Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматики
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

**Бакалаврська кваліфікаційна робота на тему:**

# ПРОГРАМНИЙ МОДУЛЬ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА QR-КОДОМ

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Кудрик В.Я.
(прізвище та ініціали)
Керівник: к.т.н., доцент кафедри КН
Колодний В.В.
(прізвище та ініціали)
«04» червня 2025 р.
Рецензент: к.т.н., доцент кафедри САІТ
Варчук І.В.
(прізвище та ініціали)
«05» червня 2025 р.

**Допущено до захисту**
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
"05" травня 2025 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Кудрику Владиславу Ярославовичу

1. Тема роботи: Програмний модуль розпізнавання товарів за QR-кодом.
керівник роботи: Колодний Володимир Володимирович, к.т.н., доцент
затверджені наказом вищого навчального закладу "20" травня 2025 року №97

2. Строк подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи:
Кількість QR-кодів для одночасного розпізнавання – 1шт;
Кількість товарів для розпізнавання – не менше 1000шт;
Мова програмування – об'єктно-орієнтована.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
вступ, аналіз сучасних технологій розпізнавання товарів за QR-кодом, розробка програмного модуля розпізнавання товарів за QR-кодом, програмна реалізація модуля розпізнавання товарів за QR-кодом, висновки, список використаних джерел, додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
структура програмного модуля розпізнавання товарів за QR-кодом; схема алгоритму розпізнавання товарів за QR-кодом; загальна UML-діаграма класів

6. Консультанти розділів проєкту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Колодний В.В., к.т.н., доцент | | |
| | | | |
| | | | |
| | | | |

6. Дата видачі завдання 05 травня 2025 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1. | Аналіз сучасних технологій розпізнавання товарів за QR-кодом | 05.05.25 | 07.05.25 | |
| 2. | Розробка програмного модуля розпізнавання товарів за QR-кодом | 08.05.25 | 11.05.25 | |
| 3. | Програмна реалізація модуля розпізнавання товарів за QR-кодом | 12.05.25 | 15.05.25. | |
| 4. | Розробка інструкції користувача. | 16.05.25 | 29.05.25 | |
| 5. | Висновки та аналіз отриманих результатів | 30.05.25 | 01.06.25 | |
| 6. | Оформлення матеріалів до захисту БКР | | 04.06.25 | |

Студент _____ Кудрик В.Я.
( підпис ) (прізвище та ініц

Керівник проєкту (роботи) _____ Колодний В.
( підпис ) (прізвище та ініц

# АНОТАЦІЯ

Кудрик В.Я. Програмний модуль розпізнавання товарів за QR-кодом. Бакалаврська кваліфікаційна робота складається з 75 сторінок формату А4, на яких є 22 рисунків, 3 таблиці, список використаних джерел містить 20 найменуваннь.

В роботі обґрунтовано доцільність розробки програмного модуля розпізнавання товарів за QR-кодом. Проаналізовано переваги та недоліки програм аналогів для сканування QR-кодів та розпізнавання їх вмісту. Створено структурну схему кодування/декодування товару за QR-коду. Розроблено алгоритм роботи програмного модуля розпізнавання товарів за QR-кодом. Розроблено UML-діаграми програмного модуля розпізнавання товарів за QR-кодом. Описано процес розробки модуля сканування QR-коду.

Програмний модуль розроблено на об’єктно-орієнтованій мові програмування PHP.

Ключові слова: PHP, QR-код, розпізнавання, товар, сканування.

# ABSTRACT

Kudryk V. Software Module for Product Recognition via QR Code. The bachelor's qualification work consists of 77 A4 pages, containing 22 figures, 3 tables, and a list of 20 references.

The work substantiates the feasibility of developing a software module for product recognition using QR codes. It analyzes the advantages and disadvantages of existing software analogues for scanning and recognizing QR code content. A structural scheme for encoding/decoding product information via QR code is developed. An algorithm for the operation of the product recognition software module is proposed. UML diagrams of the module are created. The process of developing the QR code scanning module is described.

The software module is developed using the object-oriented programming language PHP.



Keywords: PHP, QR code, recognition, product, scanning.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** Актуальність розпізнавання товарів за QR-кодом обумовлена загальною тенденцією до цифровізації бізнес-процесів, автоматизації облікових операцій та інтеграції інтелектуальних систем управління в торгівлі, логістиці та сфері обслуговування. У сучасних умовах ринку підприємства стикаються з необхідністю оптимізації обробки великих обсягів даних, підвищення точності ідентифікації продукції та скорочення часу на її обробку. QR-коди, як двовимірні штрих-коди, стали універсальним інструментом для кодування різноманітної інформації про товари, включаючи артикул, серійний номер, дату виробництва, термін придатності, склад, партію, дані виробника, інструкції з використання, гіперпосилання на вебресурси тощо.

Завдяки компактності та високій ємності QR-кодів, зчитування інформації відбувається за частки секунди за допомогою звичайних мобільних пристроїв або промислових сканерів, що значно полегшує роботу персоналу на складах, у торгових залах та під час доставки. Це забезпечує автоматизоване оновлення залишків, пришвидшує інвентаризацію, підвищує точність операцій і мінімізує людський фактор. У сфері електронної комерції QR-коди використовуються як частина безперервного цифрового ланцюга – від моменту замовлення до отримання товару кінцевим споживачем, сприяючи прозорості й простежуваності ланцюга постачання.

Крім того, розпізнавання QR-кодів дозволяє інтегрувати інформаційні потоки з корпоративними системами управління ресурсами (ERP), системами управління взаємовідносинами з клієнтами (CRM), платіжними сервісами та службами доставки. Це сприяє створенню єдиного інформаційного середовища, у якому всі учасники процесу мають доступ до актуальної інформації в режимі реального часу.

У контексті сучасних викликів — таких як пандемія, потреба в безконтактному обслуговуванні, безпеці споживача та мінімізації фізичної взаємодії з товарами — QR-коди стали ефективним засобом організації дистанційного доступу до інформації. Зчитування коду з етикетки чи пакування дозволяє покупцеві

отримати повний опис товару, переглянути сертифікати якості, відгуки або інструкції без необхідності фізичного контакту з персоналом.

Розпізнавання товарів за QR-кодом є не лише технічним рішенням, а важливою складовою сучасної цифрової інфраструктури бізнесу. Його актуальність зростає з кожним роком, оскільки технологія дозволяє поєднати швидкість, точність, гнучкість та безпеку в єдиному механізмі взаємодії між товаром, постачальником і кінцевим споживачем.

У зв’язку з цим розробка програмного модуля розпізнавання товарів за QR-кодом є досить актуальним дослідженням.

**Метою бакалаврської кваліфікаційної роботи** є розширення функціональних можливостей програмного забезпечення розпізнавання товарів за QR-кодом.

**Об’єктом дослідження** є процес розпізнавання товарів за QR-кодом.

**Предметом дослідження** є програмні засоби розпізнавання товарів за QR-кодом.

Для досягнення поставленої мети необхідно виконати такі задачі:

- обґрунтувати доцільність розробки програмного модуля розпізнавання товарів за QR-кодом;
- проаналізувати переваги та недоліки програм-аналогів для розпізнавання товарів за QR-кодом;
- розробити алгоритм розпізнавання товарів за QR-кодом;
- здійснити програмну реалізацію модуля розпізнавання товарів за QR-кодом;
- виконати тестування програмного модуля та провести аналіз отриманих результатів.

# 1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА QR-КОДОМ

## 1.1 Використання QR-коду для розпізнавання товарів

У 1960-х роках, коли Японія вступила в період високого економічного зростання, у багатьох районах почали з’являтися супермаркети, що продають широкий асортимент товарів – від продуктів харчування до одягу. Касові апарати, які потім використовувалися на касах у цих магазинах, вимагали введення ціни вручну. Через це багато касирів страждали від оніміння зап’ястя та синдрому зап’ястного каналу. «Касири відчайдушно прагнули якимось способом полегшити свій тягар»[1]. Винахід штрих-кодів дозволив вирішити цю проблему.

Згодом була розроблена POS-система, в якій ціна товару автоматично відображалася на касовому апараті при скануванні оптичним датчиком штрихкоду товару, а інформація про товар відправлялася на комп’ютер. Однак у міру поширення використання штрих-кодів стали очевидними й їхні обмеження. Найбільш помітним був той факт, що штрих-код може містити лише 20 буквено-цифрових символів або близько того.

Оскільки QR-код є відкритим кодом, який може використовувати будьхто, він використовується не тільки в Японії, а й у країнах по всьому світу. Оскільки правила його використання були передбачені, а кодекс стандартизовано, його використання поширилося далі. У 1997 році він був затверджений як стандарт AIM для використання в галузі автоматичної ідентифікації. У 1999 році він був затверджений як стандартний 2D-код Японськими промисловими стандартами і внесено стандартний 2D-символ у стандартні форми транзакцій EDI Японської асоціації виробників автомобілів. Більше того, у 2000 році він був затверджений ISO як один із міжнародних стандартів. В даний час використання QR-коду настільки поширене, що без перебільшення можна сказати, що він використовується скрізь у світі.

У той час як використання QR-коду поширилося по всьому світу, нові типи QR-коду для задоволення більш складних потреб створювалися один за одним.

Мікро QR-код був створений для задоволення потреби в менших кодах. Він настільки малий, що його можна надрукувати на невеликій площі, і він став стандартом JIS у 2004 році. У 2008 році був випущений код iQR, який має невелику площу, незважаючи на велику здатність кодування, дозволяє використовувати прямокутні модулі коду [2]. Крім того, тип QR-коду, який реалізує обмеження читання, був розроблений, щоб задовольнити вимоги користувачів щодо підвищеного рівня конфіденційності тощо. «FrameQR» був представлений у 2014 році. FrameQR може покращити дизайн розробленого коду, вільно поєднуючи ілюстрації та фотографії.

QR-код (Quick Response – «швидкий відгук») – піксельний штрих-код у формі квадрата. Від простого штрих-коду відрізняється двомірністю - читається горизонтально та вертикально. QR-код містить більше інформації, ніж штрих-код, наприклад:

- посилання на сайт компанії, окремого продукту або завантаження програми;
- платіжні реквізити швидкої оплати;
- дані для верифікації або доступу до певної інформації (паролі, логіни).

QR-код вигадав Масахіро Хара - інженер Denso Wave Inc., дочірньої компанії Toyota. Ресурсів штрих-кодів не вистачало при бурхливому економічному зростанні Японії та збільшенні кількості товарів у 90-х. У нових кодах інформації зберігалося більше. Перший QR-код з'явився у 1994 році, його використала автомобільна промисловість. Згодом компанія Denso Wave відкрила технологію для всіх. З 2000-х QR-коди визнав світ – технологія забезпечувала прозорість виробництва з відстеженням кожної деталі. QR-код пройшов сертифікацію ISO, що підтверджує, що продукція компанії підтримує міжнародні стандарти.

Щоб генерувати QR-коди потрібно розуміти, як вони працюють. Принцип використання QR-кодів полягає в тому, що роздрукований або намальований код поміщається на об'єкт, після чого він може бути зчитаний і розшифрований за допомогою пристрою, який має функціонуючу камеру і встановлене програмне забезпечення, яке декодує сам QR-код.

Рисунок 1.1 - Приклад QR-коду

Інформація в QR-коді розташовується в двох напрямках - горизонтально і вертикально. Завдяки такому розташуванню даних в QR-коді, він здатний зберігати у багато разів більше інформації, ніж його попередники, включаючи різні типи даних: цифри, букви, ієрогліфи, символи і т.д. Максимальній обсяг інформації різних типів даних, який поміщається в один QR-код, представлено в таблиці 1.1.

Таблиця 1.1 - Типи даних і максимальний можливий обсяг в QR-кодах

| Тип даних | Максимальний обсяг (символ) | Можливі символи |
|---|---|---|
| Числові дані | 7089 | 9,8,7,6,5,4,3,2,1,0 |
| Символьні дані | 4296 | A–Z, $, %, *, +, -, ., /, space, : |
| Бінарна інформація, байт | 2953 | JIS X 0201 |
| Ієрогліфи | 1817 | JIS-X/0208 |

Ще однією перевагою QR-коду є його здатність відновлювати інформацію, що міститься в ньому, навіть якщо певна частина символів на зображенні QR-коду була пошкоджена або не розпізнана. Це стало можливим завдяки системі корекції помилок на базі кодів Ріда-Соломона.

Максимальна кількість кодових слів, яке може бути відновлено, становить до 30%. У відповідності зі специфікацією QR-код має 4 ступеня корекції помилок: L - 7%, M - 15%, Q - 25%, H - 30%. Чим вище ступінь корекції помилок, тим менше даних можна зашифрувати і помістити в QR-код.

Сам QR-код складається з певного набору міток і безпосередньо пікселів, які представляють собою закодоване повідомлення, збережене в QR-коді.

На будь-якому QR-коді обов'язково повинні бути присутніми наступні види міток:

- позиціонування. Область необхідна для детектування коду;

- номер версії. Визначає яка версія коду використовується (від 1 до 40);

- синхронізація. Дублюється в двох напрямках, і дозволяють знизити ймовірність виникнення помилок при зчитуванні, системної інформації (наприклад, версія, тип даних і т.п.);

- формат. Необхідні для визначення типи даних закодованих в коді.

- вирівнювання. Використовуються для кращого позиціонування коду під час обробки (при версії QR-коду вище 1);

- рівень виправлення помилки. Дозволяють визначити, який рівень захищеності від перешкод був використаний на етапі кодування для правильного вибору способу виявлення можливих помилок в коді.

У всіх існуючих програмах, які зчитують і декодують QR-коди, реалізований простий алгоритм виявлення QR-коду на зображенні, отриманому з камери. Потім реалізована стандартна процедура декодування інформації з QR-коду. Однак цей алгоритм розпізнавання вимагає дуже чіткого позиціонування спеціально виділеної області на пристрої, за допомогою якого відбувається розпізнання і визначеного розташування QR-коду в просторі. Після того, як спеціально визначена область на пристрої чітко збіглася з границями QR-коду, відбувається пошук трьох міток позиціонування, про які йшлося раніше.

Ці мітки розташовані строго в певних місцях виділеної області зображення. Недолік такого підходу полягає в тому, що QR-код може перебувати в будь-якій області на зображенні, і користувачеві необхідно самому попередньо фокусуватися на необхідному QR-коді, і стежити за тим, щоб область для зйомки QR-коду збігалася з самим QR-кодом, який потрібно розкодувати (рис. 1.2).


Рисунок 1.2 - Мітки та дані на QR-коді

QR код - це монохромна картинка, на якій деякі пристрої (наприклад смартфон зі спеціальним додатком) розпізнають текст. Цим текстом може бути не тільки проста фраза, але і, хоч це і не входить в офіційну специфікацію, посилання, номер телефону або візитна картка. Такі коди найчастіше використовують, щоб закодувати посилання і роздрукувати її на плакаті або візитці.

Процес генерації QR коду ділиться на кілька чітких кроків:

- кодування даних;
- додавання службової інформації та заповнення;
- поділ інформації на блоки;
- створення байтів корекції;
- об'єднання блоків;
- розміщення інформації на QR коді.

Кодування даних. QR код підтримує кілька способів кодування даних, в залежності від того, які символи використовуються: цифрове, буквено-цифрове, кандзі (китайсько-японські ієрогліфи) і побайтово кодування. Цифрове кодування має на увазі використання тільки цифр від 0 до 9, буквено цифрове - прописні букви

латинського алфавіту, цифри і символи $% * + -. /: І пробіл. Спочатку треба створити порожню послідовність біт, яка далі буде заповнюватися.

Цифрове кодування. Цей тип кодування вимагає 10 біт на 3 символи. Вся послідовність символів розбивається на групи по 3 цифри, і кожна група (тризначне число) переводиться в 10-бітове двійкове число і додається до послідовності біт. Якщо загальна кількість символів не кратне 3, то якщо в кінці залишається 2 символи, отримане двозначне число кодується 7 битами, а якщо 1 символ, то 4 бітами. Наприклад, є рядок «12345678», який треба закодувати. Ми розбиваємо її на числа: 123, 456 і 78, потім переводимо кожне з них в двійковий вигляд: 0001111011, 0111001000 і 1001110, і об'єднуємо це в один потік: 000111101101110010001001110.

Буквено-цифрове кодування. У цьому випадку на 2 символи потрібно 11 біт інформації. Вхідний потік символів розділяється на групи по 2, в групі кожен символ кодується згідно таблиці внизу, значення першого символу в групі множиться на 45 і додається до значення другого символу. Отримане число переводиться в 11-бітове двійкове число і додається до послідовності біт. Якщо в останній групі 1 символ, то його значення відразу кодується 6-бітовим числом і додається до послідовності біт.

Побайтове кодування. Це універсальний спосіб кодування, яким можна закодувати будь-які символи. Єдиним недоліком методу є відносно низька щільність інформації. В цьому випадку текст кодується в будь-якому кодуванні (рекомендований в UTF-8) і отримана сукупність електронних даних береться в незмінному вигляді (табл. 1.2).

Додавання службової інформації є важливим етапом при створенні QR-коду, оскільки саме на цьому етапі визначаються параметри, що впливають на надійність зчитування коду в умовах пошкодження, спотворення чи обмеженого освітлення. Одним із ключових параметрів є рівень корекції помилок. Цей параметр визначає здатність QR-коду залишатися придатним для зчитування навіть за наявності часткових пошкоджень — наприклад, подряпин, забруднень, розмиття або перекриття частини коду графічним елементом (як це часто трапляється в дизайнерських реалізаціях).

Таблиця 1.2 - Значення символів в буквено-цифровому кодуванні.

| Значення | Символ | Значення | Символ | Значення | Символ | Значення | Символ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 12 | C | 24 | O | 36 | Пробіл |
| 1 | 1 | 13 | D | 25 | P | 37 | $ |
| 2 | 2 | 14 | E | 26 | Q | 38 | % |
| 3 | 3 | 15 | F | 27 | R | 39 | * |
| 4 | 4 | 16 | G | 28 | S | 40 | + |
| 5 | 5 | 17 | H | 29 | T | 41 | - |
| 6 | 6 | 18 | I | 30 | U | 42 | . |
| 7 | 7 | 19 | J | 31 | V | 43 | / |
| 8 | 8 | 20 | K | 32 | W | 44 | : |
| 9 | 9 | 21 | L | 33 | X | | |
| 10 | A | 22 | M | 34 | Y | | |
| 11 | B | 23 | N | 35 | Z | | |

Існує чотири рівні корекції помилок, закладені в стандарт QR-кодів:

- L (Low) — забезпечує відновлення до 7% даних. Підходить для ідеальних або майже ідеальних умов друку і зчитування;

- M (Medium) — дозволяє відновити до 15% пошкоджених даних. Це найпоширеніший варіант, оскільки забезпечує баланс між розміром QR-коду і його надійністю;

- Q (Quartile) — дає змогу відновити до 25% інформації. Рекомендується для середовищ з підвищеним ризиком часткового пошкодження або складних умов експлуатації;

- H (High) — найвищий рівень корекції, дозволяє зчитати код навіть при пошкодженні до 30% площі. Часто використовується у випадках, коли поверхня з кодом може бути сильно пошкоджена або QR-код інтегрується у фірмовий стиль із перекриттям логотипом.

Слід враховувати, що чим вищий рівень корекції, тим більший обсяг службової інформації необхідно вбудовувати у QR-код. Це, своєю чергою, зменшує кількість корисної інформації, яку можна закодувати при незмінному фізичному

розмірі зображення, або потребує збільшення самого QR-коду для збереження того ж обсягу даних.

Ще одним важливим параметром QR-коду є його версія, яка прямо впливає на його розміри та інформаційну місткість. Всього існує 40 версій QR-кодів:

Версія 1 — це найменший QR-код розміром 21×21 модуль (де модуль — це найменший "піксель" коду).

Кожна наступна версія збільшує розмір коду на 4 модулі з кожного боку. Таким чином, версія 40 має розмір 177×177 модулів.

Вибір версії залежить від трьох основних чинників:

- обсягу даних, що потрібно зашифрувати (текст, URL, контактна інформація, геодані тощо);

- типу даних (цифрові, буквено-цифрові, байтові, ієрогліфи);

- обраного рівня корекції помилок: чим вищий рівень, тим більше службових блоків, і тим більша потрібна версія.

Наприклад, QR-код, який кодує короткий URL з низьким рівнем корекції, може бути реалізований уже у версії 2 або 3. Натомість для зашифрування тексту на кілька сотень символів із високим рівнем корекції (H) може знадобитися версія 20 або навіть більше.

Розуміння ролі службової інформації, рівня корекції та версії QR-коду є критично важливим для оптимального проєктування QR-кодів у реальних застосуваннях, де важлива як надійність зчитування, так і розмір/розміщення коду на упаковці, пристрої чи рекламному матеріалі.

### 1.2 Аналіз відмінностей QR-коду та штрих-коду

QR-код - це прямий спадкоємець штрих-коду. Ось тільки в основу другого лягла технологія азбуки Морзе, що використовувалася для автоматизації різного товару і техніки. І десятиліттями штрих-код був єдиним нормальним варіантом маркування. Звичні смуги і цифри вже давно стали загальноприйнятим явищем для будь-якого сучасника. Однак можливості штрих-коду обмежені.

Ключова відмінність QR-коду від традиційного штрихкоду - він розпізнається сканером як двовимірне зображення. Для нормалізації зображення при зчитуванні і зниження ймовірності помилки код містить кілька великих квадратів в одному з кутів, а також безліч дрібніших синхронізуючих точок, розосереджених по всій площі коду. Кумедний момент: специфікація QR-коду описує тільки сам принцип побудови коду, але не формат даних, зашифрованих в ньому. Це створює ціле поле для експериментів, які не закінчуються до цього дня.

Лінійний код може вмістити в себе від 20-ти до 30-ти символів, чого часом недостатньо. Японські фахівці поставили перед собою мету - розширити можливості штрих-коду, але з класичним підходом це було неможливо. І на арені з'являються двомірні (матричні) коди, серед яких головним по праву став QR-код.

Традиційні штрих-коди, які ми часто скануємо в супермаркетах, щоб дізнатися ціну продукту, обмежені тільки 20 буквено-цифровими символами по горизонталі. Вони грунтуються на американському стандарті відстеження товарів UPC (Universal Product Code = універсальний код товару), розробленому в 1973 році (рис. 1.3).


Рисунок 1.3 – Приклад використання штрихкоду

Його недоліки: якщо штрих-код пошкоджений, інформація недоступна. А також він не може шифрувати ієрогліфи.

На відміну від звичайного штрих-коду QR-код має низку позитивних якостей:

- збільшення обсягу закодованої інформації в кілька разів;

- інформація не дублюється символами, зрозумілими людині;

- на вибір є кілька варіантів виконання.

Фактично складно назвати QR-код чимось концептуально новим. Все-таки технологія вкрай близька до класичного штрих-коду. Однак різниця є. QR-код - це свого роду сполучна ланка між реальністю і віртуальним світом, як би дивно це не звучало. Можливості, які відкрили QR-коди, дійсно набагато ширше, ніж були раніше. Будь-який сучасний телефон або планшет може без проблем зчитувати інформацію з QR-коду за частки секунди. І інформація ця може бути найрізноманітнішою: дані про продукцію, посилання на офіційний сайт.

### 1.3 Характеристика та аналіз програм аналогів сканерів QR-коду

«QR Code – сканер QR кода» може зчитувати всі типи QR-кодів включаючи: текст, URL, ISBN, продукт, контакти, події календаря, адреси та повідомлення електронної пошти, місце розташування, Wi-Fi і багато інших форматів.

Переваги: «QR Code – сканер QR кода» має дуже зручний та простий інтерфейс, що дозволить швидко освоїти функціонал даного продукту.

Недоліки: «QR Code – сканер QR кода» використовується лише для зчитування інформації та не вносить ніякі зміни до параметрів товару, що є зручно для отримання інформації, та не може бути використано для автоматизації роздрібної торгівлі; за відсутності інформації про товар на запропонованому сайті, сканер втрачає свою користь.

Пристрій можна використовувати як сканер QR-кодів для отримання додаткової інформації про об’єкти, що мають такий код. Наприклад, ви побачили пальто в рекламному оголошенні в журналі й бажаєте дізнатися про найближчу точку продажу. Якщо оголошення містить QR-код, програма Сканер qr кода - QR code scanner використовує його, щоб отримати доступ до веб-вмісту, наприклад веб-сторінки з додатковою інформацією або карти найближчих точок продажу. Програма NeoReader підтримує більшість стандартних типів штрих-кодів (рис. 1.4).


Рисунок 1.4 - Інтерфейс програми QR Code

Переваги: «Сканер qr кода - QR code scanner» має дуже зручний та простий інтерфейс, що дозволить швидко освоїти функціонал даного продукту.

Недоліки: Неможливе зчитування інформації з QRкоду з подальшим опрацюванням. Не проводиться обробка зчитаної інформації (рис. 1.5).

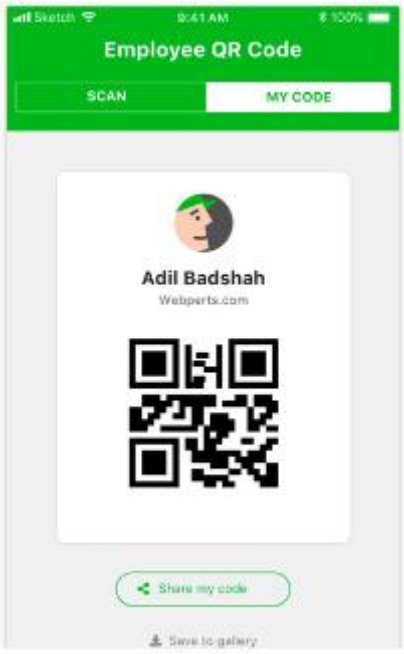


Рисунок 1.5 - Інтерфейс програми Сканер qr кода - QR code scanner

Додаток для сканування QR-кодів — це зручний мобільний застосунок, розроблений для швидкого та легкого сканування й розшифрування QR-кодів за допомогою камери смартфона. Завдяки сучасному та інтуїтивно зрозумілому інтерфейсу користувачі можуть миттєво отримувати доступ до закодованої інформації, такої як веб-посилання, контактні дані або відомості про товари. У додатку передбачено журнал історії, який зберігає раніше відскановані коди для зручного повторного доступу, а також функцію створення й налаштування власних QR-кодів — як для особистого, так і для бізнес-використання.

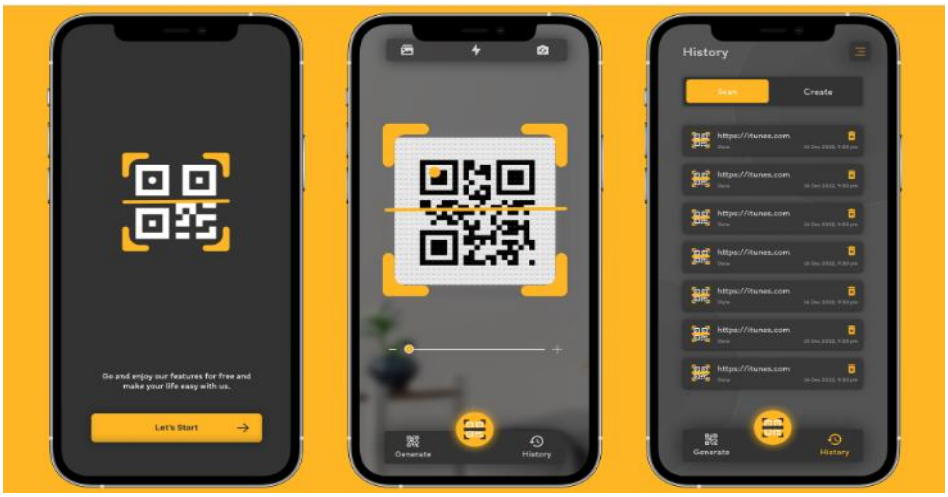


Рисунок 1.6 - Інтерфейс програми Сканер qr кода - QR code scanner

### 1.4 Висновок до розділу 1

Отже, в наш час QR-коди можна побачити практично скрізь, тому що багато компаній і користувачів вже змогли переконатися в їх ефективності та універсальності. Легко пізнавані чорно-білі квадрати допомагають людям вирішити різні завдання в багатьох сферах життєдіяльності.

В даному розділі обгрунтовано використання QR-кодів для розпізнвання товарів. Проаналізовано основні відмінності штрихкодів від QR-кодів. Проаналізовано переваги та недоліки програм аналогів для сканування QR-кодів та розпізнавання їх вмісту.

# 2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА QR-КОДОМ

## 2.1 Обгрунтування вибору методу підготовки інформації для кодування товарними кодами

Перед кодуванням інформацію потрібно відповідним чином підготувати, при цьому інформація має бути структурована і мінімізована. Для цього найкраще підходить формат JSON.

JSON (JavaScript Object Notation) — це легкий формат обміну даними. Людині легко читати і писати. Машинам легко аналізувати та генерувати. Він заснований на підмножині стандарту мови програмування JavaScript ECMA-262 3-е видання - грудень 1999 року. JSON - це текстовий формат, який повністю не залежить від мови, але використовує угоди, знайомі програмістам сімейства мов C, включаючи C, C++, C#, Java, JavaScript, Perl, Python та багато інших. Ці властивості роблять JSON ідеальною мовою обміну даними.

JSON побудований на двох структурах:

- колекція пар ім’я/значення. У різних мовах це реалізується як об’єкт , запис, структура, словник, хеш-таблиця, список із ключами або асоціативний масив;

- упорядкований список значень. У більшості мов це реалізується як масив , вектор, список або послідовність.

Це універсальні структури даних. Практично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Має сенс, щоб формат даних, взаємозамінний з мовами програмування, також був заснований на цих структурах.

У JSON вони набувають наступних форм.

Об’єкт – це неупорядкований набір пар ім’я/значення. Об'єкт починається з { лівої дужки і закінчується } правою дужкою. За кожною назвою йде двокрапка, а пари ім’я/значення відокремлюються комою (рис. 2.1) .

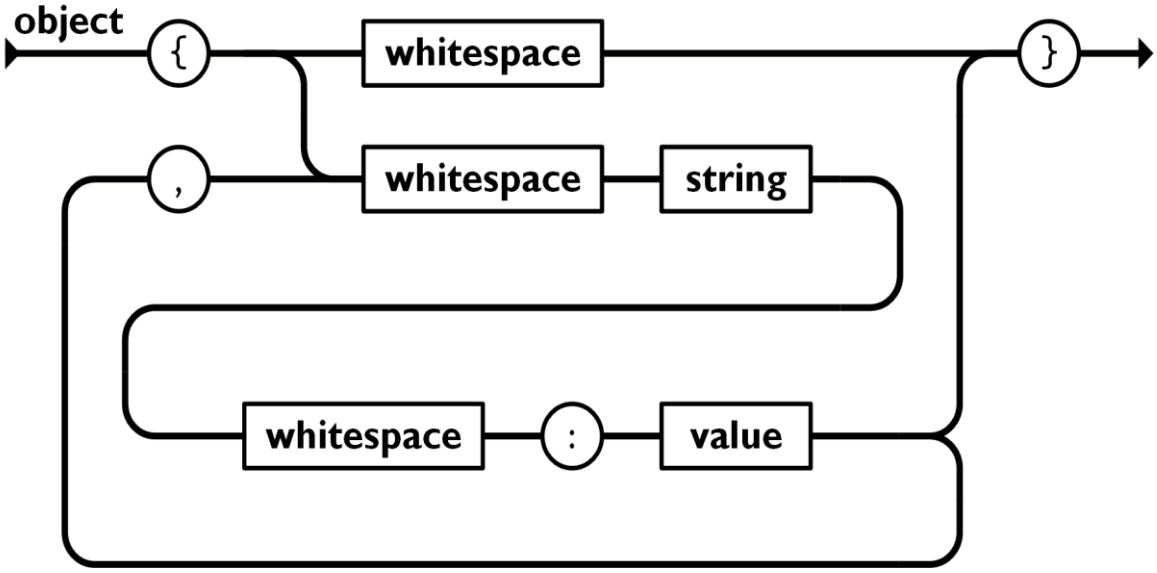


Рисунок 2.1 - Структура об’єкту в JSON

Масив — це впорядкована колекція значень, які зберігаються у певній послідовності та доступні за індексом. Масиви використовуються для організації даних, що мають однаковий тип або логічно пов’язані між собою, наприклад: список чисел, перелік рядків, об'єкти тощо. У багатьох мовах програмування, зокрема в JavaScript, Python чи JSON-структурах, масиви мають однакову синтаксичну форму.

Масив починається з лівої дужки [ і закінчується правою дужкою ]. Значення (value) відокремлюються комою (рис. 2.2).

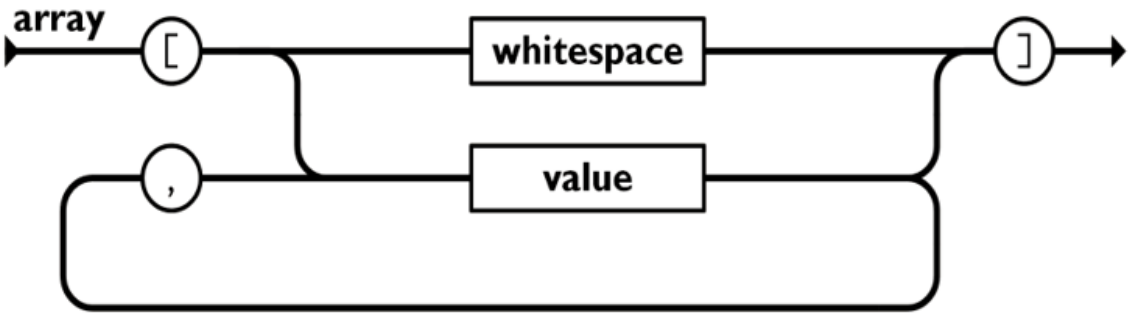


Рисунок 2.2 - Структура масиву в JSON

Значенням може бути рядки, числа, масиви, логічні значення та інші об’єктні літерали. Ці конструкції можуть бути вкладеними (рис. 2.3).

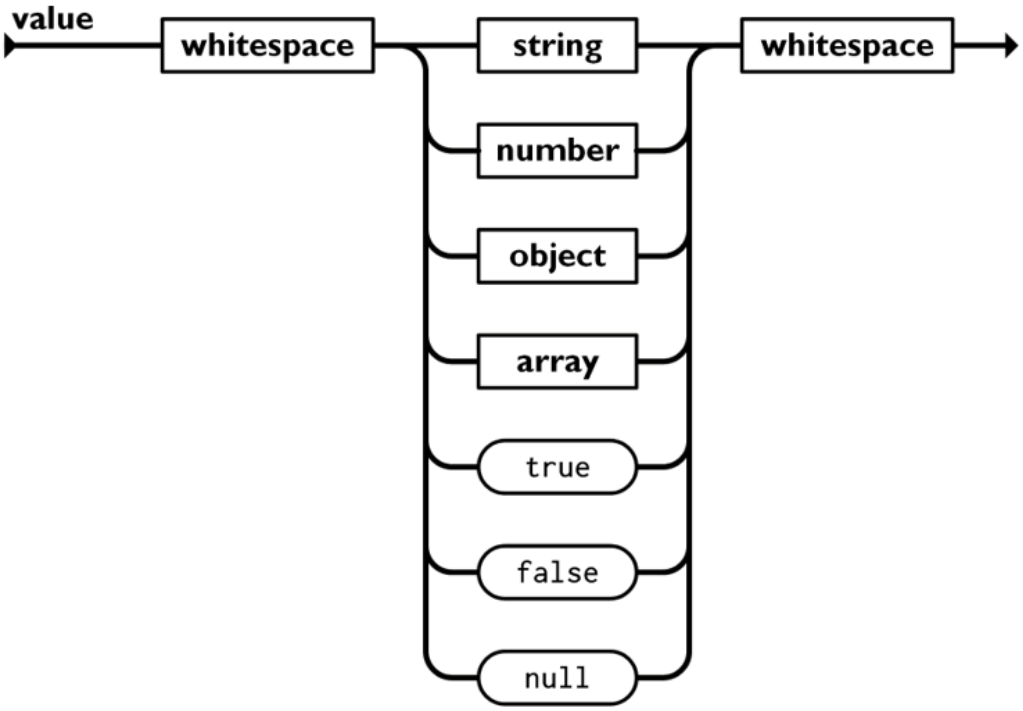


Рисунок 2.3 - Структура значення в JSON

## 2.2 Створення стурктурної схеми кодування/декодування товару за QR-коду

Для того щоб коректно кодувати да декодувати потрібну нам інформацію ми маємо надати їй відповідну структуру (рис. 2.4).

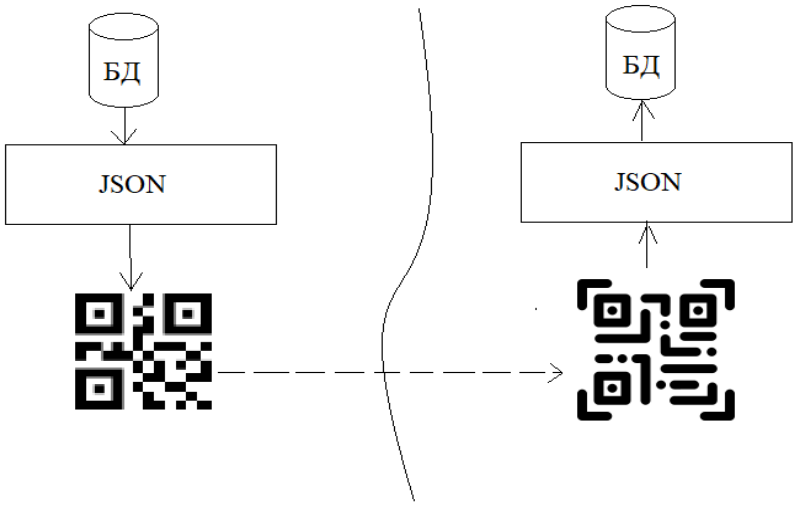


Рисунок 2.4 - Схема кодування і декодування товару за QR-кодом

Кодування (serialization) — це процес перетворення об’єкта товару (тобто структурованих даних, які містять властивості товару) у формат JSON-рядка. Цей рядок є текстовим представленням даних, що зберігає структуру об’єкта у вигляді пар "ключ–значення", придатну для зберігання, передавання мережею або обміну між різними системами, такими як сервер, мобільний застосунок або база даних.

Кодування є особливо актуальним у сферах, де необхідна автоматизована передача даних - наприклад, в електронній комерції, CRM-системах, маркетплейсах, інвентаризаційних платформах або мобільних додатках. JSON (JavaScript Object Notation) підтримується більшістю мов програмування та забезпечує легкість інтеграції, гнучкість і читабельність як для людини, так і для машини.

Для опису товару нам потрібні наступні характеристики:

- найменування;
- одиниця виміру;
- штрих-код;
- вага;
- об’єм;
- термін придатності.

У JSON форматі це буде виглядати наступним чином

```
[
{
“name”: “Йогурт”,
“unit”: “штук”,
“barcode”: 9876543210,
 “weight”: 0.320,
“volume”: 0.3,
“expiration”: 15
}
]
```

Такий формат легко надіслати через мережу (наприклад, у тілі HTTP-запиту), зберегти у файл чи базу даних, або використати для генерації QR-коду з усіма необхідними властивостями товару. У мовах програмування, таких як Python, JavaScript, Java або C#, для кодування в JSON використовуються вбудовані бібліотеки, наприклад:

- Python: json.dumps(об’єкт)
- JavaScript: JSON.stringify(об’єкт)
- Java: Gson.toJson(об’єкт) (через бібліотеку Gson)
- C#: JsonSerializer.Serialize(об’єкт)

Кодування у JSON дозволяє стандартизувати дані товарів, забезпечити сумісність між системами та автоматизувати бізнес-процеси в цифровому середовищі.

### 2.3 Опис існуючих режимів аналізу даних двовимірного QR-коду

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 2.5 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.

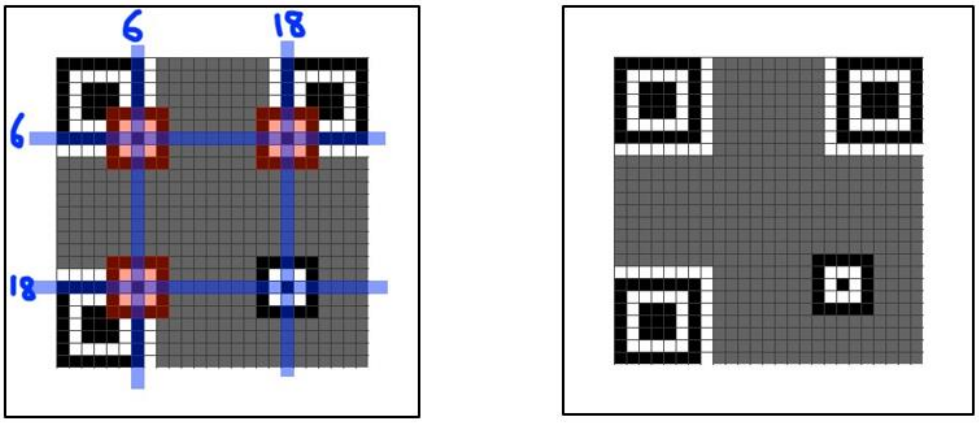


Рисунок 2.5 – Некоректне та правильне розміщення шаблонів вирівнювання в матриці QR-коду

Для подальшої ілюстрації розміщення шаблону вирівнювання, на рисунку 2.6 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.

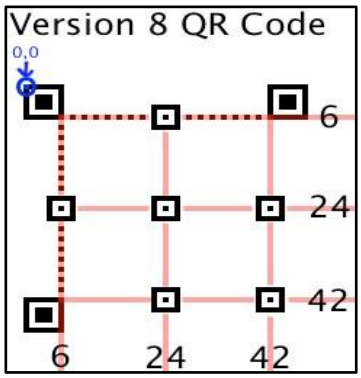


Рисунок 2.6 – Демонстрація розміщення коректного шаблону вирівнювання на QR коді версії 8

Чотири режими кодування включають такі символи:

- числовий режим призначений для десяткових цифр від 0 до 9;

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

- 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;

- 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].

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 буквеноцифровому 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.

## 2.4 Розробка клієнт-серверної архітектури програмного модуля розпізнавання товарів за QR-кодом

Усі веб-сервіси в Інтернеті використовують архітектуру клієнт-сервер. Вихідні файли програми зберігаються на сервері, який обробляє запити, що надходять через інтернет від кількох користувачів.

На стороні клієнта веб-додаток працює у браузері, наприклад Google Chrome або Mozilla Firefox. Інтерфейс користувача програми надсилається на клієнтський комп’ютер у вигляді сторінок, написаних на HTML (Hypertext Markup Language), де браузер інтерпретує та відображає їх. Клієнтські комп’ютери забезпечують інтерфейс, який дозволяє користувачеві комп’ютера запитувати сервери та

переглядати результати, повернуті сервером. Веб-сервери чекають запитів від клієнтів, а потім відповідають на них. Обмін даними здійснюється на апаратному рівні, на основі протоколу TCP/IP і HTTP або захищеного протоколу HTTPS вищого логічного рівня [4]. Хоча протокол HTTP був розроблений у 1990 році, він швидко розвивається. Існує дві версії протоколу, основна відмінність полягала в тому, що для завантаження сторінки можна було підключити лише один сервер, що дало величезний приріст продуктивності. Мультиплексування - дозволяє надсилати кілька запитів одночасно через одне з'єднання. Раніше, використовуючи HTTP / 1.1, доводилося чекати наступних передач, щоб кожен новий запит був виконаний. Пріоритет – запитам призначаються рівні залежності, які сервер може використовувати для швидшого надання ресурсів з вищим пріоритетом.

Клієнти часто знаходяться на персональних комп’ютерах, а сервери – в інших місцях мережі, як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти і сервер мають окремі завдання, які вони виконують за замовчуванням (рис. 2.7).

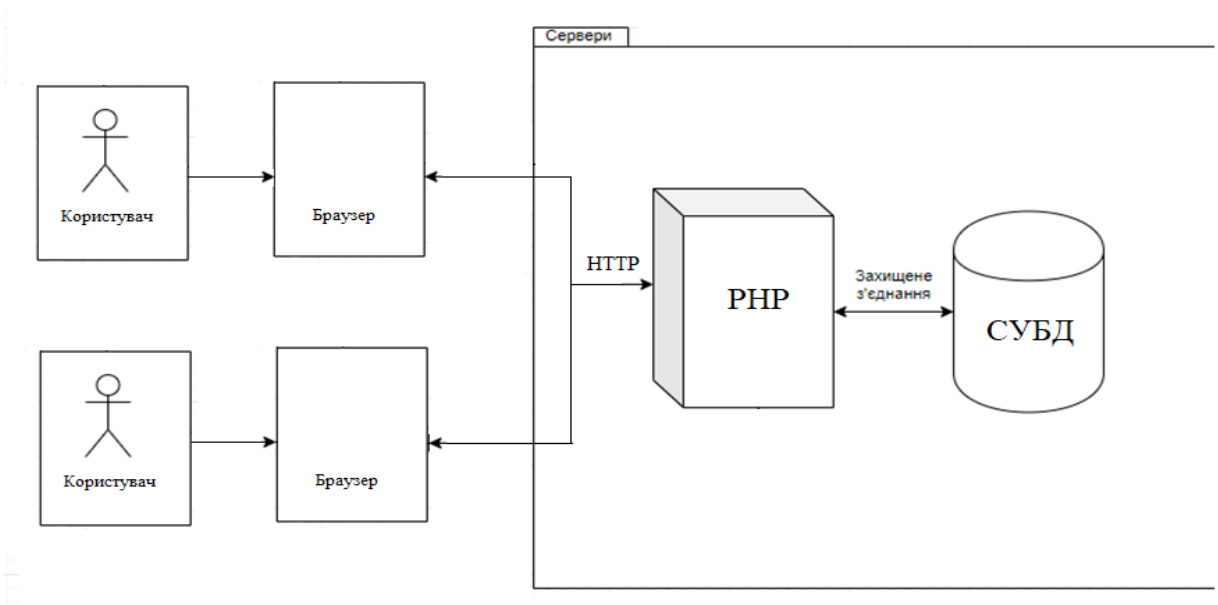


Рисунок 2.7 - Схема руху інформації між користувачами і СУБД

Ця обчислювальна модель передбачає розмежування обов’язків: клієнт виступає як ініціатор запитів, тобто взаємодіє з користувачем, збирає або передає

дані, тоді як сервер виконує обробку запитів, зберігання даних, автентифікацію користувачів, обслуговування баз даних або обчислення складної логіки. Наприклад, коли користувач відкриває вебсайт інтернет-магазину, його пристрій (клієнт) надсилає запит на сервер, який повертає HTML-сторінку з товаром, завантажує зображення, ціни, інформацію про наявність тощо.

Ця модель є особливо ефективною з кількох причин:

- розподіл навантаження: клієнт обробляє лише інтерфейс та взаємодію з користувачем, а сервер концентрується на зберіганні й обробці даних;

- масштабованість: сервери можуть бути централізовано оновлені, модернізовані або дубльовані для обслуговування великої кількості клієнтів;

- безпека та контроль: критичні дані та логіка зберігаються на стороні сервера, що дозволяє захистити інформацію від несанкціонованого доступу;

- централізоване управління: адміністратор системи може оновлювати серверну частину без необхідності змін на клієнтських пристроях.

У реальному світі приклади клієнт-серверної архітектури зустрічаються всюди: банківські додатки, електронна пошта, соціальні мережі, онлайн-магазини, медичні системи, CRM-платформи. Навіть простий перегляд веб-сторінки в браузері — це взаємодія клієнта з веб-сервером.

Модель "клієнт-сервер" є основоположною концепцією сучасних розподілених систем і забезпечує ефективну, масштабовану та гнучку взаємодію між користувачем і обчислювальними ресурсами.

## 2.5 Розробка схеми алгоритму роботи програмного модуля розпізнавання товарів за QR-кодом

Розпізнавання товарів за QR-кодом – це сучасний технологічний процес, який використовує спеціальні програми або мобільні додатки, що за допомогою камери пристрою зчитують невеликий квадратний код, розміщений на упаковці товару. Цей QR-код містить закодовану інформацію, яка може включати найважливіші дані про

товар: його назву, ціну, дату виготовлення, термін придатності, серійний номер, виробника та інші характеристики.

Коли користувач наводить камеру свого смартфона або спеціального сканера на QR-код, програма миттєво розпізнає цей код, виконує його декодування і відображає всю отриману інформацію на екрані. Це дає змогу швидко і точно отримати деталі про товар без необхідності шукати інформацію вручну або вводити її вручну, що знижує ймовірність помилок і значно економить час.

Впровадження технології розпізнавання за QR-кодами має важливі переваги для бізнесу і споживачів. Для продавців це означає автоматизацію процесу обліку товарів, спрощення інвентаризації і покращення контролю за наявністю товарів на складах. Для покупців — це зручний спосіб миттєво перевірити інформацію про продукт, його справжність, дату виготовлення та інші важливі параметри. Такий підхід також сприяє підвищенню прозорості та довіри у торгових відносинах (рис. 2.8).

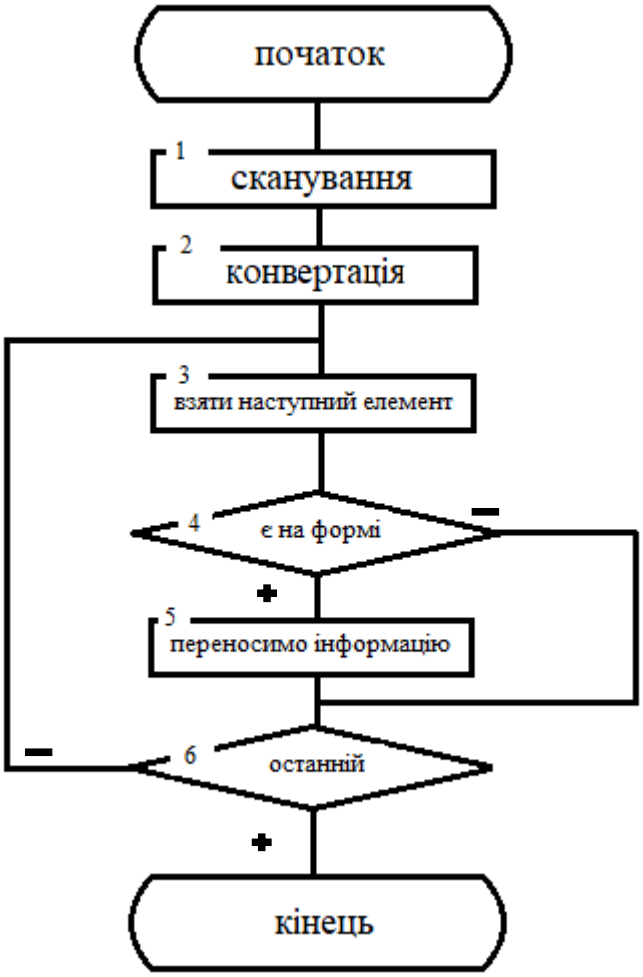

Рисунок 2.8 - Загальний алгоритм функціонування системи розпізнавання товарів за QR-кодом

Крім того, розпізнавання товарів за QR-кодом підтримує інтеграцію з різноманітними системами управління – касовими апаратами, CRM та ERP, що забезпечує ефективне управління продажами і обліком у режимі реального часу. Ця технологія також активно використовується для впровадження безконтактних розрахунків і інноваційних маркетингових рішень.

Розпізнавання товарів за QR-кодом – це не просто інструмент швидкого зчитування інформації, а комплексне рішення, яке значно підвищує ефективність торгових процесів, покращує сервіс і робить взаємодію з товарами максимально комфортною і безпечною для всіх учасників ринку.

Алгоритм роботи програмного модуля розпізнавання товарів за QR-кодом складається з таких кроків.

Крок 1. На першому етапі відбувається сканування QR-коду.

Крок 2. Програма конвертує прийнятий QR-код у масив елементів, що описують параметри товару.

Крок 3. Беремо черговий елемент масиву.

Крок 4. Перевіряємо чи даний параметр відображений на формі. Якщо елемент присутній на формі, то переходимо до кроку 5. Інакше, переходимо до кроку 6.

Крок 5. Переносимо отриману інформацію з елементу масиву в об’єкт на формі.

Крок 6. Перевіряємо чи це не останній елемент, якщо елемент останній, то переходим до кроку 7. Інакше, переходимо до кроку 3.

Крок 7. Закінчення процесу сканування.

## 2.6 Розробка діаграми прецедентів програмного модуля розпізнавання товарів за QR-кодом

Основним функціоналом програмного модуля є дешифратор та шифратор двовимірних зображень.

Система містить також альтернативний потік, в залежності від того чи користувач зареєстрований чи ні. При реєстрації є додатковий функціонал програмного модуля, який відображає профіль користувача, його особисті дані, а також можливість їх змінити (рис. 2.9).

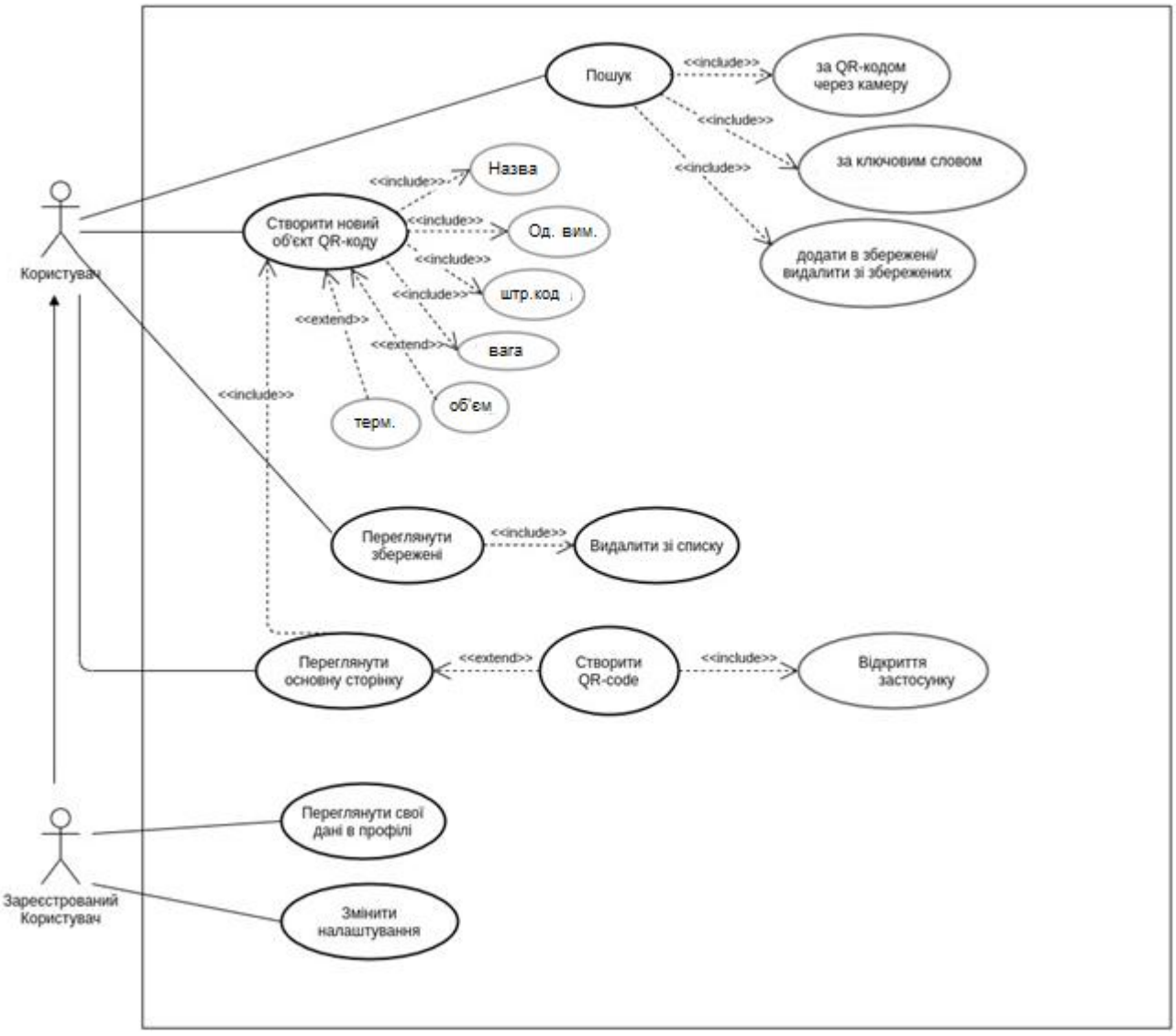


Рисунок 2.9 – Діаграма прецедентів програмного модуля розпізнавання товарів за QR-кодом

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 редагувати дані. Проте саме альтернативний потік приносить найбільшу практичну користь, коли очікуваний та цільовий результат не досягнуто.

Опис діаграми користувацьких потоків

Під час сканування двовимірного зображення безпосередньо визначається тип коду: це є QR-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 (рис. 2.10).

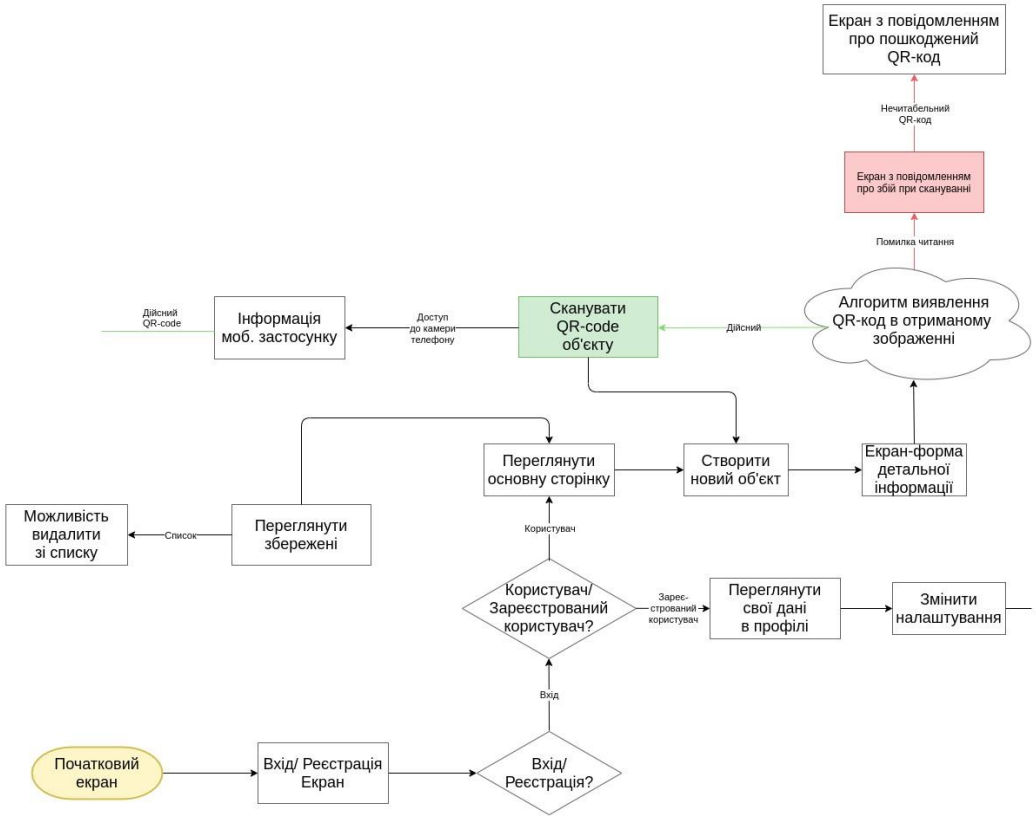


Рисунок 2.10 – Діаграма користувацьких потоків (user flow)

Написання коду і реалізація моделі програмного модуля найкраще зображена на діаграмі класів (рис. 2.11). Додавання нового об’єкту розпізнаного товару разом з двовимірним зображенням передбачає етапи заповнення даними користувача та

створення на основі цих даних QR-коду. Після цього створений QR-код відображається на головному екрані програми, а введені дані записано в БД, де також є записи про кожний створений QR-код.

Зареєстрований користувач має додатковий функціонал - зберігання обраних товарів в БД, та запис особистих даних в профілі. Список збережених містить інформацію про товари.

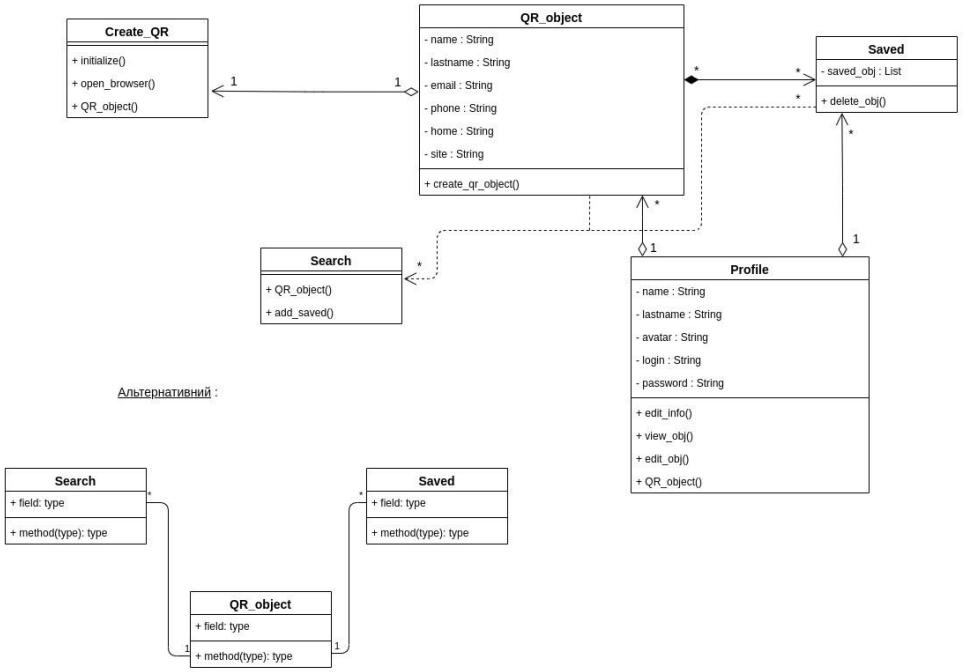


Рисунок 2.11 - Діаграма класів програмного модуля

У ншому випадку всі 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. Для того, щоб реалізувати інтерфейс, використовується ключове слово implements.

Для зберігання даних нам знадобляться моделі БД. 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/, і є обов’язковим для ініціалізації усіх моделей.

Клас Model 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 товари. Для цього створюється таблиця з назвою Card, яка визначає дані, які можна зберігати. Кожна картка містить ідентифікатор, назву, одиницю виміру, штрих-код, вагу, об’єм, термін придатності. 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 товару, 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].

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

- Ідентифікатор —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;

- Товар —зберігається як символьний тип даних.

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

Операції з базою даних, CRUD операції, передбачають наступні кроки взаємодії основних модулів програми (рис. 2.12):

- 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.

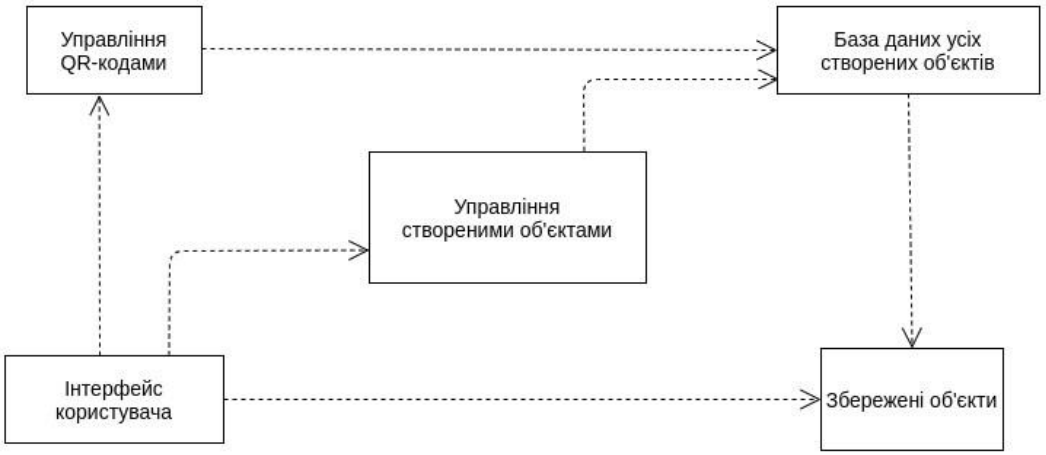


Рисунок 2.12 – Діаграма розгортання програмного модуля

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.

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.

Отже, спочатку 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 ().

## 2.7 Висновок до розділу 2

В даному розділі було обгрунтовано вибір методу підготовки інформації про товари для кодування товарними кодами. Створено структурну схему кодування/декодування товару за QR-коду. Проведено опис існуючих режимів аналізу даних двовимірного QR-коду. Розроблено алгоритм роботи програмного модуля розпізнавання товарів за QR-кодом. Розроблено UML-діаграми програмного модуля розпізнавання товарів за QR-кодом.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА QR-КОДОМ

## 3.1 Обґрунтування вибору мови програмування та середовища розробки

Технології веб-розробок швидко розвивається, і розробники серверів стикаються з вибором між потужними мовами, такими як Java, C, Perl, і сучасними веб-мовами, такими як PHP, JavaScript, Ruby, Go.

PHP - був створений в 1994 році Расмусом Лерфордом. Він створив програмну оболонку (інтерпретатор), яка встановлюється як модуль для веб-сервера Apache або Nginx. Спочатку розроблений як препроцесор гіпертекстової сторінки, PHP можна легко інтегрувати в HTML, однак цей підхід сьогодні не є гарною практикою, але був очевидним для початківців. Це сприяло популярності мови, тому 80% веб-сайтів в Інтернеті написані на PHP [9].

Програма, яка перекладає код, написаний однією мовою програмування, на іншу, називається транслятором. Упорядник також є перекладачем. Перекладає код, написаний мовою високого рівня, у машинний код. Процес компіляції створює двійковий виконуваний файл, який можна запустити без компілятора. Перекладач – це зовсім інша категорія. Інтерпретатор не перекладає в код, а виконує його. Інтерпретатор аналізує програмний код і виконує кожен його рядок. Ви повинні використовувати інтерпретатор кожного разу, коли ви запускаєте цей код. Інтерпретатори значно поступаються компіляторам за продуктивністю, оскільки двійковий код виконується набагато швидше. Але інтерпретатори дозволяють повністю контролювати програму під час її виконання . Що стосується PHP, то він не є ні компілятором, ні інтерпретатором. PHP є чимось середнім між компілятором та інтерпретатором. Сценарій надсилається на вхід PHP. Він перекладає його, перевіряючи синтаксис, у спеціальний байт-код (внутрішнє представлення). Тоді PHP виконує байт-код, а не код самої програми, і не створює виконуваний файл. Байт-код набагато компактніший, ніж звичайний програмний код, що робить його легшим і швидшим для інтерпретації або виконання.

Переваги PHP:

- працює на всіх найпопулярніших операційних системах (Windows, MacOS, Linux), має власні версії пакетів розробки на PHP, а це означає, що ви можете створювати сайти на будь-якій з цих операційних систем;

- PHP може працювати в поєднанні з різними веб-серверами: Apache, Nginx, IIS;

- простота і легкість створення. Загалом, маючи невеликий досвід програмування на PHP, ви можете створювати прості веб-сторінки;

- Синтаксис PHP схожий на синтаксис C, тому, знаючи C або одну з мов з подібним синтаксисом, буде легше освоїти PHP;

- PHP підтримує роботу з великою кількістю систем баз даних MySQL, MSSQL, Oracle, Postgre, MariaDB, MongoDB тощо;

- постійний розвиток PHP, з'являються нові версії з новими функціями, адаптація мови програмування до нових завдань. І, як правило, перейти на нову версію не складно.

На даний момент веб-сторінки створюються у двох версіях PHP: старішій версії 7 і відносно новій версії 8 [9].

Основні відмінності в інноваціях:

- рефакторинг базових структур даних;

- покращена конвенція для виклику віртуальних машин;

- новий API параметрів розбору;

- новий менеджер пам'яті;

- численні вдосконалення виробників віртуальних машин;

- значно зменшено споживання пам'яті;

- покращені функції __call () і __callStatic ();

- покращене з'єднання струн;

- покращений пошук символів у рядках.

JavaScript (JS) — динамічна, об'єктно-орієнтована[5] прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої

кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв вебсторінок для надання їм інтерактивності;

- створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js);

- програмування на боці сервера (Node.js, Express.js );

- стаціонарних застосунків (Electron, NW.js);

- мобільних застосунків (React Native, Cordova);

- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);

- всередині PDF-документів тощо.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має кілька властивостей, притаманних функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою С має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;

- функції як об'єкти першого класу;

- обробка винятків;

- автоматичне приведення типів;

- автоматичне збирання сміття;

- анонімні та стрілочні функції

JavaScript містить декілька десятків вбудованих об'єктів, які поділяються на групи: фундаментальні (Object, Function, Boolean, Symbol), помилки (група об'єктів Error), числа та дати (Number, BigInt, Math Date), текстові (String, RegExp), індексовані (група об'єктів Array), ключові (Map, Set, WeakMap, WeakSet), для роботи з структурованими даними (ArrayBuffer, Atomics, DataView, JSON), абстрактні (Promise, Generator), рефлекційні (Reflect, Proxy), групи Intl та WebAssembly. Крім того, JavaScript містить набір вбудованих операцій, що керують логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений у порівнянні з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, наприклад, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може знаходитися в тексті програми після неї.

## 3.2 Опис розробки модуля сканування та генерації QR-коду

Інтерфейс розробленої програми являє собою односторінковий додаток.

Односторінковий додаток (SPA) – це веб-додаток або веб-сайт, який взаємодіє з користувачем шляхом динамічного переписування поточної веб-сторінки з новими даними з веб-сервера замість стандартного методу веб-браузера, який завантажує цілі нові сторінки. Метою є швидші переходи, завдяки чому веб-сайт буде більше схожий на нативну програму .

У SPA оновлення сторінки ніколи не відбувається; натомість весь необхідний код HTML, JavaScript і CSS або витягується браузером із завантаженням однієї сторінки [1], або відповідні ресурси динамічно завантажуються та додаються на сторінку за потреби, як правило, у відповідь на дії користувача.

Перш за все розглянемо реалізацію серверної частини, яка реалізована за допомогою мови PHP. Основним модулем буде програма, що буде обслуговувати

запити від браузера і готувати у відповідь нові елементи сторінки або змінювати існуючі. В свою чергу клієнтська частина побудована таким чином, щоб реагувати на дії користувача і генерувати відповідні HTTPS запити до серверної частини. Після цього потрібно прийняти відповідь від сервера і відповідним чином обробити її.

Однією з проблем є проблема взаємодії браузерної частини з ручним зчитувачем QR-коду, що реалізується у вигляді віртуальної клавіатури.

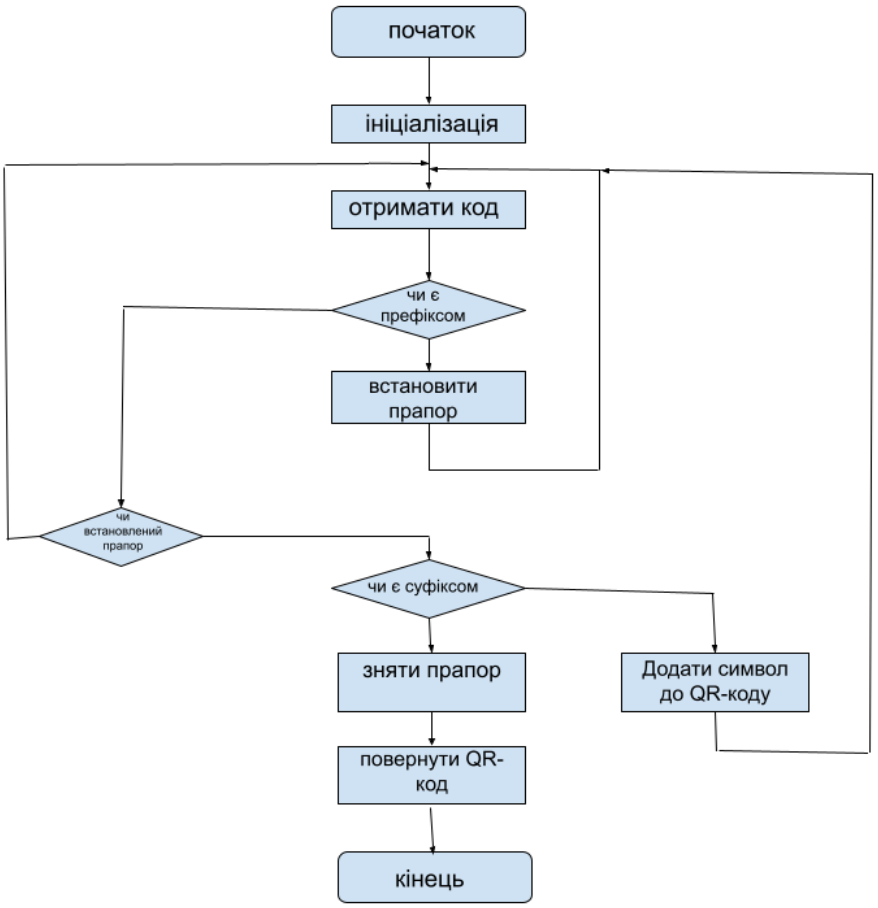


Рисунок 3.1 - Алгоритм зчитування QR-коду

Для цього була створена відповідна функція.

```
let barcodeRead = '';
let readingBarcode = false;
let prefix = 2;
let suphix = 13;
```

```
let handleKeyPress = (e) => {
    if (e.keyCode === prefix) {
        // Start of barcode
        readingBarcode = true;
        e.preventDefault();
        return;
    }
    if (readingBarcode) {
        e.preventDefault();
        if (e.keyCode === suphix) {
            readingBarcode = false;
            barcodeRead = '';
            parse_barcode(barcodeRead);
            return;
        }
        // Append the next key to the end of the list
        barcodeRead += e.key;
    }
}
$(window).bind('keypress', handleKeyPress);
```

Далі, якщо ми хочемо передати інформацію про товар іншому користувачеві, ми повинні сформувати QR-код за сканованим товаром.

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 полотні QR-коду.

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 послідновністю 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.

## 3.3 Тестування та аналіз результатів роботи програмного модуля розпізнавання товарів за QR-кодом.

Головне вікно програми містить поля для вводу інформації:

- Назва
- Одиниця виміру
- Штрих-код
- Вага
- Об’єм

- Термін придатності

Також є панель кнопок, для того щоб вона з’явилась потрібно навести на неї мишкою, після чого вибрати потрібну нам кнопку, а саме “Відсканувати QR-код”.

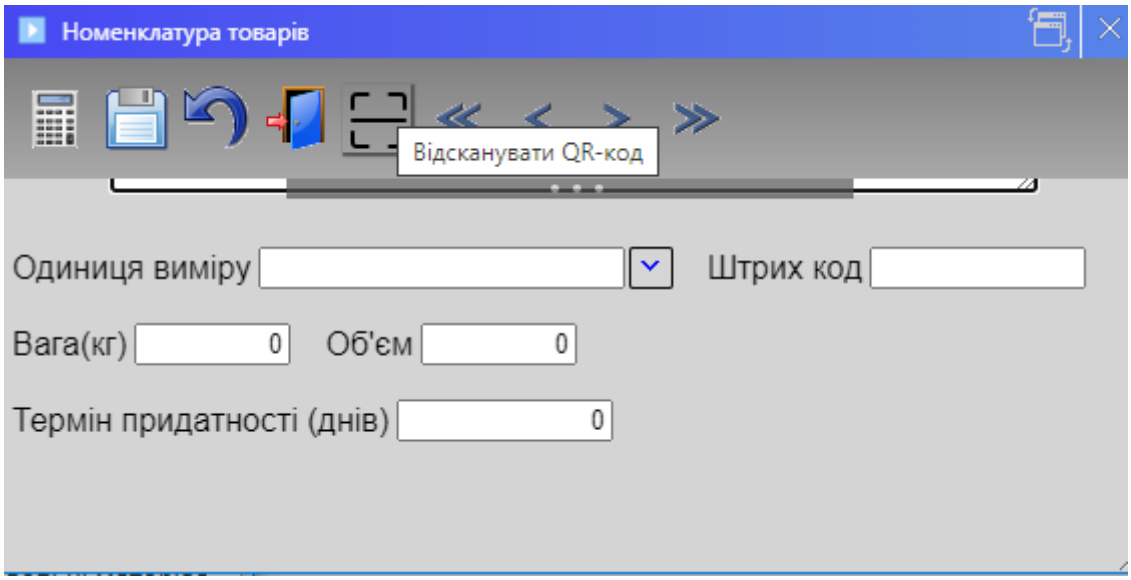


Рисунок 3.2 - Кнопка “Відсканувати QR-код”.

Після цього з’явиться модальне вікно із запрошенням провести сканування QR-коду за допомогою спеціального пристрою для зчитування (сканера).

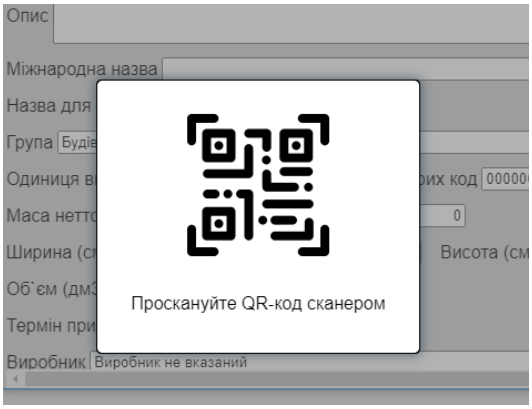


Рисунок 3.3 - Модальне вікно очікування сканування

Після того як сканування було проведено,у разі якщо QR код коректний, то інформація з зчитаного QR-коду буде перенесена у відповідні поля.Після цього

користувач може прийняти рішення про збереження інформації, або він може проігнорувати її і закрити вікно.

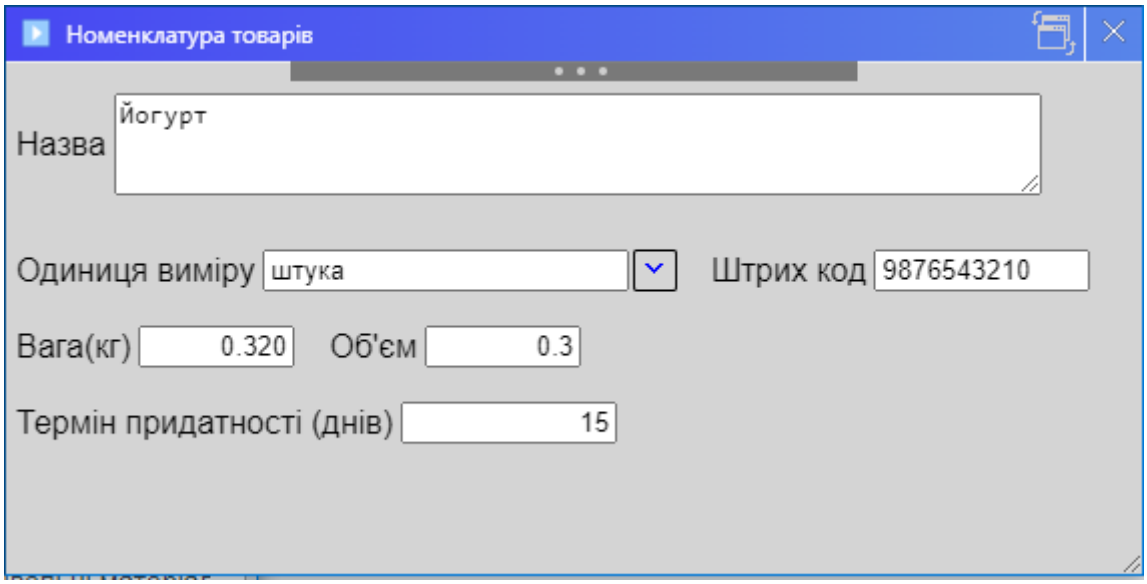


Рисунок 3.4 - Вікно з отриманою інформацією

Якщо користувач обирає збереження, програма автоматично фіксує отримані дані у базі даних і додає їх до відповідного документа (накладної).

У разі потреби користувач може відредагувати деякі поля перед збереженням — додати коментар, змінити об’єм або вибрати інші обиниці виміру. Після остаточного підтвердження всі зміни фіксуються у БД, і товар стає доступним для подальших операцій з ним: наприклад, продажу, переміщення, списання.

Якщо ж користувач вирішує не зберігати інформацію, всі зчитані дані анулюються, а вікно сканування повертається у початковий стан, готовий до наступного сканування. Це дозволяє швидко обробляти велику кількість товарів без зайвих затримок та помилок.

Отже, проведемо порівняння функціональних можливостей програм розпізнавання товарів за QR-кодом (табл. 3.1)

Таблиця 3.1 – Порівняння функціональних можливості програм розпізнавання товарів за QR-кодом

| | QR Code – сканер QR коду | Сканер qr кода – qr code scanner | Програмний модуль розпізнавання товарів за QR-коду |
|---|---|---|---|
| Дружній інтерфейс | + | + | + |
| Можливість подальшої обробки інформації про товар | - | - | + |
| Можливість вносити зміни до інформації про товар | - | - | + |

З таблиці 3.1 видно, що розроблений програмний модуль володіє новими функціональними можливостями: Подальша обробка інформації про товар; Внесення змін до інформації про товар.

### 3.4 Висновок до розділу 3

У даному розділі було обгрунтовано вибір мови програмування. В ході даного аналізу для розробки програмного модуля розпізнавання товарів за QR-кодом було вибрано мову програмування PHP, Javascript. Описано процес розробки модуля сканування QR-коду. Виконано тестування розробленого модуля, яке підтвердило розширення функціональних можливостей програмного забезпечення розпізнавання товарів за QR-кодом.

# ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи розроблено програмний модуль розпізнавання товарів за QR-кодом.

В роботі обгрунтовано використання QR-кодів для розпізнвання товарів. Проаналізовано основні відмінності штрихкодів від QR-кодів. Проаналізовано переваги та недоліки програм аналогів для сканування QR-кодів та розпізнавання їх вмісту.

Обгрунтовано вибір методу підготовки інформації про товари для кодування товарними кодами. Створено структурну схему кодування/декодування товару за QR-коду. Проведено опис існуючих режимів аналізу даних двовимірного QR-коду. Розроблено алгоритм роботи програмного модуля розпізнавання товарів за QR-кодом. Розроблено UML-діаграми програмного модуля розпізнавання товарів за QR-кодом.

Обгрунтовано вибір мови програмування. В ході даного аналізу для розробки програмного модуля розпізнавання товарів за QR-кодом було вибрано мову програмування PHP, Javascript. Описано процес розробки модуля сканування QR-коду. Виконано тестування розробленого модуля, яке підтвердило розширення функціональних можливостей програмного забезпечення розпізнавання товарів за QR-кодом принаймні на 2 додаткові функції.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Як створити ефективний QR-код, за яким будуть переходити. [Електронний ресурс]. Режим доступу:https://web-promo.ua/ua/blog/kak-sozdat-effektivnyj-qr-kod-po-kotoromu-budut-perehodit.

2. Історія виникнення, застосування та перспективи розвитку технології qr-кодування. [Електронний ресурс]. Режим доступу: https://phm.cuspu.edu.ua/nauka/naukovo-populiarni-publikatsii/755-istoriya-vynyknennya-zastosuvannya-ta-perspektyvy-rozvytku-tekhnolohiyi-qr-koduvannya.html

3. Генерація QR-коду в PHP [Електронний ресурс]. Режим доступу: https://snipp.ru/php/qr-code

4. Найкращий спосіб зчитувати ручний сканер. [Електронний ресурс]. Режим доступу:https://overcoder.net/q/244373/javascript-D0%BA%D0%B0%D0%BA-%D0%BB%D1%83%D1%87%D1%88%D0%B5-%D0%B2%D1%81%D0%B5%D0%B3%D0%BE-%D1%87%D0%B8%D1%82%D0%B0%D1%82%D1%8C-%D1%80%D1%83%D1%87%D0%BD%D0%BE%D0%B9-%D1%81%D0%BA%D0%B0%D0%BD%D0%B5%D1%80-%D1%88%D1%82%D1%80%D0%B8%D1%85-%D0%BA%D0%BE%D0%B4%D0%B0

5. Introducing JSON [Electronic resource] – Mode of access: https://www.json.org/json-en.html

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

7. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

8. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

9. Путятін, Є. П., Гороховатський, В. О., & Матат, О. О. (2006). Методи та алгоритми комп'ютерного зору: навч. посібник.

10. Стаття Вікіпедії – Javascript. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/JavaScript

11. Навчальні матеріали [Електронний ресурс]. Режим доступу:: https://developer.mozilla.org/ru/docs/Web/JavaScript.

12. PHP Documentation. [Electronic resource] – Mode of access: https://www.php.net/docs.php.

13. Вікіпедія – Односторонній додаток URL: https://en.wikipedia.org/wiki/Single-page_application.

14. PHP Tutorial [Electronic resource] – Mode of access: https://www.w3schools.com/php/.

15. PHP [Електронний ресурс]. Режим досутупу: https://uk.wikipedia.org/wiki/PHP

16. Побудова діаграми варіантів використання – URL: http://moodle.ipo.kpi.ua/.

17. QR Code Tutorial: Introduction. URL: https://www.thonky.com/qrcode-tutorial/introduction (дата звернення 28.05.2025).

18. Structure of The QR Code. URL: https://myqrbc.com/structure-of-theqr-code-how-is-the-data-coded/ (дата звернення 27.05.2025).

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

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

## ДОДАТОК А (обов'язковий)
## ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль розпізнавання товарів за QR-кодом

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 24,89 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Колодний В.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Кудрик В.Я.
(підпис) (прізвище, ініціали)

# ДОДАТОК Б (довідниковий)
# Інструкція користувача

Головне вікно програми містить поля для вводу інформації:

- Назва
- Одиниця виміру
- Штрих-код
- Вага
- Об’єм
- Термін придатності

Також є панель кнопок, для того щоб вона з’явилась потрібно навести на неї мишкою, після чого вибрати потрібну нам кнопку, а саме “Відсканувати QR-код”.

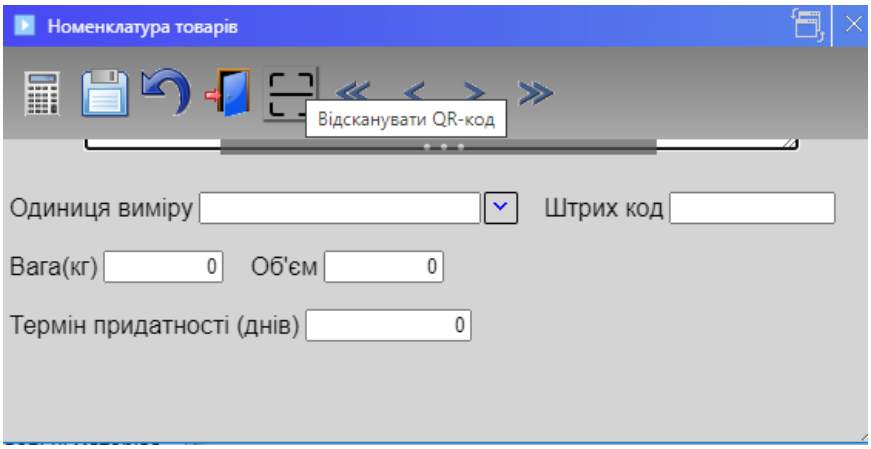


Рисунок Б.1 - Кнопка “Відсканувати QR-код”.

Після цього з’явиться модальне вікно із запрошенням провести сканування QR-коду за допомогою спеціального пристрою для зчитування (сканера).

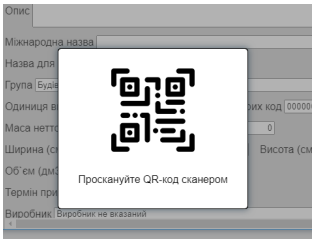


Рисунок Б.2 - Модальне вікно очікування сканування

Після того як сканування було проведено,у разі  якщо QR код коректний, то інформація з зчитаного QR-коду буде перенесена у відповідні поля.Після цього користувач може прийняти рішення про збереження інформації, або він може проігнорувати її і закрити вікно.

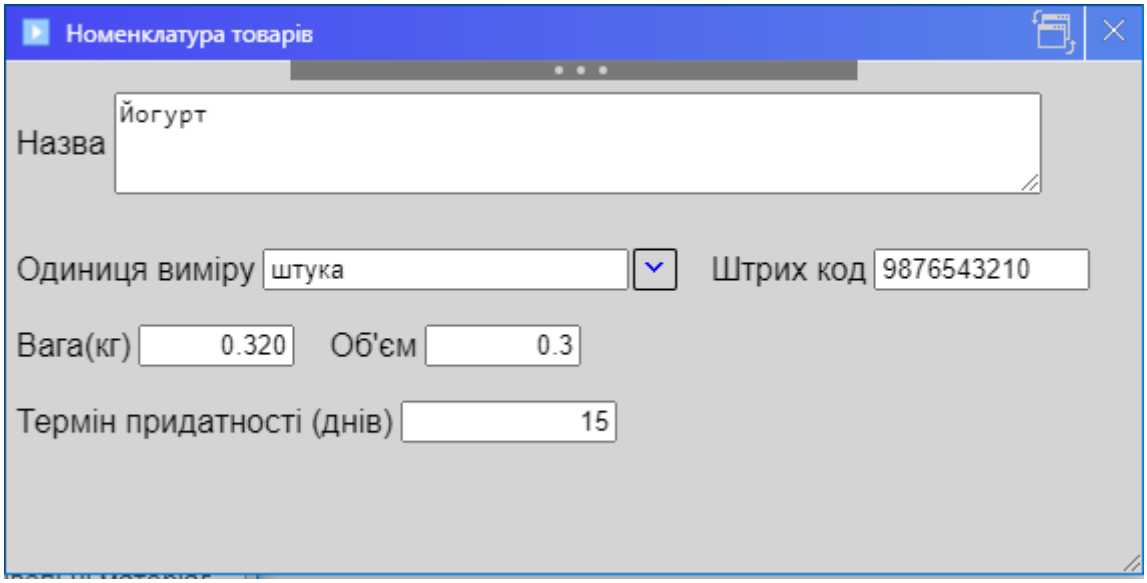


Рисунок Б.3 - Вікно з отриманою інформацією

Якщо користувач обирає збереження, програма автоматично фіксує отримані дані у базі даних і додає їх до відповідного документа (накладної).

# ДОДАТОК В (обов'язковий)
# Лістинг програми

### Клас Main

exchanhe.php

```
<?php

$in_element = '';
$in_event = '';
if (isset($_POST["element"]));
	$in_element = $_POST["element"];
if (isset($_POST["event"]));
	$in_event = $_POST["event"];

// Create variable for connection to Oracle
$conn = '';

require_once 'library.php';

if (!session_start()) {
	$return = run_request('mainForm', 'login', 'Не можу почати сесію!');
}
else {
	$error_login = "";
	if	(!isset($_COOKIE['PHPSESSID']) || $_COOKIE['PHPSESSID']!=session_id()) {
		// Перший раз запитуэмо Login & Password
		$in_element = 'mainForm';
		$in_event = 'login';
	}
	elseif ( $in_element=='loginButton' && $in_event=='onclick' ) {
		// Прийшла відповідь з Login & Password
		if (!check_captcha()) {
			$error_login = "Капча відхилена сервером!";
		}
		else {
			$conn = get_connect(true);
			if ($conn) {
				$in_element = 'mainForm';
				$in_event = 'start';
			}
			else {
				$m = oci_error();
				if ($m['code']==1017)
					$error_login = "Невірний логін та/або пароль!";
			elseif ($m['code']==12545)
					$error_login = "Сервер не відповідає! Спробуйте пізніше.";
				else
					$error_login = $m['message'];
			}
		}
	}
	else {
		if (login_check()) {
			$conn = get_connect(false);
			if (!$conn) {
```

```
                                $in_element = 'mainForm';
                                $in_event = 'start';
                        }
                }
                else {
                        $in_element = 'mainForm';
                        $in_event = 'login';
                }
        }
        $return = run_request($in_element, $in_event, $error_login);
        if (isset($_SESSION['CUR_CONN']['NEED_CLOSE_CONNECTION'])) {
                oci_close($conn);
        }
}
if ($return)
        echo(json_encode($return));
return;

function run_request($in_element, $in_event, $message) {
        if (isset($_POST['date_beg']) && $in_element!='menu_Exit') {
                store_dates($_POST['date_beg'], $_POST['date_end']);
        }

        $return = [];
        if ($in_element=="mainForm" && $in_event=="login") {
                $return[] = array("element" => "mainForm", "type"=>"html",
"value"=>html_login_form($message));
                $return[] = array("element" => 'mainForm', "type"=>"script", "function"=>"grecaptcha");
        }

        elseif ($in_element=="loginButton" && $in_event=="onclick") {
                $return[] = array("element" => "loginInfo", "type"=>"html", "value"=>$message);
                $return[] = array("element" => 'mainForm', "type"=>"script", "function"=>"grecaptcha.reset"
);
        }

        elseif ($in_element=="mainForm" && $in_event=="start") {
                $id = "startPage";
                $return[] = array("element" => "mainForm", "type"=>"html", "value"=>html_titlebar());
                $return[] = array("element" => "mainForm", "type"=>"html+", "value"=>html_menubar());
                $return[] = array("element" => "mainForm", "type"=>"html+",
"value"=>html_modal_container());
        }

        elseif ($in_element=="get_enterprises" && $in_event=="onclick") {
                $return[] = array("element" => "modal_window", "type"=>"html",
"value"=>html_get_enterprises());
                $return[] = array("element" => 'mainForm', "type"=>"script", "function"=>"showmodal");
        }

        elseif ($in_element=="window" && $in_event=="remove") {
                unset_window_id($_POST["w_id"]); //Видаляэмо з _CONTEXT посилання на вікно та
табулятор(и)
        }

        elseif (substr($in_element,0,5)=="menu_" && $in_event=="onclick") {
                $id=substr($in_element,5);
```

```
                if ($id=="Exit") {
                        my_close_session();
                        $return[] = array("element" => "mainForm", "type"=>"html",
"value"=>html_login_form($message));
                        $return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"grecaptcha");
                }
                else {
                        $menu_item = $_SESSION['CONTEXT']['APPLICATION_MENU'][$id];
                        $tablename = $menu_item['TABLE_NAME'];
                        if ($menu_item['ID_QUERY_GROUPS']) {
                                // Запускаємо звіт
                                $return =
create_form_query_parameters($menu_item['ID_QUERY_GROUPS'],
                                $menu_item['ID_APPLICATION_PLUGINS']);
                        }
                        else if ($tablename) {
                                $w_id = get_window_id('form_ch');
                                $prm = [];
                                load_table_structure($tablename);
                                $struct = $_SESSION['TABLES_STRUCT'][$tablename];

                                $attribute = 'APP_MENU.'.$id.'~'.$tablename;
                                if (isset($struct['SESSION_GRID_ADD_FILTERS'][$attribute])) {

        $prm[$struct['SESSION_GRID_ADD_FILTERS'][$attribute]['COLUMN_NAME']] =
 $struct['SESSION_GRID_ADD_FILTERS'][$attribute]['CONSTANT_VALUE'];
                                }

                                if (isset($struct['SESSION_TAB_COLUMNS']['DTDOC'])) {
                                        // Додаємо фільтр по даті
                                        $date_beg = $_POST['date_beg'];
                                        $date_end = $_POST['date_end'];
                                        $prm['DTDOC'] = array('DATE_BEG'=>$date_beg,
'DATE_END'=>$date_end);
                                }

                                $t_prop = html_table($tablename, $w_id, $prm);
                                $return[] = array("element" => "mainForm", "type"=>"window", "id"=>$w_id,
"tablename"=>$tablename, "caption"=>$t_prop["caption"], "w_style"=>"form_ch");
                                $return[] = array("element" => "b1_".$w_id, "type"=>"html",
"value"=>$t_prop["html"]);
                                $return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"add_tabulator", "id"=>$t_prop["id"], "definition"=>$t_prop["definition"],
"definition_filter"=>$t_prop["definition_filter"]) ;
                        }
                        else {
                                $e['message'] = "Недостатньо привілеїв для доступу\n або пункт меню
порожній!";
                                $return = create_error_message($e, '');
                        }
                }
        }

        elseif (substr($in_element,0,4)=="btn_") {
                if (isset($_POST['data']))
                        $rows = json_decode($_POST['data'], true);
```

```
		if (isset($_POST['fk'])) {
			$prm = [];
			$fld_name = $_POST['fk'];
			$fk = substr($in_element,7);
			if (isset($_POST['tablename'])) {
				$tablename = $_POST['tablename'];
				load_table_structure($tablename);
			}
			else {
				$tmp = explode('~', substr($in_element,7));
				$tn = $_SESSION['CONTEXT']['_TABULATORS'][$tmp[1]]['TABLENAME'];
				if
(isset($_SESSION['TABLES_STRUCT'][$tn]['SESSION_REFERENCES'][$fld_name])) {
					$tablename_prefix = 'VS_';
					$tablename =
$_SESSION['TABLES_STRUCT'][$tn]['SESSION_REFERENCES'][$fld_name]["R_TABLE_NAME"];
					if ($tablename_prefix)
						$tablename = $tablename_prefix.substr($tablename,3);
	//				error_log('tablename= '.$tablename);
					$srf =
$_SESSION['TABLES_STRUCT'][$tn]['SESSION_REFERENCE_FILTERS'][$fld_name];
					if ($srf) {
						foreach ($srf As $f_filt) {
							if ($f_filt['TYPE_VALUE'] == 0)		// Const
								$prm[$f_filt['R_COLUMN_NAME_VALUE']] =
$f_filt['CONSTANT_VALUE'];
							elseif ($f_filt['TYPE_VALUE'] == 1)	// Колонка
								$prm[$f_filt['R_COLUMN_NAME_VALUE']] =
$rows[$f_filt['COLUMN_NAME_VALUE']];
						}
					}
				}
			}
			$w_id = get_window_id('form_ch');
			$t_prop = html_table($tablename, $w_id, $prm);
			$return[] = array("element" => "mainForm", "type"=>"window", "id"=>$w_id,
"tablename"=>$tablename, "caption"=>$t_prop["caption"], "w_style"=>"form_ch");
			$return[] = array("element" => "b1_".$w_id, "type"=>"html",
"value"=>$t_prop["html"]);
			$return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"add_tabulator", "id"=>$t_prop["id"], "definition"=>$t_prop["definition"],
"definition_filter"=>$t_prop["definition_filter"], "fk"=>$fk);
		}
		elseif (isset($_POST['btn_type'])) {
	//(substr($in_element,0,8)=="btn_edt_") {
			$btn_type = $_POST['btn_type'];
			$id_tabulator = $_POST['id_tabulator'];
			if ($btn_type == 'new') {
				$return = get_edit_form_div($id_tabulator, $btn_type);
			}
			elseif ($btn_type == 'edit') {
				$return = get_edit_form_div($id_tabulator, $btn_type, $rows);
			}
			elseif ($btn_type == 'save') {
				$return = run_update($id_tabulator, $_POST['mode'], $rows, false);
			}
			elseif ($btn_type == 'exit') {
```

```
                        $return = run_update($id_tabulator, $_POST['mode'], $rows, true);
                    }
                    elseif ($btn_type == 'delete') {
                        $return = run_delete($id_tabulator, $rows);
                    }
                    elseif ($btn_type == 'find' || $btn_type == 'refresh') {
                        $return = set_filter($id_tabulator, $rows);
                    }
                    elseif ($btn_type == 'qr_scan') {
                        $return = [];
                        $html = "<div class=\"qr_image\"><img
src=\"images/message_qr_scan.png\"></div><div class=\"error_mess\">Проскануйте QR-код
сканером</div>";
                        $return[] = array("element" => "modal_window", "type"=>"html",
"value"=>$html);
                        $return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"showmodal");
                        $return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"startqr");
                    }
                    elseif ($btn_type == 'run_spr') {
                        $tablename = $_POST['tname'];
                        $fname = $_POST['fname'];
                        $r_fname = $_POST['r_fname'];
                        $prm[$fname] = $rows[$r_fname];
                        $w_id = get_window_id('form_ch');
                        $t_prop = html_table($tablename, $w_id, $prm);
                        $return[] = array("element" => "mainForm", "type"=>"window", "id"=>$w_id,
"tablename"=>$tablename, "caption"=>$t_prop["caption"], "w_style"=>"form_ch");
                        $return[] = array("element" => "b1_".$w_id, "type"=>"html",
"value"=>$t_prop["html"]);
                        $return[] = array("element" => 'mainForm', "type"=>"script",
"function"=>"add_tabulator", "id"=>$t_prop["id"], "definition"=>$t_prop["definition"],
"definition_filter"=>$t_prop["definition_filter"], "fk"=>substr($in_element,7));
                    }
                    elseif ($btn_type == 'preview') {
                        $id = $_POST['id_query_group'];
                        $id_form = $_POST['id_form'];
                        $w_id = substr($id_form, 8);  //"form_b1_$w_id"
                        $dataset = run_query_data($id, $w_id, $rows);
                        $file_name = '';
                        if (isset($_POST['id_report'])) {
                            $id_report = $_POST['id_report'];
                            $app_rep =
$_SESSION['CONTEXT']['SESSION_QUERY_GROUPS'][$id]['APPLICATION_REPORTS'][$id_report]['
FILE_NAME'];
                            $caption =
$_SESSION['CONTEXT']['SESSION_QUERY_GROUPS'][$id]['APPLICATION_REPORTS'][$id_report]['
NAME'];
                            $report_name = strtolower($app_rep).'.php';
                            $dataset['report_attrib'] = get_report_attrib($id_report);
                            $file_name =
'_REPORTS/'.$_SESSION['CUR_CONN']['schema'].'/'.$report_name;
                        }
                        else if (isset($_POST['id_plugin'])) {
                            $id_plugin = $_POST['id_plugin'];
                            $app_rep =
```

```
$_SESSION['CONTEXT']['APPLICATION_PLUGINS'][$id_plugin]['FILE_NAME'];
                                        $caption =
$_SESSION['CONTEXT']['APPLICATION_PLUGINS'][$id_plugin]['CAPTION'];
                                        $report_name = strtolower(pathinfo($app_rep,
PATHINFO_FILENAME)).'.php';
                                        $file_name =
'_PLUGINS/'.$_SESSION['CUR_CONN']['schema'].'/'.$report_name;
                                }
                                if (file_exists($file_name)) {
                                        $out_file='document_'.$id_form;       // extension визначається в
самому плагіні. Обовязково!!!
                                        require($file_name);

                                        if (file_exists('_TMP/'.$out_file)) {
                                                if (pathinfo($out_file, PATHINFO_EXTENSION)=='pdf')
                                                        $return[] = array("element" => 'mainForm',
"type"=>"script", "function"=>"run_ext_window", "url"=>"run_ext.php?f=$out_file", "title"=>$caption);
                                                elseif (pathinfo($out_file, PATHINFO_EXTENSION)=='xls')
                                                        $return[] = array("element" => 'mainForm',
"type"=>"script", "function"=>"download", "url"=>"run_ext.php?f=$out_file", "fname"=>$out_file);
                                        }
                                        else {
                                                $e['message'] = "Нажаль, при створенні документу
сталася помилка.";
                                                $return = create_error_message($e, '');
                                        }
                                }
                                else {
                                        $e['message'] = "Вибачте, але файл <span style=\"font-
weight:bold\">$report_name</span> для цього звіту ще в розробці.<br>Ми відмітили, що Вам
потрібен цей звіт, тому зробимо його найближчим часом.";
                                        $return = create_error_message($e, '');
                                        error_log("!!! report_name $report_name need to do !!!");
                                }
                        }
                }
        }
        elseif ($in_event == "set_displayvalue") {
                $return = set_displayvalue(json_decode($_POST['prm'], true));
        }
        elseif ($in_event == "switch_controls") {
                $return = switch_controls($_POST['id'], $_POST['name'], json_decode($_POST['data'],
true));
        }


        return $return;
}

$tablename = $_SESSION['CONTEXT']['_TABULATORS'][$id_tabulator]['TABLENAME'];
        $columnname = $_SESSION['FORM_EDIT'][$tablename]['CONTROLS'][$id]['COLUMN_NAME'];
        $classname =
strtolower($_SESSION['FORM_EDIT'][$tablename]['CONTROLS'][$id]['CONTROL']);
        $session_tab_columns =
$_SESSION['TABLES_STRUCT'][$tablename]['SESSION_TAB_COLUMNS'][$columnname];
        $control_prop = $_SESSION['FORM_EDIT'][$tablename]['CONTROL_PROPERTIES'][$id];
        $onchange = '';
```

```
	if
(isset($_SESSION['FORM_EDIT'][$tablename]['SESSION_SWITCH_CONTROLS']['SWITCH_LIST']))
		if (in_array($columnname,
$_SESSION['FORM_EDIT'][$tablename]['SESSION_SWITCH_CONTROLS']['SWITCH_LIST']))
			$onchange = ' onchange="Switch_Controls(this)"';

	$elm_id = $w_id.'~'.$id_tabulator.'~'.$columnname;
	$html_text = '';
	if (isset($control_prop['TOP'])) {		// Контрол є на формі
		if ($columnname=="")
			$value=$classname;
		else
			$value=$columnname;

//		error_log("classmame=$classname");
//		error_log(print_r($control_prop, true));

		$width = (isset($control_prop['WIDTH']) ? $control_prop['WIDTH'] : '100').'px';

		$disabled = '';
		if
(isset($_SESSION['TABLES_STRUCT'][$tablename]['SESSION_COLUMN_PRIVS'][$columnname]))
			if
(!$_SESSION['TABLES_STRUCT'][$tablename]['SESSION_COLUMN_PRIVS'][$columnname]['UPD'])
				$disabled = ' disabled';
		if (!$disabled && (isset($control_prop['ENABLED']) ||
in_array($columnname,array('PK','ID','IP'))))
			$disabled = ' disabled';

		if ($classname=="ag_form_edit") {
			//
		}
		elseif ($classname=="ag_label") {
			$html_text = (isset($control_prop['CAPTION']) ? $control_prop['CAPTION'] :
'Caption не вказано');
			$title = (isset($control_prop['TOOLTIPTEXT']) ?
'title="'.$control_prop['TOOLTIPTEXT'].'"' : '');
			$html_text = "<label class=\"ag_label\" $title>$html_text</label>";
		}
		elseif ($classname=="ag_checkbox") {
			$html_text = "<input id=\"$elm_id\" type=\"checkbox\" class=\"checkbox\"
name=\"$columnname\" frmId=\"$frmid\"$disabled$onchange>";
			if (isset($control_prop['CAPTION']))
				$html_text .= '<label
class="ag_label">'.$control_prop['CAPTION'].'</label>';
		}
		elseif ($classname=="ag_datetime") {
			$html_text = "<input id=\"$elm_id\" type=\"date\" class=\"datetime\"
name=\"$columnname\" frmId=\"$frmid\"$disabled$onchange>";
		}

		elseif ($classname=="ag_combobox") {
			if (isset($control_prop['IS_INDEXVALUE'])) {
				$delta_index = (isset($control_prop['DELTA_INDEX']) ?
$control_prop['DELTA_INDEX'] : 0);
				$items = explode(',',$control_prop['ROWSOURCE']);
				for ( $i=0, $len=count($items); $i<$len; $i++ ) {
```

```
                                        $html_text .= '<option>'.$items[$i].'</option>';
                                }
                                $html_text = "<select id=\"$elm_id\" class=\"select\"
name=\"$columnname\" delta_index=$delta_index style=\"width:$width\"
frmId=\"$frmid\"$disabled$onchange>$html_text</select>";
                        }
                        else {
                                $my_cs = (isset($control_prop['MY_CONTROLSOURCE']) ?
$control_prop['MY_CONTROLSOURCE'] : '');
                                $upd_cs = (isset($control_prop['UPD_CONTROLSOURCE']) ?
$control_prop['UPD_CONTROLSOURCE'] : '');
                                $tn = (isset($control_prop['TABLE_NAME']) ?
$control_prop['TABLE_NAME'] : '');;

                                $cn_2 = '';
                                $r_cn_2 = '';
                                $tn = '';
                                if
(isset($_SESSION['TABLES_STRUCT'][$tablename]['SESSION_REFERENCES'][$columnname])) {
                                        $session_references =
$_SESSION['TABLES_STRUCT'][$tablename]['SESSION_REFERENCES'][$columnname];

                                        $prefix = '';
                                        if ($tn)
                                                $prefix = substr($tn,0,3);
                                        $tn = $session_references['R_TABLE_NAME'];
                                        if ($prefix == 'VS_' || $prefix == 'VW_')
                                                $tn = $prefix.substr($tn,3);

                                        if (isset($session_references['COLUMN_NAME_2'])) {
                                                $cn_2 = $session_references['COLUMN_NAME_2'];
                                                $r_cn_2 = $session_references['R_COLUMN_NAME_2'];
                                        }
                                        // Підміняємо перший елемент в $upd_cs
                                        $a_upd_cs = explode(',', $upd_cs);
                                        $a_upd_cs[0] = $session_references['R_COLUMN_NAME'];
                                        if ($r_cn_2)
                                                $a_upd_cs[1] = $r_cn_2;
                                        $upd_cs = implode(',', $a_upd_cs);
                                }
                                // Підміняємо перший та можливо другий елемент в $my_cs
                                $a_my_cs = explode(',', $my_cs);
                                $a_my_cs[0] = $columnname;
                                if ($cn_2)
                                        $a_my_cs[1] = $cn_2;
                                $my_cs = implode(',', $a_my_cs);

                                $my_dv = (isset($control_prop['MY_DISPLAYVALUE']) ?
'my_dv="'.$control_prop['MY_DISPLAYVALUE'].'"' : '');
                                $upd_dv = (isset($control_prop['UPD_DISPLAYVALUE']) ?
'upd_dv="'.$control_prop['UPD_DISPLAYVALUE'].'"' : '');

                                $html_text = "<input id=\"$elm_id\" type=\"text\" class=\"textbox\"
name=\"$columnname\" tablename=\"$tn\" my_cs=\"$my_cs\" upd_cs=\"$upd_cs\" $my_dv $upd_dv
fk_value=\"\" style=\"width:$width\" frmId=\"$frmid\"$disabled$onchange>";
                                $html_text .= "<div class=\"fk_button\" title=\"\"><img id=\"btn_fk_$elm_id\"
class=\"\" fk=\"$elm_id\" src=\"images/down.png\" evnt=\"onclick\"></div>";
```

```
                        }
                }

                elseif ($classname=="ag_textbox") {
                        if ((isset($control_prop['HEIGHT']) && $control_prop['HEIGHT']>24) ||
$session_tab_columns['DATA_SCALE']>60) {
                                $html_text = "<textarea id=\"$elm_id\" name=\"$columnname\" cols=\"60\"
rows=\"3\" frmId=\"$frmid\"$disabled$onchange></textarea>";
                        }
                        else {
                                if ($session_tab_columns['DATA_TYPE']=='NUMBER')
//                                      $html_text = '<input class="number" type="number"
id="'.$columnname.'" pattern="\d+(\.\d{6})?" name="'.$columnname.'" step="0.000001"
style="width:'.(isset($control_prop['WIDTH']) ? $control_prop['WIDTH'] : 100).'px">';
                                        $html_text = "<input id=\"$elm_id\" type=\"text\" class=\"number\"
name=\"$columnname\" style=\"width:$width\" frmId=\"$frmid\"$disabled$onchange onkeypress=\"return
validate_number(event);\">";
                                else
                                        $html_text = "<input id=\"$elm_id\" type=\"text\" class=\"textbox\"
name=\"$columnname\" style=\"width:$width\" frmId=\"$frmid\"$disabled$onchange>";
                        }
                }
                else
                        error_log("create_form_element: classname $classname not exist!");
        }
        return $html_text;
}
function get_toolBar($type, $id_tabulator, $form_id, $id_plugin='') {
$is_admin = ($_SESSION['CUR_CONN']['IS_ADMIN'] == 'YES');
        $html = '';
        if ($type == 'form_ch') {
                $tablename = $_SESSION['CONTEXT']['_TABULATORS'][$id_tabulator]['TABLENAME'];
                $html .= get_toolbar_button ($id_tabulator, 'new.png', 'new', 'Добавити новий F2,
Дублювати Ctrl+F2');
                $html .= get_toolbar_button ($id_tabulator, 'edit.png', 'edit', 'Редагувати F4, Групове
редагування Shift+F4');
                $html .= get_toolbar_button ($id_tabulator, 'view.png', 'view', 'Перегляд F3');
                $html .= get_toolbar_button ($id_tabulator, 'delete.png', 'delete', 'Видалити Delete, F8;
Один запис Ctrl+F8; Групове видалення Shift+F8');
                $html .= get_toolbar_button ($id_tabulator, 'refresh.png', 'refresh', 'Оновлення Ctrl+R');
                $html .= get_toolbar_button ($id_tabulator, 'find.png', 'find', 'Пошук F7, Ctrl+F7, Shift+F7');
                $html .= get_toolbar_button ($id_tabulator, 'find_clear.png', 'find_clear', 'Очистити пошук
Ctrl+Пробіл');
}
elseif ($type == 'form_edit') {
                $html .= get_toolbar_button ($id_tabulator, 'calculator.png', 'calc', 'Перерахувати
документ');
                $html .= get_toolbar_button ($id_tabulator, 'save.png', 'save', 'Зберегти Ctrl+S');
                $html .= get_toolbar_button ($id_tabulator, 'undo.png', 'undo', 'Перечиати дані Ctrl+U');
                $html .= get_toolbar_button ($id_tabulator, 'exit.png', 'exit', 'Зберегти та вийти F12');
                $html .= get_toolbar_button ($id_tabulator, 'wztop.png', 'go_top', 'Перший запис
Ctrl+Home');
                $html .= get_toolbar_button ($id_tabulator, 'wzback.png', 'go_back', 'Попередній запис
Ctrl+Вверх');
                $html .= get_toolbar_button ($id_tabulator, 'wznext.png', 'go_next', 'Наступний запис
Ctrl+Вниз');
                $html .= get_toolbar_button ($id_tabulator, 'wzend.png', 'go_end', 'Останній запис
```

```
Ctrl+End');
                $html .= get_toolbar_button ($id_tabulator, 'qr_scanner.png', 'qr_scan', 'Відсканувати
QR-код');
        }
$html = '<div class="toolbar"><div class="button_container"><div class="button_flex"
form_id="'.$form_id.'">'.$html.'</div></div><span><img src=images/3.png></span></div>';

        return $html;
}

function get_toolbar_button ($id, $img, $btn_type , $title, $id_plugin='') {
        $html = "";
  $html = "<div class=\"button\" title=\"$title\">
        <img id=\"btn_{$btn_type}_$id\" class=\"button_image\" src=\"images/$img\" evnt=\"onclick\"
  tb=\"$id\" btn_type=\"$btn_type\">
  </div>";
        }
  return $html;
  }
```

**as.js**

```
  var dragObject;
  var mouseOffset;
  var globalCursor_Index = 0;
  var globalzIndex = 1000;
  var arr_tabulator = [];
  var menu_cleaner = []; // Перелік нод для яких потрібно скинути visibiliy
var barcodeRead = '';
var readingBarcode = false;
var prefix = 2;
var suphix = 13;
var handleKeyPress = (e) => {
   if (e.keyCode === prefix) {
      // Start of barcode
      readingBarcode = true;
      e.preventDefault();
      return;
   }
   if (readingBarcode) {
      e.preventDefault();
      if (e.keyCode === suphix) {
         readingBarcode = false;
         barcodeRead = '';
      parse_barcode(barcodeRead);
         return;
      }
      // Append the next key to the end of the list
      barcodeRead += e.key;
   }
}
  window.onload = function() {
    do_http('exchange.php', {element: 'mainForm', event: 'start'});
  };

  document.addEventListener("click", function(evnt){
        // Якщо клік на вкладеному елементі, шукаємо предка, доки не знайдемо елемент з
  вказаним ID
```

```
	let node = evnt.target;
	if (node.className=='img_close')
		win_manager.closeWindow(node,evnt);
	else if (node.className=='img_show_child')
		win_manager.showChild(node,evnt);
	else {
		while (!node.id && node.parentElement) {
			node = node.parentElement;
		}

		let is_make_query = true;
		let evnt_type = node.getAttribute('evnt');
		if (evnt_type=='onclick') {

			let prm = {element: node.id, event: evnt_type};
			var data = {};
			if (node.hasAttribute('data-frmId')) {
				// Якщо це form, то додаємо до параметрів значення всіх контролів цієї
форми
				let frmId = node.dataset.frmid;
				for(let item of document.querySelectorAll('[frmId]')) {
					if (item.getAttribute('frmId')==frmId)
						if (item.id=='g-recaptcha')
							prm[item.id] = grecaptcha.getResponse();   //Для
капчи получаем респонсе
						else
							prm[item.id] = item.value;
				}
			}

			else if (node.hasAttribute('fk')) {
				// Нажали на кнопку Forign Key.
				let fk_node = document.getElementById(node.getAttribute('fk'));
				if (fk_node.disabled)
					return;
				prm['fk'] = fk_node.name;
				if (fk_node.hasAttribute('tablename'))
					prm['tablename'] = fk_node.getAttribute('tablename');
				let frmId = get_formId(node);
				prm['data']=JSON.stringify(get_data_from_form(frmId));
			}


			// Ховаємо меню
			else if (node.id.slice(0,5)=="menu_") {
				while (node.className != "topmenu") {
					if (node.className =="submenu") {
						// Ховаємо меню
						node.style.visibility = "hidden";
						menu_cleaner.push(node);
					}
					node = node.parentElement;
				}
				// За рухом мишки очищуємо стилі visibility, щоб працював :hover
				document.onmousemove = MenuCleaner_mouseMove;
			}
```

```
                    // Если это кнопки тулбара
                    else if (node.hasAttribute('btn_type')) {
                        prm['btn_type']=node.getAttribute('btn_type');
                        prm['id_tabulator']=node.getAttribute('tb');
                        let table = arr_tabulator[prm['id_tabulator']];
                        if (['edit', 'delete'].includes(prm['btn_type'])) {
                            let rows = table.getRows('selected');
                            data = rows[0].getData();
                            prm['data']=JSON.stringify(data);
                        }
                        else if (['save', 'exit'].includes(prm['btn_type'])) {
                            // Для edit_form підміняємо значення в data на введені користувачем в форму
                            let frmId = get_formId(node);
                            if (frmId) {
                                prm['id_form']=frmId;
                                prm['mode'] = document.getElementById(frmId).getAttribute('mode');
                                prm['data']=JSON.stringify(get_data_from_form(frmId));
                            }
                        }
                        else if (['find', 'find_clear'].includes(prm['btn_type'])) {
                            var id = prm['id_tabulator'];
                            var tb_id = 'tb_'+id;
                            var tb_filter_id = tb_id+'_filter';
                            var was_find_mode = false;
                            if (typeof arr_tabulator[id]["find_mode''] !== 'undefined')
                                was_find_mode = arr_tabulator[id]["find_mode''];
                            else {
                                // Визвали перший раз, додаємо 2 записи
                                arr_tabulator[id]["_filter''].addRow({});
                                arr_tabulator[id]["_filter''].addRow({});
                            }
                            document.getElementById(tb_id).style.display = ( was_find_mode ? 'block' : 'none');
                            document.getElementById(tb_filter_id).style.display = ( was_find_mode ? 'none' : 'block');
                            if (was_find_mode) {

    prm['data']=JSON.stringify(arr_tabulator[id]["_filter''].getData());
                                prm['mode'] = 'find';
                            }
                            else
                                is_make_query = false;

                            arr_tabulator[id]["find_mode''] = !was_find_mode;
                        }
                        else if (prm['btn_type'] == 'refresh') {
                            // передаємо поточний фільтр

    prm['data']=JSON.stringify(arr_tabulator[prm['id_tabulator']]["_filter''].getData());
                            prm['mode'] = 'refresh';
                        }
                        else if (prm['btn_type'] == 'run_spr') {
                            // перехід до звязаних таблиць
                            let rows = table.getRows('selected');
                            if (rows.length != 1) {
```

```
                                        is_make_query = false;
                                        alert('Потрібно виділити 1 запис!');
                                }
                                else {
                                        prm['data']=JSON.stringify(rows[0].getData());
                                        prm['tname'] = node.getAttribute('tname');
                                        prm['fname'] = node.getAttribute('fname');
                                        prm['r_fname'] = node.getAttribute('r_fname');
                                }
                        }
                        else if (prm['btn_type'] == 'preview') {
                                let frmId = get_formId(node);
                                if (frmId) {
                                        prm['id_form']=frmId;
                                        prm['id_query_group'] = node.getAttribute('tb');
                                        if (node.hasAttribute('id_report'))
                                                prm['id_report'] = node.getAttribute('id_report');
                                        if (node.hasAttribute('id_plugin'))
                                                prm['id_plugin'] = node.getAttribute('id_plugin');
                                        prm['data']=JSON.stringify(get_data_from_form(frmId));
                                }
                        }
                }

                // Відсилаємо на сервер запит з параметрами.
                if (is_make_query) {
                        var db = document.getElementById('global_date_beg');
                        if (db) {
                                prm['date_beg'] = dt_convert(db.value, true);
                                prm['date_end'] =
  dt_convert(document.getElementById('global_date_end').value, true);
                        }
                        do_http('exchange.php', prm);
                }
        }
    }
  });
function do_http(url, params, headers)
{
        var gci = start_wait_cursor();
        var data = '';
        if (typeof params !== "undefined") {
                for (key in params)
                {
                        data = (data=='' ? '' : data+'&') + key + '=' + encodeURIComponent(params[key]);
                }
        }

        var xhr = new XMLHttpRequest();

        xhr.open('POST', url, true);
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.send(data);

        xhr.onreadystatechange = function(){
                if (xhr.readyState != 4) return;
                if (xhr.status != 200) {
```

```
                alert(xhr.status + ': ' + xhr.statusText);
            } else {
                if (xhr.responseText)
                    after_http(JSON.parse(xhr.responseText));
            }
            stop_wait_cursor(gci);
        }
}
function after_http(response) {
        for (let i = 0; i<response.length; i++) {
            if (response[i].type=='script')
                if (response[i].function=='grecaptcha')
                    grecaptcha.render('g-recaptcha', {'sitekey' : '6LdGMrodAAAAAJEk2Nntr--
iOANiMijfPWYxkDV0'});
                else if (response[i].function=='grecaptcha.reset')
                    grecaptcha.reset();
                else if (response[i].function=='showmodal') {
                    element = document.getElementById("modal_shape");
                    element.style.display = "block";
                }
                else if (response[i].function=='add_tabulator') {
                    element = document.getElementById('tb_'+response[i].id);
                    tbl_prop = JSON.parse(response[i].definition);
                    arr_tabulator[response[i].id] = new Tabulator(element, tbl_prop);
                    arr_tabulator[response[i].id]["id_tabulator"] = response[i].id;
                    if (response[i].fk) {
                        arr_tabulator[response[i].id]["fk"] = response[i].fk;
                        arr_tabulator[response[i].id].on("rowDblClick", function(e, row){
                            fk_element = document.getElementById(this.fk);
                            let tmp = this.fk.split('~');
                            prefix = tmp[0]+'~'+tmp[1]+'~';
                            my_cs = fk_element.getAttribute("my_cs").split(',');
                            upd_cs = fk_element.getAttribute("upd_cs").split(',');
                            let CurRow = row.getData();
                            // Прописуємо значення для всіх полів вказаних в upd_cs
                            for (let k=0; k<upd_cs.length; k++) {
                                if (k==0) { //
                                    fk_element.value =
CurRow[fk_element.getAttribute("upd_dv")];
                                    fk_element.setAttribute('fk_value',
CurRow[upd_cs[k]]);
                                    if (fk_element.onchange!=null)
                                        Switch_Controls(fk_element);
                                }
                                else {
                                    tmp_elm =
document.getElementById(prefix+my_cs[k]);
                                    if (tmp_elm) {
                                        if (tmp_elm.hasAttribute('fk_value')) {
                                            tmp_elm.value =
CurRow[tmp_elm.getAttribute("upd_dv")];
                                            tmp_elm.setAttribute('fk_value',
CurRow[upd_cs[k]]);
                                        }
                                        else {
                                            tmp_elm.value =
CurRow[upd_cs[k]];
```

```
                                                                }
                                                            }
                                                        }
                                                    }
                                                    win_manager.closeWindow(e.target, e);
                                                });
                                            }
                                            arr_tabulator[response[i].id].on("rowSelectionChanged", function(data,
rows){
                                                stat_element = document.getElementById("stat_"+this.id_tabulator);
                                                if (stat_element)
                                                    stat_element.innerHTML = "Виділено "+data.length+" рядків
з "+this.getDataCount();
                                            });
                                            if (response[i].definition_filter) {
                                                // Додаємо настройки табулятора для фільтра
                                                element = document.getElementById('tb_'+response[i].id+'_filter');
                                                tbl_prop = JSON.parse(response[i].definition_filter);
                                                arr_tabulator[response[i].id]["_filter"] = new Tabulator(element,
tbl_prop);
                                            }
                                        }
                                        else if (response[i].function=='load_from_selected') {
                                            load_from_selected(response[i].element, response[i].tabulator);
                                        }
                                        else if (response[i].function=='one_record_refresh') {
                                            id = response[i].id_tabulator;
                                            key = response[i].key;
                                            if (response[i].mode=='delete') {
                                                arr_tabulator[id].deleteRow(key);
                                            }
                                            else {
                                                if (response[i].mode=='new') {
                                                    arr_tabulator[id].addRow([response[i].data]);

        document.getElementById(response[i].element).setAttribute('mode', 'edit');
                                                }
                                                else if (response[i].mode=='edit') {
                                                    arr_tabulator[id].updateData([response[i].data]);
                                                }
                                                if (!response[i].close)
                                                    load_data_to_form_edit(response[i].element, id,
response[i].data);
                                            }
                                        }
                                        else if (response[i].function=='set_displayvalue') {
                                            document.getElementById(response[i].id).value = response[i].my_dv;
                                        }
                                        else if (response[i].function=='requery') {
                                            arr_tabulator[response[i].id_tabulator].setData();
                                        }
                                        else if (response[i].function=='closeWindow') {

        win_manager.closeWindow(document.getElementById(response[i].element));
                                        }
                                        else if (response[i].function=='en_dis_control') {
                                            document.getElementById(response[i].element).disabled =
```

```
!response[i].enabled;
                        }
                        else if (response[i].function=='run_ext_window') {
                                url = response[i].url;
                                title = response[i].title;
                                var w = window.open(url, '_blank');
                                w.onload = function() {
                                        w.document.getElementsByTagName('head')[0]
                                                .appendChild(w.document.createElement('title'))
                                                .appendChild(w.document.createTextNode('АС-Комплекс |
'+title));
//                                              setTimeout(function(){
//                                                      w.document.title = 'АС-Комплекс | '+title;
//                                              }, 1000);
                                }
                        }
                        else if (response[i].function=='download') {
                                var element = document.createElement('a');
                                element.setAttribute('href', response[i].url);
                                element.setAttribute('download', response[i].fname);
                                element.style.display = 'none';
                                document.body.appendChild(element);
                                element.click();
                                document.body.removeChild(element);
                        }
                        else if (response[i].function==”scanqr”) {
                                $(window).bind('keypress', handleKeyPress);
                        }
                        else {
//                              eval(response[i].value);
                        }
                else {
                        if (response[i].type=="window") {
                                win_manager.create_window_blank(response[i]);
                        }
                        else {
                                element = document.getElementById(response[i].element);
                                if (element) {
                                        if (response[i].type=='html')
                                                element.innerHTML = response[i].value;
                                        else if (response[i].type=='o_html')
                                                element.outerHTML = response[i].value;
                                        else if (response[i].type=='html+')
                                                element.innerHTML += ' ' + response[i].value;
                                        else if (response[i].type=='+html')
                                                element.innerHTML = response[i].value + ' ' +
element.innerHTML;
                                }
                        }
                }
        }
}
```

## ДОДАТОК Г (обов'язковий)

# ІЛЮСТРАТИВНА ЧАСТИНА

# «ПРОГРАМНИЙ МОДУЛЬ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА QR-КОДОМ»

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Кудрик В.Я.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри КН
Колодний В.В.
(прізвище та ініціали)

03.06.25р

Вінниця ВНТУ - 2025 рік


Рисунок Г.1 – Ключова відмінність QR-коду від традиційного штрихкоду

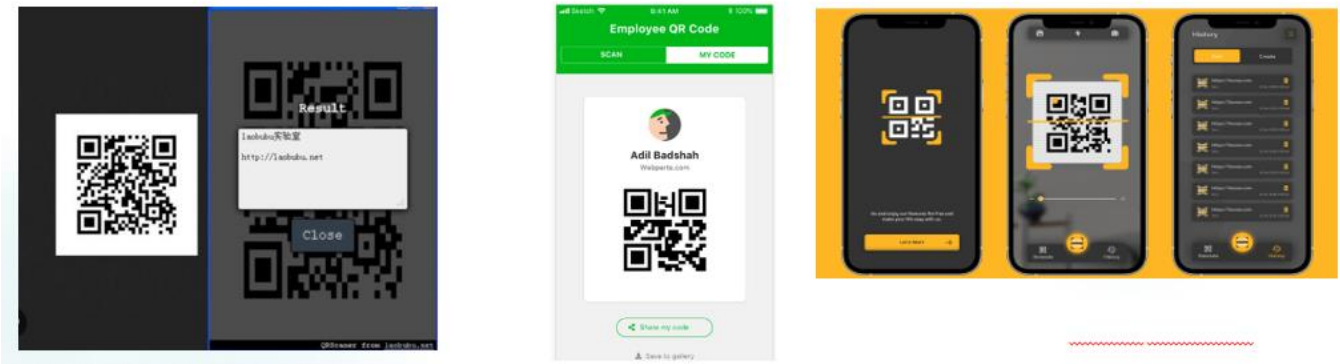


Рисунок Г.2 – Програми аналоги для розпізнавання товарів за QR-кодом

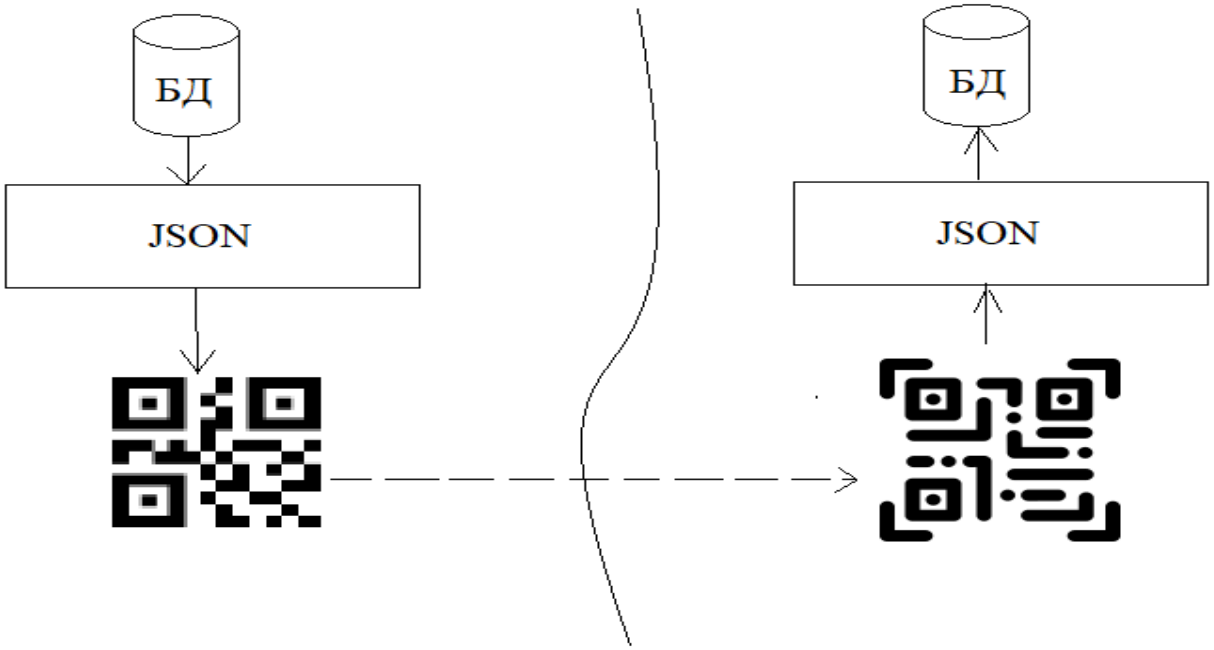


Рисунок Г.3 – Схема кодування і декодування інформації про товар з використанням JSON

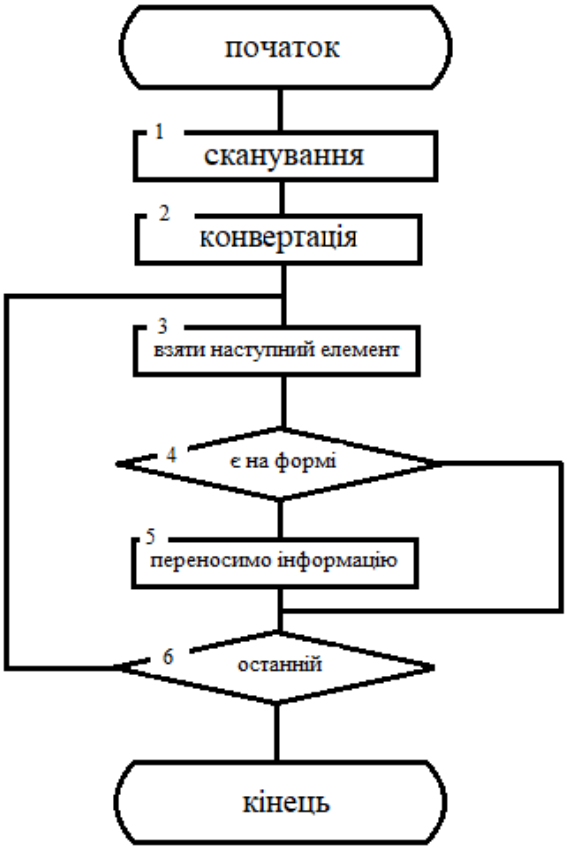


Рисунок Г.4 – Загальний алгоритм функціонування програмного модуля розпізнавання товарів за QR-кодом

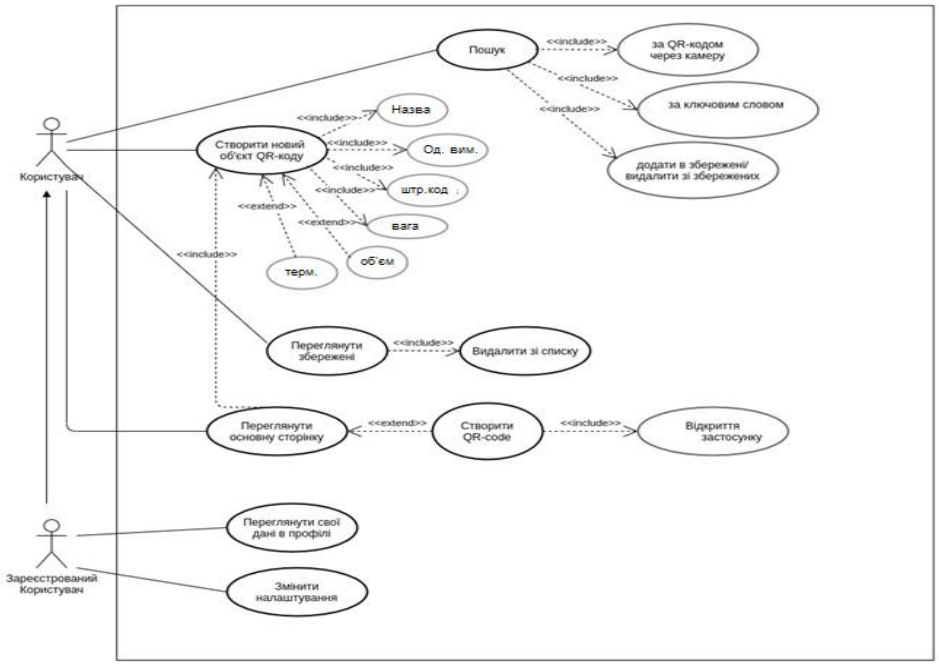


Рисунок Г.5 – Діаграма прецедентів програмного модуля розпізнавання товарів за QR-кодом

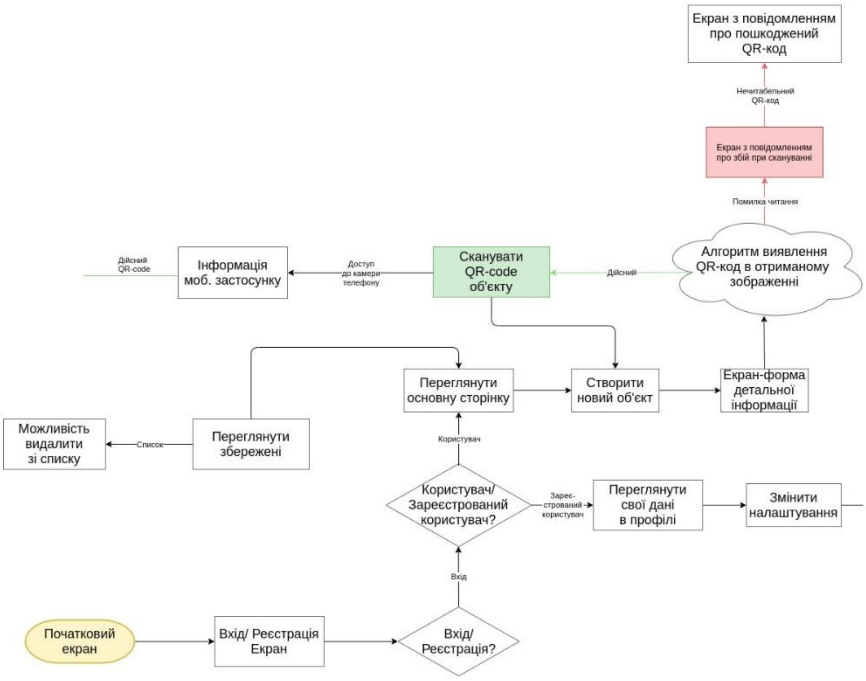


Рисунок Г.6 – Діаграма користувацьких потоків (user flow)

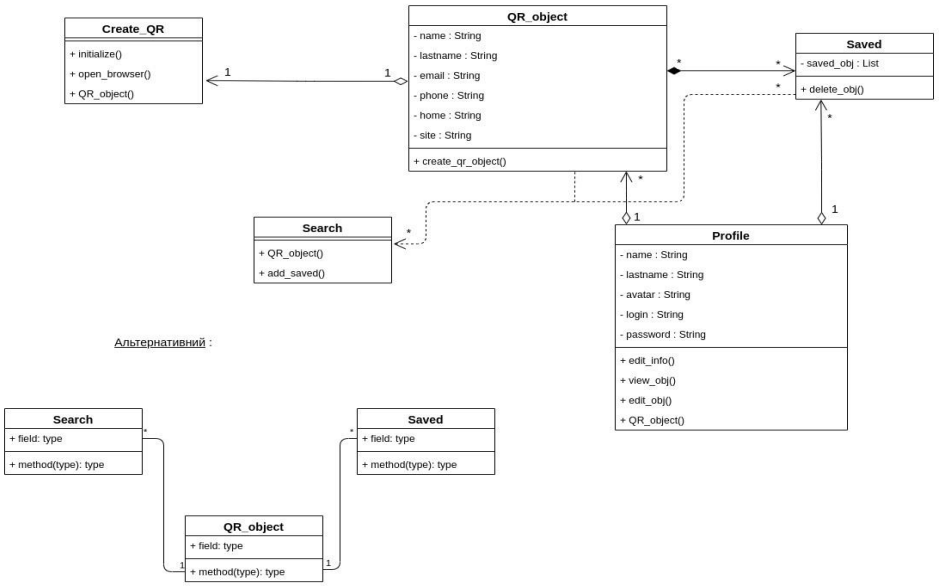


Рисунок Г.7 – Діаграма класів програмного модуля розпізнавання товарів за QR-кодом

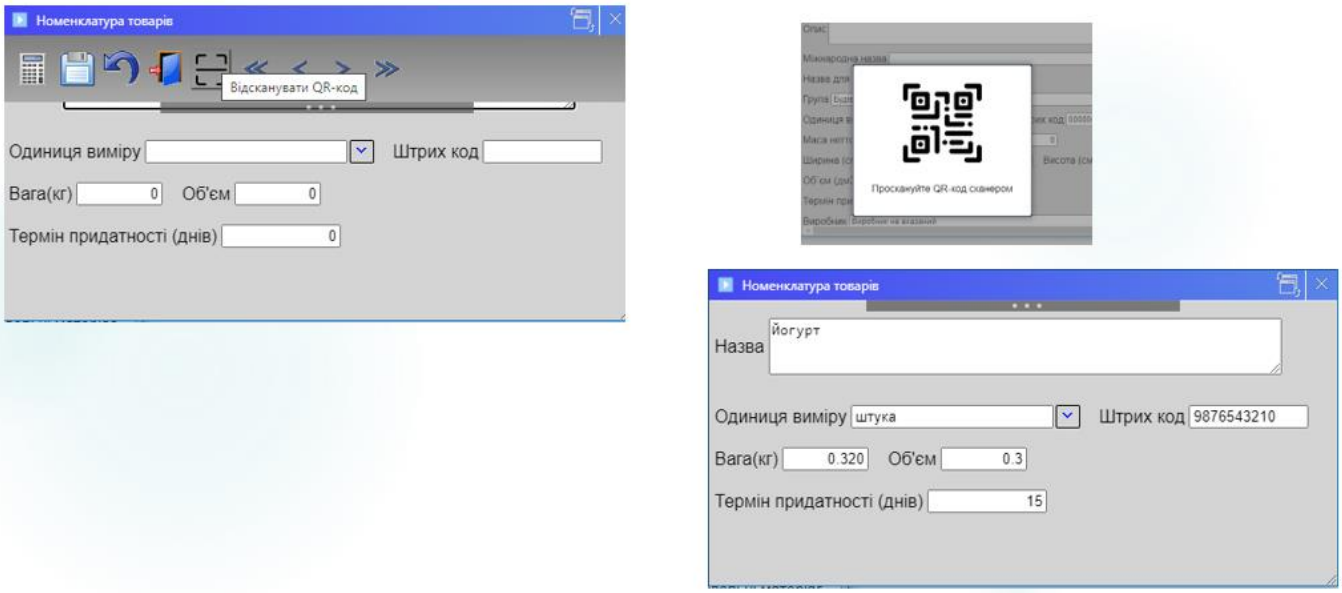


Рисунок Г.8 – Інтерфейс програмного модуля розпізнавання товарів за QR-кодом