним парольною автентифікацією;

– Повинен бути передбачений обліковий запис адміністратора;

– Паролі відсилаються лише раз при вході в додаток;

– Логін введений користувачем повинен зберігатись і для наступного входу;

– Пароль при авторизації та реєстрації не повинен відображатись текстом.

3. Вимоги до функціональних можливостей:

– Можливість реєстрації користувачів;

– Можливість авторизації користувачів;

– Можливість додавання інформації про «Категорії продукції»;

– Можливість виведення інформації про «Категорії продукції»;

– Можливість редагування інформації про «Категорії продукції»;

– Можливість видалення інформації про «Категорії продукції»;

– Можливість додавання інформації про «Продукції»;

– Можливість виведення інформації про «Продукції»;

– Можливість редагування інформації про «Продукції»;

– Можливість видалення інформації про «Продукції»;

– Можливість додавання інформації про «Замовлення продукції»;

– Можливість виведення інформації про «Замовлення продукції»;

– Можливість редагування інформації про «Замовлення продукції»;

– Можливість видалення інформації про «Замовлення продукції»;

– Можливість додавання інформації про «Адреса доставки продукції до клієнта»;

– Можливість виведення інформації про «Адреса доставки продукції до клієнта»;

– Можливість додавання інформації про «Рейтинг продукції за користувачами»;

– Можливість виведення інформації про «Рейтинг продукції за користувачами»;

– Можливість доступу до місцезнаходження користувача;

– Можливість створення профілю користувача.

4. Вимоги до звітності:

– Можливість виведення ексклюзивної продукції;

– Можливість перегляду фінансів (кількості грошей за замовлену продукцію).

Розроблений мобільний додаток повинен відповідати усім вищеперерахованим вимогам.

1.3 Висновки до першого розділу

В першому розділі кваліфікаційної роботи було проведено аналіз предметної області, тобто були досліджені доступні програмні інструменти для розробки мобільного додатку – мова програмування та інтегроване середовище розробки.

Була сформована постановка задачі – була розглянута організація замовника, були поставлені вимоги до розробки мобільного додатку та виділені основні сутності, що присутні в мобільному додатку.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ПЕКАРНІ «ГЕРЧАК»

2.1 Проектування бази даних

Враховуючи складність структури проектованого мобільного додатку виділимо наступні типи інформаційних елементів мобільного додатку пекарні «ГЕРЧАК»:

– файли структурних елементів мобільного додатку;

– інформаційні таблиці БД.

Сформулюємо переліки та структуру інформаційних файлів та файлів структурних елементів мобільного додатку. Провівши аналіз архітектури мобільного додатку та інформації збереженої в ній сформуємо список основних інформаційних елементів:

– файл підключення до бази даних;

– файли структурних модулів мобільного додатку;

– файли системних компонент;

– файли структурних блоків компонент.

Сформувавши список файлів проектованого мобільного додатку перейдемо до проектування структури таблиць БД. Проаналізувавши предметну область та вимоги до мобільного додатку сформуємо перелік інформаційних сутностей (див. таблицю 2.1).

Таблиця 2.1 – Перелік та призначення інформаційних сутностей

Назва сутності	Призначення
Users	Список зареєстрованих користувачів
Category	Список категорій продукції
Foods	Список продукції
FoodsOnCart	Список замовлень продукції
UserDeliveryAddress	Адреса замовника
UsersRating	Рейтинг продукції за користувачами

Визначившись з переліком інформаційних сутностей можемо перейти до проектування концептуальної моделі БД.

Концептуальна модель найбільш повно відповідає потребам проектування баз

знань та має дві області понять, що побудовані за принципом ієрархічного дерева.

Перша область – це дерево типів даних, друга – дерево даних. Дерево типів описує структуру дерева даних, тому без дерева типів немає ніякої логічної цілісності дерева даних. Рівень спрощення – рівень деталізації подання про об’єкт реального світу, достатній нам для його опису та подальшого використання, що володіє певними властивостями.

Властивість об’єкта – це одна з характеристик об’єкта реального світу, інформацію, про яку зберігають в базі даних. Тип – набір властивостей та подій об’єкта, описаних як єдиний комплекс. При цьому, залежно від рівня спрощень, властивістю типу може бути інший тип.

Об’єкт – сукупність типів та властивостей, об’єднаних в один тип, здатний описати об’єкт реального світу. Зв’язок – це властивість типу або властивості типу, яка характеризує взаємозв’язок типів в дереві даних або спосіб зміни значення властивості об’єктного типу відповідно.

Бувають три типи зв’язків: включення в дереві даних, вставка з іншого типу значення властивості типу та посилання на екземпляр типу в дереві даних.

Включення дозволяє будувати дерево даних. Перелік та призначення інформаційних сутностей наведено в табл 1.1.

Концептуальна схема відповідної бази даних наведена на рисунку 2.1. Концептуальна модель містить 6 таблиць бази даних мобільного додатку.

Між двома таблицями встановлено зв’язок один до одного. Такий зв’язок позначений напрямленою лінією. Початок лінії відповідає відношенню «один».

Таблиця Users використовується для встановлення зв’язку «один до багатьох» з таблицями FoodsOnCart, UserDeliveryAddress.

Таблиця Category використовується для встановлення зв’язку «один до одного» з таблицею Foods.

Таблиця Foods використовується для встановлення зв’язку «один до багатьох» з таблицями FoodsOnCart, UsersRating.

Таблиця FoodsOnCart використовується для встановлення зв’язку «один до багатьох» з таблицями Users, Foods.

Таблиця UsersDeliveryAddress використовується для встановлення зв’язку «один до одного» з таблицею Users. Таблиця UsersRating використовується для

встановлення зв’язку «один до багатьох» з таблицею Foods.

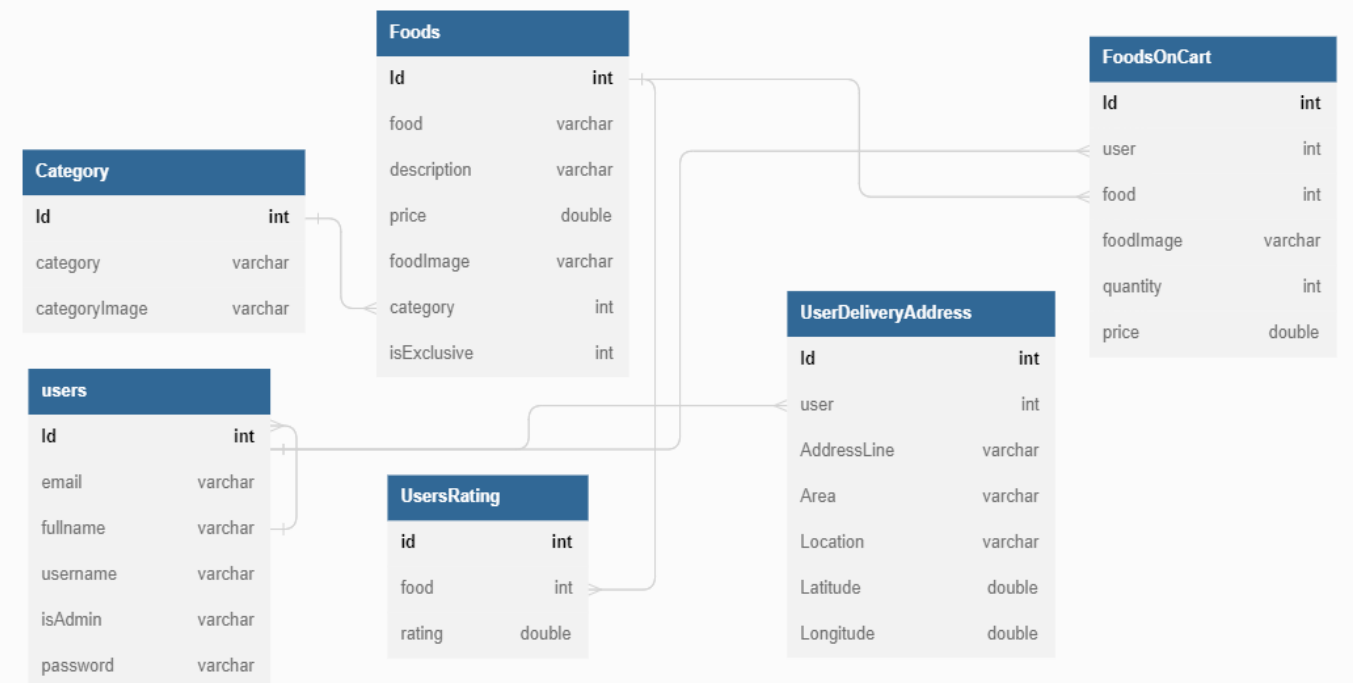

Рисунок 2.1 – Концептуальна модель бази даних пекарні «ГЕРЧАК»

Оскільки в якості моделі даних вибрано реляційну, то розглянемо деякі особливості спроєктованої концептуальної схеми бази даних «Пекарня». Проаналізуємо зв’язки між сутностями бази даних. Двосторонніх зв’язків, тобто типу «багато-до-багатьох» не має у розробленій схемі. Зв’язків між трьома та більше сутностями також не виявлено. У концептуальній моделі даних присутній багатозначний атрибут «Логін» сутності «Користувач».

Тому потрібно проаналізувати ще раз предметну область та транзакції, які будуть виконуватись в базі даних. Пошук за логіном користувача виконуватись не буде, тому не потрібна декомпозиція цього атрибуту для визначення деякої сутності. Отже, розроблена концептуальна схема проектується у логічну схему, а кожна сутність – у відношення. Зв’язки між відношеннями моделюються за допомогою механізму первинних і зовнішніх ключів. Після концептуального проектування перейдемо до логічного проектування БД.

На цьому етапі визначаємо ті вимоги підтримки цілісності даних, які необхідно реалізувати в локальній логічній моделі даних користувача. Їхнє призначення складається в підтримці постійної внутрішньої погодженості інформації, організованої у вигляді бази даних. На даному етапі завдання полягає в тому, щоб

установити, які саме вимоги підтримки цілісності даних необхідні. Ми розглянемо чотири типи вимог підтримки цілісності:

– обов'язкові дані;
– цілісність сутностей;
– цілісність посилань;
– вимоги підприємства.

Необхідно установити, які з атрибутів завжди повинні містити одне з припустимих значень. Якщо говорити більш конкретніше то це атрибути, що завжди повинні мати значення, відмінні від NULL. Атрибути Код_категорії, Назва, Зображення відношення Категорії продукції завжди повинні містити значення, відмінні від порожнього.

Але на атрибут Назва цього ж відношення дана вимога не поширюється, і цей атрибут цілком можуть мати значення NULL, що означає що не важливо яка назва категорії продукції або її назва категорії невідома.

Аналогічно проаналізуємо інші відношення бази даних «Пекарня». Атрибути Код_продукції, Назва, Опис, Ціна, Зображення, Категорія продукції відношення Продукція завжди повинно містити значення, відмінні від порожнього.

Але на атрибут Ціна цього ж відношення дана вимога не поширюється, і цей атрибут цілком можуть мати значення NULL, що означає що в не важливо яка ціна за продукцію не відома.

Атрибут первинного ключа сутності «Продукція» не може мати значення NULL. Тому кожен екземпляр відношення Продукція обов'язково повинен мати конкретне значення атрибута його первинного ключа Код_продукції. Атрибути, що входять у значення первинного ключа кожної сутності, були визначені на попередніх етапах.

Перевіримо кожне відношення і, за необхідності, задаємо значення NOT NULL для первинних ключів кожного відношення бази даних «Пекарня». Зв’язки між сутностями моделюються за допомогою переміщення в дочірнє відношення копії первинного ключа батьківського відношення.

Поняття цілісності посилань означає, що якщо зовнішній ключ дочірнього відношення містить деяке значення, то це значення повинне посилатися на існуюче і коректне значення ключа в батьківському відношенні.

Підтримка цілісності посилань організовується за допомогою необхідних обмежень для значень первинних і зовнішніх ключів.

Варто вказати умови для кожного зовнішнього ключа, що повинні виконуватися при відновленні або видаленні відповідного значення первинного ключа.

У цьому випадку застосуємо одну з запропонованих стратегій - NO ACTION, CASCADE, SET NULL, SET DEFAULT або NO CHECK. Розглянемо вимоги продажів продукції. Ці вимоги, ще називають бізнес-правилами, які визначаються тими методами й обмеженнями, що прийняті на підприємстві.

Одним з них є те, що клієнт пекарні не може редагувати дані в базі даних. Перейдемо до проектування таблиць БД.

Створимо спочатку головні таблиці бази даних «Пекарня», вказавши первинний ключ для кожної таблиці.

Тоді потрібно виконати запит для створення підлеглих таблиць, задавши як первинний ключ так і зовнішній. База даних міститиме шість таблиць:

– Користувачі;
– Категорії продукції;
– Продукція;
– Замовлення продукції;
– Адреса доставки продукції;
– Рейтинг продукції за користувачами.

Отже, виконаємо SQL-запит для створення таблиці Користувачі:

Лістинг 2.1 – Текст SQL-запиту для створення таблиці Users бази даних «Pekarnya»

```
CREATE TABLE [users] (
     [Id] int PRIMARY KEY IDENTITY(1, 1),
     [email] nvarchar(255),
     [fullname] nvarchar(255),
     [username] nvarchar(255),
     [isAdmin] nvarchar(255),
     [password] nvarchar(255)) GO ALTER TABLE [UserDeliveryAddress]
ADD FOREIGN KEY ([user]) REFERENCES [users] ([Id]) GO ALTER TABLE
[FoodsOnCart] ADD FOREIGN KEY ([user]) REFERENCES [users] ([Id]) GO
ALTER TABLE [users] ADD FOREIGN KEY ([Id]) REFERENCES [users]
([fullname])GO
```

Далі створимо таблицю Категорія продукції:

Лістинг 2.2 – Текст SQL-запиту для створення таблиці Category бази даних «Pekarnya»

```
CREATE TABLE [Category] (
     [Id] int PRIMARY KEY IDENTITY(1, 1),
     [category] nvarchar(255),
     [categoryImage] nvarchar(255))GO
ALTER TABLE [Foods] ADD FOREIGN KEY ([category]) REFERENCES [Category]
([Id]) GO
```

Далі створимо таблицю Продукція:

Лістинг 2.3 – Текст SQL-запиту для створення таблиці Foods бази даних «Pekarnya»

```
CREATE TABLE [Foods] (
     [Id] int PRIMARY KEY IDENTITY(1, 1),
     [food] nvarchar(255),
     [description] nvarchar(255),
     [price] double,
     [foodImage] nvarchar(255),
     [category] int,
     [isExclusive] int) GO ALTER TABLE [Foods] ADD FOREIGN KEY
([category]) REFERENCES [Category] ([Id]) GO ALTER TABLE [FoodsOnCart]
ADD FOREIGN KEY ([food]) REFERENCES [Foods] ([Id]) GO
ALTER TABLE [UsersRating] ADD FOREIGN KEY ([food]) REFERENCES [Foods]
([Id]) GO
```

Далі створимо таблицю Замовлення продукції:

Лістинг 2.4 – Текст SQL-запиту для створення таблиці FoodsOnCart бази даних «Pekarnya»

```
CREATE TABLE [FoodsOnCart] (
     [Id] int PRIMARY KEY IDENTITY(1, 1),
     [user] int,
     [food] int,
     [foodImage] nvarchar(255),
     [quantity] int,
     [price] double ) GO
ALTER TABLE [FoodsOnCart] ADD FOREIGN KEY ([food]) REFERENCES [Foods]
([Id])
GO
ALTER TABLE [FoodsOnCart] ADD FOREIGN KEY ([user]) REFERENCES [users]
([Id])GO
```

Далі створимо таблицю Адреса доставки продукції:

Лістинг 2.5 – Текст SQL-запиту для створення таблиці UserDeliveryAddress бази даних «Pekarnya»

```
CREATE TABLE [UserDeliveryAddress] (
     [Id] int PRIMARY KEY IDENTITY(1, 1),
     [user] int,
     [AddressLine] nvarchar(255),
     [Area] nvarchar(255),
     [Location] nvarchar(255),
     [Latitude] double,
     [Longitude] double) GO
ALTER TABLE [UserDeliveryAddress] ADD FOREIGN KEY ([user]) REFERENCES
[users] ([Id])GO
```

Далі створимо таблицю Рейтинг продукції за користувачами:

Лістинг 2.6 – Текст SQL-запиту для створення таблиці UserRating бази даних «Pekarnya»

```
CREATE TABLE [UsersRating] (
     [id] int PRIMARY KEY IDENTITY(1, 1),
     [food] int,
     [rating] double )GO
ALTER TABLE [UsersRating] ADD FOREIGN KEY ([food]) REFERENCES [Foods]
([Id])GO
```

В лістингу 2.1 наведено запит на створення таблиці «Користувачі» яка призначена для авторизації та реєстрації користувачів.

В лістингу 2.2 наведено запит на створення таблиці «Категорія продукції» яка призначена для додавання, редагування, видалення та виведення інформації про категорії продукції.

В лістингу 2.3 наведено запит на створення таблиці «Продукція» яка призначена для додавання, редагування, видалення та виведення інформації про продукцію.

В лістингу 2.4 наведено запит на створення таблиці «Замовлення» яка призначена для додавання, редагування та виведення інформації про категорії замовлення продукції.

В лістингу 2.5 наведено запит на створення таблиці Адреса доставки продукції яка містить адресу користувачів.

В лістингу 2.6 наведено запит на створення таблиці Рейтинг продукції за користувачами яка містить рейтинг продукції.

2.2 Проектування архітектури мобільного додатку пекарні «ГЕРЧАК»

Для того, щоб реалізувати всі потрібні функції додатку, він повинен містити декілька екранів для того, щоб не перевантажувати інтерфейс елементами. Для цього було розроблено наступні екрани, а частини інтерфейсу мобільного додатку зображено на рисунку 2.2:

– екран реєстрації (RegisterActivity), за допомогою якого користувач проходить реєстрацію;

– екран авторизації (LoginActivity), за допомогою якого користувач проходить авторизацію;

– екран домашньої сторінки користувача (HomeUserActivity), за допомогою якого користувач переглядає список продукції;

– екран додавання продукції до корзини (OrdersActivity);

– екран дозволу використання місцезнаходження (PermissionMapActivity);

– екран оплати за замовлення (UserCheckoutPayProcessActivity);

– екран підтвердження оплати за замовлення (OrderDoneActivity);

– екран профілю користувача (ProfileActivity);

– екран перегляду категорій продукції (CategoryActivity);

– екран домашньої сторінки Адміністратора (HomeAdminActivity);

– екран панелі Адміністратора (AdminActivity);

– екран додавання Адміністратора (AddSubAdmin);

– екран додавання категорій продукції (AddCategoryActivity);

– екран оновлення категорій продукції (UpdateCategoryActivity);

– екран перегляду категорій продукції в адміністративній панелі (AdminCategoryActivity);

– екран перегляду продукції в адміністративній панелі (AdminFoodActivity);

– екран додавання продукції (AddFoodActivity);

– екран оновлення продукції (UpdateFoodActivity).

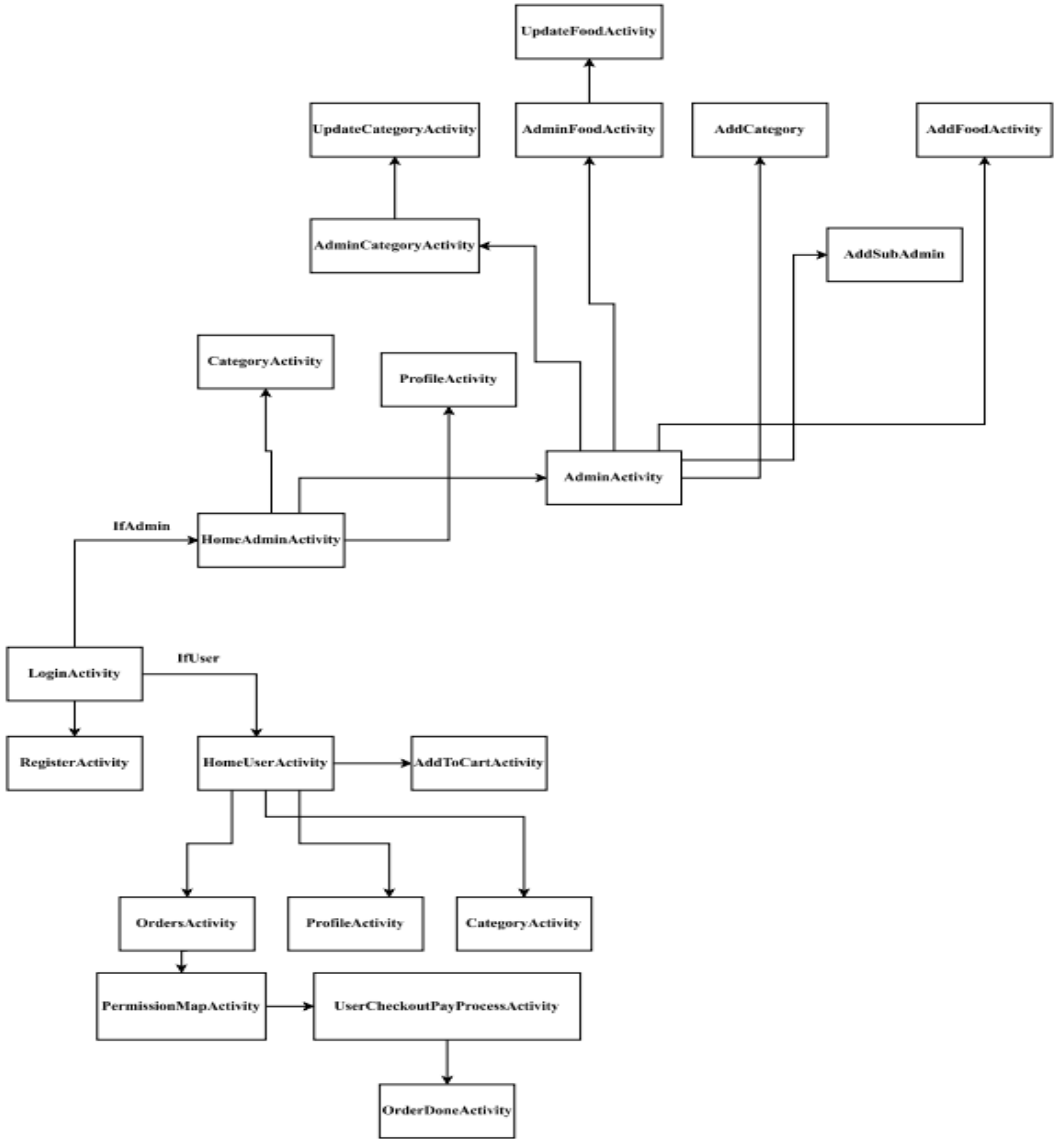

Рисунок 2.2 – Граф можливих зв’язків між екранами

На рисунку 2.2 схематично зображені екрани додатку та шляхи можливих переходів. При запуску додатку відкривається екран авторизації. Це зроблено для того, щоб не авторизовані користувачі не могли ніяким чином використати застосунок.

Після авторизації користувач потрапляє на екран перегляду продукції. З цього екрану перемістити на екран, який зв’язаний з додаванням продукції до корзини. Вгорі екрану з інформацією про продукцію розміщене пошукове поле для пошуку продукції за назвою.

2.3 Опис інтерфейсу мобільного додатку пекарні «ГЕРЧАК»

При запуску мобільного додатку відкривається екран домашньої сторінки, який

зображений на рисунку 2.3:

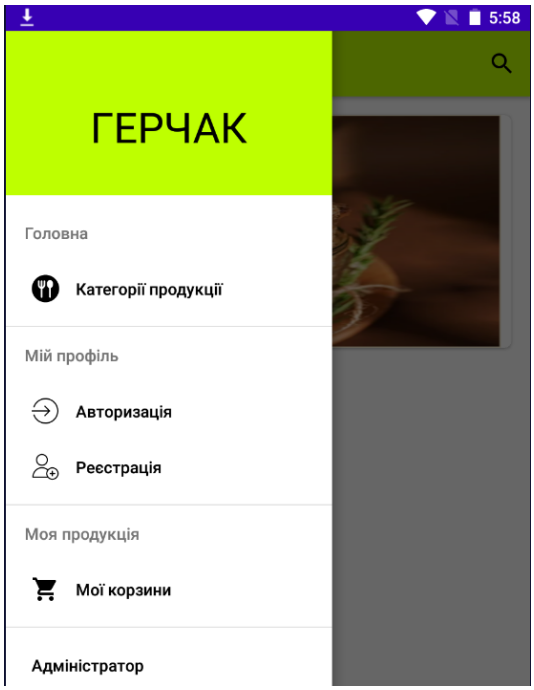

Рисунок 2.3 – Екран домашньої сторінки

При натисканні пункту меню «Авторизація» відбувається перехід на екран авторизації користувача. Екран авторизації наведений на рисунку 2.4:

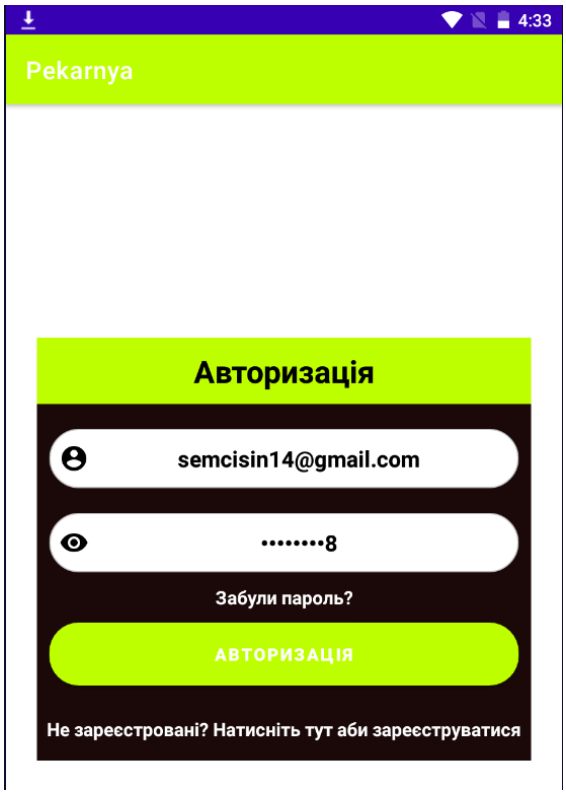

Рисунок 2.4 – Екран авторизації користувача

Після авторизації користувача відбудеться перехід на екран перегляду категорій продукцій та продукції. Екран перегляду категорій продукцій та продукцій наведений на рисунку 2.5:

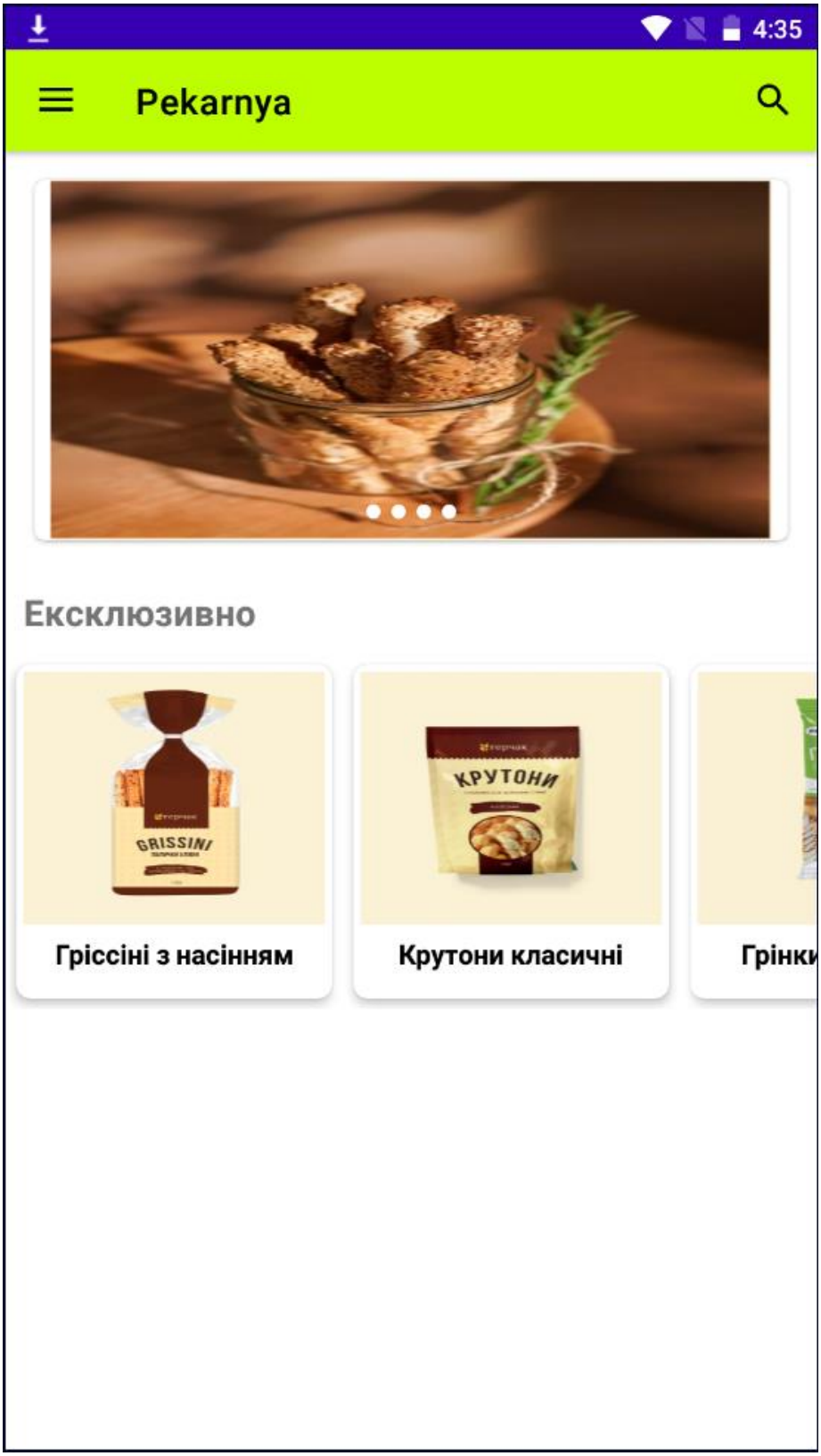

Рисунок 2.5 – Екран перегляду категорій продукцій та продукцій

Як бачимо на рисунку 2.5 видно кнопку пошуку, за допомогою якої ми можемо

у поле пошуку ввести назву продукції. Екран результату пошуку продукції наведено на рисунку 2.6:

Рисунок 2.6 – Екран результату пошуку продукції

Після пошуку, користувач може додати продукцію до корзини. Спочатку необхідно вибрати кількість продукції. Після додавання продукції до корзини

можемо побачити, яку продукцію ми додали до корзини. Екран виведення продукцій, доданих до корзини наведено на рисунку 2.7:

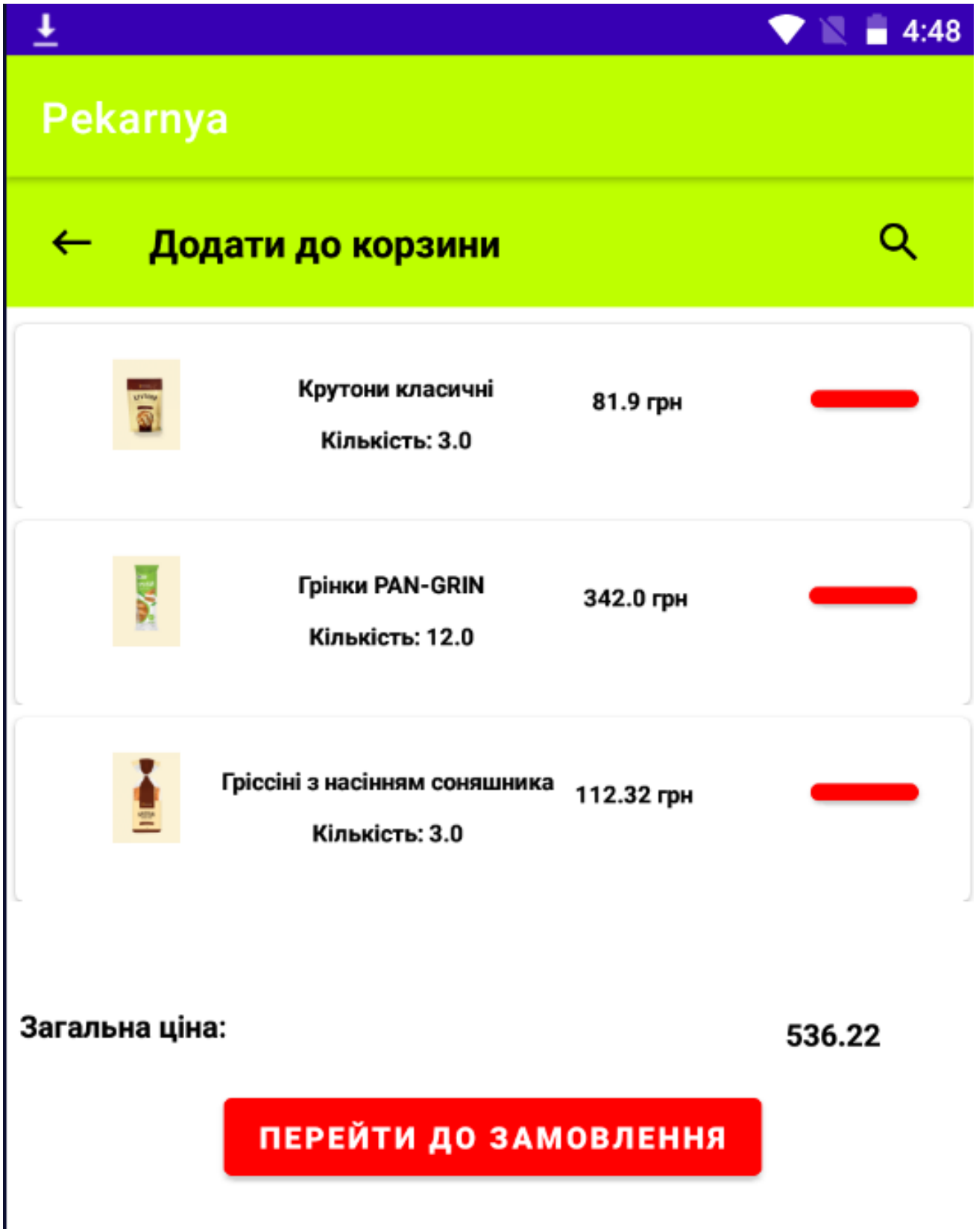

Рисунок 2.7 – Екран виведення продукцій, доданих до корзини

Після натискання кнопки перейти до замовлення можемо на мапі встановити

адресу куди варто доставити продукцію. Екран встановлення адреси на мапі наведено на рисунку 2.8, а екран виведення інформації про те що продукцію замовлено наведено на рисунку 2.9.

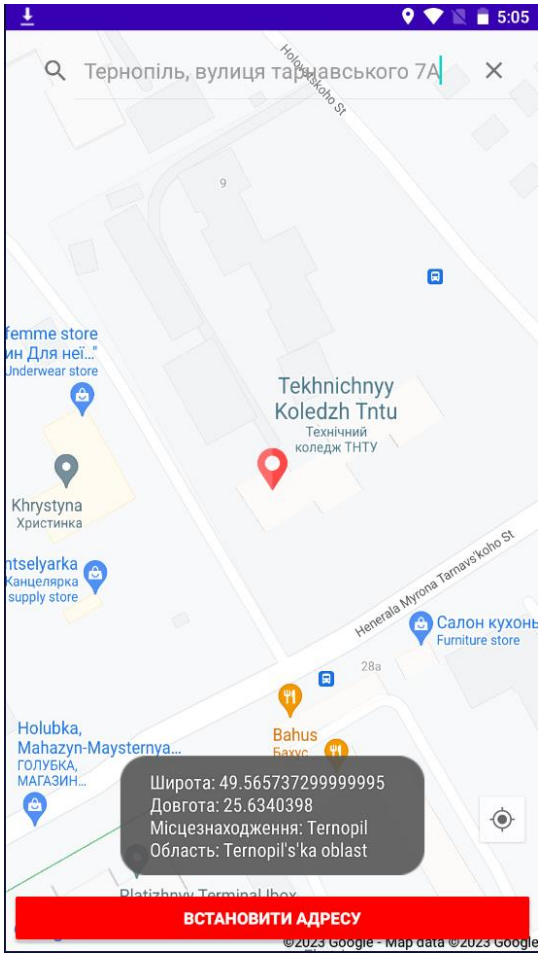

Рисунок 2.8 – Екран встановлення адреси

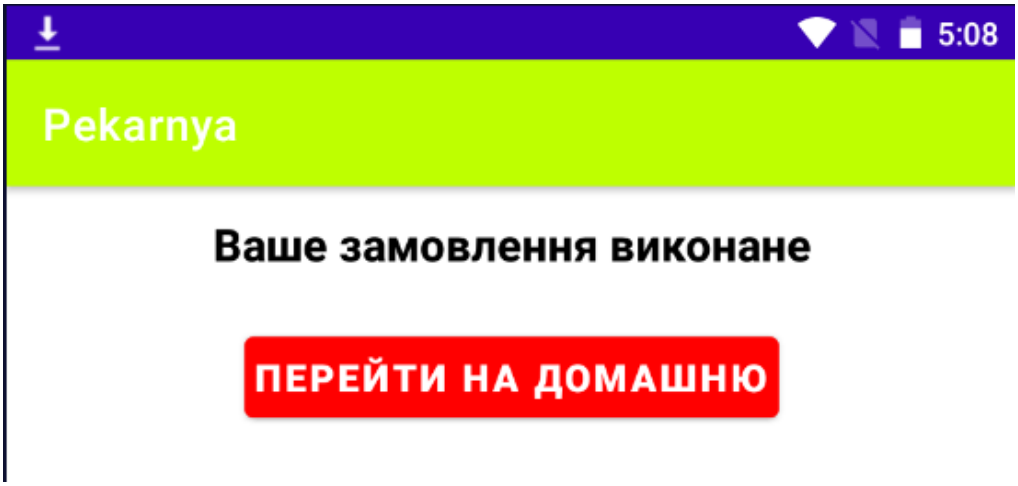

Рисунок 2.9 – Екран виведення повідомлення про те що замовлення продукції виконане

Під час встановлення адреси встановлюється наступна інформація: широта, довгота місцезнаходження та область.

2.4 Тестування та валідація мобільного додатку пекарні «ГЕРЧАК»

Для валідації коду мобільного додатку пекарні «ГЕРЧАК» було використане інтегроване середовище розробки Android Studio, яке рекомендоване для розробки мобільних додатків, і було обране для розробки мобільного додатку. Запуск валідації був проведений з використанням утиліти Інспекції коду [6], показаний на рисунку 2.10.

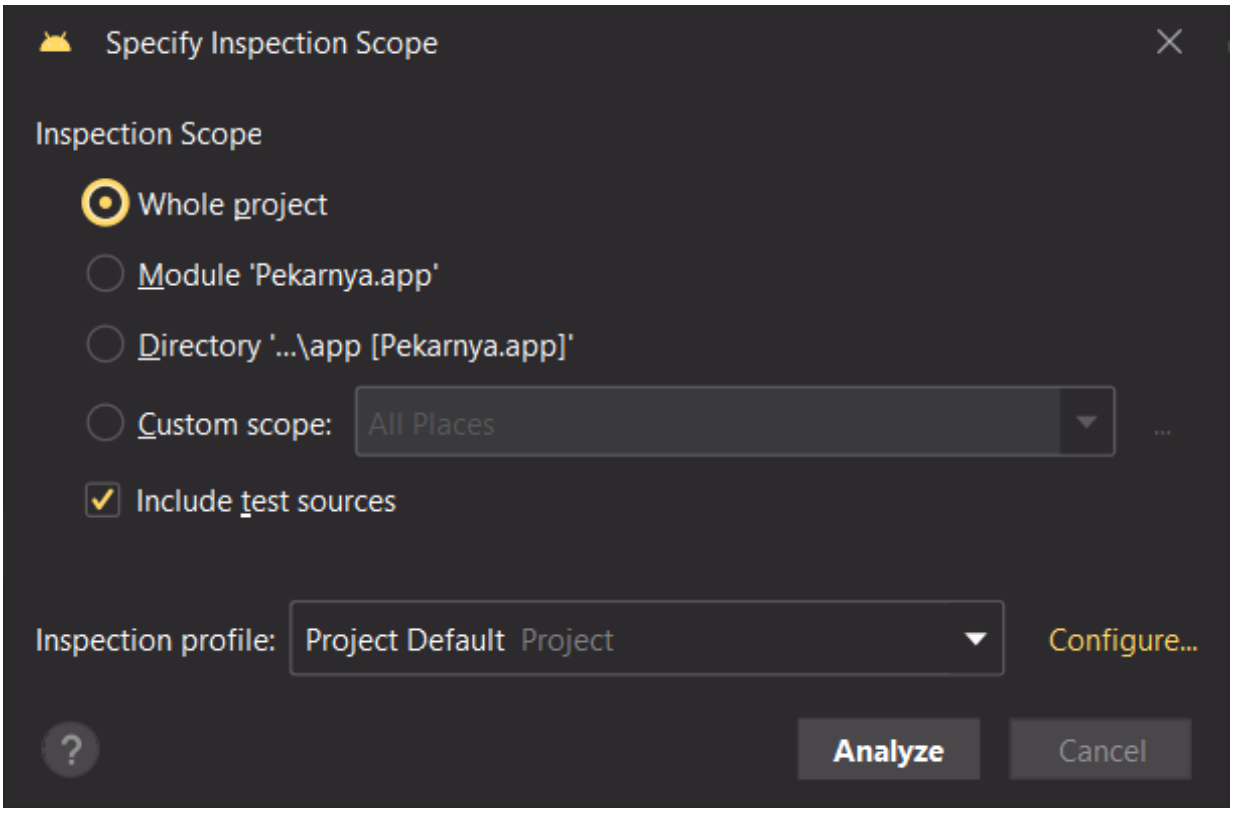

Рисунок 2.10 – Утиліта інспекції коду

Всього було виявлено помилки, які належать чотирьом групам – Android, Java, Kotlin та Proofreading. Остання група містить інформацію про типографічні помилки, які зображені на рисунку 2.11, майже всі випадки пов’язані з словом «обов’язковим», якого не існує в англійській мові, що викликає відповідні попередження про незрозумілі назви змінних. Відповідно цю групу можна пропустити.

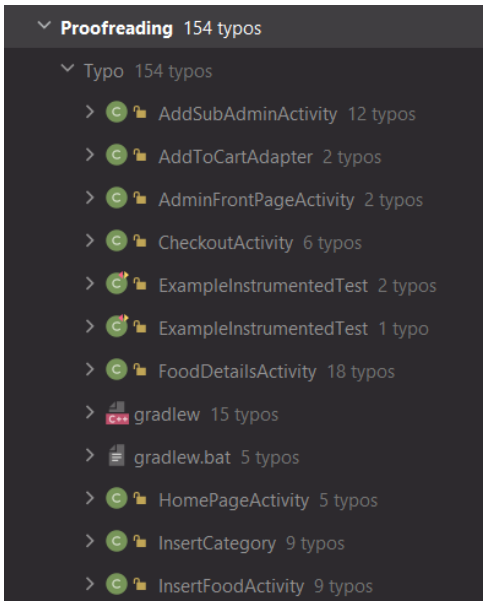

Рисунок 2.11 – Категорія попереджень Proofreading

Перша група, показана на рисунку 2.12, містить попередження для типових Android файлів – конфігурацій та ресурсів.

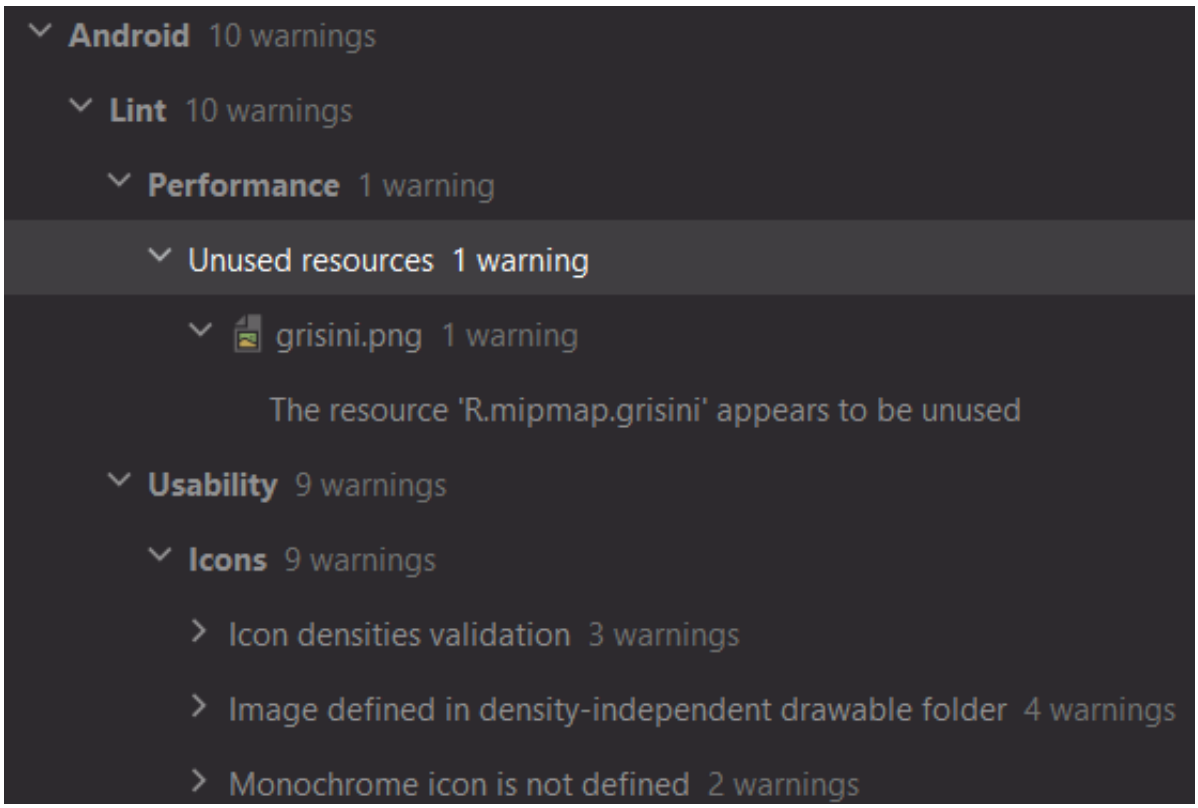

Рисунок 2.12 – Категорія попереджень Android

Був проведений детальний аналіз попереджень:

– «Невикористані ресурси» – Ресурс “R.mipmap.grisini”, здається, не використовується;

– «Перевірка щільності піктограм» – У папках “drawable-hdpi”, “drawable-xhdpi”, “drawable-xxhdpi” відсутні малюнки ic_back_animated.xml, ic_back_vector.xml, ic_menu_animated.xml, ic_menu_vector.xml;

– «Відхилено зображення в папці, які не залежать від щільності» – Знайдено зображення cookie.png, grin.png, gris.png, krut.png у папці без щільності;

– «Монохромний значок не визначено» – На піктограмі мобільного додатку відсутній монохромний тег.

Наступні дві групи стосуються зауважень щодо синтаксису двох мов програмування – Java та Kotlin [7], показані на рисунку 2.13. Попередження для Java пропущені, оскільки синтаксис Java є необхідним для роботи.

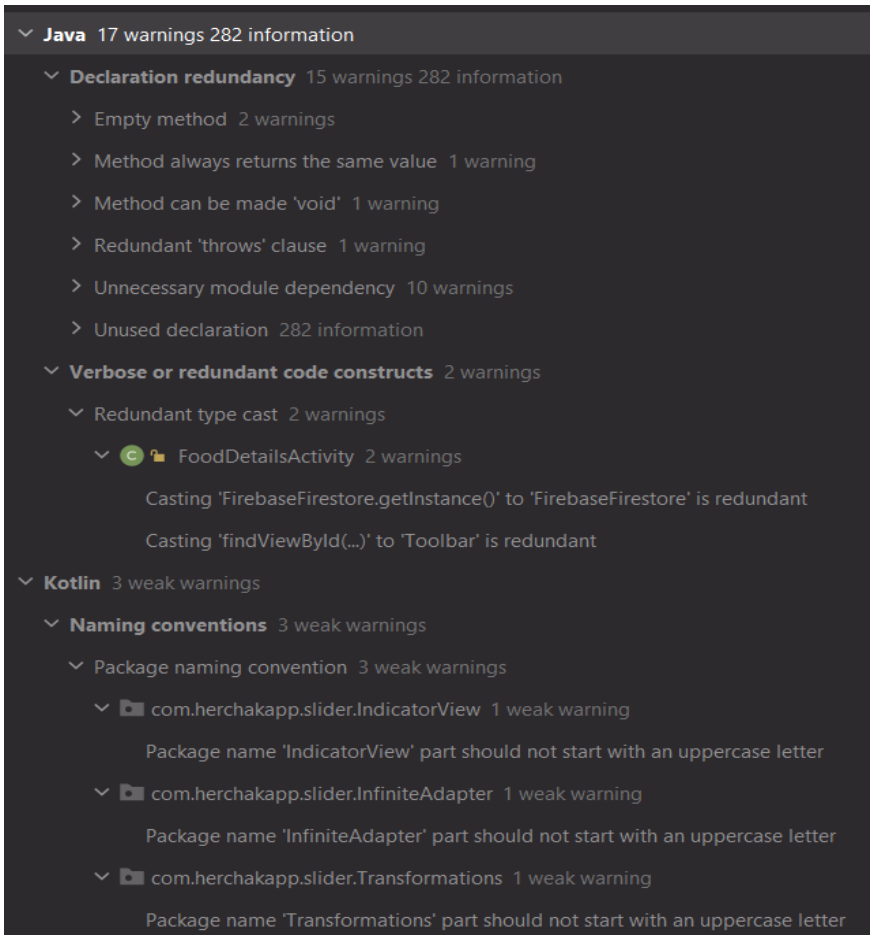

Рисунок 2.13 – Категорії попереджень для мов програмування Java та Kotlin

Були розглянуто попередження для Kotlin:

«Правила іменування пакетів» – Назви пакетів «indicatorView», «infiniteAdapter», «Transformations» не повинні починатися з великої літери.

Процес тестування роботи мобільного додатку пекарні «ГЕРЧАК» було

проведено з використанням технології чорного ящика. Тобто перевірялося конкретне виконання усіх функціональних можливостей додатку, порівнюючи вхідні та вихідні значення, не звертаючи увагу на прихований процес роботи коду.

2.5 Висновки до другого розділу

В другому розділі кваліфікаційної роботи був описаний процес створення бази даних, проектування, розробки та тестування мобільного додатку пекарні «ГЕРЧАК». Був розроблений мобільний додаток для пекарні «ГЕРЧАК». Було проведено тестування та валідацію мобільного додатку пекарні «ГЕРЧАК».

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Шляхи підвищення життєдіяльності людини

Зазвичай, робота програміста не включає важкого фізичного напруження, що може негативно впливати на здоров'я. Проте, дослідження показують, що фізичні тренування значно поліпшують стан та функції основних органів і систем людини.

Взагалі, існують два види фізичної роботи: статична і динамічна. При статичній роботі розхід енергії збільшується, що потребує підвищеного обміну речовин. Така робота швидко викликає втому через тривале напруження м'язів та недостатність кровообігу.

Якщо робота вимагає руху кінцівок або інших частин тіла, її називають динамічною. Така робота розподіляє навантаження поетапно, що не спричиняє застою кровопостачання та кисневого обміну, тому людина менше втомлюється.

Стан та робота м'язів залежать від навантаження. Якщо програміст правильно підбирає середні значення ритму і навантаження, це покращує його продуктивність та затримує появу втоми.

При підборі навантаження слід враховувати професійну діяльність, особистість та фізичний стан людини. Зазвичай, програмісти не мають фізичної підготовки, що може погіршувати стан серцево-судинної, дихальної та центральної нервової систем.

Для покращення стану цих систем та підвищення життєдіяльності людини необхідно включити раціональне харчування. Крім того, існують інші способи покращення, такі як фармакологічні препарати, лікарські рослини, фізіотерапія, масаж, загартовування тощо.

Харчування є найважливішим фактором, що впливає на організм людини, оскільки забезпечує енергетичні ресурси для утворення гормонів та ферментів, що регулюють обмін речовин в тканинах.

Раціональне харчування має три принципи: відповідність енергетичним витратам організму, відповідність хімічного складу їжі фізіологічним потребам організму та різноманітність продуктів.

Психологічні методи підвищення життєдіяльності та відновлення працездатності включають психотерапію, психопрофілактику та психогігієну.

Психопрофілактика включає вправи, що навчають релаксації м'язів та контролю над розумовою діяльністю.

Психогігієна включає взаємовідносини з людьми, гармонію з природою, комфортні умови побуту та різноманітні види відпочинку.

Під час розробки та використання програмного забезпечення важливо не забувати про шляхи підвищення життєдіяльності, особливо для осіб, що належать до першої групи фізичної зайнятості (особи, які переважно зайняті розумовою працею), якою є програмісти.

3.2 Інструкція для обслуговуючого персоналу на випадок виникнення аварії, пожежі

Виготовлення продукції потребує суворого дотримання певних умов. Основними умовами, які можуть призвести до пожеж, є контроль температурного режиму та утримання приміщення в сухості.

Неправильне функціонування нагрівальних елементів може призвести до короткого замикання або поломки, що загрожує пожежею та безпеці персоналу пекарні.

У разі пожежі персонал повинен негайно повідомити оперативно-рятувальну службу цивільного захисту за номером 101.

Під час дзвінка необхідно надати інформацію про адресу, кількість поверхів будівлі, місце виникнення пожежі та наявність людей. Також можуть запитати ваше прізвище для подальшої співпраці.

Після повідомлення про пожежу необхідно негайно і спокійно повідомити всіх про термінову евакуацію, використовуючи систему оповіщення, якщо така є. Треба перерахувати всіх евакуйованих і в разі відсутності працівників дізнатися їхнє місцеперебування і передати цю інформацію представнику оперативно-рятувальної служби, яка прибула на місце.

Наступним етапом є гасіння загоряння або осередку пожежі до прибуття оперативно-рятувальної служби. Для цього необхідно відключити електрику в будівлі та почати гасіння за допомогою первинних засобів пожежогасіння.

Якщо в приміщенні є системи протипожежного захисту, треба перевірити їхню

роботу або активувати їх.

Якщо немає прямої загрози, потрібно евакуювати матеріальні цінності згідно з планом першочерговості евакуації, а також забезпечити безпеку печаток, штампів.

При зустрічі з оперативно-рятувальною службою цивільного захисту, слід повідомити керівника служби про наступне: наявність людей у будівлі, яким загрожує пожежа, їхню кількість, місце виникнення пожежі, горючі приміщення та можливе поширення вогню і диму, а також місцезнаходження пожежних гідрантів.

Зважаючи на те, що ми користуємось цим посібником на пекарні, важливо знати, як діяти у випадку пожежі.

3.3 Вимоги до профілактичних медичних оглядів для працівників ПК

У програмістів, які працюють з ПК протягом тривалого часу, можуть виникати проблеми, пов'язані зі здоров'ям, такі як серцево-судинні захворювання, головні болі, болі в суглобах і т.д.

Отже, працівники, що використовують комп'ютери, повинні пройти обов'язкові медичні огляди. Обов'язковість таких оглядів для працівників, які працюють з електронно-обчислювальними машинами, встановлена в Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПіН 3.3.2.007-98) [8].

Медичні огляди працівників, які працюють з візуальними дисплейними терміналами електронно-обчислювальних машин, включаючи колективно використовувані та персональні комп'ютери, є обов'язковими.

Ці огляди проводяться попередньо при прийомі на роботу і періодично протягом трудової діяльності. Вони здійснюються згідно з вимогами Порядку проведення медичних оглядів певних категорій працівників, який був затверджений Міністерством охорони здоров'я від 21.05.2007 р. № 246 [9].

Медичні огляди повинні проводитись раз на два роки і включати консультацію з терапевта, невропатолога та офтальмолога. У разі необхідності можуть бути залучені лікарі інших спеціальностей.

Метою цих оглядів є вчасне виявлення захворювань, впливу загальних і професійних факторів праці, постійного контролю за станом здоров'я в умовах

шкідливих та небезпечних виробничих факторів, розробка програм оздоровлення та реабілітації працівників, які потрапляють у групу ризику.

Проведення медичних оглядів також має на меті оцінку придатності працівників. Це включає перевірку гостроти зору, реакції, адаптації та стану ока, а також оцінку загального стану організму.

Остаточне рішення про придатність до роботи приймається індивідуально з урахуванням особливостей працівника, умов праці та результатів додаткових медичних оглядів.

Хронічні форми психічних захворювань, захворювання ендокринної системи, серйозні проблеми з бронхіальною системою, гіпертонічна хвороба III стадії та інші захворювання є протипоказаннями для роботи з комп'ютером.

Протягом медичного огляду роботодавець повинен зберігати робоче місце та заробітну плату працівника. Після огляду роботодавець повідомляє працівника про можливість продовжувати або припинити роботу.

Під час виконання кваліфікаційної роботи бакалавра, яка включала роботу з ПК, були дотримані відповідні профілактичні огляди.

3.4 Висновки до третього розділу

В третьому розділі кваліфікаційної роботи описано шляхи підвищення життєдіяльності людини. Розглянуто інструкцію для обслуговуючого персоналу на випадок виникнення аварії, пожежі та вимоги до профілактичних медичних оглядів для працівників ПК.

ВИСНОВКИ

Сучасне виробництво характеризується постійним ростом об’ємів інформації, що оброблюється та збільшенням вимог до якості управління. Це створює передумови для пошуку нових інформаційних технологій управління на основі сучасних засобів обчислювальної техніки.

У кваліфікаційній роботі освітнього рівня «Бакалавр» висвітлено питання інформаційного забезпечення теоретичні та практичні аспекти: описано склад та організацію інформаційного забезпечення, розглянуто організацію збору та передачі інформації, описано прийняті в мобільному додатку методи класифікації та кодування об’єктів, розглянуто перелік вхідних та вихідних даних, які характеризують предметну область.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Проаналізовано програмне забезпечення, що використовується для розробки мобільних додатків.

– Описано організацію замовника.

– Виділено основні сутності мобільного додатку.

– Описано вимоги до розробки мобільного додатку.

В другому розділі кваліфікаційної роботи:

– Спроектовано базу даних.

– Спроектовано архітектуру мобільного додатку «ГЕРЧАК»;

– Описано інтерфейс мобільного додатку пекарні «ГЕРЧАК»;

– Проведено тестування та валідацію мобільного додатку пекарні «ГЕРЧАК»;

У розділі «Безпека життєдіяльності, основи хорони праці» описано шляхи підвищення життєдіяльності людини. Розглянуто інструкцію для обслуговуючого персоналу на випадок виникнення аварії, пожежі та вимоги до профілактичних медичних оглядів для працівників ПК.

ПЕРЕЛІК ДЖЕРЕЛ

1. Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Android_Studio – Дата доступу 28.04.2023.

2. Мова програмування Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java – Дата доступу 28.04.2023.

3. Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Firebase – Дата доступу 28.04.2023.

4. Типи мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/types-of-mobile-applications/ – Дата доступу 28.04.2023.

5. Товариство з обмеженою відповідальністю «ГЕРЧАК» [Електронний ресурс] – Режим доступу до ресурсу: https://gerchak.com.ua/ – Дата доступу 28.04.2023.

6. Inspection code on Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/write/lint/ – Дата доступу 01.05.2023.

7. Мова програмування Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Kotlin – Дата доступу 01.05.2023.

8. Наказ N 246 [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text – Дата доступу: 08.05.2023.

9. МОЗ від 21.05.2007 р. № 246 [Електронний ресурс] – Режим доступу до ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/RE14113.html – Дата доступу: 08.05.2023.

10. Sololearn. Learn to programming [Електронний ресурс] – Режим доступу до ресурсу: https://www.sololearn.com – Дата доступу 10.05.2023.

11. Розробка мобільних додатків від А до Я [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/ – Дата доступу 11.05.2023.

12. Створення мобільних додатків на iOS та Android [Електронний ресурс] – Режим доступу до ресурсу: https://ideil.com/services/mobile-apps – Дата доступу 11.05.2023.

13. Вартість мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://forbes.ua/business/mobilniy-dodatok-komu-vin-diysno-potribniy-i-skilki-

koshtue-rozrobka-17052021-1604 – Дата доступу 12.05.2023.

14. Мобільний та веб-додаток [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/mobilnij-ta-veb-dodatok-u-chomu-rizniczya/ – Дата доступу 12.05.2023.

15. Прогресивні веб-додатки [Електронний ресурс] – Режим доступу до ресурсу: https://smile-ukraine.com/ua/pwa/introduction – Дата доступу 13.05.2023.

16. Гібридні мобільні додатки та їх переваги [Електронний ресурс] – Режим доступу до ресурсу: https://web4u.in.ua/blog/g-bridn-mob-l-n-dodatki-ta-h-perevagi-18 – Дата доступу 13.05.2023.

17. Проекти та методи гібридних розробок [Електронний ресурс] – Режим доступу до ресурсу: https://visuresolutions.com/uk/requirements-management-traceability-guide/hybrid-development – Дата доступу 14.05.2023.

18. Мобільні аплікації [Електронний ресурс] – Режим доступу до ресурсу: https://www.sitegist.com/uk/services/mobile – Дата доступу 15.05.2023.

19. Кросплатформова розробка мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://webbookstudio.com/ua/mobile-development/cross-platform-app-development/ – Дата доступу 15.05.2023.

20. Тестування мобільних додатків від А до Я [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/testuvannia-mobilnykh-dodatkiv-vid-a-do-ia/ – Дата доступу 16.05.2023.

21. Переваги та недоліки кросплатформної та нативної розробки мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ua/blog/cross-platform-native-mobile-development/ – Дата доступу 16.05.2023.

22. Чим відрізняються нативні та гібридні мобільні додатки [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/chem-otlichajutsja-nativnoe-i-gibridnoe-mobilnye-prilozhenija – Дата доступу 17.05.2023.

23. Майбутнє розробки мобільних додатків: тренди 2023-25 років. Частина 1 [Електронний ресурс] – Режим доступу до ресурсу: https://ain.ua/2023/03/30/majbutnye-rozrobky-mobilnyh-dodatkiv-dominuyuchi-trendy-2023-25-rokiv/ – Дата доступу 17.05.2023.

24. Мови програмування для мобільної розробки [Електронний ресурс] –

Режим доступу до ресурсу: https://code.tutsplus.com/uk/articles/mobile-development-languages--cms-29138 – Дата доступу 18.05.2023.

25. Порівняння Java та Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://www.toptal.com/kotlin/kotlin-vs-java – Дата доступу 19.05.2023.

26. Jetpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://android-developers.googleblog.com/2023/05/whats-new-in-jetpack-compose.html – Дата доступу 20.05.2023.

27. Kotlin in Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin – Дата доступу 20.05.2023.

28. How to install Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/install –Дата доступу 21.05.2023.

29. Python Kivy Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://realpython.com/mobile-app-kivy-python/ – Дата доступу 21.05.2023.

30. Xamarin Android [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/xamarin-android – Дата доступу 22.05.2023.

31. Android NDK [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Android_NDK – Дата доступу 22.05.2023.

32. React Native [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/React_Native – Дата доступу 23.05.2023.

33. PWA – Прогресивний веб-додаток – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B2%D0%B5%D0%B1%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA – Дата доступу 23.05.2023.

34. 12 найкращих прогресивних веб-додатків PWA – Режим доступу до ресурсу: https://www.simicart.com/blog/progressive-web-apps-examples/ – Дата доступу – 23.05.2023.

35. 10 найкращих гібридних додатків – Режим доступу до ресурсу: https://designli.co/blog/5-best-hybrid-app-examples/ – Дата доступу – 24.05.2023.

36. 15 найкращих нативних додатків у 2023 році – Режим доступу до ресурсу: https://www.androidauthority.com/best-android-apps-312570/ – Дата доступу 24.05.2023.

ДОДАТКИ

Додаток А

Програмний код мобільного додатку пекарні «ГЕРЧАК»

Лістинг А.1 – Код LoginActivity

```
package com.vitaliy18.pekarnya;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.*;
public class LoginActivity extends AppCompatActivity {
     EditText email, password;
     Button login;
     TextView sign_up, forget_pass;
     FirebaseAuth fAuth;
     FirebaseFirestore fStore;
     ProgressBar progressBar;
     @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_login);
                email = findViewById(R.id.email);
            password = findViewById(R.id.password);
            login = findViewById(R.id.login);
            sign_up = findViewById(R.id.sign_up);
            forget_pass = findViewById(R.id.forget_pwd);
            fAuth = FirebaseAuth.getInstance();
            fStore = FirebaseFirestore.getInstance();
            progressBar = findViewById(R.id.progressBar);
            login.setOnClickListener(v -> {
            String emailValue = email.getText().toString().trim();
            String                    passwordValue                    =
            password.getText().toString().trim();
            if(TextUtils.isEmpty(emailValue)){
                email.setError("Email обов'язковий!");
                return;
            } if(TextUtils.isEmpty(passwordValue)){
                password.setError("Пароль обов'язковий!");
                return;
            } if(passwordValue.length() < 8){
                password.setError("Пароль повинен містити не менше 8
                символів!");
                return;
            }
            progressBar.setVisibility(View.VISIBLE);
            fAuth.signInWithEmailAndPassword(emailValue,
            passwordValue).addOnSuccessListener(authResult -> {
                Toast.makeText(LoginActivity.this, "Авторизація пройшла
                успішно", Toast.LENGTH_LONG).show();
                checkUserAccessLevel(authResult.getUser().getUid());
```

```
            }).addOnFailureListener(e -> {
                Toast.makeText(LoginActivity.this, "Помилка !! " +
                e.getMessage(), Toast.LENGTH_LONG).show();
                progressBar.setVisibility(View.GONE);
                });
            });
            sign_up.setOnClickListener(v      ->      startActivity(new
            Intent(LoginActivity.this, RegisterActivity.class)));
            forget_pass.setOnClickListener(v -> {
                String emailValue = email.getText().toString().trim();
                if(TextUtils.isEmpty(emailValue)){
                    Toast.makeText(LoginActivity.this, "Введіть власну
                    електронну    пошту    аби    скинути    пароль",
                    Toast.LENGTH_SHORT).show();
                    return;
                }
                progressBar.setVisibility(View.VISIBLE);
            });
        }
        private void checkUserAccessLevel(String uid) {
            DocumentReference df =
            fStore.collection("Users").document(uid);
            df.get().addOnCompleteListener(task -> {
                if(task.isSuccessful()){
                    DocumentSnapshot document = task.getResult();
                    if(document.exists()){
                        if(document.get("isAdmin")!= null){
                            startActivity(new
                            Intent(LoginActivity.this,
                            AdminFrontPageActivity.class));
                        } else{
                            startActivity(new
                            Intent(LoginActivity.this,
                            HomePageActivity.class));
                        }
                    }
                }
            });
        }
    }
```

Лістинг А.2 – Код RegisterActivity

```
package com.vitaliy18.pekarnya;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.auth.*;
import com.google.firebase.firestore.*;
import java.util.*;
public class RegisterActivity extends AppCompatActivity {
    EditText email, username, full_name, password, confirm_pwd;
    Button sign_up;
```

```
TextView login;
FirebaseAuth fAuth;
FirebaseFirestore fStore;
ProgressBar progressBar;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    email = findViewById(R.id.email);
    username = findViewById(R.id.username);
    full_name = findViewById(R.id.full_name);
    password = findViewById(R.id.password);
    confirm_pwd = findViewById(R.id.confirm_pwd);
    sign_up = findViewById(R.id.sign_up);
    login = findViewById(R.id.sign_in);
    fAuth = FirebaseAuth.getInstance();
    fStore = FirebaseFirestore.getInstance();
    progressBar = findViewById(R.id.progressBar);
    if(fAuth.getCurrentUser() != null){
        startActivity(new Intent(RegisterActivity.this,
        MainActivity.class));
        finish();
    }
    sign_up.setOnClickListener(v -> {
        String emailValue = email.getText().toString().trim();
        String usernameValue =
        username.getText().toString().trim();
        String full_nameValue =
        full_name.getText().toString().trim();
        String passwordValue =
        password.getText().toString().trim();
        String confirm_pwdValue =
        confirm_pwd.getText().toString().trim();
        if(TextUtils.isEmpty(usernameValue)){
            username.setError("Логін є обов'язковим!");
            return;
        }
        if(TextUtils.isEmpty(passwordValue)){
            password.setError("Пароль є обов'язковим!");
            return;
        }
        if(passwordValue.length() < 8){
            password.setError("Пароль повинен містити не менше 8
            символів!");
            return;
        }
        if(TextUtils.isEmpty(confirm_pwdValue)){
            confirm_pwd.setError("Пароль є обов'язковим!");
            return;
        }
        if(!passwordValue.equals(confirm_pwdValue)){
            confirm_pwd.setError("Паролі не співпадають!");
            return;
        }
        if(TextUtils.isEmpty(full_nameValue)){
            full_name.setError("Ім'я та прізвище є
            обов'язковими!");
```

```
                return;
            }
            if(TextUtils.isEmpty(emailValue)){
                email.setError("Email є обов'язвовим!");
                return;
            }
            progressBar.setVisibility(View.VISIBLE);
            fAuth.createUserWithEmailAndPassword(emailValue,
            passwordValue).addOnCompleteListener(task -> {
            if(task.isSuccessful()){
                FirebaseUser user = fAuth.getCurrentUser();
                Toast.makeText(RegisterActivity.this,       "Реєстрація
                пройшла успішно!", Toast.LENGTH_LONG).show();
                DocumentReference                  df                  =
                fStore.collection("Users").document(user.getUid());
                Map<String, Object> userInfo = new HashMap<>();
                userInfo.put("Email", emailValue);
                userInfo.put("UserName", usernameValue);
                userInfo.put("FullName", full_nameValue);
                userInfo.put("isUser",1);
                df.set(userInfo);
                startActivity(new         Intent(RegisterActivity.this,
                LoginActivity.class));
                finish();
            }
            else{
                Toast.makeText(RegisterActivity.this, "Помилка !! " +
                task.getException().getMessage(),
                Toast.LENGTH_LONG).show();
                progressBar.setVisibility(View.GONE);
            }
        });
    });       login.setOnClickListener(v      ->      startActivity(new
    Intent(RegisterActivity.this, LoginActivity.class))); }
}
```

Лістинг А.3 – Код CategoryFoodActivity

```
package com.vitaliy18.pekarnya;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.*;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.*;
import
com.google.android.libraries.places.api.model.AutocompleteSessionToken
;
import com.google.firebase.firestore.*;
import com.google.firebase.storage.StorageReference;
import com.herchakapp.materialsearchbar.MaterialSearchBar;
import java.util.*;
```

```
public class CategoryFoodActivity extends AppCompatActivity implements
RecyclerViewClickInterface {
    ImageView back;
    SearchView search;
    private List<CategoryModel> lastSearches;
    FirebaseFirestore firestore;
    StorageReference storageReference;
    Query query,itemQuery;
    RecyclerView dataList;
    List<FoodListModel> itemsListModel;
    List<CategoryModel>categoryList;
    FoodListAdapter itemAdapter;
    List<CategoryModel> titles;
    List<CategoryModel> images;
    private MaterialSearchBar materialSearchBar;
    CategoryAdapter adapter;
    String name;
    List<String> list;
    @SuppressLint({"MissingInflatedId", "NotifyDataSetChanged"})
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_category_food);
        itemsListModel= new ArrayList<>();
        list=new ArrayList<>();
        back= findViewById(R.id.back);
        dataList= findViewById(R.id.dataList);
        firestore= FirebaseFirestore.getInstance();
        categoryList= new ArrayList<>();
        final                                    AutocompleteSessionToken
        token=AutocompleteSessionToken.newInstance();
        materialSearchBar=findViewById(R.id.searchBar);
        itemQuery = firestore.collection("Foods");
        query = firestore.collection("Category").orderBy("category");
        query.get().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                QuerySnapshot snapshot = task.getResult();
                if (snapshot != null) {
                    for    (QueryDocumentSnapshot    querySnapshot    :
                    snapshot) {
                        CategoryModel             category             =
                        querySnapshot.toObject(CategoryModel.class);
                        category.setId(querySnapshot.getId());
                        categoryList.add(category);
                    }
                    adapter                        =                    new
                    CategoryAdapter(CategoryFoodActivity.this,
                    categoryList,CategoryFoodActivity.this);
                    GridLayoutManager    gridLayoutManager    =    new
                    GridLayoutManager(CategoryFoodActivity.this,     2,
                    GridLayoutManager.VERTICAL, false);
                    dataList.setLayoutManager(gridLayoutManager);
                    dataList.setHasFixedSize(true);
                    dataList.setAdapter(adapter);
                    new Handler().postDelayed(() -> {
                        adapter.showShimmer=false;
                        adapter.notifyDataSetChanged();
```

```
                    },1000);
                }
            } else {
                String error = task.getException().getMessage();
                Toast.makeText(CategoryFoodActivity.this,        error,
                Toast.LENGTH_SHORT).show();
                }
            });
            materialSearchBar.setOnSearchActionListener(new
            MaterialSearchBar.OnSearchActionListener() {
                @Override
                public void onSearchStateChanged(boolean enabled) {}
                @Override
                public void onSearchConfirmed(CharSequence text) {
                    startSearch(text.toString(),true,null,true);
                }
                @Override
                public void onButtonClicked(int buttonCode) {}
            });
            materialSearchBar.addTextChangeListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
            count, int after) {}
            @Override
            public void onTextChanged(final CharSequence s, int start,
            int before, int count) {
                adapter.getFilter().filter(s);
            }
            @Override
            public void afterTextChanged(Editable s) { }
        });
        back.setOnClickListener(v -> {
            Intent  intent  =  new  Intent(CategoryFoodActivity.this,
            HomePageActivity.class);
            startActivity(intent);
            finish();
        });
    }
    @Override
    public void onItemClick(FoodListModel position) {}
    @Override
    public void onButtonClick(FoodListModel position) {}
    public void getItemName(String name){
        this.name=name;
    }
}
```

Лістинг А.4 – Код InsertFoodActivity

```
package com.vitaliy18.pekarnya;
import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Build;
```

```
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.*;
import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.*;
import java.io.IOException;
import java.util.*;
public class InsertFoodActivity extends AppCompatActivity implements
    AdapterView.OnItemSelectedListener {
    Button upload, choose;
    EditText title, price, description, exclusive, foodId;
    Spinner inx;
    ImageView image_name;
    StorageReference storageRef;
    FirebaseFirestore fStoreRef;
    Collection colRef;
    Uri imageUrl;
    final int PICK_IMAGE_REQUEST = 18;
    String item;
    final String[] categories = {"Виберіть категорію продукції:",
    "Гріссіні", "Грінки", "Крутони"};
    @RequiresApi(api = Build.VERSION_CODES.N)
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insert_food);
        title = findViewById(R.id.title);
        foodId = findViewById(R.id.foodId);
        image_name = findViewById(R.id.image_name);
        inx = findViewById(R.id.inx);
        description = findViewById(R.id.description);
        exclusive = findViewById(R.id.exclusive);
        price = findViewById(R.id.price);
        upload = findViewById(R.id.itStore);
        choose = findViewById(R.id.set_image_name);
        storageRef = FirebaseStorage.getInstance().getReference();
        fStoreRef = FirebaseFirestore.getInstance();
        inx.setOnItemSelectedListener(this);
        ArrayAdapter     adapter     =     new     ArrayAdapter(this,
        android.R.layout.simple_spinner_dropdown_item, categories);
        adapter.setDropDownViewResource(android.R.layout.simple_spinne
        r_dropdown_item);
        inx.setAdapter(adapter);
        choose.setOnClickListener(v -> {
            Intent intent = new Intent();
            intent.setType("image/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent,
            "Виберіть зображення"), PICK_IMAGE_REQUEST);
        });
        upload.setOnClickListener(v -> {
        if(imageUrl != null){
```

```
ProgressDialog progressDialog = new
ProgressDialog(InsertFoodActivity.this);
progressDialog.setTitle("Додавання даних про
продукцію...");
progressDialog.show();
StorageReference ref = storageRef.child("foods/" +
UUID.randomUUID().toString());
ref.putFile(imageUrl).addOnSuccessListener(taskSnapshot ->
{
    progressDialog.dismiss();
    Toast.makeText(InsertFoodActivity.this, "Дані про
    продукцію додано!", Toast.LENGTH_LONG).show();
    ref.getDownloadUrl().addOnSuccessListener(uri -> {
        Uri downloadUrl = uri;
        String idValue = foodId.getText().toString();
        String titleValue = title.getText().toString();
        String fileUrl = downloadUrl.toString();
        Double priceValue =
        Double.parseDouble(price.getText().toString());
        String inxValue = inx.getSelectedItem().toString();
        String descriptionValue =
        description.getText().toString();
        int exclusiveValue =
        Integer.parseInt(exclusive.getText().toString());
        addData(idValue, titleValue, fileUrl, priceValue,
        inxValue, descriptionValue, exclusiveValue);
    });
}).addOnFailureListener(e -> {
    progressDialog.dismiss();
    Toast.makeText(InsertFoodActivity.this, "Помилка!
    "+e.getMessage(), Toast.LENGTH_SHORT).show();
}).addOnProgressListener(snapshot -> {
    double progress = (100.0 *
    snapshot.getBytesTransferred()/snapshot.getTotalByteCo
    unt());
    progressDialog.setMessage("Завантажено на " + (int)
    progress + "%"); }); } }); }
    @RequiresApi(api = Build.VERSION_CODES.N)
    private void addData(String idValue, String titleValue,
    String fileUrl, Double priceValue, String inxValue,
    String descriptionValue, int exclusiveValue) {
    FoodListModel food = new FoodListModel(
        idValue,
        titleValue,
        fileUrl,
        priceValue,
        inxValue,
        descriptionValue,
        exclusiveValue
    );
    fStoreRef.collection("Foods").document(food.getId())
    .set(food).addOnSuccessListener(unused ->
        Toast.makeText(InsertFoodActivity.this, "True",
        Toast.LENGTH_LONG).show()).addOnFailureListener(e -
        >
        Toast.makeText(InsertFoodActivity.this, "False",
        Toast.LENGTH_LONG).show());}
```

```
    @Override
    protected void onActivityResult(int requestCode,
    int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode,
        data);
        if(requestCode == PICK_IMAGE_REQUEST &&
        resultCode == RESULT_OK && data != null &&
        data.getData() != null){
            imageUrl = data.getData();
            try{
                Bitmap bitmap =
                MediaStore.Images.Media.getBitmap(getC
                ontentResolver(), imageUrl);
                image_name.setImageBitmap(bitmap);
            }catch (IOException e){
                e.printStackTrace();
            }
        }
    }
    @Override
    public void onItemSelected(AdapterView<?>
    adapterView, View view, int i, long l) {
        item = inx.getSelectedItem().toString();
        @Override
        public void onNothingSelected(AdapterView<?>
        adapterView) {}
}
```