НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Інструментальні засоби розпізнавання та обробки даних двовимірного зображення»

Виконала:
студентка IV курсу, групи ТІ-72
Скиба Надія Володимирівна _____

Керівник:
Доцент, к.е.н.,
Гусєва Ірина Ігорівна _____

Консультант:
_____

Рецензент:
Доцент, к.т.н., _____
Юрій Веремійчук

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ — 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ

на дипломну роботу студентці

Скибі Надії Володимирівні
(прізвище, ім’я, по батькові)

1. Тема роботи Інструментальні засоби розпізнавання та обробки даних двовимірного зображення
керівник роботи Гусєва Ірина Ігорівна, к.е.н.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ” 24 ” травня 2021р.
№ **1267.**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування Dart, фреймворк Flutter, плагін SQLite, середовище розробки Android Studio.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розробити мобільний застосунок для коректної обробки даних двовимірного зображення; розробити алгоритм для виправлення даних пошкоджених двовимірних зображень при їхньому зчитуванні; реалізувати коректне зчитування кольорових зображень; здійснити тестування системи.

5. Перелік ілюстративного матеріалу

Актуальність. Мета та задачі. Двовимірне зображення. Види пошкоджених двовимірних зображень. Код Ріда-Соломона. Алгоритм виявлення та виправлення помилки. Діаграма прецедентів. Діаграма користувацьких потоків. Діаграма класів. Модель бази даних розробленої системи. Засоби розробки. Екран сканера. Вікна мобільного застосунку. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.10.2020	
2.	Вивчення та аналіз задачі	09.10.2020 – 23.01.2021	
3.	Розробка архітектури та загальної структури системи	09.03.2021 – 27.03.2021	
4.	Розробка структур окремих підсистем	28.03.2021 – 01.04.2021	
5.	Програмна реалізація системи	05.04.2021 – 28.04.2021	
6.	Оформлення пояснювальної записки	20.05.2021 – 04.05.2021	
7.	Захист програмного продукту	14.05.2021	
8.	Передзахист	28.05.2021	
9.	Захист	15.06.2021	

Студентка _____ _____

(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

В роботі розглянуто класичні та сучасні засоби розробки програмного забезпечення для розпізнавання та обробки даних двовимірного зображення.

Метою даної роботи є дослідження застосування інструментальних засобів для коректного зчитування даних пошкоджених двовимірних зображень.

Для досягнення мети розроблено архітектуру системи, створено її модель, реалізовано алгоритм виправлення пошкоджених даних при обробці двовимірного зображення, проведено тестування системи на різних даних, проаналізовано невирішені проблеми подібних систем.

Пояснювальна записка складається зі вступу, п’яти розділів, висновку, списку використаних джерел; містить 65 сторінок, 21 рисунки, 3 таблиці та 3 додатки. Список використаних джерел включає 22 бібліографічних найменувань.

Ключові слова: двовимірне зображення, алгоритм виправлення пошкоджень, кольорові QR-коди, Код Ріда-Соломона, мобільний застосунок.

ABSTRACT

The paper considers classifiers and modern software development tools for two-dimensional image recognition and data processing.

This work aims to study the use of tools for the correct reading of data of damaged two-dimensional images.

To achieve this goal, the system architecture was developed, its model was created, an algorithm for correcting damaged data during two-dimensional image processing was implemented, the system was tested on various data, and unresolved problems of such systems were analyzed.

The explanatory note consists of an introduction, five chapters, conclusion, list of used sources; contains 65 pages, 21 figures, 3 tables and 3 applications. The list of used sources includes 22 bibliographic items.

Key words: two-dimensional image, error correction algorithm, color QR-codes, Reed-Solomon Code, mobile application.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
1. ПОСТАНОВКА ЗАДАЧІ РОЗПІЗНАВАННЯ ТА ОБРОБКИ ДАНИХ ДВОВИМІРНОГО ЗОБРАЖЕННЯ 10
2. АНАЛІЗ ПРОБЛЕМИ РОЗПІЗНАВАННЯ ТА ОБРОБКИ ДАНИХ ДВОВИМІРНОГО ЗОБРАЖЕННЯ 12
2.1. Швидке виявлення QR-коду з довільно отриманих зображень 12
2.2. Опис існуючих режимів аналізу даних двовимірного зображення 14
2.3. Алгоритм коригування помилок двовимірного зображення під час обробки даних 19
2.4. Вплив колірної гами на читабельність QR-коду 23
2.5. Огляд програмних застосунків для сканування двовимірних зображень 26
3. ЗАСОБИ РОЗРОБКИ 32
3.1. Технологія розробки мобільних застосунків Flutter та мова програмування Dart 32
3.2. Ефективний спосіб збереження даних для мобільного застосунку 35
3.3. Опис об’єктно-орієнтованої частини системи 37
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТА АЛГОРИТМУ ВИПРАВЛЕННЯ ПОШКОДЖЕНЬ 40
4.1. Опис діаграми прецедентів розробленої системи 40
4.2. Опис діаграми користувацьких потоків 42
4.3. Опис діаграми класів програмної системи 43
4.4. Архітектура та модель бази даних 45

4.5. Опис етапів розгортання створеної програмної системи 46
4.6. Опис алгоритму створення QR-кодів 49
4.7. Опис реалізації алгоритму виправлення пошкоджень 51
5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 55
5.1. Модуль формування QR-кодів 55
5.2. Функціональний модуль сканера QR-кодів 58
ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64
ДОДАТОК А 66
ДОДАТОК Б 68
ДОДАТОК В 77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ECI — Extended Channel Interpretation (Режим розширеної інтерпретації каналів)

FNC1 — Function 1 Symbol Character (спеціальний режим функції символу аналізу даних)

GS1 — General Specification (Загальна специфікація)

BOM — Byte Order Mark (позначка порядку байтів)

БД — база даних

JIT — Just-In-Time (Вчасно, на льоту)

AOT — Ahead-of-Time (Достроково)

СУБД — Система управління базами даних

ВСТУП

Двовимірне зображення — це матричний код, який розпізнається сканувальним обладнанням та має вигляд квадрату з необхідними функціональними елементами. Застосовується як у виробництві, так і в торгівлі.

В процесі створення двовимірного зображення та під час його використання можуть виникати, так звані, пошкодження матричного коду. Двовимірне зображення має можливість виправлення помилок для відновлення даних. Чотири рівні виправлення помилок доступні для вибору відповідно до робочого середовища. Підвищення цього рівня покращує можливість виправлення помилок, але також збільшує обсяг даних. Актуальність цієї теми постає під час розпізнавання двовимірного зображення.

Метою роботи є створення зручного програмного забезпечення мобільного телефону для ефективного зчитування та обробки двовимірного зображення, яке може бути пошкодженим.

Перед початком роботи над системою було поставлено задачу дипломної праці та проаналізовано проблеми, які ще не вирішені та немає реалізованих аналогів. Розглянуто алгоритм, який вирішує поставлену задачу та надано оцінку ефективності створеного програмного продукту. Також було проведено дослідження інших подібних систем, які вже використовуються.

В третьому розділі описано засоби та інструменти реалізації архітектури програмної системи, а в розділі чотири — наведено детальний опис всієї системи. Проведено ряд тестів над програмою, щоб отримати остаточну оцінку ефективності написаного алгоритму. Проводиться опис роботи користувача з програмним інтерфейсом, а також аналіз потенційних користувачів програмної системи.

В четвертому розділі розглядається програмне проектування та реалізація. Визначається кожен етап розгортання створеної програми.

В п’ятому розділі описується взаємодія користувача з програмною системою.

1. ПОСТАНОВКА ЗАДАЧІ РОЗПІЗНАВАННЯ ТА ОБРОБКИ ДАНИХ ДВОВИМІРНОГО ЗОБРАЖЕННЯ

Мета: створення зручного програмного забезпечення мобільного телефону для ефективного зчитування та обробки двовимірного зображення, яке може бути пошкодженим.

Основним модулем системи є сканер для виконання функції зчитування даних двовимірного зображення. Його призначенням є розпізнавання та обробка даних, а також функціонал, який враховує ймовірне пошкодження зображення. Іншою індивідуальною задачею є вирішення питання впливу кольору зображення на процес зчитування даних.

В процесі створення власного двовимірного зображення може бути додано різноманітні кольори та дизайн зображення, що створює так зване пошкодження матричного коду. Проте допускається невеликий відсоток відхилення від точності даних коду для того, щоб створювати свій унікальний дизайн.

Задачі, необхідні для реалізації індивідуального плану:

— Проаналізувати процес розпізнавання двовимірного зображення.

— Дослідити процес створення QR- кодів.

— Спроектувати архітектуру програмної системи.

— Розробити алгоритм програмної системи. Створити основний модуль дипломної роботи ─ модуль розпізнавання та обробки даних.

— Реалізувати зручний користувацький інтерфейс системи.

— Тестувати та перевірити ефективність роботи усієї програмної системи. Випробувати систему на різних вхідних даних:

— QR-коди з різним ступенем пошкодження;

— QR-коди усіх версій;

— великі та малі двовимірні зображення різної чіткості.

Вимоги до розробленої програмної системи:

— автоматичне визначення версії / типу QR-коду або введення вручну;

— виправлення помилок / кольору;

— настроюваний розмір виводу, відступу, кольору фону і переднього плану;

— підтримка QR-коду версії 1 – 40.

— можливість перегляду створених об’єктів, які містять дані QR-кодів;

— зручний користувацький інтерфейс, який дозволяє також змінювати поточну інформацію та додавати нові об’єкти.

Вихідним даними програми є оброблена та дешифрована інформація QR-коду, визначення типу двовимірного зображення.

2. АНАЛІЗ ПРОБЛЕМИ РОЗПІЗНАВАННЯ ТА ОБРОБКИ ДАНИХ ДВОВИМІРНОГО ЗОБРАЖЕННЯ

2.1. Швидке виявлення QR-коду з довільно отриманих зображень

Існуючі дешифратори QR-коду вимагають правильного дизайну символів, один такий символ повинен відповідають принаймні 30% площі зображення, щоб бути придатним для декодування. Дешифратори можуть вийти з ладу через різні пошкодження та спотворення. У разі неправильного зчитування даних у QR використовуються спеціальні коди, що можуть виправити дефекти читання. Також є інформація про версію коду. Максимальний обсяг даних, який можна записати в код, залежить від версії коду. Коли оновлюється версія, додаються спеціальні блоки.

Розпізнавання QR-коду на зображенні — добре поставлена задача для машинного програмного забезпечення. По-перше, в завданні досліджується об'єкт, який спеціально розроблений для «зручного» розпізнавання. По-друге, сама задача розбивається на кілька незалежних зрозумілих підзадач: локалізація QR-коду, орієнтація QR-коду і безпосередньо декодування QR-коду. Сучасні програми вже досить давно володіють хорошими бібліотеками, здатними вирішити останні два завдання: орієнтацію і декодування QR-коду. Але є інша проблема: для якісного декодування такі бібліотеки очікують на вхід хороше двовимірне зображення безпосередньо QR-коду. І навпаки, задачі локалізації та корегування дефектів на зображенні приділяється мало уваги [1].

На відміну від одновимірного штрих-коду, який повинен знаходити тонкий промінь, QR-код визначається датчиком як двовимірне зображення, але його можна прочитати в будь-якому напрямку. Три великі квадрати у кутах зображення та контрольній точці біля четвертого кута дозволяють нормалізувати розмір

зображення та його орієнтацію, а також кут, під яким знаходиться камера - зчитувач розміщений до зображення. QR-код можна також зчитати «вручну» без камери телефона, для цього вам потрібно знати особливості QR-кодів та алгоритм дешифрування інформації.

QR-код використовує двійкове кодування інформації: чорні квадрати кодуються одиницями, білі ─ нулями. Крім того, для виявлення та виправлення помилок декодування контрольні суми перевіряються за допомогою XOR операції (додавання бітів коду за модулем два до спеціальної восьмирозрядної двійкової маски, наприклад 10101010) [2]. Завдяки виправленню помилок, можна застосувати малюнок до коду, зробити його кольоровим і різнокольоровим ─ він також залишиться читабельним.

Розрізняють статичні та динамічні QR-коди. Статичний QR-код містить інформацію, вказану під час її створення. Динамічний QR-код ─ це багатофункціональний код: його можна підключити до додаткових функцій, які будуть виконуватися одночасно або змінювати один одного. На рисунку 2.1 зображено можливу архітектура декодера QR-коду.

Хоча візуально всі QR-коди схожі між собою, різні екземпляри QR-кодів в залежності від обсягу закодованих даних можуть мати різне компонування внутрішніх елементів. Крім того, великою популярністю користуються “дизайнерські” QR-коди, в яких замість частини додаткової інформації, яка гарантує якісне розпізнавання двовимірного зображення, використовуються сторонні графічні елементи (логотипи, емблеми, написи). Всі ці особливості повинні бути обов'язково враховані при побудові методів розпізнавання та обробки QR-кодів.

Використання QR-кодів не обмежується ніякими ліцензіями, вони також описані та опубліковані в якості стандартів ISO [3]. Найбільше визнання QR-код отримав серед користувачів мобільного зв’язку.

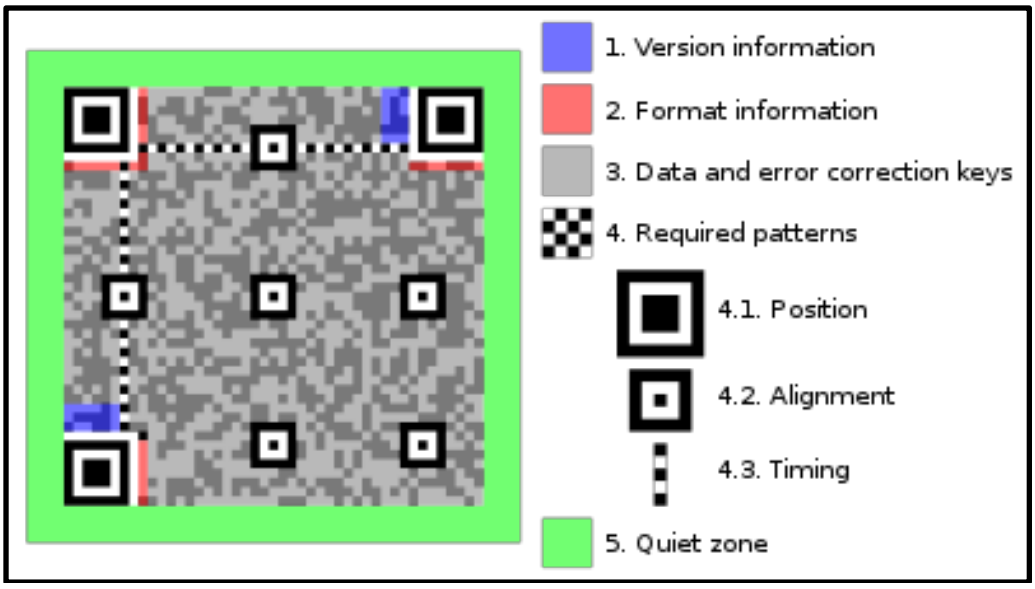

Рисунок 2.1 — Архітектура дешифратора QR-коду

Перевагою QR-кодів є висока ймовірність розпізнавання інформації навіть у випадку його пошкодження. QR-кодування є зручним, використовується в багатьох сферах, набуває популярності і лише почало використовуватися в банківській справі та електронній комерції. Серед альтернативних технологій найбільш розповсюджені RFID та NFC.

2.2. Опис існуючих режимів аналізу даних двовимірного зображення

Двовимірне зображення кодує рядок тексту. Стандарт QR-коду має чотири режими кодування тексту: числовий, буквено-цифровий, байтовий та Kanji. Кожен режим кодує текст як рядок бітів (1s та 0s), але кожен режим використовує інший метод для перетворення тексту в біти. Кожен метод оптимізований для генерації найкоротшого можливого рядка бітів для відповідного типу даних. У цьому підрозділі досліджено та проаналізовано ефективність використання кожного режиму.

QR-коди версії 2 та новішої повинні мати шаблони вирівнювання. Місця розташування шаблонів вирівнювання визначаються в таблиці розташувань

об’єднання шаблонів вирівнювання. Цифри повинні використовуватися як об’єднання координат рядків і стовпців [4]. Наприклад, версія 2 має цифри 6 і 18. Це означає, що центральні модулі шаблонів вирівнювання мають бути розміщені в (6, 6), (6, 18), (18, 6) та (18, 18) . Проте шаблони вирівнювання повинні бути введені в матрицю після розміщення шаблонів шукача та роздільників, і шаблони вирівнювання не повинні перекривати шаблони пошуку або роздільники. На рисунку 2.2 показано код версії 2, який описаний вище як такий, що має шаблони вирівнювання з центром у (6, 6), (6, 18), (18, 6) та (18, 18). Однак, як показано на зображенні зліва, шаблони вирівнювання, виділені червоним кольором, не можна розміщувати в матриці, оскільки вони перекривають шаблони пошуку та роздільники. Шаблони вирівнювання, які перекривають шаблони шукачів або роздільники, просто не вказано в матриці.

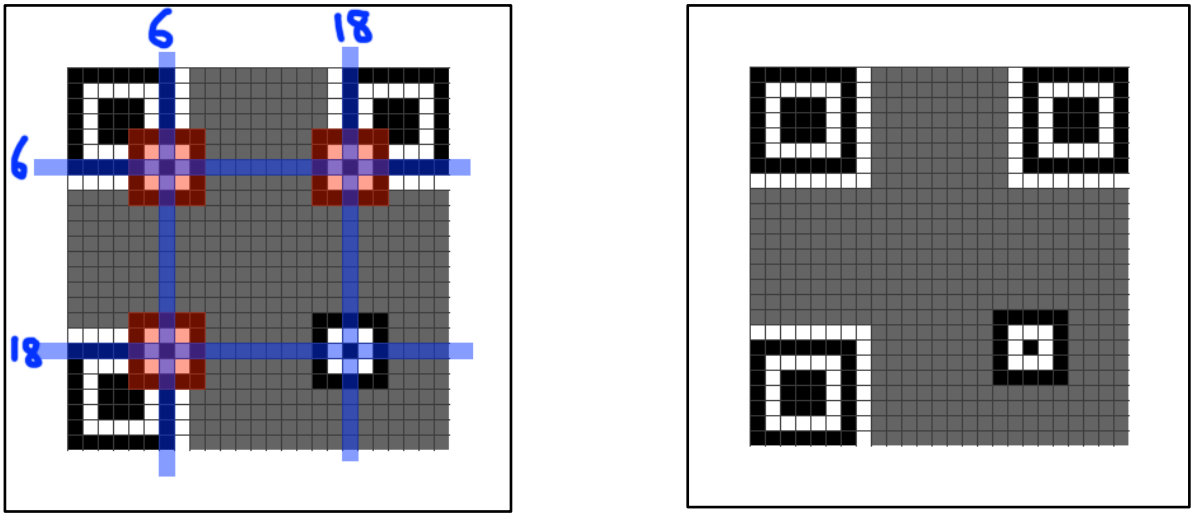

Рисунок 2.2 — Некоректне та правильне розміщення шаблонів вирівнювання в матриці QR-коду

Для подальшої ілюстрації розміщення шаблону вирівнювання, на наступному рисунку 2.3 показано схеми вирівнювання для QR-коду версії 8. У таблиці розташувань шаблонів вирівнювання перелічені 6, 24 і 42 як розташування зразків вирівнювання для версії 8. Усі комбінації цих трьох чисел використовуються як координати для зразків вирівнювання.

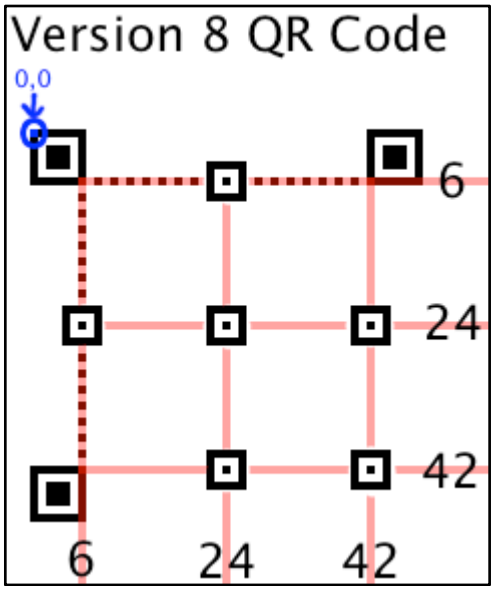

Рисунок 2.3 — Демонстрація розміщення коректного шаблону вирівнювання на QR-коді версії 8

Чотири режими кодування включають такі символи:

1. Числовий режим призначений для десяткових цифр від 0 до 9.

2. Буквено-цифровий режим призначений для десяткових цифр від 0 до 9, а також великих літер та символів $, %, *, +, -, ., /, :, а також пробілу.

3. Байтовий режим за замовчуванням призначений для символів із набору символів ISO-8859-1. Однак деякі сканери QR-коду можуть автоматично виявляти, чи використовується UTF-8 замість цього в байтовому режимі.

4. Режим Kanji призначений для двобайтових символів із набору символів Shift JIS. Хоча UTF-8 може кодувати символи Kanji, для цього він повинен використовувати три або чотири байти. З іншого боку, Shift JIS використовує лише два байти для кодування кожного символу Kanji, тому режим Kanji ефективніше стискає ці символ. Якщо весь вхідний рядок складається з символів у двобайтовому діапазоні Shift JIS, рекомендується використовувати режим Kanji [4] Також можна використовувати кілька режимів в межах одного QR-коду, які описані далі в цьому підрозділі.

Режим розширеної інтерпретації каналів (ECI) безпосередньо визначає набір символів (наприклад, UTF-8). Однак деякі пристрої для зчитування QR-кодів не підтримують режим ECI і не розуміють QR-кодів, які його використовують.

Режим структурованого додавання кодує дані у кількох QR-кодах, максимум до 16 QR-кодів.

Режим FNC1 дозволяє QR-коду функціонувати як штрих-код GS1 [5].

Деякі зчитувачі QR-кодів можуть розпізнати, коли UTF-8 використовується в байтовому режимі. Оскільки всі символи Shift JIS мають представлення в UTF-8, можна використовувати байтовий режим для Kanji з кодуванням UTF-8.

Однак Kanji в UTF-8 кодуються трьома байтами (або чотирма, у рідкісних випадках), тоді як символи Shift JIS кодуються двома або одним байтом. Іншими словами, не вдасться помістити стільки символів у QR-код, якщо використовувати UTF-8 у байтовому режимі для Kanji. Використання режиму Kanji для Shift JIS дає найбільшу ємність. Тому реалізація цього режиму залежить від потреб користувачів розробленої системи.

Інший метод — поставити позначку порядку байтів UTF-8 (BOM) перед вхідним текстом. Деякі зчитувачі QR-коду прочитають позначку порядку байтів і зрозуміють, що текст закодований в UTF-8. Не всі зчитувачі QR-кодів можуть правильно інтерпретувати це. Знак порядку байтів для UTF-8 — це набір з трьох чисел, переведених у шістнадцяткову систему: 0xEF 0xBB 0xBF.

Щоб вибрати найбільш ефективний режим для QR-коду, необхідно подивитись на символи у вхідному рядку та перевірте наступні умови:

— Якщо вхідний рядок складається лише з десяткових цифр (від 0 до 9), використовується числовий режим.

— Якщо числовий режим не застосовується, і якщо всі символи вхідного рядка можна знайти в колонці буквено-цифрової таблиці, використовується буквено-цифровий режим. Малі літери не можуть кодуватись в буквено-цифровому режимі — лише великі літери.

— Якщо є символ, якого немає в колонці буквено-цифрової таблиці, але його можна закодувати у ISO 8859-1, використовується байтовий режим. Як зазначено

вище, пристрої для зчитування QR-кодів можуть розпізнати UTF-8 у байтовому режимі.

— Якщо всі символи знаходяться в наборі символів Shift JIS, використовується режим Kanji. Натомість символи Shift JIS можуть кодуватися в UTF-8, тому можна використовувати байтовий режим для Kanji, але, як правило, ефективніше використовувати Shift JIS і використовувати режим Kanji для символів Kanji.

Загальна структура кодування QR є послідовністю 4-бітових індикаторів з довжиною корисного навантаження, що залежить від режиму індикатора. Чотирибітові індикатори використовуються для вибору режиму кодування та передачі іншої інформації. Після кожного індикатора, який вибирає режим кодування, є поле довжини, яке вказує, скільки символів закодовано в цьому режимі. Кількість бітів у полі довжини залежить від кодування та версії символу.

Буквено-цифрове кодування зберігає повідомлення більш компактно, ніж може байтовий режим, але не може зберігати малі літери і має лише обмежений вибір розділових знаків, яких достатньо для елементарних веб-адрес.

У таблиці виправлення помилок згадуються "група 1" і "група 2", а також "кількість блоків". Це означає, що кодові слова даних повинні бути розділені на дві групи, і всередині кожної групи кодові слова даних можуть бути розбиті на блоки. Кодові дані розбиваються послідовно (тобто, починаючи з кодового слова 1, потім кодового слова 2 і т.д.). Для коду 5-Q є дві групи, перша з яких повинна бути розбита на 2 блоки, кожна з яких містить 15 кодових слів даних, а друга з яких має бути розбита на 2 блоки, кожна з яких містить 16 даних кодового слова [6].

Можна використовувати кілька режимів в одному QR-коді, включаючи індикатор режиму перед кожним розділом байтів, що використовує цей режим. Специфікація QR-коду пояснює, як найбільш оптимально перемикати режими. Проте метою даної роботи є дослідження оптимального та ефективного способу аналізу двовимірного зображення, і розробка програмної системи на основі обраного методу. Застосування декількох режимів аналізу даних вимагає

додаткового часу на обробку QR-коду та використання більшого розміру пам’яті пристрою, що є неоптимальним для розробленої програмної системи.

2.3. Алгоритм коригування помилок двовимірного зображення під час обробки даних

Після того, як обрано відповідний режим кодування тексту, наступним кроком є кодування тексту. Кожен режим кодування призначений для створення найкоротшого можливого рядка бітів для символів, які використовуються в цьому режимі. Кожен режим використовує інший метод для перетворення вхідного тексту в рядок бітів. Результатом цього кроку є рядок бітів, який розділений на кодові слова даних, довжина яких складає 8 бітів.

Функція виправлення помилок QR-коду реалізована шляхом додавання коду Ріда-Соломона до вихідних даних. Можливість виправлення помилок залежить від обсягу даних, які потрібно виправити. Наприклад, якщо потрібно закодувати 100 кодових слів QR-коду, 50 з яких потрібно виправити, потрібно 100 кодових слів коду Ріда-Соломона, оскільки Код Ріда-Соломона вимагає подвоєної кількості кодових слів для виправлення. У цьому випадку загальна кількість кодових слів становить 200, 50 з яких можна виправити. Таким чином, коефіцієнт виправлення помилок для загальної кількості кодових слів становить 25%. Це відповідає виправленню помилок QR-коду Рівня Q.

Чотири рівні виправлення помилок доступні для вибору користувачами відповідно до робочого середовища (рисунок 2.4). Підвищення цього рівня покращує можливість виправлення помилок, але також збільшує обсяг даних QR-коду.

Щоб вибрати рівень виправлення помилок, слід враховувати різні фактори, такі як робоче середовище та розмір QR-коду. Рівень Q або H можна вибрати для заводського середовища, де QR-код забруднюється, тоді як рівень L можна вибрати

для чистого середовища з великим обсягом даних. Як правило, найбільш часто вибирається Рівень M (15%).

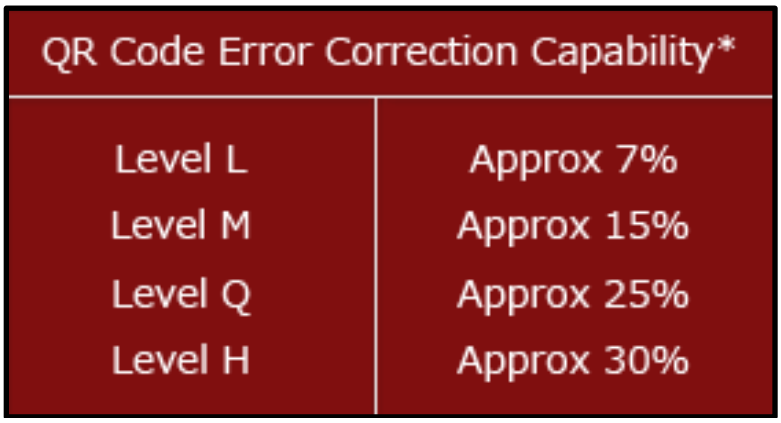

Рисунок 2.4 — Швидкість відновлення даних для загальної кількості кодових слів (кодове слово — це одиниця, яка створює область даних. Одне кодове слово QR-коду дорівнює 8 бітам)

Різні розміри QR-кодів називаються версіями. Доступно сорок версій. Найменша версія — це версія 1, розмір якої становить 21 піксель на 21 піксель. Версія 2 — 25 пікселів на 25 пікселів. Найбільша версія — версія 40, розміром 177 на 177 пікселів. Кожна версія на 4 пікселі більша за попередню версію.

Кожна версія має максимальну ємність, залежно від використовуваного режиму. Крім того, рівень виправлення помилок ще більше обмежує потужність.

QR-код найбільшої ємності — 40-L (версія 40, рівень виправлення помилок L). Це максимально можлива кількість символів, яку може містити один QR-код. Версії 40-M, 40-Q та 40-H мають меншу потужність, оскільки їм потрібно більше місця для більшої кількості виправлених кодових слів. Нижче наведена таблиця 2.1, в якій перелічено місткість 40-Q для чотирьох режимів кодування QR-коду.

Принциповою частиною роботи QR-кодів є те, що чим більше даних вкладено в них, тим більше рядків і стовпців модулів буде введено в QR-код для компенсації збільшеного навантаження даних. Зі збільшенням рівня виправлення помилок це означає, що також буде збільшуватися кількість рядків і стовпців модулів, необхідних для зберігання вихідних даних, плюс збільшення кількості

резервних кодових слів. Це відображається також на QR-коді безпосередньо — він стає більш щільним та деталізованим [7].

Таблиця 2.1. Опис ємності символів для різних режимів аналізу даних

Режим кодування	Максимальна к-сть символів, яку може містити 40-Q код у цьому режимі
Числовий	3993 символів
Буквено-цифровий	2420 символів
Байтовий	1663 символів
Kanji	1024 символів

Кожен режим кодування має чотирирозрядний індикатор режиму, який його ідентифікує. Кодовані дані повинні починатися з відповідного індикатора режиму, який вказує на режим, що використовується для бітів, які йдуть після нього. У наступній таблиці 2.2 перелічені індикатори режиму для кожного режиму кодування.

Наприклад, якщо кодується HELLO WORLD у буквено-цифровому режимі, індикатор режиму дорівнює 0010.

Таблиця 2.2. Індикатор режиму для ідентифікації кодування

Режим кодування	Індикатор режиму
Числовий	0001
Буквено-цифровий	0010
Байтовий	0100
Kanji	1000
ECI	0111

Індикатор підрахунку символів — це рядок бітів, що представляє кількість символів, які кодуються. Індикатор підрахунку символів повинен бути розміщений

після індикатора режиму. Крім того, показник підрахунку символів повинен мати певну кількість бітів, залежно від версії QR.

Потрібно підрахувати кількість символів у вихідному введеному тексті, а потім перетворити це число у двійкове. Довжина індикатора підрахунку символів залежить від режиму кодування та версії QR-коду, яка буде використовуватися. Щоб зробити двійковий рядок відповідної довжини, він додається ліворуч з 0s. Наприклад, якщо кодувати HELLO WORLD у QR-коді версії 1 у буквено-цифровому режимі, індикатор підрахунку символів повинен мати довжину 9 біт. Кількість символів HELLO WORLD дорівнює 11. У двійковій системі 11 дорівнює 1011. Цей запис переноситься ліворуч, щоб він становив 9 біт: 000001011. Потім поміщається це після індикатора режиму, щоб отримати наступний бітовий рядок: 0010 000001011.

Приклад HELLO WORLD кодується на сторінці кодування буквено-цифрового режиму. Перетворення в бітовий рядок в таблиці 2.3:

Таблиця 2.3. Приклад перетворення запису у бітовий рядок

Індикатор режиму	Індикатор підрахунку символів	Закодовані дані
0010	000001011	01100001011 01111 00011010001011100 10110111000 1001 1010100 001101

QR-сканери зчитують як кодові слова даних, так і кодові слова для виправлення помилок. Порівнюючи ці дві опції, сканер може визначити, чи правильно зчитані дані, і може виправити помилки, якщо ці дані було неправильно зчитано.

В оригінальному поданні Reed & Solomon (1960), кожне кодове слово коду Ріда-Соломона є послідовністю значень функції полінома ступеня менше *k*. Для того, щоб отримати кодове слово коду Ріда-Соломона, символи повідомлення

(кожен в межах алфавіту розміром q) обробляються як коефіцієнти полінома p ступеня менше k, над кінцевим полем F з елементами q. У свою чергу, поліном p обчислюється за $n \leq q$ різних точок поля F та послідовності значень - це відповідне кодове слово. Поширені варіанти для набору балів оцінки включають {0, 1, 2, ..., n - 1}, {0, 1, α, α2, ..., αn − 2} або для $n < q$, { 1, α, α2, ..., αn − 1}, ..., де α - примітивний елемент F [8].

Двовимірні зображення використовують корекцію помилок Ріда-Соломона над кінцевим полем (F_{256}), елементи якого кодуються як байти по 8 бітів; байти *b7b6b5b4b3b2b1b0* зі стандартним числовим значенням кодує елемент поля, де α приймається як примітивний елемент, що задовольняє F_{256}.

У коді Рід-Соломона використовується один із 37 різних поліномів F_{256} зі ступенями від 7 до 68, залежно від того, скільки байт виправлення помилок додає код. У більших QR-символах повідомлення розбивається на кілька блоків коду Ріда – Соломона. Розмір блоку вибирається таким чином, що не робиться спроб виправити більше 15 помилок на блок; це обмежує складність алгоритму декодування. Потім блоки коду перемежовуються, що робить менш вірогідним, що локалізовані пошкодження символу QR перевершать потужність будь-якого окремого блоку [9].

Завдяки виправленню помилок можна створити індивідуальні QR-коди, які все ще правильно скануються, але містять навмисні помилки, щоб зробити їх більш читабельними або привабливими для людського ока, а також включити кольори, логотипи та інші функції в QR-код блок.

2.4. Вплив колірної гами на читабельність QR-коду

Під час створення кольорового QR-коду, кольори слід ретельно підбирати. В іншому випадку мобільний телефон чи інший пристрій може не обробити двовимірне зображення. Наприклад, якщо фон темний з темним переднім планом,

пристрій, можливо, не зможе розпізнати інформаційні елементи і буде ігнорувати QR-код. Те саме відбувається, якщо і фон, і передній план світлі. Повинен бути контраст: світлий фон і темний передній план, або темний фон і світлий передній план.

У QR-коді темні біти завжди інтерпретуються як дані, тому точки завжди повинні мати темніший колір, ніж фон. Надійність QR-коду також визначається різницею контрастності між крапками та фоном, тому блідо-сірі крапки на білому тлі будуть важко зчитуватись деякими скануючим пристроями, і код може працювати неправильно.

Тому можна обирати будь-які кольори, але слід переконатись, що точки мають значно темніший колір, ніж фон, рекомендується принаймні на 70% темніше, щоб забезпечити надійне сканування. І завжди перевіряти створений QR-код, щоб переконатися, що він працює, використовуючи вибрані кольори.

Кольоровий QR-код визначається методом сегментації кольорових зображень. Згідно з теорією сегментації кольорових зображень, найпростішим та найефективнішим способом сегментації є і полягає безпосередньо відповідно до моделі кольорового простору RGB для сегментації зображень [10]. Кольорова модель RGB є моделлю восьми вершин чорного, білого, червоного, зеленого, синього, жовтого, пурпурового, темно-зеленого ─ види кольорів, які найбільш легко розпізнати. Але враховуючи обробку кольорового QR-коду з точністю зчитування обладнання, водночас, використаний колір є більш вразливим до зовнішніх факторів, таких як світлові перешкоди, які спричиняють неправильне читання двовимірного зображення.

Якщо додавати логотип або якусь картинку до двовимірного зображення, щоб створити 3D-сприйняття QR-коду, потрібно вирішити, яку частину кодування перекривати обраним логотипом (рисунок 2.5). Ключ до створення цих вражаючих дизайнерських кодів полягає у тому, щоб скористатися тим фактом, що до 30% даних QR-коду можуть бути відсутніми або пошкодженими та все одно скануватися. QR-коди можуть бути сформовані з вбудованим коефіцієнтом

виправлення помилок 0%, 10%, 20% або 30%. Вбудований коефіцієнт виправлення помилок 30% додає більше “шуму” (додаткових полів) всередині коду, але ці додаткові поля в коді можуть потім бути видалені, щоб створити місце для логотипу чи інших цікавих зображень.

Мінімальний розмір спеціального QR-коду повинен бути не менше 2х2 см для сканування без будь-яких проблем. “Тиха зона” (quite zone) — це, по суті, простір навколо кутів QR-коду (рисунок 2.4). Належний інтервал QR-коду може допомогти сканерам розрізнити оточення та код. Код стає нечитабельним, якщо простір між тихою зоною та модулем надмірно зменшений. Мінімальним критерієм для тихої зони є те, що вона принаймні в чотири рази повинна перевищувати ширину модулів QR-коду. Не слід переповнювати тиху зону, вводячи в код багато даних, зменшуючи тим самим розмір пікселів.

Рисунок 2.5 — Приклад створеного двовимірного зображення з використанням кольорів та логотипу

Фон двовимірного зображення можна змінити, щоб користувачі розуміли, яку інформацію містить QR-код. Фон може містити зображення або просто кольори. Індивідуальний QR-код зберігає свою читабельність, лише якщо редагується не більше 30% коду. Не варто брати низький рівень коригування

помилок, оскільки у випадку використання індивідуальних двовимірних зображень, втрачається можливість їх сканування [11].

Індивідуальні QR-коди з логотипом та відповідною кольоровою палітрою можуть допомогти клієнтам ефективно розпізнати будь-яку компанію без особливих труднощів. Синхронізація кольорів та логотипів бренду покращує впізнаваність бренду на 80%.

2.5. Огляд програмних застосунків для сканування двовимірних зображень

Використання QR-кодів не обмежується ніякими ліцензіями, вони також описані та опубліковані в якості стандартів ISO. Найбільше визнання QR-код отримав серед користувачів мобільного зв'язку: встановивши програму-дешифратор, абонент може після сканування коду заносити в свій мобільний пристрій текстову інформацію, додавати контакти в адресну книгу, переходити по web-посиланнях, відправляти SMS-повідомлення та ін. За допомогою QR-кодів організовують конкурси та рольові ігри. Є багато безкоштовних додатків, які дозволяють закодувати повідомлення.

1. The QR Stuff. Сканер QR-коду для щоденного використання. Коли існує QR-код — чи це платіжні шлюзи, туристична інформація, відео чи будь-який інший тип QR-коду, використовується камера пристрою для зчитування QR-коду. Потрібно навести камеру на QR-код, який потрібно відсканувати, і отримати дані. При першому скануванні QR-коду необхідно надати сканеру дозвіл на доступ до камери на пристрої, але після цього не потрібно кожного разу отримувати запит. Сканувати QR-код у будь-який час і в будь-якому місці та створювати коди за допомогою QR-генератора.

Особливості сканування QR-коду:

— сканування всіх типів QR-кодів;

— детальні картки даних QR-коду;

— історія сканування;

— швидке пакетне сканування QR-кодів.

Генератор QR-коду:

— можливість створення необмеженої кількості QR-кодів;

— експорт та обмін QR-кодами.

Особливості сканера штрих-коду:

— Сканування всіх типів штрих-кодів, такі як Code 39, Code 128 або UPC;

— Зберігання, експорт та обмін усіма штрих-кодами (рисунок 2.6).

Проте даний застосунок не забезпечує коректне сканування двовимірного зображення, якщо воно є пошкодженим. Питання безпеки даних та безпеки сканування QR-кодів не є реалізованими.

Єдиним контекстом, в якому загальні QR-коди можуть містити виконувані дані, є тип даних URL [12]. Ці URL-адреси можуть розміщувати код JavaScript, який може бути використаний для використання вразливостей у програмах хост-системи, таких як зчитувач, веб-браузер або переглядач зображень, оскільки зчитувач зазвичай надсилає дані до програми, пов’язаної з типом даних використовується QR-кодом.

Динамічні QR-коди становлять особливий ризик. Дані, що зберігаються на них, можна змінити після їх створення або представити різні дані на різних типах пристроїв. Зростання проблеми безпеки QR-кодів також збігається із зростанням криптовалюти. QR-коди вводять дані, а Bitcoin — це дані, тому використання QR-кодів для викрадення даних Bitcoin було неминучим.

У разі відсутності використання програмного забезпечення зловмисні QR-коди в поєднанні з дозвільним зчитувачем все одно можуть загрожувати даним пристрою та конфіденційності користувачів. Ці пошкоджені двовимірні зображення легко створюються, і їх можна наносити на безпечні QR-коди. На смартфоні налаштування сканера можуть дозволяти використання камери, повний доступ до Інтернету, читання / запис контактних даних, GPS, читання історії браузера, читання / запис локального сховища та глобальні системні зміни [13].

Тому питання безпеки даних та інформації в двовимірних зображеннях є досить актуальною та важливою.

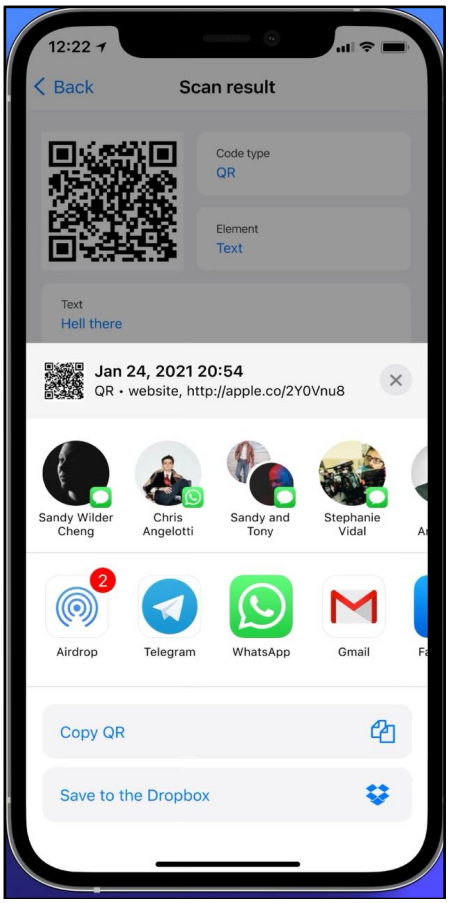

Рисунок 2.6 — Функція експорту та копіювання двовимірного зображення у застосунку QR Stuff

Ризики включають посилання на небезпечні веб-сайти з використанням браузера, увімкнення мікрофона / камери / GPS, а потім передачу цих каналів на віддалений сервер, аналіз конфіденційних даних (паролів, файлів, контактів, транзакцій), та надсилання електронних листів, повідомлення SMS / IM або DDOS-пакети, як частину ботнету (англ. botnet від robot і network), пошкоджуючи налаштування конфіденційності, викрадаючи особисті дані, і навіть містити найбільш шкідливі скрипти (англ. script) або віруси. Ці дії можуть відбуватися у фоновому режимі, поки користувач бачить лише сканер, який відкриває, здавалося б, нешкідливу веб-сторінку. Реальним прикладом є зловмисний QR-код, який

змусив телефони, які його сканували, надсилати преміум-тексти за плату в 6 доларів США за кожен [14].

2. QR/Barcode Scanner. Сканер QR / штрих-коду діє як камера в додатку, яка сканує та розпізнає QR-коди та штрих-коди, зареєстровані у вашому сервісі Magento. Після завершення сканування клієнти будуть перенаправлені на відповідні сторінки товару, щоб продовжити свою покупку.

Особливості застосунку:

— доступний у використанні — відкрити мобільний пристрій для того, щоб встановити код та просканувати;

— простота установки та налаштування;

— підтримка iOS та Android;

— можливість створення штрих-кодів та QR-кодів для необмеженої кількості продуктів;

— налаштування штрих-кодів та QR-кодів для всіх типів продуктів Magento;

— можливість для користувачів швидко сканувати штрих-коди за допомогою програми та перенаправляти їх на сторінку деталей продукту.

Супутні функції:

— голосовий пошук;

— список бажань.

Функціонал програмного продукту QR/Barcode Scanner дозволяє обрати та вивантажити двовимірне зображення з галереї телефону одразу в сканер застосунку, що є зручною функцією. Однак питання ефективності сканування двовимірних зображень також не є вирішеною.

3. Pageloot QR код Сканер.

— Історія QR-сканування. Відслідковування усіх сканувань — модуль Pageloot QR-код сканер Інструмент для отримання збереженого списку сканувань. Можливість безкоштовної реєстрації для включення функції “Історія QR-сканувань” (рисунок 2.7).

Збереження закладки зчитувача Pageloot QR код як для iPhone, так і для Android. Зручна можливість додати закладку на головний екран для того, щоб функціонал цього мобільного застосунку працював як основний для мобільного пристрою. Також доступ до безкоштовної версії застосунку онлайн. Можливість безкоштовно придумати кілька цільових конструкцій.

Рисунок 2.7 — Модуль історії QR-сканувань мобільного застосунку Pageloot

— Безпека сканування QR-кодів. Безпечне сканування — блокування шкідливих посилань і гарантія того, що відсканові дані безпечні для перегляду. QR-сканер показує безпечний попередній перегляд, який можна перевірити.

— Теги та папки. Ведення структури користувацьких QR-кодів — можливість групування їх по папках або організації за тегами. Пошук необхідних кодів за допомогою пошуку і фільтрів.

Можливість зберегти закладку зчитувача QR-коду Pageloot та генератора QR-коду для iPhone або Android.

Можливість додати закладку на головний екран, щоб ця сторінка працювала як власний пристрій для зчитування QR-коду сканером.

Можливість отримати доступ до безкоштовного зчитувача QR-кодів в Інтернеті. Дана система розроблена з метою збереження історії сканування QR-коду.

QR-коди можуть містити шкідливі посилання. Функціонал використання інструментів сканування QR Pageloot для перевірки сканувань. Створена функція SafeScan допоможе переглянути попередній вміст посилань QR-коду.

Програмний застосунок вирішує важливі питання безпеки сканування даних двовимірних зображень, що є актуальною задачею. Проте також не розглядається тема коригування втрати даних.

Одним з варіантів вирішення проблеми безпеки даних двовимірних зображень може бути SQR код. Код безпечного швидкого реагування (SQR) — це QR-код, який містить сегмент "приватних даних" після термінатора замість зазначених байт формату "ec 11" [15]. Цей приватний сегмент даних потрібно розшифрувати за допомогою ключа шифрування. Це може бути використано для зберігання приватної інформації та управління внутрішньою інформацією компанії.

Коди SQR були розроблені Фондом FORUS для забезпечення безпечних транзакцій та опубліковані під ліцензією Creative Commons. Рішення SQR гарантує цілісність вихідних даних, а також дійсність сторони-джерела. Рядок платіжних інструкцій складається з електронних даних інструкцій зі сканованого QR-коду, доданих до криптографічного хешу SHA-2. Потім дайджест повідомлення можна зашифрувати, використовуючи приватний ключ відправника, який потім створює цифровий підпис повідомлення. Цей підпис підтверджує цілісність даних та надійність відправника. Це підтверджує особу відправника, і те, що ці дані не були підроблені під час передачі [16]. Завдяки вбудовуванню URL-адреси та усіх змінних, необхідних для здійснення електронної комерції, оплати рахунків та однорангових платежів, у поєднанні з цифровим сертифікатом виключається можливість підміни, фальсифікації та атаки посередині цього процесу сканування двовимірного зображення.

3. ЗАСОБИ РОЗРОБКИ

3.1. Технологія розробки мобільних застосунків Flutter та мова програмування Dart

Для вирішення і реалізації поставлених задач обраних стек технологій для мобільної розробки, що надає всі необхідні інструменти розробки запроектованого програмного забезпечення. Особливості реалізації та діаграму розгортання представлено на рисунку 3.1.

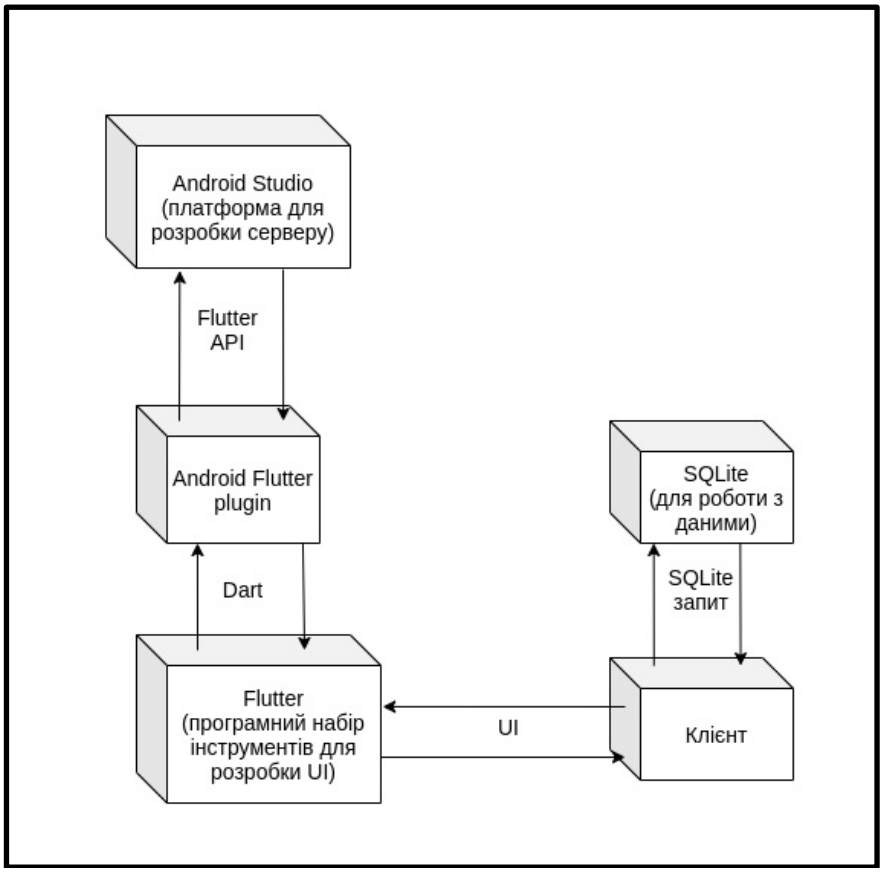

Рисунок 3.1 — Діаграма розгортання

Об'єкт FlutterEngine представляє єдиний контекст виконання Flutter. Це означає, що FlutterEngine контролює ізоляцію Dart (створений код на Dart, який

починається з точки входу, як main). Це також означає, що FlutterEngine встановлює ряд стандартних каналів платформи, які потрібні всім програмам Flutter; він включає підтримку переглядів платформи, він знає, як намалювати текстуру за допомогою інтерфейсу Flutter, і він обробляє всі інші фундаментальні вимоги для виконання єдиного додатка Flutter/Dart. Крім того, програма Android може містити кілька FlutterEngines одночасно (рисунок 3.2).

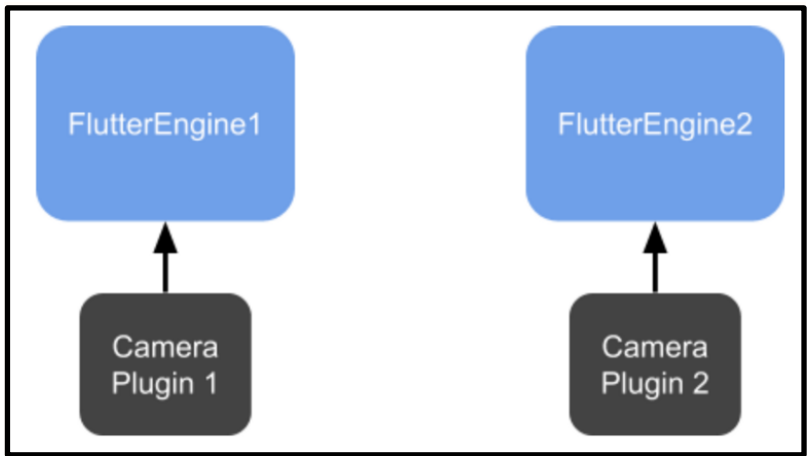

Рисунок 3.2 — Архітектура ядра плагіну Flutter

Плагін підтримує розробку мобільних нативних застосунків. Flutter надає розробникам простий і продуктивний спосіб створення та розгортання міжплатформових високопродуктивних мобільних програм для Android та iOS. Встановлення цього плагіна (англ. plug-in — підключати) також завантажує плагін Dart, необхідний для мобільної розробки.

Основна концепція додавання плагіна до програми Flutter означає застосування цього плагіна до одного FlutterEngine. Наприклад, якщо програма Flutter вимагає доступу до камери, ця можливість досягається реєстрацією плагіна камери в конкретному екземплярі FlutterEngine. Ця реєстрація виконується для користувача автоматично за допомогою GeneratedPluginRegistrant, але важливо розуміти, що кожен FlutterEngine підтримує власний набір плагінів Flutter [17].

Детальний огляд Flutter показує, що це більш вичерпна альтернатива інструментам для розробки мобільних додатків React Native та Xamarin. Подібно до інших рішень, Flutter допомагає створювати додатки для Android та iOS з єдиної

кодової бази. Його відрізняє від решти те, що він не працює як обгортка поверх рідних компонентів інтерфейсу. Натомість він малює внутрішній інтерфейс, що створюється з нуля.

Конструктор інтерфейсу Flutter має унікальний стек технологій, що відрізняє його від інших крос-платформних рішень для розробки та забезпечує його найбільш ефективні функції.

Плагін мови програмування Dart надає гнучкі інструменти мобільної розробки в поєднання з Flutter. З важливих особливостей Dart можна виділити наступне:

— розумна допомога в кодуванні для Dart, що включає заповнення коду, форматування, навігацію, наміри, рефакторинг;

— інтеграція з pub та Dart Analysis Server

— IDE виявляє проблеми з кодом на льоту і пропонує способи їх автоматичного виправлення;

— запуск і налагодження командного рядку та веб-програми Dart прямо в IDE за допомогою вбудованого налагоджувача;

— запуск та налагодження тестів;

— створення нових проектів Dart на екрані запуску IDE.

Компілятор JIT та AOT. Flutter використовує два різні методи для компіляції програм під час розробки додатків та для розгортання: Just-In-Time (JIT) та компіляція Ahead-of-Time (AOT). Метод JIT забезпечує функцію гарячого перезавантаження, що прискорює процес розробки, тоді як AOT забезпечує швидкий час запуску та стабільну роботу додатків Flutter після випуску.

Двигун Flutter побудований на C ++. Він забезпечує швидке та ефективне виконання та компілятор Dart, обробляє базові API та плагіни, взаємодіє з графічною бібліотекою для візуалізації коду та обробляє файлові та мережевий ввід-вивід.

Графічна бібліотека Skia. На відміну від інструментів конкурентів, Flutter самостійно малює свої віджети (англ. widget). Він використовує бібліотеку Skia для

візуалізації інтерфейсу користувача замість того, щоб покладатися на компоненти певної платформи, що постачаються з кожним пристроєм. Це зменшує зв'язок між програмами Flutter та мобільними платформами та мінімізує шанси на розрив інтерфейсу користувача на будь-якій конкретній версії ОС або пристрої.

Швидке (гаряче) перезавантаження для швидкої розробки. Гаряче перезавантаження — це визначна функція Flutter, яка дозволяє розробникам побачити зміни, які вони застосовують до коду, менш ніж за секунду на емуляторах, симуляторах та апаратному забезпеченні. Більше того, гаряче перезавантаження Flutter є дуже важливим, тому що розробники можуть швидко виконувати ітерацію на глибоко вкладеному в програму екрані, не починаючи з головного екрана після кожного перезавантаження. Таким чином, робочий процес програмування безперервно підвищує продуктивність команди розробки та дозволяє експериментувати без затримок.

3.2. Ефективний спосіб збереження даних для мобільного застосунку

Якщо розробляти програму, яка повинна зберігати і запитувати великі обсяги даних на локальному пристрої, розглядається можливість використання бази даних замість локального файлу або сховища ключ-значення. Загалом, бази даних забезпечують більш швидкі вставки, оновлення та запити порівняно з іншими локальними рішеннями збереження.

Flutter надає спеціальний пакет, cloud_firestore для програмування з Cloud Firestore. Для того, щоб підключити цю платформу необхідно виконати наступні кроки. Створити новий додаток Flutter в студії Android, product_firebase_app. Замінити код запуску за замовчуванням (main.dart) на код product_rest_app. Скопіювати файл Product.dart із product_rest_app у папку lib [18]. Скопіювати папку assets з product_rest_app в product_firebase_app і додайте assets у файл pubspec.yaml. Налаштувати пакет cloud_firestore у файлі pubspec.yaml, залежності: cloud_firestore:

^ 0.9.13 + 1. Використати останню версію пакета cloud_firestore. Студія Android попередить, що pubspec.yaml оновлено. Пакет Cloud Firestore. Натиснути опцію Отримати залежності. Студія Android отримає пакет з Інтернету та правильно налаштує його для програми.

Плагін SQLite — це найбільш популярний спосіб для зберігання даних на мобільних пристроях. Для створення програмного забезпечення використано пакет sqflite для використання SQLite бази даних. Sqflite — одна з найбільш часто використовуваних і актуальних бібліотек для підключення SQLite бази даних в Flutter.

Плагін SQLite для Flutter підтримує iOS, Android та MacOS:

— підтримка транзакцій та пакетів;

— автоматичне управління версіями під час запуску;

— модулі для вставки / запиту / оновлення / видалення запитів;

— операції БД, виконані у фоновому потоці на iOS та Android.

Плагін SQLite є універсальним та портативним, що стосується програмного забезпечення. Як рушій баз даних, SQLite забезпечує широкі крос-платформні функціональні можливості та портативність. Зокрема, можливість копіювати та передавати дані між 32 та 64-розрядними ОС, а також між великою та малої архітектурою. По суті, SQLite може працювати та компілюватися в різних системах, включаючи Windows, Linux та Mac OS. Більше того, оскільки база даних SQLite зберігається лише в одному файлі, а не в колекції окремих файлів, вона значно гнучкіша з точки зору її переносимості. Це ще більше розширює діапазон обчислювальних систем, які SQLite може охоплювати.

Налаштування та адміністрування БД є значно простими. З самого початку, SQLite має нульову конфігурацію і має мінімальні вимоги. БД не включає файли конфігурації та працює без серверів. Все ці характеристики прирівнюються до надійної роботи, включаючи високий рівень надійності та загальної стабільності.

3.3. Опис об’єктно-орієнтованої частини системи

У Dart кожен об'єкт є екземпляром класу, і всі класи походять від Object. Dart також має успадкування на основі міксину, і це перекриває відсутність кількох успадкувань. Цей тип успадкування дозволяє повторне використання кількох тіл класу та існування рівно одного суперкласу.

Можна створити об’єкт визначеного класу, використовуючи один з його конструкторів. Імена конструктора можуть бути як іменем класу ClassName або ClassName.identifier. Конструктори можуть мати аргументи для надання необхідних значень для ініціалізації нових об'єктів і бувають декількох типів:

— за замовчуванням (default) — коли не оголошено конструктор, а надається конструктор за замовчуванням, який не має аргументів і викликає конструктор no-argument у суперкласі; важливо, якщо оголошено конструктор, конструктора за замовчуванням не буде, а якщо розширено клас, його конструктор не успадковується;

— іменований (named) — використовується, коли потрібно реалізувати кілька конструкторів для класу або надати додаткову ясність;

— перенаправлений (redirecting) — коли необхідно перенаправити певний конструктор в той самий клас; його тіло порожнє, а виклик конструктора з’являється після «:» ;

— постійний (constant) — використовується, коли потрібні предмети, які ніколи не змінюються; при виклику таких конструкторів необхідно використовувати ключове слово const, інакше буде створено констант;

— фабричний (factory) — при реалізації конструктора, який не завжди створює новий екземпляр свого класу; фабричний конструктор може повернути кешований екземпляр, зробити об’єднання об’єктів або може повернути екземпляр підтипу; фабричні конструктори не мають до цього доступу.

Dart-класи також можуть поводитися як функції (їх можна викликати, брати аргументи і щось повертати). Щоб реалізувати це, потрібно визначити метод call() у середині класу.

Генератори використовуються, коли потрібно обробляти послідовність значень. Dart підтримує два типи генераторних функцій:

— синхронний генератор, який повертає об'єкт, що підлягає ітерації. Для реалізації синхронного генератора позначено тіло функції як sync* і використано оператор yield для повернення значень.

— і асинхронний генератор, який повертає об’єкт Stream. Для реалізації асинхронного генератора позначено тіло функції як async* і використано оператор yield для повернення значень. Якщо використано рекурсивні виклики, поліпшення продуктивності можна досягти за допомогою yield*.

Усі неініціалізовані змінні за замовчуванням мають значення null. Крім того, всі змінні, які не є остаточними, генеруватимуть неявний геттер і сеттер. Останні, не генерують сетера. За замовчуванням змінні є змінними екземпляра. Якщо вони ініціалізуються при оголошенні (замість конструктора або методу), їх значення встановлюється під час створення екземпляра, який знаходиться перед виконанням конструктора та його списку ініціалізаторів. Для створення змінної класу використано статичне ключове слово. Вони корисні для загальнокласового стану та констант. Вони не ініціалізуються, доки не будуть використані.

Методи — це функції, що забезпечують поведінку об’єкта. Як і у випадку зі змінними, тут також є два варіанти: екземпляр та методи класу. Методи екземпляра на об'єктах можуть отримувати доступ до змінних екземпляра і this.

Сканер QR-коду, який працює як на iOS, так і на Android, вбудовуються в систему разом з Flutter. Інтеграція з Flutter є плавною, набагато кращою, ніж перехід до власного Activity або ViewController для виконання сканування.

Коли розпізнається QR-код, ідентифікований текст буде встановлено в “результат” типу штрих-код, який містить вихідний текст як властивість “код” типу

“Рядок” та тип сканованого коду як “формат” властивості, який є визначеним у бібліотеці модуля.

За замовчуванням встановлена передня камера мобільного телефону, спалах вимкнено (цю функцію можна активувати кнопкою). Можна призупинити потік сканування двовимірного зображення, а також відновити його за допомогою кнопок на екрані сканеру. Необхідне програмне забезпечення телефону принаймні SDK 21 (Android 5.0) та принаймні iOS 8.

Основні переваги створеної системи:

— побудовано на QR-Dart;

— автоматичне визначення версії / типу QR-коду або введення вручну;

— підтримує QR-коду версії 1 - 40;

— виправлення помилок / кольору;

— настроюваний розмір виводу, відступу, кольору фону і переднього плану;

— підтримка накладених зображень;

— не потрібне підключення до Інтернету.

Останньою задачею було описати засоби та інструменти реалізації архітектури програмної системи, а саме: мову програмування Dart, плагін Flutter, а також SQL. SQLite це найбільш популярний спосіб для зберігання даних на мобільних пристроях. Для створення програмного забезпечення використано пакет sqflite для використання SQLite бази даних. Для роботи обрано середовище розробки Android Studio. Також використано СУБД SQLite та бібліотеки плагінів Dart та Flutter. Проведено ряд тестів над програмою,

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТА АЛГОРИТМУ ВИПРАВЛЕННЯ ПОШКОДЖЕНЬ

4.1. Опис діаграми прецедентів розробленої системи

Розроблене програмне забезпечення націлене на створений сканер QR-кодів, який має високий рівень коригування пошкоджень зображення та втрати даних. Тому основним функціоналом мобільного застосунку є дешифратор двовимірних зображень. Цей модуль створено для мобільного застосунку, який має сферу застосування у бізнесі — електронні візитівки. Використання QR-коду на візитній картці має вагоме значення, тому що потенційні клієнти або роботодавці можуть додавати вашу контактну інформацію на свої мобільні пристрої без клопоту вручну вводити їх. Все, що їм потрібно зробити, це сканувати та зберегти.

Для створення такої електронної картки необхідно надати свої контактні дані: ім’я, прізвище, компанія, електронна адреса, опис, телефон (рисунок 4.1). Інформація зберігаються в БД, а далі перед тим як відтворити на інтерфейсі користувача разом з QR-кодом, створюється відповідно двовимірне зображення з поточними даними. Двовимірне зображення застосовує двійкове кодування інформації: чорні квадрати кодуються одиницями, білі ─ нулями. Крім того, для виявлення та виправлення помилок декодування контрольні суми перевіряються за допомогою XOR операції (додавання бітів коду за модулем два до спеціальної восьмирозрядної двійкової маски, наприклад 10101010). Завдяки виправленню помилок, можна застосувати малюнок до коду, зробити його кольоровим і різнокольоровим ─ він також залишиться читабельним.

Система містить також альтернативний потік, в залежності від того чи користувач зареєстрований чи ні. При реєстрації є додатковий функціонал мобільного застосунку, який відображає профіль користувача, його особисті дані,

а також можливість їх змінити. Введення нового паролю в розділі налаштувань реалізовано відповідним чином.

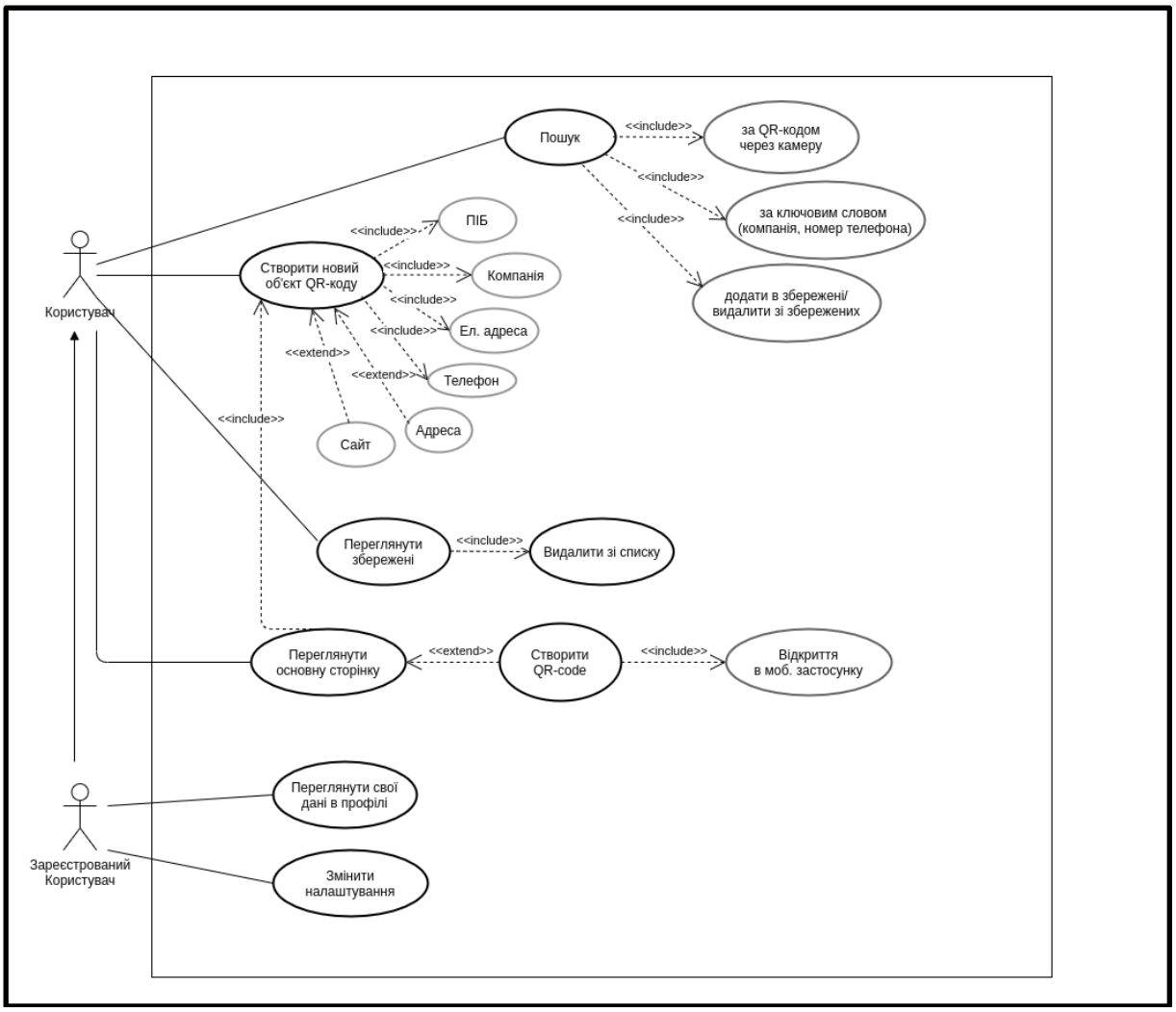

Рисунок 4.1 — Діаграма прецедентів розробленої системи

Прецедент реалізовано як набір основних сценаріїв. Але не завжди та не у кожному сценарії відповідна ціль є досяжною. На прикладі створеного програмного забезпечення, це обмежений функціонал для незареєстрованого користувача, який не має можливості зберігати власну інформацію та електронні візитки, та редагувати їх дані. Проте саме альтернативний потік приносить найбільшу практичну користь, коли очікуваний та цільовий результат не досягнуто.

4.2. Опис діаграми користувацьких потоків

Під час сканування двовимірного зображення безпосередньо визначається тип коду: це є QR-код чи штрих-код. При позитивному розвитку сценарію дані з двовимірного зображення оброблено і отримано результат сканування. В іншому випадку провести зчитування неможливо через пошкодження QR-коду і втрату необхідних даних (рисунок 4.2).

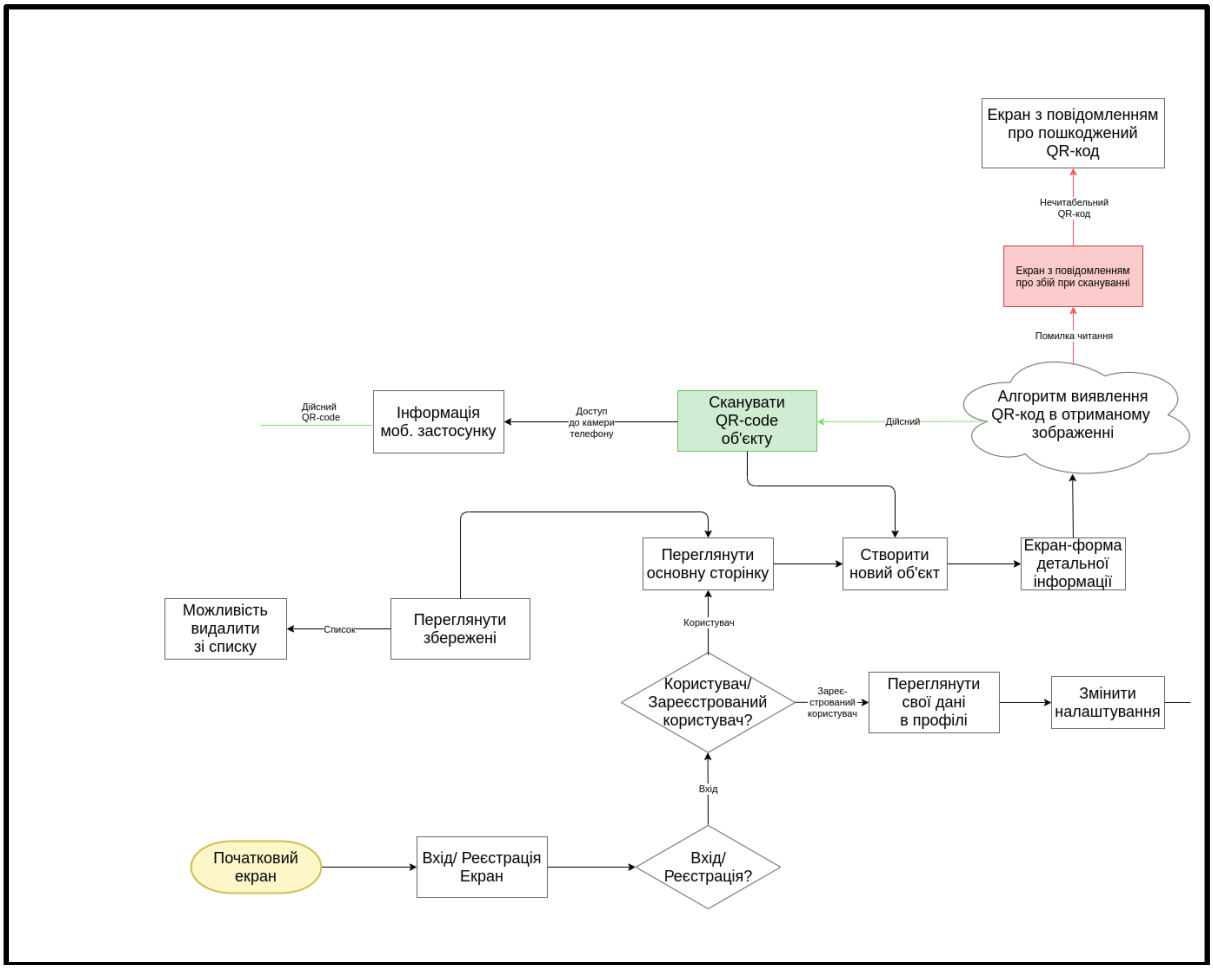

Рисунок 4.2 — Діаграма користувацьких потоків (user flow)

Можливість виправлення помилок залежить від обсягу даних, які потрібно виправити. Чотири рівні виправлення помилок доступні для вибору користувачами відповідно до робочого середовища. Підвищення цього рівня покращує можливість виправлення помилок, але також збільшує обсяг даних QR-коду. В створеній системі обрано високий рівень виправлення помилок Ріда-Соломона Q.

Інформація про формат двовимірного зображення записує дві речі: рівень виправлення помилок та шаблон маски, що використовується для символу. Маскування використовується для розбиття шаблонів в області даних, які можуть бути нерозпізнані сканером, наприклад великі порожні області або незрозумілі функції, схожі на позначки локатора. Шаблони масок визначаються на сітці, яка повторюється за необхідності, щоб охопити весь символ. Модулі, що відповідають темним областям маски, перевернуті. Інформація про формат захищена від помилок, а дві повні копії містяться в кожному символі QR-коду. Набір даних повідомлень розміщується справа наліво зигзагоподібно. У більших символах це ускладнюється наявністю шаблонів вирівнювання та використанням безлічі блоків виправлення помилок, які чергуються.

4.3. Опис діаграми класів програмної системи

Написання коду і реалізація моделі системи найкраще зображена на діаграмі класів (рисунок 4.3). Додавання нового об’єкту електронної візитівки разом з двовимірним зображенням передбачає етапи заповнення даними користувача та створення на основі цих даних QR-коду. Після цього створена візитка відображається на головному екрані застосунку, а введені особисті дані записано в БД, де також є записи про кожний створений QR-код.

Зареєстрований користувач має додатковий функціонал — зберігання обраних візитних карток до списку улюблених, та запис особистих даних в профілі. Список збережених містить картки, а саме важлива інформація — назва компанії, телефон, електронна адреса та телефон. Зміна паролю передбачає введення користувачем свій старий пароль і заповнення поля з новим. Інші дані профілю користувач також має можливість змінити в налаштуваннях: ці поля показують останні створені значення для конкретного поля, та можливість їх зміни, і тим самим зміни записів в БД.

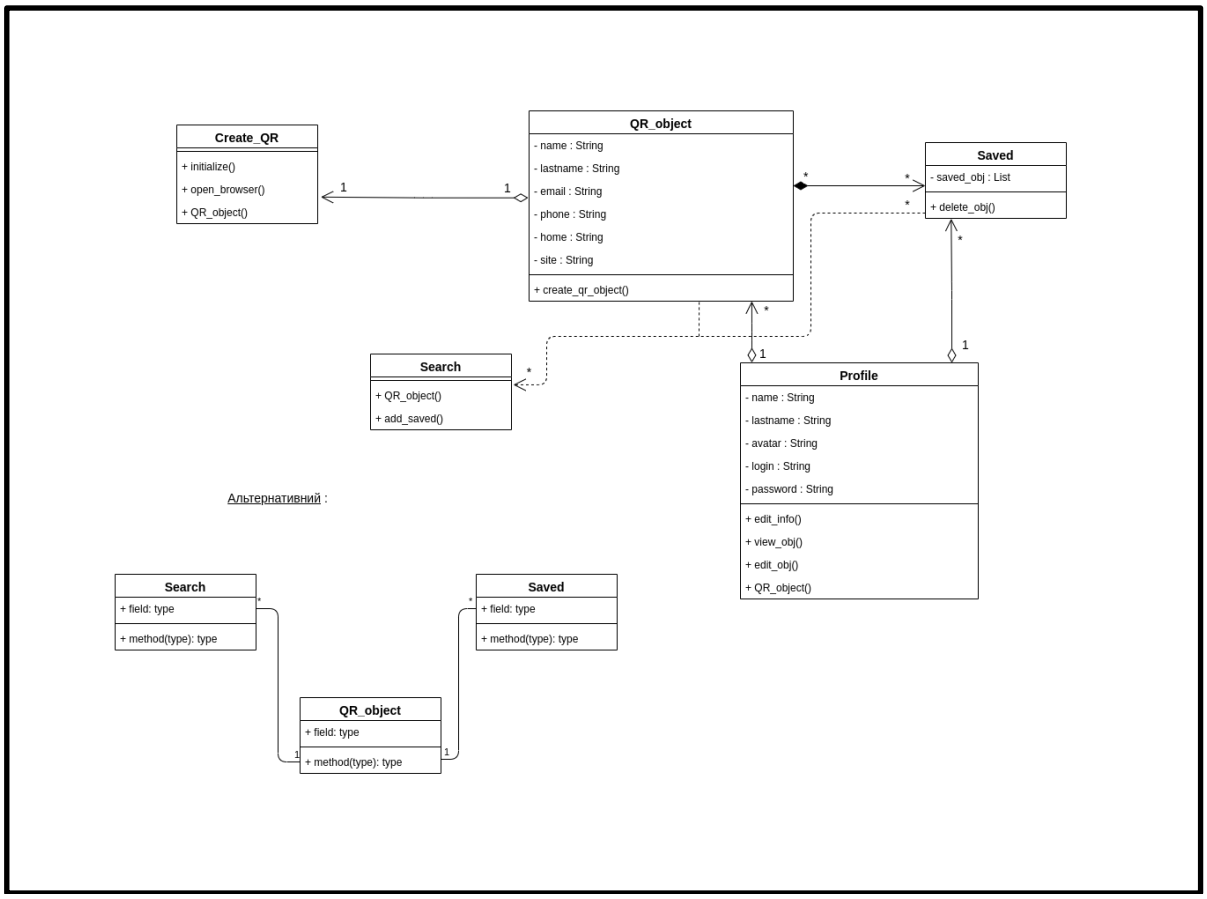

Рисунок 4.3 — Діаграма класів програмного застосунку

Пошук візитної картки за допомогою фільтрів за полями імені, номеру телефону, назви компанії, електронної адреси. Дані про існуючі електронні картки зберігаються з БД і за допомогою SQL-запитів дістаються з неї відповідно до фільтрів.

Дарт не має ключового слова interface. Натомість усі класи неявно визначають інтерфейс. Тому є можливість реалізувати будь-який клас, а також створити абстрактний клас, який буде розширений (або реалізований) конкретним класом. Абстрактні класи можуть містити абстрактні методи (з порожніми тілами). Інтерфейс — це план або шаблон класу. Цей план містить визначення змінних екземпляра та методи екземпляра, які клас повинен реалізувати. Це корисно для забезпечення послідовної структури класів. Dart неявно визначає клас як інтерфейс, отже, називається неявним інтерфейсом. Для того, щоб реалізувати інтерфейс, використовується ключове слово implements.

4.4. Архітектура та модель бази даних

Для зберігання даних нам знадобляться моделі БД (рисунок 4.4). Простий клас моделі даних дозволяє застосовувати необхідні методи для перетворення прийнятного для SQLite формату даних в об'єкт, який може бути використаний в застосунку. Абстрактний клас Model буде служити базовим класом для моделей даних. Цей файл знаходиться в lib/models/model.dart, і є обов’язковим для ініціалізації усіх моделей [19].

Клас Модель дуже простий і зручний для визначення властивостей/методів (таких як ідентифікатор таблиці), які можна очікувати від моделей даних. Це дозволяє створити одну або кілька конкретних моделей даних, що відповідають базовому шаблону проектування та наслідуються від нього.

Всередині метода, який створює екземпляр бази даних SQLite, вноситься інформація про створену БД та її ім’я. Якщо база даних з таким іменем ще не існує, автоматично викликається інший метод, який створює цю БД.

Далі створено таблицю для зберігання інформації про електронні візитки. Для цього створюється таблиця з назвою Card, яка визначає дані, які можна зберігати. Кожна картка містить ідентифікатор, прізвище, ім’я, назву компанії, зображення, телефон, електронну адресу і опис. Отже, вони представлені у вигляді вісьмох стовпців у таблиці карток.

Для визначення поведінки об'єкта в класі застосовуються методи. Наприклад, клас Profile, який представляє інформацію користувача в профілі, має такі методи, як коригування власних даних в профілі, перегляд створеного об’єкту електронної візитки, зміна даних в цій картці та додавання до неї QR-коду. Також варто врахувати, що деякі поля у відповідних методах не дозволяють значення null, тобто не можуть бути пустими. Тому є необхідність представлення початкових значень цих полів, або використання nullable-типів, що дозволяє не створювати початкові значення для записів [20].

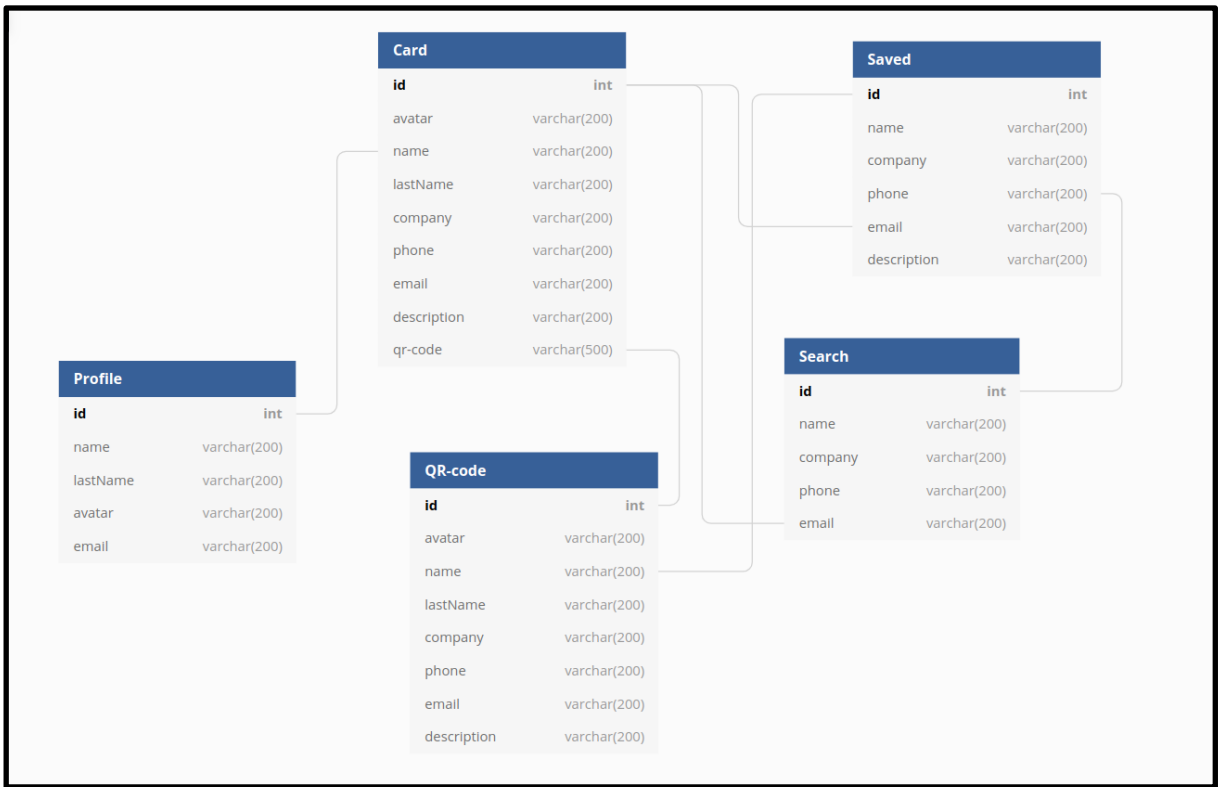

Рисунок 4.4 — Модель бази даних розробленої системи

Основні три типи цих записів, які використані для створення таблиці:

— Ідентифікатор — це Dart int і зберігається як INTEGER тип SQLite даних. Також хорошою практикою є використання ідентифікатора як первинного ключа для таблиці для покращення часу запитів та їх оновлення в БД [21].

— Компанія — це також Dart і зберігається як символьний тип даних.

— Опис — це рядок який зберігається як TEXT тип даних SQLite.

4.5. Опис етапів розгортання створеної програмної системи

Операції з базою даних, CRUD операції, передбачають наступні кроки взаємодії основних модулів системи (рисунок 4.5):

— додавання залежності;

— визначення моделі даних Card;

— відкриття бази даних;

— створення нових об’єктів та їх записів у відповідну таблицю;

— отримання списку карток;

— оновлення записів в БД;

— видалення об’єктів карток з бази даних.

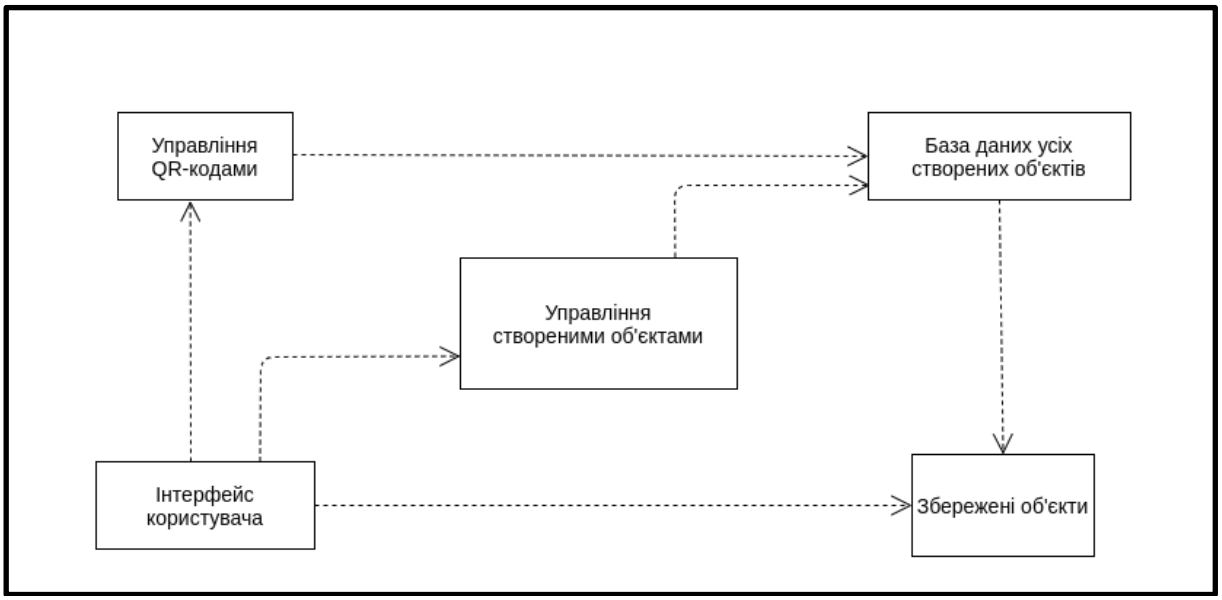

Рисунок 4.5 — Діаграма розгортання програмного застосунку

У папці lib створюється інша папка, яка називається services, і всередині неї створено клас, який називається DatabaseHandler. Цей клас подбає про всі операції щодо бази даних SQLite. Перш ніж ініціалізувати базу даних, потрібно вказати розташування файлу, який буде створений і який буде містити базу даних. Для цього потрібно додати ще один плагін, який називається path, тому потрібно перейти до pubspec.yaml і додати dependencies. Тепер всередині класу DatabaseHandler можна ініціалізувати базу даних [22].

Отже, метод getDatabasePath () знаходиться всередині пакету sqflite, і він отримує розташування бази даних за замовчуванням. Метод openDatabase () також знаходиться всередині пакету sqflite, і він приймає обов'язковий рядок як аргумент, який буде шляхом до бази даних.

Використано метод join (), який знаходиться всередині шляху пакету, він приєднує даний шлях до єдиного шляху, тому, наприклад, отримуємо datapath / example.db. Зворотний виклик onCreate () викликано після створення бази даних, і він виконає відповідний запит sql, який створить користувачів таблиці.

Всередині класу DatabaseHandler створено інший метод для додавання користувачів до бази даних. Метод insertUser () бере список користувачів, потім проганяється цикл всередині колекції і додається кожний користувач до таблиці користувачів. Метод insert () приймає такі параметри String table, значення <String, Object?>, і саме тому метод toMap () створено у класі моделі.

Для отримання даних створено інший метод у класі DatabaseHandler. Отже, тут використано метод query () і надано йому рядок користувачів, який є іменем таблиці. Отже, це вибирає всі стовпці з таблиці користувачів. Тоді, оскільки queryResult повертає List, використано метод map () для перетворення List <Map <String, Object? >> в List <User>.

Для видалення даних створено відповідний метод. За допомогою методу delete () передано ім'я таблиці, а потім вказано відповідно до якого стовпця потрібно видалити рядок у таблиці бази даних.

У файлі main.dart видалено весь код, пов’язаний із програмою лічильника Flutter, і всередині _MyHomePageState додано відповідний функціонал. Отже, створено екземпляр класу DatabaseHandler (), а потім викликано initalizeDb () для створення бази даних, яка буде містити таблицю користувачів. Коли Future завершено, викликаємо addUsers (). Далі створено декілька тестових користувачів і додано їх до списку, а потім викликано метод insertUser (). Після цього setState (() {}); буде викликано, що відновить дерево віджетів. Усередині методу build () використано віджет FutureBuilder для виклику методу retrieveUsers ().

Ключове слово factory функціонує так, щоб не створювати нових об'єктів, коли викликається Іменований конструктор. Використано Named Constructor для перетворення з типу Map в модель класу. Після цього створено метод перетворення з моделі класу в тип Map, і функцію для перетворення з класу моделі у формат JSON у вигляді рядка. Також створюємо функцію для перетворення відповіді з API в відповідний клас моделі.

Щоб видалити користувача, використано віджет Dismiss для того, щоб видалити елемент зі списку та видалити користувача з бази даних, викликавши метод deleteUser ().

4.6. Опис алгоритму створення QR-кодів

Процес генерації QR кода поділяється на декілька чітких етапів:

— кодування даних;

— додавання інформації про версію та заповнення;

— поділ інформації на блоки;

— створення байтів корекції;

— об’єднання блоків;

— розміщення інформації на QR коді.

QR код підтримує кілька способів кодування даних, в залежності від того, які символи використовуються: цифровий, буквено-цифровий, Kanji (китайсько-японські ієрогліфи) і побітовий рівні кодування.

Ще одна властивість QR коду — його версія (чим вона більша, тим більший розмір). Всього існує 40 версій. Номер версії залежить від кількості закодованої інформації і від рівня корекції. Зазначають максимальну кількість корисної інформації разом із службовою (в бітах), яке можна закодувати в QR коді відповідної версії. Тепер потрібно перед послідовністю біт, отриманої в попередньому пункті, додати на початку два поля: спосіб кодування і кількість даних. Спосіб кодування — поле довжиною 4 біта, яке має наступні значення: 0001 для цифрового кодування, 0010 для буквено-цифрового і 0100 для побітового. Кількість даних — це кількість кодованих символів, а для побітового — кількість байт (а не біт в отриманої послідовності), представлене у вигляді двійкового числа певної довжини.

На етапі заповнення є послідовність біт даних, кількість біт в якій не кратне 8. Треба доповнити її нулями так, щоб її довжина стала кратна 8. Тепер цю

послідовність біт можна розбити на групи по 8 біт і представити у вигляді послідовності байт. Якщо кількість біт в поточній послідовності байт менше того, яке потрібно для обраної версії, то її треба доповнити по черзі байтами 11101100 і 00010001. Таким чином, отримано сукупність цифрових даних, довжина якої відповідає обраній версії QR коду.

Сукупність цифрових даних, отримана на попередньому етапі, поділяється на визначену для версії і рівня корекції кількість блоків. Якщо кількість блоків дорівнює одному, то цей етап можна пропустити. Для того, щоб визначити кількість байт в кожному блоці, треба розділити всю кількість байт на кількість блоків даних. Якщо це число не ціле, то треба визначити залишок від ділення. Цей залишок визначає скільки блоків з усіх доповнені (такі блоки, кількість байт в яких більше на один, ніж в інших). Важливо, що доповненими блоками повинні бути не перші блоки, а останні. Блок заповнюється байтами з даних повністю. Коли поточний блок повністю заповнюється, чергу переходить до наступного. Байтів даних повинно вистачити рівно на всі блоки, ні більше і ні менше.

На даному етапі є кілька блоків даних і стільки ж блоків байтів корекції, їх потрібно об'єднати в один потік байтів. Робиться це в такий спосіб: з кожного блоку даних по черзі береться один байт інформації, коли черга доходить до останнього блоку, з нього береться байт і черга переходить до першого блоку. Так триває до тих пір, поки в кожному блоці не закінчаться байти. Якщо в поточному блоці вже немає байтів, то він пропускається — таке відбувається, коли звичайні блоки вже порожні, а в доповнених ще є по одному байту.

Маємо сукупність цифрових даних, яка готова для розміщення на полотно. Полотно складається з модулів — елементарних квадратів. Пошукові елементи розміщуються в верхніх та лівих кутках. Смуги синхронізації починаються від нижнього правого чорного модуля і йдуть, чергуючи чорні та білі модулі, вниз і вправо до протилежних пошукових візерунків. При нашаруванні на візерунок вирівнювання він повинен залишитися без змін. Код версії дублюється в двох місцях, причому дзеркально, тобто вказавши колір модуля в координатах (x, y),

можна сміливо вказувати такий же колір в координатах (y, x). Модулі в цих місцях шикуються наступним чином:1 — чорний, 0 — білий.

Заповнення відбувається біт за бітом з байтів даних. Якщо даних не вистачає, то простір, що лишився заповнюється нульовими модулями.

4.7. Опис реалізації алгоритму виправлення пошкоджень

Існує щонайменше два типи дешифраторів Ріда-Соломона: несистематичні і систематичні дешифратори.

Обчислення несистематичних коригувальних кодів Ріда-Соломона здійснюється множенням інформаційного слова на породжений поліном, в результаті чого утворюється кодове слово, яке повністю відрізняється від вихідного інформаційного слова, а тому для безпосереднього вживання категорично непридатне. Для приведення отриманих даних в початковий вигляд потрібно виконати ресурсномісткі операції декодування, навіть якщо дані не спотворені і не вимагають відновлення.

При систематичному кодуванні, навпаки, вихідне інформаційне слово залишиться незмінним, а коригувальні коди (так звані символи парності) додаються в його кінець, завдяки чому до операції декодування доводиться вдаватися лише в разі дійсного руйнування даних. Обчислення несистематичних коригувальних кодів Ріда-Соломона здійснюється діленням інформаційного слова на породжений поліном. При цьому всі символи інформаційного слова зсуваються на $n - k$ байт вліво, а на місце, що звільнилося записується $2t$ байт залишку.

Архітектурно дешифратор являє собою сукупність зсувних регістрів (shift registers), об'єднаних за допомогою суматорів і множників, що функціонують за правилами арифметики Галуа. Зсувний регістр (інакше званий регістром зсуву) представляє послідовність елементів пам'яті, званих розрядами, кожен з яких містить один елемент поля Галуа $GF(q)$. Символ, який міститься в розряді, залишаючи цей розряд, переміщується на вихідну лінію. Одночасно з цим розряд

забирає символ, що знаходиться на його вхідний лінії. Заміщення символів відбувається дискретно, в строго певні проміжки часу, звані тактами.

При апаратній реалізації зсувного регістру його елементи можуть бути об'єднані як послідовно, так і паралельно. При послідовному об'єднанні пересилання одного m-розрядного символу вимагає m-тактів, в той час як при паралельному вона здійснюється всього за один такт.

Низька ефективність програмних реалізацій шифраторів Ріда-Соломона пояснюється тим, що розробник не може здійснювати паралельне об'єднання елементів зсувного регістру і змушений працювати з тією шириною розрядності, яку «нав'язує» архітектура даної машини. Однак створити 4-елементний 8-бітний регістр зсуву паралельного типу на процесорах сімейства IA32 цілком реально.

Декодування кодів Ріда-Соломона є досить складним завданням, вирішення якого виливається в громіздкий, заплутаний і надзвичайно неочевидний програмний код, що вимагає від розробника великих знань у багатьох областях вищої математики. Типова схема декодування, що отримала назву авторегресійна спектрального методу декодування, складається з наступних кроків (рисунок 4.6):

обчислення синдрому помилки (синдромний декодер);

— побудова полінома помилки, здійснюване або за допомогою високоефективного, але складно реалізованого алгоритму Берлекемпа-Мессі, або за допомогою простого, але повільного Евклідового алгоритму;

— знаходження коренів даного полінома зазвичай вирішується алгоритмом Ченя;

— визначення характеру помилки, яке зводиться до побудови бітової маски, що обчислюється на основі звернення алгоритму Форне або будь-якого іншого алгоритму звернення матриці;

— нарешті, виправлення помилкових символів шляхом накладення бітової маски на інформаційне слово і послідовного інвертування всіх пошкоджених бітів через операцію XOR.

Синдром є залишком ділення слова, що декодується, $c(x)$ на породжений поліном $g(x)$, і, якщо цей залишок дорівнює нулю, кодове слово вважається непошкодженим. Ненульовий залишок свідчить про наявність щонайменше однієї помилки. Залишок від ділення дає многочлен, що не залежить від вихідного повідомлення і визначається виключно характером помилки (syndrome - грецьке слово, що позначає сукупність ознак і / або симптомів, що характеризують захворювання).

Обчислення синдрому помилки відбувається ітеративно, так що обчислення результуючого полінома (також званого відповіддю від англійського «answer») завершується безпосередньо в момент проходження останнього символу парності через фільтр. Всього потрібно $2t$ циклів «прогону» декодуємих даних через фільтр, — по одному прогону на кожен символ результуючого полінома.

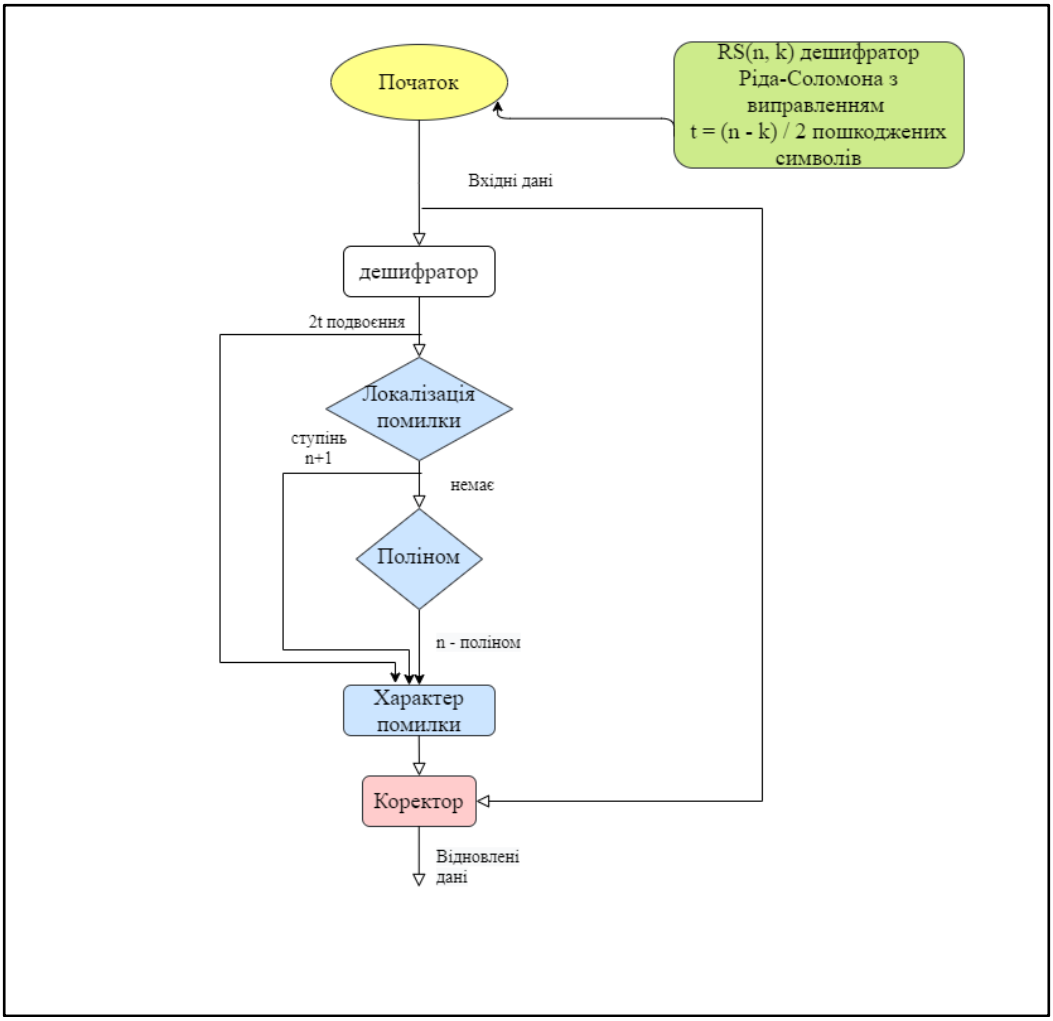

Рисунок 4.6 — Алгоритм дешифратора Ріда-Соломона

Отриманий синдром описує конфігурацію помилки, але ще не описує, які саме символи отриманого повідомлення були пошкоджені. Ступінь синдромного полінома, що дорівнює $2t$ набагато менше ступеня полінома повідомлення, що дорівнює n, і між їхніми коефіцієнтами немає прямої відповідності. Поліном, коефіцієнти якого безпосередньо відповідають коефіцієнтам пошкоджених символів, називається поліномом локатора помилки і за загальноприйнятою угодою позначається грецькою буквою L (лямбда).

Коли поліном локатора помилки відомий, його корінь визначає місце розташування пошкоджених символів в прийнятому кодовому слові. Залишається цей корінь знайти. Найчастіше для цього використовується алгоритм Ченя (Chien search), аналогічний за своєю природою зворотному перетворенню Фур'є і фактично зводиться до примітивного перебору (brute force, exhaustive search) всіх можливих варіантів. Якщо результат наближається до нуля — вважається, що необхідні корені знайдені.

Отже, на даному етапі відомо, які символи кодового слова пошкоджені, але поки ще не готові відповісти на питання: як саме вони пошкоджені. Використовуючи поліном синдрому і корінь полінома локатора, можна визначити характер руйнувань кожного з перекручених символів. Зазвичай для цієї мети використовується алгоритм Форне (Forney), що складається з двох стадій: спочатку шляхом згортки полінома синдрому поліномом локатора L отримано певний проміжний поліном, умовно позначається грецькою літерою W. Потім на основі W-полінома обчислюється нульова позиція помилки (zero error location), яка в свою чергу ділиться на похідну від L-полінома. В результаті виходить бітова маска, кожен з встановлених бітів якої відповідає пошкодженому біту і для відновлення кодового слова в початковий вигляд всі пошкоджені біти повинні бути інвертовані, що здійснюється за допомогою логічної операції XOR.

На цьому процедура декодування прийнятого кодового слова вважається закінченою. Залишається відсікти $n - k$ символів парності, і отримане інформаційне слово готове до використання.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1. Модуль формування QR-кодів

Мета електронної візитної картки — створити єдиний простий файл, яким можна поділитися, з усіма іншими контактними даними. Таким чином, одержувач не мусить вводити дані вручну, щоб стежити за сторінками у соціальних мережах або зв'язуватися, використовуючи детальну інформацію. Електронні візитні картки допомагають ділитися з іншими людьми інформацією про те, яка людина є і чим займається у бізнесі. Це також допомагає максимізувати присутність в Інтернеті та збільшити власні можливості.

Останні чотири століття застосовуються паперові візитки. В основному більшість паперу виготовляється з дерева. Найбільше розчаровує те, що понад 88% візитних карток викидають за тиждень після обміну. Тепер можливо уявити масштаби, скільки дерев вирубано. Для того, щоб навести беззаперечні аргументи проти використання паперових візитівок, наведено наступну статистику. Збільшення продажів компанії на кожні 2000 паперових карток зменшує ріст прибутків на 2,5%. Картки викинуто менш ніж за тиждень — 88% від усіх створених візитівок за тижневий період. Люди, які викинули картку, оскільки в даний момент їм не потрібна послуга — 63% від тих, хто отримав паперові візитівки. Люди, які додають інформацію про бізнес у електронну візитну картку — всього 9% від кількості людей, що користується візитними картками для ведення власного бізнесу.

Усі електронні візитні картки служать насамперед одній меті. Більшість використовує багато сучасних платформ (LinkedIn, Twitter, Instagram та інших), що

може ділитися усіма подробицями та інформацією одночасно. Ось де найбільш ефективні та доречні віртуальні візитні картки.

На головному екрані мобільного застосунку можна переглянути існуючі візитні картки в своєму профілі. Якщо натиснути на цю картку, то вона буде перевернута і позаду прикріплений QR-код з зашифрованою інформацією по даній картці (рисунок 5.1).

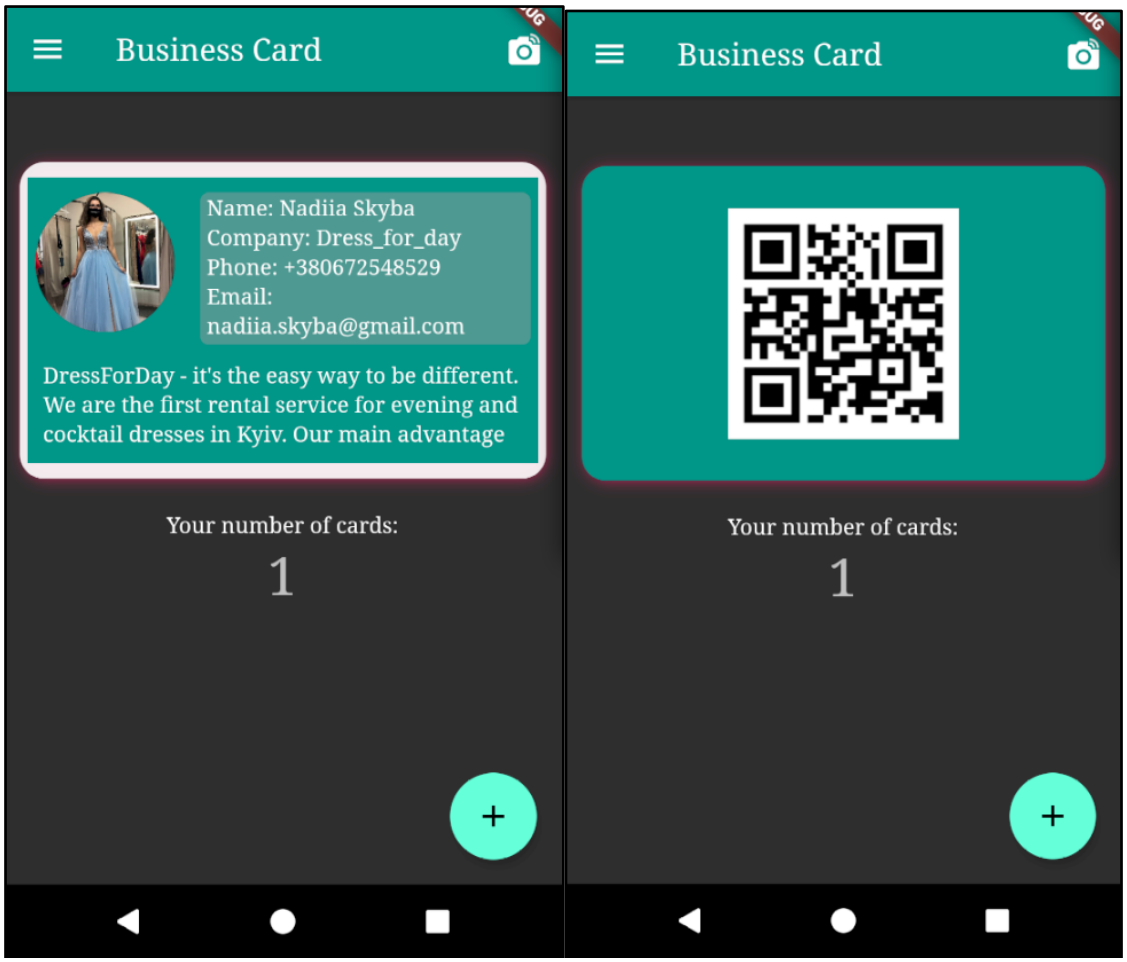

Рисунок 5.1 — Зовнішній вигляд основної сторінки програмного застосунку

Справді вигідно використовувати електронні візитні картки. Це інноваційна альтернатива паперовим візиткам. Наведено деякі з переваг використання таких карток, які варті уваги:

— можливість показати історію, що лежить за вашим брендом при першій зустрічі з потенційними клієнтами;

— можливість зручно зберігати всі контакти, зв’язки та навіть облікові записи соціальних мереж у програмному застосунку;

— сучасні застосунки електронних візитівок автоматично зберігають контактну інформацію, таку як ім’я, номер телефону, електронну адресу та фото людей, яким надано інформацію про власну картку;

— сучасні застосунки електронних візитівок підтримують відео, аудіо та фірмову графіку тат логотип безпосередньо всередині картки, завдяки цій ефективній функції є можливість привернути увагу людей;

— можливість ділитися електронними картками навіть з людьми, у яких програма не встановлена на мобільному пристрої;

— електронні візитки є економічно вигідними, це допоможе заощадити гроші, які довелося б витратити на паперові візитки;

— можливість ефективно змінити контактні дані, скажімо, потрібно змінити адресу електронної пошти. Якщо використовувати паперові картки, тоді потрібно передрукувати їх. Але за допомогою електронних візитних карток можна персоналізувати їх у будь-який час на ходу;

— спільний доступ до електронної візитної картки простіший, ніж будь-коли раніше, можливість відправляти та отримувати карту на будь-які відстані за одну процедуру натискання;

— і останнє, але не менш важливе: електронні візитні картки екологічні, рятують дерева, рятують планету.

Електронні візитівки економічно вигідні, і не потрібно турбуватися про їхню втрату. Створений програмний застосунок має дуже зручні можливості, які є корисними в сфері бізнесу. Електронні картки не обмежують місця — можна додати на власну картку більше або менше інформації. Окрім звичайної контактної інформації (наприклад, імені, компанії, електронної пошти та номера телефону), можна наповнити картку фотографіями чи відео, логотипом, профілями в соціальних мережах тощо.

Створена програмна система дозволяє виготовити декілька електронних візитних карток з різною інформацією на кожній картці, тож є можливість мати картку для робочих контактів, для клієнтів, а іншу для друзів. На відміну від інших програм для візитних карток, за допомогою реалізованого програмного забезпечення є можливість поділитися сласною візитною карткою з ким завгодно, навіть якщо її у тої людини немає.

На головній сторінці візитівок з двовимірним зображенням, яке відкривається в повноекранному режимі, якщо на нього натиснути (рисунок 5.2), відображаються усі створені картки, та їхня кількість виводиться на екран.

Рисунок 5.2 — Широкоформатний розмір QR-коду для зручності його сканування

5.2. Функціональний модуль сканера QR-кодів

Сканер QR-кодів, який коригує втрату даних можна відкрити, натиснувши на значок камери на головному екрані застосунку. При переході до сканеру потрібно

надати дозвіл на використання мобільним застосунком камери телефону. Після доступу до камери слід навести виділену червоним область сканування на двовимірне зображення. За декілька секунд в полі результат отримано відповідь після сканування коду (рисунок 5.3). Також можна використовувати спалах при темній порі доби або в темному приміщенні, натиснувши на відповідну кнопку на екрані сканера. Кнопки пауза та старт зупиняють потік сканування та відновлюють його відповідно. За замовчуванням встановлено використання сканером передньої камери, але можна змінити на фронтальну, натиснувши на кнопку “Camera facing back”.

Рисунок 5.3 — Аналіз та розпізнавання даних, поміщених у двовимірне зображення на прикладі розробленої системи

В процесі створення власного двовимірного зображення може бути додано різноманітні кольори та дизайн до зображення, що створює так зване пошкодження матричного коду (рисунок 5.4). Проте допускається невеликий відсоток відхилення від точності даних коду для того, щоб створювати свій дизайн. Для цього в програмну систему додано та реалізовано спеціальний алгоритм коригування пошкоджень Ріда-Соломона з високим рівнем виправлення втрати даних.

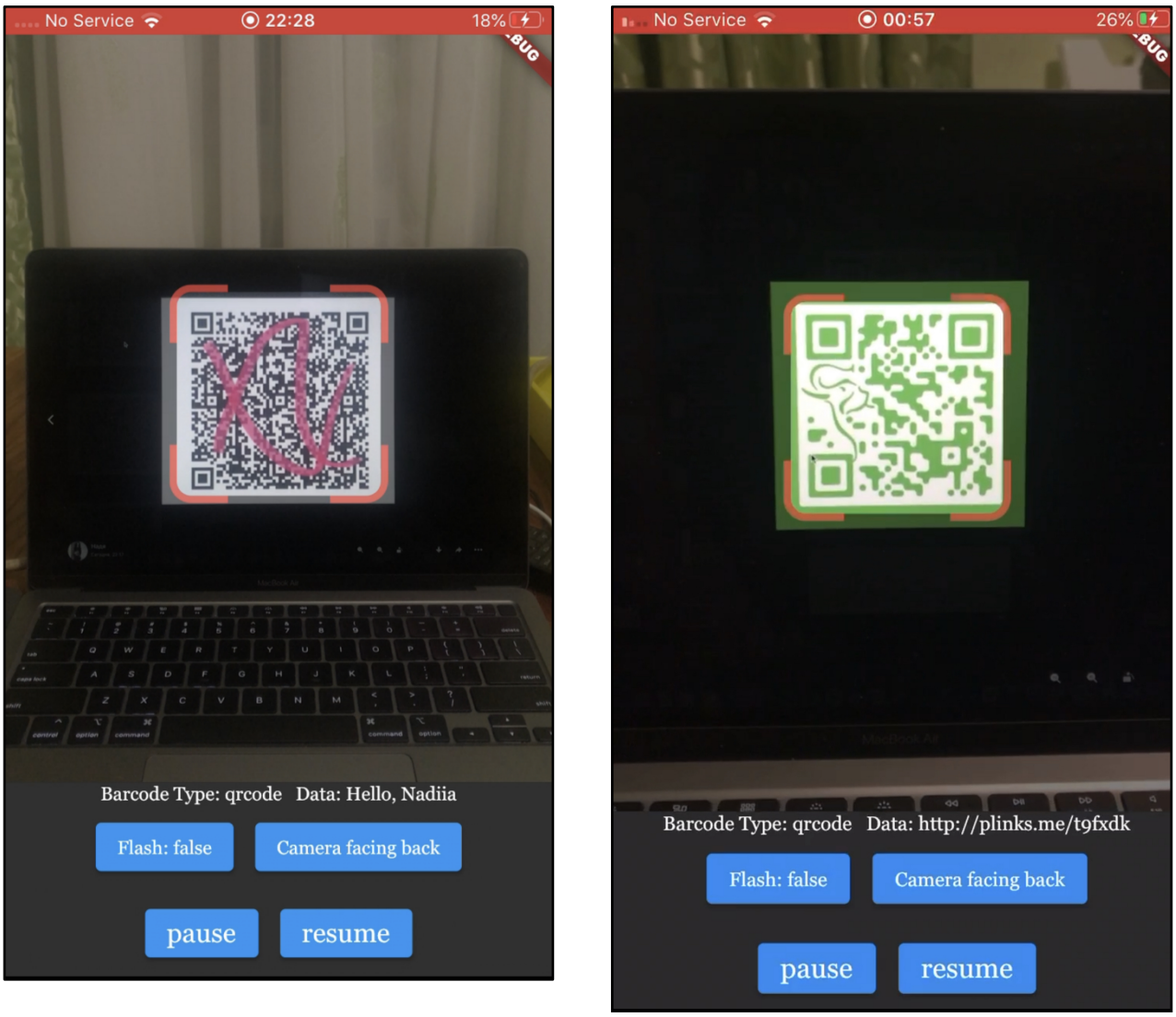

Рисунок 5.4 — Приклад сканування пошкоджених кодів власним дизайном та кольорами

Профіль користувача має вигляд списку з відповідними вікнами, на які можна перейти (рисунок 5.5). Зверху міститься основна інформація про користувача та його фото, яке також можна відкрити в широкоформатному режимі. Перехід між вікнами мобільного застосунку для профілю реалізовано у вигляді списку, функціональні модулі якого створені для головної сторінки, сторінки зі списком збережених об’єктів візитівок та сторінки налаштувань.

Розділ налаштувань надає можливість змінювати контактну інформацію користувача та пароль. Введена раніше інформація зберігається в полях введення. Реалізовано підтвердження нового паролю та перевірка на рівність значень двох полів — введення нового паролю та його підтвердження. Тільки в разі того, що ці два значення однакові буде створено новий пароль.

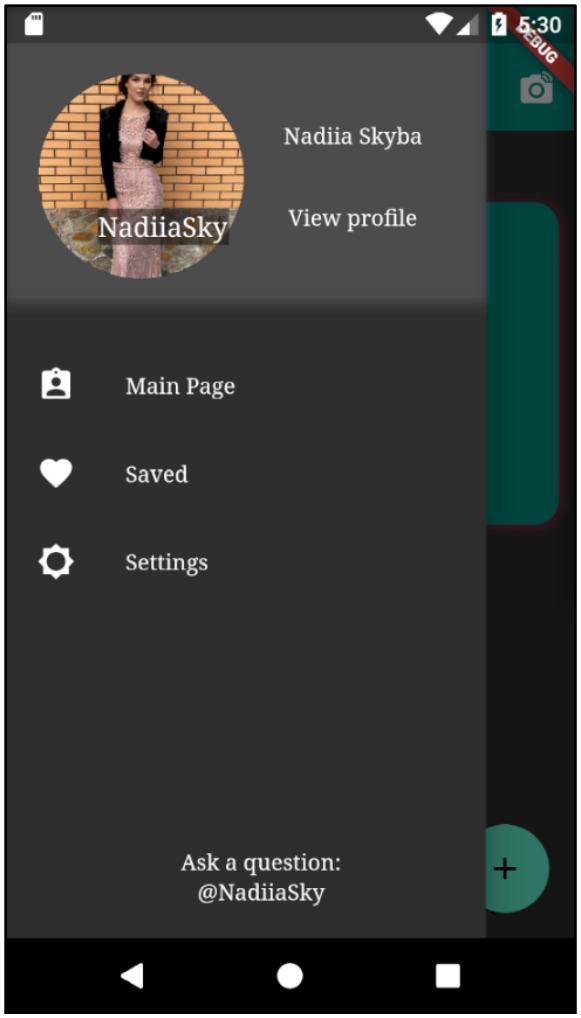

Рисунок 5.5 — Інтерфейс профілю розробленої системи

Аналогічну форму також має і функція додавання нової візитної картки, на яку можна перейти, натиснувши на кнопку “+” на головному екрані мобільного застосунку. Форма має поля, які необхідно заповнити для створення нового об’єкту (рисунок 5.6). Написано регулярні вирази для перевірки коректності вводу мобільного телефону та електронної пошти відповідного формату.

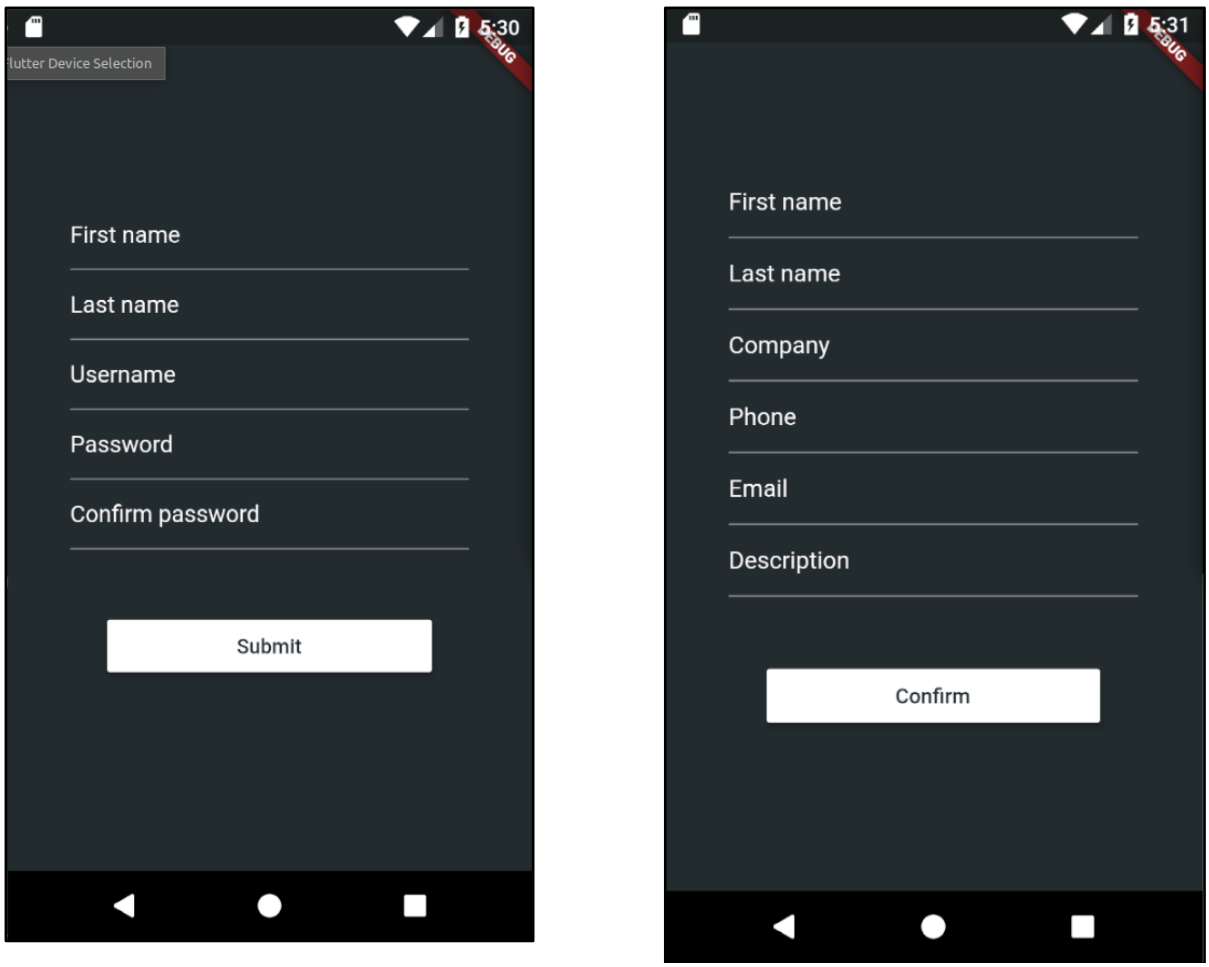

Рисунок 5.6 — Форма введення інформації для нових записів та об’єктів

Застосування двовимірного зображення — новий спосіб використання візитних карток у соціальних мережах та на сервісах обміну. Це дозволяє обмінюватись контактним даними, професійною інформацією, зразками роботи та багато іншого лише одним сканування за допомогою камери телефону. Такий спосіб введення бізнесу та інформації залучає нових клієнтів та можливі інвестиції.

ВИСНОВКИ

У даній роботі вивчено і проаналізовано проблему обробки та розпізнавання даних двовимірного зображення.

— Проаналізовано процес розпізнавання двовимірного зображення.

— Досліджено процес створення QR-кодів.

— Спроектовано архітектуру системи.

— Розроблено алгоритм, який вирішує поставлену задачу дипломної роботи — модуль розпізнавання та обробки даних.

— Описано реалізацію зручного користувацького інтерфейсу, засоби та інструменти реалізації архітектури програмної системи. Для роботи обрано середовище розробки Android Studio. Також використано СУБД SQLite та бібліотеки плагінів Dart та Flutter.

— Проведено ряд тестів над програмою, щоб отримати остаточну оцінку ефективності написаного алгоритму. Проведено ряд експериментів з пошкодженими зображеннями. Досліджено предметну область та сферу застосування створеного сканеру QR-кодів.

Розроблено програмне забезпечення для вирішення наступних ключових задач:

— коригування помилок при втраті чи пошкодженні даних двовимірного зображення;

— коректна обробка двовимірних зображень при використанні різноманітної колірної гами;

— автоматичне визначення типу двовимірного зображення та версії QR-коду;

— відсутність необхідності підключення до Інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Winter Mick. Scan me: Everybody's Guide to the Magical World of QR Codes / Mick Winter — Westsong Publishing, 2011. — 144 p.

2. A call for more (and better) QR code experiences. [Електронний ресурс] — Режим доступу: https://medium.com/swlh/a-call-for-more-and-better-qr-code-experiences-2872cd97ec56.

3. Boyles Aric. The Complete Guide to QR Codes Kindle Edition / Aric Boyles — QR-Codes.com, 2012. — 35 p.

4. QR-кодування та альтернативні технології? [Електронний ресурс] — Режим доступу: https://ofp.cibs.ubs.edu.ua/files/1403/14zhoqta.pdf.

5. Wikiversity. Reed-Solomon codes for coders [Електронний ресурс] — 2021. Режим доступу: https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders

6. Keti Matt. Reed-Solomon Error-correcting Codes — The Deep Hole Problem / Matt Keti — Department of Mathematics University of California, Irvine, 2012. — 20 p.

7. Augot D., Morain F. Discrete Logarithm Computations Over Finite Fields Using Reed-Solomon Codes / Daniel Augot, François Morain — HAL, 2012. — 17p.

8. Thonky. QR Code Tutorial [Електронний ресурс] — Thonky.com's, 2020 — Режим доступу: https://www.thonky.com/qr-code-tutorial/introduction.

9. Zhu G., Wan D. Computing Error Distance of Reed-Solomon Codes / Guizhen Zhu, Daqing Wan — TAMC, 2012. — 224 p.

10. Does the color of the QR code affect its reliability? [Електронний ресурс] — Режим доступу: https://ofp.cibs.ubs.edu.ua/files/1403/14zhoqta.pdf.

11. Li D., Zhang B., Xie H. The Design of Color QRCode With Logo / Dong Li, Boliang Zhang, Hongan Xie — Information engineering department, Engineering College of CAPF, 2016 — 49 p.

12. You F., Zhang Q., Welt B. Research on Color Matching Model for Color QR Code / Fei You, Qingli Zhang, Bruce Welt — Journal of Applied Packaging Research, 2018 — 12 p.

13. Quanlei B. Color picture segmentation in HSV color space / Quanlei Bao — Software Guide, 2010 — 171 p.

14. Tongzhe Li, Kent D. M. To scan or not to scan: the question of consumer behavior and QR codes on food packages / Tongzhe Li, Kent D. Messer — Journal of Agricultural and Resource Economics, 2019 — 47 p.

15. Taveerad N., Vongpradhip S. Development of color QR code for increasing capacity / Nutchanad Taveerad, Sartid Vongpradhip — 11th International Conference on Signal-Image Technology & Internet-Based Systems, 2015 — 32 p.

16. Huaguo J. Study and Application of Encoding and Decoding Algorithms for Colored Two-dimensional Code on Mobile Terminals / Jia Huaguo — Hangzhou, Zhejiang University of Technology, 2009 — 17 p.

17. Zaccagnino C. Programming Flutter: Native, Cross-Platform Apps the Easy Way (The Pragmatic Programmers) / Carmine Zaccagnino — Pragmatic Bookshelf, 2020 — 370 p.

18. Payne R. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps / Rap Payne — Apress, 2019 — 334 p.

19. Sande J., Galloway M. Dart Apprentice (First Edition): Beginning Programming with Dart / Jonathan Sande, Matt Galloway — Bowker, 2021 — 295 p.

20. Buckett C. Dart in Action / Chris Buckett — Manning Publications, 2013 — 424 p.

21. Kreibich J. A. Using SQLite: Small. Fast. Reliable. Choose Any Three / Jay. A. Kreibich — O'Reilly Media, 2010 — 530 p.

22. Nield T. Getting Started with SQL: A Hands-On Approach for Beginners / Thomas Nield — O'Reilly Media, 2016 — 134 p.

ДОДАТОК А

Інструментальні засоби розпізнавання та обробки даних двовимірного зображення

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ72138_21Б

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕ ПС_ ТІ72138_21Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕ ПС_ ТІ72138_21Б 12-1	QRScanner .dart	Компонент, що містить алгоритм сканування пошкодженого QR-коду
УКР.НТУУ”КПІ”_ТЕФ_АПЕ ПС_ ТІ72138_21Б 12-2	card.dart	Компонент збереження інформації візитівки та додавання даних в БД
УКР.НТУУ”КПІ”_ТЕФ_АПЕ ПС_ ТІ72138_21Б 12-3	addCard.dart	Компонент додавання візитівки, збереження її інформації та виведення на головний екран

ДОДАТОК Б

Інструментальні засоби розпізнавання та обробки даних двовимірного зображення

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ72138_21Б 12-1

Аркушів 9

Київ — 2021

card.dart

```
class Card{
  final int id;
  final String imageUrl;
  final String name;
  final String lastName;
  final String company;
  final String phone;
  final String email;
  final String description;


Card(this.id, this.imageUrl, this.name, this.lastName, this.company, this.phone, this.email,
this.description);


  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'avatar': imageUrl,
      'name': name,
      'lastName': lastName,
      'company': company,
      'phone': phone,
      'email': email,
      'description': description,
    };
  }


  @override
  String toString() {
    return 'Card{id: $id, avatar: $imageUrl, name: $name, lastName: $lastName,'
        ' company: $company, phone: $phone, email: $email, description: $description}';
  }
```

```
}
```

QRScanner.dart

```
class QRViewExample extends StatefulWidget {
  const QRViewExample({
    Key key,
  }) : super(key: key);

  @override
  State<StatefulWidget> createState() => _QRViewExampleState();
}

class _QRViewExampleState extends State<QRViewExample> {
  Barcode result;
  QRViewController controller;
  final GlobalKey qrKey = GlobalKey(debugLabel: 'QR');

  // In order to get hot reload to work we need to pause the camera if the platform
  // is android, or resume the camera if the platform is iOS.
  @override
  void reassemble() {
    super.reassemble();
    if (Platform.isAndroid) {
      controller.pauseCamera();
    }
    controller.resumeCamera();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: <Widget>[
          Expanded(flex: 4, child: _buildQrView(context)),
          Expanded(
            flex: 1,
            child: FittedBox(
              fit: BoxFit.contain,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: <Widget>[
```

```
if (result != null)
  Text(
      'Barcode Type: ${describeEnum(result.format)}   Data: ${result.code}')
else
  Text('Scan a code'),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: <Widget>[
    Container(
      margin: EdgeInsets.all(8),
      child: ElevatedButton(
          onPressed: () async {
            await controller?.toggleFlash();
            setState(() {});
          },
          child: FutureBuilder(
            future: controller?.getFlashStatus(),
            builder: (context, snapshot) {
              return Text('Flash: ${snapshot.data}');
            },
          )),
    ),
    Container(
      margin: EdgeInsets.all(8),
      child: ElevatedButton(
          onPressed: () async {
            await controller?.flipCamera();
            setState(() {});
          },
          child: FutureBuilder(
            future: controller?.getCameraInfo(),
            builder: (context, snapshot) {
              if (snapshot.data != null) {
                return Text(
                    'Camera facing ${describeEnum(snapshot.data)}');
              } else {
                return Text('loading');
              }
            },
          )),
    )
  ],
),
```

```
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.center,
                  children: <Widget>[
                    Container(
                      margin: EdgeInsets.all(8),
                      child: ElevatedButton(
                        onPressed: () async {
                          await controller?.pauseCamera();
                        },
                        child: Text('pause', style: TextStyle(fontSize: 20)),
                      ),
                    ),
                    Container(
                      margin: EdgeInsets.all(8),
                      child: ElevatedButton(
                        onPressed: () async {
                          await controller?.resumeCamera();
                        },
                        child: Text('resume', style: TextStyle(fontSize: 20)),
                      ),
                    )
                  ],
                ),
              ],
            ),
          ),
        )
      ],
    ),
  );
 }

  Widget _buildQrView(BuildContext context) {
    // For this example we check how width or tall the device is and change the scanArea and
overlay accordingly.
    var scanArea = (MediaQuery.of(context).size.width < 400 ||
        MediaQuery.of(context).size.height < 400)
        ? 150.0
        : 300.0;
    // To ensure the Scanner view is properly sizes after rotation
    // we need to listen for Flutter SizeChanged notification and update controller
    return QRView(
      key: qrKey,
```

```
      onQRViewCreated: _onQRViewCreated,
      overlay: QrScannerOverlayShape(
          borderColor: Colors.red,
          borderRadius: 10,
          borderLength: 30,
          borderWidth: 10,
          cutOutSize: scanArea),
    );
  }

  void _onQRViewCreated(QRViewController controller) {
    setState(() {
      this.controller = controller;
    });
    controller.scannedDataStream.listen((scanData) {
      setState(() {
        result = scanData;
      });
    });
  }

  @override
  void dispose() {
    controller?.dispose();
    super.dispose();
  }
}
```

addCard.dart

```
class AddCard extends StatefulWidget{

  @override
  _AddCardState createState() => _AddCardState();
}

class _AddCardState extends State<AddCard> {
  final _formKey = new GlobalKey<FormState>(); // outside the build() method

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: Container(
          color: Color.fromRGBO(36, 43, 47, 1),
```

```
        padding: const EdgeInsets.symmetric(horizontal: 43.0),
        child: Form(
          key: _formKey,
          child: Container(
            alignment: Alignment.center,
            child: SingleChildScrollView(    // new line
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  _buildFirstName(),
                  _buildLastName(),
                  _buildCompany(),
                  _buildPhone(),
                  _buildEmail(),
                  _buildDescription(),
                  _buildConfirmButton(context)
                ],
              ),
            ),
          ),
        ),
      ));
  }
}

InputDecoration _buildInputDecoration(String hint, String iconPath) {
  return InputDecoration(
      focusedBorder: UnderlineInputBorder(
          borderSide: BorderSide(color: Color.fromRGBO(252, 252, 252, 1))),
      hintText: hint,
      enabledBorder: UnderlineInputBorder(
          borderSide: BorderSide(color: Color.fromRGBO(151, 151, 151, 1))),
      hintStyle: TextStyle(color: Color.fromRGBO(252, 252, 252, 1)),
      icon: iconPath != '' ? Image.asset(iconPath) : null,
  );
}

Widget _buildFirstName() {
  return TextFormField(
    validator: (value) =>
    value.isEmpty ? "First name cannot be empty" : null,
    style: TextStyle(
        color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
    decoration:
```

```
    _buildInputDecoration("First name", ''),
  );
}

Widget _buildLastName() {
  return Container(
      //margin: const EdgeInsets.only(left: 40),
      child: TextFormField(
        validator: (value) =>
        value.isEmpty ? "Last name cannot be empty" : null,
        style: TextStyle(
            color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
        decoration: _buildInputDecoration("Last name", ''),
      ));
}


Widget _buildCompany() {
  return Container(
      child: TextFormField(
        validator: (value) => value.isEmpty ||
            (value.isNotEmpty) //&& value != _passwordController.text)
            ? "Must not be empty"
            : null,
        style: TextStyle(
            color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
        decoration: _buildInputDecoration("Company", ''),
      ));
}


Widget _buildPhone() {
  return Container(
      child: TextFormField(
        validator: (value) =>
        value.isEmpty ? "Phone cannot be empty and must be +380()" : null,
        style: TextStyle(
            color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
        decoration: _buildInputDecoration("Phone", ''),
      ));
}
Widget _buildEmail() {
  return TextFormField(
    validator: (value) => !isEmail(value) ? "Sorry, we do not recognize this email address" : null,
    style: TextStyle(
```

```
      color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
   decoration: _buildInputDecoration("Email", ''),
  );
}

bool isEmail(String value) {
  String regex =
     r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$';

  RegExp regExp = new RegExp(regex);

  return value.isNotEmpty && regExp.hasMatch(value);
}
Widget _buildDescription() {
  return TextFormField(
    validator: (value) =>
    value.length <= 6 ? "Description must be 6 or more characters in length" : null,
    style: TextStyle(
       color: Color.fromRGBO(252, 252, 252, 1), fontFamily: 'RadikalLight'),
    decoration:
    _buildInputDecoration("Description", ''),
  );
}
Widget _buildConfirmButton(BuildContext context) {
  return Container(
    margin: const EdgeInsets.only(top: 43.0),
    width: MediaQuery.of(context).size.width * 0.62,
    child: RaisedButton(
      child: const Text(
        "Confirm",
        style: TextStyle(
            color: Color.fromRGBO(40, 48, 52, 1),
            fontFamily: 'RadikalMedium',
            fontSize: 14),
      ),
      color: Colors.white,
      elevation: 4.0,
      onPressed: () {
        //_validateAndSubmit();
      },
    ),
  );
}
```

ДОДАТОК В

Інструментальні засоби розпізнавання та обробки даних двовимірного зображення

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ72138_21Б 13-1

Аркушів 10

Київ — 2021

АНОТАЦІЯ

Додаток містить опис мобільного застосунку для коректного розпізнавання та обробки двовимірних зображень, з урахуванням можливого пошкодження їхнього матричного коду. Вимоги визначенні в розділі 1, а саме:

— автоматичне визначення версії / типу QR-коду або введення вручну;

— виправлення помилок / кольору;

— настроюваний розмір виводу, відступу, кольору фону і переднього плану;

— підтримка QR-коду версії 1 – 40;

— перегляд створених об’єктів, які містять дані двовимірного зображення;

— зручний користувацький інтерфейс, який дозволяє також змінювати поточну інформацію та додавати нові об’єкти.

Мобільний застосунок розроблений мовою програмування Dart з використанням технології Flutter та плагіну бази даних SQLite у редакторі вихідного коду Android Studio.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ 80
ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 81
ОПИС ЛОГІЧНОЇ СТРУКТУРИ 82
ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ 83
ВИКЛИК І ЗАВАНТАЖЕННЯ 84
ВХІДНІ ДАНІ 85
ВИХІДНІ ДАНІ 86

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів мобільного застосунку для розпізнавання та обробки двовимірних зображень, що виконують деякі завдання, визначенні в розділі 1. Додаток Б містить програмний код цих компонентів.

Для роботи мобільного застосунку потрібно мати мобільний пристрій на платформі Android версії Nougat (7.1.1) і вище, або iOS версії 8 і вище. Мобільний пристрій має бути оснащений камерою.

Мобільний застосунок розроблений мовою програмування Dart з використанням технології Flutter у редакторі вихідного коду Android Studio.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений мобільний застосунок надає користувачу наступний функціонал:

— Реєстрація/авторизація в системі.

— Внесення даних для електронної візитівки.

— Створення та збереження електронних карток.

— Розпізнавання та обробка QR-кодів.

— Перегляд інформації профілю та сторінки з візитними картками.

— Редагування особистих даних.

— Збереження електронних карток до списку улюблених.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, застосунок складається з двох частин: сканер QR-кодів з виправленням пошкоджень матричного коду та користувацький інтерфейс.

Реалізація дешифратора QR-кодів містить наступні логічні етапи: алгоритм дешифрування та алгоритм виправлення пошкоджень даних.

Задача створення дешифратора розбивається на кілька незалежних зрозумілих підзадач: локалізація QR-коду, орієнтація QR-коду і безпосередньо декодування QR-коду.

Алгоритм виправлення помилок QR-коду реалізована шляхом додавання коду Ріда-Соломона до вихідних даних. Можливість виправлення помилок залежить від обсягу даних, які потрібно виправити.

Створення функціоналу користувацького інтерфейсу розбивається на декілька основних частин:

— Реалізація додавання нових об’єктів електронних карток та збереження їх у базу даних.

— Представлення профілю користувача з відповідними налаштуваннями— функціональні елементи, що відображають інформацію і структурні елементи сторінок мобільного застосунку (заголовки, іконки, кнопки і т. д.).

— Зміна особистої інформації користувача та запис її в БД.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для роботи мобільного застосунку потрібно мати мобільний пристрій на платформі Android версії Nougat (7.1.1) і вище, або на платформі iOS версії 8 і вище.

Мобільний пристрій має бути оснащений камерою.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для завантаження мобільного застосунку на Android необхідно відкрити проект програмного забезпечення в Android Studio та запустити віртуальний пристрій, або під’єднати мобільний пристрій через шнур до комп’ютера. Для інсталяції застосунку на пристрій на платформі iOS потрібно відкрити та запустити програмний проект в середовищі розробки XCode.

ВХІДНІ ДАНІ

Вхідні дані для розпізнавання та обробки двовимірного зображення:

— QR-коди з різним ступенем пошкодження;

— QR-коди усіх версій;

— великі та малі двовимірні зображення різної чіткості.

Вхідна інформація для функціоналу користувацького інтерфейсу:

— дані для створення нового об’єкту візитної картки (ім’я, прізвище, адреса електронної пошти, назва компанії, мобільний телефон і опис компанії);

— дані користувача, які потрібно персоналізувати і які зберігаються в профілі (ім’я, прізвище, ім’я користувача, пароль);

ВИХІДНІ ДАНІ

Вихідна інформація модуля дешифратора (сканування двовимірного зображення):

— оброблена та дешифрована інформація QR-коду;

— визначений тип двовимірного зображення.

Вихідні дані користувацького інтерфейсу:

— створення нового об’єкту електронних карток та зображення їх на головному екрані мобільного застосунку;

— збережені дані профілю користувача;

— список збережених об’єктів електронних візитівок.

— кількість створених візитних карток, що відображається на головному екрані під створеними об’єктами.