**Міністерство освіти і науки України**
**Луцький національний технічний університет**
**Факультет комп'ютерних та інформаційних технологій**
**Кафедра комп'ютерних наук**

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

# РОЗРОБКА ТА ДОСЛІДЖЕННЯ WEB-ДОДАТКУ ДЛЯ QR КОДУ

# DEVELOPMENT AND RESEARCH OF A WEB APPLICATION FOR QR CODE

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КНм-21
Малецький Віталій Анатолійович
_____
(підпис)
Керівник: д.пед.н., професор
Тулашвілі Юрій Йосипович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Валерій Олександрович
_____
(підпис)

Луцьк – 2025 року

**ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерних наук*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *122 Комп'ютерні науки*
Освітня програма: *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Валерій ЛІЩИНА
«14» травня 2025 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА**
**ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ**

Малецький Віталій Анатолійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка та дослідження web-додатку для QR коду»
Керівник д.пед.н., професор Тулашвілі Юрій Йосипович
затверджені наказом закладу вищої освіти від «*14*» *травня* *2025* р. № 255/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«05» грудня 2025 р.*

3. Вихідні дані до роботи: _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
*Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, експерементальне дослідження результативності предмету дослідження.*

5. Перелік графічного матеріалу:

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Аналіз проблематики за темою роботи та постановка завдань дослідження* | *Тулашвілі Ю. Й.* | | |
| *Теоретичне дослідження та практична реалізація предмету дослідження* | *Тулашвілі Ю. Й.* | | |
| *Експеременталье дослідження результативності предмету дослідження* | *Тулашвілі Ю. Й.* | | |
| *Показник запозичень тексту* | _____ % | | |
| *Інструментальна перевірка* | *Кошелюк В. А.* | | |
| *Нормоконтроль* | *Сачук В. О.* | | |
| *Гарант ОПП* | *Ліщина В. О.* | | |

7. Дата видачі завдання *«14» травня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи бакалавра | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| *1* | *Провести огляд літературних джерел по темі кваліфікаційної роботи* | *до 30.06.2025 р* | |
| *2* | *Провести аналіз загальної проблеми і вибір напрямків дослідження* | *до 01.09.2025 р.* | |
| *3* | *Розробити функціональну схему роботи програмного продукту* | *до 01.10.2025 р* | |
| *4* | *Описати засоби розробки об'єкта проектування* | *до 15.10.2025 р.* | |
| *5* | *Практична реалізація об'єкта проектування* | *до 10.11.2025 р.* | |
| *6* | *Провести експеременталье дослідження результативності предмету дослідження* | *до 25.11.2025 р.* | |
| *7* | *Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру* | *до 05.12.2025 р.* | |

Здобувач вищої освіти _____ Віталій МАЛЕЦЬКИЙ

Керівник роботи _____ Юрій ТУЛАШВІЛІ

# АНОТАЦІЯ

Малецький В. А. Розробка та дослідження web-додатку для QR-коду. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, 3 розділів, висновків, списку використаних джерел та додатків.

Робота присвячена створенню та дослідженню інформаційної системи, що забезпечує повний життєвий цикл роботи з QR-кодами: генерацію, зберігання, керування та аналіз використання. Обґрунтовано актуальність теми, проаналізовано стандарти та існуючі сервіси, спроєктовано трирівневу клієнт-серверну архітектуру на основі стеку Django – PostgreSQL – React, розроблено інформаційну та програмну моделі системи, реалізовано REST-API, рольову модель доступу й підсистему збору статистики сканувань.

Експериментально оцінено продуктивність та зручність використання web-додатку: виміряно час генерації QR-кодів, час відповіді при скануванні за різних навантажень, проведено опитування користувачів. Результати показали низький час відповіді, стабільну роботу під навантаженням та високу суб'єктивну оцінку інтерфейсу. Практичне значення роботи полягає у можливості використання розробленого web-додатку як внутрішньої системи керування QR-кодами в організаціях та як навчально-методичної бази для дисциплін з інженерії програмного забезпечення та web-розробки.

Ключові слова: QR-код, web-додаток, Django, React, PostgreSQL, REST-API, динамічні QR-коди, аналітика сканувань.

# ABSTRACT

Vitalii Maletskyi. Development and research of a web application for qr code. Manuscript.

Master's thesis in Computer Science. Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, 3 chapters, conclusions, a list of references, and appendices.

Focuses on the design and analysis of an information system that supports the full life cycle of QR codes: generation, storage, management and usage analysis. The relevance is justified, existing standards and services are reviewed, and a three-tier client-server architecture based on the Django – PostgreSQL – React stack is designed. An information and software model of the system is developed, including a REST API, role-based access control and a subsystem for collecting scan statistics.

The web application’s performance and usability are evaluated experimentally: QR code generation time, response time for scanning under different loads, and user satisfaction are measured. The results show low response times, stable operation under significant load and high usability scores. The practical value of the work lies in the possibility of using the developed web application as an internal QR code management system in organizations and as a teaching resource for software engineering and web development courses.

Keywords: QR code, web application, Django, React, PostgreSQL, REST API, dynamic QR codes, scan analytics.

**ЗМІСТ**

# ВСТУП

Сучасна цифрова трансформація охоплює практично всі сфери діяльності – від бізнесу та освіти до державного управління. Одним з ключових інструментів для швидкого доступу до цифрового контенту став QR-код, який дозволяє миттєво переходити за посиланням, завантажувати файли, відкривати візитівки, підписуватись на події чи виконувати платіжні операції. Після пандемії COVID-19 та активної цифровізації публічних послуг використання QR-кодів різко зросло: вони стали стандартним елементом у маркетингових кампаніях, логістичних рішеннях, системах електронних квитків, освітніх платформах та внутрішніх інформаційних системах організацій.

Попри широку доступність онлайн-генераторів QR-кодів, більшість із них орієнтовані на разове створення статичних кодів і не забезпечують комплексного керування, збирання статистики сканувань, гнучкої інтеграції з інформаційними системами та належного рівня безпеки. Для організацій, яким необхідно системно працювати з великою кількістю QR-кодів (університети, компанії, заклади культури, сервісні підприємства), актуальною є розробка спеціалізованого web-додатку, що дозволяє централізовано створювати, організовувати, оновлювати та аналізувати використання QR-кодів, а також керувати правами доступу користувачів.

Окремого значення набувають питання захисту даних та достовірності цільових ресурсів, на які ведуть QR-коди. Наявність фішингових посилань, підміни контенту та відсутність контролю за оновленням інформації створюють ризики як для кінцевих користувачів, так і для організацій, що використовують QR-коди у своїй діяльності. Тому розробка web-додатку має поєднувати зручність роботи з кодами, підтримку сучасних web-технологій та механізми безпечного доступу й моніторингу.

Таким чином, тема кваліфікаційної роботи «Розробка та дослідження web-додатку для QR-коду» є актуальною з точки зору розвитку прикладних

інформаційних систем, цифровізації сервісів та підвищення ефективності використання QR-кодів у різних предметних областях.

Мета роботи полягає в розробці та дослідженні web-додатку для генерації, зберігання, керування та аналітики QR-кодів, який забезпечує зручний користувацький інтерфейс, гнучку конфігурацію параметрів кодів, підтримку авторизації та моніторинг їх використання.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

− провести аналіз сучасного стану використання QR-кодів у веб-орієнтованих системах та існуючих програмних рішень для їх генерації та керування;

− дослідити наявні алгоритми кодування інформації в QR-код, бібліотеки та інструментарій для роботи з QR-кодами в середовищі web-розробки;

− обґрунтувати вибір архітектури web-додатку, технологічного стеку, моделей даних та способів інтеграції з іншими системами;

− спроєктувати інформаційну, функціональну та програмну архітектуру web-додатку із використанням UML-діаграм;

− реалізувати прототип web-додатку для генерації та керування QR-кодами, включаючи модулі автентифікації користувачів, адміністрування кодів та перегляду статистики;

− розробити методику експериментального дослідження швидкодії, надійності та зручності користування розробленим web-додатком, а також провести порівняння з існуючими сервісами;

− виконати обробку та аналіз результатів експерименту, сформулювати висновки щодо ефективності запропонованого рішення та можливих напрямів його подальшого розвитку.

Об’єкт дослідження – процес організації та підтримки взаємодії користувачів із цифровими ресурсами за допомогою QR-кодів у web-середовищі.

Предмет дослідження – моделі, методи та програмні засоби розробки web-додатку для генерації, зберігання, керування та аналітики QR-кодів.

У роботі використовуються методи аналізу та синтезу наукових джерел, методи системного аналізу, UML-моделювання, математичне моделювання та формалізація інформаційних процесів, алгоритмічні методи побудови QR-кодів, методи проєктування і розробки web-орієнтованих інформаційних систем, а також експериментальні методи дослідження продуктивності та якості програмного забезпечення.

Наукова новизна роботи полягає в розробці узагальненої архітектурної моделі web-додатку для керування QR-кодами, що поєднує засоби генерації статичних та динамічних QR-кодів, механізми централізованого керування контентом та підсистему збирання й аналізу статистики сканувань на основі журналів запитів до серверу додатку.

Практичне значення одержаних результатів полягає у можливості використання розробленого web-додатку в організаціях різного профілю для створення власних систем QR-навігації: в університетах (навігація кампусом, доступ до електронних матеріалів, реєстрація на події), у закладах культури та туризму, сервісних компаніях, торговельних мережах. Програмний продукт може бути розгорнутий на власних серверах або в хмарному середовищі та адаптований до конкретних потреб замовника.

# РОЗДІЛ 1
# АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

## 1.1 Аналіз сучасного стану проблеми та предметної області

QR-код (Quick Response Code) належить до двовимірних штрихових кодів матричного типу, які кодують інформацію як по горизонталі, так і по вертикалі, що забезпечує значно більшу інформаційну ємність у порівнянні з одновимірними штрихкодами [1-2]. Міжнародний стандарт ISO/IEC 18004:2015 та його національна адаптація ДСТУ ISO/IEC 18004:2019 визначають вимоги до структури символіки QR-коду, методів кодування даних, форматів символів, рівнів корекції помилок і алгоритмів декодування.

Технологія QR-коду була розроблена компанією DENSO WAVE у 1994 році для потреб автомобільної промисловості Японії й спочатку застосовувалась для маркування комплектуючих на виробництві [3]. Згодом, завдяки відкритості специфікації та просуванню QR-коду як відкритого стандарту, він набув широкого поширення у різних галузях: логістиці, роздрібній торгівлі, маркетингу, транспорті, медицині, освіті та електронних платежах [3-4].

Стандарти [5] визначають кілька варіантів символіки QR-коду (Model 1, Model 2, Micro QR, Frame QR), що відрізняються максимальною кількістю модулів, обсягом кодованих даних та можливостями дизайну. Рівні корекції помилок (L, M, Q, H) дозволяють зберігати коректність декодування навіть за часткового пошкодження зображення, що є критично важливим у реальних умовах експлуатації (забруднення етикеток, складні умови освітлення).

Останніми роками простежується суттєве зростання використання QR-кодів. Аналітичні дослідження показують, що після пандемії COVID-19 QR-коди стали базовим інструментом для реалізації безконтактних сервісів – меню в ресторанах, реєстрації відвідувачів, електронних квитків, операцій [6-8]. Дослідження в бізнес-середовищі демонструють майже двократне зростання

використання QR-кодів у маркетингових кампаніях та комунікації з клієнтами в період 2018-2020 років [7].

У наукових публікаціях останніх років QR-коди розглядаються як базовий інструмент зв'язку фізичного та цифрового простору. З'являються оглядові роботи, присвячені структурі, алгоритмам кодування, помилкостійкості та прикладним аспектам їх використання [5-6]. Наприклад, у [5] узагальнено сучасні підходи до формування матриці QR-коду, схеми корекції помилок та варіанти застосувань у мобільних платежах, логістиці, маркетингу, системах автентифікації та відстеження об'єктів. У роботі [6] акцентовано увагу на перспективах використання QR-кодів у поєднанні з хмарними сервісами та мобільними додатками.

Важливим напрямом досліджень є аналіз безпеки та конфіденційності застосувань QR-кодів. Оглядова праця [7] містить класифікацію ризиків, пов'язаних із використанням QR-кодів (фішингові посилання, підміна контенту, шкідливі переспрямування), та пропонує загальні рекомендації щодо захисту, зокрема використання підписаних URL-адрес, валідації цільових ресурсів на стороні сервера та інтеграції з інструментами моніторингу безпеки.

Дослідження поведінкових аспектів показують, що сприйняття QR-кодів користувачами значною мірою залежить від контексту використання, дизайну та довіри до бренду або джерела, яке надає код [8-9]. Зокрема, Shin у [9] аналізує чинники, що впливають на готовність користувачів сканувати QR-коди – очікувана користь, зручність, сприйняття ризику та рівень технічної грамотності. Новіші роботи демонструють, що у сфері роздрібної торгівлі й електронної комерції QR-коди сприяють підвищенню залученості споживачів та формуванню поведінкових патернів, пов'язаних із використанням безготівкових розрахунків та «розумних» етикеток [10].

У сфері опитувань та збору даних QR-коди застосовуються для залучення респондентів до веб-анкет, що дозволяє зменшити бар'єр входу та скоротити час доступу до опитувальника [11]. В освітніх системах вони використовуються для швидкого доступу до навчальних ресурсів, проведення тестів, реєстрації

відвідування, організації квестів і гейміфікованої діяльності. Таким чином, QR-коди стали універсальним механізмом зв’язку між друкованими матеріалами, фізичними об’єктами та цифровими сервісами.

Разом з тим переважна більшість існуючих публічних онлайн-генераторів QR-кодів орієнтована на прості разові операції, не забезпечує централізованого керування кодами, гнучкого адміністрування прав доступу, інтегрованої аналітики сканувань і механізмів контролю безпеки. Для організацій, що працюють з великою кількістю QR-кодів (університети, компанії, заклади культури, торговельні мережі), актуальним стає створення спеціалізованих web-додатків, здатних підтримувати повний життєвий цикл QR-коду – від генерації та прив’язки до контенту до моніторингу використання й аналізу ефективності.

Саме ці обмеження існуючих інструментів та зростаючі вимоги до безпечного й керованого використання QR-кодів визначають актуальність розробки та дослідження web-додатку, присвяченого комплексній роботі з QR-кодами в організаційному контексті.

## 1.2 Огляд і аналіз методів та засобів розробки web-додатків для роботи з QR-кодами

Сучасні web-додатки для роботи з QR-кодами, як правило, будуються на клієнт-серверній архітектурі із чітким поділом на фронтенд, бекенд і рівень зберігання даних. Найбільш поширеним підходом є трирівнева архітектура «клієнт – сервер застосунку – база даних» з використанням REST-орієнтованих web-сервісів для обміну даними між клієнтом і сервером. Бекенд відповідає за реалізацію бізнес-логіки, включаючи генерацію QR-кодів, керування користувачами, фіксацію подій сканування та формування аналітичних звітів; фронтенд забезпечує інтерфейс для взаємодії з користувачем і візуалізацію даних.

На серверній частині широко застосовуються фреймворки Django та Flask на мові Python, а також Node.js у поєднанні з Express для JavaScript-орієнтованих

рішень [11]. Django пропонує цілісну екосистему для побудови web-додатків: вбудовану ORM, систему автентифікації, механізми маршрутизації, шаблонізатор, інструменти для створення REST-API та інтегрований адміністративний інтерфейс. У працях детально описано організацію структури Django-проєктів, роботу з моделями даних, побудову API та підходи до забезпечення безпеки web-застосунків.

Для збереження даних застосовуються реляційні СКБД PostgreSQL і MySQL, а також документо-орієнтовані бази даних (наприклад, MongoDB). У контексті систем для керування QR-кодами реляційні бази даних залишаються поширеним вибором завдяки підтримці транзакцій, цілісності даних та складних запитів, необхідних для побудови аналітичних звітів за подіями сканування, користувачами та проєктами. Приклади побудови моделей даних для web-додатків на основі Django й PostgreSQL наведено в.

Фронтенд-частина сучасних систем, орієнтованих на інтенсивну взаємодію з користувачем, найчастіше реалізується з використанням фреймворків React, Vue.js або Angular. React, офіційна документація якого постійно оновлюється й підтримує сучасні підходи до побудови інтерфейсів на основі компонентів та хуків, широко застосовується для розробки односторінкових застосунків (SPA). Книги та інші посібники з фронтенд-розробки демонструють типові патерни організації стану, маршрутизації, керування формами та інтеграції з REST-API, що особливо актуально для панелей керування QR-кодами.

Для безпосередньої роботи з QR-кодами існує значна кількість бібліотек, як на стороні сервера, так і на стороні клієнта. У серверних фреймворках на Python застосовуються бібліотеки для генерації QR-кодів у форматах PNG, SVG або Base64, які інтегруються у бізнес-логіку для побудови зображень на вимогу. У JavaScript-екосистемі поширені клієнтські бібліотеки, що дозволяють як відображати QR-коди, так і декодувати їх за допомогою камери пристрою. Оглядові публікації [5-7] узагальнюють вимоги до таких бібліотек: підтримка різних рівнів корекції помилок, можливість налаштування розмірів та дизайну коду, продуктивність при інтеграції в мобільні та web-сценарії.

У прикладних дослідженнях розглядаються різні варіанти архітектури систем керування QR-кодами. Частина авторів описує хмарні платформи, які дозволяють генерувати, розповсюджувати, читати та логувати QR-коди (наприклад, концепція Q platform® від DENSO). В інших роботах фокус робиться на вбудованих компонентах для мобільних додатків, інтегрованих із серверними частинами для відстеження сканувань та аналізу поведінки користувачів [5-7].

Ключовими характеристиками, що визначають якість таких систем, є:

− масштабованість – здатність обслуговувати зростаючу кількість користувачів, кодів і подій сканування без значної деградації продуктивності;

− безпека – захист від ін’єкції шкідливих посилань, підміни контенту, несанкціонованого доступу до аналітики;

− зручність інтерфейсу – інтуїтивність, адаптивність до різних пристроїв, швидкість виконання типових операцій;

− аналітичні можливості – наявність гнучких інструментів для відстеження динаміки сканувань, побудови графіків, експорту звітів.

Огляд наукових праць і практичних рішень показує, що значна частина існуючих систем є або вузькоспеціалізованими (наприклад, лише для маркетингових кампаній або мобільних платежів), або комерційними сервісами з обмеженим доступом до вихідного коду та можливостей інтеграції. Це створює нішу для розробки відкритішого, гнучкого web-додатку, який можна адаптувати під потреби університету, компанії чи іншої організації, інтегрувати з наявними ІС, а також використовувати як навчальний і дослідницький полігон для студентів-програмістів.

### 1.3 Постановка завдання на кваліфікаційну роботу

На основі проведеного огляду формулюється узагальнена постановка задачі. Необхідно розробити web-додаток, що забезпечує генерацію, збереження

та адміністрування QR-кодів, підтримує авторизацію користувачів, розмежування ролей і надає засоби аналітики використання QR-кодів.

У загальному вигляді web-додаток повинен реалізовувати такі функції: реєстрація та автентифікація користувачів; створення, редагування й видалення QR-кодів з різними типами контенту (URL-адреса, текст, контактні дані, посилання на файл); групування QR-кодів за проєктами; автоматичне генерування коротких посилань-переадресацій для реалізації динамічних QR-кодів; зберігання подій сканування з фіксацією часу, типу пристрою та джерела переходу; відображення статистики у вигляді таблиць і графіків; експорт даних для подальшої обробки.

До вхідних даних належать облікові записи користувачів з різними ролями (адміністратор, редактор, переглядач), параметри QR-кодів та пов’язаний із ними контент, конфігурація рівня доступу до окремих проєктів, а також інформація про події сканування. Вихідними даними є згенеровані зображення QR-кодів, короткі посилання для їх використання, а також аналітичні звіти за параметрами використання.

# РОЗДІЛ 2
# ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ ДЛЯ QR-КОДУ

## 2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання

Розробка web-додатку для роботи з QR-кодами потребує чіткого формалізованого опису архітектури, інформаційних потоків, моделі даних та алгоритмів функціонування. На цьому етапі формуються теоретичні засади системи, які згодом реалізуються у вигляді конкретних програмних модулів.

Архітектура розробленого web-додатку базується на трирівневій клієнт-серверній моделі, у якій фронтенд, бекенд і база даних логічно відокремлені та взаємодіють через стандартизований інтерфейс. На рисунку 2.1 подано узагальнену архітектуру системи. У верхній частині схеми відображено веббраузер або мобільний браузер користувача, через який здійснюється доступ до React-інтерфейсу. Клієнтська частина надсилає HTTP(S)-запити до REST-API, реалізованого на стороні бекенду за допомогою фреймворку Django. Сервер обробляє ці запити, взаємодіє з базою даних PostgreSQL для збереження та отримання інформації про користувачів, проєкти, QR-коди та події сканування, а також використовує бібліотеку генерації QR-кодів для формування растрових або векторних зображень.

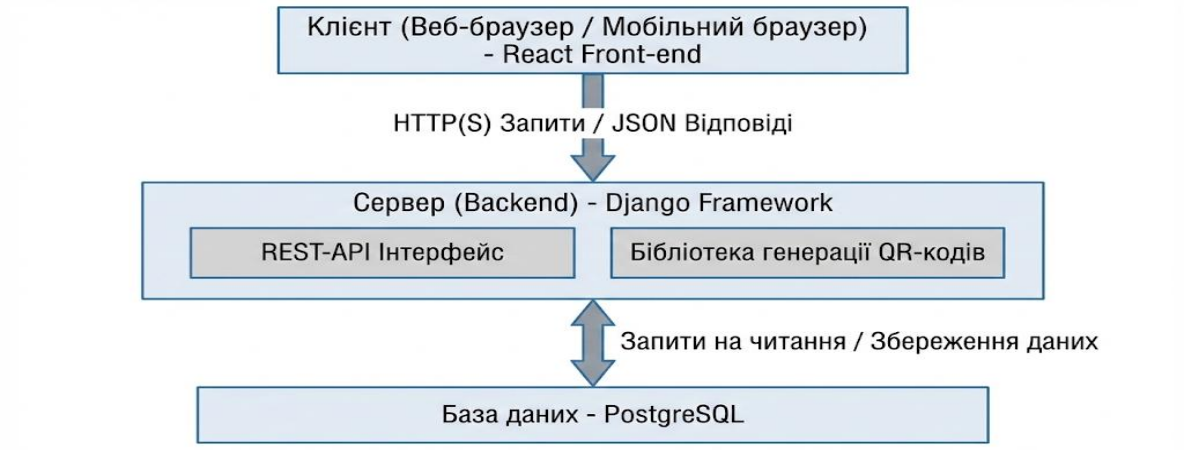


Рисунок 2.1 – Узагальнена архітектура web-додатку для роботи з QR-кодами

Функціональна модель системи відображається за допомогою UML-діаграми прецедентів, наведеної на рисунку 2.2. У якості акторів виступають «Адміністратор», «Редактор» та «Переглядач». Адміністратор відповідає за створення облікових записів, керування ролями, налаштування глобальних параметрів системи та моніторинг стану сервісу. Редактор створює й редагує проєкти, генерує статичні та динамічні QR-коди, змінює їхній контент і аналізує статистику в межах власних проєктів. Переглядач має доступ до перегляду списку QR-кодів та аналітичних звітів, але не може змінювати структуру системи. На діаграмі виділено прецеденти «Увійти до системи», «Керувати проєктами», «Генерувати QR-код», «Переглядати статистику», «Експортувати звіт», які задають основні сценарії використання web-додатку.

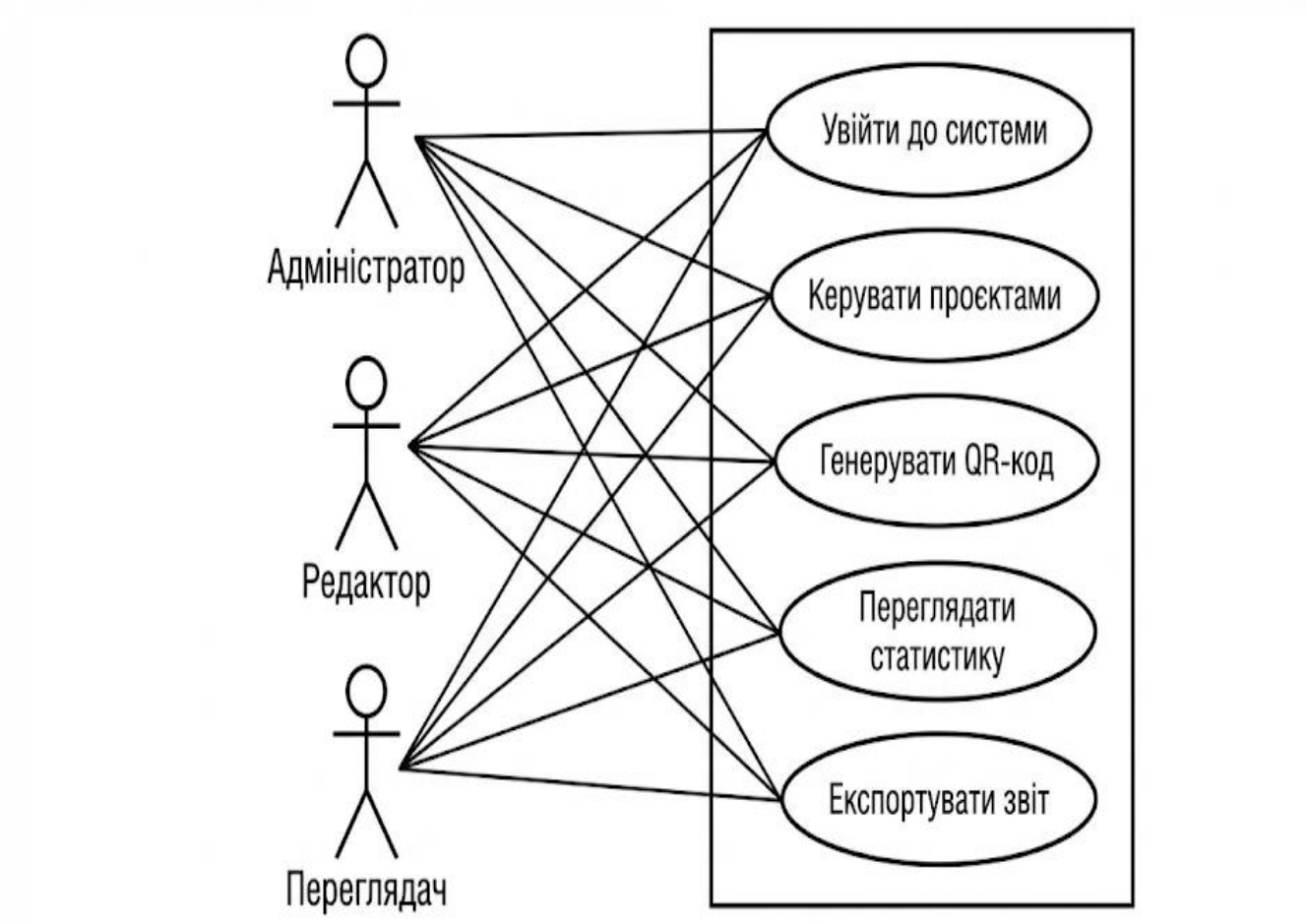


Рисунок 2.2 – UML-діаграма прецедентів web-додатку для QR-кодів

Інформаційну структуру системи описує ER-модель бази даних, подана на рисунку 2.3. Основними сутностями є: `User` (користувач системи), `Role` (роль), `Project` (проєкт), `QrCode` (QR-код) та `ScanEvent` (подія сканування). Користувач може мати одну або кілька ролей, тому між сутностями `User` і `Role` існує зв'язок «багато-до-багатьох», реалізований через проміжну таблицю. Проєкт належить певному власникові (зв'язок «багато-до-одного» з `User`), а кожен QR-код прив'язаний до конкретного проєкту. Подія сканування пов'язана з конкретним QR-кодом та включає інформацію про час сканування, IP-адресу, тип клієнта й інші атрибути.

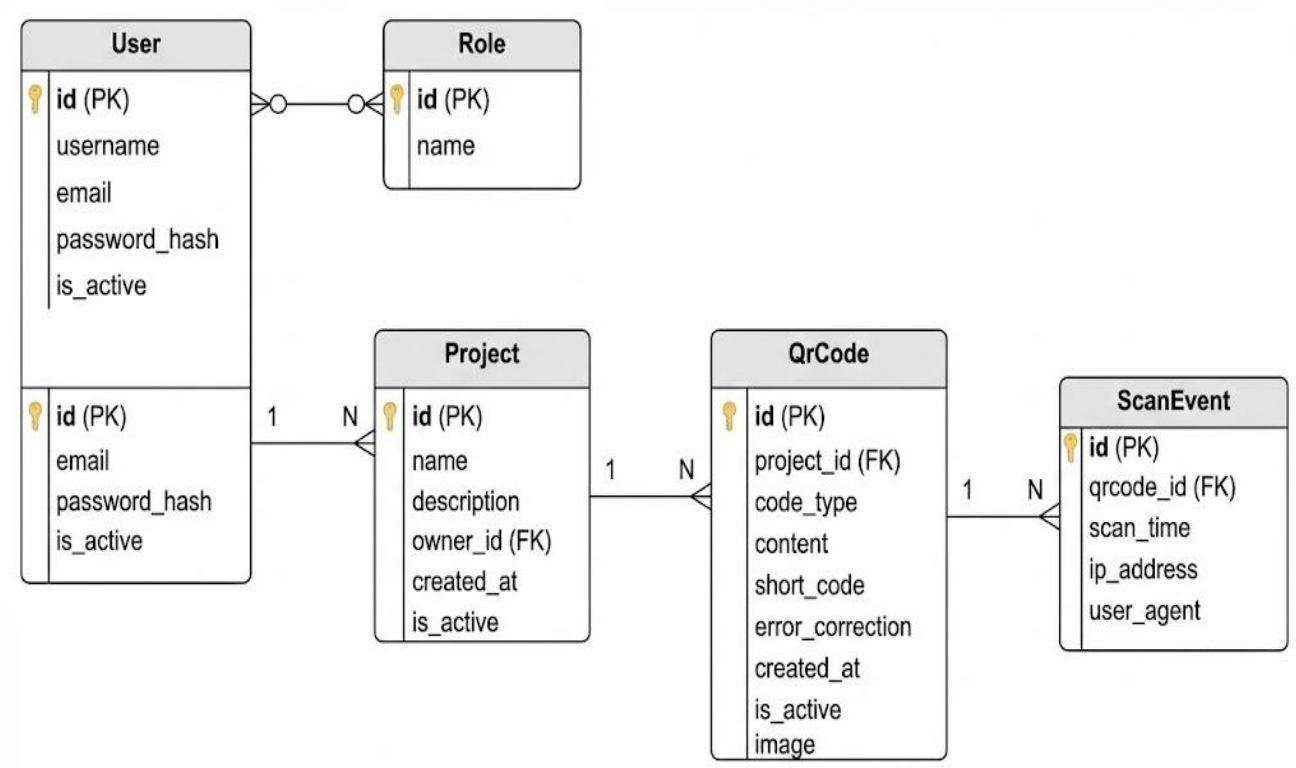


Рисунок 2.3 – ER-діаграма бази даних web-додатку

У таблиці 2.1 наведено перелік основних сутностей і ключових атрибутів бази даних. Така формалізація дозволяє систематизувати інформаційну модель та пов'язати її з подальшою реалізацією у вигляді Django-моделей.

Таблиця 2.1 – Основні сутності та атрибути бази даних web-додатку

| Сутність | Атрибут | Тип даних (логічний) | Призначення |
|---|---|---|---|
| User | id | цілочисловий | Унікальний ідентифікатор користувача |
| | username | рядковий | Логін для входу |
| | email | рядковий | Електронна пошта |
| | password_hash | рядковий | Хешований пароль |
| | is_active | булевий | Статус активності |
| Role | id | цілочисловий | Ідентифікатор ролі |
| | name | рядковий | Назва ролі (admin, editor, viewer) |
| Project | id | цілочисловий | Ідентифікатор проєкту |
| | name | рядковий | Назва проєкту |
| | description | текстовий | Опис проєкту |
| | owner_id | цілочисловий (FK) | Посилання на власника (User) |
| QrCode | id | цілочисловий | Ідентифікатор QR-коду |
| | project_id | цілочисловий (FK) | Посилання на проєкт |
| | code_type | рядковий | Тип контенту (URL, TEXT, VCARD тощо) |
| | content | текстовий | Закодований в QR-коді контент |
| | short_code | рядковий | Укорочений ідентифікатор для URL |
| | error_correction | рядковий | Рівень корекції помилок |
| | created_at | дата/час | Дата й час створення |
| | is_active | булевий | Ознака активності коду |
| ScanEvent | id | цілочисловий | Ідентифікатор події |
| | qrcode_id | цілочисловий (FK) | Посилання на QR-код |
| | scan_time | дата/час | Час сканування |
| | ip_address | рядковий | IP-адреса клієнта |
| | user_agent | текстовий | Інформація про клієнтський пристрій |

Алгоритм роботи з QR-кодами передбачає два ключові процеси: генерацію зображення та обробку подій сканування. Схема алгоритму генерації QR-коду подана на рисунку 2.4. На початковому етапі здійснюється валідація вхідних даних: перевіряється коректність URL-адреси або іншого типу контенту, довжина рядка, відсутність заборонених символів. Далі визначається режим кодування (numeric, alphanumeric, byte), виконується побудова інформаційних блоків та застосовується алгоритм корекції помилок на основі коду Ріда-Соломона. Після формування матриці модулів відбувається рендеринг QR-коду у вибраному графічному форматі (PNG, SVG чи Base64), а сформоване зображення зберігається на диску або у вигляді двійкового поля в базі даних.

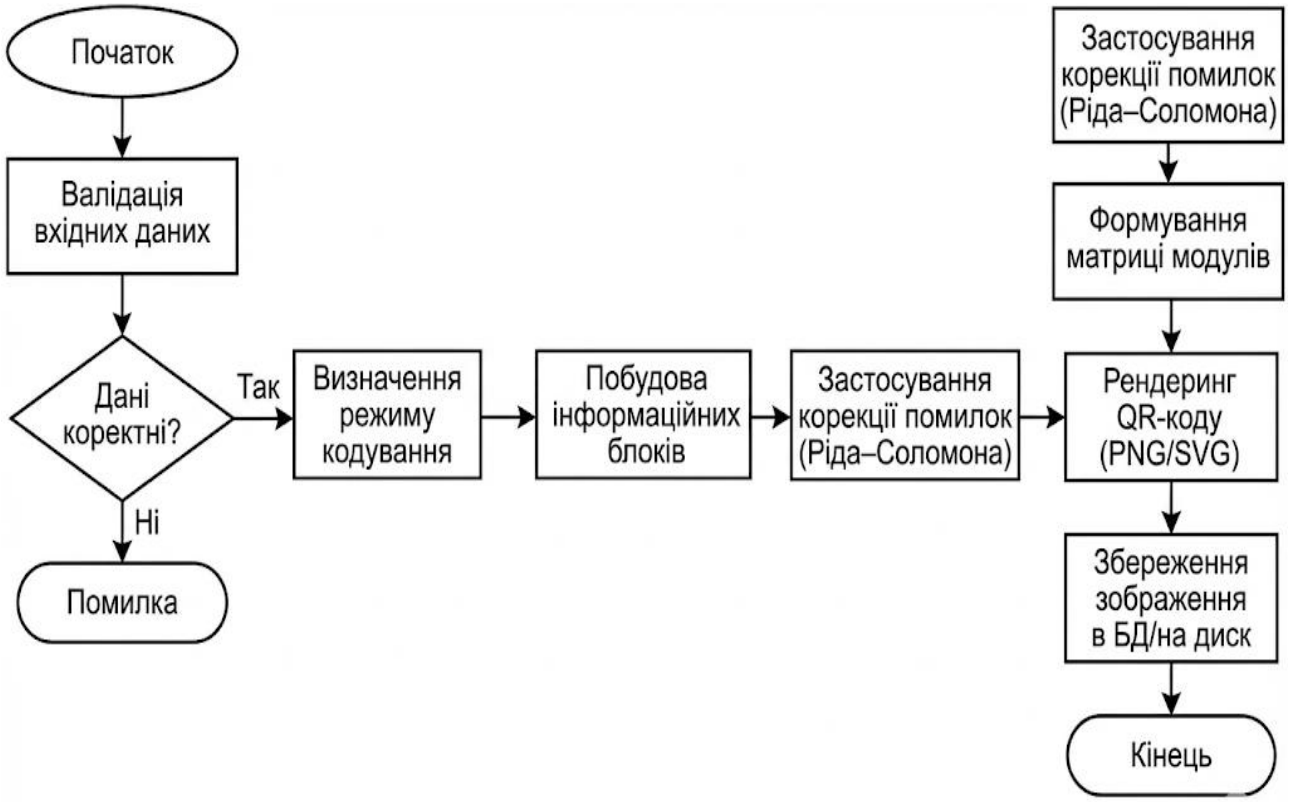


Рисунок 2.4 – Алгоритм генерації QR-коду у web-додатку

Процес обробки сканування динамічного QR-коду представлено на діаграмі послідовності (рис. 2.5). Після сканування коду мобільний пристрій формує HTTP-запит до сервера за укороченою адресою, у якій міститься короткий ідентифікатор `short_code`. Бекенд отримує запит, виконує пошук відповідного запису `QrCode` у базі даних, створює новий запис `ScanEvent`, зберігаючи інформацію про час сканування, IP-адресу та `User-Agent`, після чого здійснює HTTP-переадресацію (код 302) на цільовий ресурс, який заданий у полі `content`. Така схема дозволяє поєднати гнучкість динамічних QR-кодів (можливість змінювати цільове посилання без перевипуску коду) і отримання розширеної статистики використання.

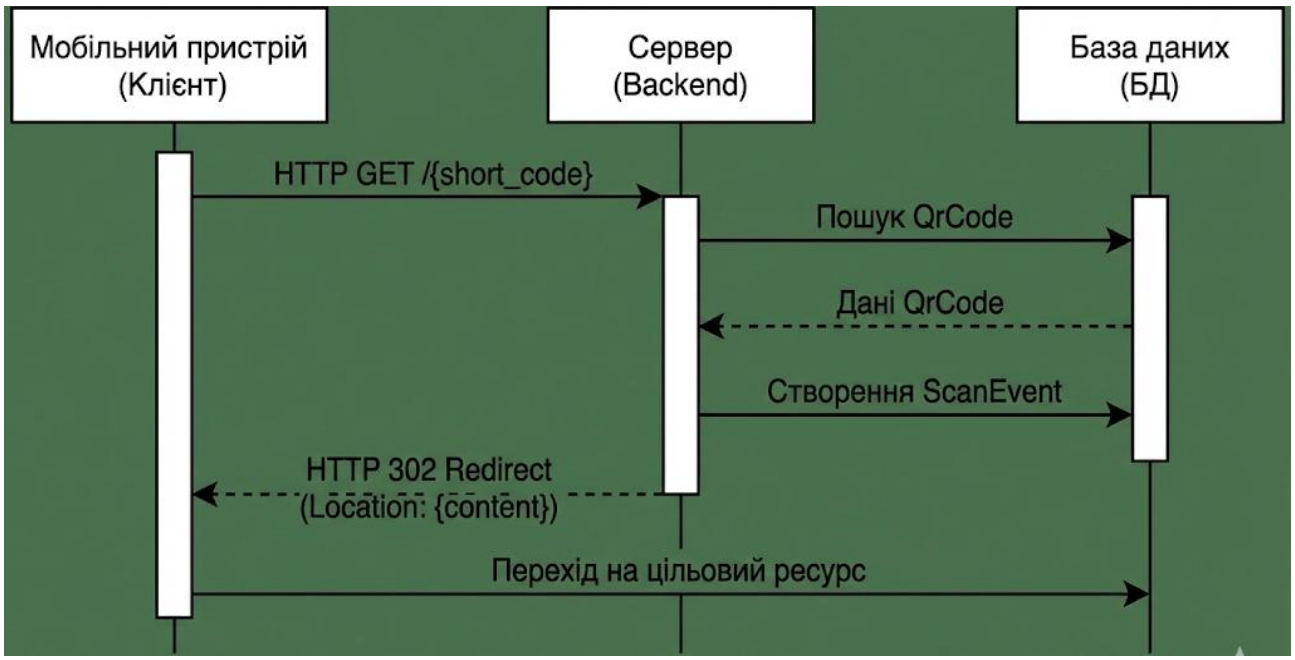


Рисунок 2.5 – Діаграма послідовності обробки сканування QR-коду

Описані архітектурні, інформаційні та алгоритмічні рішення безпосередньо базуються на проаналізованих технологіях та бібліотеках і створюють теоретичну основу для подальшої практичної реалізації web-додатку.

## 2.2 Практична реалізація об’єкта проєктування

Практична реалізація web-додатку для роботи з QR-кодами виконується на основі обґрунтованої в попередньому підрозділі архітектури та інформаційної моделі. Серверна частина реалізована у вигляді Django-проєкту, фронтенд – за допомогою фреймворку React, а дані зберігаються в базі PostgreSQL.

Структура серверного проєкту відображена на рисунку 2.6. Базовий каталог містить конфігураційний модуль `config` з файлами `settings.py`, `urls.py`, `wsgi.py`, а також окремі додатки `accounts`, `qrcodes`, `analytics` та `api`, кожен з яких відповідає за окремий аспект предметної області. У додатку `accounts` визначені моделі користувачів і ролей, `qrcodes` містить моделі та логіку генерації QR-кодів, у `analytics` розміщено моделі та бізнес-логіка, пов’язані з подіями сканування, а додаток `api` забезпечує REST-інтерфейс для взаємодії фронтенду з бекендом.

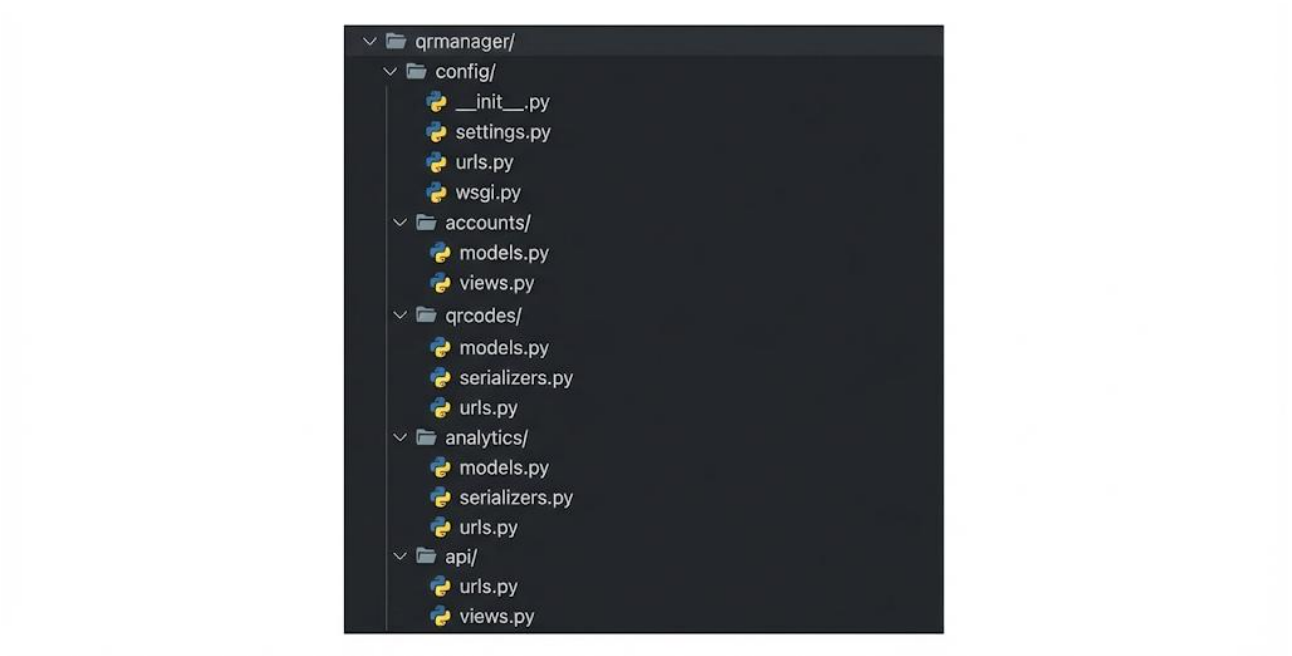


Рисунок 2.6 – Структура каталогів Django-проєкту web-додатку

Фрагмент дерева проєкту можна подати як допоміжний текстовий лістинг, що полегшує розуміння архітектури.

Лістинг 2.1 – Фрагмент структури каталогів Django-проєкту

```
qrmanager/
├─ config/
│   ├─ __init__.py
│   ├─ settings.py
│   ├─ urls.py
│   └─ wsgi.py
├─ accounts/
│   ├─ models.py
│   ├─ views.py
│   ├─ serializers.py
│   └─ urls.py
├─ qrcodes/
│   ├─ models.py
│   ├─ services.py
│   ├─ views.py
│   ├─ serializers.py
│   └─ urls.py
├─ analytics/
│   ├─ models.py
│   ├─ views.py
│   ├─ serializers.py
│   └─ urls.py
└─ api/
    ├─ urls.py
    └─ views.py
```

кінець лістингу 2.1

Моделі бази даних у Django відповідають ER-діаграмі, розглянутій у підрозділі 2.1. Фрагмент моделі `Project` наведено в лістингу 2.2. Клас містить базові атрибути проєкту, посилання на власника та метод рядкового представлення, що використовується в адмін-панелі.

Лістинг 2.2 – Модель Project у Django

```
# qrcodes/models.py

from django.conf import settings
from django.db import models


class Project(models.Model):
    name = models.CharField(max_length=200, verbose_name="Назва проєкту")
    description = models.TextField(blank=True, verbose_name="Опис проєкту")
    owner = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="projects",
        verbose_name="Власник"
    )
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата
створення")
    is_active = models.BooleanField(default=True, verbose_name="Активний")

    class Meta:
        verbose_name = "Проєкт"
        verbose_name_plural = "Проєкти"
        ordering = ["-created_at"]

    def __str__(self):
        return self.name
```

кінець лістингу 2.2

Модель `QrCode`, наведена в лістингу 2.3, реалізує сутність QR-коду. Вона містить посилання на проєкт, тип контенту, власне контент, укорочений код для динамічних посилань, рівень корекції помилок та поле для зберігання зображення.

Лістинг 2.3 – Модель QrCode у Django

```
# qrcodes/models.py

class QrCode(models.Model):
    CODE_TYPE_CHOICES = [
        ("URL", "Web-посилання"),
        ("TEXT", "Текст"),
        ("VCARD", "Візитівка"),
    ]
```

```
    ERROR_CORRECTION_CHOICES = [
        ("L", "L – 7%"),
        ("M", "M – 15%"),
        ("Q", "Q – 25%"),
        ("H", "H – 30%"),
    ]

    project = models.ForeignKey(
        Project,
        on_delete=models.CASCADE,
        related_name="qrcodes",
        verbose_name="Проєкт"
    )
    code_type = models.CharField(
        max_length=10,
        choices=CODE_TYPE_CHOICES,
        default="URL",
        verbose_name="Тип контенту"
    )
    content = models.TextField(verbose_name="Контент")
    short_code = models.CharField(
        max_length=32,
        unique=True,
        verbose_name="Короткий код"
    )
    error_correction = models.CharField(
        max_length=1,
        choices=ERROR_CORRECTION_CHOICES,
        default="M",
        verbose_name="Рівень корекції"
    )
    image = models.ImageField(
        upload_to="qrcodes/",
        blank=True,
        null=True,
        verbose_name="Зображення QR-коду"
    )
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата
створення")
    is_active = models.BooleanField(default=True, verbose_name="Активний")

    class Meta:
        verbose_name = "QR-код"
        verbose_name_plural = "QR-коди"
        ordering = ["-created_at"]

    def __str__(self):
        return f"{self.get_code_type_display()} ({self.short_code})"
```

кінець лістингу 2.3

Події сканування моделюються класом `ScanEvent`, наведений у лістингу 2.4. Для кожного сканування зберігається посилання на QR-код, час сканування, IP-адреса та інформація про клієнтський пристрій.

Лістинг 2.4 – Модель ScanEvent у Django

```
# analytics/models.py

class ScanEvent(models.Model):
    qrcode = models.ForeignKey(
        "qrcodes.QrCode",
        on_delete=models.CASCADE,
        related_name="scan_events",
        verbose_name="QR-код"
    )
    scan_time = models.DateTimeField(auto_now_add=True, verbose_name="Час
сканування")
    ip_address = models.GenericIPAddressField(
        null=True, blank=True, verbose_name="IP-адреса"
    )
    user_agent = models.TextField(blank=True, verbose_name="User-Agent")

    class Meta:
        verbose_name = "Подія сканування"
        verbose_name_plural = "Події сканування"
        ordering = ["-scan_time"]

    def __str__(self):
        return f"{self.qrcode.short_code} @ {self.scan_time}"
```

кінець лістингу 2.4

Відповідність моделей Django та таблиць бази даних схематично подана в таблиці 2.2.

Таблиця 2.2 – Відповідність моделей Django і таблиць БД

| Модель Django | Таблиця БД | Основні поля |
|---|---|---|
| Project | qrcodes_project | id, name, description, owner_id, created_at |
| QrCode | qrcodes_qrcode | id, project_id, code_type, content, short_code, error_correction, image, created_at |
| ScanEvent | analytics_scanevent | id, qrcode_id, scan_time, ip_address, user_agent |
| User | accounts_user | id, username, email, password, is_active |
| Role | accounts_role | id, name |

Наступним кроком є реалізація REST-інтерфейсу. Для створення та отримання QR-кодів використовується клас-представлення на основі Django REST Framework. Фрагмент реалізації наведено в лістингу 2.5.

Лістинг 2.5 – REST-представлення для роботи з QR-кодами

```
# qrcodes/views.py

from rest_framework import viewsets, permissions
from .models import QrCode, Project
from .serializers import QrCodeSerializer
from .services import generate_qr_image
```

```
class QrCodeViewSet(viewsets.ModelViewSet):
    queryset = QrCode.objects.all()
    serializer_class = QrCodeSerializer
    permission_classes = [permissions.IsAuthenticated]

    def get_queryset(self):
        return QrCode.objects.filter(
            project__owner=self.request.user,
            project__is_active=True
        )

    def perform_create(self, serializer):
        qrcode = serializer.save()
        generate_qr_image(qrcode)
```

кінець лістингу 2.5

У лістингу 2.6 наведено сервісну функцію, яка безпосередньо генерує зображення QR-коду. У реальній роботі тут використовується конкретна бібліотека для генерації QR-кодів, але в лістингу подано узагальнений приклад.

Лістинг 2.6 – Сервіс генерації зображення QR-коду

```
# qrcodes/services.py

import io
import qrcode
from django.core.files.base import ContentFile


def generate_qr_image(qrcode_obj):
    qr = qrcode.QRCode(
        version=None,
        error_correction=getattr(qrcode.constants,
f"ERROR_CORRECT_{qrcode_obj.error_correction}"),
        box_size=10,
        border=4,
    )
    qr.add_data(qrcode_obj.content)
    qr.make(fit=True)

    img = qr.make_image(fill_color="black", back_color="white")

    buffer = io.BytesIO()
    img.save(buffer, format="PNG")
    file_name = f"qr_{qrcode_obj.short_code}.png"
    qrcode_obj.image.save(file_name, ContentFile(buffer.getvalue()), save=True)
```

кінець лістингу 2.6

Окремий ендпоінт реалізує сценарій обробки сканування динамічного QR-коду за коротким кодом. Фрагмент такого представлення наведено в лістингу 2.7.

Після знаходження відповідного `QrCode` створюється запис `ScanEvent`, а потім виконується переадресація на цільовий ресурс.

Лістинг 2.7 – Обробка сканування динамічного QR-коду

```
# api/views.py

from django.shortcuts import get_object_or_404, redirect
from django.utils import timezone

from qrcodes.models import QrCode
from analytics.models import ScanEvent


def redirect_by_short_code(request, short_code: str):
    qrcode = get_object_or_404(QrCode, short_code=short_code, is_active=True)

    ScanEvent.objects.create(
        qrcode=qrcode,
        scan_time=timezone.now(),
        ip_address=request.META.get("REMOTE_ADDR"),
        user_agent=request.META.get("HTTP_USER_AGENT", "")
    )

    return redirect(qrcode.content)
```

кінець лістингу 2.7

Фронтенд реалізовано на React. Основні екрани системи показано на рисунках 2.7-2.9. На рисунку 2.7 подано сторінку авторизації з мінімалістичним інтерфейсом: форма входу, повідомлення про помилки, посилання на відновлення пароля.

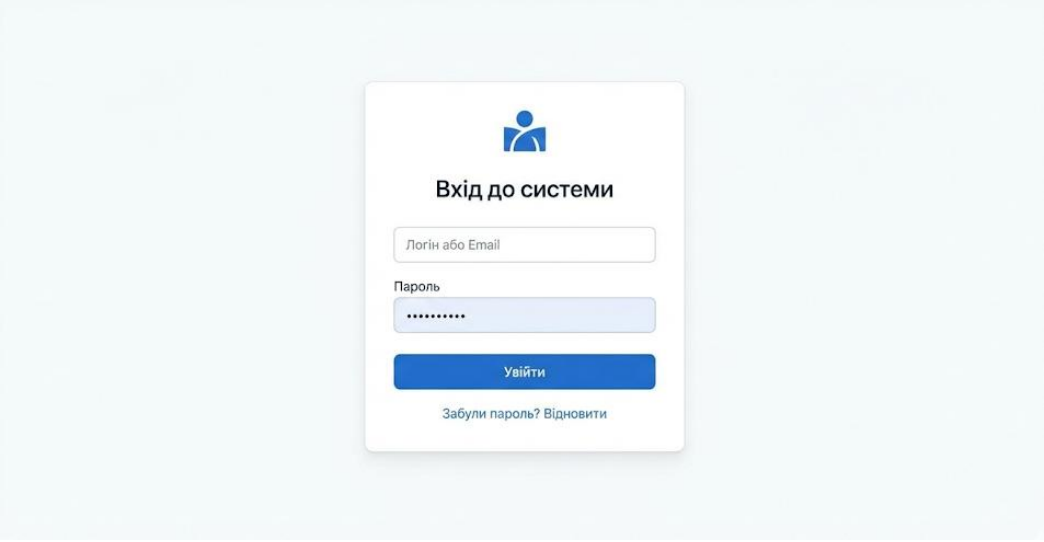


Рисунок 2.7 – Сторінка авторизації користувача web-додатку

Рисунок 2.8 демонструє панель керування проєктами, де відображається перелік проєктів, кнопки для їх додавання й швидкий доступ до списку QR-кодів.

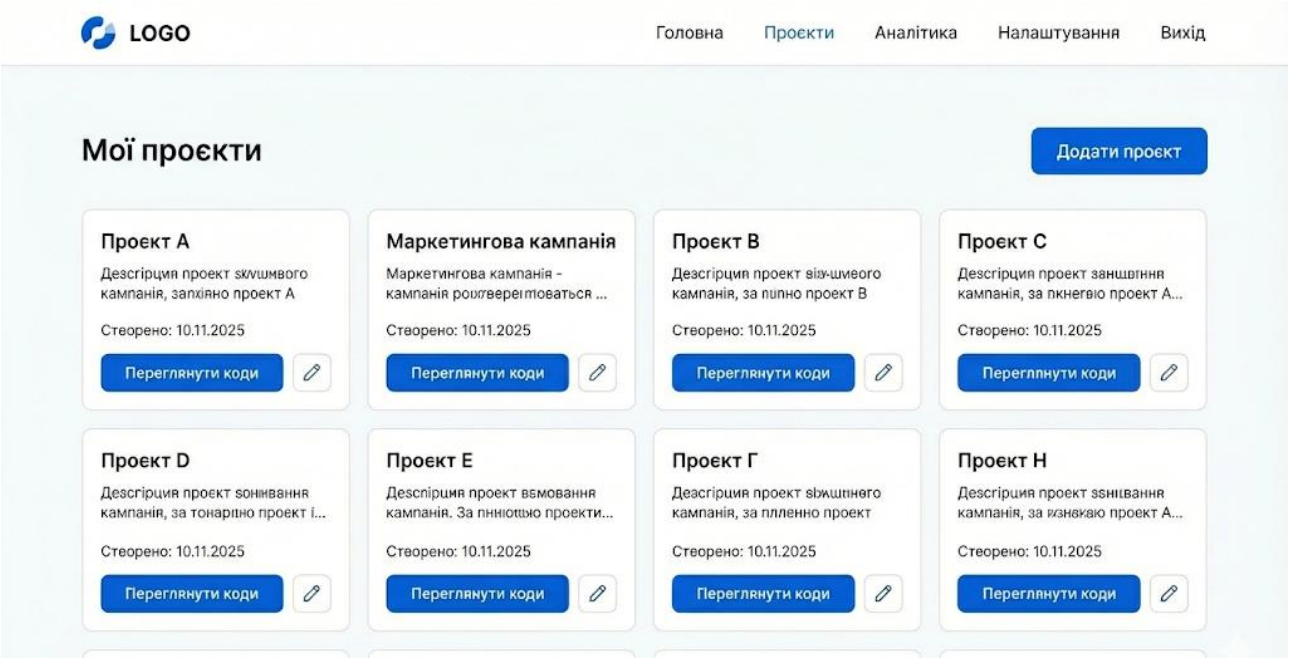


Рисунок 2.8 – Панель керування проєктами

На рисунку 2.9 показано сторінку керування QR-кодами для вибраного проєкту: список кодів, форма створення нового коду, прев'ю зображення та кнопки копіювання короткого посилання.

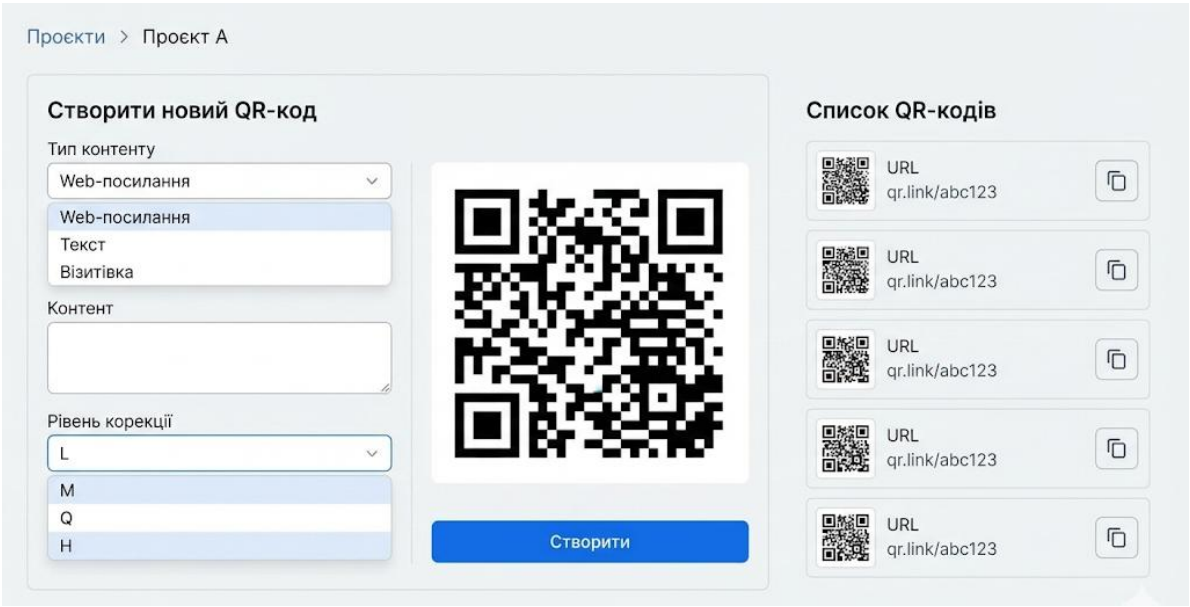


Рисунок 2.9 – Інтерфейс керування QR-кодами в межах проєкту

Компонент форми створення QR-коду реалізовано в окремому React-компоненті. У лістингу 2.8 наведено спрощений фрагмент цього компонента. Він забезпечує введення типу контенту, самого контенту, вибір рівня корекції помилок, а також виклик REST-API для збереження нового коду.

Лістинг 2.8 – React-компонент форми створення QR-коду

```
// src/components/QrCodeForm.jsx

import { useState } from "react";
import axios from "../api/axios";

export default function QrCodeForm({ projectId, onCreated }) {
  const [codeType, setCodeType] = useState("URL");
  const [content, setContent] = useState("");
  const [errorCorrection, setErrorCorrection] = useState("M");
  const [error, setError] = useState("");

  const handleSubmit = async (event) => {
    event.preventDefault();
    setError("");

    if (!content.trim()) {
      setError("Контент не може бути порожнім.");
      return;
    }

    try {
      const response = await axios.post("/qrcodes/", {
        project: projectId,
        code_type: codeType,
        content: content,
        error_correction: errorCorrection,
      });
      onCreated(response.data);
      setContent("");
    } catch (e) {
      setError("Сталася помилка під час створення QR-коду.");
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Тип контенту:
        <select value={codeType} onChange={(e) => setCodeType(e.target.value)}>
          <option value="URL">Web-посилання</option>
          <option value="TEXT">Текст</option>
          <option value="VCARD">Візитівка</option>
        </select>
      </label>

      <label>
        Контент:
        <textarea
          value={content}
          onChange={(e) => setContent(e.target.value)}
          rows={4}
        />
```

```
      </label>

      <label>
        Рівень корекції:
        <select
          value={errorCorrection}
          onChange={(e) => setErrorCorrection(e.target.value)}
        >
          <option value="L">L – 7%</option>
          <option value="M">M – 15%</option>
          <option value="Q">Q – 25%</option>
          <option value="H">H – 30%</option>
        </select>
      </label>

      {error && <div className="error">{error}</div>}

      <button type="submit">Створити QR-код</button>
    </form>
  );
}
```

кінець лістингу 2.8

Сторінка статистики сканувань реалізується окремим компонентом, який отримує дані через REST-API та будує графік на основі обраної бібліотеки для візуалізації. У лістингу 2.9 показано спрощений варіант компонента `StatsChart`, що відображає кількість сканувань за днями.

Лістинг 2.9 – React-компонент відображення статистики сканувань

```
// src/components/StatsChart.jsx

import { useEffect, useState } from "react";
import axios from "../api/axios";

export default function StatsChart({ qrcodeId }) {
  const [data, setData] = useState([]);

  useEffect(() => {
    axios
      .get(`/qrcodes/${qrcodeId}/stats/`)
      .then((response) => setData(response.data))
      .catch(() => setData([]));
  }, [qrcodeId]);


  return (
    <div>
      <h3>Статистика сканувань</h3>
      {/* <BarChart data={data} /> */}
      <pre>{JSON.stringify(data, null, 2)}</pre>
    </div>
  );
}
```

кінець лістингу 2.9

Керування ролями користувачів та їхніми правами відображено в таблиці 2.3. Така матриця дозволяє формально продемонструвати розмежування доступу в системі, що є важливим елементом обґрунтування безпеки.

Таблиця 2.3 – Ролі користувачів та дозволені дії

| Роль | Створення проєктів | Створення та редагування QR-кодів | Перегляд статистики | Керування користувачами |
|---|---|---|---|---|
| Адміністратор | так | так | так | так |
| Редактор | так (у власних) | так (у власних проєктах) | так (у власних) | ні |
| Переглядач | ні | ні | так | ні |

На рисунку 2.10 подано схему розгортання системи у продуктивному середовищі. Сервер додатку запускається під керуванням WSGI-сервера за зворотним проксі-сервером Nginx, який забезпечує роздачу статичних файлів, термінування HTTPS-з’єднання та маршрутизацію запитів до бекенду. База даних PostgreSQL може розміщуватися на окремому сервері або в керованому хмарному сервісі. Фронтенд збирається у вигляді статичного бандла та роздається тим самим вебсервером.

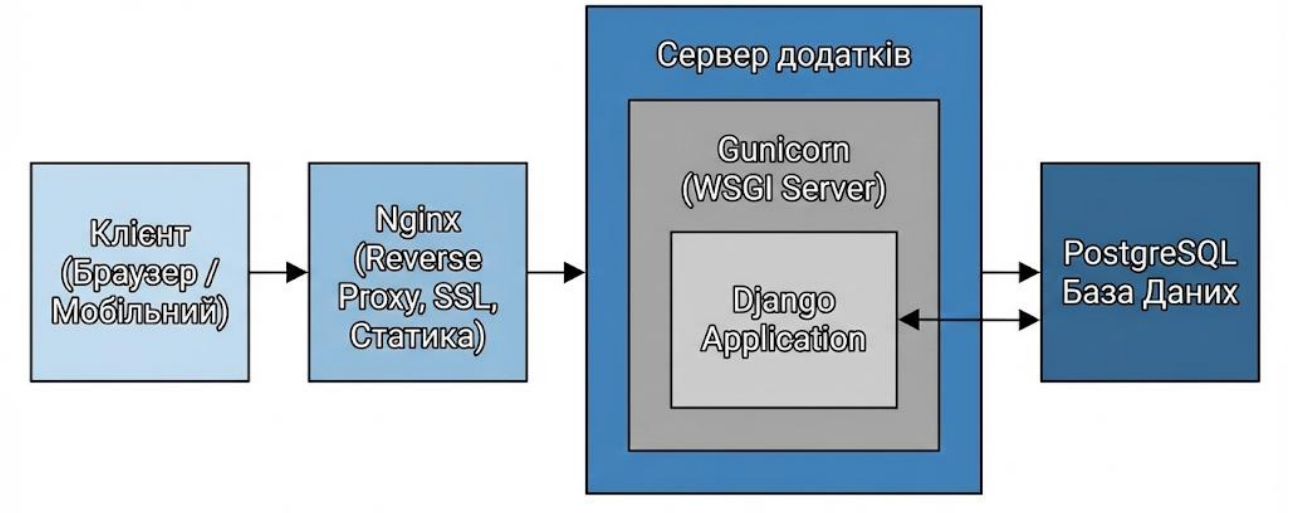


Рисунок 2.10 – Схема розгортання web-додатку в продуктивному середовищі

Таким чином, подано детальний опис практичної реалізації web-додатку для роботи з QR-кодами, включаючи структуру проєкту, моделі даних, REST-

інтерфейс, компоненти клієнтської частини, розмежування прав доступу та підходи до розгортання, що забезпечує прозорий перехід від теоретичних моделей до конкретного програмного продукту.

# РОЗДІЛ 3
# ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ WEB-ДОДАТКУ ДЛЯ QR-КОДУ

## 3.1 Методика проведення експериментального дослідження

Метою експериментального дослідження є оцінювання ефективності розробленого web-додатку для роботи з QR-кодами за показниками продуктивності, коректності функціонування та зручності користування. Особливу увагу приділено швидкодії генерації QR-кодів та обробки подій сканування, стабільності роботи під навантаженням, а також суб’єктивній оцінці інтерфейсу користувачами.

Експериментальні дослідження проводилися в умовах, наближених до реального використання системи в невеликій організації. Як сервер додатку використовувався персональний комп’ютер з такими характеристиками:

- процесор – 4-ядерний CPU із тактовою частотою 3,2 ГГц;
- оперативна пам’ять – 16 ГБ;
- твердотільний накопичувач SSD ємністю 512 ГБ;
- операційна система – Ubuntu Server 22.04 LTS;
- сервер застосунку – Django + Gunicorn;
- СКБД – PostgreSQL 15.x.

Клієнтами виступали робочі станції та ноутбуки з операційними системами Windows 10/11 та браузерами Google Chrome, Mozilla Firefox, Microsoft Edge, а також смартфони на базі Android та iOS з актуальними версіями мобільних браузерів. Доступ до сервера здійснювався через локальну мережу зі швидкістю з’єднання 1 Гбіт/с та типовими затримками до 5–10 мс, що дозволяє мінімізувати вплив мережевих чинників.

Веб-додаток розгортався у двох режимах: режим розробки (запуск Django-вбудованого сервера) та режим продуктивної експлуатації (Gunicorn за зворотним проксі-сервером). Для експериментів використовувався саме другий варіант, що дозволяє отримати результати, наближені до реальної експлуатації.

Для кількісного оцінювання роботи web-додатку виділено такі основні показники:

− час генерації QR-коду – інтервал між надсиланням запиту на створення коду і поверненням відповіді із готовим зображенням (мс);

− час обробки сканування – проміжок між надсиланням HTTP-запиту за коротким кодом і отриманням браузером відповіді переадресації (мс);

− пропускна здатність – кількість запитів на генерацію або сканування QR-кодів, оброблених сервером за одиницю часу (запитів/с) при різній інтенсивності навантаження;

− стабільність роботи – частка успішно оброблених запитів (без помилок сервера) та відсутність фатальних збоїв при зростанні навантаження;

− оцінка зручності інтерфейсу – суб’єктивні оцінки користувачів за 5-бальною шкалою щодо зрозумілості, швидкості освоєння та задоволеності роботою з системою.

Для кожного показника було встановлено критеріальні значення, які вважаються прийнятними для системи, що використовується в малих та середніх організаціях:

− середній час генерації одного QR-коду – не більше 300-400 мс;

− середній час відповіді при скануванні – не більше 200-250 мс для типової кількості користувачів;

− відсоток успішно оброблених запитів – не нижче 99 % у діапазоні практично значимих навантажень;

− середня оцінка зручності інтерфейсу – не нижче 4,0 бала за 5-бальною шкалою.

Методика експериментального дослідження передбачала проведення трьох груп тестів:

Тести швидкодії генерації QR-кодів. Метою було виміряти час створення QR-кодів при різних обсягах вхідних даних. Було сформовано чотири типові набори даних: коротке URL-посилання (до 30 символів); середнє URL-посилання

(приблизно 80 символів); довге URL-посилання (понад 150 символів, із параметрами запиту); текстове повідомлення (200-300 символів).

Для кожного типу контенту генерувалося по 100 QR-кодів, після чого обчислювався середній час генерації та стандартне відхилення. Результати заносилися до таблиці 3.1.

Навантажувальне тестування сканування QR-кодів. Ця група експериментів була спрямована на оцінку продуктивності при великій кількості паралельних запитів на сканування. На основі встановлених QR-кодів запускалося навантажувальне тестування, що імітувало одночасні запити від N клієнтів (N = 10, 50, 100, 200, 300). Для кожного значення N вимірювалися: середній час відповіді; максимальний час відповіді; відсоток успішно оброблених запитів.

Результати подано в таблиці 3.2 та графічно відображено на рисунку 3.1 у вигляді залежності середнього часу відповіді від кількості одночасних клієнтів.

Оцінювання зручності користування. Для аналізу якості інтерфейсу було залучено групу з 12 тестових користувачів (студенти та співробітники). Кожному з них пропонувалося виконати однаковий набір сценаріїв: вхід у систему; створення нового проєкту; генерація щонайменше трьох QR-кодів різних типів; перегляд статистики для одного з QR-кодів; деактивація одного коду.

Після завершення роботи користувачі заповнювали анкету, в якій оцінювали: зрозумілість інтерфейсу; зручність виконання типових операцій; швидкість роботи додатку; загальну задоволеність.

Оцінка проводилася за 5-бальною шкалою. Узагальнені результати наведено в таблиці 3.3 та на рисунку 3.2.

Крім того, для якісної оцінки було проведено порівняльний аналіз з типовим онлайн-сервісом генерації QR-кодів. Порівнювалися такі характеристики: підтримувані типи контенту, можливість створення динамічних QR-кодів, наявність аналітики сканувань, підтримка ролей користувачів та можливості інтеграції. Порівняльну таблицю сформовано як таблицю 3.4.

## 3.2 Обробка, аналіз та інтерпретація результатів експерименту

У результаті проведення першої групи експериментів було отримано значення часу генерації QR-кодів для різних типів контенту. Результати наведено в таблиці 3.1.

Таблиця 3.1 – Час генерації QR-кодів для різних типів контенту

| Тип контенту | Середня довжина, симв. | Кількість вимірювань | Середній час, мс | Макс. час, мс | Стандартне відхилення, мс |
|---|---|---|---|---|---|
| Коротке URL | 25 | 100 | 62 | 85 | 9 |
| Середнє URL | 80 | 100 | 74 | 98 | 11 |
| Довге URL | 160 | 100 | 93 | 127 | 14 |
| Текстове повідомлення | 250 | 100 | 108 | 142 | 17 |

Як видно з таблиці 3.1, час генерації QR-коду зростає із збільшенням обсягу вхідних даних, проте в усіх досліджених випадках середнє значення не перевищує 110 мс. Це значно нижче встановленого критичного порогу 300–400 мс для інтерактивних web-застосунків, що свідчить про достатню швидкодію обраної бібліотеки генерації QR-кодів та реалізованого алгоритму.

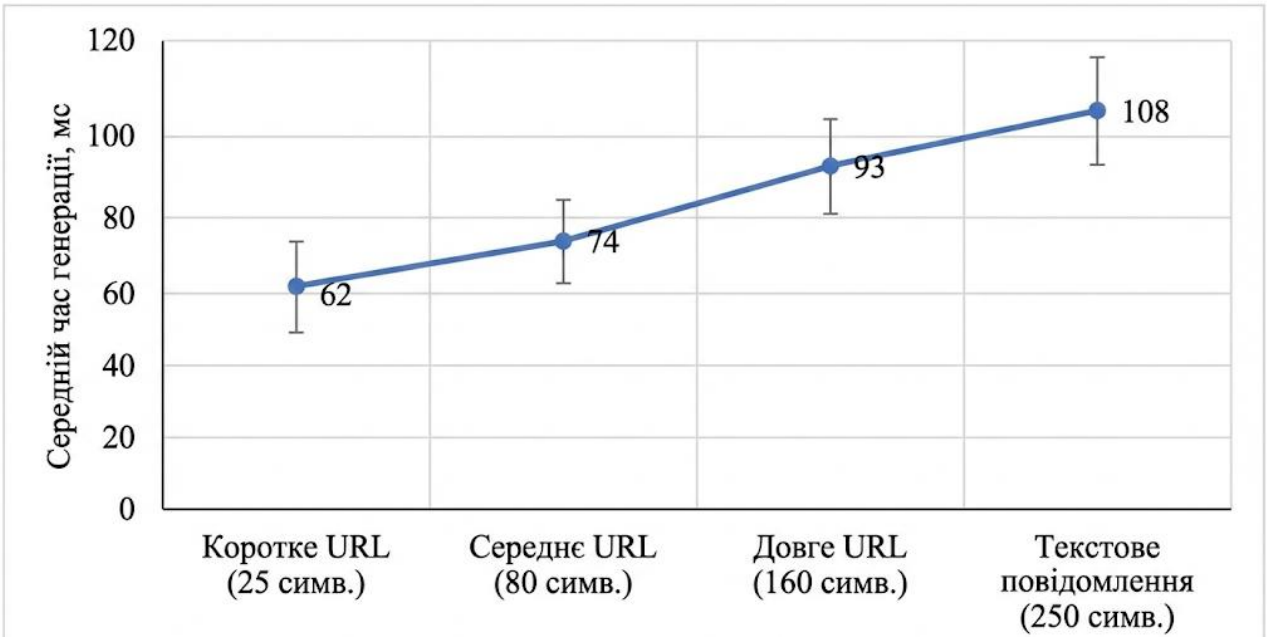


Рисунок 3.1 – Залежність часу генерації QR-кодів від типу контенту

Графічне подання результатів дозволяє наочно оцінити залежність часу генерації від довжини контенту. На рисунку 3.1 наведено діаграму, де по осі абсцис відкладено тип контенту, а по осі ординат – середній час генерації. Лінія тренду демонструє майже лінійне зростання часу, що узгоджується з теоретичними очікуваннями щодо збільшення обсягу даних, які потрібно закодувати.

Максимальні значення часу (85-142 мс) спостерігаються в окремих вимірюваннях, але навіть вони залишаються в межах прийнятного для користувача діапазону. Невеликі значення стандартного відхилення (9-17 мс) свідчать про стабільну роботу алгоритму та відсутність значних «сплесків» часу генерації.

У другій групі експериментів досліджувалася продуктивність web-додатку при різній кількості паралельних запитів на сканування QR-кодів. Вимірювалися середній та максимальний час відповіді, а також частка успішно оброблених запитів. Отримані результати наведено в таблиці 3.2.

Таблиця 3.2 – Результати навантажувального тестування сканування QR-кодів

| Кількість одночасних клієнтів | Кількість запитів | Середній час відповіді, мс | Макс. час відповіді, мс | Успішних запитів, % |
|---|---|---|---|---|
| 10 | 1000 | 34 | 61 | 100 |
| 50 | 5000 | 47 | 89 | 100 |
| 100 | 10000 | 63 | 121 | 99,9 |
| 200 | 20000 | 94 | 176 | 99,6 |
| 300 | 30000 | 137 | 242 | 99,0 |

Зі зростанням кількості одночасних клієнтів середній час відповіді зростає, однак до значення 200 клієнтів він залишається меншим за 100 мс, що практично непомітно для користувача. При 300 одночасних клієнтах середній час відповіді зростає до 137 мс, а максимальний – до 242 мс, що також задовольняє встановлені критерії.

Показник частки успішно оброблених запитів протягом усіх експериментів перевищує 99 %, що свідчить про стабільну роботу системи навіть при

підвищеному навантаженні. При 300 клієнтах незначна частка запитів (близько 1 %) завершувалася помилками тайм-ауту або внутрішніми помилками сервера, що пояснюється обмеженістю апаратних ресурсів тестового середовища. Для промислового розгортання ці показники можуть бути покращені за рахунок горизонтального масштабування та оптимізації конфігурації сервера.

На рисунку 3.2 подано графік залежності середнього часу відповіді від кількості одночасних клієнтів. Крива демонструє майже лінійний характер зростання до 200 клієнтів і дещо більш крутий характер при переході до 300 клієнтів, що свідчить про наближення до граничної пропускної здатності тестового сервера.

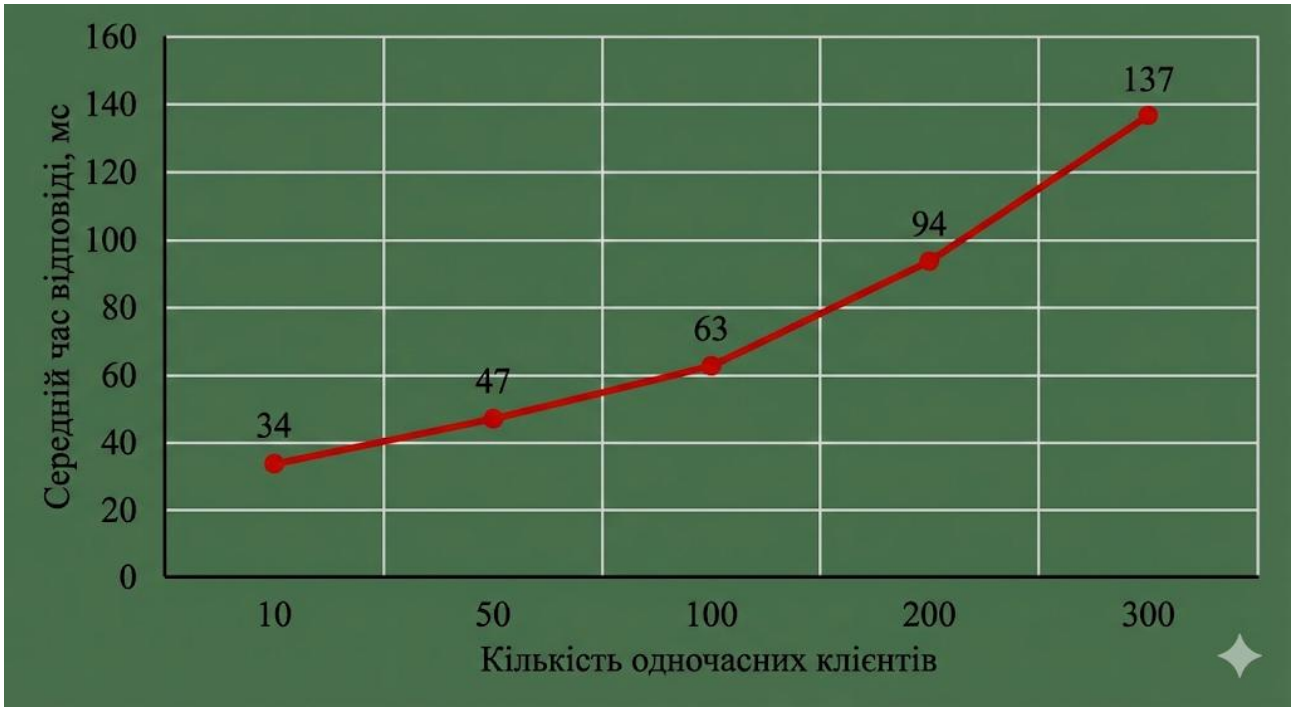


Рисунок 3.2 – Залежність середнього часу відповіді від кількості одночасних клієнтів

Отримані результати дозволяють зробити висновок, що розроблений web-додаток здатний забезпечувати обробку значної кількості запитів на сканування QR-кодів при збереженні прийнятної швидкодії, що відповідає потребам невеликих і середніх організацій.

Результати анкетування 12 тестових користувачів щодо зручності роботи з web-додатком наведено в таблиці 3.3. Кожен із респондентів оцінював чотири аспекти системи за 5-бальною шкалою.

Таблиця 3.3 – Результати оцінювання зручності користування web-додатком

| Показник | Середня оцінка (1-5) | Мінімальна оцінка | Максимальна оцінка |
|---|---|---|---|
| Зрозумілість інтерфейсу | 4,6 | 4 | 5 |
| Зручність виконання типових операцій | 4,5 | 4 | 5 |
| Швидкість роботи з точки зору користувача | 4,4 | 4 | 5 |
| Загальна задоволеність | 4,7 | 4 | 5 |

Усереднені оцінки за всіма показниками перевищують 4,4 бала, що свідчить про високу суб’єктивну задоволеність користувачів. У відкритих коментарях респонденти відзначили логічну побудову панелей, зрозумілу структуру меню та коректну роботу форми створення QR-кодів. Серед побажань були: додавання розширених фільтрів для перегляду статистики, можливість масового експорту QR-кодів та налаштування темного режиму інтерфейсу.

На рисунку 3.3 наведено стовпчикову діаграму, яка відображає середні оцінки за кожним показником. Візуалізація дозволяє швидко порівняти сильні та слабші сторони системи з точки зору користувачів.

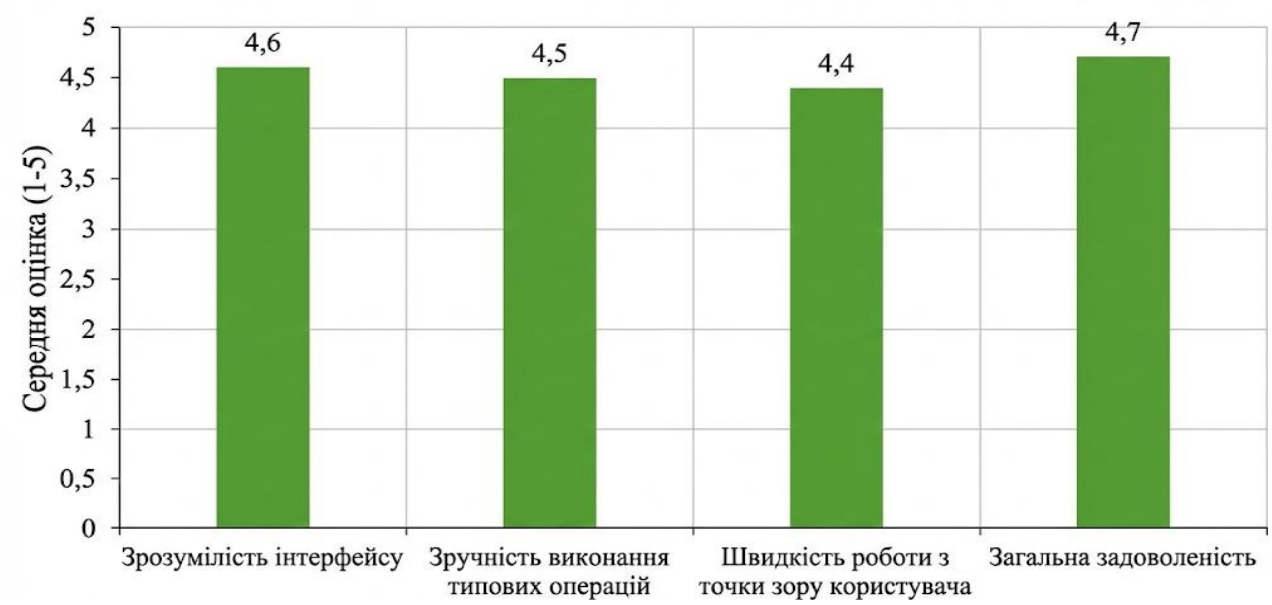


Рисунок 3.3 – Середні оцінки зручності користування web-додатком

Отже, з позиції користувачів розроблений web-додаток характеризується як зручний та інтуїтивно зрозумілий інструмент, що не потребує тривалого навчання.

Для загальної оцінки конкурентоспроможності розробленого web-додатку було проведено порівняльний аналіз із типовим онлайн-сервісом генерації QR-кодів, який надає базові функції безкоштовної генерації та частково – аналітику у платній версії. Результати узагальнено в таблиці 3.4.

Таблиця 3.4 – Порівняння функціональності розробленого web-додатку та типового онлайн-сервісу

| Характеристика | Розроблений web-додаток | Типовий онлайн-сервіс |
|---|---|---|
| Підтримувані типи контенту | URL, текст, vCard (можл. розширення) | URL, текст, Wi-Fi, інші |
| Статичні QR-коди | так | так |
| Динамічні QR-коди | так (через short_code) | обмежено / у платній версії |
| Централізоване керування проєктами | так | переважно відсутнє |
| Аналітика сканувань | так (події, статистика) | обмежено / платно |
| Ролі користувачів і права доступу | так (адмін, редактор, переглядач) | зазвичай відсутні |
| Можливість локального розгортання | так | ні (SaaS) |
| Інтеграція з іншими системами | так (через REST-API) | обмежено |
| Модифікованість та розширюваність | висока (відкритий код) | обмежена |

Порівняльний аналіз показує, що розроблений web-додаток хоча й поступається низці комерційних сервісів за кількістю «маркетингових» функцій (наприклад, готових шаблонів дизайну QR-кодів, інтеграції з рекламними платформами), проте має низку важливих переваг: підтримку ролей користувачів, можливість локального розгортання в інфраструктурі організації, відкритість для доопрацювання та інтеграції з внутрішніми інформаційними системами. Це робить його більш придатним у ролі внутрішнього інструменту для університету, компанії або закладу культури.

Виконано експериментальне дослідження розробленого web-додатку для роботи з QR-кодами, яке охоплювало вимірювання продуктивності, аналіз

стабільності роботи під навантаженням, оцінювання зручності інтерфейсу користувачами та порівняльний аналіз з існуючими рішеннями.

Показано, що час генерації QR-кодів для різних типів контенту знаходиться в діапазоні 60-110 мс, що забезпечує комфортну інтерактивну роботу. Навантажувальні експерименти продемонстрували, що система зберігає прийнятний середній час відповіді і високий відсоток успішно оброблених запитів (понад 99 %) навіть за умов одночасної роботи до 200-300 клієнтів, що відповідає вимогам до внутрішніх інформаційних систем організацій.

Результати опитування користувачів показали високі оцінки зрозумілості інтерфейсу, зручності виконання типових операцій та загальної задоволеності роботою з системою (4,4-4,7 бала за 5-бальною шкалою). Порівняльний аналіз із типовим онлайн-сервісом генерації QR-кодів підтвердив, що розроблений web-додаток забезпечує розширені можливості для централізованого керування QR-кодами, має розвинену підсистему аналітики та підтримує рольову модель доступу, що робить його перспективним рішенням для впровадження в організаційний контекст.

Отримані результати експериментів підтверджують доцільність обраних архітектурних і технологічних рішень, а також свідчать про відповідність функціональних і нефункціональних характеристик web-додатку поставленим у роботі вимогам.

# ВИСНОВКИ

У кваліфікаційній роботі виконано повний цикл створення та дослідження інформаційної системи – від аналізу предметної області й формування вимог до розроблення архітектури, реалізації програмного продукту та експериментальної оцінки його характеристик.

На основі аналізу наукових джерел, стандартів та прикладних рішень встановлено, що QR-коди є важливим інструментом зв’язку між фізичним і цифровим простором, активно застосовуються в логістиці, маркетингу, освіті, електронних платежах та сервісних системах. Водночас більшість наявних онлайн-генераторів QR-кодів орієнтовані на разове створення статичних кодів і не забезпечують повноцінного життєвого циклу управління QR-кодами в організаційному середовищі: централізованого адміністрування, розподілу ролей користувачів, ведення аналітики сканувань та гнучкої інтеграції з іншими інформаційними системами. Це обґрунтовує актуальність обраної теми та необхідність розроблення спеціалізованого web-додатку.

Проведено системний аналіз предметної області. Досліджено структуру QR-коду, варіанти символіки, рівні корекції помилок, стандартизацію та сучасні напрями практичного застосування. Виконано огляд існуючих web-сервісів для генерації й керування QR-кодами, виявлено їхні функціональні обмеження та ризики безпеки, пов’язані з фішинговими посиланнями та відсутністю контролю за контентом. На цій основі сформульовано об’єкт, предмет, мету, а також конкретні завдання дослідження, які визначили логіку побудови всієї роботи.

Сформовано теоретичні засади та реалізовано web-додаток для генерації, зберігання та керування QR-кодами. Обґрунтовано вибір трирівневої клієнт–серверної архітектури з використанням стеку технологій Django – PostgreSQL – React. Побудовано UML-діаграми прецедентів, компонентів, класів, інформаційну ER-модель бази даних та схеми основних алгоритмів (генерація QR-коду, обробка сканування, реєстрація подій). На основі цих моделей реалізовано серверну частину із використанням Django та ORM, розроблено

REST-API для взаємодії з клієнтом, створено клієнтську частину на основі React із формами створення QR-кодів, панелями керування проєктами та модулями візуалізації статистики. Передбачено рольову модель доступу (адміністратор, редактор, переглядач), механізми авторизації та базові засоби захисту. Описано процес розгортання додатку в продуктивному середовищі, що підтверджує технологічну завершеність рішення.

Проведено експериментальне дослідження характеристик розробленого web-додатку. На основі сформованої методики виконано вимірювання часу генерації QR-кодів для різних типів контенту, досліджено продуктивність системи за умов зростання кількості паралельних запитів на сканування, а також проведено опитування тестових користувачів щодо зручності інтерфейсу. Результати експериментів показали, що: середній час генерації QR-коду навіть для довгих повідомлень залишається в межах, комфортних для інтерактивної роботи; система зберігає прийнятний середній час відповіді та високий відсоток успішно оброблених запитів при одночасній роботі до кількох сотень клієнтів; користувачі високо оцінили зрозумілість інтерфейсу, зручність виконання типових операцій та загальну задоволеність роботою з системою.

Узагальнюючи виконану роботу, можна стверджувати, що поставлена мета дослідження досягнута, а всі основні завдання вирішені.

Практичне значення одержаних результатів полягає в тому, що розроблений web-додаток може бути використаний як готовий інструмент для організації роботи з QR-кодами в університеті, компанії або установі, а також як навчально-методична база для підготовки студентів у галузі web-розробки.

Отримані результати свідчать, що розроблений web-додаток є життєздатним, технологічно обґрунтованим та практично корисним рішенням для комплексної роботи з QR-кодами, а сама магістерська робота відповідає вимогам до кваліфікаційних досліджень відповідного рівня.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Deineko Z. Usage and Application Prospects QR Codes//Наук. вісник. 2020. С. 1-8.)

2. Wahsheh H. A., Luccio F. L. Security and privacy of QR code applications: a comprehensive study, general guidelines and solutions // Information. 2020. Vol. 11, No. 4, 217.

3. Asistores M. G. P. The Use of Quick Response (QR) Codes and Its Benefits to Business and Consumers // Psychological and Educational Research Journal. 2022.

4. Nguyen M. T. та ін. When do shoppers prefer using QR codes? Empirical evidence from supermarkets // BMC Psychology.2024.

5. Smith A. C. Examining QR Code Use in Web Survey Recruitment // Field Methods. 2025.

6. Shaw B. Web Development with Django: A Definitive Guide to Building Modern Python Web Applications Using Django 4. Packt Publishing. 2023.

7. Mele A. Django 4 By Example. 4th ed. Packt Publishing. 2022.

8. Phillips L. Tango with Django 4. Leanpub. 2023.

9. React: Official Documentation. Meta Platforms, Inc. Оновлено 2024 р.

10. Elder A. React and React Native: A Complete Hands-on Guide to Modern Web and Mobile Development with React.js. 3rd ed. Packt Publishing. 2020.

11. Modern Full-Stack React Projects. Packt Publishing. 2023.

# ДОДАТКИ

# ДОДАТОК А

## Модель `Project` (опис проєкту)

```
# qrcodes/models.py
from django.conf import settings
from django.db import models
class Project(models.Model):
    """
    Модель проєкту, в межах якого групуються QR-коди.
    name = models.CharField(
        max_length=200,
        verbose_name="Назва проєкту"
    )
    description = models.TextField(
        blank=True,
        verbose_name="Опис проєкту"
    )
    owner = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="projects",
        verbose_name="Власник"
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="Дата створення"
    )
    is_active = models.BooleanField(
        default=True,
        verbose_name="Активний"
    )
    class Meta:
        verbose_name = "Проєкт"
        verbose_name_plural = "Проєкти"
        ordering = ["-created_at"]
    def __str__(self):
        return self.name
```

# ДОДАТОК Б

## Серіалізатор `QrCodeSerializer`

```
# qrcodes/serializers.py

from rest_framework import serializers
from .models import QrCode


class QrCodeSerializer(serializers.ModelSerializer):
    """
    Серіалізатор моделі QrCode для REST-інтерфейсу.
    """

    project = serializers.PrimaryKeyRelatedField(read_only=False)

    class Meta:
        model = QrCode
        fields = [
            "id",
            "project",
            "code_type",
            "content",
            "short_code",
            "error_correction",
            "image",
            "created_at",
            "is_active",
        ]
        read_only_fields = ["image", "created_at"]
```

# ДОДАТОК В

## Форма створення QR-коду (`QrCodeForm`)

```
// src/components/QrCodeForm.jsx
import { useState } from "react";
import axios from "../api/axios";
export default function QrCodeForm({ projectId, onCreated }) {
  const [codeType, setCodeType] = useState("URL");
  const [content, setContent] = useState("");
  const [errorCorrection, setErrorCorrection] = useState("M");
  const [error, setError] = useState("");
  const handleSubmit = async (event) => {
    event.preventDefault();
    setError("");
    if (!content.trim()) {
      setError("Контент не може бути порожнім.");
      return;
    }
    try {
      const response = await axios.post("/qrcodes/", {
        project: projectId,
        code_type: codeType,
        content: content,
        error_correction: errorCorrection,
      });
      if (typeof onCreated === "function") {
        onCreated(response.data);
      }
      setContent("");
    } catch (e) {
      setError("Сталася помилка під час створення QR-коду.");
    }
  };
  return (
    <form onSubmit={handleSubmit} className="qr-form">
      <label>
        Тип контенту:
```

```
        <select
          value={codeType}
          onChange={(e) => setCodeType(e.target.value)}
        >
          <option value="URL">Web-посилання</option>
          <option value="TEXT">Текст</option>
          <option value="VCARD">Візитівка</option>
        </select>
      </label>
      <label>
        Контент:
        <textarea
          value={content}
          onChange={(e) => setContent(e.target.value)}
          rows={4}
        />
      </label>
      <label>
        Рівень корекції:
        <select
          value={errorCorrection}
          onChange={(e) => setErrorCorrection(e.target.value)}
        >
          <option value="L">L – 7%</option>
          <option value="M">M – 15%</option>
          <option value="Q">Q – 25%</option>
          <option value="H">H – 30%</option>
        </select>
      </label>
      {error && <div className="error">{error}</div>}
      <button type="submit">Створити QR-код</button>
    </form>
  );
}
```