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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка системи Chronicon для управління часом та завданнями для компаній та фрілансерів

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Хітрих Б.Б.
(підпис) (прізвище та ініціали)

Керівник _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

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

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Хітрих Богдан Богданович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи Chronicon для управління часом та завданнями для компаній та фрілансерів.

Керівник роботи Небесний Руслан Михайлович, доктор філософії, доцент кафедри КН.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 28 червня 2025р.

3. Вихідні дані до роботи Інтернет джерела про технології розробки сайту

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ. Розділ 1. Аналіз предметної області та постановка завдання. 1.1 Аналіз предметної області системи Chronicon. 1.2 Формування вимог. 1.3 Опис акторів та їх використання системи Chronicon . 1.4 Висновок до першого розділу. Розділ 2. Проектування системи Chronicon. 2.1 Вступ і вибір інструментів для створення системи Chronicon. 2.2 Дизайн користувацького інтерфейсу. 2.3 Проектування баз даних. 2.4 Структура реляційної бази даних. 2.5 Проектування архітектури системи 2.6 Висновок до другого розділу. Розділ 3. Практична частина . 3.1 Розробка серверної частини. 3.2 Розробка клієнтської частини. 3.3 Тестування системи . 3.4 Розгортання на хостингу. 3.5 Висновок до третього розділу. Розділ 4. Безпека життєдіяльності,основи охорони праці. 4.1 Ергономічні проблеми безпеки життєдіяльності. 4.2 Організація охорони праці на виробництві. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

Слайди:1 Тема. 2 Актуальність,об'єкт,предмет дослідження.3 Мета роботи. 4 Чому управління часом і завданнями важливе. 5 Поточні інструменти та їх недоліки . 6 Функціональні вимоги. 7 Нефункціональні вимоги. 8 Основні актори системи. 9 Проектування архітектури. 10 Основні сторінки. 11 Бази даних. 12 Допоміжні сервіси. 13 Реалізація серверної частини. 14 Реалізація клієнтської частини. 15 Основні сценарії тестування. 16 Розгортання на хостингу. 17 Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	*Виконано*
2.	Підбір джерел про веб-технології для розробки інтерактивних веб-сайтів	31.01.2025-03.02.2025	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2025-06.02.2025	*Виконано*
4.	Виконання дослідження щодо існуючих інструментів та технологій для розробки веб-сайтів	07.02.2025-09.02.2025	*Виконано*
5.	Розроблення структури та функціоналу системи “Chronicon ”	20.05.2025-01.06.2025	*Виконано*
6.	Оформлення розділу «Аналіз предметної області та постановка завдання»	02.06.2025-05.06.2025	*Виконано*
7.	Оформлення розділу «Проектування системи Chronicon»	06.06.2025-08.06.2025	*Виконано*
8.	Оформлення розділу «Практична частина»	09.06.2025-11.06.2025	*Виконано*
9.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2025-13.06.2025	*Виконано*
10.	Виконання завдання до підрозділу «Основи охорони праці»	14.06.2025-15.06.2025	*Виконано*
11.	Оформлення кваліфікаційної роботи	16.06.2025-17.06.2025	*Виконано*
12.	Нормоконтроль	18.06.2025-19.06.2025	*Виконано*
13.	Перевірка на плагіат	20.06.2025	*Виконано*
14.	Попередній захист кваліфікаційної роботи	21.06.2025	*Виконано*
15.	Захист кваліфікаційної роботи	29.06.2025	

Студент _____ Хітрих Б.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка системи Chronicon для управління часом та завданнями для компаній та фрілансерів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Хітрих Богдан Богданович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-43 // Тернопіль, 2025 // С. 55, рис. – 18, табл. – 9, додат. – 4 , бібліогр. – 41.

Ключові слова: менеджмент часу, менеджмент завдань, фриланс, компанії, Blazor, .NET.

Кваліфікаційна робота присвячена розробці системі Chronicon для управління часом та завданнями для компаній та фрілансерів.

У першому розділі описується предметна область і визначено функціональні та нефункціональні вимоги до системи а також її акторів.

У другому розділі спроектовано архітектуру веб-застосунку, структуру баз даних та вибір технологічного стека (.NET, Blazor WebAssembly, SQL Server, MongoDB тощо).

У третьому розділі описано реалізацію ключових модулів: авторизації та ролей, управління компаніями та фрілансерами, створення й трекінгу завдань, а також проведено тестування функціоналу й інтерфейсу. Об'єкт дослідження – система Chronicon; предмет дослідження – методи й засоби інтеграції управління завданнями та обліку часу в єдиній системі.

ANNOTATION

Development of the Chronicon System for Time and Task Management for Companies and Freelancers // Qualification work of the educational level “Bachelor” // Khitrykh Bohdan Bohdanovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2025 // P. 55, fig. – 18, tab. – 9, annexes – 4, references – 41.

Keywords: time management, task management, freelance, companies, Blazor, .NET.

The qualification work is dedicated to development of the Chronicon System for Time and Task Management for Companies and Freelancers. The aim of the work is to design, implement, and test an efficient web-based system that integrates time tracking and task management for both corporate and freelance users.

The first section describes the subject area and defines the functional and non-functional requirements of the platform.

In the second section, the architecture of the web application, the database structure, and the chosen technological stack (including .NET, Blazor WebAssembly, SQL Server, and MongoDB) are designed.

The third section covers the implementation of key modules– authorization and roles, company and freelancer management, task creation and tracking– and presents the results of functional and interface testing. The object of research is the Chronicon web platform; the subject of research is the methods and tools for integrating task management and time tracking into a single system.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – інтерфейс програмування застосунків.

Blazor – клієнтський фреймворк WebAssembly від Microsoft для побудови SPA на C#.

ООП – об'єктно-орієнтоване програмування

C# (англ. C Sharp) – ООП мова програмування платформи .NET.

DB (англ. Database) – база даних.

DI (англ. Dependency Injection) – впровадження залежностей, патерн для керування життєвим циклом об’єктів.

Docker – платформа для контейнеризації й ізоляції сервісів.

GridFS (англ. Grid File System) – механізм фрагментованого зберігання великих файлів у MongoDB.

HTTP (англ. HyperText Transfer Protocol) – протокол обміну гіпертекстовими даними в мережі.

JWT (англ. JSON Web Token) – стандарт компактної та безпечної передачі інформації між сторонами.

MongoDB – документоорієнтована СУБД.

REST (англ. Representational State Transfer) – архітектурний стиль будови веб-сервісів.

SQL (англ. Structured Query Language) – мова запитів до реляційних баз даних.

Typesense – високопродуктивний пошуковий рушій.

UI (англ. User Interface) – користувацький інтерфейс.

UX (англ. User Experience) – досвід користувача.

JWT-токен – маркер безпеки, що використовується для автентифікації та авторизації в API.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації, стрімкого розвитку цифрових технологій та зростання попиту на ефективне управління проєктами й робочим часом, автоматизація процесів організації праці стає необхідною умовою конкурентоспроможності компаній. Управління завданнями та облік робочого часу є критичними факторами підвищення продуктивності як у корпоративному секторі, так і серед фрилансерів. Тому розробка інтегрованих систем для моніторингу та оптимізації робочих процесів, контролю виконання завдань і ефективної комунікації є актуальним напрямком сучасних досліджень в галузі інформаційних технологій та систем управління.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення ефективності у сфері управління проєктами шляхом створення системи для менеджменту завдань і обліку робочого часу, що об'єднує функціонал для компаній і фрилансерів.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– проаналізувати сучасні розробки у галузі;

– визначити функціональні та нефункціональні вимоги до розроблюваної системи;

– спроектувати архітектуру системи;

– розробити серверну частину API та клієнтську частину системи;

– провести тестування функціональності системи.

Практичне значення одержаних результатів. Практичне значення результатів полягає в розробці дієвої системи, яку можуть використовувати підприємства або фрилансери для оптимізації управління проектною діяльністю, підвищення контролю над виконанням завдань, обліку робочого часу співробітників і поліпшення комунікації в команді.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області системи Chronicon

У сучасному світі інформаційних технологій, вебсайти для ефективного управління завданнями і обліку робочого часу співробітників є важливою частиною ефективної діяльності компанії. Ці веб-платформи дають можливість оптимізувати процеси планування і координації команди, а також підвищити ефективність взаємодії між учасниками проєктів.

Система Chronicon є комплексним рішенням як для компаній, так і для фрилансерів.

У даній сфері існує низка аналогічних рішень, які визначили певні стандарти функціональності та взаємодії користувачів із системами:

– Trello – це менеджер, який надає користувачу можливість керувати завданнями. Він має простий і неперевантажений інтерфейс, однак відстеження часу обмежене без сторонніх інтеграцій.

– Jira – це багатофункціональна система управління проєктами, орієнтована на великі команди. Функціонал даної системи дозволяє керувати завданнями, але її складно налаштувати, і це дорого для невеликих команд.

– Toggl Track – це інструмент виключно для відстеження часу.

– ClickUp – це комплексна платформа, яка керує завданнями, проектами та часом, і має широкий набір параметрів. Однак це може зайняти значну кількість часу, щоб вивчити інтерфейс для повного використання [1].

На відміну від вище згаданих рішень, система Chronicon є універсальною і безкоштовною. Також вона надає зручний функціонал для управління завданнями і робочим часом як командам, так і фрілансерам.

1.2 Формування вимог

У випадку розробки системи Chronicon правильне визначення вимог є критичним. Чітко сформульовані вимоги дозволяють уникнути численних ризиків: непорозуміння між замовником і розробником, перевитрати ресурсів, затримки в термінах реалізації та незадоволення кінцевих користувачів [2].

Саме тому в даній роботі вимоги систематизовано за двома основними категоріями:

– функціональні – що система повинна робити;

– нефункціональні – якими характеристиками система повинна володіти.

Функціональні вимоги визначають конкретні дії, які має виконувати система Chronicon для забезпечення потреб користувачів. У таблиці 1.1 наведено перелік основних функціональних вимог до системи. Вони охоплюють сценарії взаємодії користувача з платформою, включаючи реєстрацію, авторизацію, створення та керування завданнями, облік робочого часу, комунікацію в командах, а також інтеграцію з іншими модулями системи.

Таблиця 1.1 – Функціональні вимоги

Назва	**Опис**
1	**2**
Реєстрація користувачів	Забезпечення створення облікових записів із підтвердженням електронної пошти.
Авторизація користувачів	Можливість входу в систему за допомогою email і пароля.
Управління компаніями	Створення компаній, додавання учасників, призначення ролей (адміністратор, менеджер, співробітник).

Продовження таблиці 1.1

1	2
Створення та управління проєктами	Можливість створення проєктів, встановлення відповідальних осіб.
Створення та управління завданнями	Додавання завдань до проєктів із зазначенням статусу, пріоритету, виконавцями.
Відстеження робочого часу	Старт, пауза, завершення обліку часу.
Рольова модель доступу	Налаштування прав доступу залежно від ролі користувача.
Чат з користувачами	Можливість створити чат і листуватись з користувачами
Створення і використання акаунта фрилансера	Можливість створити акаунт і управляти завданнями від компаній

Нефункціональні вимоги встановлюють характеристики, які повинна мати система Chronicon для забезпечення зручності, безпеки, масштабованості та надійності. У таблиці 1.2 наведено основні нефункціональні вимоги.

Таблиця 1.2 – Нефункціональні вимоги

Назва	**Опис**
1	**2**
Зручність використання (UX/UI)	Інтуїтивно зрозумілий інтерфейс, що мінімізує кількість дій для виконання основних функцій.
Продуктивність	Час відгуку сервера не повинен перевищувати 2 секунд при стандартному навантаженні.

Продовження таблиці 1.2

1	2
Надійність	Забезпечення безперервної роботи системи з мінімальним простоєм.
Безпека	Використання шифрування даних, захист API через JWT-токени, безпечне зберігання паролів.
Масштабованість	Можливість додавання нових функціональних модулів без значних змін в існуючій архітектурі.
Підтримуваність	Структура коду має забезпечувати легке обслуговування та розширення.

Систематизація вимог допомагає врахувати не тільки функціональність системи, але й аспекти її надійності, зручності та подальшого розвитку.

1.3 Опис акторів та їх використання системи Chronicon

У процесі розробки системи Chronicon було визначено ключових акторів, що беруть участь у її роботі. Кожен актор має власні особливості взаємодії з системою, набір можливостей і відповідальність.

Важливо, що всі актори системи спочатку є користувачами, але в подальшому їх роль визначається вибором типу діяльності або приєднанням до компанії [3].

У цьому підрозділі наведено докладний опис кожного основного актора системи.

Користувач – це загальне поняття для будь-якої особи, яка зареєструвалася в системі Chronicon. Під час реєстрації користувач обирає свій шлях: стати

фрілансером або створити власну компанію. Також користувач може приєднатися до існуючої компанії за запрошенням.

Користувачі мають унікальні облікові записи, захищені паролем, і доступ до функціоналу відповідно до своєї ролі.

Фрилансер – це користувач, який працює самостійно , не створюючи компанію.

Основні можливості:

– створювати особистий профіль з можливістю зазначити навички;

– контактувати із замовником;

– взаємодіяти з поставленими задачами.

Фрилансери використовують систему для власного контролю за робочим часом і оптимізації процесу роботи із замовниками.

Власник компанії – це користувач, який під час реєстрації або роботи в системі обрав створення нової компанії.

Власник компанії має розширені права, зокрема:

– створення облікового запису компанії;

– створення та налаштування ролей з відповідними правами;

– запрошення співробітників до компанії;

– управління проєктами та завданнями всередині компанії.

Власник компанії не тільки керує своєю організацією, але й визначає політику доступу співробітників шляхом гнучкого налаштування ролей за системою RBAC [4].

Співробітник компанії – це користувач, який отримав запрошення до існуючої компанії і прийняв його.

Співробітник отримує доступ до системи відповідно до тієї ролі, яка була йому призначена власником компанії.

Основні можливості співробітника можуть включають:

– перегляд та виконання призначених завдань;

– відстеження часу виконання роботи;

– участь у командній взаємодії (залишення коментарів, оновлення статусів завдань).

Кожен співробітник має обмежений рівень доступу відповідно до політики компанії, що забезпечує безпеку та гнучкість управління в системі. Важливо врахувати, що власник компанії теж може бути її співробітником і виконувати задачі [5].

Незареєстрований користувач має доступ до обмежених функцій системи Chronicon. Він може переглянути останні новини, зв'язатись з службою підтримки сайту та почати процес реєстрації з подальшим вибором шляху використання системи.

На рисунку 1.1 подано як незареєстрований користувач може взаємодіяти з системою.

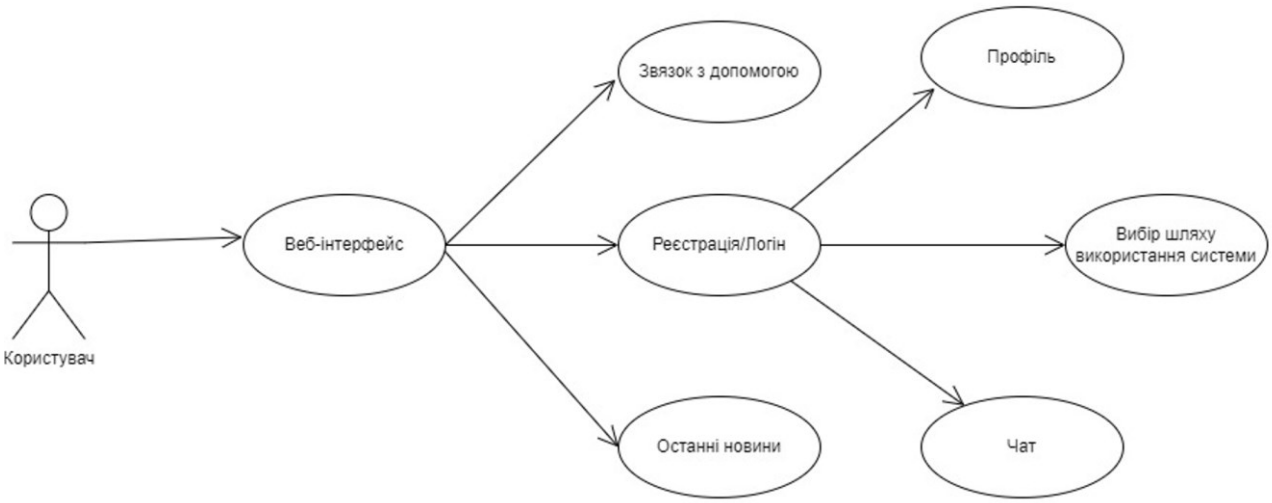


Рисунок 1.1 – Діаграма використання користувача

Користувач може обрати один із трьох варіантів використання системи Співробітник, Власник компанії, Фрилансер.

Фрилансер використовуючи веб-інтерфейс має можливість створити свій профіль ,переглянути актуальні задачі ,вести облік часу (див. рисунок 1.2).

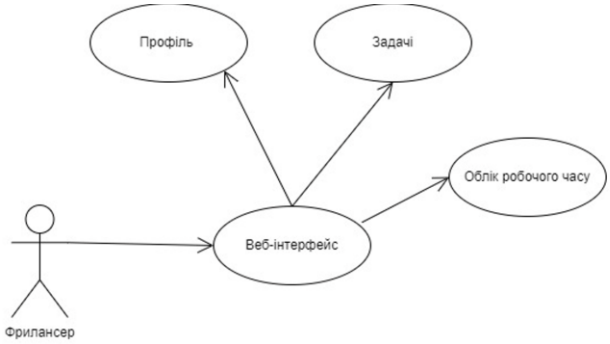


Рисунок 1.2 – Діаграма використання фрилансера

Наступною діаграмою буде використання власника компанія яка зображена на рисунку 1.3.

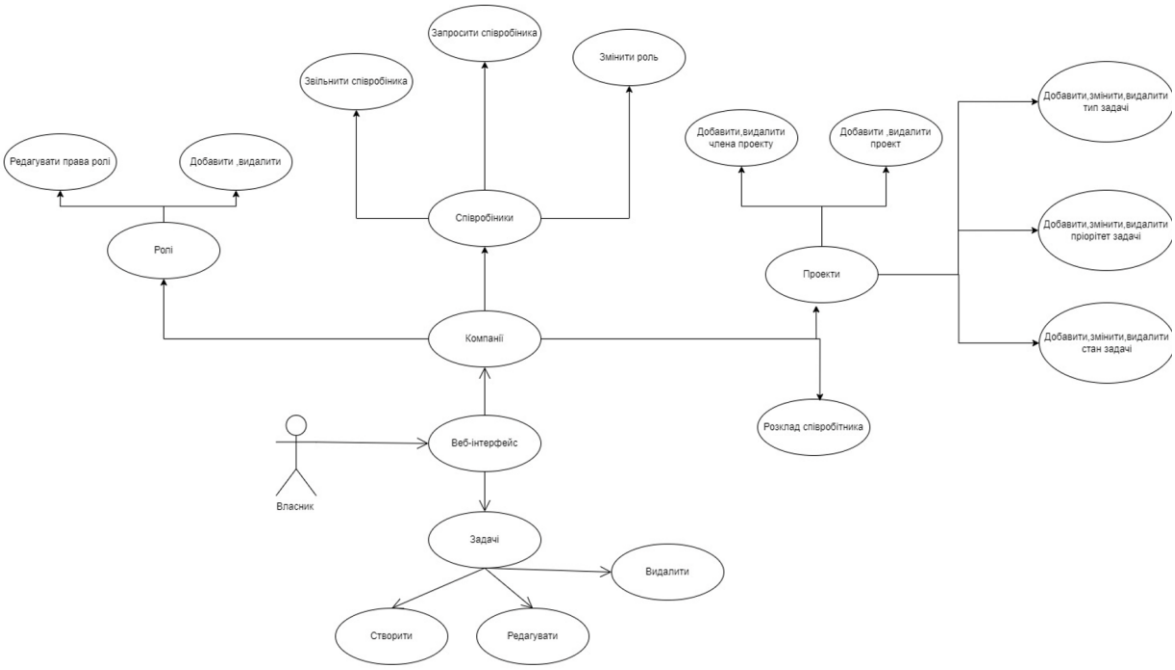


Рисунок 1.3 – Діаграма використання власника

Власник має повний доступ до більшості функцій системи Chronicon. Він визначає, як розвиватиметься проєкт, за якими принципами створюватимуться задачі та які співробітники їх виконуватимуть.

Співробітник – це тип користувача який не має чітко визначених можливостей оскільки всі його права та функції визначає власник компанії через систему ролей.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було розглянуто основних конкурентів платформи. Також було сформульовано основні функціональні і не функціональні вимоги і детально описано акторів системи. Це дозволяє приступити до проектування системи Chronicon.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ CHRONICON

2.1 Вступ і вибір інструментів для створення системи Chronicon

Розробка сучасної системи потребує ретельного підбору технологій, що забезпечать стабільність, масштабованість, безпеку та зручність подальшого супроводу системи. У процесі створення системи Chronicon обрано набір інструментів, що дозволяють реалізувати повнофункціональну систему із розділенням клієнтської та серверної частин, ефективною роботою з базами даних, пошуком, захистом та логуванням.

У рамках цієї роботи система реалізована у вигляді веб-сайту. Основу платформи становить .NET як серверна частина та Blazor WebAssembly як клієнтська, що забезпечує повноцінну взаємодію між користувачем та API [6][7].

Для зберігання даних застосовується комбінація реляційної та документоорієнтованої баз даних SQL та MongoDB відповідно.

Крім цього, до складу системи входять інструменти для обробки пошуку (Typesense), безпеки файлів (ClamAV), логування (Seq) та підтримки інфраструктури за допомогою Docker, Caddy, GridFS,та Docker Mailserver .

2.2 Дизайн користувацького інтерфейсу

Ціллю підрозділу є проектування UI/UX концепції для забезпечення відповідності баз даних і сервісів реальним сценаріям використання. Підрозділ зосереджується на проектуванні інтуїтивно зрозумілих та ефективних інтерфейсів для користувачів. Крім того, він працює над покращенням якості обслуговування [8].

2.2.1 Тоновість, кольори та стильова бібліотека

В інтерфейсі Chronicon використовується мінімалістичний дизайн на базі Bootstrap 5 (див. рисунок 2.1) з кастомізованими стилями, що відповідають корпоративній палітрі та принципам доступності [9].

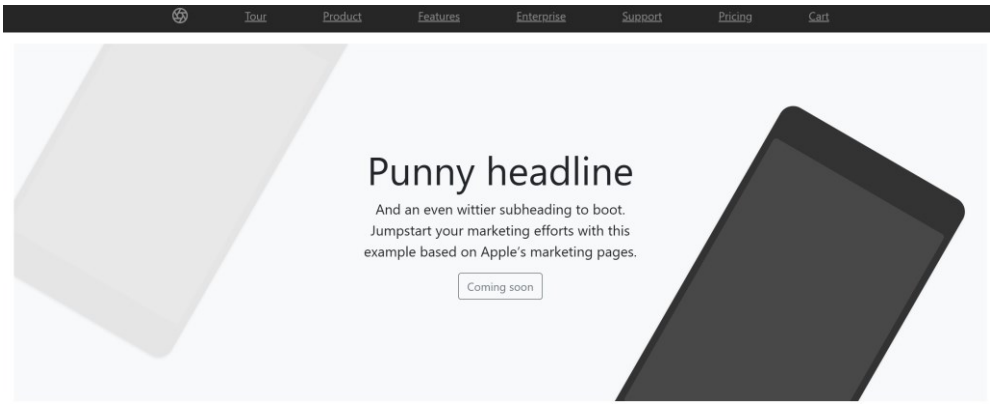


Рисунок 2.1 – Приклад використання Bootstrap

Нижче в таблиці 2.1 наведено перелік основних кольорів інтерфейсу та їх призначення:

Таблиця 2.1 – Кольорова палітра

Категорія	**Колір**	**HEX**	**Призначення**
1	**2**	**3**	**4**
Primary	зелений	#28a745	Активні дії, головні кнопки, підтвердження
Secondary	синій	#007bff	Підкреслення активних табів, посилання
Neutral light	світло-сірий	#f8f9fa	Фонові панелі, картки, відступи
Neutral dark	темно-сірий	#343a40	Основний текст, заголовки

Продовження таблиці 2.1

1	2	3	4
Accent	бірюзовий	#17a2b8	Інформаційні повідомлення, іконки статусів

Ця палітра та бібліотека гарантують уніфікованість інтерфейсу, швидке прототипування й високу доступність [8].

У дизайні використовується типографіка, заснована на шрифті "Helvetica Neue" як базовому. Заголовки мають розміри від 24 px для H3 до 48 px для H1 з жирністю від 600 до 700. Для відступів і вирівнювання застосовується сіткова система Bootstrap 5 з кроком 8 px, використовуючи утиліти m- та p-. Контейнери вирівнюються по центру екрана відповідно до точок перелому: sm – 576 px, md – 768 px, lg – 992 px, xl – 1200 px.

2.2.2 Контрастність та доступність

Усі текстові компоненти відповідають вимогам WCAG 2.1 AA(контрасніть не менше 4:5:1 для основного тексту). Інтерактивні елементи (кнопки, посилання) оснащені виразним фокус-станом. Обрана палітра кольорів і типографічна система формують цілісну візуальну концепцію продукту, забезпечують зручність сприйняття та відповідають актуальним стандартам доступності.

2.2.3 Узагальнений огляд сторінок

Нижче подано зведену таблицю 2.2. В ній описно ключові сторінки системи Chronicon із вказівкою основних компонентів і призначення.

Таблиця 2.2 – Зведена таблиця сторінок

Сторінка	Основні компоненти	Призначення
Landing	Hero–секція (заголовок, підзаголовок, фонова ілюстрація), CTA	Презентація продукту, залучення користувача
Header & Nav	Логотип, глобальне меню (Work, Agile, Companies, Tasks, Account), профіль	Головна навігація по модулях
Account	Вкладки (Account, Freelance account), поля введення, кнопки Save/Add	Налаштування профілю користувача
Стартовий майстер	Карточка «Choose type» з чекбоксами, кнопка Next, іконка Home	Конфігурація типу акаунта
Company Dashboard	Sidebar (Add Company + список), Tab-контейнер (Employees, Schedule)	Управління компаніями та ролями
Create Issue	Ліва панель: RichText–редактор, Title; права панель: Properties, кнопки	Створення та налаштування задачі,чат
Agile	Рядок фільтрів (Company, Project, Types, Priorities, States), Preset list	Збереження та застосування фільтрів
Work Schedule	Таймер (hh:mm:ss), статус, кнопка Start, DatePicker, таблиця сесій	Облік робочого часу та винятків

Ці сторінки дозволяють користувачу виконувати в повному обсязі повсякденні задачі. Вони забезпечують зручний доступ до ключових функцій системи, таких як перегляд і редагування завдань, контроль часу, управління проєктами та комунікація з командою.

2.2.4 Header із глобальним меню

Хедер виконує ключову роль у забезпеченні послідовної навігації між функціональними модулями системи Chronicon. Згідно з принципами UI/UX архітектури, він завжди доступний у верхній частині вікна та містить елементи навігації [10].

Ліворуч розташовано клікабельний логотип «Chronicon», що повертає користувача на лендінг-сторінку. Логотип зображено на рисунку 2.2.

Рисунок 2.2 – Логотип в Header

Використання зрозумілої піктограми годинника підкреслює функціональне призначення платформи.

Праворуч розташовно меню розділів що зображено на рисунку 2.3. Активний пункт підкреслюється кольором #007bff (Secondary), що відповідає рекомендаціям по візуальній ієрархії.

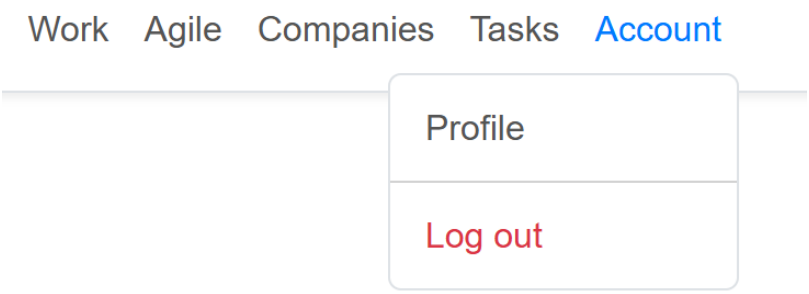


Рисунок 2.3 – Меню навігації

При наведенні курсора застосовується легка зміна фону (hover-ефект), що покращує зворотній зв’язок із користувачем і розгортається додаткове меню з підпунктами.

Таким чином, хедер виконує функції орієнтації, навігації та глобальної ідентифікації системи, забезпечуючи єдину точку входу до всіх основних розділів платформи [11].

2.3 Проектування баз даних

У даному підрозділі буде детально розглянуто структури баз даних, включаючи діаграми сутностей та їхні зв’язки. Особливу увагу буде приділено структурі реляційної бази даних [12].

2.3.1 SQL

В веб системі Chronicon реляційна база даних використовує спеціальну структуровану мову запитів SQL. Ця мова використовується для створення, зміни та отримання даних у реляціній базі даних. База даних керується системою Microsoft SQL Server розроблена компанією Microsoft логотип зображено на рисунку 2.4 [13].

Рисунок 2.4 – Логотип MSS

Microsoft SQL Server має багато плюсів і ось його основні особлисті:

– Підтримує мову SQL для роботи з таблицями, зв’язками та транзакціями.

– Глибоко інтегрується з платформою .NET, що спрощує роботу через Entity Framework [14].

– Забезпечує надійність, масштабованість і безпеку даних.

– Має зручні засоби адміністрування, зокрема SQL Server Management Studio (SSMS).

У проєкті SQL Server використовується для зберігання основних структурованих даних: користувачів, проєктів, задач, ролей, сесій роботи тощо.

2.3.2 Mongo DB

MongoDB – це документоорієнтована база даних, що зберігає дані у вигляді JSON-подібних документів. Вона не потребує фіксованої схеми, підтримує вкладені структури та є ефективною для зберігання гнучких, слабо зв’язаних даних.На рисунку 2.5 зображено логотип [15].

Рисуноку 2.5 – Логотип MongoDB

У проєкті MongoDB використовується для:

– зберігання вкладених структур без складних реляційних зв’язків (наприклад пресети);

– зберігання файлів (зображень, файлів) через GridFS.

GridFS дозволяє зберігати великі файли у вигляді фрагментованих блоків прямо в базі MongoDB, що спрощує їх асоціацію з повідомленнями, користувачами чи іншими об'єктами системи [16].

2.4 Структура реляційної бази даних

У системі основні структуровані дані : користувачів, компанії, проєкти, задачі, ролі, сесії тощо зберігаються в SQL базі даних. Дані організовано у вигляді таблиць із чітко визначеними зв’язками, що забезпечує цілісність і узгодженість інформації [17]. На рисунку 2.6 зображено список таблиць які система Chronicon використовує.

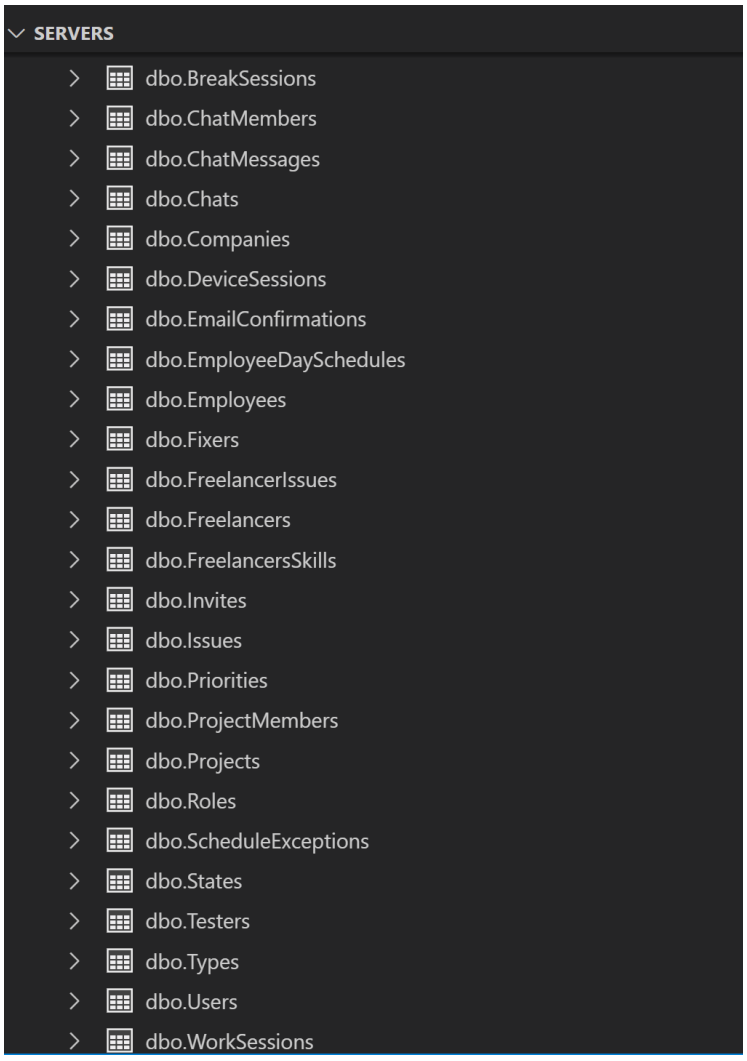


Рисунок 2.6 – Список таблиць в реляційній базі

Додатково для відображення схеми бази даних використано ER-діаграму, яка відображає сутності, їх атрибути та взаємозв’язки [18]. Діаграми та їх опис наведені в додатку Г.

2.5 Проектування архітектури системи

Архітектура системи Chronicon побудована за принципом розділення на окремі логічні компоненти, кожен з яких виконує свою роль у функціонуванні системи. Такий підхід дозволяє досягти високого рівня масштабованості, спрощує супровід коду та забезпечує гнучкість у впровадженні нових функцій.

Система складається з двох основних рівнів (див. рисунок 2.7):

– Клієнтський рівень (Frontend) – побудовано на базі Blazor WebAssembly, що дозволяє виконувати C#-код безпосередньо в браузері та знижує витрати замовника на хостинг. Клієнт взаємодіє з API сервером через HTTP-запити, обробляє інтерфейс користувача, відображає дані, отримані з сервера, та забезпечує динамічну роботу без повного перезавантаження сторінок [7].

– Серверний рівень (Backend) – побудований на основі .NET. Серверна частина відповідає за обробку бізнес-логіки, автентифікацію та авторизацію користувачів, взаємодію з базами даних, формування API-відповідей, а також інтеграцію зі сторонніми сервісами (логування, пошук, перевірка безпеки) [19].

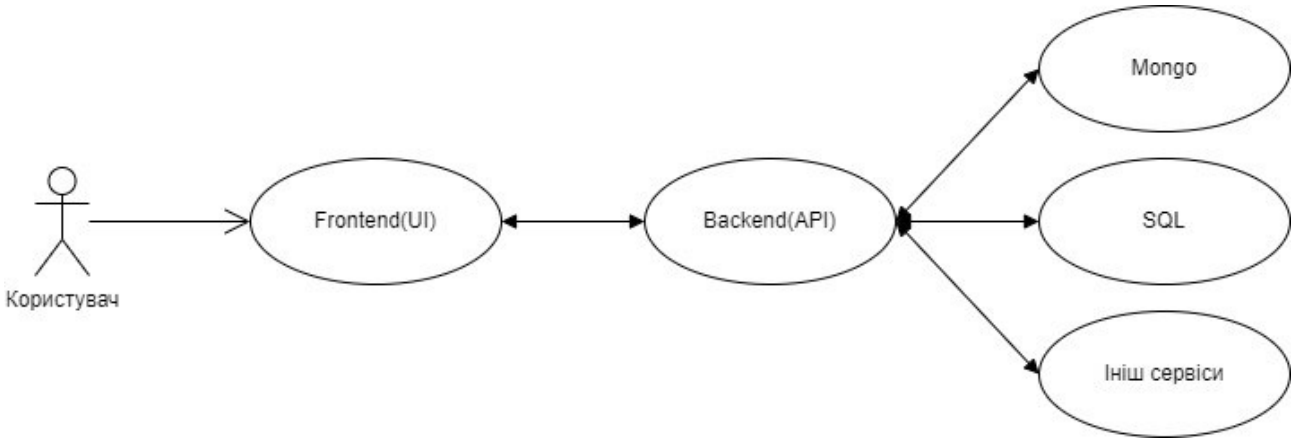


Рисунок 2.7 – Загальна архітектура проекту

Крім основних рівнів, система інтегрується з допоміжними сервісами через Docker-контейнери. Це дає змогу ізолювати сервіси та запускати їх незалежно одне від одного.

Таким чином, архітектура системи Chronicon поєднує сучасні підходи до розробки, забезпечуючи баланс між продуктивністю, безпекою та зручністю розгортання [20].

2.5.1 Проектування допоміжних сервісів

Для забезпечення додаткової функціональності та підвищення зручності використання, безпеки і продуктивності платформи Chronicon, інтегровано низку зовнішніх сервісів. Усі вони працюють у складі Docker-інфраструктури як окремі контейнери, що забезпечує їхню ізоляцію, масштабованість та легкість розгортання [21].

ClamAV – антивірусний рушій, який використовується для перевірки всіх файлів, що завантажуються користувачами. Цей сервіс інтегрований на серверному рівні й дозволяє виявляти потенційно небезпечний контент ще до його збереження в системі. Його використання є важливим аспектом забезпечення кібербезпеки платформи [22].

Typesense – це високошвидкісний пошуковий рушій з підтримкою повнотекстового пошуку. У системі Chronicon він використовується для реалізації швидкого й релевантного пошуку по профілях фрилансерів.

Typesense зручний у використанні, не потребує складної конфігурації та працює значно швидше за традиційні запити до SQL або MongoDB при пошукових запитах [23].

Seq – сервіс для централізованого збору і візуалізації логів з бекенду. Кожна дія користувача, помилка або подія бізнес-логіки автоматично записується у вигляді структурованого логу. Це значно спрощує діагностику проблем, моніторинг системи та аудит дій користувачів [24].

Docker Mailserver – використовуєиться для обробки поштових повідомлень (реєстрація, підтвердження електронної пошти, запрошення до компанії тощо) Це готове комплексне рішення, яке містить SMTP-сервер, фільтрацію спаму та

веб-інтерфейс. Завдяки контейнеризації його інтеграція в інфраструктуру не потребує ручної конфігурації на хості [25].

Ці супутні сервіси не лише покращують користувацький досвід, а й підвищують загальний рівень надійності, зручності обслуговування та безпеки системи. Їхнє використання дозволяє фокусувати розробку на основному функціоналі системи, делегуючи технічні аспекти перевірки, пошуку та логування перевіреним рішенням.

2.5.2 Проектування інфраструктури розгортання

Інфраструктура системи Chronicon побудована з урахуванням сучасних DevOps-практик. Основна мета – забезпечити просте, контрольоване та безпечне розгортання всіх компонентів системи, а також їхню ізоляцію і масштабованість. Для цього використовуються Docker-контейнери, Caddy як зворотний проксі-сервер. Docker дозволяє упакувати кожен компонент платформи (API, фронтенд, бази даних, сторонні сервіси) в окремий контейнер (див. рисунок 2.8).

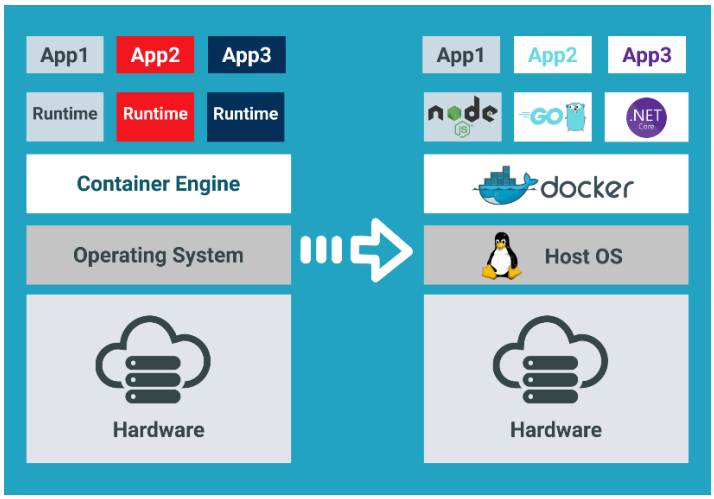


Рисунок 2.8 – Принцип контеризації

У корені кожного основного модуля (наприклад, API) розміщено власний Dockerfile, який описує інструкції для збирання відповідного сервісу [26].

Caddy виконує роль reverse proxy, що приймає всі вхідні HTTP/HTTPS-запити й перенаправляє їх на відповідні сервіси. Основною перевагою Caddy є автоматична видача SSL-сертифікатів через Let's Encrypt, що дозволяє забезпечити захищене з'єднання без додаткової конфігурації. Caddy також дозволяє легко налаштувати маршрутизацію запитів до API, фронтенду, SignalR Hub або інших служб [27].

2.6 Висновок до другого розділу

В даному розділі було спроектовано основу системи Chronicon а саме UI/UX, базу даних і сторонні сервіси. Було детально розглянуто архітектуру реляційної бази даних. Це дозволить розробити систему як відповідає сучасним стандартам.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Розробка серверної частини

Серверна частина системи Chronicon структурована за принципами модульної архітектури, що дозволяє розділити функціональність системи на незалежні логічні компоненти. Такий підхід підвищує зручність супроводу, повторне використання коду та масштабування окремих модулів.

Архітектура системи реалізована з використанням чітко розділених проектів: API, Modules, та Core, що дозволяє досягти високого рівня масштабованості, повторного використання коду та підтримуваності.

Проєкт API відповідає за конфігурацію всього застосунку та ініціалізацію залежностей. Тут зосереджено:

– запуск сервера (Program.cs, Startup.cs);

– глобальні middleware (авторизація, логування, обробка помилок);

– конфігурація DI-контейнера;

– підключення модулів.

Проєкт Modules містить бізнес-логіку, розділену на функціональні блоки (IAM, ProjectManagement, Communication тощо). Кожен модуль включає:

– контролери – точки входу в API (REST);

– сервіси – основна бізнес-логіка;

– reqms / resms – моделі запитів і відповідей (DTO);

– окремі простори імен для чіткої ізоляції модулів.

Такий підхід дозволяє масштабувати систему без конфліктів між модулями та спрощує тестування.

Кожен модуль може містити власні сервіси, моделі,контролери та репозиторії, що дозволяє ефективно ізолювати логіку. Взаємодія між модулями реалізується через спільні інтерфейси або сервіси, зареєстровані в DI-контейнері.

Проєкт Core є фундаментальним. Він включає:

– Сутності (Entities) – моделі бази даних.

– DbContext – конфігурація EF Core.

– Інтерфейси репозиторіїв і UnitOfWork.

– Загальні класи: результати, винятки, атрибути, хелпери.

Core не має залежностей від інших частин, що дозволяє його легко повторно використовувати або тестувати ізольован.

Уся архітектура (див. рисунок 3.1) реалізована з підтримкою впровадження залежностей (Dependency Injection), що дозволяє змінювати реалізації без модифікації споживача, та відповідає принципам SOLID [28].

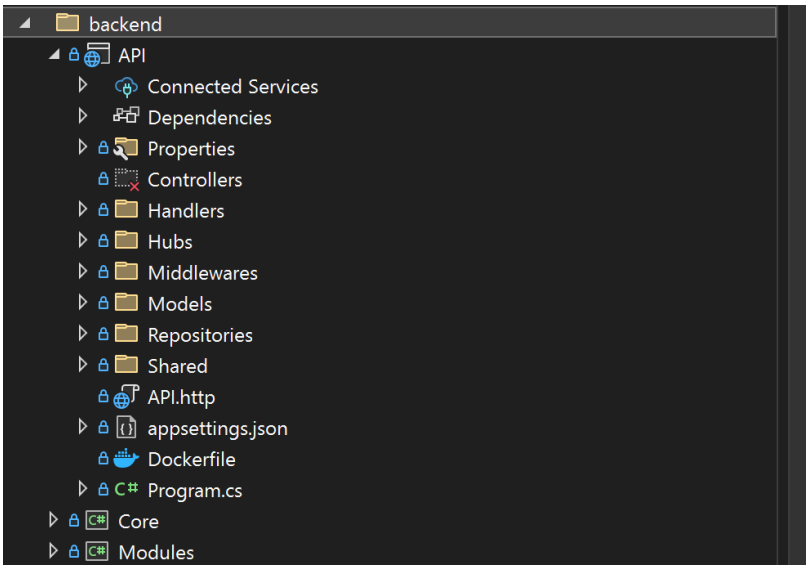


Рисунок 3.1 – Архітектура backend

Для логування використовується інтеграція з Seq, що дозволяє збирати й аналізувати структуровані логи приклад зображено на рисунку 3.2. Безпека файлів забезпечується через виклик ClamAV, а Typesense використовується для побудови пошукового функціоналу по профілям фрілансерів.

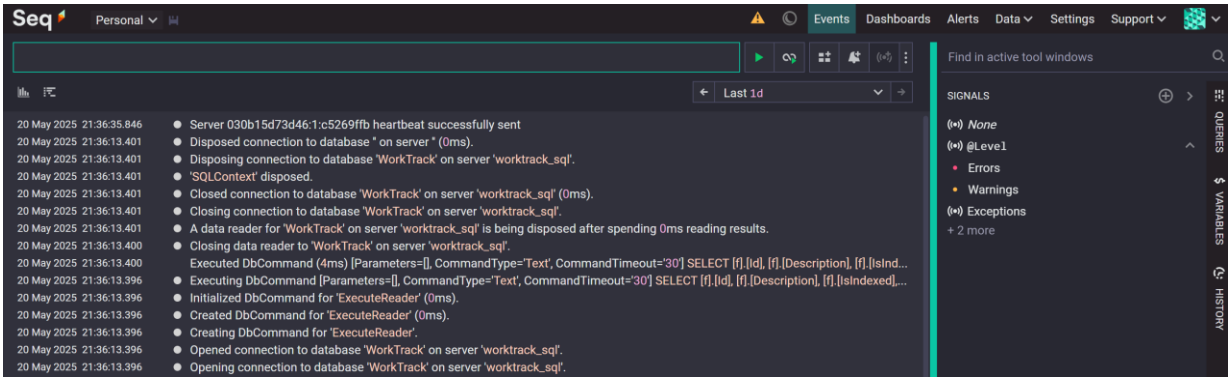


Рисунок 3.2 – Логи в систему Seq

Весь бекенд працює в Docker-контейнерах, що дозволяє легко керувати середовищами розробки й продакшену, а також масштабувати окремі сервіси у майбутньому.

3.1.1 Взаємодія з базами даних

Для взаємодії з реляційною і нереляційною базою даних у проєкті використовується Entity Framework Core (EF Core) — ORM-бібліотека, яка дозволяє працювати з таблицями бази даних через C#-класи без необхідності написання SQL-запитів вручну (14).

Абстракції у Core

У проєкті Core визначено лише інтерфейси репозиторіїв (IRepository<T>) та базові абстракції (див. рисунок 3.3):

– IRepository<T> – основний дженерик-інтерфейс, який містить методи для роботи з сутностями: GetAll(), GetById(), Add(), Update(), Delete().

– Специфічні інтерфейси (IUserRepository, IProjectRepository тощо), що розширюють базовий функціонал.

Також у Core зосереджено:

– IUnitOfWork – інтерфейс для агрегування доступу до репозиторіїв та керування транзакціями.

– AppDbContext – оголошення DbSet<> для всіх сутностей.

Це дозволяє уникнути жорстких залежностей та зробити систему легко тестованою.

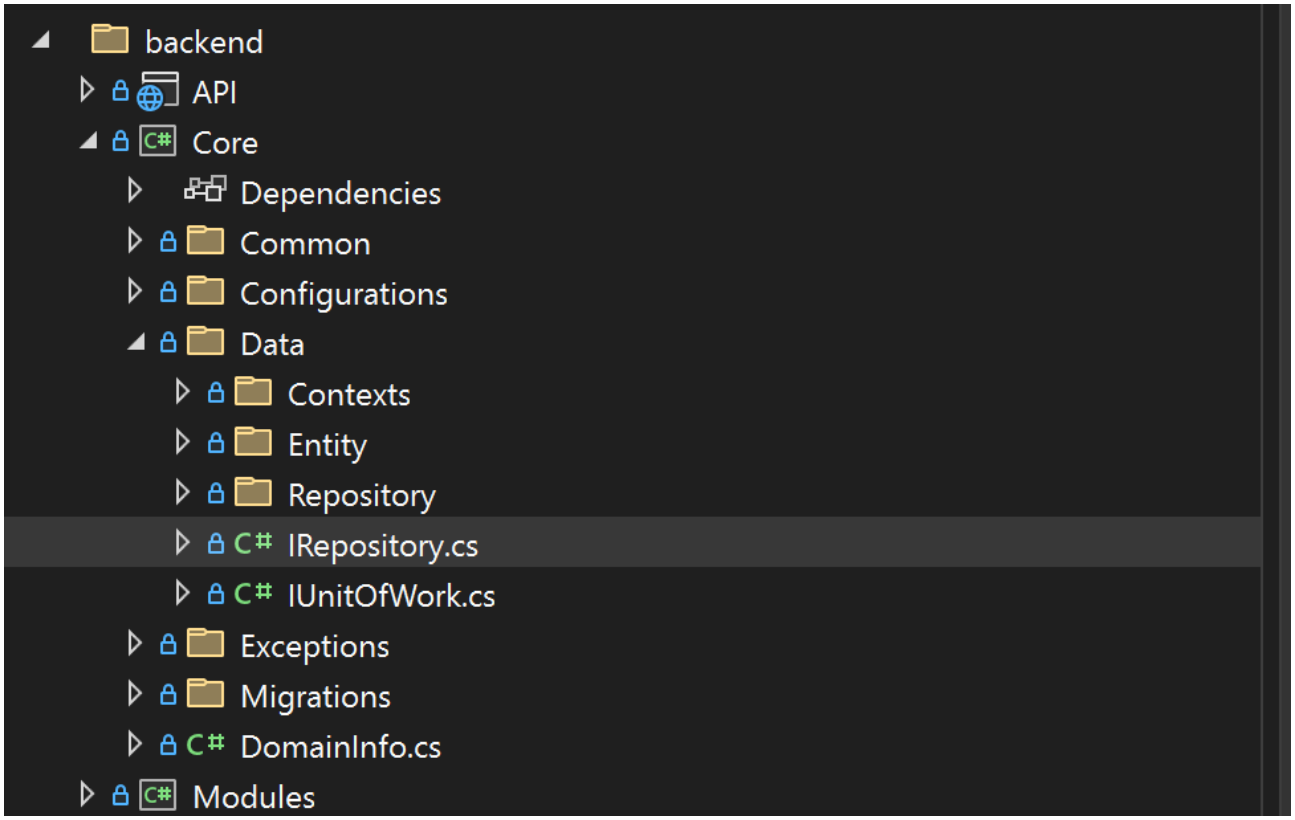


Рисунок 3.3 – Core бібліотека класів

У проєкті API реалізовані всі конкретні класи репозиторіїв:

– Repository<T> – базова реалізація IRepository<T> з використанням EF Core.

– UserRepository, ProjectRepository, ChatRepository – специфічні реалізації для окремих модулів.

Також тут реалізовано UnitOfWork – клас, який відповідає за виклик SaveChanges() і транзакції.

Централізація у проєкті API дозволяє чітко розділити інфраструктурну логіку і спрощує конфігурацію залежностей у DI-контейнері [41].

Транзакції та узгодженість

У складних сценаріях (наприклад, створення задачі разом із чатом і виконавцями) реалізовано підтримку транзакцій (див. рисунок 3.4) EF Core. Це дозволяє гарантувати цілісність даних – або всі зміни будуть збережені, або відхилені повністю у разі помилки [29].

```
public async Task<bool> DeleteIssueAsync(string userId, string issueId)
{
    await sqlUnitOfWork.BeginTransactionAsync();
    try
    {
        var issue = await issueRepository.FindAsync(issueId);
        var role = await employeeRepository.GetRoleByUserId(userId, issue.CompanyId);
        if (role?.DeleteIssue != true) throw new NotAccessException();
        var testers = await testerRepository.WhereAsync(p => p.IssueId == issueId);
        foreach (var tester in testers)
        {
            await testerRepository.DeleteAsync(tester);
        }
        var fixers = await fixerRepository.WhereAsync(p => p.IssueId == issueId);
        foreach (var fixer in fixers)
        {
            await fixerRepository.DeleteAsync(fixer);
        }
        var images = HtmlHelper.ExtractImagesId(issue.Description ?? "");
        foreach (var image in images)
        {
            await fileHandler.DeleteFileAsync(image);
        }
        var chat = await chatRepository.FindAsync(issue.Id);
        await chatRepository.DeleteAsync(chat);


        var result = await issueRepository.DeleteAsync(issue);
        await sqlUnitOfWork.CommitAsync();
        return result;
    }
    catch (Exception)
    {
        await sqlUnitOfWork.RollbackAsync();
        throw;
    }
```

Рисунок 3.4 – Приклад використання транзакцій

Такий підхід поєднує гнучкість абстракцій, контроль за транзакціями та можливість централізованого керування всіма запитами до бази в одному місці.

3.1.2 Авторизація і захист даних

Забезпечення безпечного доступу до даних є критичним компонентом системи Chronicon. Для цього реалізовано повноцінну авторизацію на основі JWT-токенів, рольову модель доступу (RBAC), а також низку внутрішніх перевірок у контролерах і сервісах. Дані користувачів захищено як на рівні передачі, так і зберігання [4][30] .

Аутентифікація (JWT)

Для підтвердження особи користувача використовується аутентифікація через JWT (JSON Web Token). Після успішного входу в систему API генерує токен, що містить:

– унікальний ідентифікатор користувача;

– термін дії.

Токен передається у заголовку Authorization: Bearer і автоматично перевіряється через middleware. Цей механізм дозволяє реалізувати безстейтову авторизацію – кожен запит перевіряється окремо, без збереження сесій на сервері [31].

Рольова модель доступу

У системі реалізовано гнучку рольову модель (RBAC). Ролі (RoleEntity) належать до конкретної компанії і визначають перелік прав:

– управління задачами, фрілансерами, співробітниками;

– створення/редагування/видалення об'єктів;

– перегляд задач, звітів, часу.

Права доступу враховуються на рівні сервісів і контролерів. У разі недостатніх прав виконується перевірка, й повертається помилка 403 Forbidden.

Захист паролів і токенів

Паролі зберігаються у вигляді хешу з сіллю, що унеможливлює відновлення початкового пароля навіть у разі витоку бази. На рисунку 3.5 показано збереженого користувача і хешований пароль. Токени оновлення (RefreshToken) зберігаються окремо в DeviceSessionEntity разом з інформацією про пристрій та термін дії [32].

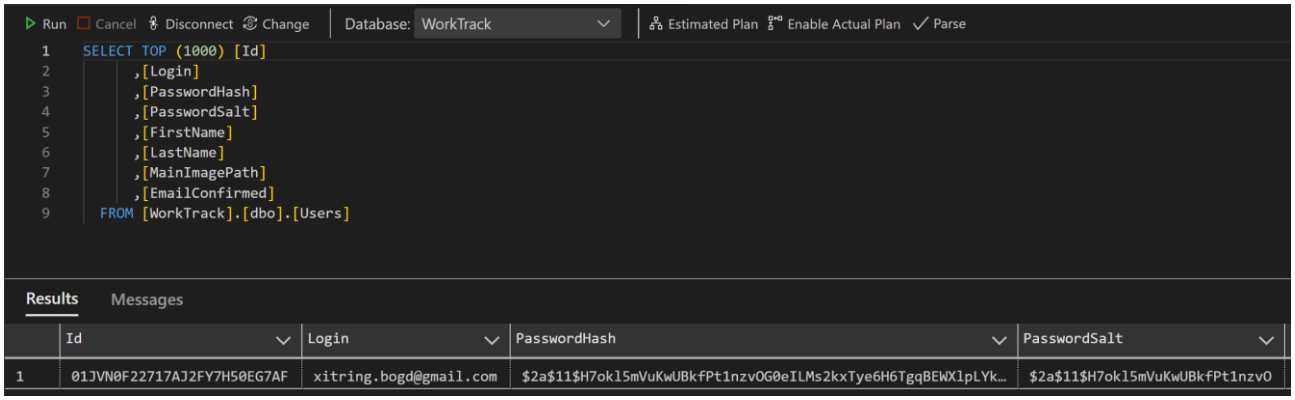


Рисунок 3.5 – Користувач збережаний в sql

Завдяки поєднанню аутентифікації, рольової авторизації, валідації та контрольованого доступу, веб-платформа Chronicon забезпечує захищену роботу з даними відповідно до сучасних стандартів безпеки.

3.2 Розробка клієнтської частини

Клієнтська частина системи Chronicon реалізована з використанням Blazor WebAssembly – сучасної технології, яка дозволяє запускати .NET-код безпосередньо в браузері користувача. Це забезпечує єдність стеку розробки (C# як на клієнті, так і на сервері) та зменшує потребу у використанні JavaScript для реалізації взаємодії з API [7].

Основними перевагами використання Blazor WebAssembly є:

– Кросплатформеність: застосунок працює в усіх сучасних браузерах без необхідності встановлення додаткових плагінів.

– Повноцінна SPA-архітектура: сторінки завантажуються один раз, після чого навігація та оновлення вмісту відбуваються без перезавантаження.

– Реактивний інтерфейс: використання компонентів дозволяє ефективно оновлювати інтерфейс у відповідь на зміну стану.

– Виконаня коду в користувача в браузері: весь код виконується в браузері що дозволяє економити на обчислювальних ресурсах замовнику.

Фронтенд-проєкт структуровано за принципами розділення відповідальностей. Основна папка frontend містить два проекти Infrastructure і WEB. В Infrastructure знаходяться сервіси для взаємодії з API (використання HttpClient для виконання запитів) та інші загальні сервіси, а WEB – це основний проєкт (див. рисунок 3.6) [40].

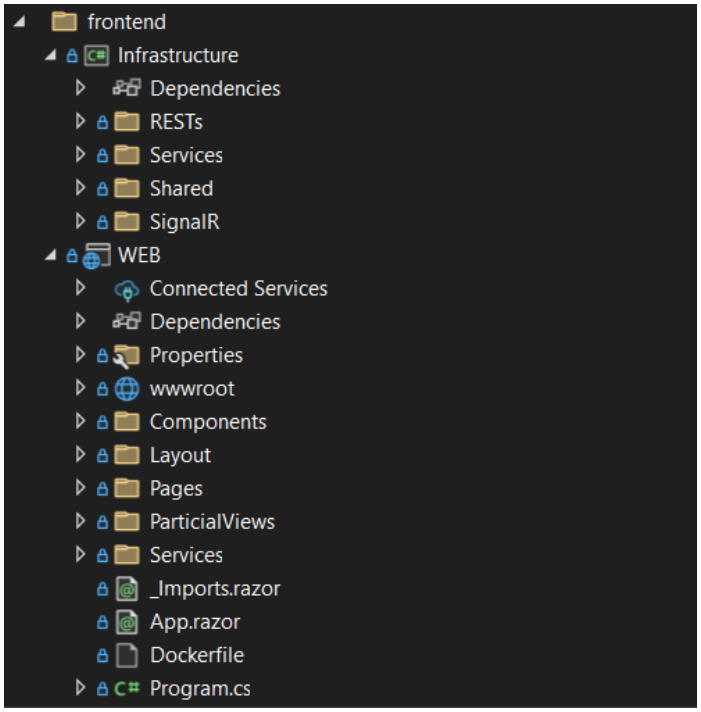


Рисунок 3.6 – Архітектура frontend

Кожна функціональна частина інтерфейсу (наприклад, поляв вводу, діалоги і тд.) реалізована у вигляді компонентів, які повторно використовуються в межах різних сторінок застосунку. Завдяки цьому код залишається чистим і модульним.

Взаємодія між клієнтом і сервером відбувається через HTTP-запити до REST API.Для цього використовується Nuget пакет RESTween [33].

JWT-токен, який зберігається у браузері автоматично додається до заголовків запитів. У випадку втрати доступу або завершення сесії користувач перенаправляється на сторінку входу.

Крім того, для реалізації реального часу (наприклад, у чатах або системних сповіщеннях) використовується SignalR, який працює через WebSocket-з'єднання з серверним ChatHub [34].

Як результат, клієнтська частина системи Chronicon побудована з урахуванням сучасних вимог до інтерфейсів: вона є швидкою, адаптивною, безпечною та зручною для розширення.

3.3 Тестування системи

У цьому підрозділі описано підхід ручного функціонального тестування веб-інтерфейсу. Тестування виконувалося шляхом покрокового проходження основних користувацьких сценаріїв та перевірки відповідності фактичних результатів очікуваним [35].

Нижче наведені таблиці з тестовими сценаріями, кроками їхнього виконання та очікуваними результатами.Тестові сценарії розділені на групи відповідно до їхнього модуля в системі.

Першим модулем для тестування є авторизація і реєстрації.В таблиці 3.1 наведено сценарії тестування.

Таблиця 3.1 – Сценарії тестування модуля авторизації

Сценарій	**Кроки тестування**	**Очікуваний результат**
Реєстрація	Провести реєстрацію на сторінці /auth	Повідомлення про успіх реєстрації, редірект на сторінку відображення Account.
Логін	Провести логін на сторінці /auth	Повідомлення про успіх логіну, редірект на сторінку відображення Account.

Усі перелічені сценарії модуля авторизації та реєстрації були виконані успішно (див. додаток Д). Переходимо до тестування модуля управління компаніями, в якому перевіряються можливості зі створення компаній,

запрошення співробітників, створення проектів та керування ролями. В таблиці 3.2 наведено основні тести.

Таблиця 3.2 - Основні тести модуля компанії

Сценарій	**Кроки тестування**	**Очікуваний результат**
Створення компанії	Створити компанії на сторінці /start	Нова компанія з'являється в лівому меню, відображається у списку компаній
Запрошення співробітника	Обрати компанію на вкладці Invite ввести email співробітника і запросити.	Відображається повідомлення "Запрошення відправлено"
Створення проекту	На сторінці /companies-dashboard обираємо пункт Projects і створюємо проект.	Проект повинен зявитись в загальному списку.
Стоврення ролі	Переходимо в управління ролями на сторінці /companies-dashboard,створюємо роль.	Роль повинна відображатись в таблиці.

Результати виконання тестів модуля управління компаніями підтвердили, що всі операції обробляються коректно та без помилок (див. додаток Д). Далі розглянемо тестові сценарії для модулів задач і комунікації які наведені в таблиці 3.3.

Таблиця 3.3 – Тести модуля задач і комунікації.

Сценарій	Кроки тестування	Очікуваний результат
Створення задачі	Перейти на сторінку /create-issue і стоврити задачу.	Новий запис задачі з коректними даними відображається на /agile
Тестування чату	Надіслати повідомлення в чат задачі.	Повідомлення повинне відобразитись в усіх учасників
Створення пресета	На сторінці /agile створити пресет.	Пресет повинен відобразитись вибраним .

Тестування модуля задач і комунікації, що включає створення задач, надсилання повідомлень у чаті та керування пресетами, також пройшло успішно (див. додаток Д). Наступним кроком є перевірка функціоналу менеджменту робочого часу. Основні сценарії відображені в таблиці 3.4

Таблиця 3.4 – Основні сценарії тестування модуля

Сценарій	Кроки тестування	Очікуваний результат
Створення робого графіку	На сторінці упрвління компанією перейти в пункт Employee Schedule і налаштувати робочі години	Співробітник отримає власний робочий графік.
Управління таймером сесії	Назначити співробітнику робочі години і з його акаунта провірити таймерю.	Таймер коректно працює.

У ході тестування модуля менеджменту робочого часу було перевірено створення індивідуального графіка співробітника та роботу таймера сесії – всі сценарії виконані безвідмовно (див. додаток Д). Завершальним елементом тестування є модуль роботи з фрілансерами. В таблиці 3.5 наведено сценарії тестування.

Таблиця 3.5 – Сценарії тестування

Сценарій	**Кроки тестування**	**Очікуваний результат**
Створення акаунта фрілансера	На сторінці /account створити акаунт фрілансера	Успішне створення фрілансер і його подальша індексація системою
Пошук фрілансера і надання йому задачі	Перейти на сторінку /search-frellancer і надіслати йому задачу	Вона повинна відображатись в фрілансера в вхідних

Тестові сценарії модуля фрілансерів, що охоплюють реєстрацію акаунту та призначення завдань, також пройшли успішно (див. додаток Д). Таким чином, усі функціональні модулі системи підтвердили свою відповідність вимогам та готові до експлуатації.

У результаті проведеного ручного функціонального тестування веб-інтерфейсу та API було перевірено ключові сценарії, серед яких реєстрація та логін користувачів, створення компанії і запрошення співробітників, робота з задачами і таймером сесій, а також обмін повідомленнями через SignalR. Тестування виконувалося в актуальних версіях браузерів Microsoft Edge, а база даних була попередньо заповнена мінімальним набором даних. Жодних критичних помилок не виявлено: форми коректно валідують обов’язкові поля, інтерфейс адаптивно підлаштовується під різні пристрої, а всі операції (створення записів, запуск/паузу/зупинку таймера, миттєвий обмін повідомленнями) відпрацьовують бездоганно. Таким чином, всі тест-кейси пройдені успішно, й система готова до подальшого розгортання на хостингу.

3.4 Розгортання на хостингу

У цьому підрозділі описано покроковий процес виведення системи Chronicon у продакшен через Docker та налаштування CI/CD за допомогою GitHub Actions.

Система Chronicon розгорталась на хостингу з предвстановленою системою Debian-11.0.Доступ здійснювався по SSH.

Для викладення використовувався docker з docker-compose .Код наведено в додатку А.

Автоматизовано процес було з використаням CI/CD GitHub який використовуючи ключ RSA підключався до віддаленого хостингу і виконував автоматичне встановлення. Процес CI/CD запускався при коміті в гілку main [36]. Код находиться в додатку Б.

Для автоматичного процесу видачі HTTPS-сертифікату використано Caddy, який «з коробки» підтримує ACME (Let’s Encrypt). Достатньо вказати домен і email для сповіщень – і Caddy подбає про отримання та оновлення сертифікатів. Код в додатку В.

Обовязковим елементом https є необхідність домену.Для цього було куплено домен chronicon.site.

Після всіх налаштувань було запущено процес CI/CD і наша веб-платформа стала доступною за адресою chronicon.site (див. рисунок 3.7) .

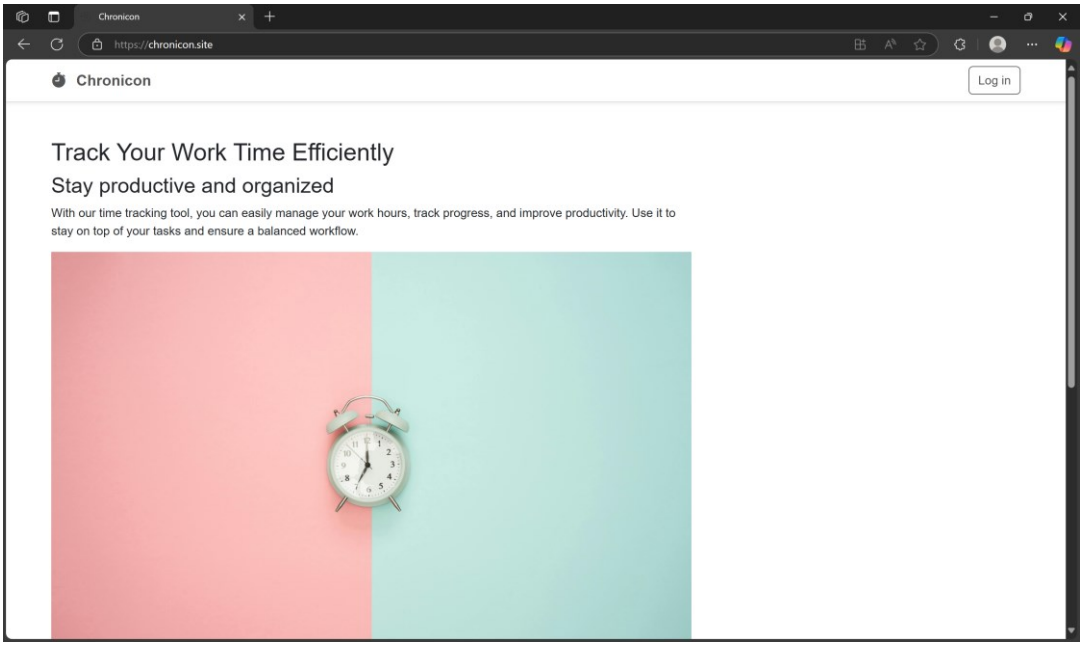


Рисунок 3.7 - Сайт доступний з інтернету

Як результат ми отримали можливість вільно використовувати нашу веб-платформу з інтернету.

3.5 Висновок до третього розділу

У даному розділі було розроблено , розгорнуто і протестовано систему Chronicon, яка стала ключовим елементом для організації управління робочим часом та завданнями. Особлива увага була приділена процесу налаштування безперервної інтеграції і доставки (CI/CD), що дозволило автоматизувати розгортання платформи та зробити її доступною для користувачів через домен chronicon.site. Завдяки цьому процесу вдалося спростити підтримку платформи, забезпечити її стабільність та оперативність оновлень.

Крім того, використання Caddy для автоматичного отримання HTTPS-сертифікатів гарантувало високий рівень безпеки даних користувачів. Інтеграція ACME (Let’s Encrypt) дозволила автоматично оновлювати сертифікати, забезпечуючи постійний захист інформації. Ці заходи зробили платформу надійною, доступною та готовою до масштабування.

Chronicon об'єднала низку функціональних модулів, таких як управління задачами, часом, комунікацією, та підтримку IAM (управління ідентифікацією та доступом), що дозволяє користувачам ефективно планувати та реалізовувати робочі процеси. У результаті, дане рішення стало важливим інструментом для оптимізації діяльності різноманітних команд та підприємств.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні проблеми безпеки життєдіяльності

Ергономіка – це галузь науки, яка досліджує людину в умовах професійної діяльності з метою забезпечення найвищої продуктивності, безпеки та комфорту.

Ергономіку розділяють на такі основні напрямки:

– вивчення особливостей людини та технічних засобів з метою налаштування їх на максимальну ефективність в умовах виробництва;

– формулювання ключових прицнипів, які з урахуванням антропометричних особливостей людини мають стати основою при створенні нових технічних засобів і технологій;

– пошуки шляхів оптимізації системи "людина — техніка — середовище".

В ергономіці використовують більше 10 величин розмірів тіла людини і її частин. Туди входить у тому числі ріст у положенні стоячи, довжину тіла з витягнутою вгору рукою, ширину плечей та інше [37].

При дослідженнях ці дані відіграють важливу роль. Адже неправильне положення тіла працівника призводить до різноманітних проблем :

– швидке накопичення втоми тіла;

– зменшення швидкості і зниження якості виконання робочих обовязків;

– повільна реакція на небезпеку.

Тому для ефективної роботи співробника він повинен знаходитись в оптимальному положення тіла при яких він не відчуватиме психологічне і фізичне напруження і як результат буде робити менше помилок у процесі виробчничої діяльності.

Правильне облаштування робочого місця та його конструкції(висоти,регулювання сидіння,достатній простір для ніш ,належна зони досяжності) забезпечує оптимальне положення тіла .

В людського організму є величина працездатність. Вона описує функціональні можливості організму і характеризується кількість та якістю роботи, виконаною за певний проміжок часу. Працездатність може змінюватись протягом роботи.

Розрізняють працездатність людини на три основних фази, які почергово змінюють одна одну в процесі трудової діяльності:

– фазу наростання працездатності;

– фазу високої стійкості працездатності;

– фазу зниження працездатності.

Остання фаза відбувається через накопичення втоми. Втома – це погіршення працездатності організму, яке проявляється у специфічних змінах фізіологічних функцій і погіршенні працездатності працівника

Занадто напружена розумова діяльність може призводити до перевтоми працівника і має своїми проявами:

– порушення сну (аж до стійкого безсоння);

– зменшення опору до впливу несприятливих зовнішніх умов;

– підвищення нервово-емоційної збудливості тощо.

Для профілактики втоми працівника можна застосувати наступні способи:

– створення зони відпочинку і відповідних умов під час обідньої перерви у середині робочого часу;

– зміна робочого графіку для короткочасних регламентованих перерв у робочий час.

При виконанні робіт, що вимагають великої нервової напруги й уваги, швидких і точних рухів рук, рекомендується використовувати не часті, але короткотривалі перерви по 5 - 10 хвилин кожна [37].

З фізіологічної точки зору особливість інтелектуальної роботи полягає в тому, що мозок людини, крім координації всіх процесів, сам по собі є головним активно працюючим органом.

Розумовій діяльності властиві такі фізіологічні прояви:

– мала рухливість;

– вимушена одноманітна робоча поза;

– приплив крові до працюючого мозку й підвищення артеріального кров'яного тиску;

– напруження функцій зорового аналізатора тощо.

Інсують наступні форми розумової праці:

– операторська. В операторів обчислювальних машин швидкість енерговитрат зростає на 60-100%;

– управлінська;

– творча. При інтенсивній інтелектуальній роботі потреба головного мозку в енергії становить 15-20% від загального обміну в організмі при тому, що маса мозку не перевищує 2% від загальної маси тіла;

– праця медичних працівників;

– праця викладачів. При виступі із публічною лекцією швидкість енерговитрат зростає на 94%;

– праця учнів і студентів.

Ергономічний підхід дозволяє підвищити продуктивності, безпеки та комфортності праці. Використовуючи антропометричні дані та оцінюючи фізіологічні особливості (зокрема витрати енергії мозком, фази працездатності й прояви втоми), можна проєктувати робочі місця й організовувати робочі процеси так, щоб мінімізувати навантаження й помилки. Короткі, регулярні перерви та правильно налаштоване обладнання допомагають попередити перевтому і зберегти увагу й швидкодію працівника протягом усього робочого дня.

4.2 Організація охорони праці на виробництві

Управління охороною праці здійснює роботодавець. Він зобов’язаний створити на робочому місці в структурних підрозділах умови праці відповідно

до нормативно-правових актів, а ще забезпечити дотримання вимог законодавства щодо прав працівників у галузі охорони праці.

Згідно з ст. 13 Закону України “Про охорону праці” роботодавець :

– забезпечує виконання необхідних профілактичних заходів відповідно до обставин;

– розробляє комлексні заходи для досягнення встановлених нормативів та підвищення існуючого рівня охорони праці;

– забезпечує належне утримання будівель і споруд,виробничого обладнання та устаткування і моніторинг за їх технічним станом;

– забезпечує усунення причин що призводять до нещасних випадків;

– проводить аудити охорони праці;

– розробляє і затверджує положення,інструкції і інші акти з охорони праці що діють в межах підприємства.

Сукупність органів управління підприємством, які на підставі комплексу нормативної документації виконують завдання і фнукції управління з метою забезпечення здорових,безпечних і високопродуктивних умов праці називають СУОП.

Чинне законодавство зобов’язує роботодавця забезпечити функціонування системи охороною праці адже в противному випадку роботодавець несе беспосередню відповідальність за порушення зазначених вимог [38].

На підприємстві, де працюють 50 і більше осіб, роботодавець створює службу охорони праці, як самостійний структурний підрозділ. Вона може функціонувати у вигляді одного співробітника:

– на підприємстві де працює менше 50 осіб – особа за сумісництвом, яка має відповідну підготовку;

– на підприємстві де працює менше 20 осіб – залучаються сторонні спеціалісти на договірних засадах, які мають відповідну підготовку.

Роботодавець беспосередньо керує службою охорони. Комплектується служба фахівцями, що мають вищу освіту і стаж роботи за профілем цього виробництва не менше трьох років.

Основні завдання служби охорони праці:

– навчання працівників
– забезпечення безпечності технологічних процесів
– нормалізація санітарно-гігєнічних умов праці
– забезпечення працівників засобами індивідуального захисту
– забезпечення оптимальних режимів праці та відпочинку

Ефективна організація охорони праці передбачає постійну співпрацю роботодавця з найманими працівниками та їх представниками, створення з ними спільних органів управління. До таких суб'єктів належить комісія з питань охорони праці підприємства [39]. Вона створюється лише за рішенням загальних зборів (конференції) найманих працівників, є постійно діючим консультативно-дорадчим органом найманих працівників та роботодавця, тому на відміну від служби охорони праці її рішення мають рекомендаційний характер.

4.3 Висновок до четвертого розділу

У четвертому розділі було розглянуто два важливих аспекти – ергономічні проблеми безпеки життєдіяльності та організацію охорони праці на підприємстві.

У першій частині опусується поняття ергономіки і її основні напрямки. Також описано фази працездатності людини і причини виникнення втоми з наслідками для працівника.Було описано способи профілакти перевтоми.

У другій частині проаналізовано, як на практиці організовується охорона праці на підприємстві. Вказано обов’язки роботодавця відповідно до законодавства, зокрема статті 13 Закону України «Про охорону праці», а також умови створення служби охорони праці залежно від кількості працівників.

Загалом у розділі розглянуто теоретичні основи й практичні підходи до створення безпечних, здорових та ефективних умов праці, що є необхідними для стабільного функціонування будь-якого підприємства.

ВИСНОВКИ

В даній кваліфікаційній роботі освітнього рівня «Бакалавр» було розроблено систему Chronicon для управління часом та завданнями для компаній та фрілансерів.

В перешому розділі кваліфікаційної роботи:

– подано загальні відомості про предметну область систем управління робочим часом та завданнями;

– розглянуто основні сценарії взаємодії ключових акторів;

– висвітлено функціональні та нефункціональні вимоги ;

– проаналізовано існуючі аналогічні рішення,визначено їх переваги та недоліки.

В другому розділі роботи:

– визначено основні технології для реалізації;

– сформовано архітектуру проекту і баз даних;

– спроектовано допоміжні сервіси.

В третьому розділі роботи була розроблена система і протестована на широкому наборі ручних тестів які показали повну функціональність платформи.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено ергономічні проблеми безпеки життєдіяльності і організація охорони праці на виробництві.

Практичне значення даної роботи полягає в тому, що система Chronicon може бути використана для:

– моніторингу робочого часу співробітників і фрілансерів;

– автоматизації обліку завдань;

– планування ресурсів і навантаження;

– аналізу ефективності роботи;

– підтримки віддаленої роботи.

У підсумку варто зазначити, що поставлена мета – підвищення ефективності у сфері управління проєктами шляхом створення системи для менеджменту завдань і обліку робочого часу – була досягнута. Розроблена платформа Chronicon успішно поєднує функціонал, необхідний як для компаній, так і для фрилансерів, що підтверджується її повною працездатністю, протестованою у процесі реалізації.

.

ПЕРЕЛІК ДЖЕРЕЛ

1. ClickUp. Trello vs. Jira: Which Project Management Tool Is Right for You? [Електронний ресурс] – Режим доступу до ресурсу: https://clickup.com/blog/trello-vs-jira/ (дата звернення: 03.06.2025).

2. Stfalcon. Як написати специфікацію вимог до програмного забезпечення. *Stfalcon.* [Електронний ресурс] – Режим доступу до ресурсу: https://stfalcon.com/uk/blog/post/How-to-Write-a-Software-Requirements-Specification (дата звернення: 02.06.2025).

3. Wikipedia. Use case diagram. *Wikipedia.* [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Use_case_diagram (дата звернення: 31.05.2025).

4. F., Ferraiolo D., A., Cugini J. та R., Kuhn D. *Role-Based Access Control (RBAC): Features and Motivations.* New Orleans, LA : National Institute of Standards and Technology (NIST), 1995. Proceedings of the 11th Annual Computer Security Applications Conference. сс. 241–248. [Електронний ресурс] – Режим доступу до ресурсу: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=916537 (дата звернення: 05.06.2025).

5. Ruslan Nebesnyi, Volodymyr Pasichnyk, Nataliia Kunanets, Nataliia Veretennikova, Oksana Kunanets.*Formation of IT Project Implementation Team*. 2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT). Т. 2, сс. 203–206.[Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1109/CSIT49958.2020.9322005 (дата звернення: 04.06.2025).

6. Microsoft. Introduction to .NET. *Microsoft Learn.* [Електронний ресурс].- Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/core/introduction (дата звернення: 26.05.2025).

7. Microsoft. Blazor ASP.NET Core. *Microsoft Learn.* [Електронний ресурс] Режим доступу до ресурсу :https://learn.microsoft.com/en-

us/aspnet/core/blazor/?view=aspnetcore-9.0&WT.mc_id=dotnet-35129-website (дата звернення: 26.05.2025).

8. Foxminded. User interface – це не просто прикраса продукту. *Foxminded.* [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/user-interface-tse/ (дата звернення: 31.05.2025).

9. Bootstrap. Introduction. *getbootstrap.com.* [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 23.05.2025).

10. Tubik. Best Practices for Website Header Design. [Електронний ресурс] – Режим доступу до ресурсу: https://uxplanet.org/best-practices-for-website-header-design-e0d55bf5f1e2 (дата звернення:08.06.2025).

11. H., Pickering. Building Accessible Menu Systems. *Smashing Magazine.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.smashingmagazine.com/2017/11/building-accessible-menu-systems (дата звернення: 05.06.2025).

12. aCode. Що таке SQL та Бази даних? *aCode.* [Електронний доступ] – Режим доступу до ресурсу: https://acode.com.ua/sql-intro/ (дата звернення: 03.06.2025).

13. Вікіпедія. Microsoft SQL Server. *Вікіпедія.* [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 28.05.2025).

14. Microsoft. Entity Framework. *Microsoft Learn.* [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/entity-framework (дата звернення: 25.05.2025).

15. MongoDB, Inc. MongoDB Manual. *MongoDB Docs.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/manual/ (дата звернення: 29.05.2025).

16. GridFS. *MongoDB Manual.* [Електронний ресурс] – Режим доступу до ресурсу:https://www.mongodb.com/docs/manual/core/gridfs/?msockid=1a54b2cf4423630704d4a653450b623f (дата звернення: 23.05.2025).

17. Guru99. Нормалізація баз даних (Database Normalization). [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/uk/database-normalization.html (дата звернення 01.06.2025).

18. dbdiagram.io. dbdiagram Docs. *dbdiagram.io.* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.dbdiagram.io/ (дата звернення:13.05.2025).

19. Microsoft. First Web API: Create a Web API with ASP.NET Core. *Microsoft Learn.* [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-9.0&tabs=visual-studio (дата звернення: 22.05.2025).

20. Microsoft. .NET Microservices: Architecture for Containerized .NET Applications. *Microsoft Learn.* [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/architecture/microservices (дата звернення: 01.05.2025).

21. Docker Inc. Docker Desktop Documentation. *Docker Docs.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.docker.com/ (дата звернення: 20.05.2025).

22. ClamAV. ClamAV Documentation. *docs.clamav.net.* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.clamav.net/ (дата звернення: 12.05.2025).

23. Typesense. Typesense Documentation. *Typesense.* [Електронний ресурс] – Режим доступу до ресурсу: https://typesense.org/docs/ (дата звернення: 30.05.2025).

24. Datalust. An overview of Seq. *Datalust Documentation.* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.datalust.co/docs/an-overview-of-seq (дата звернення : 01.05.2025).

25. Awesome Docker Compose. Docker Mailserver. *awesome-docker-compose.com.* [Електронний ресурс] – Режим доступу до ресурсу:

https://awesome-docker-compose.com/apps/email-servers/docker-mailserver (дата звернення: 19.05.2025).

26. Docker Inc. Docker Compose Documentation. *Docker Docs.* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/compose/ (дата звернення: 20.05.2025).

27. Caddy. Caddy Documentation. *caddyserver.com.* [Електронний ресурс] – Режим доступу до ресурсу: https://caddyserver.com/docs/ (дата звернення: 26.05.2025).

28. Бранець Іван. Чому SOLID – важлива складова мислення програміста. Розбираємося на прикладах з кодом. *DOU.* [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/solid-principles/ (дата звернення: 07.06.2025).

29. Microsoft. Транзакції в EF Core. *EF Core Documentation.* [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/ef/core/saving/transactions (дата звернення: 02.06.2025).

30. Небесний Руслан Михайлович та Шевчук Олександр Валерійович.*Заходи безпеки інформації у комп'ютерних системах.*Тернопіль : ТНТУ, 2018. сс. 239–240. [Електронний ресурс] – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/handle/lib/25406 (дата звернення 26.05.2025).

31. GeeksforGeeks. JSON Web Token (JWT). [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/json-web-token-jwt/ (дата звернення: 27.05.2025).

32. McCarvill Amanda. The Role of Secure Authentication and Authorization in Application Security. *Bright Security.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.brightsec.com/blog/the-role-of-secure-authentication-and-authorization-in-application-security/ (дата звернення: 07.06.2025).

33. Хітрих Богдан Богданович. RESTween. *NuGet Gallery.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.nuget.org/packages/RESTween (дата звернення: 26.05.2025).

34. Microsoft. Introduction to SignalR. *Microsoft Learn.* [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr (дата звернення: 26.05.2025).

35. Небесний Руслан Михайлович та Слободян Любомир Михайлович. *Актуальність Проблем Оцінювання Якості Програмного Забезпечення.* Тернопіль : ТНТУ, 2024. сс. 232-233. [Електронний ресурс] – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/handle/lib/25400 (дата звернення: 25.05.2025).

36. GitHub Inc. What is CI/CD? GitHub Resources. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/resources/articles/devops/ci-cd (дата звернення : 31.05.2025).

37. Н.І. Андрейчук; Ю. В. Кіт; С. В. Шибанов; О. В.Шерстньова. *Охорона праці : навч. посіб.* Видавництво "Львівська політехніка", 2021. с. 276 с.[Електронний ресурс] – Режим доступу до ресурсу: https://opcb.kpi.ua/wp-content/uploads/2014/05/kp-bzdot-i-cz-bzd2018.pdf (дата звернення: 26.05.2025).

38. І. Івах.*Основи охорони праці* : Видавництво Київ Кондор, 2010.с.464. [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.lib.vntu.edu.ua/books/2021/Ivah_2010_464.pdf (дата звернення: 04.06.2025).

39. О. Курепін. *Основи охорони праці : навч. посіб.* Миколаїв : МНАУ, 2022.с.23. [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.lib.vntu.edu.ua/books/2021/Ivah_2010_464 (дата звернення: 10.06.2025).

40. Kharchenko, A., Halay, I., & Bodnarchuk, I. (2016). Multicriteria architecture choice of software system under design and reengineering. 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT), 4–8.

41. Kharchenko, A., Bodnarchuk, I., & Yatcyshyn, V. (2014). The method for comparative evaluation of software architecture with accounting of trade-offs. American Journal of Information Systems, 2(1), 20–25.

ДОДАТКИ

Додаток А

Код docker-compose

```
version: "3.8"

services:
  mongo:
    image: mongo:latest
    container_name: worktrack_mongo
    restart: unless-stopped
    ports:
      - "27017:27017"
    volumes:
      - mongo_data:/data/db

  sqlserver:
    image: mcr.microsoft.com/mssql/server:2017-latest
    container_name: worktrack_sql
    restart: unless-stopped
    ports:
      - "1431:1433"
    environment:
      ACCEPT_EULA: "Y"
      SA_PASSWORD: "YourStrong!Passw0rd"
    volumes:
      - sql_data:/var/opt/mssql
    healthcheck:
      test: >
        /opt/mssql-tools/bin/sqlcmd
          -S localhost
          -U sa
          -P $${SA_PASSWORD}
          -Q "SELECT 1"
          >/dev/null 2>&1
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s

  seq:
    image: datalust/seq
    container_name: worktrack_seq
    environment:
      - ACCEPT_EULA=Y
    ports:
      - "5341:80"
      - "5342:5341"
    volumes:
      - seq_data:/data
    restart: unless-stopped

  mailserver:
    image: mailserver/docker-mailserver:latest
    container_name: mailserver
    hostname: mail
    domainname: chronicon.com
    ports:
```

```
      - "587:587"
    volumes:
      - maildata:/var/mail
      - mailstate:/var/mail-state
      - ./config/:/tmp/docker-mailserver/
    environment:
      - ENABLE_SPAMASSASSIN=0
      - ENABLE_CLAMAV=0
      - ENABLE_FAIL2BAN=0
      - ENABLE_POSTGREY=0
      - ONE_DIR=1
      - DMS_DEBUG=0
      - ENABLE_AMAVIS=0

    cap_add:
      - NET_ADMIN
    restart: unless-stopped

  typesense:
    image: typesense/typesense:29.0.rc9
    container_name: worktrack_typesense
    restart: unless-stopped
    ports:
      - "8108:8108"
    environment:
      - TYPESENSE_DATA_DIR=/data
      - TYPESENSE_API_KEY=xyz
    volumes:
      - typesense_data:/data

  api:
    build:
      context: ./src/backend
      dockerfile: API/Dockerfile
    container_name: worktrack_api
    restart: unless-stopped
    depends_on:
      mongo:
        condition: service_started
      sqlserver:
        condition: service_healthy
      typesense:
        condition: service_started
      seq:
        condition: service_started
    environment:
      DataBases__SQL:
"Server=worktrack_sql;Database=WorkTrack;User
Id=sa;Password=YourStrong!Passw0rd;TrustServerCertificate=True"
      DataBases__Mongo__ConnectionString:
"mongodb://worktrack_mongo:27017"
      DataBases__Mongo__Name: "TaskManager"
```

```
          JwtSettings__Secret:
"asddddddddddddddddddafwerhyjhgtreqwfghhm,mhgfdsad1"
          JwtSettings__Issuer: "your_issuer"
          JwtSettings__Audience: "your_audience"
          Typesense__Host: "http://worktrack_typesense:8108"
          Typesense__Key: "xyz"
          ClamAVS__Host: "worktrack_clamav"
          ClamAVS__Port: "3310"
          Seq: "http://worktrack_seq:5341"
          Smtp__Host: "mailserver"
          Smtp__Port: "587"
          Smtp__EnableSsl: "false"
          Smtp__Username: "noreply@chronicon.site"
          Smtp__Password: "46810rk5377"
        expose:
          - "80"

      ui:
        build:
          context: ./src/frontend
          dockerfile: WEB/Dockerfile
        container_name: worktrack_ui
        restart: unless-stopped
        volumes:
          - ui_wwwroot:/app/wwwroot
        depends_on:
          - api
        expose:
          - "80"

      caddy:
        image: caddy:alpine
        container_name: caddy_proxy
        ports:
          - "80:80"
          - "443:443"
        volumes:
          - ./Caddyfile:/etc/caddy/Caddyfile
          - caddy_data:/data
          - caddy_config:/config
          - ui_wwwroot:/app/wwwroot
        depends_on:
          - ui
    volumes:
      mongo_data:
      sql_data:
      typesense_data:
      seq_data:
      maildata:
      mailstate:
      caddy_data:
      caddy_config:
      ui_wwwroot:
```

Додаток Б

Код deploy для CI/CD

```
name: Deploy via SCP & Compose

on:
  push:
    branches:
      - master

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout repository
      uses: actions/checkout@v4

    - name: Set AppVersion in appsettings.json
      run: |
        ver=$(date +'%Y%m%d%H%M')
        echo "Setting version to $ver"
        sed   -i   "s/\"AppVersion\":   \".*\"/\"AppVersion\":
\"$ver\"/" src/backend/API/appsettings.json


    - name: Clean target folder on server
      uses: appleboy/ssh-action@v0.1.7
      with:
        host: ${{ secrets.SERVER_HOST }}
        username: ${{ secrets.SERVER_USER }}
        key: ${{ secrets.SERVER_SSH_KEY }}
        script: |
          rm -rf ~/worktrack/* || true

    - name: Copy project files to server
      uses: appleboy/scp-action@v0.1.4
      with:
        host: ${{ secrets.SERVER_HOST }}
        username: ${{ secrets.SERVER_USER }}
        key: ${{ secrets.SERVER_SSH_KEY }}
        source: "."
        target: "~/worktrack"

    - name: Deploy with Docker Compose
      uses: appleboy/ssh-action@v0.1.7
      with:
        host: ${{ secrets.SERVER_HOST }}
        username: ${{ secrets.SERVER_USER }}
        key: ${{ secrets.SERVER_SSH_KEY }}
        command_timeout: 30m
        script: |
```

```
              cd ~/worktrack
              docker compose down || true
              docker volume rm worktrack_ui_wwwroot || true
              docker compose pull || true
              docker compose up -d --build --force-recreate
          - name: Create mail user after startup
            uses: appleboy/ssh-action@v0.1.7
            with:
              host: ${{ secrets.SERVER_HOST }}
              username: ${{ secrets.SERVER_USER }}
              key: ${{ secrets.SERVER_SSH_KEY }}
              script: |
                cd ~/worktrack
                docker compose exec mailserver \
                  setup email add noreply@chronicon.site 46810rk5377
|| true
```

Додаток В

Код Caddy

```
    http://localhost {
        encode zstd gzip

        root * /app/wwwroot
        file_server

        header           /service-worker.js           Content-Type
application/javascript
        header         /manifest.webmanifest          Content-Type
application/manifest+json
    }


    chronicon.site {
        encode zstd gzip

        root * /app/wwwroot
        file_server


        @websocket path /chathub*
        reverse_proxy @websocket worktrack_api:80 {
            header_up Host {host}
            header_up Upgrade {header:Upgrade}
            header_up Connection {header:Connection}
        }

        handle /api/* {
            reverse_proxy worktrack_api:80
        }


    }

    mail.chronicon.site {
        encode zstd gzip

        reverse_proxy modoboa:80
    }

    logs.chronicon.site {
        reverse_proxy worktrack_seq:80
    }
```

Додаток Г

ER-діаграми реляційної бази даних

Група сутностей (див. рисунок Г.1), пов’язана з управлінням обліковими записами користувачів та авторизацією, включає базову таблицю користувачів та таблицю активних сесій пристроїв.

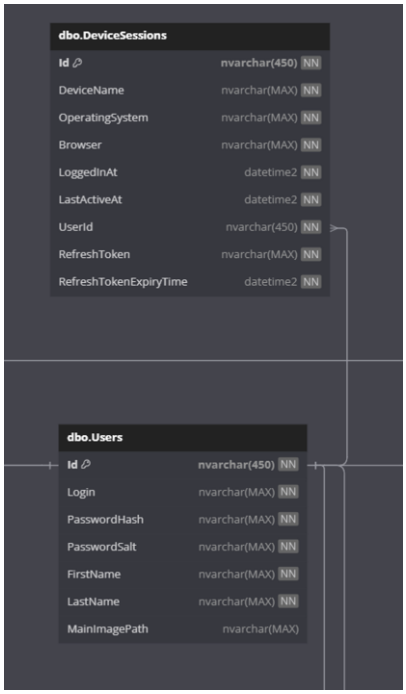


Рисунок Г.1 – Схема таблиць Users і DeviceSessions

Таблиця Users містить основну інформацію про обліковий запис користувача: логін, хешований пароль, ім’я та прізвище, а також шлях до зображення профілю. Усі поля є обов’язковими, крім аватарки.

Таблиця DeviceSessions відображає дані про активні сесії користувачів на пристроях. Кожен запис містить назву пристрою, операційну систему, браузер, час входу та останньої активності. Також зберігається токен оновлення (RefreshToken) і термін його дії. Через поле UserId встановлюється зв’язок із таблицею Users.

Цей блок є основою для реалізації механізмів авторизації, валідації доступу, багатопристроєвої роботи та підтримки безпеки сесій користувачів.

Наступна група таблиць (див. рисунок Г.2) відповідає за зберігання інформації про фрілансерів, їх навички та задачі, які компанії призначають їм для виконання. Структура дозволяє системі вести облік вільних виконавців, формувати базу спеціалістів за напрямками та керувати співпрацею з ними через окремі задачі.

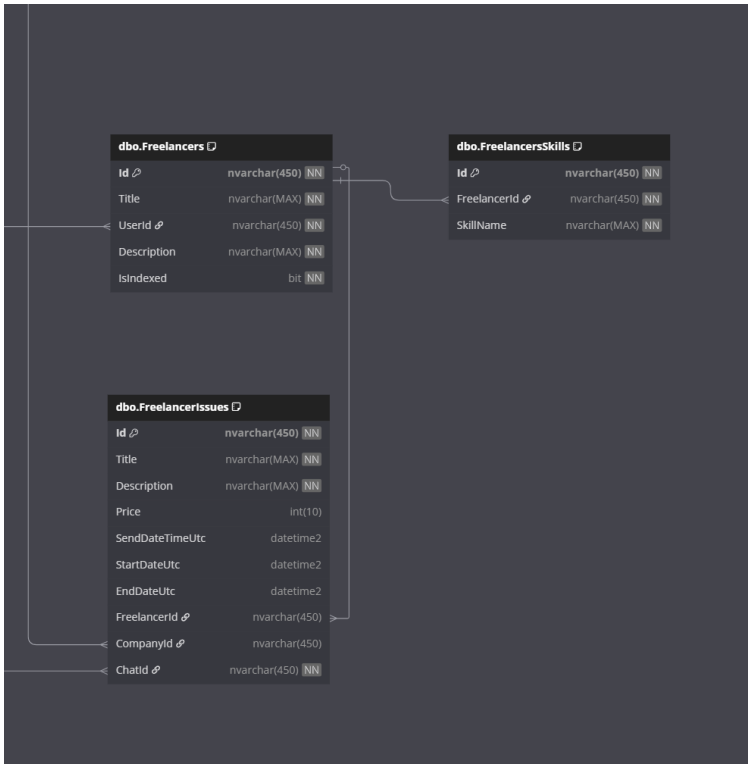


Рисунок Г.2 – Схема таблиць фрілансера

Таблиця Freelancers зберігає базову інформацію про фрілансера: назву профілю (Title), опис, ідентифікатор користувача (UserId) та прапорець IsIndexed, який вказує на включення профілю у публічний пошук.

Таблиця FreelancersSkills реалізує список навичок фрілансера. Один фрілансер може мати багато записів з назвами навичок (SkillName), що дозволяє гнучко формувати профіль виконавця.

Таблиця FreelancerIssues – це задачі, які компанії надсилають конкретним фрілансерам. Поля включають опис задачі, вартість (Price), строки виконання (StartDateUtc, EndDateUtc) і дату відправлення. Зв’язки з CompanyId та ChatId

дозволяють контролювати, яка компанія створила запит і через який чат здійснюється комунікація.

Ця група сутностей дає змогу формувати окрему логіку фріланс-взаємодії, відокремлену від штатних працівників компанії, з можливістю асинхронного призначення задач та подальшого контролю їх виконання.

Для модуль комунікації який забезпечує обмін повідомленнями між користувачами всередині системи було створено таблиці а саме : ChatMembers , Chats,ChatMessages .Чати є універсальним каналом комунікації між учасниками задач, а повідомлення зберігаються з підтримкою вкладень, часу створення та видалення.Схема зображена на рисунку Г.3

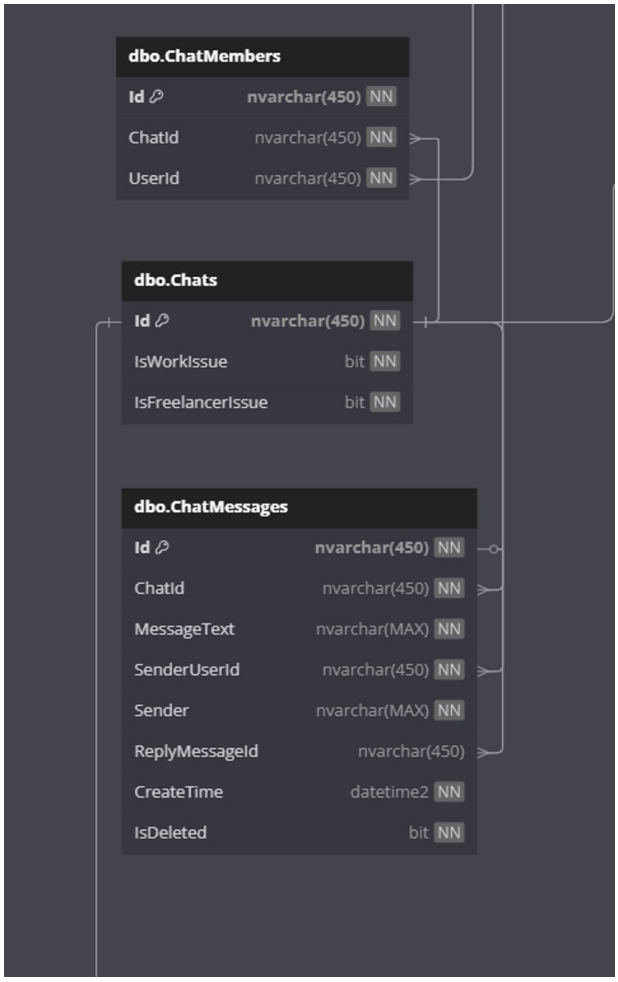


Рисунок Г.3 – Схема моделя комунікації

Таблиця Chats є центральною сутністю, яка представляє окрему розмову. Вона має два логічні прапорці: IsWorkIssue та IsFreelancerIssue, які визначають, з яким типом задачі пов'язаний чат.

Таблиця ChatMembers реалізує зв’язок M:N між чатами та користувачами один користувач може брати участь у багатьох чатах, і навпаки. Це дозволяє гнучко формувати списки учасників розмови.

Таблиця ChatMessages містить безпосередньо текст повідомлень (MessageText), дані відправника (SenderUserId, Sender) та поле ReplyMessageId для реалізації логіки відповідей (self-reference). Також зберігається дата створення повідомлення (CreateTime) і логічний прапорець IsDeleted, який дозволяє реалізувати м’яке видалення.

Ця частина структури інтегрується з іншими модулями системи, забезпечуючи внутрішню взаємодію між учасниками задач.

Група таблиць (див. рисунок Г.4) відповідає за ідентифікацію користувачів як працівників у межах компаній, формування ролей із правами доступу та облік запрошень. Це ключовий блок, що реалізує модель прав доступу на основі ролей (RBAC) та структурує багатокористувацьку взаємодію в системі.

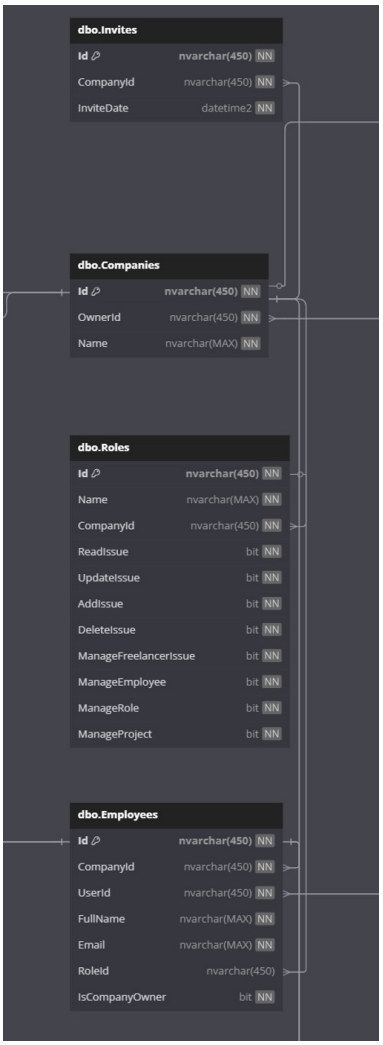


Рисунок Г.4 – Схема модуля компанії

Таблиця Companies зберігає основну інформацію про компанії. Поле OwnerId – це зовнішній ключ на користувача, який є власником компанії.

Таблиця Employees пов’язує користувача (UserId) із компанією (CompanyId). Також зберігає повне ім’я, email, роль (RoleId) і прапорець IsCompanyOwner. Один користувач може бути працівником лише однієї компанії.

Таблиця Roles містить назви ролей та набір логічних прапорців, які вказують на надані права: створення, редагування, перегляд задач, управління іншими користувачами, проєктами, фрілансерами тощо. Зв’язок з CompanyId дозволяє кожній компанії мати свій власний набір ролей.

Таблиця Invites використовується для формування запрошень нових співробітників. Містить ідентифікатор компанії та дату запрошення (InviteDate).

Ця структура дозволяє гнучко керувати доступом, розмежовувати права між працівниками та реалізовувати логіку запрошень до організацій.

Блок таблиць (див. рисунок Г.5) надалі реалізує логіку фіксації робочого часу працівників, планових графіків, перерв та винятків у розкладі. Його призначення – забезпечити облік продуктивності, контроль навантаження та формування звітності по робочому часу.

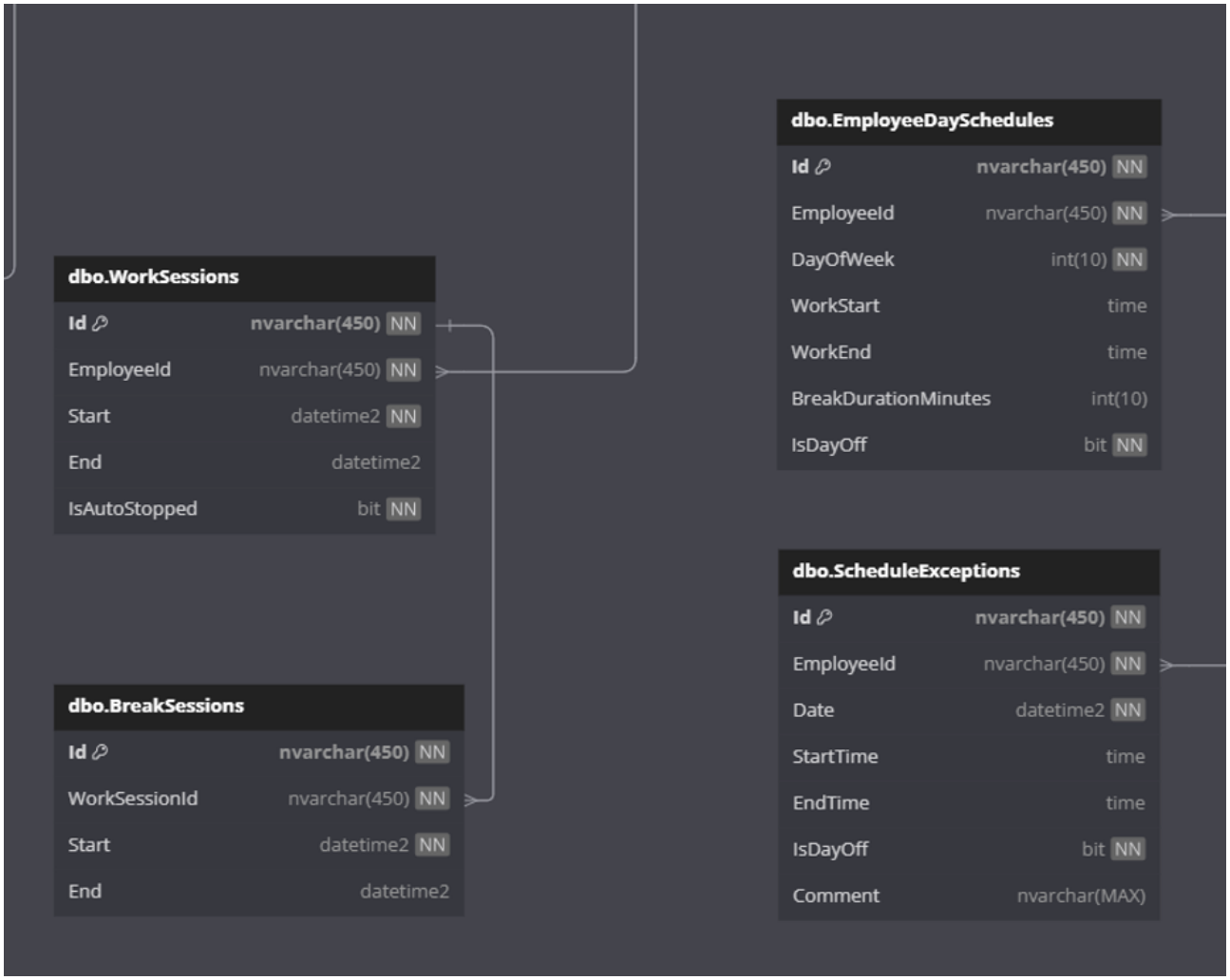


Рисунок Г.5 – Схема обліку робочого часу

Таблиця WorkSessions фіксує кожен факт початку і завершення робочого дня працівника. Також зберігається прапорець IsAutoStopped, який дозволяє

виявити автоматично завершені сесії у випадку, якщо користувач не натиснув “завершити”.

Таблиця BreakSessions пов’язана з робочими сесіями (WorkSessionId) і відображає фактичні перерви працівника з часом початку та завершення.

Таблиця EmployeeDaySchedules визначає базовий графік працівника по днях тижня: час початку, завершення, тривалість перерви (BreakDurationMinutes) і прапорець IsDayOff.

Таблиця ScheduleExceptions дозволяє враховувати винятки з графіка — наприклад, відпустку, лікарняний або зміну робочого часу на конкретну дату. Може мати Comment для пояснення причини.

Завдяки цій структурі система підтримує як стандартний режим роботи, так і змінні графіки з ручним контролем та фіксацією порушень.

Цей блок реалізує функціональність для створення проєктів, постановки задач, їхнього виконання, тестування, класифікації та контролю стану. Структура (див. рисунок Г.6) підтримує гнучку модель проєктної взаємодії з можливістю відстеження виконавців, пріоритетів і прогресу.

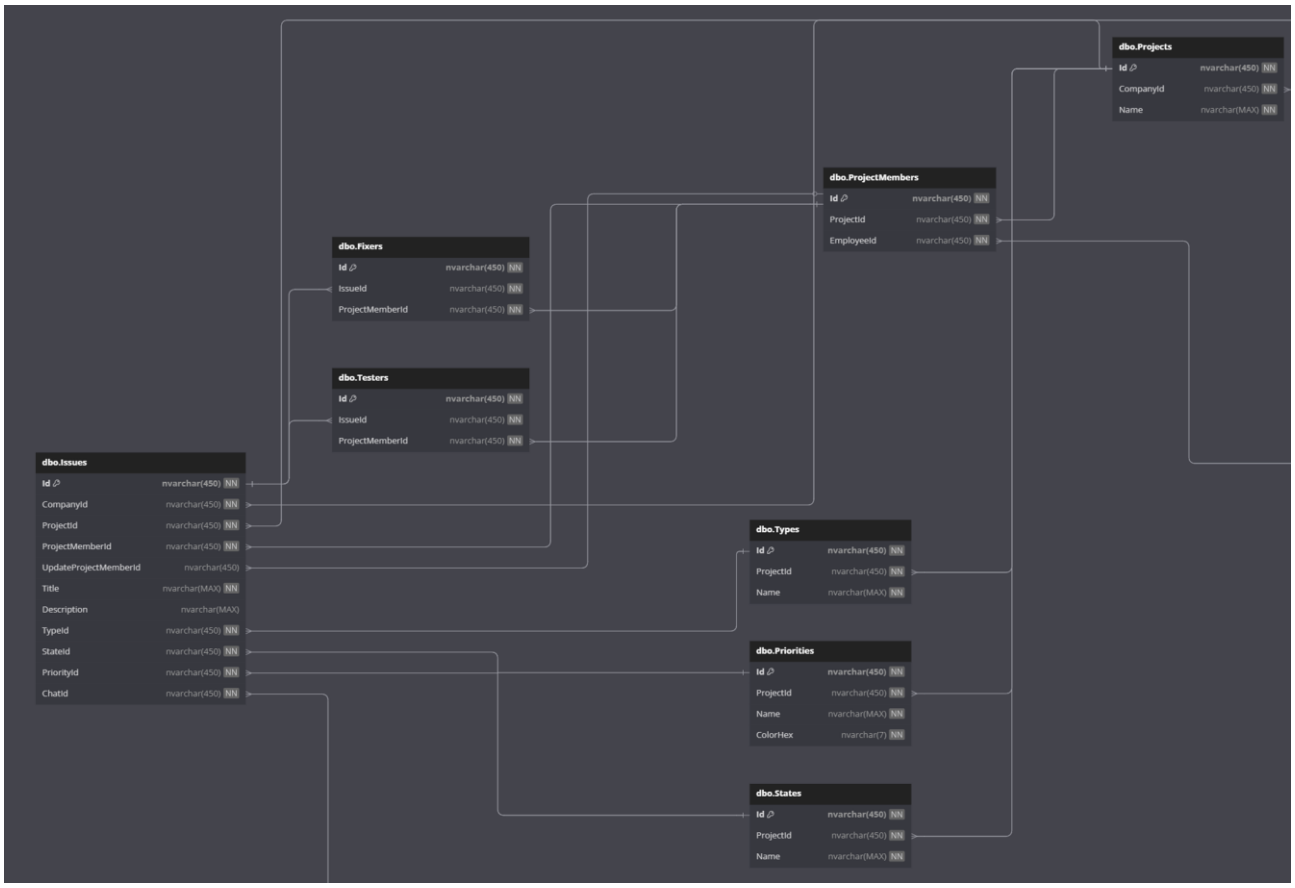


Рисунок Г.6 – Схема модуля задач

Таблиця Projects зберігає базову інформацію про проєкт, включаючи зв’язок із компанією (CompanyId) та назву проєкту (Name).

Таблиця ProjectMembers реалізує зв’язок між працівниками (EmployeeId) та проєктами. Вона дозволяє одному співробітнику брати участь у кількох проєктах.

Таблиця Issues є центральною частиною цього блоку. Кожна задача (Issue) пов’язана з:

– компанією (CompanyId)

– проєктом (ProjectId)

– виконавцем (ProjectMemberId)

– оновлювачем (UpdateProjectMemberId)

– типом (TypeId), станом (StateId), пріоритетом (PriorityId)

– чатом (ChatId) – для комунікації по задачі

Таблиці Fixers та Testers дозволяють призначати додаткових співробітників на роль виконавця (Fixer) або тестувальника (Tester) для задачі.

Вони зберігають ідентифікатор задачі (IssueId) та учасника проєкту (ProjectMemberId).

Таблиці Types, States, Priorities – допоміжні класифікатори задач. Усі вони прив'язані до конкретного проєкту (ProjectId) і зберігають назви:

– Type – тип задачі (bug, feature, enhancement…)

– State – поточний статус задачі (open, in progress, closed…)

– Priority – пріоритет із кольоровим позначенням (ColorHex)

Ця структура дозволяє масштабовано організовувати командну роботу над задачами з розмежуванням ролей і контролем виконання.

Додаток Д

Тестування функціональності системи

Першим кроком в тестуванні є перевірка модуля авториазції.

Переходимо на /auth, вводимо свій email, натискаємо continue, після чого на пошту приходить код, який ми вводимо для підтвердження, що зображено на рисунку Д.1.

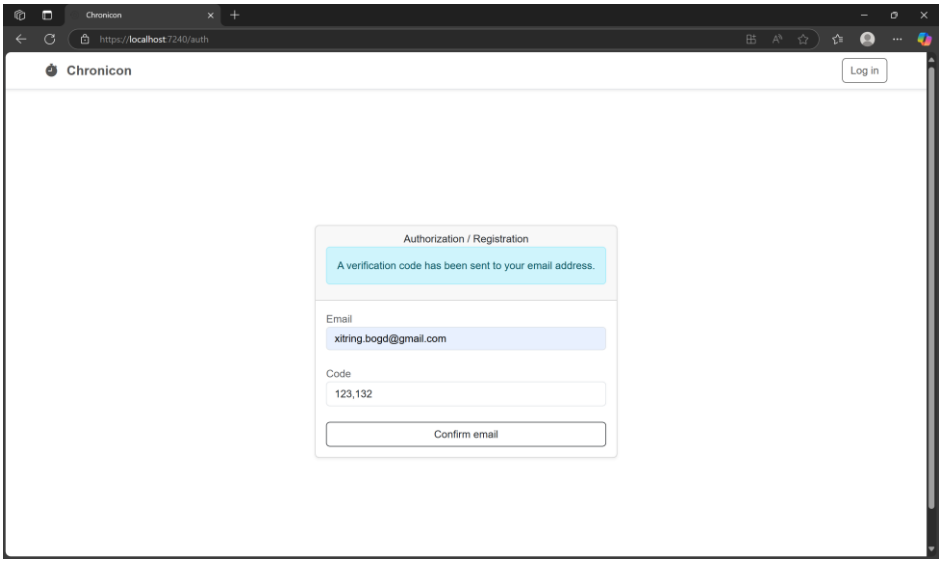


Рисунок Д.1 - Підтвердження пошти

Після підтвердження вводимо дані користувача (див. рисунок Д.2).

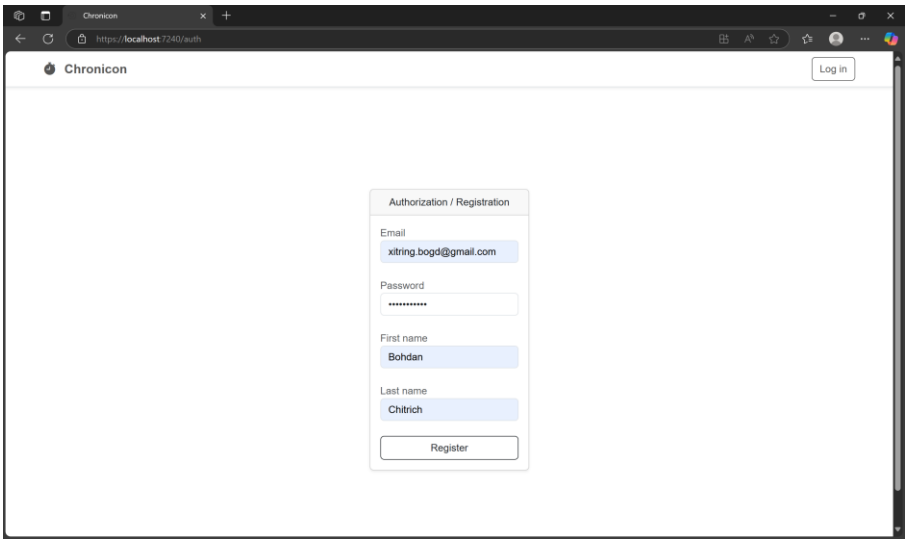


Рисунок Д.2 - Ввід даних користувача

Якщо все пройшло успішно користувача залогінит і він зможе перейти на екран акаунта що і зображено на рисунку Д.3

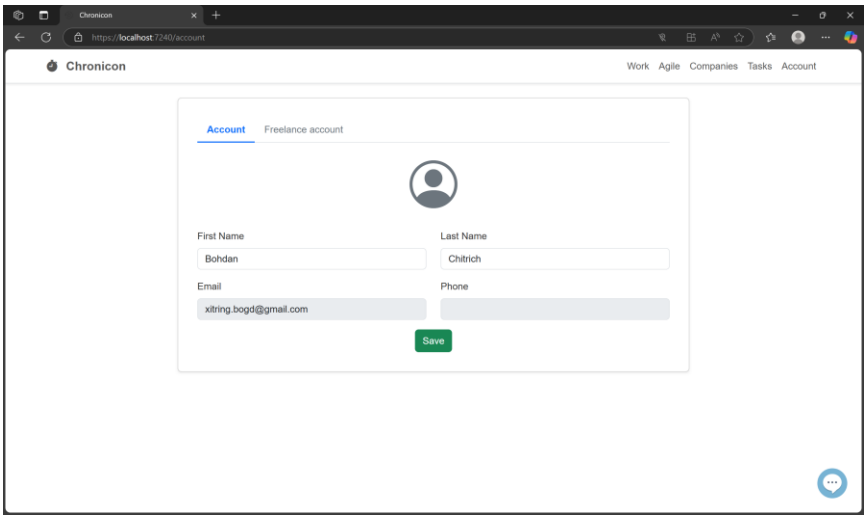


Рисунок Д.3 - Успішна реєстрація

Як результат користувач був успішно зареєстрований в системі.Виходим з акаунта і проводим логін (див. рисунок Д.4).

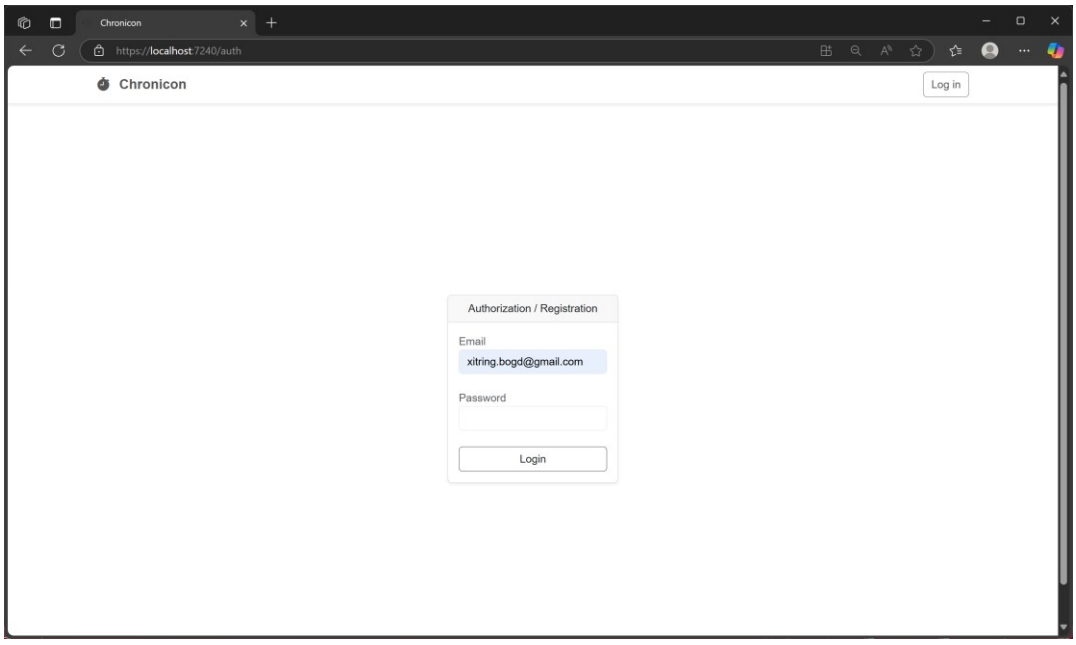


Рисунок Д.4 – Процес логіну

Після чого отримуємо успішний редірект на акаунт і отримуємо токен авторизації (див. рисунок Д.5).

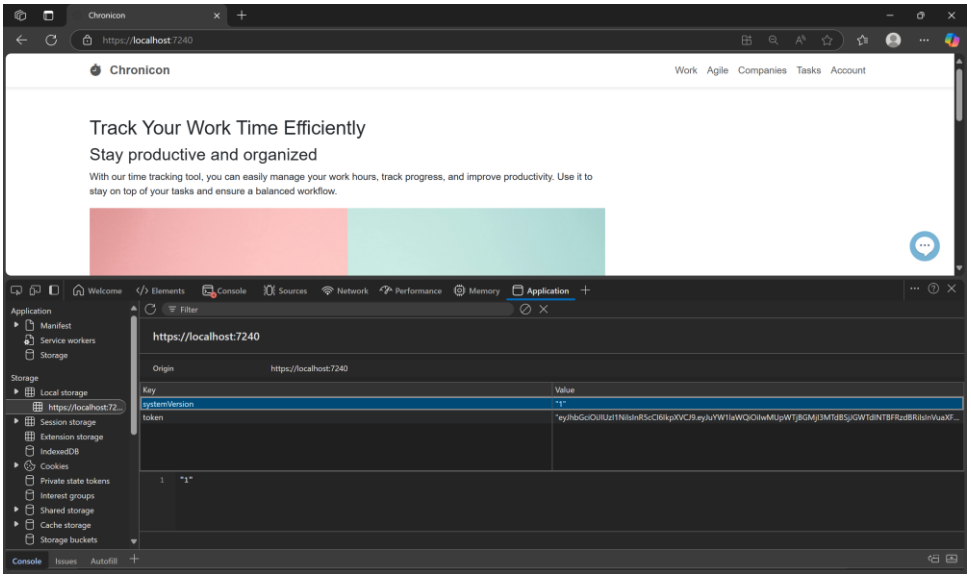


Рисунок Д.5 – Успішний логін і токен авторизації

Після успішної авторизації користувач може приступити до створення компанії що і буде наступним кроком. Для початку процесу створення компанії переходим на сторінку /start. В ибираєм пункт Owner, після чого вводимо ім'я компанії (див. рисунок Д.6).

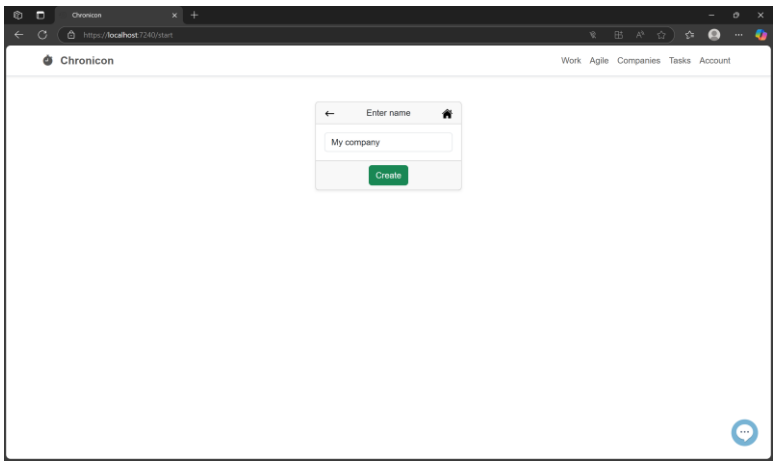


Рисунок Д.6 – Введення імені компанії

Натискаємо Create і після чого нас перекидає на сторінку Dashboard.Також ми отримали повідомлення про успішне створення (див. рисунок Д.7).

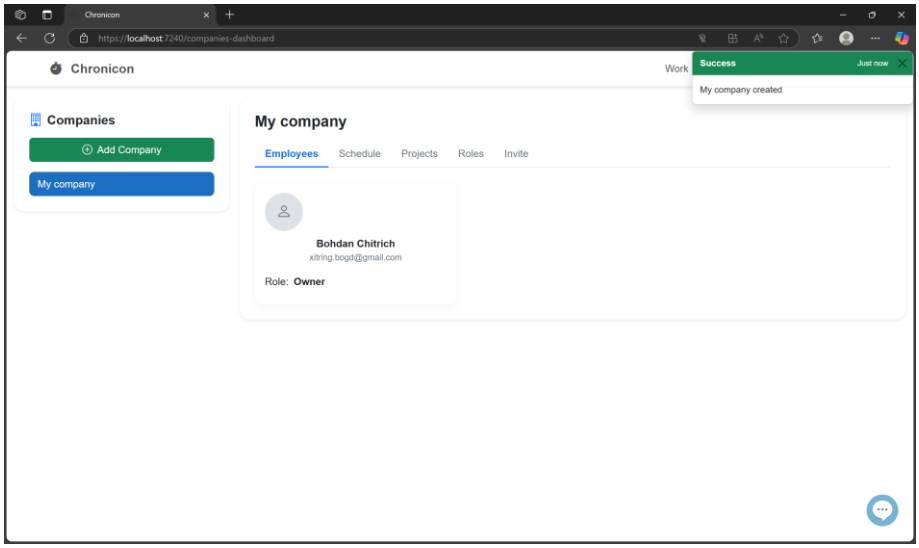


Рисунок Д.7 - Успішно пройдений тест

Компанію було успішно створено і тест пройдено. Для наступного тесту потрібно створити другий акаунт щоб його запросити до компанії. Після створення акаунта з компанії власника надсилаємо запрошення на пошту (див рисунок Д.8).

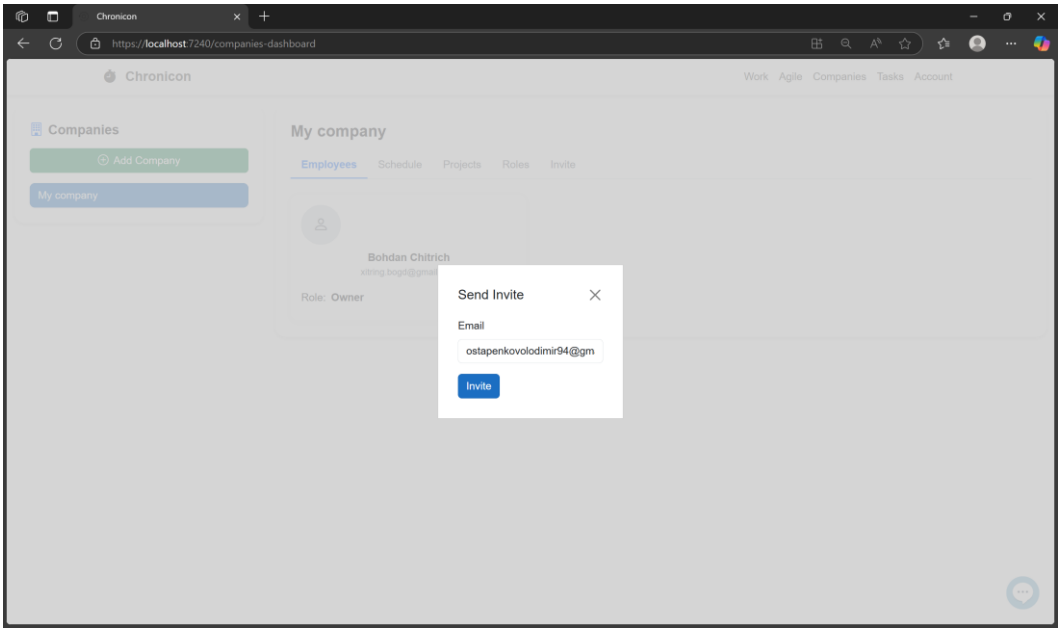


Рисунок Д.8 – Форма відправлення запрошення

Після чого на відповідну пошту прийде повідомлення з запрошенням (див. рисунок Д.9).

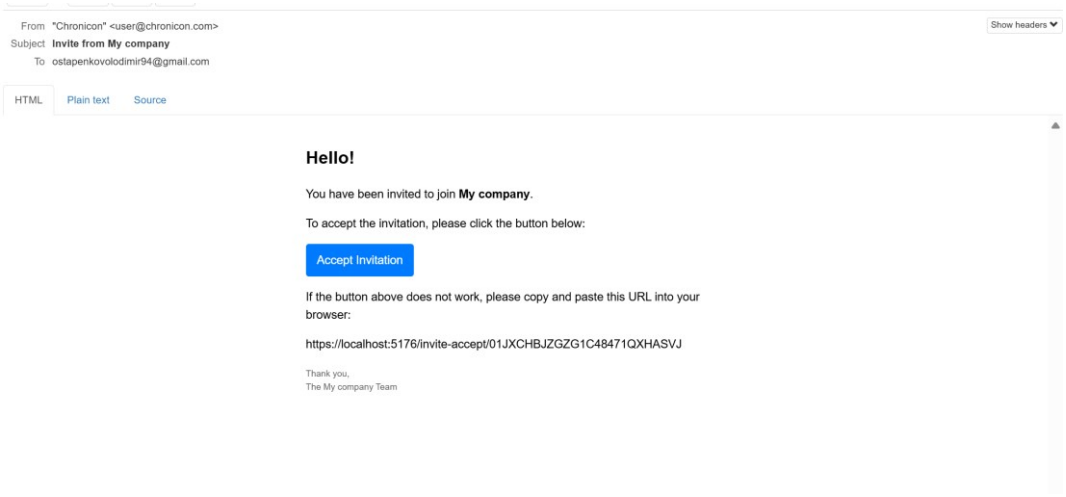


Рисунок Д.9 – Повідомлення отримане на пошті

Переходим за посиланням і нам відкривається екран де можна або прийняти або відмовити в запрошенні (див. рисунок Д.10).

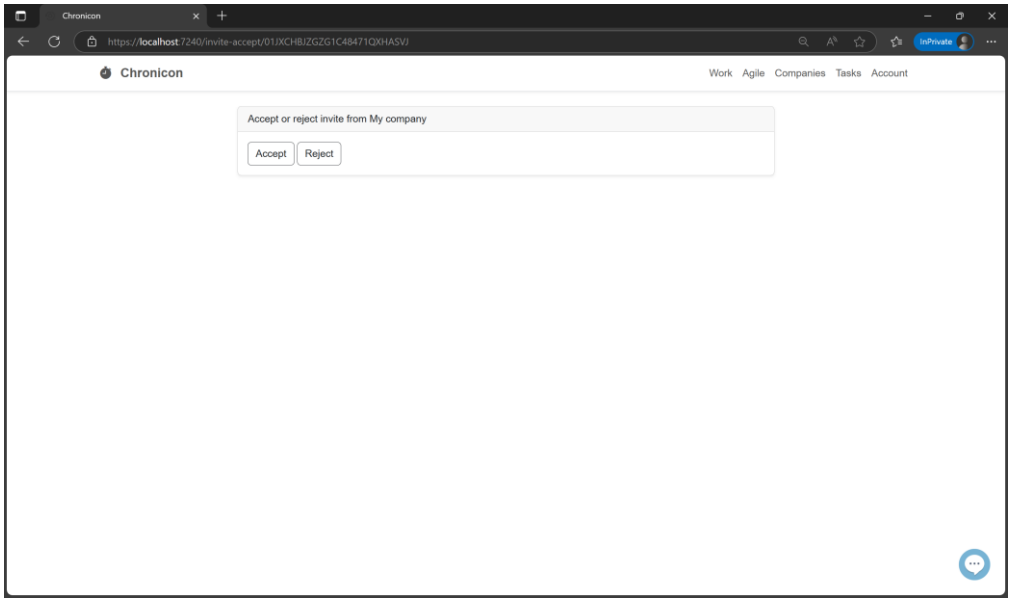


Рисунок Д.10 – Екран прийому запрошення

Приймаємо запрошення після чого в компанії переходим на екран Employees і в списку повинен появитись наш новий співробінник що і відбувалось на рисунку Д.11.

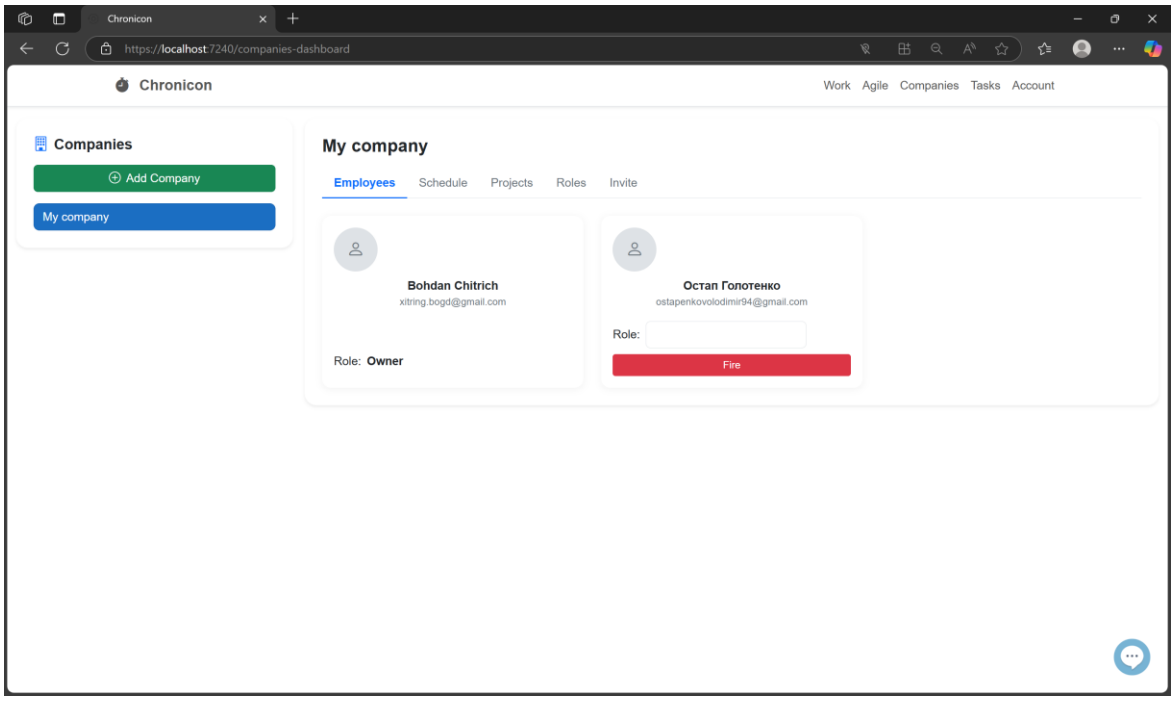


Рисунок Д.11 – Новий співробінник в компанії

Тепер спробуємо створити роль для нового співробінника .Переходимо на закладку Roles і жмемо кнопку добавити нову роль.Після чого зявляється діалог в якому вводимо імя ролі (див. рисунок Д.12) і нажимаєм Add.

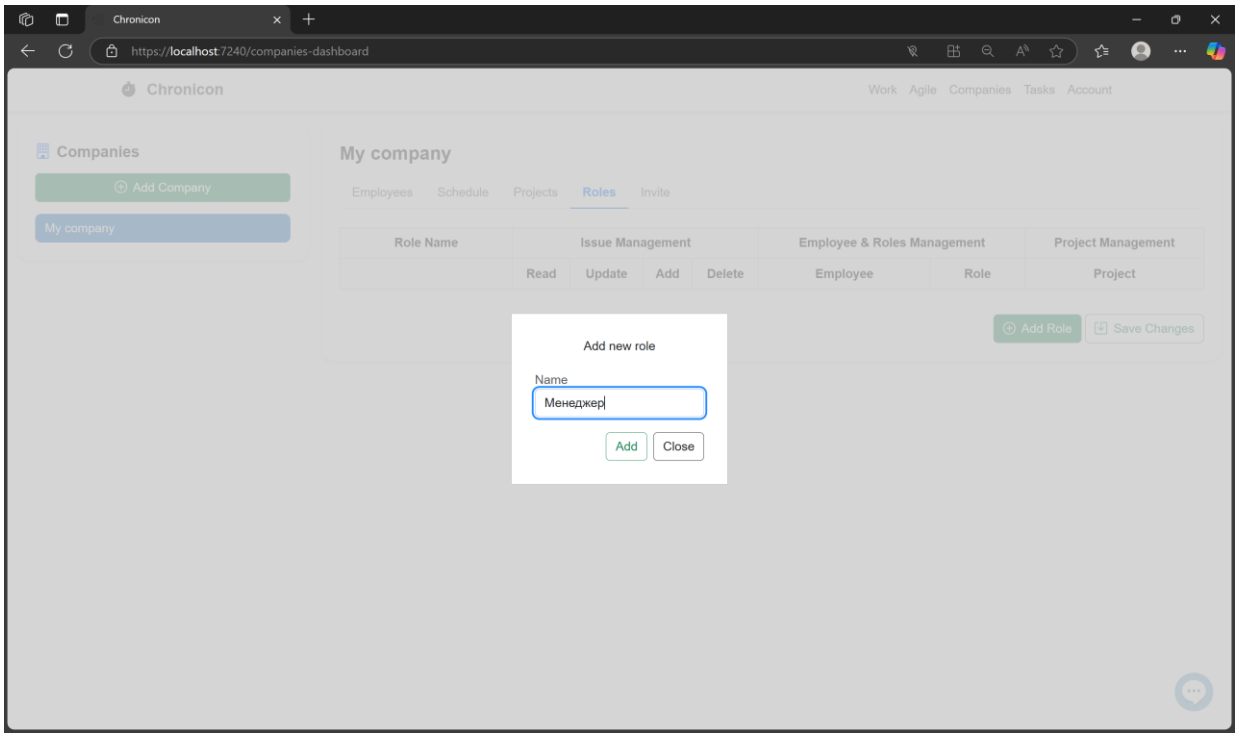


Рисунок Д.12 – Створення ролі.

Створена роль повинна відобразитись в таблиці що і відбулось успішно (див. рисунок Д.13).

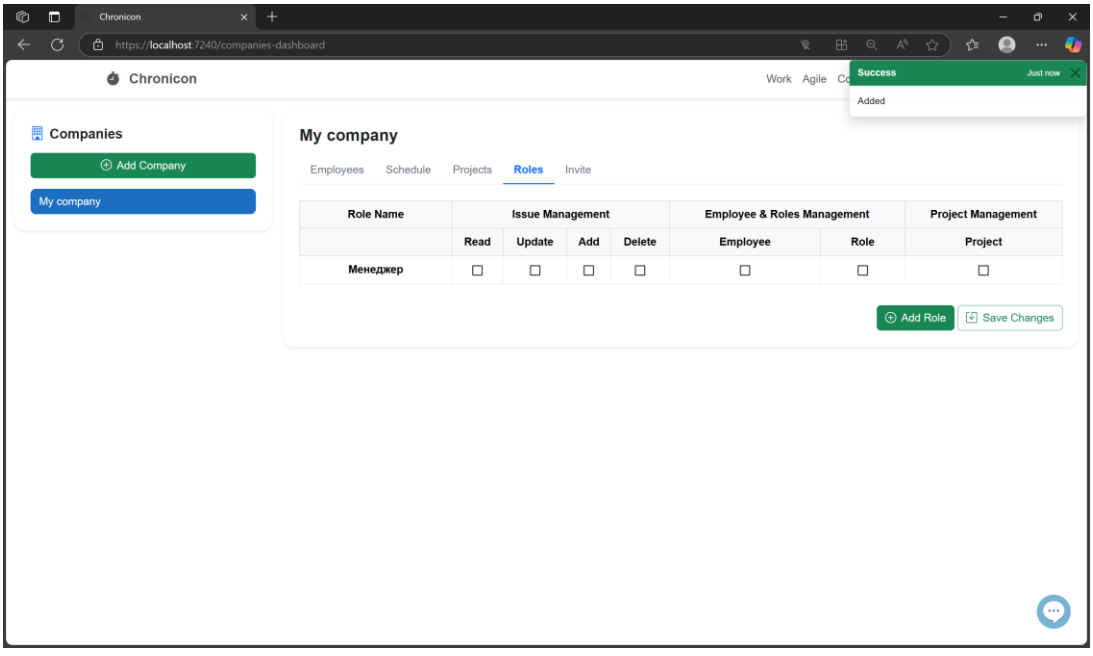


Рисунок Д.13 – Список ролей з новою створеною

Після створення нової ролі ми можемо назначити її нашому раніше створеному співробінник (див. рисунок Д.14)

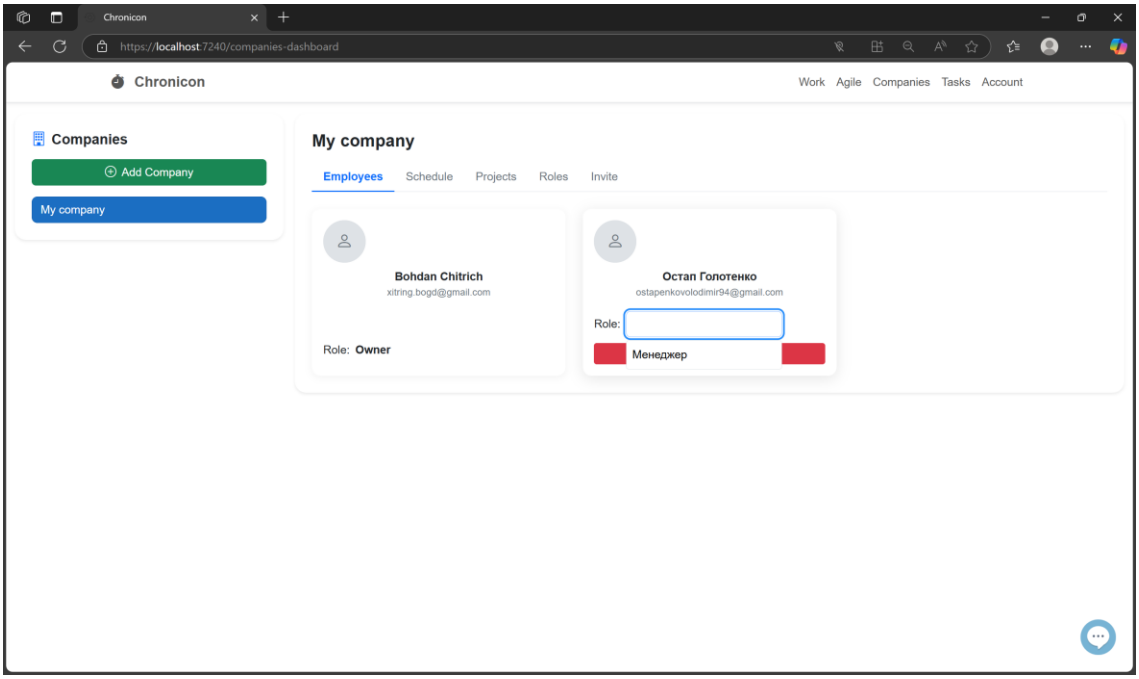


Рисунок Д.14 – Назначення ролі співробінику

Через те що роль була створена без прав з боку співробінника в нього є доступ лише до вкладки Employees для перегляду всіх співробінників що продемонстровано на рисунку Д.15

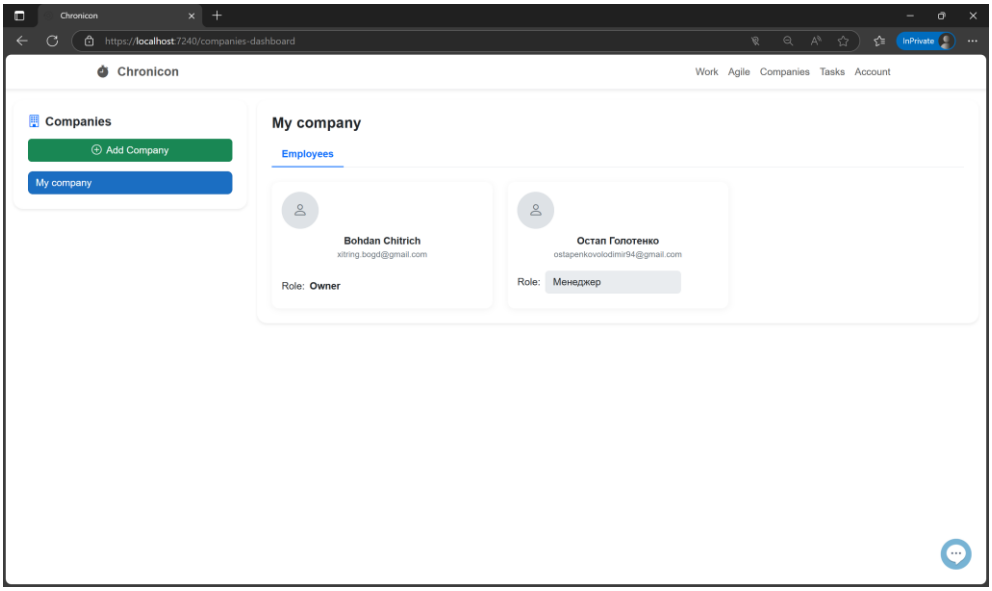


Рисунок Д.15 – Доступні закладки для співробінника

Міняєм роль добавивши права на редагування проектів. Після чого у співробіника появилась вкладка проектів (див. рисунок Д.16).

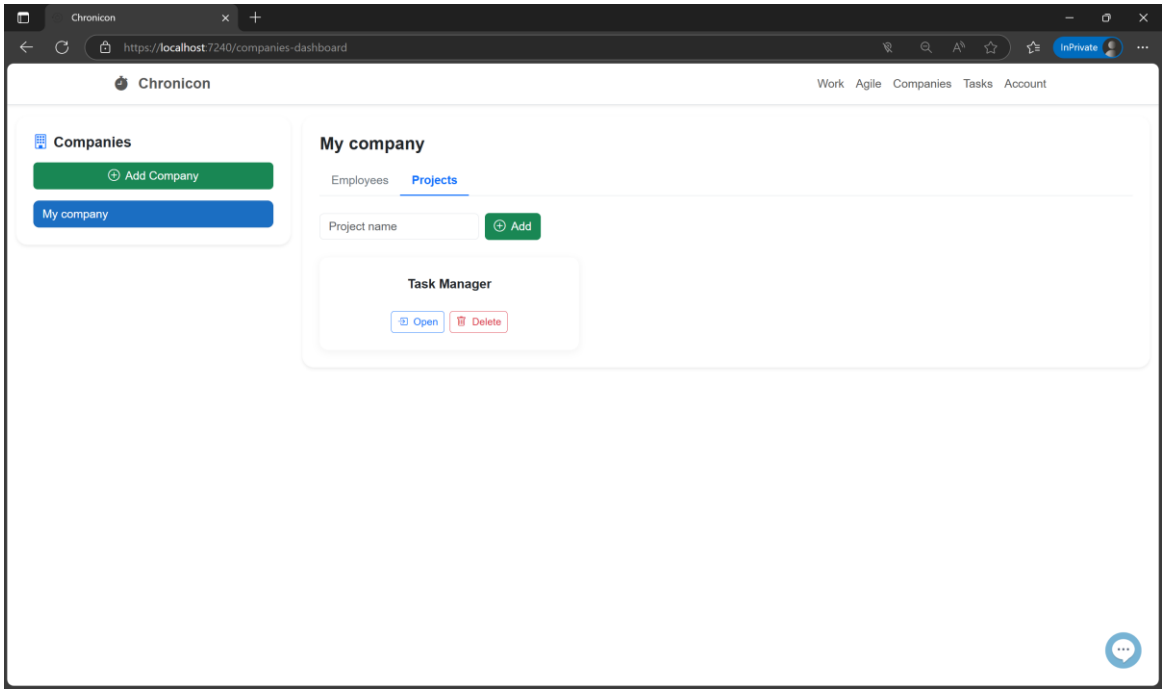


Рисунок Д.16 – Вкладка проектів

Це демонструє правильне функціонування системи ролей і прав в системі.

Спробуємо від імені співробітника створити новий проект введемо імя “Chronicon Project” і нажимаєм кнопку Add Після чого новий проект відобразиться у вигляді карточки в загальному списку що зображено на рисунку Д.17.

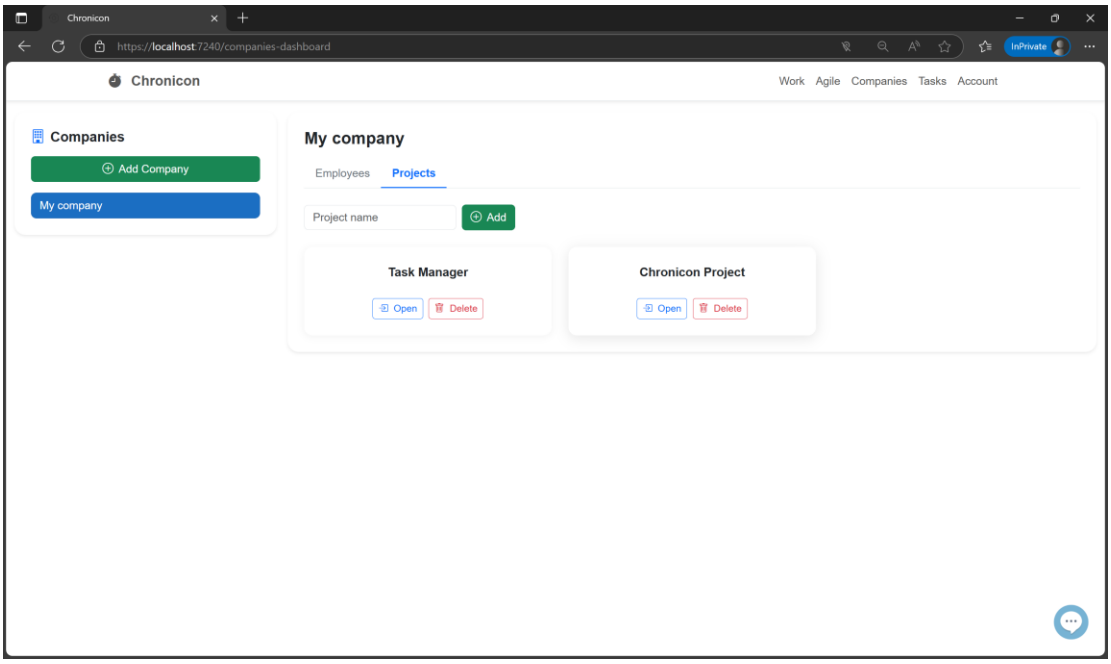


Рисунок Д.17 – Створений проект

Всі тести даного модуля були пройдені успішно що демонструє його повну функціональність.Після створення компанії і її налаштування перейдемо до тестування модулів задач і комунікаії.І першим тестом буде створення задачі і назначення фіксера.Переходимо на сторінку /create-issue (див. рисунок Д.18).

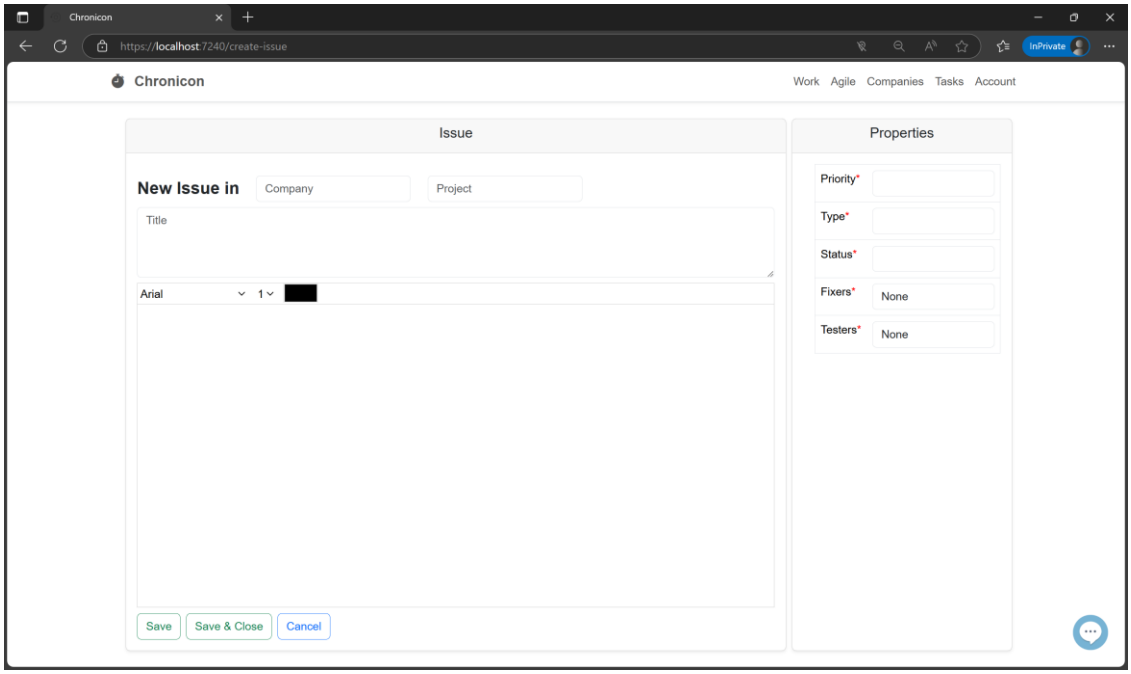


Рисунок Д.18 – Вигляд сторінки створення задачі

Заповнюєм відповідні поля і вибираєм фіксера співробінника а тестера власника (див. рисунок Д.19).

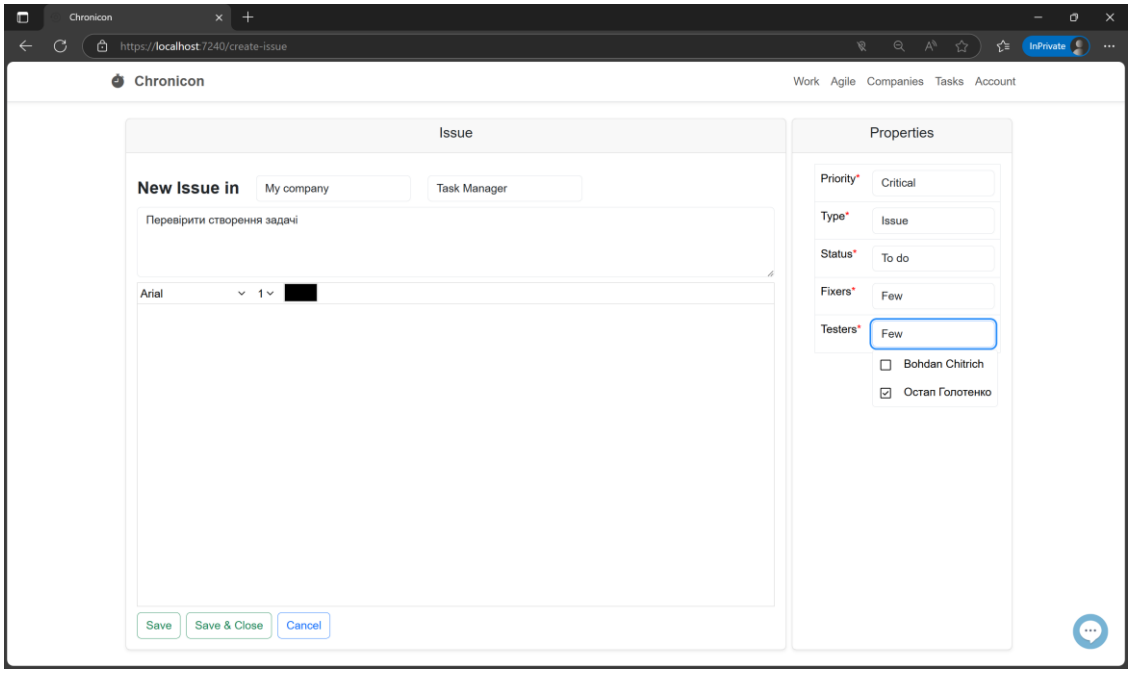


Рисунок Д.19 – Заповнені поля задачі

Нажимаємо кнопку Save після чого задача буде збережена а збоку відкриється чат даної задачі (див. рисунок Д.20).

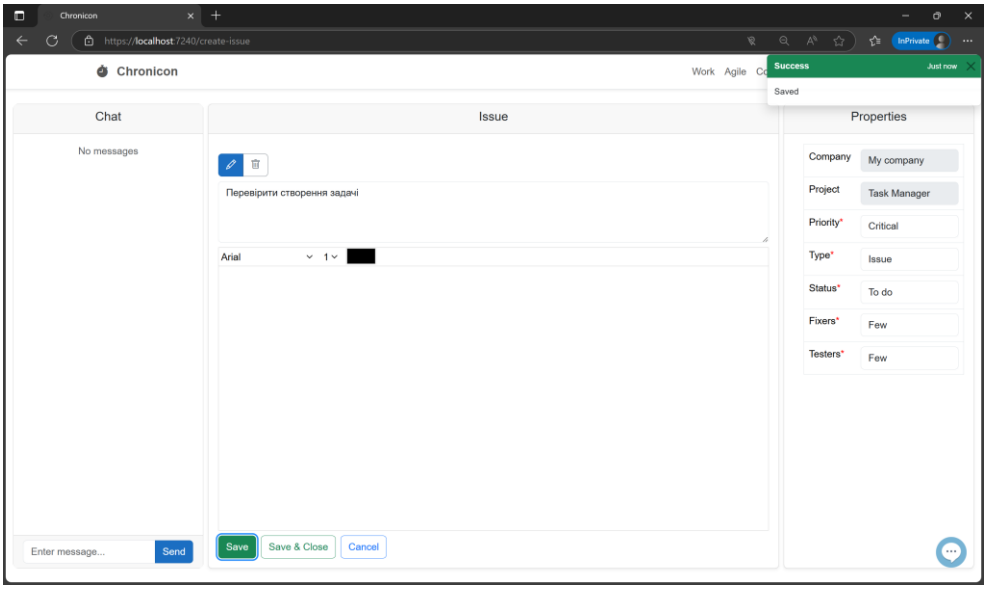


Рисунок Д.20 – Створена задача і відкритий чат

Тепер від імені тестера переходимо на екран /agile і там успішно створюємо новий пресет і виставляємо фільтри як продемонстровано на рисунку Д.21.

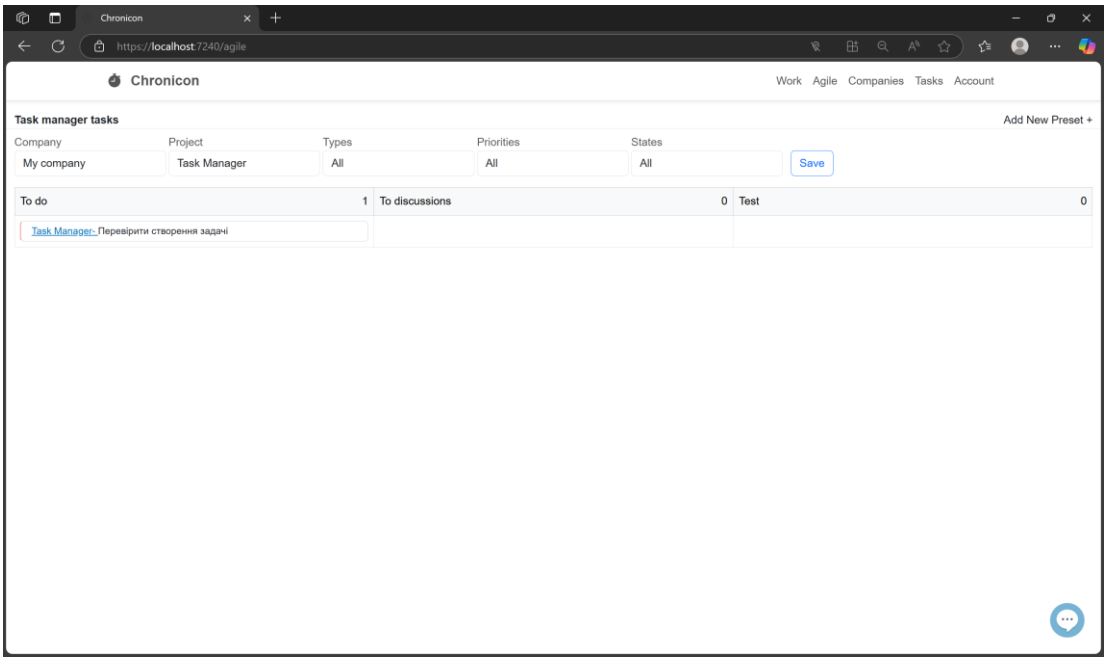


Рисунок Д.21 – Сторінка /agile з створеним пересетом

Серед списку задач, показана створена нами раніше, клікаємо на неї і переходимо на екран перегляду (див. рисунок Д.22).

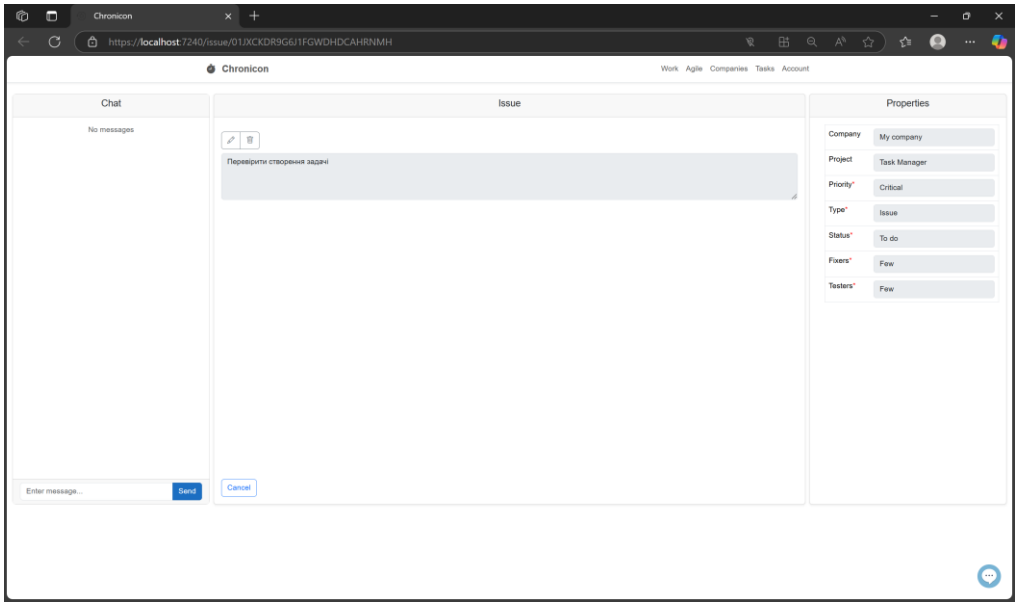


Рисунок Д.22 – Відкрита задача в режимі перегляду

Тепер спробуємо відправити повідомлення в чат (див. рисунок Д.23).

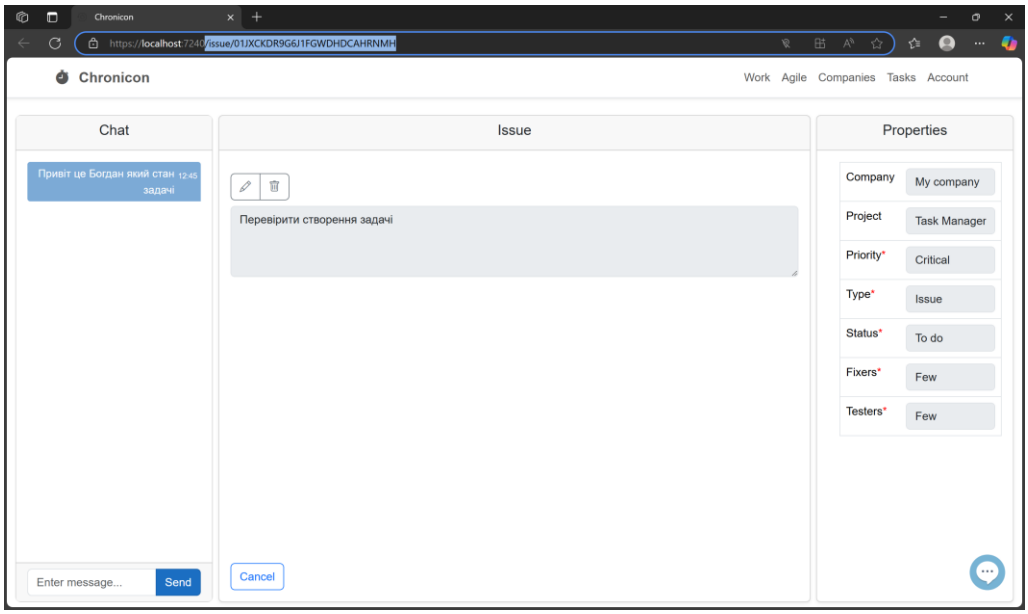
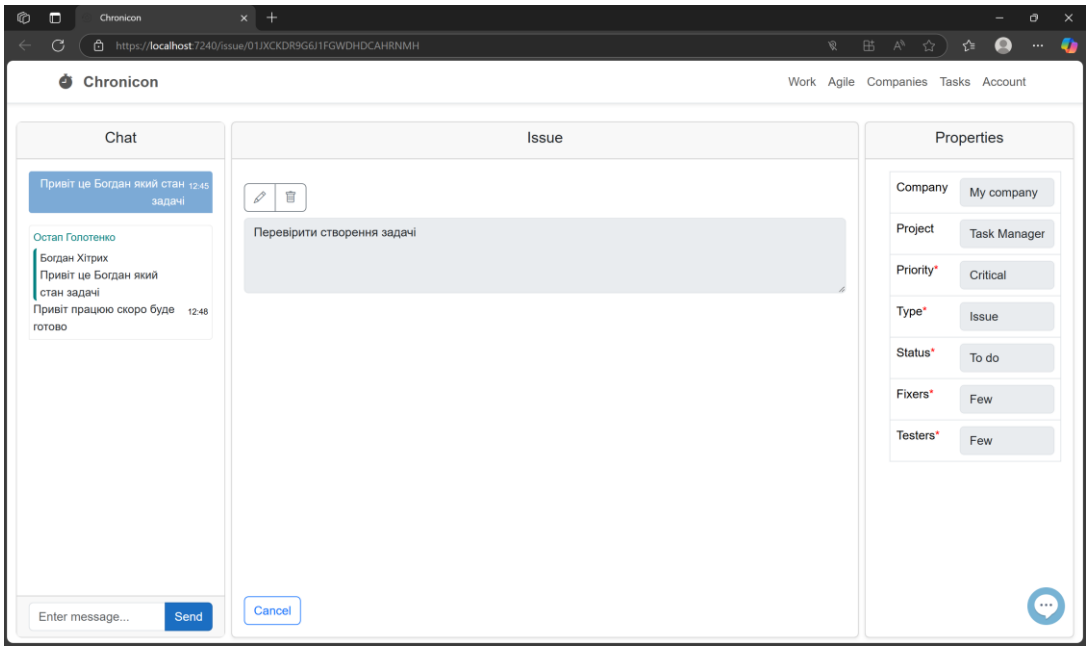


Рисунок Д.23 – Відправлене повідомлення

Після цього відповідаємо на повідомлення й отримуємо відповідь (див. рисунок Д.24).



Рисукно Д.24 – Отримана відповідь

Дані тести продемонстрували нам, що модуль задач і комунікації працюють так як це було заплановано.

Наступним етапом є перевірка таймера для цього спочатку потрібно вказати робочі години для співробітника. Переходимо на dashboard і встановлюєм робочі години (див. рисунок Д.25).

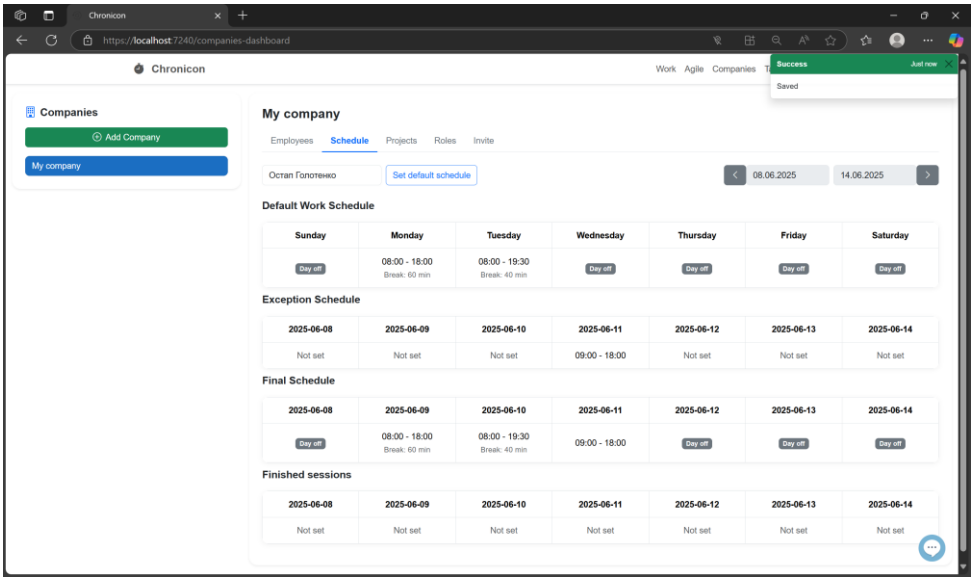


Рисунок Д.25 – Робочі години співробіника

Після цього співробітник може використовувати таймер на сторінці /work-schedule. На рисунку Д.26 зображено запущений таймер.

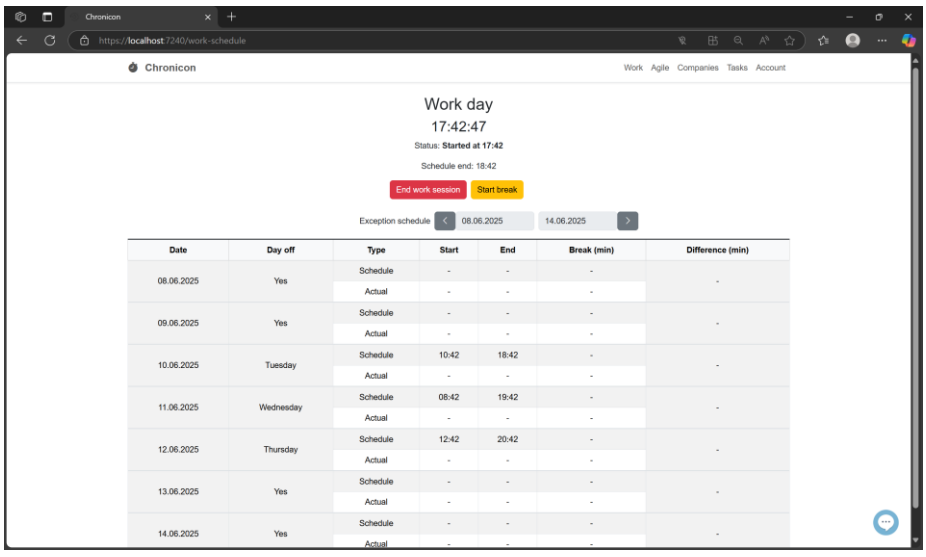
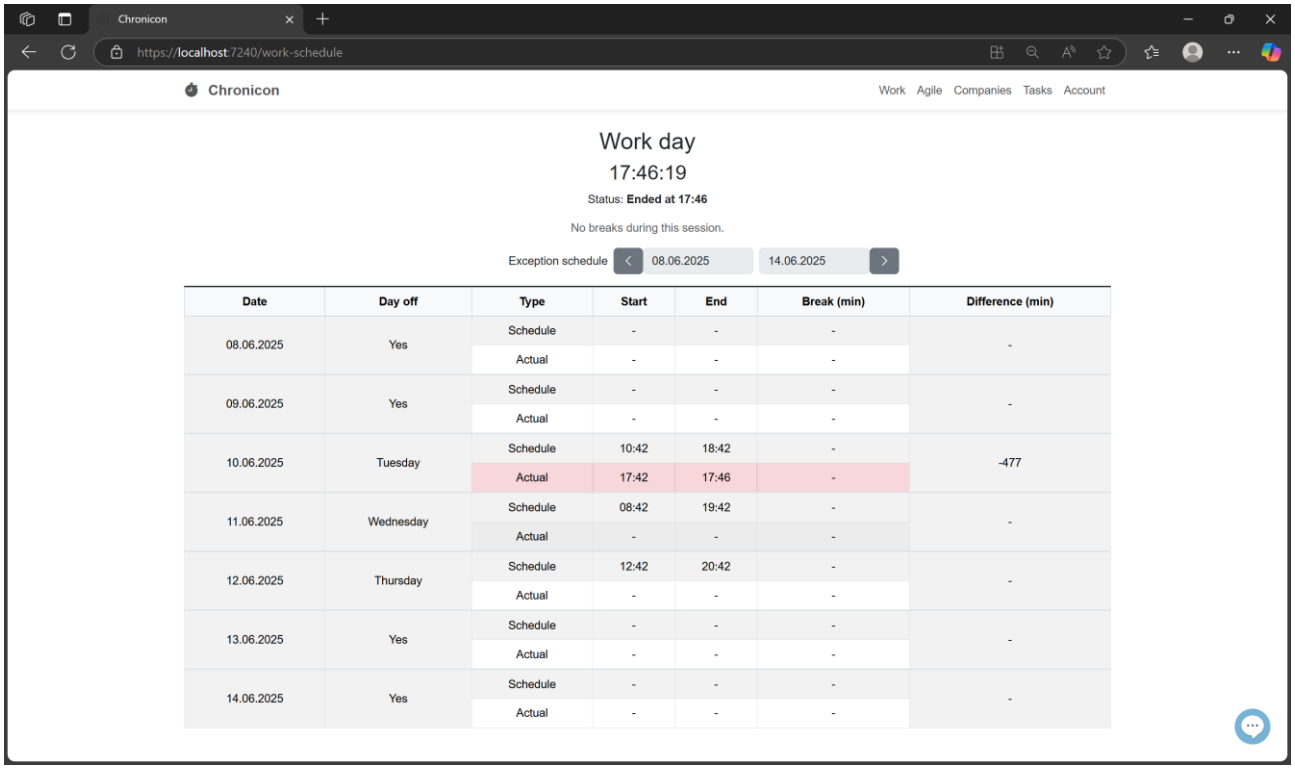


Рисунок Д.25 – Розпочатий таймер

Зупиняєм таймер і після цього користувачу показується які години він працював і статистику по робочим хвилина на рисунку Д.26



Date	Day off	Type	Start	End	Break (min)	Difference (min)
08.06.2025	Yes	Schedule	-	-	-	-
		Actual	-	-	-	
09.06.2025	Yes	Schedule	-	-	-	-
		Actual	-	-	-	
10.06.2025	Tuesday	Schedule	10:42	18:42	-	-477
		Actual	17:42	17:46	-	
11.06.2025	Wednesday	Schedule	08:42	19:42	-	-
		Actual	-	-	-	
12.06.2025	Thursday	Schedule	12:42	20:42	-	-
		Actual	-	-	-	
13.06.2025	Yes	Schedule	-	-	-	-
		Actual	-	-	-	
14.06.2025	Yes	Schedule	-	-	-	-
		Actual	-	-	-	

Рисунок Д.26 – Таблиця робочих годин

Тести даного модуля показали що він функціонує так як і проектувалось. Останім модулем для тестування є модуль фрилансера.Розпочнемо тест з створення акаунта після чого повинна початись його індексація в Typesense (див. рисунок Д.27).

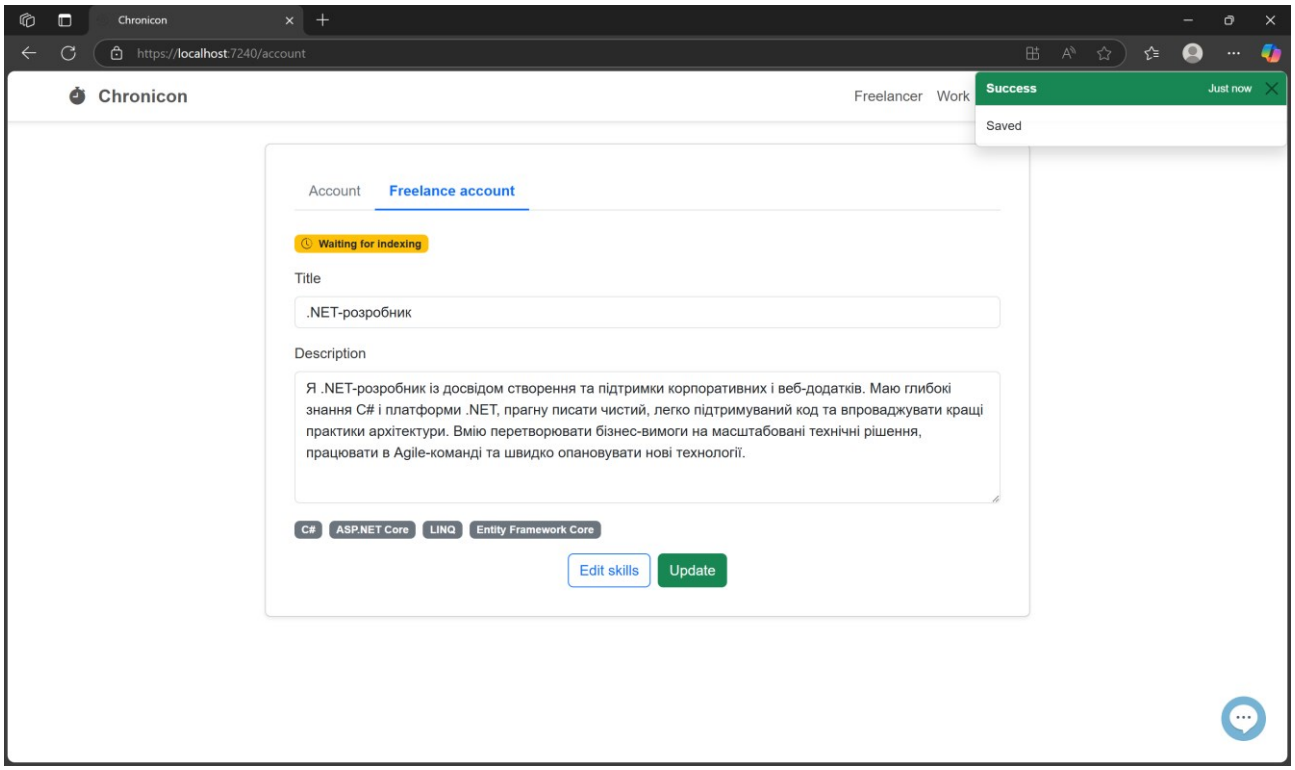


Рисунок Д.27 – Створений акаунт фрилансера

Після індексації від імені компанії переходим на сторінку пошуку фрилансерів(див. рисунок Д.28). І виконуєм пошук за ключовим словом C# .

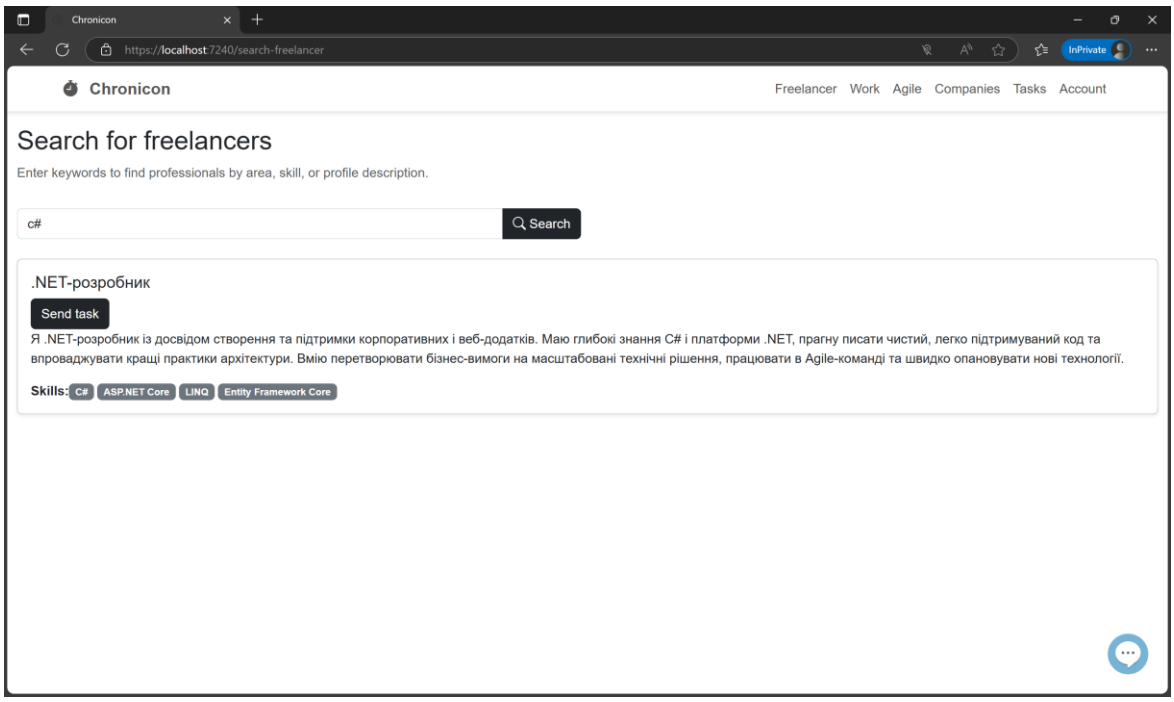


Рисунок Д.28 – Пошук фрилансерів

Для точної перевірки також вводим випадкові символи в пошук щоб перевірити пошуковий рушій .Як результат не отримуєм жодного результату що зображено на рисунку Д.29.

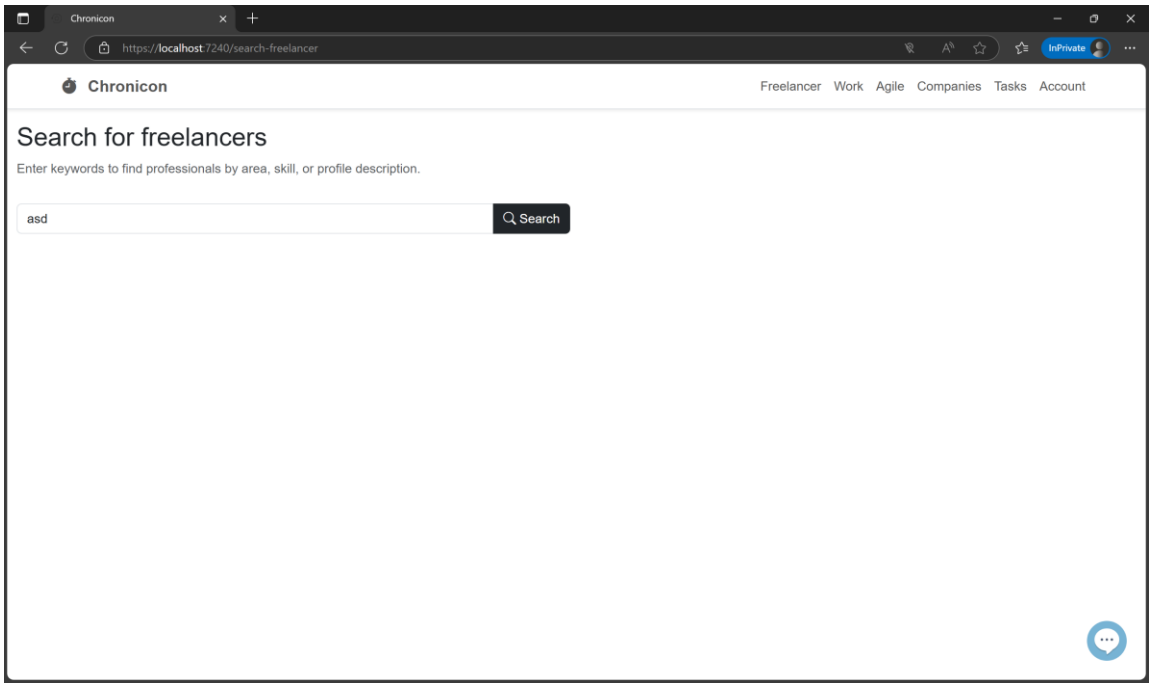


Рисунок Д.29 – Результат пошуку з випадковим пошуком

Створюєм задачу і надсилаєм фрилансеру на розгляд(див. рисунок Д.30).

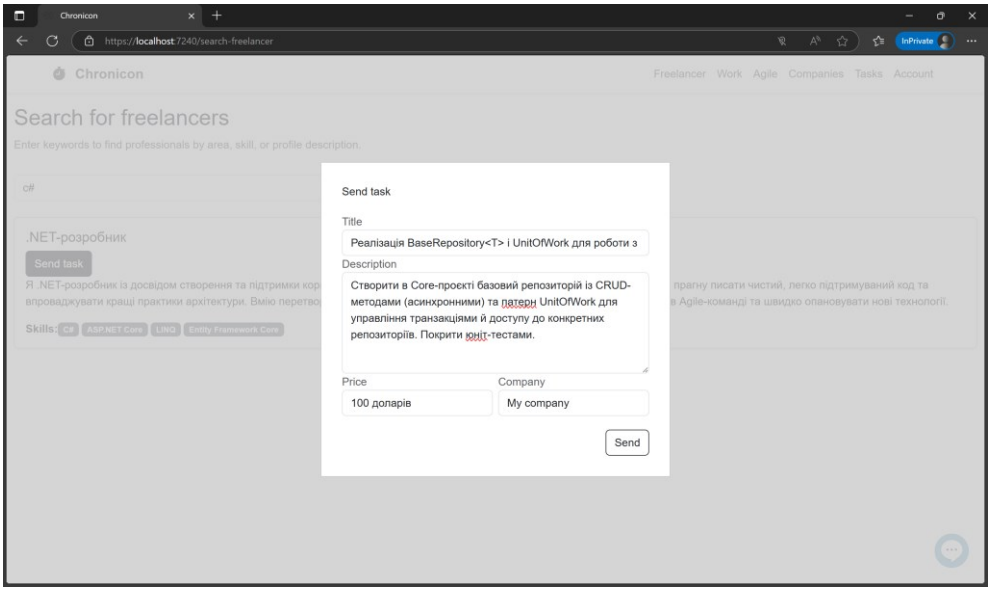


Рисунок Д.30 – Надсилання задачі фрилансеру

Після надсилання задачі вона відображається у всхідних задачах (див. рисунок Д.31).

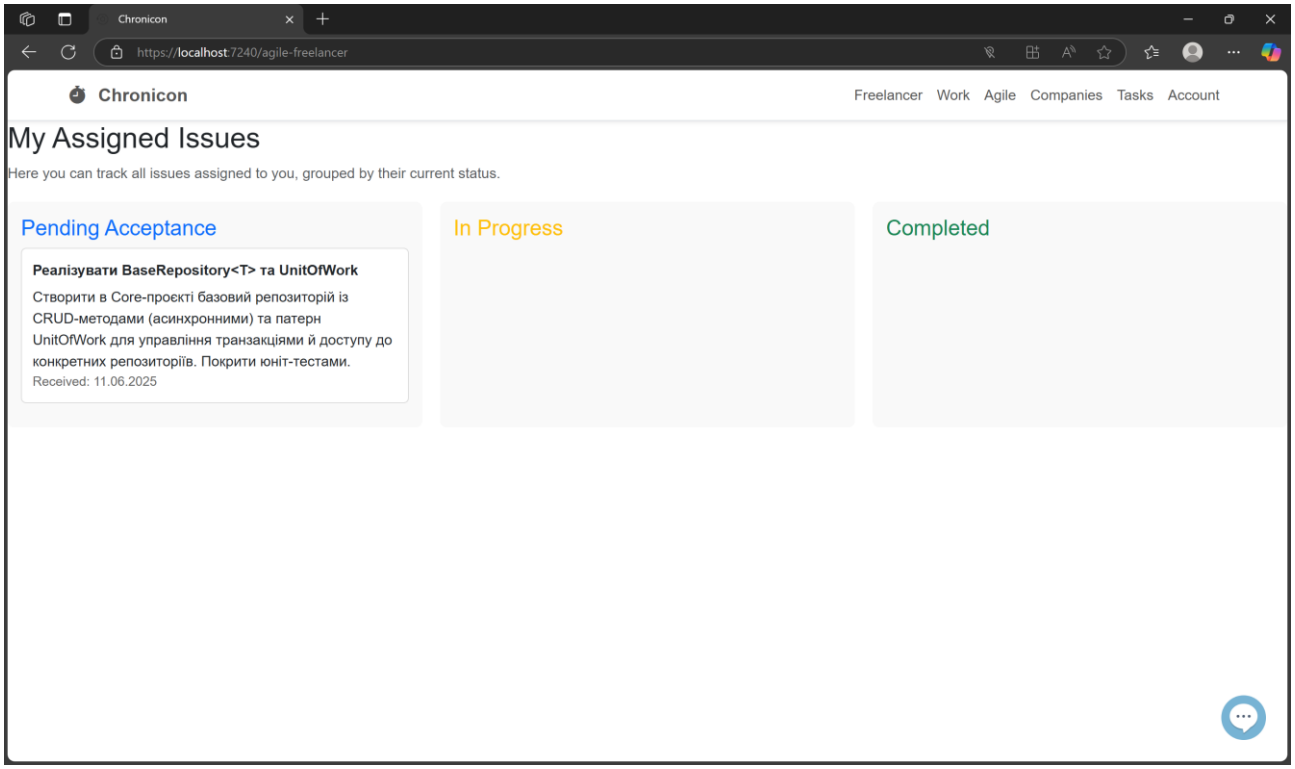


Рисунок Д.31 – Задачі у вхідних

Усі тести були успішно пройдені, що свідчить про стабільну та правильну роботу нашої системи. Це означає, що всі ключові функції виконуються відповідно до очікувань, без помилок або збоїв, що підтверджує її готовність до подальшого використання або впровадження в реальному середовищі.