

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «вебтехнології»)
на тему: «Вебсистема Вело-паспорт»

Виконав студент групи ВТп-22-1
ІЛЛАРІОНОВ Ігор Олегович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ДУНЄВ Сергій Сергійович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«16» липня 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ІЛЛАРІОНОВ Ігор Олегович

Керівник роботи: ВЛАСЕНКО Олег Васильович

Тема роботи: «Вебсистема Вело-паспорт»,

затверджена Наказом закладу вищої освіти від **«16» липня 2026 р., №396/с**

Вихідні дані для роботи: вебсистема для ведення цифрового паспорта для велосипедів, фіксації статусу власності та взаємодії з уповноваженими користувачами.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	16.07.2026р.	16.07.2026р
2	Власенко О.В.	16.07.2026р	16.07.2026р
3	Власенко О.В.	16.07.2026р	16.07.2026р

РЕФЕРАТ

У кваліфікаційній роботі розглянуто питання проєктування та реалізації вебсистеми Вело-паспорт, призначеного для створення цифрового паспорта велосипеда, централізованого зберігання ідентифікаційних даних, фотоматеріалів, поточного статусу та історії зміни стану велосипеда. Окрему увагу приділено контролю доступу до даних власника, підтримці публічного пошуку за серійним номером, передачі права власності та інтеграції із зовнішніми сервісами через машинні облікові записи.

У роботі проаналізовано предметну область цифрової ідентифікації велосипедів, досліджено наявні аналоги сервісів реєстрації та перевірки велосипедів, обґрунтовано вибір клієнт-серверної архітектури, документно-орієнтованої бази даних та сучасних засобів автентифікації.

Метою роботи є аналіз, проєктування та реалізація вебсистеми Вело-паспорт для обліку велосипедів, оперативного повідомлення про викрадення або повернення та забезпечення безпечної взаємодії між власниками, адміністраторами, представниками поліції й зовнішніми сервісами.

Випускна робота містить 62 сторінки основного тексту, 34 ілюстрації, 8 таблиць, 4 додатки.

КЛЮЧОВІ СЛОВА: ЦИФРОВИЙ ПАСПОРТ, ВЕЛОСИПЕД, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, АВТЕНТИФІКАЦІЯ, РОЛЬОВИЙ ДОСТУП, РЕЄСТР

					ІПЗ.КР.Б-121.26 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсистема Вело-паспорт	Літ.	Арк.	Аркушів
Розроб.		Ілларіонов І.О.					3	69
Керівник		Власенко О.В.				ФІКТ, гр. ВТп-22-1		
Рецензент		Дунєв С. С.						
Зав. Каф.		Вакалюк Т.А.						

ABSTRACT

This qualification thesis addresses the design and implementation of the Cycle Pass web system, developed for creating a digital bicycle passport, centralized storage of identification data, photos, current status, and state history of a bicycle. Special attention is paid to controlling access to owner data, supporting public search by serial number, transferring ownership, and integrating with external services via machine accounts (M2M).

The domain of digital bicycle identification was analyzed, existing analogous bicycle registration and verification services were examined, and the choice of a client-server architecture, document-oriented database, and modern authentication mechanisms was justified.

The goal of this thesis is the analysis, design, and implementation of the Cycle Pass web portal for bicycle management, prompt notification of theft or recovery, and providing secure interaction between owners, administrators, police representatives, and external services.

The graduation thesis contains 62 pages of main text, 34 illustrations, 8 tables, and 4 appendices.

KEYWORDS: DIGITAL PASSPORT, BICYCLE, CLIENT-SERVER ARCHITECTURE, AUTHENTICATION, ROLE-BASED ACCESS, REGISTRY

					ІПЗ.КР.Б-121.26 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Ілларионов І.О.			Вебсистема Вело-паспорт	Літ.	Арк.	Аркушів	
Керівник		Власенко О.В.					4	69	
						ФІКТ, зр. ВТн-22-1			
Рецензент		Дунєв С. С.							
Зав. Каф.		Вакалюк. Т.А.							

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧ	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного забезпечення.....	12
1.3. Вибір архітектури веборієнтованої системи	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	21
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСИСТЕМИ ВЕЛОСИПЕДНОГО ПАСПОРТУ.....	26
2.1. Визначення варіантів використання та структури вебсистеми.....	26
2.2. Розробка баз даних	31
2.3. Реалізація функціоналу вебреєстру велосипедів	36
Висновки до другого розділу	39
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБСИСТЕМИ ...	40
3.1. Розгортання вебсистеми	40
3.2. Структура інтерфейсу	43
3.3. Тестування роботи вебсистеми.....	54
Висновки до третього розділу.....	57
ВИСНОВОК	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ.....	60
ДОДАТКИ.....	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

API – Application Programming Interface

UI – User Interface

JS – JavaScript

JWT – JSON Web Token

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

СУБД – система управління базами даних

SPA – Single Page Application

RBAC – Role-Based Access Control

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми зумовлена тим, що велосипед є поширеним міським транспортом, однак його ідентифікація після купівлі, передачі іншому власнику або викрадення часто залишається фрагментованою. Серійний номер, фотографії, історія зміни статусу та підтвердження власника зберігаються в різних місцях або взагалі не фіксуються у цифровому вигляді. Через це власнику складно швидко підтвердити належність велосипеда, а іншим користувачам — перевірити інформацію про знайдений або підозрілий велосипед. За даними правоохоронної статистики більшості європейських країн, викрадення велосипедів належать до найпоширеніших майнових злочинів [3, 6]. Розвиток вторинного ринку через онлайн-майданчики додатково підвищує ризики: покупець фактично не має інструменту, щоб перевірити походження велосипеда до моменту оплати [1, 4]. Отже, актуальним є створення централізованої веборієнтованої системи цифрового паспорта велосипеда, яка поєднує захищене зберігання даних про власника, публічну перевірку статусу за серійним номером і прозорий механізм передачі права власності.

На сьогодні відсутній єдиний загальнодоступний реєстр, який пов'язував би серійний номер велосипеда з підтвердженням власником та його поточним статусом. Інформація розпорошена між окремими сервісами, обліковими записами магазинів і відомчими системами, тож підтвердити право власності чи перевірити походження велосипеда під час купівлі на вторинному ринку доволі складно. Централізована вебсистема з публічним пошуком і прозорим механізмом передачі власності дає власникам, покупцям та уповноваженим користувачам єдину точку доступу до достовірних відомостей.

Сучасні веборієнтовані системи дають змогу організувати єдине середовище доступу через браузер, де дані можуть бути захищені засобами автентифікації, рольового доступу та контрольованого відображення. Для предметної області велосипедів це створює практичну основу для цифрового паспорта, який поєднує облік, публічну перевірку за серійним номером, фіксацію

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

статусу «active», «stolen» або «recovered» та безпечну передачу велосипеда новому власнику.

Мета дослідження полягає в аналізі, проєктуванні та реалізації програмного комплексу Вело-паспорт, який забезпечує створення цифрового паспорта велосипеда, керування записами власника, пошук відкритої інформації, зміну статусу велосипеда, завантаження фотографій та взаємодію з уповноваженими користувачами.

Завданням дослідження є досягнення поставленої мети шляхом виконання визначених етапів:

- здійснення детального аналізу аналогів у сфері систем реєстрації та ідентифікації велосипедів;
- дослідження існуючих архітектурних підходів до побудови веборієнтованих систем;
- проєктування архітектури вебсистеми Вело-паспорт;
- обґрунтування та вибір інструментальних засобів;
- реалізація моделі розмежування прав доступу та моделювання бізнес-процесів реєстрації, зміни статусу й передачі велосипеда;
- програмна реалізація серверної та клієнтської частини вебсистеми;
- проведення тестування працездатності роботи вебсистеми.

Об'єктом дослідження є вебтехнології для розробки вебсистеми цифрового паспорта велосипеда, які забезпечують реалізацію функціоналу, необхідного для автоматизації процесів обліку, перевірки та передачі велосипедів.

Предметом дослідження є структура та методи реалізації веборієнтованого програмного комплексу Вело-паспорт, що включають програмні алгоритми, моделі даних, архітектурні моделі та технологічні методи побудови односторінкових вебдодатків (SPA) для автоматизації взаємодії між власниками велосипедів і уповноваженими користувачами, а також засоби автентифікації та криптографічні механізми захисту даних.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Методи аналізу та інформаційна база. Для вирішення поставлених завдань у роботі застосовано методи порівняльного моніторингу існуючих аналогів інформаційних систем та методи структурного аналізу з використанням діаграм прецедентів і проєктування схем даних.

Наукова новизна роботи полягає в поєднанні у межах єдиної веборієнтованої системи трьох підходів, які в наявних сервісах реалізовано розрізнено: публічної перевірки велосипеда за серійним номером без автентифікації, захищеного керування записами власника на основі делегованої автентифікації та контрольованого доступу сторонніх застосунків за принципом багатоорендності. На відміну від аналогів, у яких взаємодію з партнерами організовано переважно на організаційному рівні, у запропонованому рішенні доступ сторонніх застосунків до службових даних регулюється програмно — через набір дозволів у токени та фільтрацію відповіді API. Це дає змогу поєднати відкритість публічної перевірки з надійним захистом персональних даних власника в межах однієї системи.

Отже, практичне значення роботи полягає у створенні вебсистеми цифрового паспорта велосипеда, стійкої до ризиків несанкціонованого доступу та витoku персональних даних власників, із забезпеченням достовірності облікових записів велосипедів і якості взаємодії між власниками, покупцями та уповноваженими користувачами.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧ

1.1. Постановка задачі

Облік велосипедів пов'язаний із потребою швидко встановлювати належність транспортного засобу конкретному власнику, перевіряти його статус і зберігати підтвердні матеріали. У традиційному підході власник має чек, фотографії та серійний номер окремо, тому під час втрати, продажу або перевірки велосипеда виникає проблема неповноти даних. Для стороннього користувача або представника поліції відсутність єдиного цифрового запису ускладнює перевірку інформації.

У зв'язку з цим актуальною є задача розробки вебсистеми Вело-паспорт, яка надає власнику можливість зареєструвати велосипед, зберігати модель, виробника, серійний номер, фотографії та поточний статус, а також формувати одноразове посилання для передачі права власності іншому користувачу. Публічна частина системи повинна дозволяти виконувати пошук за серійним номером без розкриття приватних даних власника.

На основі описаної проблематики сформульовано перелік функціональних вимог до вебсистеми. Система повинна забезпечувати реєстрацію велосипеда з обов'язковими полями (виробник, модель, серійний номер) та довільним набором фотографій; зберігання і зміну поточного статусу велосипеда — «активний», «викрадений» або «повернений»; редагування облікового запису його власником; формування одноразового посилання чи QR-коду для передачі права власності; публічний пошук за серійним номером без розкриття персональних даних; а також розмежований доступ для власника, представника поліції, адміністратора та сторонніх застосунків. Кожна з цих вимог безпосередньо впливає з потреби усунути фрагментованість обліку, описану вище.

Окрім функціональних, до системи висуваються й нефункціональні вимоги. Передусім це захищеність: персональні дані власника та службова інформація мають бути доступні лише авторизованим користувачам, а публічна

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

частина — розкривати виключно ту інформацію, що потрібна для перевірки статусу. Система повинна забезпечувати передбачувану швидкодію публічного пошуку незалежно від зростання кількості записів, стабільну доступність основних сценаріїв та цілісність даних під час одночасної роботи різних категорій користувачів. Важливою є також розширюваність — можливість додавати нові функції та типи користувачів без переробки наявної логіки.

Задача розглядається в розрізі кількох категорій користувачів, кожна з яких має власні цілі. Власник велосипеда потребує зручного інструменту для реєстрації, оновлення та передачі своїх записів; неавторизований користувач або потенційний покупець — можливості швидко перевірити походження та статус велосипеда за серійним номером; представник поліції — доступу до переліку велосипедів, позначених як викрадені; адміністратор — засобів модерації та контролю платформи. Окрему категорію становлять сторонні застосунки, які за наявності відповідних дозволів можуть зберігати службові дані у власному ізольованому просторі. Формалізація цих ролей визначає межі відповідальності системи та вимоги до розмежування доступу.

Під час постановки задачі враховано й низку обмежень та припущень. Система не виконує фінансових операцій і не обробляє платежів, покладаючи автентифікацію на зовнішній сервіс Auth0, а зберігання зображень — на Cloudinary. Передбачається, що користувачі мають доступ до мережі Інтернет і сучасного веббраузера, а перевірка фізичної відповідності велосипеда серійному номеру залишається поза межами програмної системи й покладається на користувача. Ці обмеження дають змогу сфокусувати розробку на основній меті — створенні достовірного цифрового обліку велосипедів.

Для розв’язання задачі доцільно використати клієнт-серверну архітектуру з окремою серверною частиною, яка виконує бізнес-логіку, та клієнтською частиною, що забезпечує зручну взаємодію з користувачем. Серверний API має перевіряти токени доступу, обмежувати дії за ролями, фільтрувати приватні поля та гарантувати, що змінювати запис велосипеда може лише його власник або користувач із відповідними повноваженнями.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Для реалізації системи використано платформу Node.js, серверний фреймворк NestJS, базу даних MongoDB із бібліотекою Mongoose, клієнтський фреймворк Next.js та сервіс Auth0 для автентифікації. Такий набір інструментів відповідає вимогам до модульності, розширюваності та безпечної обробки запитів.

Використання цього підходу забезпечує масштабованість, можливість подальшого розширення функціоналу та зрозумілий розподіл відповідальності між інтерфейсом, серверною логікою, сховищем даних і зовнішніми інтеграціями.

1.2. Аналіз аналогів програмного забезпечення

На ринку вже існують сервіси, які частково вирішують задачу реєстрації та пошуку велосипедів. Одним із відомих прикладів є Bike Index [5] (Рис. 1.1), де користувачі можуть створювати записи про велосипед, додавати опис, фотографії та позначати його як викрадений.

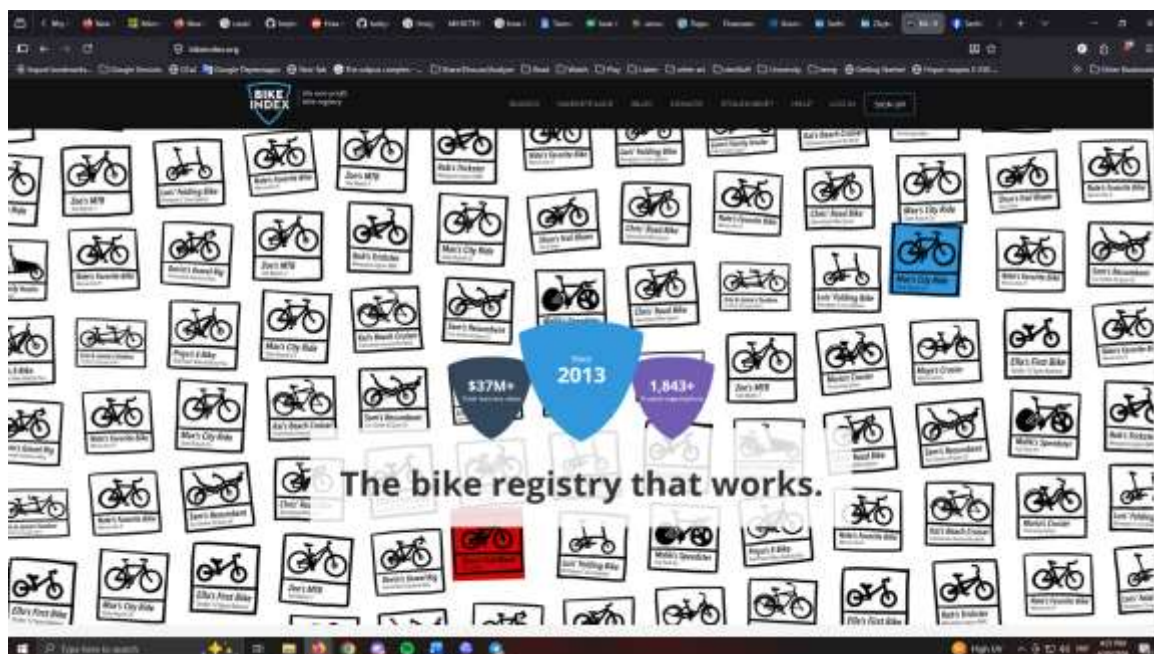


Рис. 1.1. Головна сторінка Bike Index

Bike Index є однією з найбільших відкритих некомерційних платформ реєстрації велосипедів, у базі якої налічуються сотні тисяч записів, зроблених користувачами з різних країн. Сервіс працює за моделлю спільноти: власники самостійно вносять інформацію про свої велосипеди, а у разі викрадення

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

позначають запис відповідним статусом, після чого оголошення стає публічним і поширюється серед учасників платформи, магазинів та сервісних майстерень. Окрему увагу приділено співпраці з правоохоронними органами й веломагазинами, які можуть перевіряти серійні номери під час купівлі-продажу вживаних велосипедів. Такий підхід підвищує ймовірність повернення викраденого транспортного засобу, проте більшість взаємодій ґрунтується на відкритості даних, а не на розмежуванні прав доступу між різними категоріями користувачів.

З функціональної точки зору Bike Index забезпечує зручний пошук, публічний профіль велосипеда та механізм повідомлення про викрадення. Водночас у розроблюваній системі додатково передбачено рольову взаємодію з адміністраторами та поліцією, а також ізольоване зберігання даних зовнішніх сервісів.

Сторінка пошуку в Bike Index має зручну можливість фільтрації та пошуку як за серійним номером, так і за описом велосипеда. (Рис. 1.2)

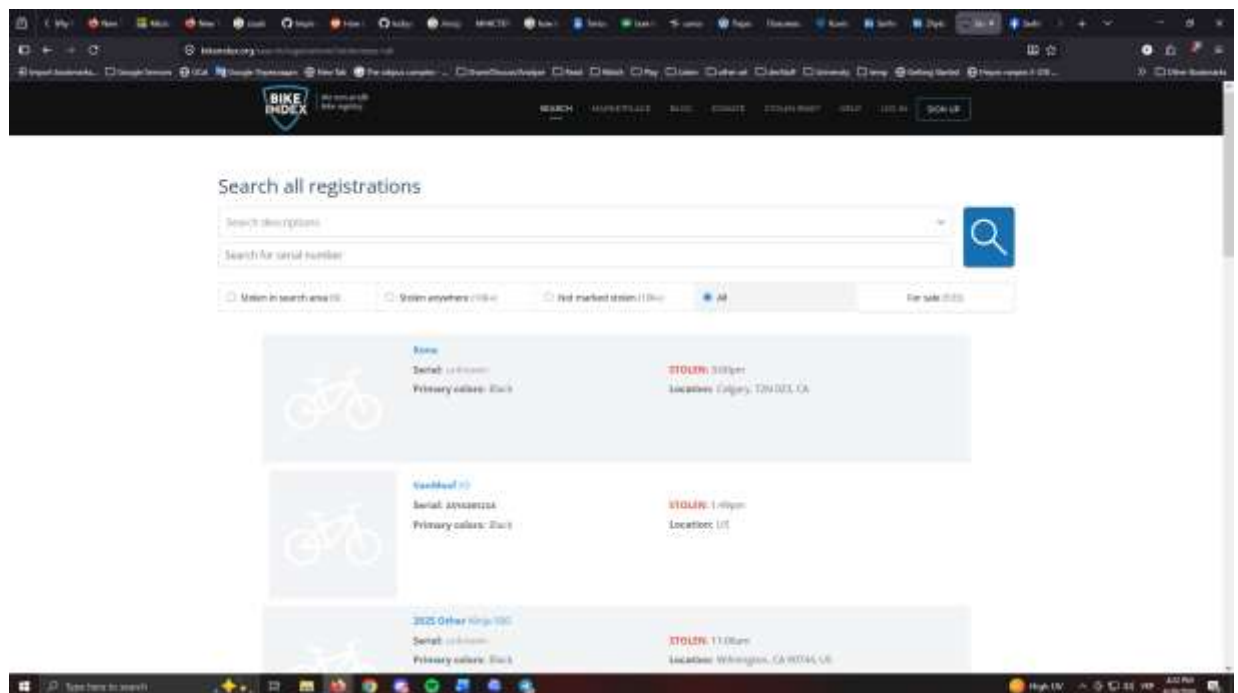


Рис. 1.2. Пошук велосипеда за ідентифікаційними даними

Іншим прикладом є Project 529 [25], який поєднує реєстрацію велосипеда, позначення його як викраденого та взаємодію з локальними спільнотами й

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

правоохоронними органами (Рис. 1.3). Сервіс демонструє практичну цінність централізованої інформації про велосипед у разі втрати або перевірки власності.

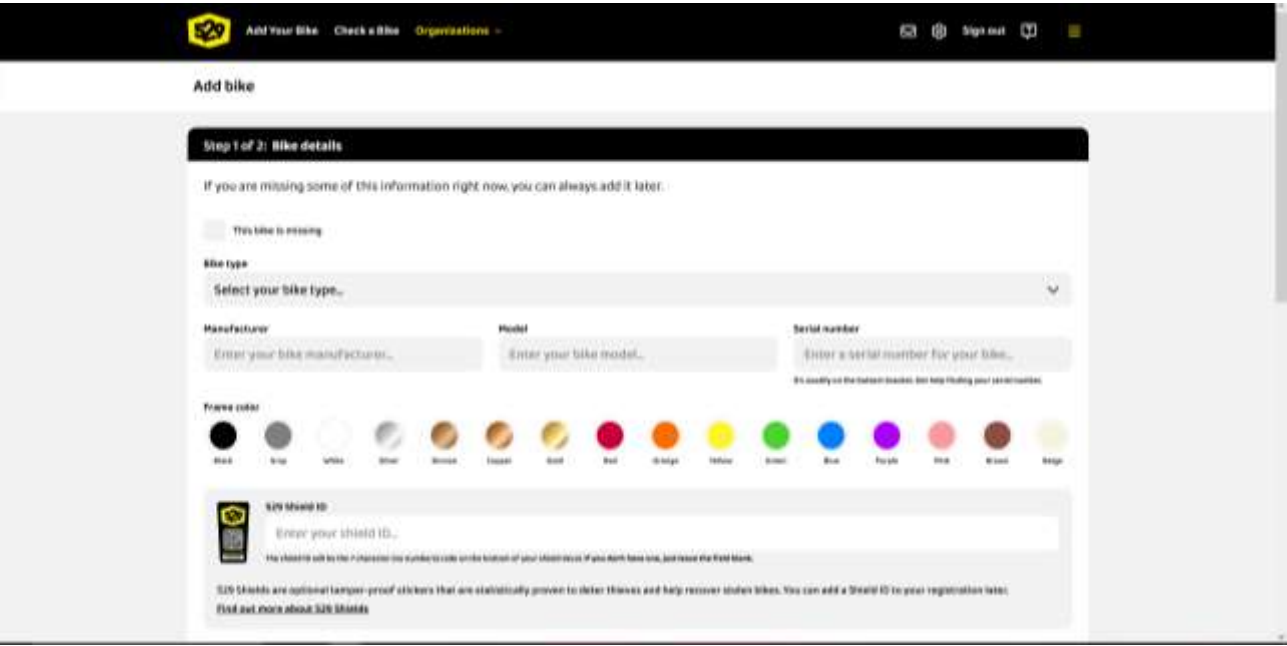


Рис. 1.3. Сторінка реєстрації велосипеда у Project 529

Особливістю Project 529 є поєднання програмної реєстрації з фізичною ідентифікацією велосипеда: користувачам пропонується захисний маркувальний знак (наклейка-щит), який наноситься на раму й пов'язується з обліковим записом велосипеда в системі. Реєстрація здійснюється через застосунок 529 Garage, де зберігаються серійний номер, фотографії та характеристики велосипеда, а у разі викрадення власник миттєво формує сповіщення, що розсилається спільноті користувачів і партнерам у відповідному регіоні. Сервіс тісно співпрацює з поліцейськими підрозділами низки країн, які використовують його як джерело даних під час розшуку та повернення велосипедів. Завдяки такому поєднанню онлайн-обліку, фізичного маркування та партнерської мережі Project 529 демонструє високу ефективність у протидії викраденням, водночас залишаючись зорієнтованим переважно на північноамериканський ринок.

Отже, наявні рішення підтверджують актуальність цифрового паспорта велосипеда, але їх функціональність і правила доступу не завжди відповідають потребам локальної вебсистеми з чітким розмежуванням ролей.

Аналогічні підходи застосовуються у сервісах на кшталт Project 529, де основний акцент зроблено на реєстрації, пошуку та спільній протидії викраденню велосипедів (Рис. 1.4). Такі системи демонструють, що відкриті дані про велосипед мають поєднуватися з приватними даними власника, які не повинні розкриватися стороннім користувачам.

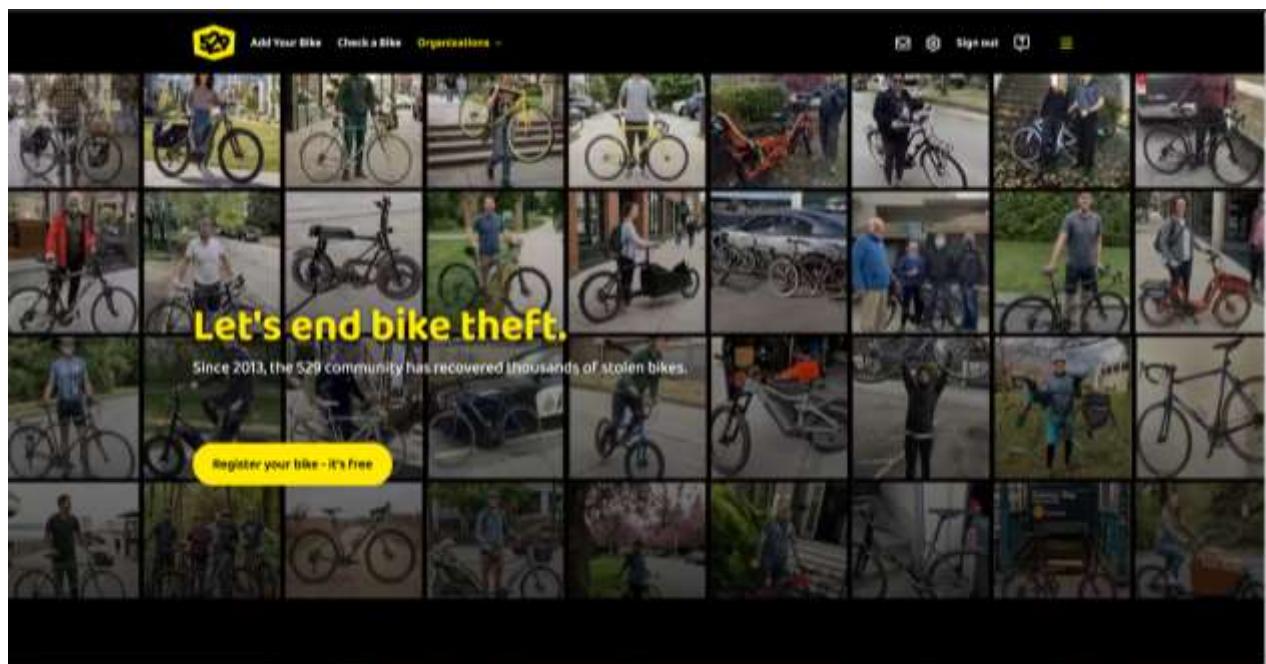


Рис. 1.4. Головна сторінка Project 529

На головній сторінці подібних сервісів зазвичай виділено дві дії: реєстрацію велосипеда та перевірку інформації про вже створений запис. Така структура є доречною і для Вело-паспорт, оскільки різні категорії користувачів мають різні сценарії взаємодії з вебсистемою.

Для власника ключовими є керування власними велосипедами, перегляд статусу, додавання фотографій та передача права власності. Для неавторизованого користувача основним сценарієм є публічний пошук за серійним номером, а для представника поліції — перегляд велосипедів, позначених як викрадені.

У сервісах-аналогах важливу роль відіграє форма пошуку та картка велосипеда, оскільки саме через них користувач отримує стислу відповідь щодо статусу транспортного засобу.

		Гларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

У Project 529 цей підхід реалізовано через головну сторінку з пошуковим полем та окрему сторінку деталей велосипеда (Рис. 1.5).

Рис. 1.5. Форма публічного пошуку велосипеда

Форма пошуку Project 529 дозволяє швидко перевірити велосипед за серійним номером або кодом наліпки й одразу побачити його статус.

Інтерфейс таких сервісів має бути лаконічним, оскільки користувач часто звертається до нього в ситуації термінової перевірки або повідомлення про проблему.

Окремої уваги заслуговує модель взаємодії Project 529 із зовнішніми учасниками процесу протидії викраденням. Сервіс функціонує не як ізольована база записів, а як платформа, що об'єднує власників велосипедів, спільноту користувачів, підрозділи поліції та страхові компанії: зареєстровані велосипеди стають доступними для перевірки правоохоронними органами, а у разі викрадення власник публікує повідомлення, яке розсилається учасникам спільноти в регіоні. Додатковим ідентифікатором слугує захищена від зняття наліпка з унікальним кодом, за QR-посиланням якої будь-який користувач може відкрити картку велосипеда та побачити, чи не оголошений він у розшук.

Такий підхід підтверджує, що цінність реєстру визначається не лише повнотою облікового запису, а й можливістю контролювано надавати частину даних зовнішнім учасникам. Водночас у Project 529 взаємодія з партнерами реалізована переважно на організаційному рівні, через окремі угоди та відомчі кабінети, тоді як програмний інтерфейс для сторонніх систем публічно не документований. Це ускладнює автоматичне підключення сервісних майстерень, магазинів або страхових платформ, які могли б доповнювати картку велосипеда власними службовими відомостями. Порівняння функціональних можливостей розглянутих сервісів з розроблюваною системою наведено у табл. 1.1.

Таблиця 1.1.

Таблиця порівняльного аналізу сервісів реєстрації велосипедів

Критерії порівняння	Bike Index	Project 529	Розроблювана вебсистема
Пошук велосипеда за серійним номером	+	+	+
Реєстрація велосипеда власником	+	+	+
Фіксація статусу викрадення	+	+	+
Передача права власності через одноразове посилання	-	-	+
Рольовий доступ для поліції та адміністратора	-	+	+
Завантаження фотографій велосипеда	+	+	+
Ізоляція даних зовнішніх сервісів	-	-	+
QR-посилання для передачі власності	-	-	+
Фільтрація приватних даних у публічному перегляді	+	+	+

1.3. Вибір архітектури веборієнтованої системи

Для реалізації вебсистеми Вело-паспорт було обрано клієнт-серверну архітектуру [11], яка забезпечує логічне розділення системи на інтерфейс, серверну бізнес-логіку, сховище даних та зовнішні сервіси. Такий підхід дозволяє ізолювати відповідальність компонентів і спрощує розвиток функціоналу без порушення роботи вже реалізованих сценаріїв.

Для власної системи важливим є поетапний розвиток: спочатку підтримка базового функціоналу — реєстрації велосипеда й пошуку за серійним номером, далі керування статусом, завантаження фотографій, передача власності та інтеграції. Клієнт-серверна організація дає змогу додавати ці можливості окремими модулями.

Перед вибором клієнт-серверної організації було розглянуто й інші архітектурні підходи. Монолітна архітектура із серверним рендерингом сторінок (класичний підхід MVC із шаблонізатором) є простою у початковій реалізації, проте погано пристосована до сценаріїв, у яких той самий набір даних має споживатися різними типами клієнтів — вебінтерфейсом власника, публічною сторінкою пошуку та машинними клієнтами сторонніх сервісів. Повністю статичний сайт було відхилено через потребу в динамічній обробці запитів, автентифікації та зміні стану записів. Мікросервісна архітектура, попри високу масштабованість, вносить надлишкову складність розгортання й міжсервісної взаємодії, що не виправдано для обсягу та етапу розвитку проєкту. З огляду на це клієнт-серверна архітектура з єдиним API та окремим клієнтським застосунком визнана оптимальним балансом між простотою реалізації та подальшою розширюваністю.

Розділення системи на незалежні клієнтську та серверну частини дає низку практичних переваг. Фронтенд і бекенд можуть розроблятися й розгортатися окремо, а єдиний програмний інтерфейс сервера повторно використовується всіма типами клієнтів без дублювання бізнес-логіки. Такий поділ спрощує модульне тестування кожного шару та локалізацію помилок, оскільки відповідальність за подання, обробку даних і зберігання чітко розмежована.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Серверна частина побудована на NestJS, що підтримує модульну структуру, контролери, сервіси, guards, interceptors і pipes. Завдяки цьому правила доступу, валідація даних, обробка бізнес-операцій та фільтрація відповіді розміщені у відповідних шарах системи.

Клієнтська частина реалізована за допомогою Next.js та React, що дозволяє поєднати сторінки публічного пошуку, особистий кабінет власника, сторінку деталей велосипеда, адміністративну панель і представлення для поліції в одному вебзастосунку.

Окремим рівнем виступає MongoDB, де зберігаються документи велосипедів і передач власності. Документно-орієнтована модель зручна для структури, у якій запис велосипеда містить базові поля, вкладені фотографії, історію статусів і карту даних сторонніх сервісів.

Таким чином, архітектура відповідає функціональним вимогам Вело-паспорт і створює основу для подальшого масштабування системи.

З огляду на потребу захищати дані власника, у системі використано зовнішній провайдер автентифікації Auth0, JWT-токени та рольову модель доступу. Це дозволяє не зберігати паролі користувачів у власній базі даних і покладати перевірку особи на спеціалізований сервіс.

Взаємодію між клієнтом і сервером організовано за принципами REST: обмін даними відбувається у форматі JSON, а кожен запит є самодостатнім (stateless) і супроводжується токеном доступу. Такий підхід узгоджується з наявними інструментами екосистеми Node.js, спрощує кешування та документування API і не потребує додаткової інфраструктури, на відміну від альтернативних протоколів на кшталт GraphQL чи gRPC, доцільність яких для поточного набору операцій є невисокою.

Для зберігання даних обрано документо-орієнтовану СУБД MongoDB, а не реляційну модель. Запис велосипеда за своєю природою є цілісним документом зі змінною структурою: він містить базові поля, вкладений масив фотографій, історію змін статусу та карту даних сторонніх сервісів `third_party_data`. Відображення такого запису на єдиний документ уникає складних з'єднань між

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

таблицями, спрощує читання під час публічного пошуку та природно підтримує поетапне розширення схеми без міграцій жорсткої структури.

Функції, критичні з погляду безпеки та надійності, винесено до спеціалізованих зовнішніх сервісів. Автентифікацію користувачів делеговано провайдеру Auth0, який видає підписані JWT-токени та надає публічні ключі через механізм JWKS, що дає змогу серверу перевіряти токени за алгоритмом RS256, не зберігаючи паролів у власній базі. Зберігання й доставку зображень покладено на хмарний сервіс Cloudinary. Винесення цих функцій назовні зменшує площу можливих вразливостей і знімає з системи навантаження, пов'язане з обробкою медіа.

Щоб проілюструвати взаємодію обраних компонентів, доцільно розглянути проходження типового запиту в системі. Користувач виконує дію в клієнтському застосунку, який формує HTTP-запит до серверного API та додає до нього токен доступу. На сервері запит послідовно проходить рівень автентифікації, де перевіряється підпис токена, потім рівень авторизації, на якому guards зіставляють ролі й дозволи користувача з вимогами конкретного маршруту, і лише після цього потрапляє до сервісного шару з бізнес-логікою. Сервіс звертається до MongoDB через Mongoose, отримує потрібні документи, а перед відправкою відповіді interceptor фільтрує приватні поля відповідно до прав користувача. Такий конвеєр забезпечує, що на кожному етапі обробки застосовуються відповідні правила безпеки, а клієнт отримує лише дозволений йому набір даних.

Загальну організацію системи та фізичне розміщення її компонентів відображає діаграма архітектури, на якій показано клієнтський застосунок, серверний API, базу даних MongoDB і зовнішні сервіси Auth0 та Cloudinary. Клієнт взаємодіє лише із серверним API та безпосередньо з Cloudinary під час завантаження зображень, тоді як сервер виступає єдиною точкою доступу до бази даних і посередником у перевірці токенів через Auth0. Чітке визначення напрямків взаємодії між компонентами спрощує розуміння системи, полегшує її

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

супровід і слугує основою для подальшого проектування, детально розглянутого у наступному розділі.

Обрана архітектура задовольняє й нефункціональні вимоги до системи. Незалежність шарів забезпечує підтримуваність і полегшує внесення змін; винесення автентифікації та медіа до зовнішніх сервісів підвищує відмовостійкість; а розділення клієнта й сервера дає змогу масштабувати їх окремо відповідно до навантаження. Разом це створює основу для сталого розвитку Вело-паспорт без потреби в докорінній переробці системи на подальших етапах.

Отже, обрана архітектура відповідає сучасним підходам до розробки веборієнтованих систем і забезпечує баланс між простотою реалізації, безпекою та можливістю розширення.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Основною мовою розробки було обрано TypeScript [30], що є типізованою надбудовою над JavaScript і дозволяє виявляти помилки на етапі розробки. Використання TypeScript є доцільним для системи, де серверна та клієнтська частини обмінюються структурованими даними через API.

Серверна частина реалізована у середовищі Node.js із використанням NestJS [20, 22]. Node.js забезпечує асинхронну модель обробки запитів, а NestJS формує чітку архітектуру застосунку через модулі, контролери, сервіси та механізми dependency injection.

Однією з ключових переваг NestJS є підтримка guards та interceptors, що використано для перевірки власності велосипеда, ролей користувача, машинного доступу та фільтрації приватних полів у відповіді API.

Для зберігання даних використано MongoDB та Mongoose [17, 19]. Такий вибір обґрунтовано тим, що сутність велосипеда містить як фіксовані поля, так і вкладені масиви фотографій, історію статусів та карту сторонніх даних, які природно зберігати у документній структурі.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Автентифікація реалізована через Auth0 і JWT [2, 12, 16]. Сервер перевіряє підпис токена за допомогою JWKS [15], контролює audience та issuer, а також отримує ролі й дозволи з claims токена. Це зменшує обсяг власної логіки керування обліковими записами.

Клієнтську частину створено на базі Next.js і React [21, 26]. Next.js використовується для маршрутизації та server actions, а React забезпечує побудову компонентного інтерфейсу, який складається з повторно використовуваних сторінок, таблиць, діалогів і форм.

Для UI-компонентів використано PrimeReact і PrimeIcons [23], а стилізація доповнена Tailwind CSS [29]. Це дозволило швидко реалізувати таблиці, кнопки, діалоги, теги статусу, підтвердження дій і адаптивну структуру сторінок.

Для зберігання фотографій велосипедів інтегровано Cloudinary [8]. Сервер створює підписані параметри завантаження, а клієнт передає зображення безпосередньо до хмарного сховища, після чого метадані фотографії зберігаються у документі велосипеда.

Для передачі велосипеда новому власнику застосовано одноразовий токен і QR-код. Посилання діє обмежений час, а у базі даних зберігається лише хеш токена, що зменшує ризики у разі витоку службових даних.

Docker Compose [10] використано для локального запуску трьох основних сервісів: MongoDB, backend_api та frontend_app. Такий підхід спрощує розгортання і дозволяє відтворювати однакове середовище на різних робочих станціях.

Крім того, у проєкті застосовано Jest та mongodb-memory-server [14, 18] для автоматизованого тестування серверної логіки. Це дає змогу перевіряти API без постійного зовнішнього екземпляра MongoDB.

Для контролю версій та організації роботи над кодом використано систему Git, що дає змогу відстежувати зміни й безпечно розвивати функціонал у межах окремих гілок. Конфіденційні параметри — рядки підключення до бази даних та ключі доступу до Auth0 і Cloudinary — винесено у змінні середовища й не зберігаються безпосередньо в коді, що підвищує безпеку та полегшує

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

перенесення застосунку між середовищами. Єдиний стиль коду забезпечено засобами статичного аналізу ESLint і форматувальником Prettier.

Обрання єдиного технологічного стеку на основі TypeScript та Node.js для клієнтської і серверної частин має додаткове обґрунтування. Спільна мова програмування дає змогу повторно використовувати типи даних і моделі між фронтом та бекендом, що зменшує кількість помилок під час передачі структур через API та спрощує узгодження змін. Єдиний набір інструментів збирання, тестування та статичного аналізу знижує поріг входження в проєкт і пришвидшує розробку, оскільки не потребує паралельної підтримки двох різних середовищ. Крім того, екосистема Node.js має широку базу готових бібліотек для роботи з JWT, HTTP-запитами та хмарними сервісами, що безпосередньо використано в реалізації Вело-паспорт.

Використання зазначених інструментів у клієнтській і серверній частинах вебсистеми є доцільним, оскільки забезпечує продуктивність, контроль доступу, зручний інтерфейс і достатню основу для подальшого розвитку.

Отже, обраний набір інструментальних засобів і технологій відповідає сучасним вимогам до розробки веборієнтованих систем.

Для забезпечення стабільної роботи вебсистеми Вело-паспорт визначено вимоги до апаратного забезпечення та середовища виконання. Оскільки система обробляє запити користувачів, зберігає фотографії та взаємодіє із зовнішніми сервісами, інфраструктура повинна забезпечувати стабільне мережеве з'єднання, достатній обсяг пам'яті та можливість резервного копіювання.

Серверне середовище виконання потребує процесора з тактовою частотою не менше 2,5 ГГц та щонайменше 8 ГБ оперативної пам'яті, оскільки одночасно працюють контейнери бази даних і серверного API. Для самої бази даних, журналів роботи та службових файлів необхідно передбачити не менше 50 ГБ. Обов'язковими також є стабільне з'єднання з мережею Інтернету, без якого неможлива взаємодія з Auth0 і Cloudinary, і операційна система із підтримкою Docker, оскільки всі послуги запускаються в контейнерах.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Наведені апаратні вимоги розраховано на базове навантаження, характерне для етапу впровадження та початкової експлуатації системи. У разі зростання кількості записів і одночасних користувачів обсяг оперативної пам'яті та дискового простору серверного середовища може бути збільшений, а завдяки контейнерній організації окремі сервіси — насамперед серверний API — допускають горизонтальне масштабування без зміни програмного коду. Оскільки зберігання зображень винесено до Cloudinary, а автентифікація — до Auth0, навантаження на власну інфраструктуру залишається помірним навіть за суттєвого збільшення обсягу даних.

Вимоги до клієнтських пристроїв суттєво нижчі, адже вся бізнес-логіка виконується на сервері: достатньо процесора з частотою від 2,0 ГГц, 4 ГБ оперативної пам'яті та доступу до мережі Інтернет. Єдиною спеціальною умовою є наявність сучасного веббраузера з підтримкою JavaScript, оскільки клієнтську частину реалізовано як односторінковий вебзастосунок, що створює інтерфейс безпосередньо в браузері користувача.

Для зберігання інформації система використовує MongoDB. СУБД має підтримувати індексацію серійного номера велосипеда, резервне копіювання, відновлення даних та стабільну роботу з вкладеними документами.

Система повинна забезпечувати доступність публічного пошуку та особистого кабінету, а також підтримувати одночасну роботу власників, адміністраторів, представників поліції та машинних клієнтів.

Оскільки система обробляє персональні дані власників та службову інформацію про велосипеди [24], необхідно застосувати автентифікацію, рольове розмежування доступу, перевірку власності запису, фільтрацію приватних полів, обмеження прав машинних клієнтів. Сукупність цих вимог визначає базовий рівень захищеності та надійності системи, який має бути забезпечений незалежно від обраної конфігурації апаратного забезпечення. Виконання цих вимог є обов'язковою умовою для надійної та безпечної роботи системи в реальних умовах використання.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Висновки до першого розділу

У першому пункті було проаналізовано задачу створення вебсистеми цифрового паспорта велосипеда, визначено проблеми розрізненого обліку, підтвердження власності та оперативної перевірки статусу велосипеда.

У другому пункті досліджено аналоги сервісів реєстрації велосипедів, визначено їхні функціональні можливості та обмеження, які враховано під час створення Вело-паспорт.

У третьому пункті було обґрунтовано вибір клієнт-серверної архітектури, документно-орієнтованої бази даних та сучасних засобів автентифікації, що забезпечують модульність, безпеку й масштабованість системи.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСИСТЕМИ ВЕЛОСИПЕДНОГО ПАСПОРТУ.

2.1. Визначення варіантів використання та структури вебсистеми

Архітектура вебсистеми побудована на рольовій моделі доступу [27] з чітким розмежуванням функціональних можливостей користувачів. Основними учасниками системи є власник велосипеда, представник поліції, адміністратор, публічний користувач та зовнішній машинний клієнт. Кожна роль виконує окрему логічну функцію у процесі створення, перевірки, оновлення та захисту цифрового паспорта велосипеда.

Адміністратор у системі відповідає за загальний контроль платформи. Він має доступ до повного переліку велосипедів, може переглядати статуси, фільтрувати записи та змінювати стан велосипеда у межах модерації. Адміністратор має дозвіл на видалення паспортів велосипедів та перегляд сторінки поліції.

Власник велосипеда є ключовим суб'єктом системи. Він створює цифровий паспорт, додає серійний номер, виробника, модель і фотографії, змінює статус велосипеда, переглядає історію змін, редагує базову інформацію та формує одноразове посилання для передачі велосипеда іншому користувачу.

Представник поліції працює з переліком велосипедів, позначених як викрадені, і може змінювати статус на «recovered», якщо велосипед знайдено. Публічний користувач без авторизації виконує лише перевірку за серійним номером.

Зовнішній машинний клієнт (M2M) взаємодіє з API у суворо ізольованому просторі `third_party_data`. Це унеможливорює доступ партнерів до даних один одного, але дозволяє позначати окремі параметри як публічні для їх показу під час загальної перевірки велосипеда.

Діаграма ілюструє взаємодію п'яти акторів із порталом «Вело-паспорт»: адміністратор та власник управляють цифровими паспортами й статусами велосипедів, поліція переглядає викрадені записи та змінює їхній стан,

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

публічний користувач здійснює пошук за серійним номером, а зовнішній M2M-сервіс записує службові дані через API. (Рис. 2.1)

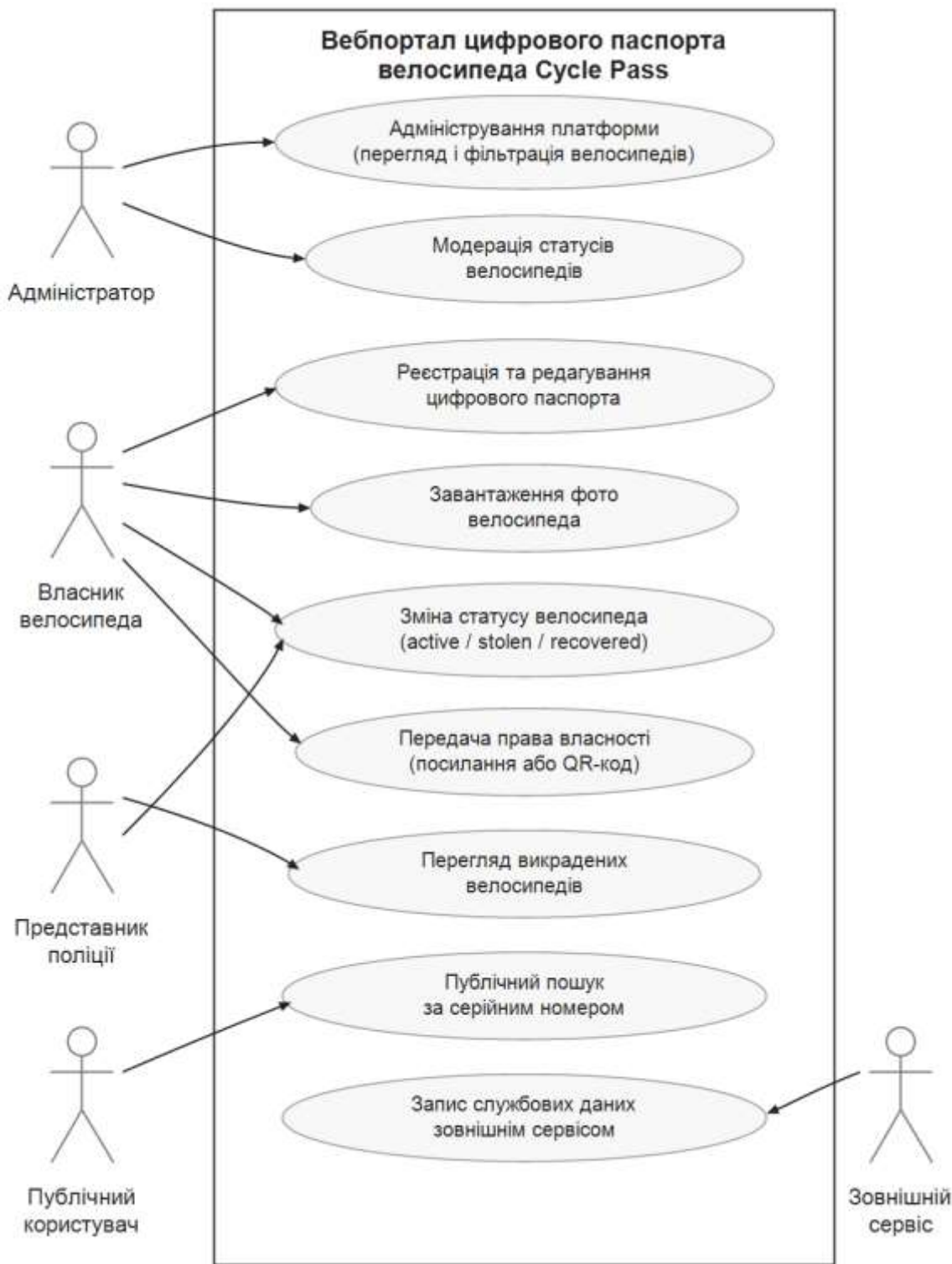


Рис. 2.1. Діаграма використання

Опис основних прецедентів наведено у таблицях 2.1 – 2.3.

Таблиця 2.1.

Опис прецеденту актора «Адміністратор»

Назва варіанту використання	Основні актори	Короткий опис
Перегляд усіх велосипедів	Адміністратор	Отримання повного списку записів платформи з можливістю фільтрації за статусом.
Модерація статусу	Адміністратор	Зміна статусу велосипеда на active, stolen або recovered у межах контролю платформи.
Перехід до поліцейського представлення	Адміністратор	Швидкий доступ до списку велосипедів, позначених як викрадені.

Даний блок охоплює адміністративні сценарії вебсистеми, пов'язані з переглядом повного переліку велосипедів, фільтрацією за статусом, контролем записів і переходом до таблиці для роботи з викраденими велосипедами.

Таблиця 2.2.

Опис прецеденту актора «Власник велосипеда»

Назва варіанту використання	Основні актори	Короткий опис
Реєстрація велосипеда	Власник	Створення цифрового паспорта із серійним номером, виробником і моделлю.
Керування статусом	Власник	Позначення велосипеда як активного, викраденого або відновленого.
Завантаження фотографій	Власник	Додавання публічних або приватних фотографій до запису велосипеда.
Передача власності	Власник	Створення одноразового посилання або QR-коду для передачі велосипеда.
Редагування паспорта	Власник	Зміна виробника та моделі велосипеда без зміни серійного номера.
Передача власності	Власник	Створення одноразового посилання або QR-коду для передачі велосипеда.
Видалення велосипеда	Власник	Видалення велосипеда з особистого кабінету після підтвердження дії.

Даний блок охоплює основні сценарії власника велосипеда: реєстрацію цифрового паспорта із серійним номером, виробником і моделлю, керування

статусом велосипеда, завантаження публічних і приватних фотографій, редагування базових даних без зміни серійного номера, а також формування одноразового посилання або QR-коду для передачі велосипеда та його видалення з особистого кабінету після підтвердження дії. Саме власник є найактивнішим користувачем системи, тому перелічені прецеденти становлять основу її повсякденного використання й безпосередньо впливають на зручність роботи з цифровим паспортом велосипеда.

Таблиця 2.3.

Опис прецедентів публічного користувача, поліції та зовнішнього сервісу

Назва варіанту використання	Основні актори	Короткий опис
Пошук велосипеда	Неавторизований користувач	Перевірка велосипеда за серійним номером без доступу до приватних даних власника.
Перегляд викрадених велосипедів	Поліція	Отримання списку записів зі статусом stolen і можливість позначити велосипед як recovered.
Запис сторонніх даних	Машинний клієнт	Збереження службового JSON-об'єкта у власному просторі third_party_data.
Передача велосипеда	Новий власник	Використання одноразового токена для отримання права власності на велосипед.
Перегляд загальнодоступної інформації про велосипед	Публічний користувач	Отримання серійного номера, моделі, виробника, статусу та публічних фотографій.

Цей розділ описує сценарії, у яких з системою взаємодіють користувачі без повного доступу до запису велосипеда. Пошук велосипеда повертає лише відкриті дані, представник поліції працює з викраденими велосипедами, а машинний клієнт записує дані у власний простір інтеграції.

Таким чином, структура вебсистеми має чітке розмежування ролей, контроль доступу до приватних даних і логічно організовану взаємодію між власниками, адміністраторами, поліцією та зовнішніми сервісами.

Програмний комплекс спроектовано за допомогою клієнт-серверної архітектури (Рис. 2.2), що гарантує чіткий розподіл відповідальності між компонентами вебсистеми. Інформаційна взаємодія між клієнтським та серверним компонентом реалізується через HTTP-запити до REST API [13].

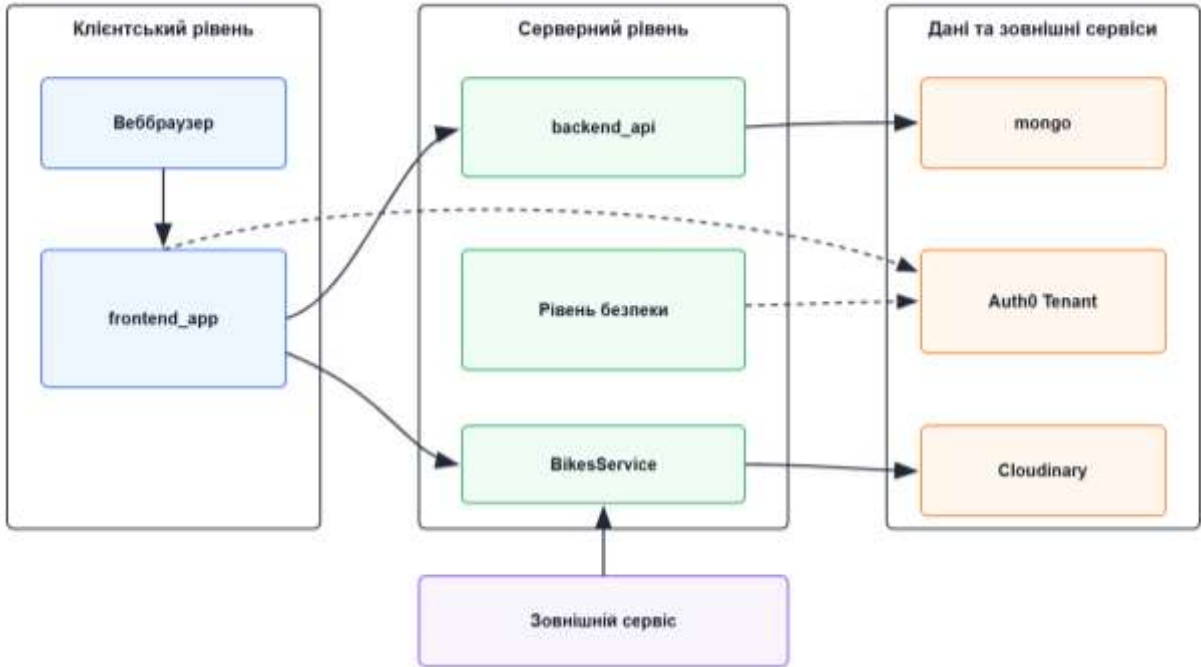


Рис. 2.2. Діаграма клієнт-серверної архітектури

Структуру даного архітектурного підходу можна розділити на три логічні рівні: рівень клієнта, рівень сервера та рівень зберігання даних.

Клієнтський рівень формується сторінками Next.js і компонентами React, які відповідають за пошук велосипеда, особистий кабінет, сторінку деталей, адміністративну панель, представлення поліції та сторінку прийняття передачі власності.

Серверний рівень вебсистеми відповідає за бізнес-логіку та контроль доступу. Усі запити надходять до NestJS API, де проходять через guards, validation pipe та, за потреби, через interceptor фільтрації даних. Контролери приймають запити, а сервіси виконують операції з базою даних.

Завершальним структурним компонентом є рівень зберігання даних. Основна інформація про велосипеди та передачі власності зберігається у

MongoDB, а фотографії завантажуються до Cloudinary, після чого в документі велосипеда зберігаються лише метадані та посилання на зображення.

Окремо від бази даних працює Auth0, який забезпечує автентифікацію користувачів і видає JWT-токени. Серверна частина перевіряє ці токени, читає ролі та дозволи, після чого приймає рішення щодо доступу до конкретної операції.

Отже, клієнт-серверна архітектура повністю відповідає вимогам розробки вебсистеми Вело-паспорт.

2.2. Розробка баз даних

Для повноцінної реалізації вебсистеми було розроблено документно-орієнтовану модель даних, яка забезпечує зберігання, обробку та цілісність інформації про велосипеди, фотографії, історію статусів, дані сторонніх сервісів і передачі права власності. Для реалізації бази даних використано MongoDB і Mongoose [9].

Розроблена структура бази даних відображає основні сутності системи без зайвої нормалізації, оскільки документ велосипеда природно об'єднує серійний номер, виробника, модель, власника, статус, фотографії та історію змін. Це зменшує кількість додаткових запитів і спрощує отримання повної картки велосипеда.

Вибір документної моделі замість реляційної зумовлений самим характером даних цифрового паспорта. У реляційній схемі один велосипед потребував би щонайменше чотирьох таблиць: основних даних, фотографій, історії статусів і даних сторонніх сервісів, а формування картки вимагало б об'єднання цих таблиць при кожному зверненні. Оскільки фотографії, історія змін і службові дані партнерів не мають самостійного значення поза конкретним велосипедом і майже завжди читаються разом із ним, зберігання їх у вкладених структурах одного документа зводить отримання повної картки до одного запиту за ключем. Додатковим аргументом є змінна структура поля `third_party_data`: кожен зовнішній сервіс записує довільний набір полів, який неможливо описати

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

фіксованою реляційною схемою без створення окремої таблиці на кожну інтеграцію.

Центральною сутністю є Bike, що містить унікальний серійний номер, поточного власника, статус і вкладені масиви фотографій та історії зміни стану. Статус може набувати значень active, stolen або recovered, а кожна зміна фіксується з часом та автором дії.

Окремою сутністю є Transfer, яка описує процес передачі велосипеда новому власнику. У ній зберігаються ідентифікатор велосипеда, хеш одноразового токена, електронна пошта попереднього власника, час завершення дії посилення та ознака використання.

Для роботи з фотографіями у документі велосипеда передбачено вкладений масив photos. Кожен елемент містить провайдера, ідентифікатор у Cloudinary, URL доставки, тип файлу, розмір, видимість, час завантаження та автора завантаження.

Для інтеграції із зовнішніми сервісами використано поле third_party_data у вигляді Map. Ключем є client_id машинного клієнта, а значенням — довільний JSON-об'єкт, який належить саме цьому сервісу.

Контроль доступу до даних забезпечується не окремими таблицями, а логікою guards та interceptor. Власник бачить повну інформацію про свій велосипед, адміністратор і поліція мають розширений доступ, сторонній користувач бачить лише публічні поля, а машинний клієнт бачить тільки власний простір даних.

На рівні схеми Mongoose визначено обмеження цілісності, які не залежать від клієнтської частини. Поле serialNumber оголошено обов'язковим і унікальним із приведенням до верхнього регістру та видаленням крайніх пробілів, тому база даних сама відхиляє спробу зареєструвати другий велосипед із тим самим номером незалежно від того, як його набрав користувач. Поле ownerEmail нормалізується до нижнього регістру та індексується, оскільки саме за ним виконується вибірка велосипедів для особистого кабінету та порівняння під час перевірки власності. Значення статусу обмежено перерахуванням

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

BikeStatus, що виключає появу довільних станів, а часові мітки createdAt і updatedAt формуються автоматично засобами Mongoose.

Індекс за серійним номером забезпечує стабільну швидкодію публічного пошуку незалежно від зростання кількості записів, адже це найчастіша операція в системі й єдина, доступна без автентифікації. Колекція передач додатково індексується за ідентифікатором велосипеда, що дає змогу швидко виявляти й скасовувати невикористані посилання перед створенням нового.

Детальний опис структур даних та пов'язаних компонентів наведено у таблицях 2.4 – 2.7.

Основну сутність предметної області описує таблиця bike, структуру якої наведено в табл. 2.4. Вона містить ключові поля запису велосипеда — виробника, модель, унікальний серійний номер, електронну пошту власника та поточний статус, а також службові часові мітки. Саме ця таблиця є центральною у моделі даних, оскільки з нею пов'язані всі інші структури.

Таблиця 2.4.

Таблиця bike

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа велосипеда.
serialNumber	string	Унікальний серійний номер, збережений у верхньому регістрі.
make	string	Виробник велосипеда.
model	string	Модель велосипеда.
ownerEmail	string	Електронна пошта поточного власника.
status	BikeStatus	Поточний статус: active, stolen або recovered.
third_party_data	Map	Дані зовнішніх сервісів, розділені за client_id.
createdAt / updatedAt	Date	Службові часові мітки MongoDB.

Фотографії велосипеда зберігаються у вигляді вкладеного масиву об'єктів photos, структуру якого показано в табл. 2.5. Кожен елемент масиву описує окреме зображення й містить його ідентифікатор у Cloudinary, URL, ознаку видимості та час завантаження, що дає змогу зберігати кілька фотографій у межах одного документа велосипеда.

Таблиця 2.5.

Об'єкт photos у схемі bike

Поле	Тип	Призначення
provider	string	Провайдер зберігання зображення, у системі використано Cloudinary.
providerId	string	Ідентифікатор файлу у Cloudinary.
cdnUrl	string	Посилання на доставку зображення.
originalFilename	string	Початкова назва файлу.
mimeType	string	Тип файлу, наприклад image/jpeg.
size	number	Розмір файлу у байтах.
visibility	BikePhotoVisibility	Рівень видимості: public або private.
uploadedAt	Date	Час завантаження фотографії.

Для взаємодії зі сторонніми застосунками призначено об'єкт third_party_data, поля якого описано в табл. 2.6. Він реалізований як відображення, у якому кожному ідентифікатору машинного клієнта відповідає ізольований простір службових даних, доступний лише цьому клієнтові згідно з правилами фільтрації.

Таблиця 2.6.

Об'єкт third_party_data

Поле	Тип	Призначення
client_id	string	Ключ простору даних зовнішнього сервісу.
data	object	Довільний JSON-об'єкт, який записує машинний клієнт.

Продовження таблиці 2.6.

scope	string	Для запису потрібен дозвіл write:third-party-data.
Ізоляція	interceptor	Машинний клієнт бачить тільки власний простір.
Власник	full access	Власник бачить усі простори сторонніх сервісів.
Сторонній користувач	hidden	Для користувача без прав поле повертається порожнім.

Процес передачі права власності описується окремою колекцією transfer, структуру якої наведено в табл. 2.7. Вона зберігає хеш одноразового токена, ідентифікатор велосипеда, термін дії посилання та ознаку використання, що забезпечує безпечний і контрольований механізм зміни власника.

Таблиця 2.7.

Таблиця transfer

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор запису передачі.
bikeId	string	Ідентифікатор велосипеда, що передається.
tokenHash	string	SHA-256-хеш одноразового токена.
fromEmail	string	Email попереднього власника.
expiresAt	Date	Час завершення дії посилання.
usedAt	Date null	Час використання токена або null.
createdAt / updatedAt	Date	Службові часові мітки MongoDB.

Отже, розроблена база даних забезпечує ефективну організацію зберігання та підтримує безпечність та ізолюваність даних між ключовими сутностями вебсистеми. Документна структура дозволяє зберігати всі важливі дані цифрового паспорта в одному записі, а допоміжна колекція передач власності забезпечує безпечну зміну власника.

2.3. Реалізація функціоналу вебреєстру велосипедів

У вебсистемі Вело-паспорт реалізовано багаторівневий комплекс заходів із захисту персональних даних, збереження конфіденційності та суворого контролю доступу до функціоналу цифрового паспорта велосипеда. Фундаментом системи ідентифікації виступає хмарний сервіс Auth0, який відповідає за автентифікацію користувачів та видачу криптографічно захищених JSON Web Tokens (JWT). Серверна частина на NestJS працює за принципом Zero Trust і перевіряє кожен вхідний токен у критичних маршрутах за допомогою асиметричного алгоритму шифрування RS256. Під час обробки запиту сервер динамічно завантажує публічні ключі через JWKS та перевіряє коректність параметрів issuer, що підтверджує авторизований сервер Auth0, та audience, що гарантує призначення токена саме для API вебсистеми Вело-паспорт. Після успішної перевірки токена сервер формує об'єкт req.user, у якому містяться ідентифікатор користувача, електронна пошта, client_id, набір scopes і ролі, що надалі використовуються компонентами захисту (Guards) для прийняття рішень щодо доступу до ресурсів.

Для захисту операцій над конкретним велосипедом реалізовано BikeOwnershipGuard, який поєднує рольову та атрибутивну моделі доступу. З параметрів URL-маршруту даний модуль отримує ідентифікатор велосипеда, завантажує відповідний запис із MongoDB та зіставляє поле ownerEmail із електронною поштою користувача з токена. У разі збігу даних доступ надається, а для забезпечення роботи правоохоронців і модераторів передбачено привілейований обхід перевірки власності для користувачів із ролями police та admin.

Окрім перевірки власності, у системі реалізовано рольове розмежування доступу за допомогою RolesGuard. Кожному маршруту, що потребує підвищених прав, призначається перелік дозволених ролей, а модуль захисту зіставляє їх із ролями, отриманими з токена користувача. Такий підхід дає змогу чітко розмежувати можливості власника велосипеда, представника поліції та адміністратора: власник керує лише своїми записами, поліція має доступ до

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

перегляду позначених велосипедів, а адміністратор виконує модерацію та контроль платформи. Окремим випадком є машинні клієнти сторонніх сервісів, доступ яких обмежується не ролями, а набором scopes у токені та спеціальним модулем ScopesGuard, що перевіряє наявність конкретного дозволу, наприклад write:third-party-data. Розділення прав на рівні окремих маршрутів спрощує подальше розширення системи новими ролями та дозволами без зміни наявної бізнес-логіки й зменшує ймовірність помилок під час надання доступу.

Функціонал передачі права власності побудовано на використанні одноразових тимчасових токенів підвищеної стійкості. Під час ініціалізації процедури система генерує випадковий токен, але зберігає в базі даних лише його SHA-256-хеш [28], запобігаючи витoku у разі скомпрометованості бази даних. Для запису встановлюється термін дії у 72 години, після чого токен анулюється, а під час прийняття велосипеда новим власником надісланий токен хешується, зіставляється з базою даних, змінює ownerEmail та позначається як використаний.

Для роботи з медіафайлами застосовано шаблон прямого підписаного завантаження до Cloudinary, що мінімізує навантаження на основний сервер застосунку. Сервер генерує часову позначку timestamp, цільову папку folder та цифровий підпис signature з використанням секретного ключа, а клієнт здійснює завантаження безпосередньо до хмарного сховища, після чого у базі даних зберігаються метадані фотографії та атрибут видимості public або private.

Усі вхідні дані, що надходять до серверної частини, проходять перевірку за допомогою DTO-класів і бібліотеки class-validator. Для кожного маршруту описано очікувану структуру запиту, типи полів та обмеження їхніх значень — наприклад, формат серійного номера, допустиму довжину текстових полів та перелік дозволених значень статусу велосипеда. Це знижує ризик збереження помилкових записів, захищає систему від типових помилок введення та зловмисних запитів і водночас полегшує роботу з інтерфейсом, оскільки користувач одразу бачить причину відмови.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Обробку помилок централізовано на рівні серверного застосунку: винятки, що виникають у сервісах і контролерах, перетворюються на уніфіковані HTTP-відповіді з відповідним кодом стану та описом причини. Завдяки цьому клієнтська частина обробляє помилки за єдиною схемою й відображає користувачеві зрозумілі сповіщення, а службові подробиці, які не призначені для показу, не потрапляють у відповідь. Такий підхід підвищує передбачуваність поведінки API та спрощує його налагодження під час подальшого розвитку системи.

Центральним елементом захисту відповідей API є `BikeDataFilterInterceptor`, який здійснює динамічну фільтрацію даних перед відправкою. Для публічних користувачів та гостей перехоплювач приховує електронну пошту власника, історію статусів `statusHistory`, приватні фотографії та сторонні дані, тоді як для машинних клієнтів реалізовано сувору багатоорендність, залишаючи у полі `third_party_data` лише інформацію, яка відповідає їхньому `client_id`.

Окремо реалізовано механізм зміни статусу велосипеда, який відображає його поточний стан — «активний», «викрадений» або «повернений». Кожна зміна статусу не лише оновлює відповідне поле запису, а й додає новий елемент до масиву `statusHistory` із зазначенням попереднього та нового значень і часу операції, завдяки чому зберігається повна хронологія станів велосипеда. Право змінювати статус розмежоване: власник може позначити свій велосипед як викрадений або повернений, тоді як представник поліції та адміністратор мають розширені повноваження в межах модерації. Історія статусів доступна лише авторизованим користувачам із відповідними правами й приховується від публічного перегляду, що дає змогу поєднати прозорість обліку з захистом чутливої інформації.

Для підтвердження надійності розроблених рішень створено наскрізні тести (e2e), які перевіряють повний життєвий цикл системи в закритому середовищі. Вони покривають автентифікацію, перевірку власності через модуль захисту, зміну статусів, ізоляцію даних машинних клієнтів, фільтрацію приватних полів, пряме завантаження фотографій, формування списків для

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

поліції та адміністраторів, а також процес передачі власності за одноразовим токеном.

Таким чином, реалізований функціонал вебреєстру велосипедів поєднує захищену автентифікацію, багаторівневий контроль доступу, перевірку вхідних даних, безпечну передачу власності та ізоляцію даних різних категорій користувачів. Поєднання цих механізмів забезпечує цілісність облікових записів велосипедів, передбачувану поведінку системи в разі помилок і створює надійну основу для подальшого підключення зовнішніх сервісів та розширення можливостей вебсистеми Вело-паспорт.

Висновки до другого розділу

У першому пункті було розглянуто ключові аспекти проектування вебсистеми Вело-паспорт, де основну увагу приділено формуванню функціональних можливостей відповідно до ролей власника, адміністратора, поліції, публічного користувача та машинного клієнта.

Також детально описано логіку взаємодії між користувачами, зокрема створення цифрового паспорта, перевірку статусу велосипеда, завантаження фотографій, передачу власності та доступ зовнішніх сервісів.

У другому пункті значну увагу приділено проектуванню документної бази даних MongoDB, яка є основою порталу. Було розглянуто структуру сутностей Bike і Transfer, вкладені фотографії та дані сторонніх сервісів.

У третьому пункті детально розглянуто функціональну реалізацію, що відповідає за автентифікацію, перевірку власності, рольовий доступ, фільтрацію даних і безпечну передачу велосипеда новому власнику.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБСИСТЕМИ

3.1. Розгортання вебсистеми

Для опису архітектури та процесів розгортання була створена діаграма розгортання (див. Додаток Г), яка демонструє взаємодію компонентів вебсистеми Вело-паспорт. Дана діаграма дозволяє зрозуміти, як клієнтська частина, серверний API, база даних, Auth0 і Cloudinary розподіляються між сервісами.

Клієнтська частина розгортається як окремий контейнер `frontend_app`, у якому Next.js-застосунок збирається у виробничому режимі та публікується на власному порту. Адреса серверного API і параметри домену Auth0 передаються через змінні середовища, тому той самий образ використовується як у локальному, так і в тестовому середовищі без зміни коду.

Серверна частина розгортається контейнером `backend_api`, який отримує через змінні середовища рядок підключення до MongoDB, адресу JWKS та ключі доступу до Cloudinary. Контейнер запускається після бази даних, оскільки під час старту Mongoose встановлює з'єднання і створює унікальний індекс за серійним номером велосипеда.

MongoDB розміщено як окремий контейнер `docker-compose`. Фотографії велосипедів не зберігаються у базі як бінарні файли, а передаються до Cloudinary, що зменшує навантаження на базу даних і спрощує доставку зображень.

Запуск середовища виконується однією командою `docker compose up`, після чого сервіси піднімаються у визначеному порядку: спочатку контейнер бази даних з окремим томом, що зберігає документи між перезапусками, далі `backend_api`, а останнім `frontend_app`. Auth0 і Cloudinary залишаються зовнішніми сервісами, тому в конфігурації зберігаються лише параметри доступу до них, а не окремі контейнери, і саме цю фізичну організацію відображає діаграма розгортання.

Конфігурування системи повністю винесено у змінні середовища, що дає змогу використовувати однакові образи контейнерів у різних середовищах без перезбирання. У файлі `docker-compose` визначено окремий набір параметрів для

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

кожного сервісу: рядок підключення до MongoDB, домен та ідентифікатори Auth0, ключі доступу до Cloudinary, а також адреси, за якими сервіси знаходять один одного у внутрішній мережі контейнерів. Конфіденційні значення не потрапляють до системи контролю версій, а зберігаються в окремому файлі оточення, що зчитується під час запуску. Такий підхід відповідає принципу відокремлення конфігурації від коду й спрощує перенесення застосунку на інший сервер або хмарну платформу.

Порядок запуску контейнерів визначається їхніми взаємними залежностями. Контейнер бази даних піднімається першим, оскільки серверна частина під час старту встановлює з'єднання з MongoDB і створює необхідні індекси; лише після готовності бази запускається `backend_api`, а вже потім `frontend_app`, який звертається до серверного API. Для контролю готовності сервісів використовуються базові перевірки стану, що дають змогу уникнути ситуації, коли застосунок звертається до ще не ініціалізованої бази даних. Завдяки цьому середовище піднімається у передбачуваному стані навіть за одночасного старту всіх контейнерів.

Для збереження даних між перезапусками контейнера MongoDB використовує окремий іменованій том `Docker cycle_pass_data`, змонтований у каталог `/data/db`. У ньому зберігаються документи велосипедів і передач власності. Відокремлення сховища від життєвого циклу контейнера дає змогу зберігати дані після його звичайного перезапуску або повторного створення, якщо сам том не видаляється. Медіафайли зберігаються у зовнішньому сервісі Cloudinary, тоді як база даних містить лише пов'язані з ними метадані та URL. За потреби резервне копіювання тому або бази MongoDB може бути організоване окремо, однак автоматичний механізм резервного копіювання в поточній реалізації не передбачений.

Взаємодія між сервісами організована через внутрішню мережу `docker-compose`, у якій контейнери звертаються один до одного за іменами сервісів, а не за фіксованими IP-адресами. Назовні публікуються лише порти, необхідні для доступу користувача до клієнтського застосунку та, за потреби, до серверного

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

API. Зовнішні сервіси Auth0 і Cloudinary доступні через мережу Інтернет, тому в конфігурації зберігаються тільки параметри підключення до них. Така організація зменшує площу відкритих інтерфейсів і підвищує захищеність розгорнутої системи.

Описана конфігурація призначена насамперед для локального й тестового середовища, проте вона легко адаптується до промислового розгортання. Для роботи у виробничому режимі достатньо розмістити контейнери на віддаленому сервері або в хмарі, підключити доменне ім'я з сертифікатом TLS і налаштувати зворотний проксі, який спрямовуватиме запити до відповідних сервісів. Оскільки застосунок уже розділений на незалежні контейнери та конфігурується через змінні середовища, перехід до промислового середовища не потребує зміни програмного коду, а лише оновлення параметрів розгортання.

Для діагностики роботи розгорнутої системи використовується стандартний механізм журналювання контейнерів. Кожен сервіс виводить повідомлення про свою роботу у стандартний потік виведення, звідки їх можна переглянути командою `docker compose logs` як для всіх контейнерів одночасно, так і для окремого сервісу. Серверна частина фіксує ключові події, старт застосунку, встановлення з'єднання з базою даних, помилки автентифікації та відхилені запити, що дає змогу відстежити проходження запиту від клієнтського застосунку через серверний API до бази даних. Такий підхід спрощує пошук причини збою під час розгортання й експлуатації, оскільки повідомлення різних сервісів зіставляються за часом і дозволяють локалізувати проблемний етап обробки. Централізоване журналювання у стандартний потік також полегшує подальше підключення зовнішніх засобів збирання логів у разі переходу до промислового середовища.

Оновлення розгорнутої системи виконується без повного перерозгортання середовища. Після внесення змін до коду відповідний образ перезбирається командою `docker compose build`, після чого повторно створюється лише той контейнер, якого стосувалися зміни, — `frontend_app` або `backend_api`, тоді як контейнер бази даних і його том залишаються незмінними. Завдяки цьому

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

оновлення застосунку не впливає на збережені дані велосипедів і передач власності, а час простою обмежується коротким перезапуском одного сервісу. Такий підхід спрощує супровід системи та дозволяє послідовно впроваджувати нові функції, не порушуючи роботу вже наявних.

3.2. Структура інтерфейсу

Клієнтську частину вебсистеми реалізовано у вигляді Next.js-застосунку. Структура інтерфейсу базується на компонентному підході, що сприяє повторному використанню коду та спрощує супровід сторінок із різними сценаріями доступу.

Візуальна частина порталу побудована з використанням PrimeReact, PrimeIcons і Tailwind CSS. Завдяки цьому інтерфейс має узгоджену типографіку, таблиці, діалоги, кнопки, теги статусу та адаптивну сітку, яка коректно працює на різних розмірах екрана.

Оскільки вебсистема постійно взаємодіє із сервером, частину операцій реалізовано через server actions Next.js. Вони отримують access token від Auth0, виконують запит до backend API та після успішної зміни даних оновлюють відповідні сторінки через revalidatePath.

Загалом логічна структура вебсистеми поділяється на п’ять основних зон: публічна сторінка пошуку, особистий кабінет власника, сторінка велосипеда, панель поліції та адміністративна панель.

Під час першого входу на вебсистема користувач потрапляє на публічну частину (Рис. 3.1).

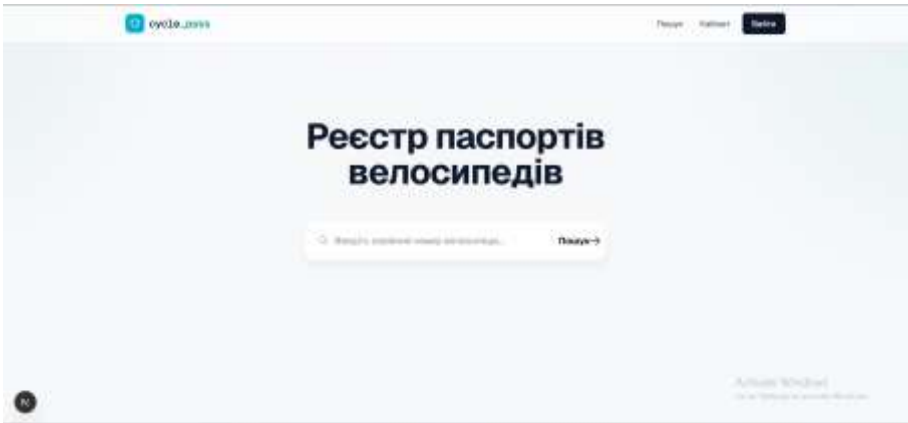


Рис. 3.1. Головна сторінка вебсистеми Вело-паспорт

На головній сторінці реалізовано основний сценарій взаємодії з неавторизованим користувачем, а саме пошук велосипеда за серійним номером. Форма містить одне поле та кнопку відправлення, що відповідає ситуації швидкої перевірки й не вимагає від користувача жодних додаткових даних, крім самого номера.

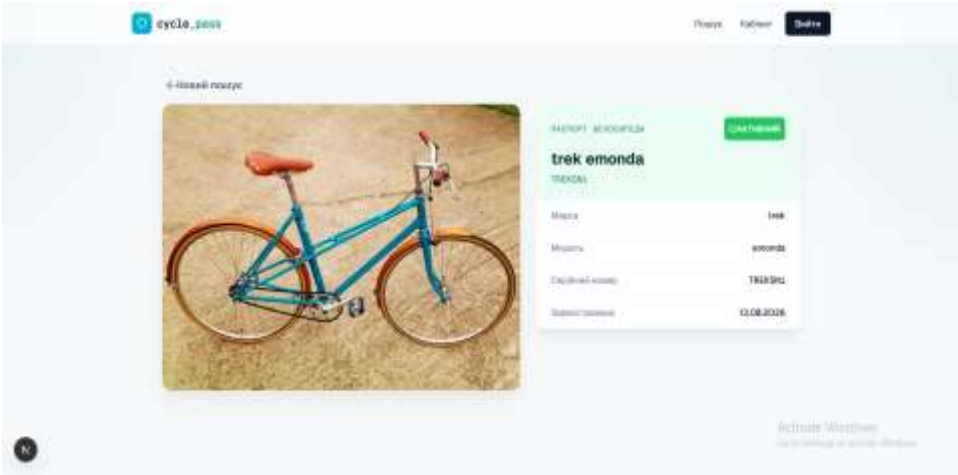


Рис. 3.2. Результат пошуку за серійним номером

Коли користувач вводить серійний номер, клієнтська частина нормалізує значення до верхнього реєстру та формує маршрут `/bikes/[serialNumber]`. На цій сторінці відбувається запит до публічного API, який повертає лише відкриті поля велосипеда (Рис. 3.2).

Особистий кабінет власника (Рис. 3.3) містить таблицю зареєстрованих велосипедів, статистичні блоки за статусами та кнопку додавання нового велосипеда. У таблиці відображаються серійний номер, виробник, модель, статус і дата реєстрації.

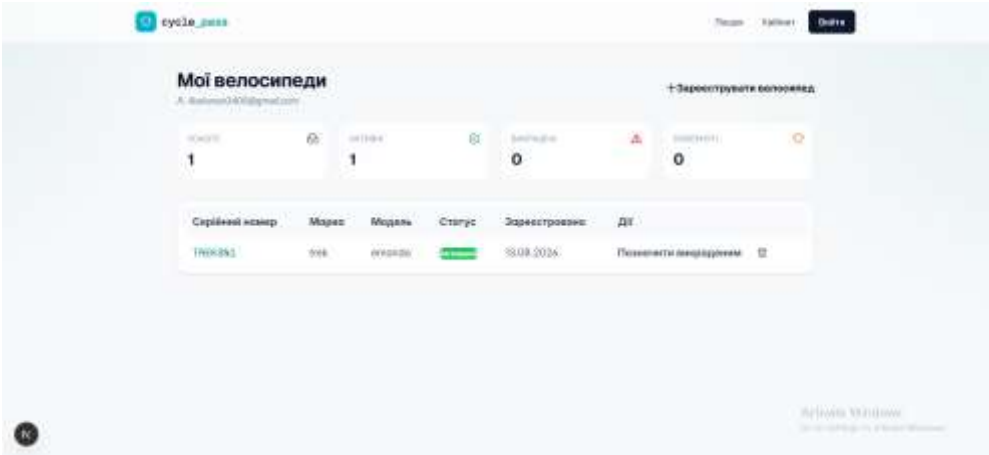


Рис. 3.3. Особистий кабінет власника велосипедів

На сторінці кабінету власник може створити новий цифровий паспорт через модальне вікно, де вводяться серійний номер, виробник і модель велосипеда (Рис. 3.4).

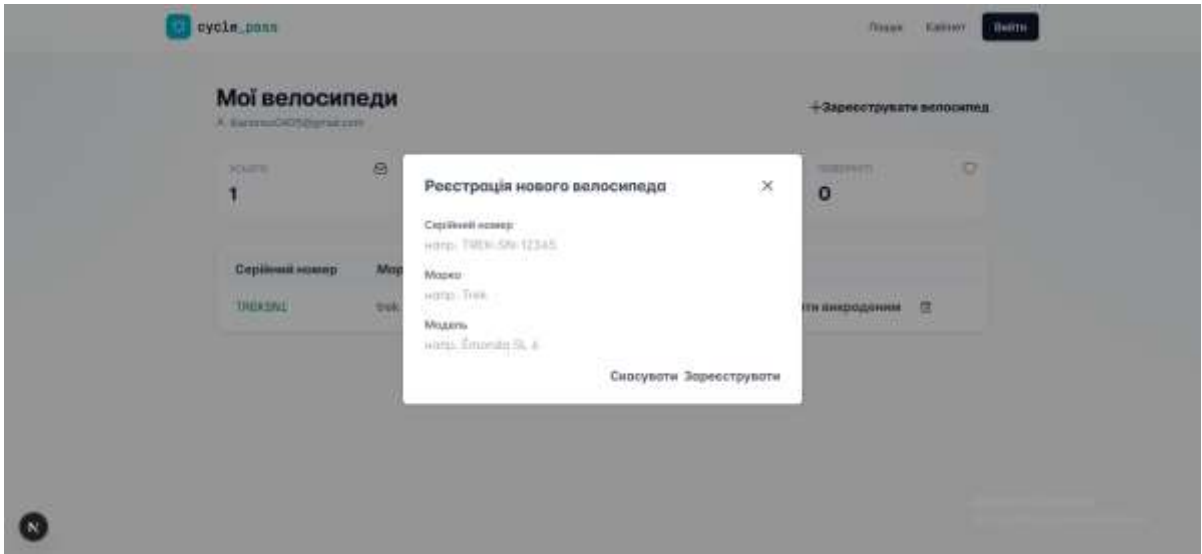


Рис. 3.4. Модальне вікно реєстрації велосипеда

Після створення запису велосипед з’являється в таблиці особистого кабінету. Власник може позначити велосипед як викрадений, відновлений або активний, а також перейти на сторінки своїх зареєстрованих велосипедів за натисканням на серійний номер

Сторінка цифрового паспорта велосипеда містить серійний номер, виробника, модель, поточний статус, дату реєстрації, власника, блок фотографій, історію статусів і дані підключених сервісів. (Рис. 3.5)

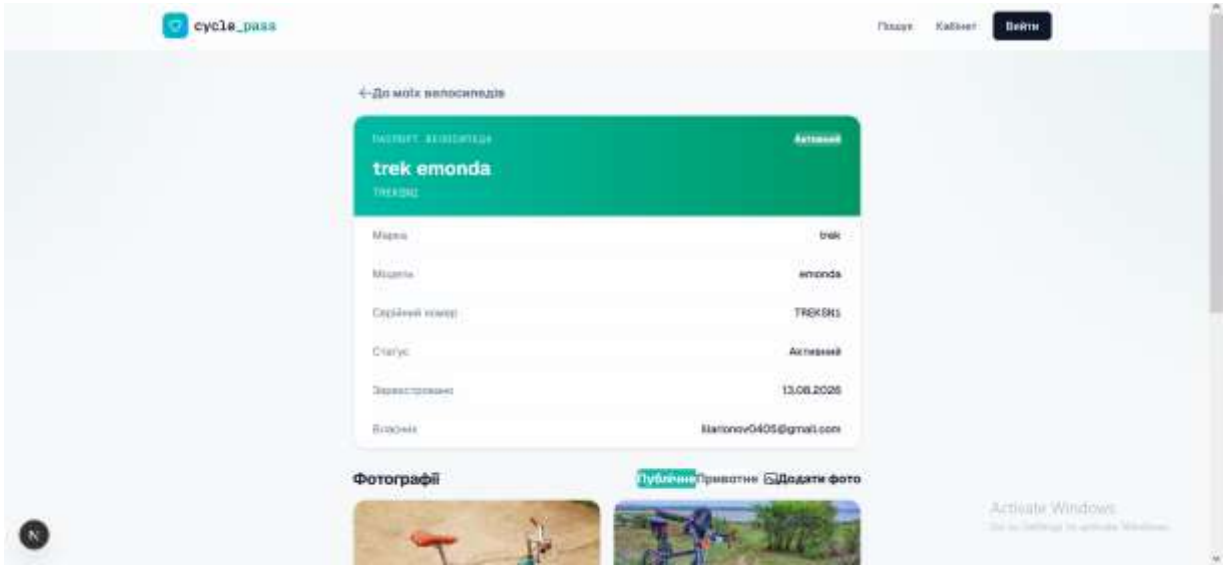


Рис. 3.5. Сторінка цифрового паспорта велосипеда

Набір доступних на цій сторінці дій залежить від ролі користувача: власникові доступне редагування виробника й моделі, зміна статусу, видалення запису та створення посилання на передачу, представникові поліції — лише позначення велосипеда як знайденого, а неавторизований користувач бачить лише довідкове представлення.

Для власника додатково доступний блок фотографій (Рис. 3.6). Він може обрати видимість public або private, після чого завантажити файл у форматі JPG, PNG або WebP розміром до 10 МБ.

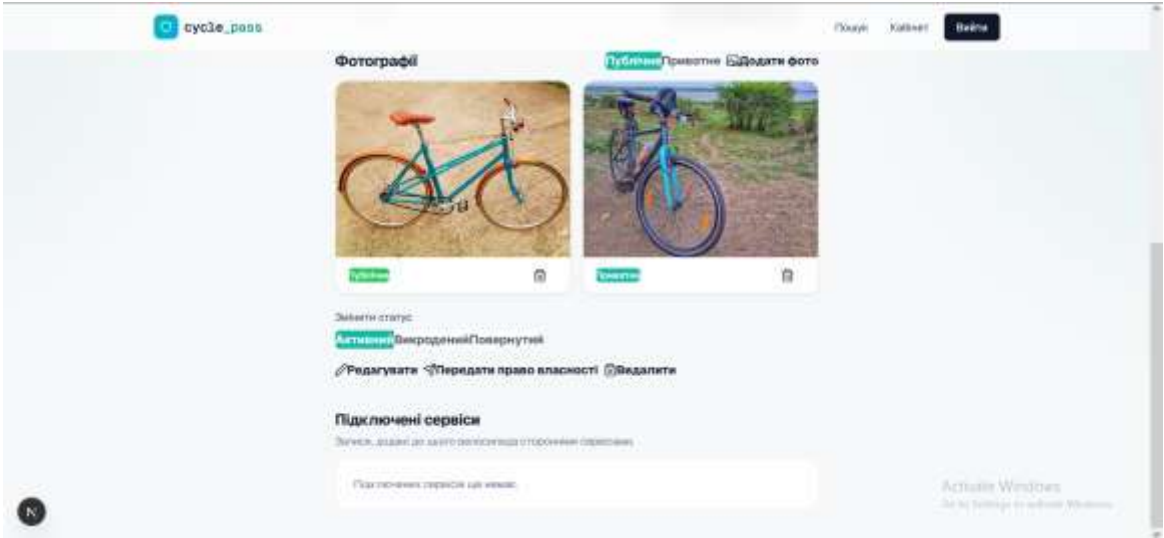


Рис. 3.6. Блок фотографій велосипеда

Передача власності реалізована через окреме модальне вікно (Рис. 3.7). Власник створює одноразове посилання, яке можна скопіювати або у вигляді QR-коду. Посилання діє 72 години і не може використовуватися повторно.

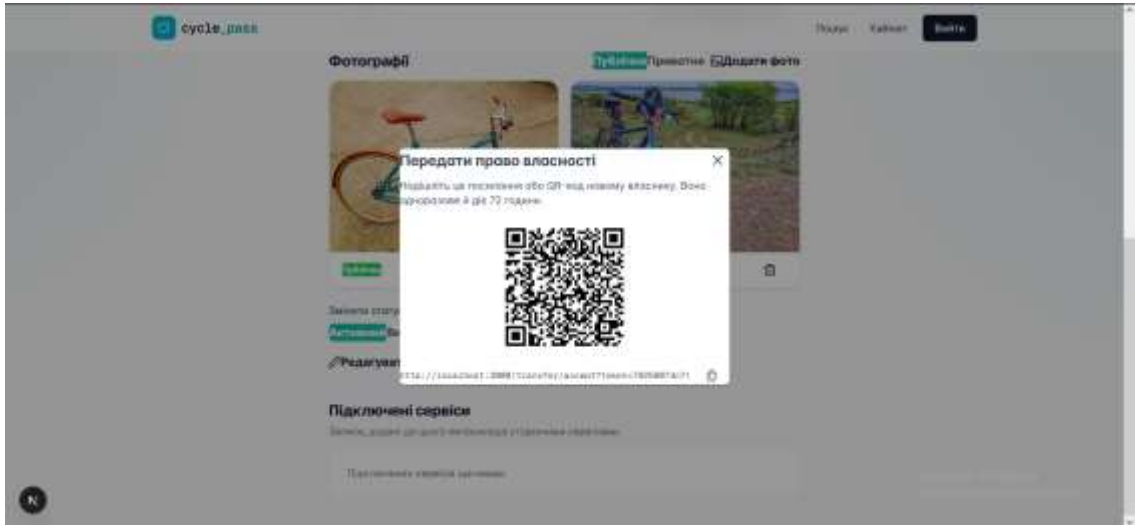


Рис. 3.7. Модальне вікно передачі власності

Коли новий користувач відкриває посилання на передачу, він переходить на сторінку прийняття (Рис. 3.8). Після автентифікації система перевіряє токен, змінює ownerEmail велосипеда та позначає передачу як використану.

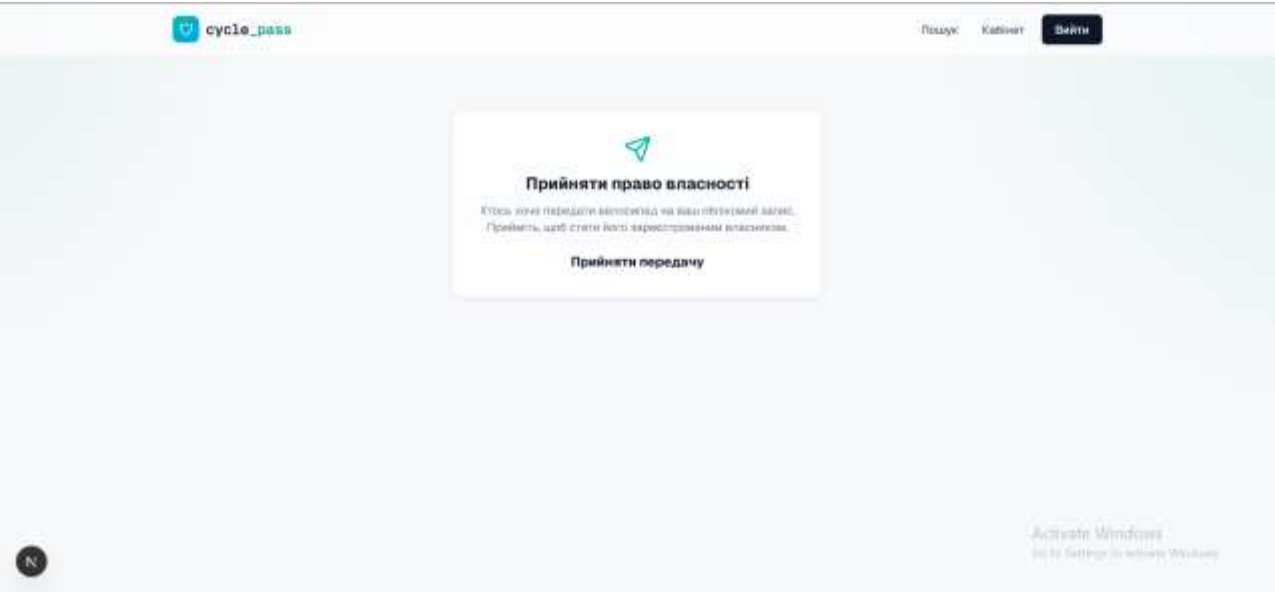


Рис. 3.8. Сторінка прийняття передачі велосипеда

Для зміни статусу велосипеда використано окремий елемент керування (Рис. 3.9). Власник, адміністратор або представник поліції можуть перевести велосипед між станами active, stolen і recovered, а система зберігає кожну зміну в історії.

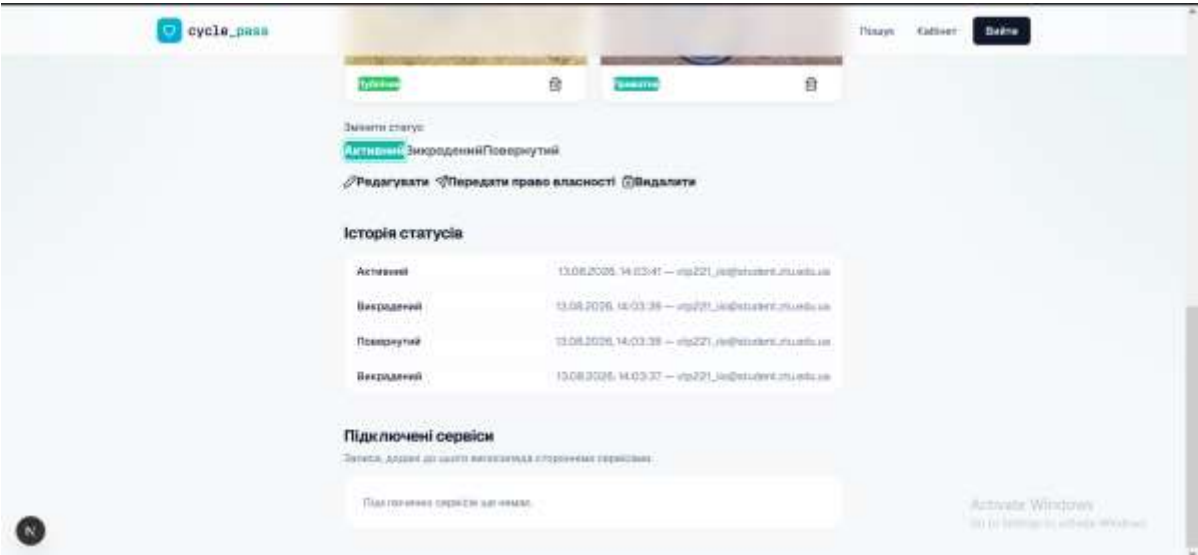


Рис. 3.9. Керування статусом велосипеда

До кожної зміни статусу додаються дата, час і користувач, який виконав дію. Це дозволяє відстежувати життєвий цикл цифрового паспорта та бачити, коли велосипед був позначений як викрадений або знайдений.

На сторінці паспорта велосипеда також відображаються дані підключених сервісів (Рис. 3.10). Якщо сторонній сервіс записав службову інформацію через машинний доступ, власник та адміністратор бачать відповідний простір даних.

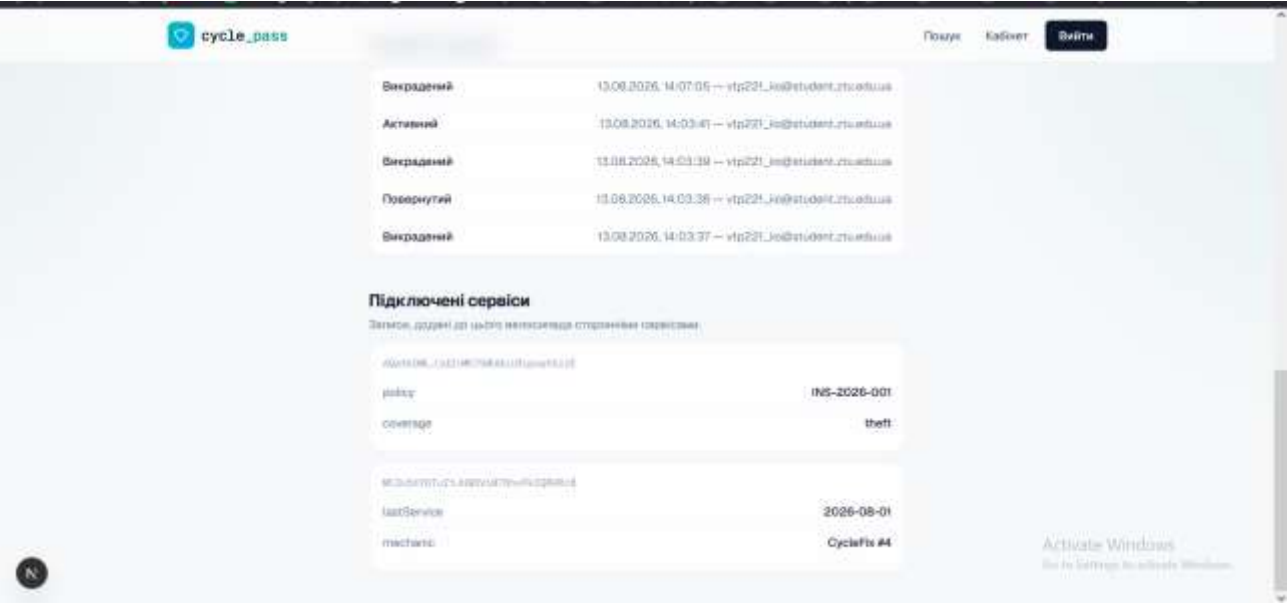


Рис. 3.10. Блок підключених сервісів

Представник поліції працює з окремою сторінкою, де показано лише велосипеди зі статусом stolen (Рис. 3.11). Для кожного запису відображаються серійний номер, виробник, модель, власник, дата реєстрації та дія для позначення велосипеда як recovered.

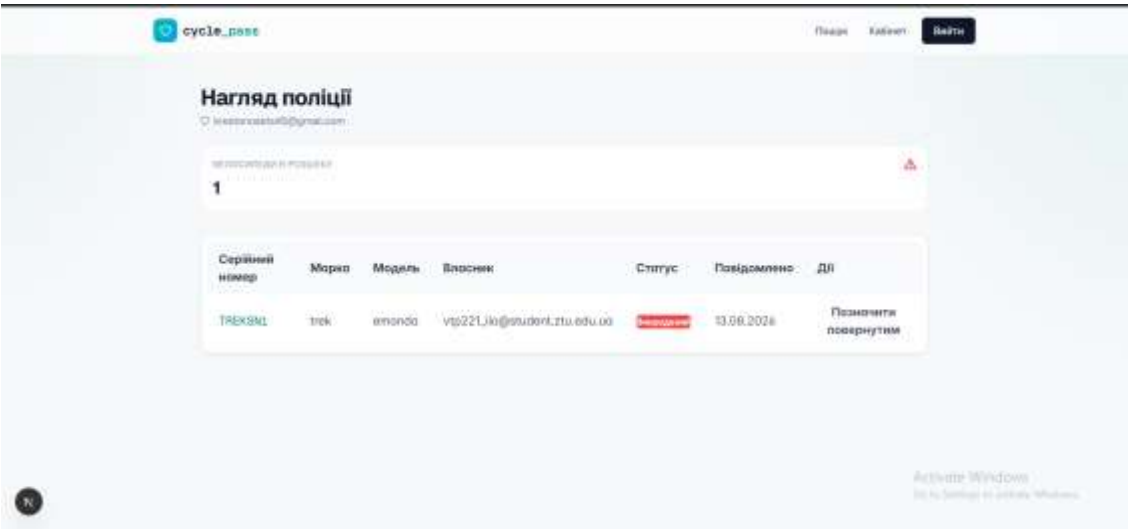


Рис. 3.11. Сторінка представника поліції

Другою захищеною сторінкою є адміністративна панель (Рис. 3.12). Вона показує повну кількість велосипедів у системі, розподіл за статусами, фільтри та таблицю для перегляду записів платформи.

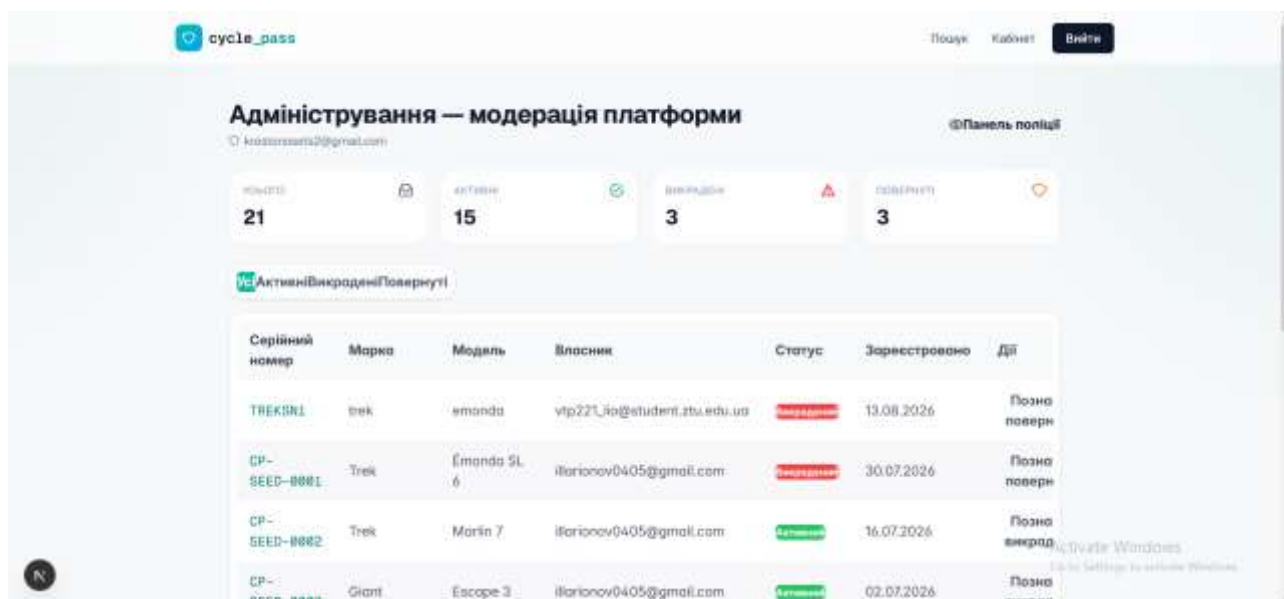


Рис. 3.12. Адміністративна панель

Єдиною відмінністю адміністративної панелі від кабінету власника є ширший огляд даних та можливість модерації шляхом видалення паспорта велосипеда. Адміністратор може переглядати записи різних користувачів, фільтрувати їх за статусом і переходити до деталей велосипеда (Рис. 3.13).

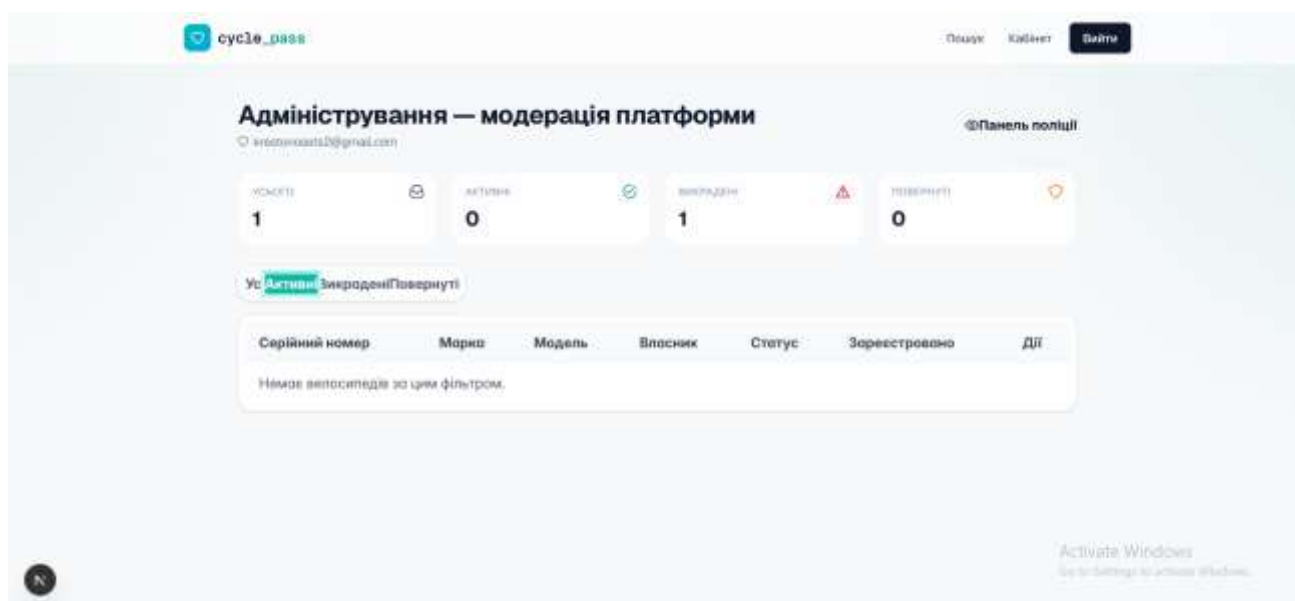


Рис. 3.13. Фільтрація велосипедів в адміністративній панелі

Обрану видимість показано позначкою на самому зображенні, тому власник візуально контролює, які знімки будуть доступні у публічній картці, а які залишаться видимими лише йому, представникові поліції та адміністратору.

[← Новий пошук](#)

 Цей велосипед знайдено як **вкрадений**. Не купуйте його. Зверніться до поліції та повідомте серійний номер, наведений нижче.

ПАСПОРТ ВЕЛОСИПЕДА

ДЛЯ ВІДРАДОНІЙ

Trek Émonda SL 6

CP-SEED-0001

Марка	Trek
Модель	Émonda SL 6
Серійний номер	CP-SEED-0001
Варіант оформлення	30.07.2026

[Відкрити повну картку](#) →

		Ілларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

У межах функціональності власника передбачено редагування базових полів велосипеда. Серійний номер залишається основним ідентифікатором, а виробник і модель можуть уточнюватися після створення запису (Рис. 3.16).

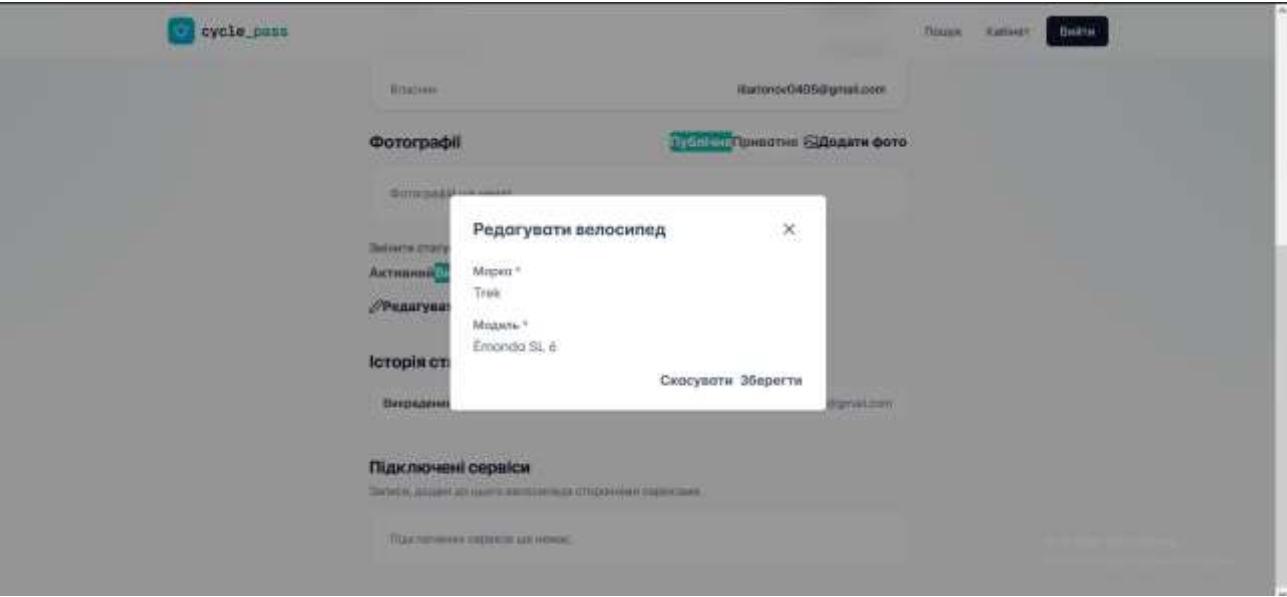


Рис. 3.16. Форма редагування даних велосипеда

Як і у випадку з іншими захищеними сторінками, доступ до особистого кабінету виконується через Auth0. Якщо користувач не автентифікований, middleware перенаправляє його до сценарію входу (Рис. 3.17).

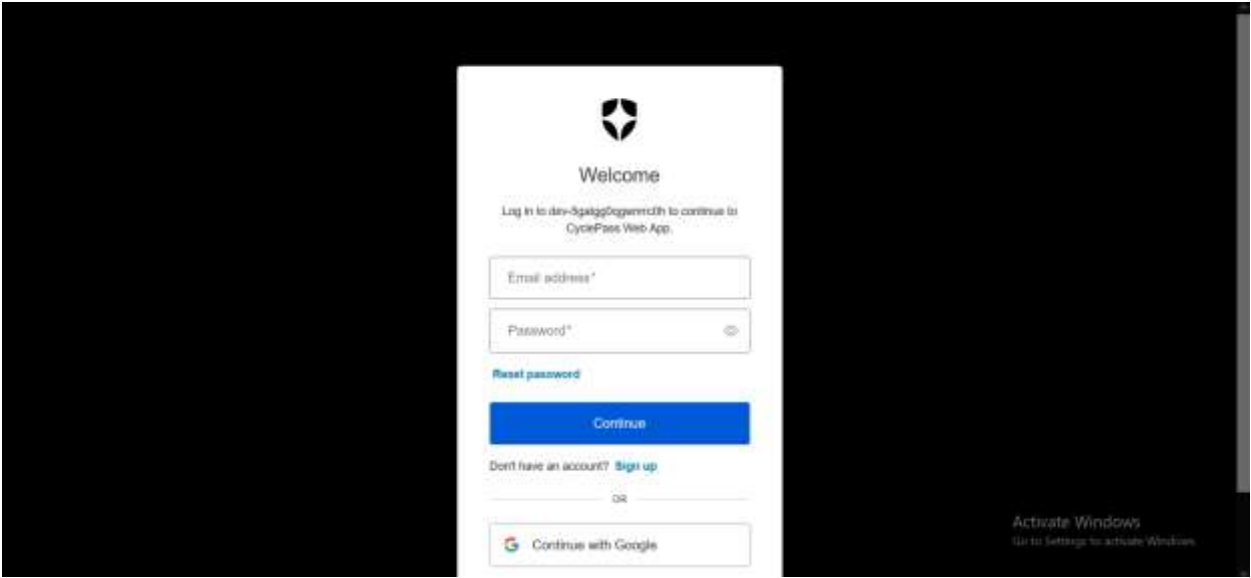


Рис. 3.17. Сторінка входу через Auth0

Останнім важливим елементом інтерфейсу є повідомлення про успішні та помилкові дії. Для цього використано Toast-компоненти, які інформують

користувача про результат реєстрації велосипеда, зміни статусу, завантаження фотографії або створення посилання на передачу (Рис. 3.18).

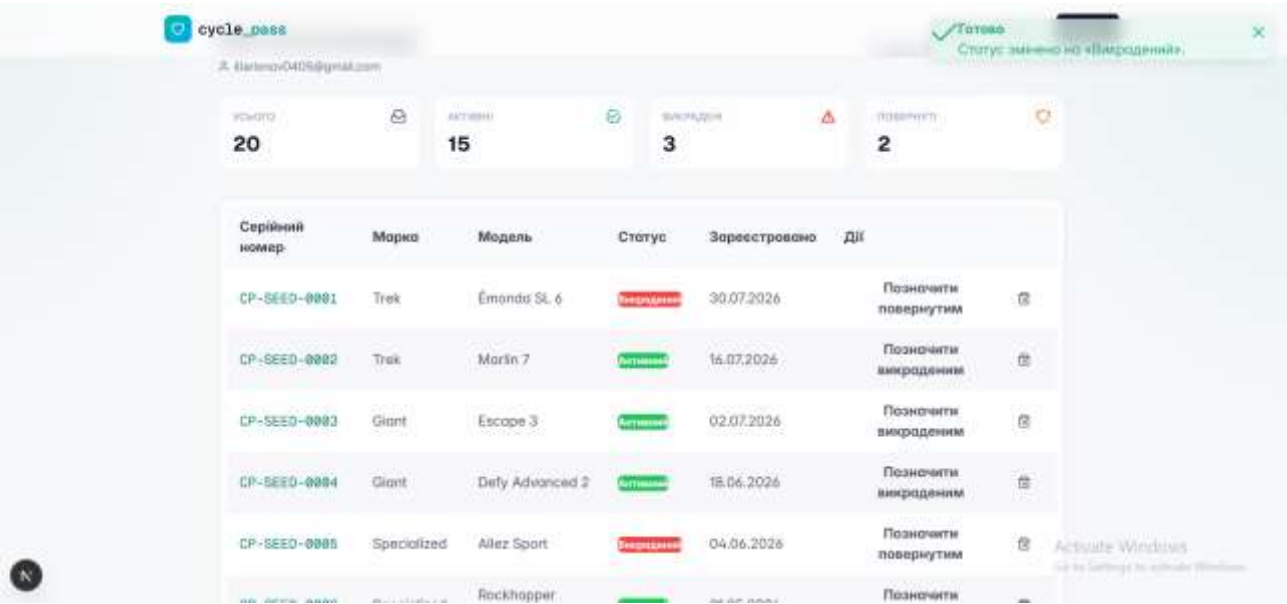


Рис. 3.18. Сповіщення про результат операції

Таблиця платформи підтримує пошук за серійним номером, що дає змогу знайти потрібний запис серед велосипедів різних власників без переходу до окремих кабінетів.

На відміну від власника, адміністратор може змінити статус (Рис. 3.19) будь-якого запису платформи, а не лише власного. Це потрібно в ситуаціях, коли власник втратив доступ до свого облікового запису або звернення про викрадення надійшло іншим каналом, і кожна така дія фіксується в історії із зазначенням адміністратора.

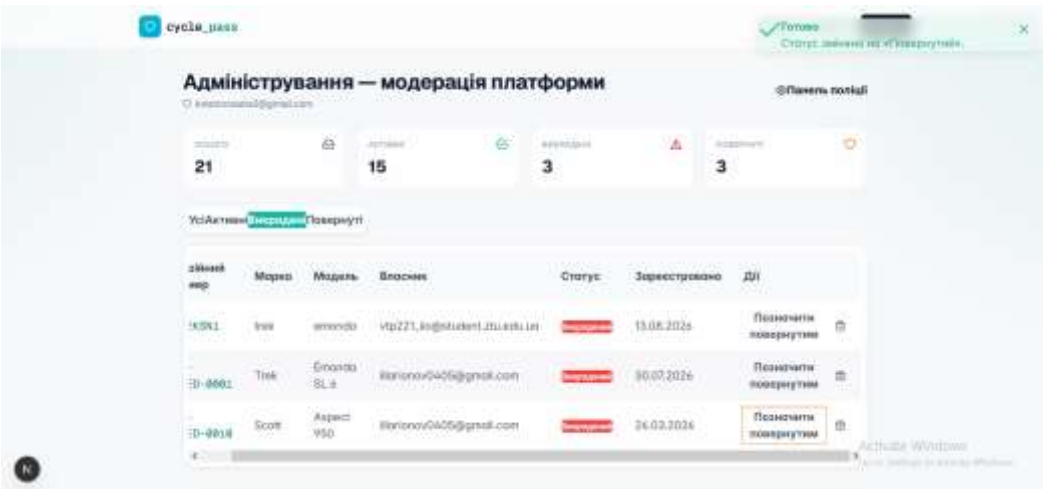


Рис. 3.19. Зміна статусу адміністратором

Видалення запису вимагає підтвердження у діалоговому вікні, що зменшує ризик видалення цифрового паспорта користувача. (Рис. 3.20)

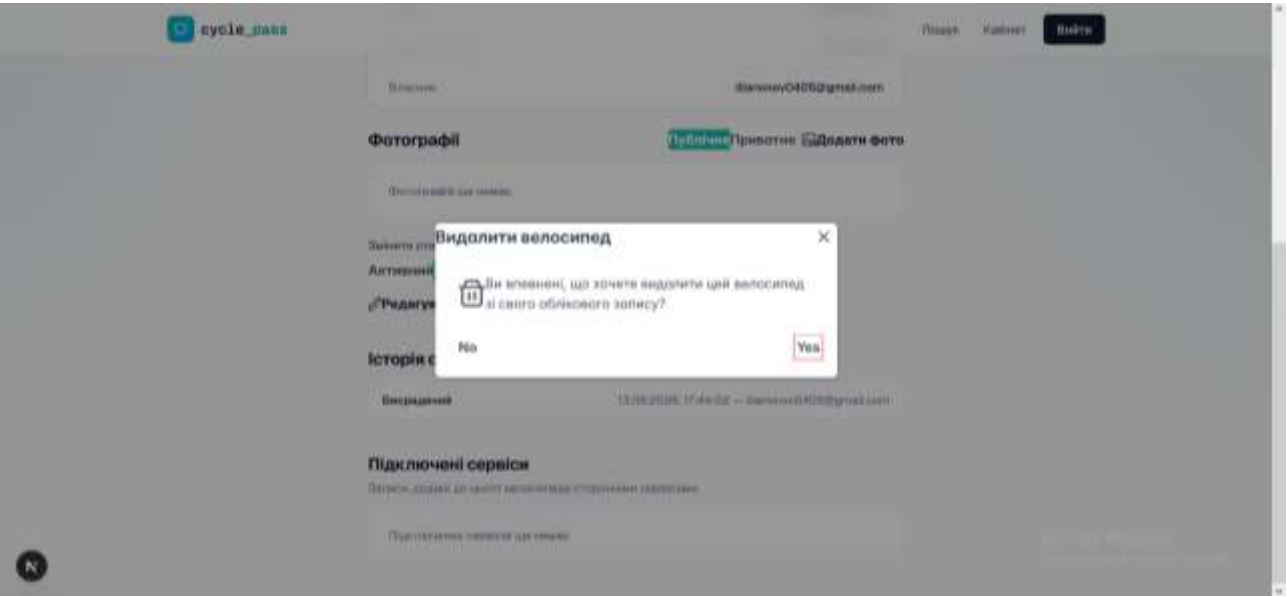


Рис. 3.20. Підтвердження видалення велосипеда

Окремою частиною роботи системи є публічний перегляд результату пошуку. Якщо велосипед не знайдено, користувач отримує зрозуміле повідомлення без розкриття внутрішньої інформації про базу даних.

Розділ інтерфейсу з підключеними сервісами (Рис. 3.21) спеціально спроектований для прозорого відображення даних, які зовнішні системи записали у межах власного простору. Це може бути службова інформація страхової компанії, сервісного центру або іншого партнера.

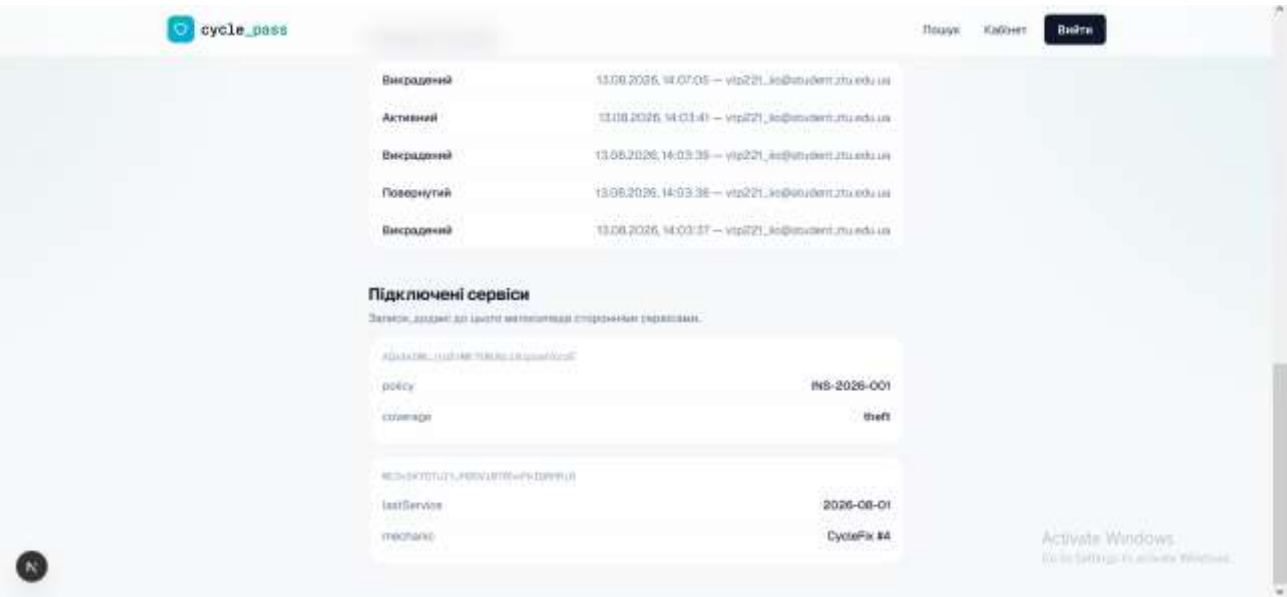


Рис. 3.21. Дані зовнішнього сервісу у картці велосипеда

3.3. Тестування роботи вебсистеми

Для перевірки працездатності реєстру Вело-паспорт було проведено комплексне мануальне тестування серверної логіки та основних користувацьких сценаріїв. Основний акцент зроблено на контролі доступу, коректності зміни статусів, роботі з фотографіями, передачі власності та фільтрації приватних даних.

Тестування виконувалося за поєднанням двох підходів: ручної перевірки основних користувацьких сценаріїв через інтерфейс і автоматизованого тестування серверної логіки. Для кожного сценарію заздалегідь визначалися очікуваний результат і умови успішності — зокрема коди HTTP-відповідей, склад повернених даних та зміни у базі. Такий підхід дав змогу перевірити не лише коректну поведінку системи за штатних умов, а і її реакцію на помилкові та несанкціоновані дії, що є критичним для системи з розмежуванням доступу.

Крім перевірки основного функціоналу, було проведено тестування валідації форм [7] і поведінки системи при спробах введення некоректних або неповних даних.

Тестування розпочато з перевірки захищених маршрутів API. Запити без токена доступу повинні відхилятися зі статусом 401, що підтверджує обов’язковість автентифікації для приватних операцій (Рис. 3.22).



```
PS C:\Users\user\Desktop\cycle_pass> curl.exe http://localhost:5000/bikes
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 43
ETag: W/"2b-dkn3rtgyn0j0603Wrtt4p4ng"
Date: Thu, 11 Aug 2020 11:30:22 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Unauthorized","statusCode":401}
PS C:\Users\user\Desktop\cycle_pass>
```

Рис. 3.22. Перевірка захищеного маршруту без токена

Проводилося тестування операцій власника. Власник може оновити власний велосипед, тоді як інший користувач отримує помилку 403. Це підтверджує коректну роботу BikeOwnershipGuard і захист запису від несанкціонованих змін (Рис. 3.23).

```

PASS test/bikes.e2e-spec.ts (5.732 s)
Bikes (e2e)
  auth
    o skipped rejects requests with no token (401)
  update
    ✓ rejects an update with no token (401) (141 ms)
    ✓ lets the owner update, blocks another user (403), admin bypasses ownership (78 ms)
    ✓ rejects invalid or unknown fields (400) (40 ms)
  )
    ✓ rejects a create with missing fields (400) (10 ms)
    ✓ returns 404 for an unknown bike (11 ms)
  malformed ids
    o skipped returns 400, not 500, for an id that i

```

Рис. 3.23. Перевірка доступу до оновлення велосипеда

Перевірялася зміна статусу велосипеда для різних ролей. Власник, представник поліції та адміністратор можуть змінювати статус, а сторонній користувач без відповідної ролі отримує заборону доступу. Додатково перевірено, що обхід перевірки власності не поширюється на видалення велосипеда (Рис. 3.24).

```

status + role bypass
  ✓ lets owner, police and admin change status (292 ms)
  ✓ blocks a non-owner with no role (403) (52 ms)
  ✓ does not let the status bypass leak to delete (police 403) (48 ms)
  ✓ records status history (actor + order), hidden from other users (161 ms)

```

Рис. 3.24. Тестування зміни статусу велосипеда

Також було проведено тестування завантаження фотографій. Інший користувач не може підписати завантаження або додати фотографію до чужого велосипеда, а власник отримує коректний підпис Cloudinary і може зберегти метадані фотографії (Рис. 3.25).

```

photos
  ✓ rejects adding a photo with no token (401) (134 ms)
  ✓ blocks a non-owner from signing an upload or adding a photo (403) (89 ms)
  ✓ does not let police or admin add photos to a bike they do not own (403) (82 ms)
  ✓ lets the owner attach/delete Cloudinary photos and filters private photos (174 ms)
data isolation
  o skipped shows a machine only its own namespace

```

Рис. 3.25. Тестування додавання фотографії

Проведено перевірку фільтрації приватних даних. Власник бачить усі фотографії, історію статусів і дані сторонніх сервісів, тоді як сторонній користувач отримує лише публічні фотографії та не бачить email власника (Рис. 3.26).

```
data isolation
  ✓ shows a machine only its own namespace (64 ms)
  ✓ shows the owner every namespace plus the owner email (13 ms)
  ✓ shows admin every namespace (privileged) (9 ms)
  ✓ hides third-party data and owner email from other users (14 ms)
```

Рис. 3.26. Перевірка фільтрації приватних даних

Також було перевірено процес передачі власності. Власник може створити одноразовий токен, викрадений велосипед не може бути переданий, новий користувач після прийняття посилання стає власником, а повторне використання токена завершується помилкою (Рис. 3.27).

```
ownership transfer
  ✓ owner creates a transfer; others cannot (403) (102 ms)
  ✓ cannot transfer a stolen bike (400) (33 ms)
  ✓ recipient accepts and becomes owner; token is single-use (52 ms)
  ✓ rejects a bogus token (404) (9 ms)
  ✓ rejects a machine accepting a transfer (403) (24 ms)

Test Suites: 1 skipped, 1 passed, 1 of 2 total
Tests:       28 skipped, 5 passed, 33 total
Snapshots:   0 total
Time:        3.337 s
Ran all test suites with tests matching "ownership transfer".
```

Рис. 3.27. Тестування передачі власності

Окрім ручної перевірки, реалізовано набір автоматизованих наскрізних тестів (e2e) на основі Jest, які виконуються в ізольованому середовищі з використанням mongodb-memory-server замість зовнішнього екземпляра бази даних. Це дає змогу запускати повний цикл перевірок багаторазово й відтворювано, не впливаючи на реальні дані. Автоматизовані тести охоплюють автентифікацію, перевірку власності, зміну статусів, ізоляцію даних машинних клієнтів, фільтрацію приватних полів, завантаження фотографій та передачу

власності за одноразовим токеном, повторюючи в автоматичному режимі ті самі сценарії, що перевірялися вручну.

За результатами проведеного тестування підтверджено, що всі основні механізми вебсистеми працюють відповідно до проєктних вимог. Контроль доступу коректно розмежовує права різних категорій користувачів, валідація відхиляє некоректні дані, фільтрація приховує приватну інформацію від неавторизованих запитів, а передача власності відбувається безпечно й лише один раз для кожного токена. Виявлені під час тестування зауваження було усунено, що дало змогу переконатися у стабільності та надійності роботи системи перед її використанням.

Висновки до третього розділу

У третьому розділі було розглянуто роботу з інтерфейсом користувача, наведено опис розгортання та описано перевірку основних сценаріїв вебсистеми Вело-паспорт.

У першому підрозділі детально описано використану архітектуру розгортання, взаємодію frontend_app, backend_api, MongoDB, Auth0 і Cloudinary.

У другому підрозділі було продемонстровано основні функціональні можливості вебсистеми, зокрема публічний пошук, особистий кабінет власника, сторінку цифрового паспорта, передачу власності, панель поліції та адміністративний перегляд.

У третьому підрозділі описано процес тестування вебсистеми, перевірено роботу контролю доступу, валідації, фільтрації приватних даних, завантаження фотографій, зміни статусів і одноразової передачі власності.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

ВИСНОВОК

Під час виконання дипломної роботи було спроектовано, реалізовано та протестовано цифровий паспорт велосипеда Вело-паспорт. У межах першого розділу дослідження було проаналізовано проблематику розрізненого та фрагментованого обліку велосипедів, що суттєво ускладнює підтвердження права власності та перевірку статусу транспортного засобу.

Детальний огляд наявних аналогів, таких як Bike Index та Project 529, дозволив визначити їхні функціональні обмеження та обґрунтувати переваги розробленого рішення, зокрема підтримку передачі права власності за одноразовими токенами або QR-кодами й ізольоване зберігання даних сторонніх сервісів. Крім того, під час дослідження було обґрунтовано вибір архітектури клієнт-сервер та технологічного стеку, що охоплює мову TypeScript, фреймворк NestJS для серверної частини, Next.js, React, PrimeReact і Tailwind CSS для клієнтської частини, СУБД MongoDB з бібліотекою Mongoose, а також сервіс автентифікації Auth0.

Другий розділ було присвячено проектуванню та безпосередній програмній реалізації вебсистеми. В рамках цих завдань розроблено рольову модель доступу з чітким визначенням варіантів використання для власника, адміністратора, представника поліції, публічного користувача та машинного клієнта (M2M).

Також спроектовано документно-орієнтовану структуру бази даних MongoDB, яка включає сутність Bike з вкладеними масивами фотографій, історією статусів statusHistory та карти third_party_data, а також колекцію Transfer для обробки запитів на зміну власності. Задля забезпечення високого рівня захисту впроваджено комплексні механізми безпеки API, серед яких перевірка JWT-токенів через Auth0 JWKS, власні контролери доступу BikeOwnershipGuard, RolesGuard, M2mGuard, ScopesGuard із дозволом write:third-party-data та BikeDataFilterInterceptor для автоматичного приховування приватних даних у відповідях.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Окремо реалізовано безпечний алгоритм передачі власності за допомогою SHA-256 хешування одноразових токенів із обмеженим строком дії тривалістю 72 години. У третьому розділі виконано роботи щодо розгортання, розробки інтерфейсу та тестування системи.

Організовано розгортання інфраструктури за допомогою Docker Compose, що об'єднує контейнери бази даних MongoDB, backend API та frontend-застосунку.

Для роботи з медіафайлами інтегровано хмарний сервіс Cloudinary, який забезпечує підписане завантаження фотографій із можливістю налаштування видимості у публічному або приватному режимі. Вебінтерфейс реалізовано на компонентній основі з підтримкою Next.js Server Actions, модальних вікон, Toast-сповіщень та кольорової індикації статусів велосипеда, зокрема active, stolen та recovered.

На завершальному етапі проведено комплексне автоматизоване e2e-тестування з використанням Jest і mongodb-memory-server, а також ручне тестування сценаріїв автентифікації, перевірки власності, валідації DTO, публічного пошуку та фільтрації приватних даних. Розроблена вебсистема Вело-паспорт є повністю працездатним і масштабованим програмним комплексом. Вона успішно вирішує проблему впорядкування ідентифікаційних даних велосипедів, спрощує процедуру підтвердження належності майна, підвищує оперативність реагування на випадки викрадення та забезпечує безпечне середовище для взаємодії власників, правоохоронних органів, адміністраторів і зовнішніх партнерських сервісів.

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. A survey on bike theft in Europe. *Nextscape Inc. (AlterLock)*. URL: <https://alterlock.net/en/a-survey-on-bike-theft-in-europe> (дата звернення: 12.08.2026)
2. Auth0 Docs. *Okta, Inc.* URL: <https://auth0.com/docs> (дата звернення: 12.08.2026)
3. Benelux-NRW position paper on bicycle theft prevention. *Benelux Union*. URL: <https://www.benelux.int/wp-content/uploads/2025/11/Benelux-NRW-position-paper-on-bicycle-theft-prevention.pdf> (дата звернення: 12.08.2026)
4. Bicycle industry in Europe: statistics report. *Statista*. URL: <https://www.statista.com/study/37851/bicycle-industry-in-europe-statista-dossier> (дата звернення: 12.08.2026)
5. Bike Index. *Bicycle registration service*. URL: <https://bikeindex.org/> (дата звернення: 12.08.2026)
6. Bike theft across Europe and securing better bike parking. *European Cyclists' Federation*. URL: <https://www.ecf.com/en/news/bike-theft-across-europe-and-securing-better-bike-parking/> (дата звернення: 12.08.2026)
7. Документація class-validator. *GitHub Repository*. URL: <https://github.com/typestack/class-validator> (дата звернення: 12.08.2026)
8. Cloudinary Upload API Reference. *Cloudinary Documentation*. URL: https://cloudinary.com/documentation/image_upload_api_reference (дата звернення: 12.08.2026)
9. Data Modeling. *MongoDB Manual*. URL: <https://www.mongodb.com/docs/manual/data-modeling/> (дата звернення: 12.08.2026)
10. Документація Docker Compose. *Docker Docs*. URL: <https://docs.docker.com/compose/> (дата звернення: 12.08.2026)
11. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : PhD dissertation. – University of California, Irvine, 2000.

		Ілларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

12. Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749. *IETF Datatracker*. URL: <https://www.rfc-editor.org/rfc/rfc6749> (дата звернення: 12.08.2026)
13. Огляд HTTP. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview> (дата звернення: 12.08.2026)
14. Документація Jest. *Jest – Delightful JavaScript Testing*. URL: <https://jestjs.io/docs/getting-started> (дата звернення: 12.08.2026)
15. Jones M. JSON Web Key (JWK). RFC 7517. *IETF Datatracker*. URL: <https://www.rfc-editor.org/rfc/rfc7517> (дата звернення: 12.08.2026)
16. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. *IETF Datatracker*. URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата звернення: 12.08.2026)
17. MongoDB Manual. *MongoDB Documentation*. URL: <https://www.mongodb.com/docs/manual/> (дата звернення: 12.08.2026)
18. Документація mongodb-memory-server. *Typegoose*. URL: <https://typegoose.github.io/mongodb-memory-server/> (дата звернення: 12.08.2026)
19. Документація Mongoose. *Mongoose ODM v8*. URL: <https://mongoosejs.com/docs/> (дата звернення: 12.08.2026)
20. Документація NestJS. *NestJS – A progressive Node.js framework*. URL: <https://docs.nestjs.com/> (дата звернення: 12.08.2026)
21. Документація Next.js. *Next.js by Vercel – The React Framework*. URL: <https://nextjs.org/docs> (дата звернення: 12.08.2026)
22. Node.js – Introduction to Node.js. *Node.js – Run JavaScript Everywhere*. URL: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> (дата звернення: 12.08.2026)
23. Документація PrimeReact. *PrimeReact – React UI Component Library*. URL: <https://primereact.org/> (дата звернення: 12.08.2026)
24. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI. *Законодавство України*. URL: <https://zakon.rada.gov.ua/laws/show/2297-17> (дата звернення: 12.08.2026)

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

25. Project 529. *Bicycle registration and recovery platform*. URL: <https://project529.com/> (дата звернення: 12.08.2026)
26. React Documentation. *React – The library for web and native user interfaces*. URL: <https://react.dev/> (дата звернення: 12.08.2026)
27. Role Based Access Control (RBAC). *NIST Computer Security Resource Center*. URL: <https://csrc.nist.gov/projects/role-based-access-control> (дата звернення: 12.08.2026)
28. Secure Hash Standard (SHS). FIPS PUB 180-4. *NIST Computer Security Resource Center*. URL: <https://csrc.nist.gov/pubs/fips/180-4/upd1/final> (дата звернення: 12.08.2026)
29. Tailwind CSS Documentation. *Tailwind CSS – Rapidly build modern websites*. URL: <https://tailwindcss.com/docs> (дата звернення: 12.08.2026)
30. TypeScript Documentation. *TypeScript – JavaScript With Syntax For Types*. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 12.08.2026)

		Глларіонов І.О.			ІПЗ.КР.Б-121.26 - ПЗ	Арк.
		Власенко О.В.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Реалізація схеми велосипеда

```
export enum BikeStatus { ACTIVE = 'active', STOLEN = 'stolen', RECOVERED =  
'recovered' }
```

```
@Schema({ timestamps: true })
```

```
export class Bike {
```

```
  @Prop({ required: true, unique: true, index: true, uppercase: true, trim: true })  
  serialNumber!: string;
```

```
  @Prop({ required: true, trim: true })  
  make!: string;
```

```
  @Prop({ required: true, trim: true })  
  model!: string;
```

```
  @Prop({ required: true, index: true, lowercase: true, trim: true })  
  ownerEmail!: string;
```

```
  @Prop({ type: String, enum: BikeStatus, default: BikeStatus.ACTIVE, index: true  
  })  
  status!: BikeStatus;  
}
```

Реалізація перевірки власності велосипеда

```
async canActivate(context: ExecutionContext): Promise<boolean> {  
  const request = context.switchToHttp().getRequest();  
  const user = request.user;  
  const bikeId = request.params.id;  
  
  const bypassRoles =  
this.reflector.getAllAndOverride<Role[]>(OWNERSHIP_BYPASS_KEY,  
[context.getHandler(), context.getClass()]) ?? [];  
  if (bypassRoles.some((role) => (user.roles ?? []).includes(role))) return true;  
  
  const bike = await this.bikesService.findOne(bikeId);  
  if (bike.ownerEmail !== user.email) {  
    throw new ForbiddenException('Access denied. You do not own this bicycle.');  }  
  return true;  
}
```

Реалізація передачі права власності

```
async createTransfer(bikeId: string, fromEmail: string) {  
  const bike = await this.bikeModel.findById(bikeId).exec();  
  if (bike.status === BikeStatus.STOLEN) {  
    throw new BadRequestException('A bike reported stolen cannot be transferred.');  }  
  await this.transferModel.deleteMany({ bikeId, usedAt: null }).exec();  
  const token = randomBytes(32).toString('hex');  
  await this.transferModel.create({  
    bikeId,  
    tokenHash: this.hashToken(token),  
    fromEmail,  
    expiresAt: new Date(Date.now() + 72 * 60 * 60 * 1000),  
  });  
  return { token, expiresAt };  
}
```

Діаграма розгортання вебсистеми Вело-паспорт

