



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

**«Інформаційна система управління рухом громадського транспорту
(Індивідуальний проект: управління розкладом)»**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б.	Група
1	Карлінський Владислав Юрійович	КН-П-213

Автор звіту

_____ Карлінський Владислав Юрійович

Керівник	Доцент кафедри ІТ та ФП, к.т.н., доцент Краковецький О.Ю.
----------	--

АНОТАЦІЯ

УДК 004.9

Д-30

Карлінський В. Ю. Інформаційна система управління рухом громадського транспорту (Індивідуальний проект: управління розкладом) : випускна кваліфікаційна робота на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». – Одеса: ОТУШ, 2025. – 25 с.

Робота присвячена розробці та впровадженню інформаційної системи **Timetracker** – веб-додатку для автоматизації обліку робочого часу персоналу, управління відпустками, погодженням табелів та аналітики продуктивності у великій компанії.

Актуальність теми зумовлена потребою у цифровізації HR-процесів, зокрема автоматизації заповнення табелів, зручного погодження запитів на відпустку, а також централізованого управління проектами, робочими кодами та контрактними умовами.

Об'єкт дослідження: алгоритми та технології управління робочими процесами показниками персоналу у корпоративному середовищі.

Предмет дослідження: система для обліку часу та адміністрування робочих процесів на прикладі веб-застосунку Timetracker.

Завдання дослідження:

- проаналізувати існуючі рішення для обліку часу та управління відпустками;
- визначити функціональні та нефункціональні вимоги до системи;
- вивчити методи автоматизації нагадувань та погодження запитів у системах обліку;
- розробити інформаційну модель системи та побудувати її логічну архітектуру;

- здійснити тестування прототипу системи з метою оцінки її зручності, точності та продуктивності.

Методи дослідження: аналіз предметної області, системне моделювання, порівняльний аналіз існуючих систем, розробка та тестування програмного забезпечення, моделювання сценаріїв використання та оцінка ефективності реалізованих рішень.

Зміст

Зміст	3
ВСТУП	4
ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ	5
1. Загальний опис проекту	5
2. Призначення веб-застосунку	5
3. Технологічний стек	6
4. Основні модулі та їх функціональність	8
4.1. Управління та редагування табелів	8
4.2. Управління пристроями	9
4.3. Управління проектами	9
4.4. Управління запитами	10
4.5. Управління контрактами користувачів	10
4.6. Управління звітністю	11
РОЗДІЛ 1	12
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ	12
РОЗДІЛ 2	15
РЕАЛІЗАЦІЯ МЕХАНІЗМІВ АДМІНІСТРАТИВНОГО ВЕБ-ЗАСТОСУНКУ	15
РОЗДІЛ 3	24
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ	24
ВИСНОВКИ	25
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	26

ВСТУП

Стрімкий прогрес інформаційних технологій активно впливає на цифрову трансформацію різних аспектів діяльності, зокрема на систему управління персоналом у великих компаніях. Згідно зі Стратегією розвитку інформаційного суспільства в Україні, ІТ є невід’ємним елементом соціально-економічного поступу та одним із головних чинників інновацій у сфері економіки. У цьому контексті особливе значення набувають рішення, що автоматизують процеси обліку трудової діяльності, включаючи моніторинг робочого часу та організацію відпусток.

Дослідження присвячено розробці та впровадженню системи обліку ефективності роботи персоналу в межах проекту Timetracker. Цей продукт вирішує ключові завдання щодо автоматизації ведення табелів робочого часу і обробки заявок на відпустки — критично важливих процесів для великих організацій. Програмне рішення реалізовано у вигляді веб-додатку, який дозволяє ефективно керувати робочими графіками, контрактами та запитами на відпустки за допомогою адміністративної панелі.

У процесі розробки застосовано сучасні технології: клієнтська частина реалізована з використанням Angular 19, серверна — на платформі .NET із мовою програмування C#. Для хостингу, CI/CD та зберігання даних залучено сервіси Azure DevOps та Azure Blob Storage, що забезпечують високу продуктивність, надійність і масштабованість.

У наступних розділах описано технічні вимоги до системи, її архітектуру та ключові компоненти. Розроблений веб-додаток включає бекенд для обробки логіки та даних, інтуїтивно зрозумілий інтерфейс для кінцевих користувачів і менеджерів, а також інтеграцію з Azure AD для управління доступом та користувацькими групами. Програмне забезпечення розгорнуто в хмарному середовищі Azure, що дозволяє досягти високої доступності та економічної ефективності.

У підсумку наведено результати реалізації проєкту та зроблено висновки щодо ролі кожного з учасників. Важливість розробки Timetracker полягає в її здатності вдосконалити управління персоналом, оптимізувати бізнес-процеси та підтримати загальну цифровізацію організаційної діяльності.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

1. Загальний опис проєкту

Проект Timetracker розроблено з метою створення інструменту для автоматизованого контролю за трудовою активністю персоналу в масштабних організаційних структурах. Головне завдання полягає в усуненні потреби в ручному заповненні табелів обліку робочого часу, у спрощенні процесу управління відпустками та координації змін у графіках, що зазвичай вимагає значних ресурсів і часто супроводжується помилками. Рішення орієнтоване на покращення ефективності HR-процесів шляхом впровадження автоматизованого обліку, прозорості обробки запитів на відпустку та надання зручного інструментарію для керівників.

Розроблений веб-додаток охоплює основні елементи керування персоналом, включаючи внесення даних до табелів, облік запитів на відпустки, управління проєктною діяльністю та робочим розкладом, а також можливість перегляду аналітичних даних. Інтерфейс системи побудований на базі Angular у поєднанні з Angular Material, що дозволяє створити зручне, гнучке та адаптивне середовище для користувачів усіх рівнів.

2. Призначення адміністративного веб-застосунку

Адміністративний інтерфейс веб-додатку Timetracker виступає центральним засобом керування системою моніторингу показників трудової діяльності персоналу. Цей інструмент надає можливість користувачам, керівникам

підрозділів і адміністраторам вищого рівня здійснювати створення, перегляд, редагування та видалення об'єктів, пов'язаних із організацією роботи, а також аналізувати результативність діяльності співробітників.

Ключовий функціонал порталу розподілено за категоріями користувачів:

Для працівників: внесення даних у табелі з прив'язкою до відповідних проєктів, створення заяв на оплачувані й неоплачувані відпустки;

Для менеджерів: перевірка, підтвердження або відхилення табелів і запитів на відпустку, а також отримання сповіщень про завдання, що залишилися невиконаними;

Для адміністраторів: конфігурація робочих розкладів, умов трудових договорів, доступ до зведеної статистики та аналітики по всій системі.

Система також реалізує механізм автоматичних нагадувань через push-нотифікації або електронну пошту, що сприяє своєчасному реагуванню на події. Крім того, передбачена можливість експорту інформації у форматі Excel для формування звітів.

3. Технологічний стек

Проект Timetracker реалізовано з використанням сучасного набору технологій, які забезпечують високу ефективність роботи системи, її масштабованість, а також спрощують розробку, впровадження та подальше обслуговування. Нижче представлено огляд технологічного стеку, що охоплює клієнтську й серверну частини, базу даних, хмарну інфраструктуру та інструменти розгортання.

Клієнтська частина:

- **Angular 19** — головний фреймворк, що використовується для побудови динамічного, модульного інтерфейсу. Завдяки оптимізованому механізму рендерингу, зменшеним розмірам збірок і потужному CLI, Angular

забезпечує комфортну та швидку розробку інтерфейсів навіть під складні сценарії.

- **Angular Material** — набір готових інтерфейсних елементів, побудованих згідно з принципами Material Design. Дає змогу створити адаптивне, сучасне та естетично привабливе UI.
- **RxJS** — бібліотека для роботи з асинхронними потоками, активно використовується для обробки подій у додатку, таких як HTTP-запити чи реакції на дії користувача.
- **Luxon** — інструмент для зручної роботи з датами й часом, що дозволяє легко реалізовувати форматування, перетворення часових зон та розрахунок тривалостей.
- **ExcelJS** — бібліотека, що дає змогу створювати та зчитувати Excel-документи прямо на фронтенді, що є зручним для експорту табелів і аналітики.
- **Cashew** — проста, але потужна бібліотека для управління станом в Angular-додатку, яка надає функціональність, подібну до Redux, але з легким API і підтримкою імутабельності.

Серверна частина:

- **ASP.NET Core** — продуктивний кросплатформовий фреймворк для побудови REST API. Завдяки підтримці middleware, вбудованій ін'єкції залежностей та оптимізації для Docker і Azure, платформа забезпечує надійну архітектуру.
- **Azure Functions** — серверлес-архітектура для виконання фонових процесів, таких як надсилання нагадувань і сповіщень. Забезпечує автоматичне масштабування та розрахунок вартості за фактичне виконання.
- **Entity Framework Core** — ORM, що полегшує взаємодію з реляційною базою даних (SQL Server). Дозволяє здійснювати запити за допомогою LINQ та керувати змінами схеми через міграції.

- **AutoMapper** — інструмент для спрощення мапінгу між доменними сутностями та DTO, що суттєво зменшує обсяг шаблонного коду.
- **FluentValidation** — бібліотека для декларативної перевірки вхідних даних, яка допомагає централізовано задавати валідаційні правила з чіткими повідомленнями про помилки.

Сховище даних:

- **Azure Blob Storage** — сервіс для зберігання неструктурованих даних, таких як лог-файли чи звіти. Його переваги — масштабованість, економічність та висока доступність.
- **SQL Server** — основна реляційна база даних, через інтеграцію з EF Core зберігає інформацію про користувачів, таблиці, проекти та запити на відпустки.

Інструменти розгортання та підтримки:

- **Azure DevOps** — платформа, що охоплює весь цикл DevOps: Git-репозиторії, планування завдань, автоматичну збірку, тестування та деплой у різні середовища.
- **Docker** — технологія контейнеризації, що гарантує ідентичні умови для локальної та хмарної інфраструктури.
- **Azure Active Directory (AD)** — рішення для централізованої автентифікації та керування правами доступу, яке дозволяє налаштувати гнучку політику доступу до ресурсів системи.

Застосування цього стеку дозволило створити гнучку, надійну та масштабовану систему **Timetracker**, здатну легко інтегруватися в існуючу інфраструктуру підприємства та підтримувати зростаючі вимоги до автоматизації управління персоналом.

4. Основні модулі та їх функціональність

Система Timetracker структурована у вигляді окремих логічних модулів, кожен з яких виконує чітко визначену роль у процесі моніторингу трудової активності співробітників. Кожен модуль відповідає за окрему функціональну область і реалізує специфічний набір операцій. Далі представлено детальний опис кожного з модулів, їхнє функціональне призначення та перелік підтримуваних дій.

4.1. Управління та редагування табелів (Активність)

Призначення:

Цей модуль дозволяє користувачам заповнювати та редагувати табелі робочого часу, які є основою для розрахунку заробітної плати та відстеження активності на проєктах. Модуль забезпечує автоматизацію процесу табелювання та інтеграцію з facturation code для точного обліку.

Функціональність:

- **Створення:** Заповнення табеля за місяць із зазначенням дат, кількості робочих годин та коментарів (за потреби).
- **Перегляд:** Відображення списку табелів користувача або детальної інформації про конкретний табель.
- **Оновлення:** Редагування записів у табелі, наприклад, зміна годин або додавання коментарів.
- **Статуси:** Управління статусами табеля (Saved, Submitted, Approved, Rejected) із можливістю відправки на перевірку менеджера.
- **Пошук:** Швидкий пошук табелів за проєктом, місяцем або facturation code.

4.2. Управління пристроями

Призначення:

Модуль відповідає за облік та управління пристроями, які видаються або повертаються працівниками. Це забезпечує прозорість і контроль за обладнанням компанії.

Функціональність:

- **Створення:** Формування запису про видачу пристрою працівнику з вказівкою характеристик (наприклад, тип, серійний номер).
- **Перегляд:** Перегляд списку всіх пристроїв або деталей конкретного пристрою, включаючи історію видачі/повернення.

- **Оновлення:** Редагування даних про пристрій, наприклад, зміна статусу чи власника.
- **Видалення:** Видалення запису про пристрій, який більше не використовується.
- **Пошук:** Пошук пристроїв за назвою, серійним номером або працівником.

4.3. Управління проєктами

Призначення:

Модуль дозволяє створювати та управляти проєктами, до яких прив'язуються таблиці та facturation code. Проєкти є основою для організації робочих процесів і обліку активності.

Функціональність:

- **Створення:** Додавання нового проєкту з назвою, клієнтом, датами початку та завершення.
- **Перегляд:** Перегляд списку всіх проєктів або деталей конкретного проєкту, включаючи призначених менеджерів і facturation code.
- **Оновлення:** Редагування даних проєкту, наприклад, зміна статусу (активний/архівний) або додавання менеджерів.
- **Видалення:** Видалення проєкту, який більше не актуальний.
- **Пошук:** Швидкий пошук проєктів за назвою, клієнтом або датами.

4.4. Управління запитами

Призначення:

Модуль забезпечує створення, обробку та управління запитами на відпустки, дозволяючи користувачам подавати запити, а менеджерам — схвалювати або відхиляти їх.

Функціональність:

- **Створення:** Подання запиту на відпустку з вибором категорії (оплачувана/неоплачувана) та дат.
- **Перегляд:** Перегляд списку всіх запитів або деталей конкретного запиту, включаючи статус (Approved/Rejected).
- **Оновлення:** Редагування запиту до його обробки менеджером.
- **Схвалення/Відхилення:** Можливість для менеджерів схвалювати або відхиляти запити з коментарями.
- **Пошук:** Пошук запитів за працівником, датами або статусом.

4.5. Управління контрактами користувачів

Призначення:

Модуль дозволяє глобальним адміністраторам налаштовувати умови контрактів користувачів, включаючи робочі графіки, групи доступу та інтеграцію з Azure AD.

Функціональність:

- **Створення:** Визначення нового контракту з умовами, такими як робочі години чи тип оплати.
- **Перегляд:** Перегляд списку всіх контрактів або деталей конкретного контракту.
- **Оновлення:** Редагування умов контракту, наприклад, зміна графіку чи груп доступу.
- **Видалення:** Видалення контракту, який більше не актуальний.
- **Інтеграція:** Налаштування груп доступу через Azure AD для автоматичного управління ролями.

4.6. Управління звітністю

Призначення:

Модуль відповідає за генерацію та експорт звітів, що відображають статистику роботи персоналу, заповнення табелів і використання відпусток. Звіти є основою для аналізу ефективності.

Функціональність:

- **Створення:** Генерація звіту на основі даних про табелі, відпустки чи проекти.
- **Перегляд:** Відображення звіту у веб-інтерфейсі з графіками та статистикою.
- **Експорт:** Експорт звіту у форматі Excel (.xlsx) для подальшого використання.
- **Фільтрація:** Налаштування параметрів звіту, наприклад, за періодом, проектом чи працівником.
- **Пошук:** Швидкий доступ до раніше створених звітів за назвою чи датою.

РОЗДІЛ 1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Відповідно до технічної документації, архітектура системи Timetracker побудована на основі трьох ключових компонентів:

- адміністративного веб-інтерфейсу,
- серверної частини (веб-сервісу API),
- сховища даних (що включає реляційну базу та хмарне файлове сховище).

Адміністративний веб-додаток розроблено для забезпечення зручного доступу до управління даними про персонал для різних типів користувачів — працівників, менеджерів та адміністраторів. Через цей інтерфейс можна виконувати базові CRUD-операції (створення, перегляд, редагування, видалення) з такими об'єктами, як табелі, запити на відпустки, контракти, пристрої та проекти. Клієнтська частина побудована за допомогою Angular 19 у поєднанні з Angular Material, що дозволяє створити гнучкий та адаптивний інтерфейс. Dodatok інтегрований із Azure Active Directory для централізованого управління ролями й доступами. Взаємодія з бекендом відбувається через REST API, що забезпечує швидкий обмін даними. Серед основних технологій, які використовуються на фронтенді: Angular, TypeScript, RxJS, Angular Material, Luxon, ExcelJS, Cashew, HTML та CSS.

Серверна частина виконує обробку запитів, що надходять із клієнтської частини, реалізовує бізнес-логіку системи та взаємодіє зі сховищем даних. Вона відповідає за стабільне виконання операцій додавання, оновлення, пошуку та видалення даних, зокрема щодо табелів, відпусток і звітності. Основою серверної частини є фреймворк ASP.NET Core, який забезпечує високу продуктивність і гнучкість у розробці. Для обробки фонових процесів, таких як надсилання сповіщень, використовуються Azure Functions. Mapінг між DTO та доменними моделями реалізовано через AutoMapper, а перевірку вхідних даних — за допомогою FluentValidation. Сервіс розгорнуто у хмарному середовищі Azure, що гарантує масштабованість і доступність. Задіяні технології: ASP.NET Core, C#, Azure Functions, AutoMapper, FluentValidation, Entity Framework Core.

Сховище даних включає два основних компоненти:

Azure Blob Storage, що використовується для зберігання нефайлової (неструктурованої) інформації — звітів, логів тощо,

SQL Server, що виступає як основна реляційна база даних для збереження ключових об'єктів системи — проєктів, заявок на відпустку, контрактів, табелів і пристроїв.

База даних ретельно нормалізована для забезпечення високої ефективності, узгодженості та збереження цілісності інформації. Робота з базою здійснюється через Entity Framework Core, який підтримує як LINQ-запити, так і систему міграцій. Цілісність забезпечується через первинні та зовнішні ключі, обмеження значень полів і правила каскадного оновлення або видалення записів.

Вся система функціонує у хмарному середовищі Azure. Фронтенд-додаток розміщується на Azure Static Web Apps, що дозволяє забезпечити швидке завантаження та зручне масштабування. Серверна частина працює в контейнерах, розгорнутих через Azure Container Apps. Розробка, тестування й розгортання автоматизовані за допомогою платформи Azure DevOps, яка забезпечує повний CI/CD-цикл — від керування репозиторіями в Git до створення пайплайнів і планування завдань у розділах Boards і Backlogs. Для уніфікації середовищ під час розробки використовувався Docker, що гарантує однакові умови для локальної і хмарної інфраструктури.

Користувачі мають доступ до системи відповідно до своєї ролі (співробітник, керівник або адміністратор), а автентифікація реалізована через Azure AD. Увесь код зберігається в Git-репозиторії, розміщеному в Azure DevOps.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ОСНОВНИХ МЕХАНІЗМІВ АДМІНІСТРАТИВНОГО ВЕБ-ЗАСТОСУНКУ

2.1. Аналіз предметної області та проектування доменних сутностей

На початковому етапі розробки адміністративного веб-застосунку системи Timetracker було проведено аналіз предметної області, пов'язаної з управлінням робочими показниками персоналу. Аналіз базувався на потребах великих організацій у автоматизації табелювання, обробки запитів на відпустки та управління проєктами. На основі цього виділено основні доменні сутності: **проєкти** (з назвою, клієнтом, датами початку та завершення), **facturation code** (billable або non-billable коди, що пов'язують користувача з проєктом), **табелі** (масиви записів із датами, годинами та коментарями), **запити на відпустки** (оплачувані або неоплачувані) та **пристрої** (обладнання, що видається працівникам). Для кожної сутності визначено основні поля, їх типи та обмеження, що забезпечують цілісність і коректність даних.

На серверній частині за допомогою **Entity Framework Core (EF Core)** налаштовано збереження даних із валідацією значень. Наприклад, для табелів забезпечено збереження масиву записів із датами, годинами та коментарями, прив'язаними до facturation code, а для пристроїв — унікальних ідентифікаторів і статусів видачі/повернення. EF Core використано для створення моделей даних із підтримкою міграцій, що спрощує оновлення структури бази даних SQL Server. Для ефективного мапінгу між доменними сутностями та DTO (Data Transfer Objects) використано бібліотеку **AutoMapper**, яка забезпечує автоматичне відображення об'єктів для операцій створення, отримання, оновлення та видалення (CRUD). Конфігурації AutoMapper налаштовано для підтримки всіх CRUD-операцій: **Get** (отримання списку сутностей із фільтрацією та пагінацією), **Autocomplete** (спрощені моделі для автодоповнення, наприклад, вибір проєкту чи facturation code), **GetId** (отримання детальної інформації за ідентифікатором), **Create** та **Update** (створення та оновлення сутностей).

Для забезпечення коректності даних реалізовано валідацію DTO-моделей за допомогою **FluentValidation**. Наприклад, для facturation code валідація перевіряє унікальність коду та відповідність типу (billable/non-billable), а для запитів на відпустки — коректність дат і категорій. Додатково створено кастомні методи валідації для перевірки бізнес-логіки, такі як перевірка прав користувача на створення чи редагування табеля або запиту на відпустку. Ці методи інтегровано в серверну логіку для забезпечення безпеки та відповідності бізнес-вимогам системи.

2.2. Реалізація серверної логіки та CRUD-операцій

На серверній частині системи Timetracker використано паттерн Репозиторій та Unit of Work для чіткого розділення обов'язків і управління транзакціями. Репозиторії відповідають за доступ до даних конкретних сутностей (наприклад, табелів, запитів на відпустки чи проєктів), а Unit of Work забезпечує координацію транзакцій між кількома репозиторіями. Серверна логіка реалізована на платформі ASP.NET Core з використанням Azure Functions для обробки фонових задач (наприклад, надсилання сповіщень) і FluentValidation для валідації вхідних запитів. AutoMapper застосовується для спрощення мапінгу даних між доменними моделями та DTO. Наприклад, ендпоінт для створення табеля включає валідацію через FluentValidation, мапінг через AutoMapper та збереження даних через Entity Framework Core.

API підтримує повний набір CRUD-операцій для всіх сутностей, включаючи:

- **Get**: Отримання списку сутностей із підтримкою фільтрації та пагінації.
- **Autocomplete**: Надання спрощених даних для автодоповнення в інтерфейсі (наприклад, вибір проєкту чи facturation code).
- **GetId**: Отримання детальної інформації про сутність за ідентифікатором.
- **Create**: Створення нової сутності з валідацією вхідних даних.
- **Update**: Оновлення існуючої сутності з валідацією.

Серверна частина розгорнута в Azure Container Apps, що забезпечує високу масштабованість і стабільність роботи системи.

2.3. Реалізація фронтенд частини

Фронтенд адміністративного веб-застосунку Timetracker розроблено на базі Angular 19 із використанням TypeScript, RxJS, Angular Material, Luxon, ExcelJS і Cashew. Для управління взаємодією з API створено базовий сервіс, який інкапсулює всі CRUD-операції (**Get**, **GetId**, **Create**, **Update**, **Autocomplete**). Цей сервіс відповідає за відправлення запитів до серверної частини, обробку відповідей і їх мапінг на моделі для фронтенду. Для спрощення роботи з формами використано **FormControl** та **FormGroup** з Angular Reactive Forms, а також реалізовано мапери для перетворення DTO-моделей у формати, придатні для відображення в інтерфейсі, і навпаки.

Інтерфейс порталу включає:

- **Модальні вікна** для створення та редагування сутностей (наприклад, табелів, запитів на відпустки, проєктів чи пристроїв). Модальні вікна використовують компоненти Angular Material для забезпечення зручного, адаптивного та сучасного вигляду.
- **Таблиці** для відображення списків даних (наприклад, списки табелів, проєктів чи пристроїв) із підтримкою сортування, фільтрації та дій (редагування, видалення, схвалення). Таблиці створено з використанням компонента **mat-table** від Angular Material.
- **Експорт даних**: Реалізовано функціональність експорту табелів і звітів у форматі Excel (.xlsx) за допомогою бібліотеки **ExcelJS**.
- **Реактивні потоки**: RxJS використовується для обробки асинхронних подій, таких як HTTP-запити, дії користувача чи оновлення стану через **Cashew** для управління станом.
- **Робота з датами**: Бібліотека **Luxon** застосовується для форматування дат, обробки часових зон і обчислень тривалості, наприклад, при заповненні табелів чи запитів на відпустки.

Після перевірки коректності взаємодії фронтенду з сервером (отримання, оновлення, збереження даних і автодоповнення) було написано юніт-тести для перевірки маперів, валідаторів і бізнес-логіки. Це забезпечує надійність і стабільність роботи інтерфейсу, включаючи коректну обробку даних і відображення в реальному часі.

2.4. Реалізація виконаних модулів

Виконана робота над системою Timetracker охоплює розробку модулів управління пристроями, шаблону веб-застосунку та сторінки Overview, які є ключовими компонентами адміністративного веб-застосунку. Кожен модуль розроблено з урахуванням потреб автоматизації управління робочими показниками персоналу. На початковому етапі дані про пристрої, таблиці та запити на відпустки відображалися в табличному вигляді з використанням компонентів Angular Material, що забезпечило зручне представлення та редагування даних. Для реалізації таблиць і навігаційних елементів використано кастомні компоненти Angular із застосуванням стилів CSS і тем Material Design, що дозволило створити сучасний і адаптивний інтерфейс із гнучкими можливостями відображення та редагування.

2.4.1. Система відстежування пристроїв

Опис роботи:

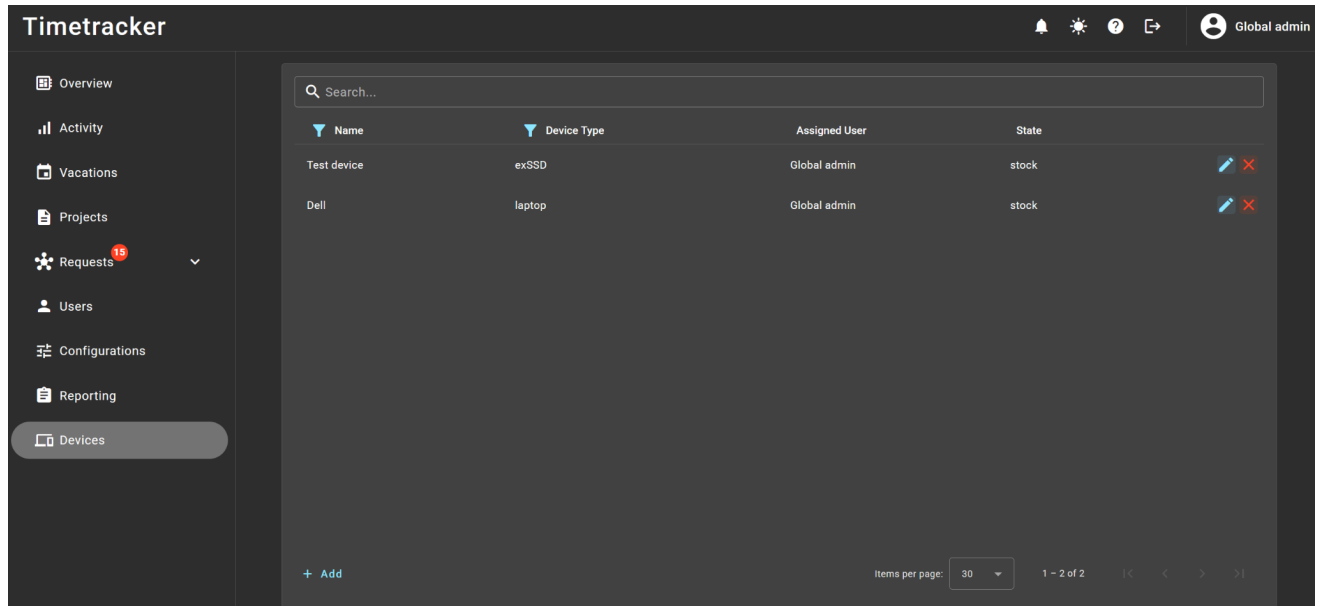
Модуль управління пристроями призначений для обліку обладнання, яке видається або повертається працівниками. Реалізація включала створення серверної логіки для обробки даних про пристрої та розробку інтерфейсу для їх управління. Модуль забезпечує прозорість і контроль за обладнанням компанії.

Реалізація на бекенді:

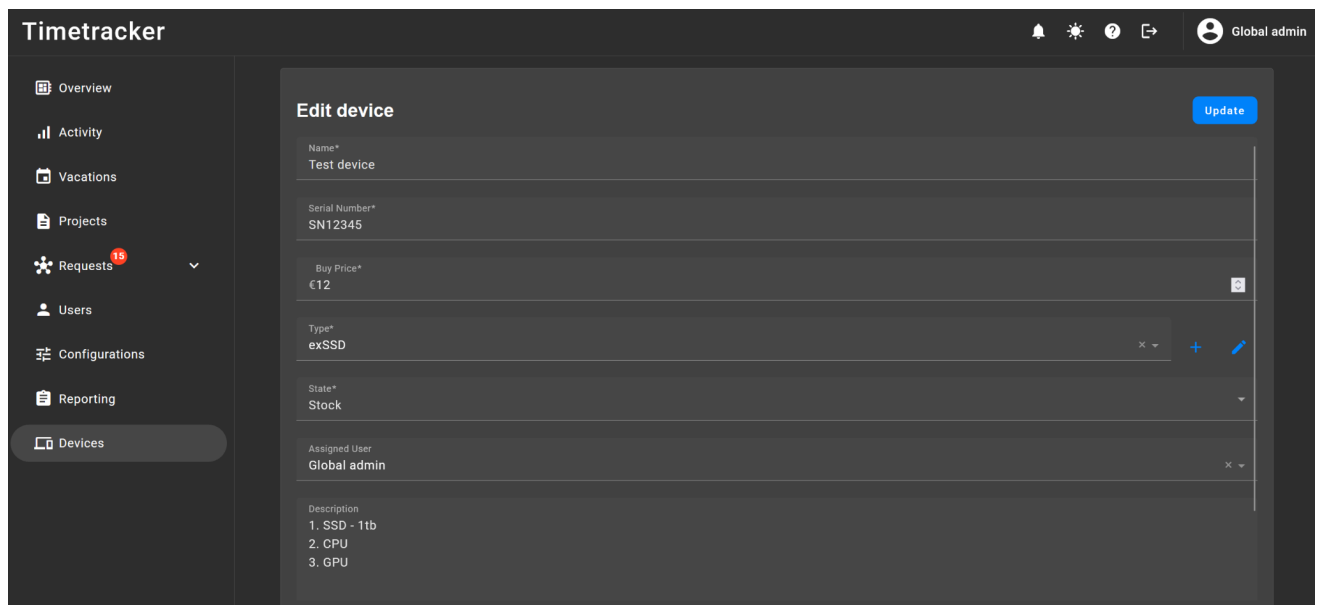
- Розроблено API-ендпоінти для CRUD-операцій (**Get**, **GetId**, **Create**, **Update**, **Delete**) за допомогою ASP.NET Core.
- Реалізовано логіку створення записів про видачу/повернення пристроїв із збереженням унікальних ідентифікаторів, типу пристрою, статусу та прив'язки до працівника.
- Використано **Entity Framework Core** для збереження даних у SQL Server із підтримкою міграцій.
- Налаштовано валідацію через **FluentValidation**, яка перевіряє унікальність ідентифікаторів пристрою та коректність даних (наприклад, статус видачі).
- Інтегровано **AutoMapper** для мапінгу між доменними моделями та DTO.

Реалізація на фронтенді:

- Створено окрему сторінку в Angular 19 для управління пристроями з використанням компонентів **Angular Material** (mat-table для списку пристроїв, модальні вікна для створення/редагування).
- Реалізовано таблицю з підтримкою сортування, фільтрації та дій (редагування, видалення).
- Налаштовано HTTP-запити до API через базовий сервіс із використанням **RxJS** для обробки асинхронних операцій.
- Використано **Angular Reactive Forms** для форм створення/редагування пристроїв із кастомними валідаторами (наприклад, перевірка унікальності серійного номера).



Малюнок 1 – Інтерфейс управління пристроями.



Малюнок 2 – Інтерфейс редагування пристрою.

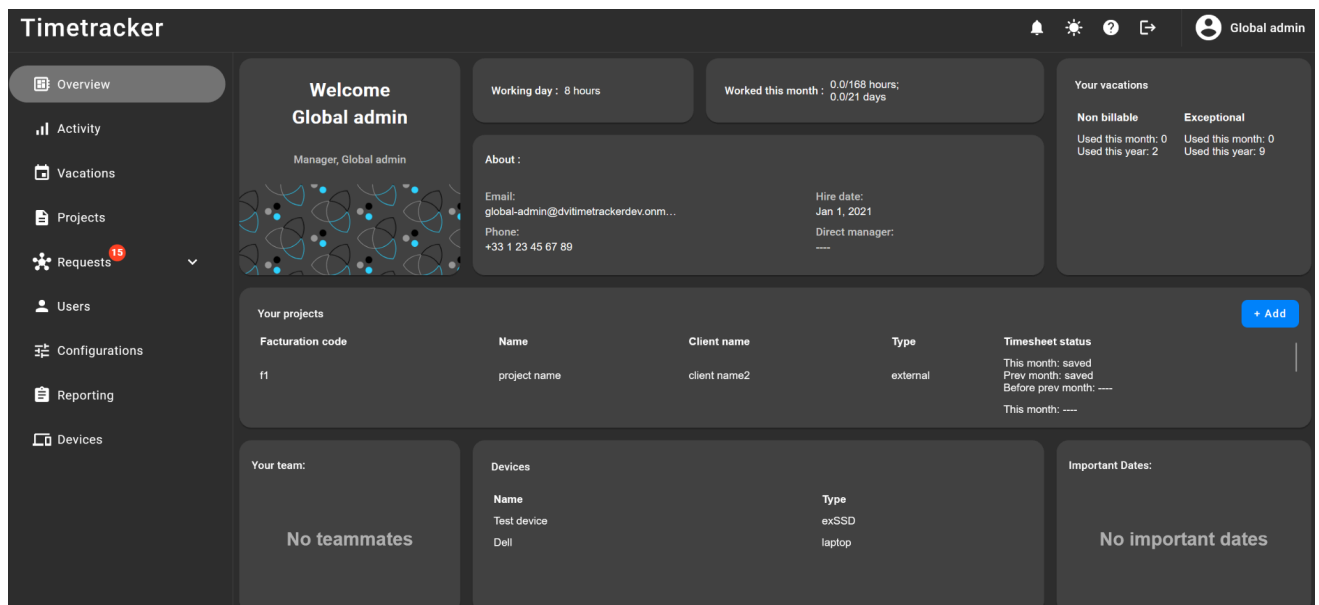
2.4.2. Створення шаблону веб-застосунку

Опис роботи:

Розроблено шаблон веб-застосунку, що включає візуальну частину інтерфейсу, зокрема бічне меню (sidebar menu) і верхню панель (toolbar). Це забезпечило єдиний стиль і зручну навігацію для всіх сторінок порталу.

Реалізація на фронтенді:

- Створено бічне меню з використанням компонента **mat-sidenav** від Angular Material, що підтримує адаптивний дизайн для різних розмірів екрана.
- Реалізовано верхню панель (**mat-toolbar**) з інтеграцією логотипу, навігаційних елементів і кнопок для швидкого доступу до основних функцій (наприклад, створення табеля чи звіту).
- Налаштовано стилі за допомогою CSS і тем Angular Material для забезпечення сучасного вигляду, що відповідає стандартам Material Design.
- Використано **Cashew** для управління станом навігації, наприклад, збереження стану відкритого/закритого меню.
- Оптимізовано продуктивність шляхом зменшення розміру бандлу за допомогою Angular CLI та lazy loading модулів.



Малюнок 3 – Інтерфейс бічного меню та верхньої панелі.

2.4.3. Створення сторінки Overview

Опис роботи:

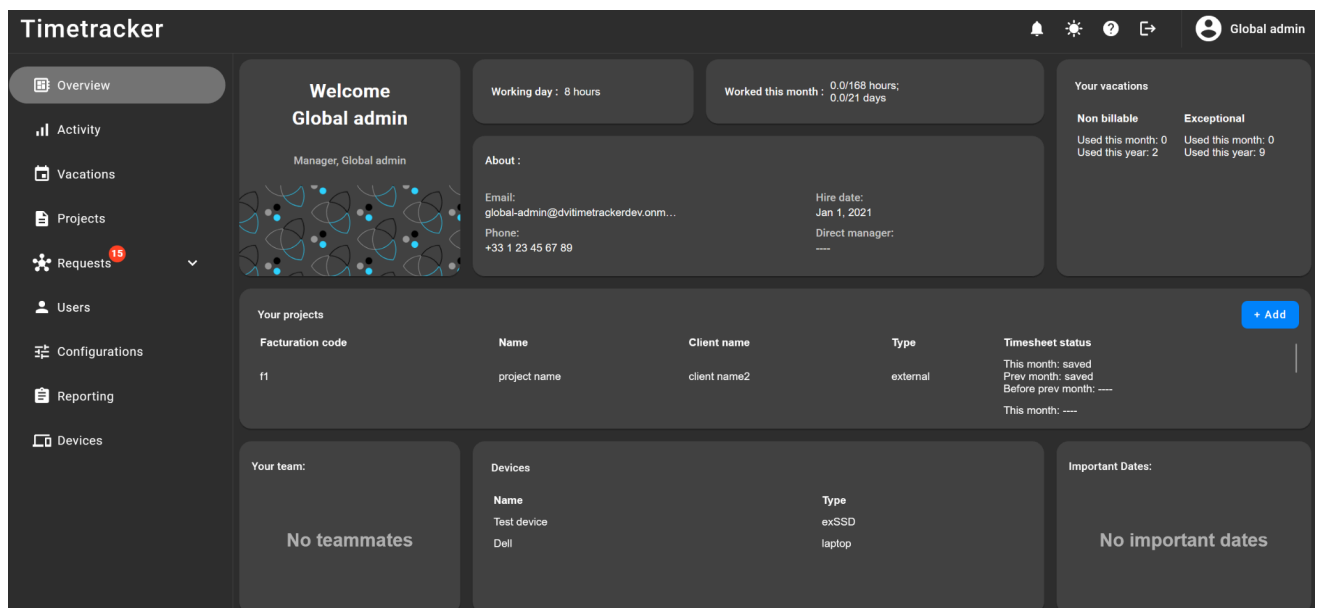
Сторінка Overview є ключовою частиною порталу, відображаючи загальну статистику та статус табелів і запитів на відпустки. Вона забезпечує швидкий доступ до основних даних для користувачів і менеджерів.

Реалізація на бекенді:

- Розроблено API-ендпоінт для отримання агрегованих даних (наприклад, кількість заповнених табелів, статусів відпусток) із підтримкою фільтрації за періодом чи проектом.
- Використано **Entity Framework Core** для формування складних запитів до SQL Server із використанням LINQ.
- Налаштовано кешування даних через **Azure Blob Storage** для оптимізації швидкості завантаження статистики.
- Застосовано **FluentValidation** для перевірки параметрів запитів (наприклад, коректність дат).

Реалізація на фронтенді:

- Створено сторінку Overview з використанням компонентів **Angular Material** (mat-card для відображення статистики).
- Налаштовано асинхронне завантаження даних через **RxJS** із підтримкою індикаторів завантаження (mat-spinner).



Малюнок 4 – Інтерфейс бічного меню та верхньої панелі.

2.5. Тестування та оптимізація

Для забезпечення якості виконаних модулів проведено юніт-тестування маперів, валідаторів і бізнес-логіки на фронтенді та бекенді. На фронтенді тести

перевіряли коректність роботи форм, мапінгу даних, відображення таблиць і графіків. На бекенді тести охоплювали валідацію DTO, збереження даних і обробку запитів. Інтеграційні тести підтвердили коректність взаємодії між фронтом, API та базою даних.

Тестування в реальному середовищі проводилося користувачами, які створювали записи про пристрої, перевіряли сторінку Overview і навігацію через бічне меню. Зворотний зв'язок вказав на необхідність оптимізації швидкості завантаження сторінки Overview для великих обсягів даних. Для цього було реалізовано кешування на бекенді та lazy loading компонентів на фронтенді. Виявлені помилки виправлено, а система оптимізована для забезпечення стабільної роботи.

2.6. Висновки щодо реалізації

Розробка системи управління пристроями, шаблону веб-застосунку та сторінки Overview дозволила створити зручний і функціональний веб-портал для управління робочими показниками персоналу. Використання **Angular 19**, **Angular Material**, **RxJS** і **Cashew** забезпечило сучасний і адаптивний інтерфейс, а **ASP.NET Core**, **Entity Framework Core**, **AutoMapper** і **FluentValidation** — стабільну та масштабовану серверну частину. Реалізація кастомних компонентів для таблиць і навігації дозволила підвищити зручність використання, а валідація на всіх рівнях і тестування підтвердили надійність системи.

РОЗДІЛ 3

РОЗДІЛ 3. РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Для тестування розробленої системи Timetracker програмне забезпечення було розгорнуто в хмарному середовищі Microsoft Azure, що забезпечує гнучке масштабування, високу доступність і управління сервісами.

Microsoft Azure використано для розгортання системи з наступними сервісами:

- **Azure SQL Database:** Керована реляційна база даних для зберігання даних про проекти, таблиці, запити на відпустки, пристрої та контракти.
Використовується план служби Standard S0: 10 DTU, 250 ГБ сховища, що забезпечує автоматичне оновлення, резервне копіювання та моніторинг без участі користувача.
- **Azure Static Web Apps:** Використовується для розгортання фронтенду адміністративного веб-застосунку, забезпечуючи швидке завантаження сторінок і високу доступність.
- **Azure Container Apps:** Застосовано для розгортання серверної частини (API), що забезпечує масштабованість і стабільну роботу.
Використовується стек часу виконання .NET 9, операційна система Linux, план служби Standard S1: 100 ACU, 1.75 ГБ пам'яті.
- **Azure Blob Storage:** Використовується для зберігання неструктурованих даних, таких як файли звітів і логів пристроїв.
- **Azure Key Vault:** Застосовано для безпечного зберігання секретів, таких як ключі API, паролі та сертифікати.
- **Azure AD:** Інтегровано для управління групами доступу та аутентифікації користувачів із різними ролями (кінцеві користувачі, менеджери, глобальні адміністратори).

Після розгортання системи в Azure було проведено заповнення бази даних тестовими даними. За допомогою адміністративного веб-застосунку створено

тестові проєкти, facturation code, таблиці, запити на відпустки та записи про пристрої. Кожен елемент системи, як-от таблиці чи запити, було доповнено текстовими коментарями, датами та іншими даними, необхідними для тестування.

Для перевірки функціональності системи проведено наступні види тестування:

- **Unit-тести:** Перевірка всіх API-ендпоінтів, бізнес-логіки, маперів і валідаторів. Тести виконано з використанням **xUnit** для бекенду та **Jasmine/Karma** для фронтенду.
- **Тестування в реальному середовищі:** Система була протестована реальними користувачами (кінцевими користувачами, менеджерами та адміністраторами), які виконували операції заповнення таблиць, створення запитів на відпустки, управління пристроями та аналізу статистики на сторінці Overview. Користувачі залишали відгуки та повідомляли про знайдені помилки.

Тестування підтвердило стабільну роботу системи, правильність архітектури бази даних і ефективність обраного технологічного стеку. Зокрема, API-ендпоінти обробляли запити швидко, а інтерфейс порталу виявився зручним завдяки використанню **Angular Material** і адаптивного дизайну. Сторінка Overview забезпечувала швидкий доступ до статистики, а модуль управління пристроями коректно відображав і оновлював дані.

Виявлені недоліки включали необхідність додаткових валідаційних механізмів для введення даних, наприклад, перевірки коректності формату дат у табелях. Зворотний зв'язок від користувачів вказав на потребу оптимізації інтерфейсу для швидшого пошуку проєктів і відображення великих обсягів даних на сторінці Overview. У перспективі планується вдосконалення системи шляхом завершення модуля Topic для коментування сутностей, додавання розширеної аналітики з експортом звітів у PDF і створення мок-сервісів для локального тестування **Azure Functions**.

ВИСНОВКИ

Проведено аналіз інформаційних систем для управління робочими показниками персоналу, виявлено значне поширення систем, що автоматизують табелювання та обробку відпусток. У якості прикладу для реалізації обрано систему Timetracker, яка забезпечує ефективне управління робочими процесами в великих організаціях.

Визначено програмні засоби та технології (**Angular 19, ASP.NET Core, Azure, Docker, Azure DevOps, Azure AD**), а також API-інтерфейси, які дозволяють розробити адміністративний веб-портал, серверну частину та сховище даних для управління робочими показниками. Ці засоби використано під час створення системи.

Спроектовано, реалізовано та випробувано адміністративний веб-портал, серверну частину (API-сервіс) та сховище даних (SQL Server і Azure Blob Storage). Система отримала необхідні засоби управління безпекою, розподілу ролей (кінцеві користувачі, менеджери, глобальні адміністратори) через інтеграцію з **Azure AD**. Реалізовано інтерфейс для заповнення табелів, управління запитами на відпустки, проєктами, пристроями та створення звітів.

Систему випробувано шляхом розгортання в хмарному середовищі Azure з використанням **Azure Static Web Apps, Azure Container Apps, Azure SQL Database і Azure Blob Storage**. Проведено unit- та інтеграційні тести, а також тестування в реальному середовищі реальними користувачами, які залишили відгуки та повідомили про помилки. Результати випробувань підтвердили працездатність системи, правильність обраної архітектури та достатню ефективність використаних технологій. Виявлено потребу в додаткових валідаційних механізмах і оптимізації інтерфейсу для швидшого доступу до даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Стратегія розвитку інформаційного суспільства в Україні.**
Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р. // Урядовий портал. URL: <https://www.kmu.gov.ua/npas/246420577>
Опис: Документ, який визначає ключові напрями розвитку інформаційних технологій в Україні, зокрема автоматизацію бізнес-процесів, що є основою для актуальності розробки систем управління персоналом, таких як Timetracker.
2. **Стратегія кібербезпеки України.** Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165.
Опис: Нормативний документ, який підкреслює важливість безпеки даних у хмарних системах, що є актуальним для інтеграції Azure AD і Azure Key Vault у Timetracker.
3. **Microsoft Azure Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/>
Опис: Офіційна документація Microsoft Azure, яка охоплює хмарні сервіси, використані для розгортання системи (Azure Static Web Apps, Azure Container Apps, Azure Blob Storage, Azure SQL Database, Azure AD).
4. **Azure SQL Database Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/azure-sql/>
Опис: Технічна документація щодо Azure SQL Database, яка використовується для зберігання реляційних даних у системі Timetracker.
5. **Azure Static Web Apps Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/static-web-apps/>
Опис: Документація до Azure Static Web Apps, яка застосовується для розгортання фронтенду адміністративного веб-застосунку.
6. **Azure Container Apps Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/container-apps/>
Опис: Документація до Azure Container Apps, яка використовується для розгортання серверної частини системи.
7. **Azure DevOps Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/devops/>
Опис: Документація до Azure DevOps, що охоплює інструменти для CI/CD, Git-репозиторії, планування спринтів і управління проєктами, використані в розробці Timetracker.

8. **Azure Active Directory Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/entra/identity/>
Опис: Документація до Azure AD, яка застосовується для управління групами доступу та аутентифікації користувачів у системі.
9. **.NET Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/>
Опис: Офіційна документація до платформи .NET, зокрема ASP.NET Core, яка використовується для створення RESTful API у серверній частині.
10. **Entity Framework Core Documentation.** Microsoft Learn. URL: <https://learn.microsoft.com/en-us/ef/core/>
Опис: Документація до Entity Framework Core, яка застосовується для роботи з реляційною базою даних SQL Server і підтримки міграцій.
11. **Angular Official Documentation.** Angular. URL: <https://angular.dev/>
Опис: Офіційна документація Angular 19, яка використовується для створення фронтенду адміністративного веб-застосунку.
12. **Material Design Documentation.** Google. URL: <https://m3.material.io/>
Опис: Документація до Angular Material, яка застосовується для створення адаптивного та сучасного інтерфейсу користувача.
13. **TypeScript Documentation.** TypeScript. URL: <https://www.typescriptlang.org/docs/>
Опис: Документація до TypeScript, який використовується для написання типізованого коду у фронтенді.
14. **RxJS Documentation.** ReactiveX. URL: <https://rxjs.dev/>
Опис: Документація до RxJS, яка застосовується для обробки асинхронних подій у фронтенді.
15. **Luxon Documentation.** Moment.js. URL: <https://moment.github.io/luxon/>
Опис: Документація до бібліотеки Luxon, яка використовується для роботи з датами та часом у фронтенді.
16. **ExcelJS Documentation.** GitHub. URL: <https://github.com/exceljs/exceljs>
Опис: Документація до бібліотеки ExcelJS, яка застосовується для генерації та експорту звітів у форматі Excel.
17. **Docker Documentation.** Docker. URL: <https://docs.docker.com/>
Опис: Документація до Docker, який використовується для створення контейнеризованих середовищ під час локальної розробки.
18. **AutoMapper Documentation.** GitHub. URL: <https://docs.automapper.org/en/stable/>
Опис: Документація до AutoMapper, який застосовується для мапінгу між доменними моделями та DTO.
19. **FluentValidation Documentation.** GitHub. URL: <https://docs.fluentvalidation.net/en/latest/>

Опис: Документація до FluentValidation, яка використовується для валідації вхідних запитів на сервері.

20. **Clean Architecture Principles.** Robert C. Martin (Uncle Bob). URL:

<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

Опис: Стаття, яка описує принципи Clean Architecture, використані для розділення бізнес-логіки, інтерфейсу та інфраструктури в системі Timetracker.

21. **ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection.** ISO. URL: <https://www.iso.org/standard/27001>

Опис: Міжнародний стандарт інформаційної безпеки, який враховувався при реалізації безпечного зберігання даних і аутентифікації через Azure AD і Azure Key Vault.