УДК 004.4 DOI https://doi.org/10.35546/kntu2078-4481.2024.3.30

Д. В. МЕЛЬНИЧЕНКО
магістр
Державний університет «Житомирська політехніка»
ORCID: 0009-0004-9632-4380

Т. А. ВАКАЛЮК
доктор педагогічних наук, професор,
завідувач кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»
ORCID: 0000-0001-6825-4697

О. В. ФАРРАХОВ
кандидат технічних наук,
в.о. вченого секретаря
Центр інформаційно-аналітичного та технічного забезпечення
моніторингу об'єктів атомної енергетики
Національної академії наук України
ORCID: 0000-0003-4988-126X

І. В. ГОРДІЄНКО
кандидат педагогічних наук, доцент,
доцент кафедри математики та економіки
Дрогобицький державний педагогічний університет
імені Івана Франка
ORCID: 0000-0001-6182-4968

ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ РОБОЧИМ ЧАСОМ

З розвитком технологій та змінами в бізнес-середовищі, автоматизація процесів обліку та планування робочого часу стає критично важливою. Використання вебзастосунків для цих завдань дозволяє підприємствам оптимізувати внутрішні процеси, підвищити продуктивність і забезпечити точність обліку робочого часу. Метою даної роботи є розробка вебзастосунку для управління робочим часом з елементами системи управління персоналом, який забезпечить автоматизацію, оптимізацію та спрощення процесів обліку та планування робочого часу. Цей вебзастосунок має на меті забезпечити ефективне використання робочих ресурсів та полегшити керування завданнями і проектами, надаючи користувачам зручний та інтуїтивний інтерфейс.

Архітектура відіграє ключову роль у реалізації системи, забезпечуючи високу масштабованість для подальшого розвитку проекту. Надійність і гнучкість архітектури дозволяє розширювати проект у майбутньому, перетворюючи його на комплексну систему управління персоналом. Це забезпечує можливість інтеграції нових функцій та модулів, які підвищують функціональність та ефективність системи.

Особливу увагу приділено зручності користувацького інтерфейсу, який дозволяє користувачам легко взаємодіяти з елементами системи, забезпечуючи їм приємний досвід роботи. Інтерфейс розроблений таким чином, щоб мінімізувати час навчання користувачів та максимізувати продуктивність їхньої роботи.

Розроблений вебзастосунок може бути впроваджений у діяльність підприємств різних галузей для автоматизації обліку та планування робочого часу. Це дозволить підвищити ефективність управління робочими процесами, знизити витрати часу на адміністративні процедури та покращити продуктивність працівників. Система також забезпечує точність даних, що сприяє оптимізації внутрішніх процесів компаній та створенню основи для подальшого розвитку повноцінної системи управління персоналом.

Ключові слова: *вебзастосунок, архітектура, персонал, управління, час.*

D. V. MELNYCHENKO
Master's Student
Zhytomyr Polytechnic State University
ORCID: 0009-0004-9632-4380

T. A. VAKALIUK
Doctor of Pedagogical Sciences, Professor,
Head of the Department of Software Engineering
Zhytomyr Polytechnic State University
ORCID: 0000-0001-6825-4697

O. V. FARRAKHOV
PhD, Acting Academic Secretary
Centre for Information, Analytical and Technical Support
for Monitoring of Nuclear Facilities
of the National Academy of Sciences of Ukraine
ORCID: 0000-0003-4988-126X

I. V. HORDIENKO
Candidate of Pedagogical Sciences, Associate Professor,
Associate Professor at the Department of Mathematics and Economics
Drohobych Ivan Franko State Pedagogical
ORCID: 0000-0001-6182-4968

WEB APPLICATION DESIGN FOR WORKING TIME MANAGEMENT

With technology development and changes in the business environment, automation of time tracking and scheduling processes is becoming critical. Using web applications for these tasks allows enterprises to optimise internal processes, increase productivity and ensure time-tracking accuracy. This paper aims to develop a web-based time management application with elements of a personnel management system that will automate, optimise and simplify the processes of time tracking and scheduling. This web application aims to ensure efficient use of work resources and facilitate task and project management by providing users with a user-friendly and intuitive interface.

The architecture plays a vital role in the system's implementation, ensuring high scalability for further project development. The architecture's reliability and flexibility allow the project to be expanded in the future, turning it into a comprehensive HR management system. This ensures that new functions and modules can be integrated to improve the system's functionality and efficiency.

Particular attention is paid to the user interface, which allows users to easily interact with the system elements and provides them with a pleasant working experience. The interface is designed to minimise the user learning curve and maximise productivity.

The developed web application can be implemented in the activities of enterprises of various industries to automate the accounting and planning of working time. This will increase the efficiency of workflow management, reduce time spent on administrative procedures and improve employee productivity. The system also ensures the accuracy of data, which helps to optimise the internal processes of companies and create the basis for the further development of a full-fledged HR management system.

Key words: *web application, architecture, personnel, management, time.*

Постановка проблеми

Сучасні підприємства, незалежно від їхньої галузі, стикаються з необхідністю ефективного управління робочим часом та персоналом. З розвитком технологій та змінами в бізнес-середовищі, автоматизація процесів обліку та планування робочого часу стає критично важливою. Використання вебзастосунків для цих завдань дозволяє підприємствам оптимізувати внутрішні процеси, підвищити продуктивність і забезпечити точність обліку робочого часу. В умовах сучасної конкуренції на ринку, такі інструменти стають стратегічно важливими для підтримання ефективності та конкурентоспроможності компаній.

Окрім того, компанії потребують інструментів, які дозволяють легко планувати та контролювати робочий час, забезпечують автоматизацію підрахунку відпрацьованих годин, управління запитами на відпустки та лікарняні, а також інші адміністративні процедури. Інтеграція всіх цих функцій у одному вебзастосунку надає підприємствам значні переваги, включаючи економію часу, підвищення точності даних та поліпшення продуктивності працівників.

Аналіз останніх досліджень і публікацій

Проблематика ефективного управління часом стає дедалі актуальнішою в наукових колах. Розглянемо деякі ключові дослідження в цій галузі. Писаревська Г. І. зосередила свою увагу на потенціалі тайм-менеджменту як інструменту для оптимізації процесів управління персоналом [4]. Група науковців об'єднала зусилля для вивчення специфіки застосування технік тайм-менеджменту в роботі державних службовців [6]. Євтушенко Г. І. та Дерев'янко В. М. провели ґрунтовний аналіз поточного стану управління робочим часом та запропонували методи підвищення ефективності впровадження тайм-менеджменту в організаціях [3]. Примак Т. Ю. та Васильчук О. В. розглянули можливості підвищення продуктивності туристичних підприємств за допомогою методів тайм-менеджменту [7]. При цьому Крикун О. О. та Медяник Ю. Г. зробили акцент на дослідженні цифрових інструментів, які можуть допомогти менеджерам створити дієву систему управління часом [5].

Формулювання мети дослідження

Метою даної роботи є розробка вебзастосунку для управління робочим часом з елементами системи управління персоналом.

Викладення основного матеріалу дослідження

Враховуючи специфіку проекту, використаємо модель керування доступом на основі дозволів (permission-based access control, PBAC), що забезпечує більш гнучке та детальне налаштування прав доступу для кожного користувача. Модель PBAC дозволяє визначати та надавати специфічні дозволи окремим користувачам або групам користувачів залежно від їхніх функціональних обов'язків та потреб. Це забезпечує підвищену безпеку та ефективність управління доступом, дозволяючи користувачам виконувати лише ті операції, на які вони мають відповідні дозволи.

Вебзастосунок орієнтований на масштабованість і гнучку архітектуру. Тому важливо впровадити більшість інструментів, які дозволять згодом реалізувати всі необхідні функції для такої великої і складної програми, як система управління персоналом. Наявність цих інструментів у системі є критично важливою, оскільки вони зможуть вирішувати більшість проблем, навіть якщо наразі функціонал, реалізований за їх допомогою, ще не є значним.

Сформулюємо вимоги до системи, беручи до уваги принцип контролю доступу на основі дозволів. Відповідно до потреб компанії користувач може мати найрізноманітніший набір дозволів, тож наведені нижче вимоги користувачів це лише декілька із прикладів потреб компанії.

Вимоги користувачів:

1. Звичайний робітник: авторизація і відновлення паролю; відстеження робочого часу; керування власним робочим часом та перегляд планувальника інших; перегляд записів про старі відпустки, наявність інформації про наявну кількість балансу відпустки, створення нового запиту на відпустку; перегляд записів про лікарняні; перегляд детальної статистики про відпрацьовані години місяця.
2. Бухгалтер: завантаження файлу із статистикою за місяць; чітко бачити відпрацьовані години користувача протягом певного періоду часу; доступ до інформації про відпустки, лікарняні інших робітників і вихідні дні.
3. HR-менеджер: керування вихідними даними; редагування даних профілю; активація та деактивація акаунту; реєстрація нових користувачів; керування відпустками інших користувачів.

Функціональні вимоги:

- відсутність традиційної реєстрації, користувача в системі має реєструвати людина із відповідним дозволом. Після отримання листа користувач завершує реєстрацію самостійно встановивши собі пароль;
- можливість зручно і швидко знайти потрібного робітника;
- потрібно надати можливість зручно відслідковувати час користувачам. Для тих, хто працює повний робочий день, має бути присутнє автоматичне нарахування 8 годин на день;
- візуалізація статистичної інформації для кращого сприйняття;
- зручний перегляд інформації про години за допомогою планувальника;
- створення сторінки налаштувань для додавання гнучкості елементам системи;
- отримувати сповіщення у вигляді email-листів коли це необхідно.

Аналіз вимог до вебзастосунку дозволив краще сформувати об'єктно орієнтовану структури системи і реалізувати потрібні елементи системи. Для прикладу розглянемо діаграму класів модулю користувачів (рис. 1). На діаграмі класів добре видно взаємодії елементів системи, розглянемо їх детальніше. **DapperContext** – створює і керує підключенням до бази даних за допомогою Dapper ORM, яка лежить в основі взаємодії із базою даних.

Абстрактний клас **Repository**, який є батьківським для усіх репозиторіїв, забезпечує базові спільні операції CRUD для сутностей, а також використовує DapperContext для взаємодії із базою даних. Кожен клас репозиторію прив'язаний до своєї таблиці в базі даних, наприклад **UserRepository** (таблиця Users) чи **SettingsRepository** (таблиця Settings). Іноді потрібна більш комплексна логіка для взаємодії із об'єктами бази даних, наприклад клас **SettingsManager** керує налаштуваннями застосунку, включаючи збереження та оновлення конфігурацій, які зберігаються в серіалізованому форматі. Таким чином використовуючи таку обгортку над репозиторієм, можемо взаємодіяти із налаштуваннями так, ніби виконуємо стандартні CRUD-операції.

Інтерфейс **IMailTemplateManager**, який реалізується класом **MailTemplateManager**, створений для того щоб читати інформацію із ресурсних файлів (.resx). В нашому випадку ресурсний файл зберігає шаблони листів для відправки. Для відправки email-листів створено інтерфейс **IMailer**, реалізований через відповідний клас **Mailer**. Mailer використовує раніше описаний IMailTemplateManager для читання шаблону листу, підстановки в нього потрібних значень та його відправки. Для зберігання таких налаштувань пошти як хост, порт, пароль та інших, використовується клас **MailSettings**, який читає ці дані із конфігурації. Хоч клас Mailer із відповідним інтерфейсом вже можна використовувати для відправки листа, набагато зручніше використовувати класи-обгортки, які будуть робити це за нас. Наприклад клас **AuthMailer** реалізовує логіку надсилання листів пов'язаних із автентифікацією (встановлення паролю), для чого необхідно просто викликати метод.

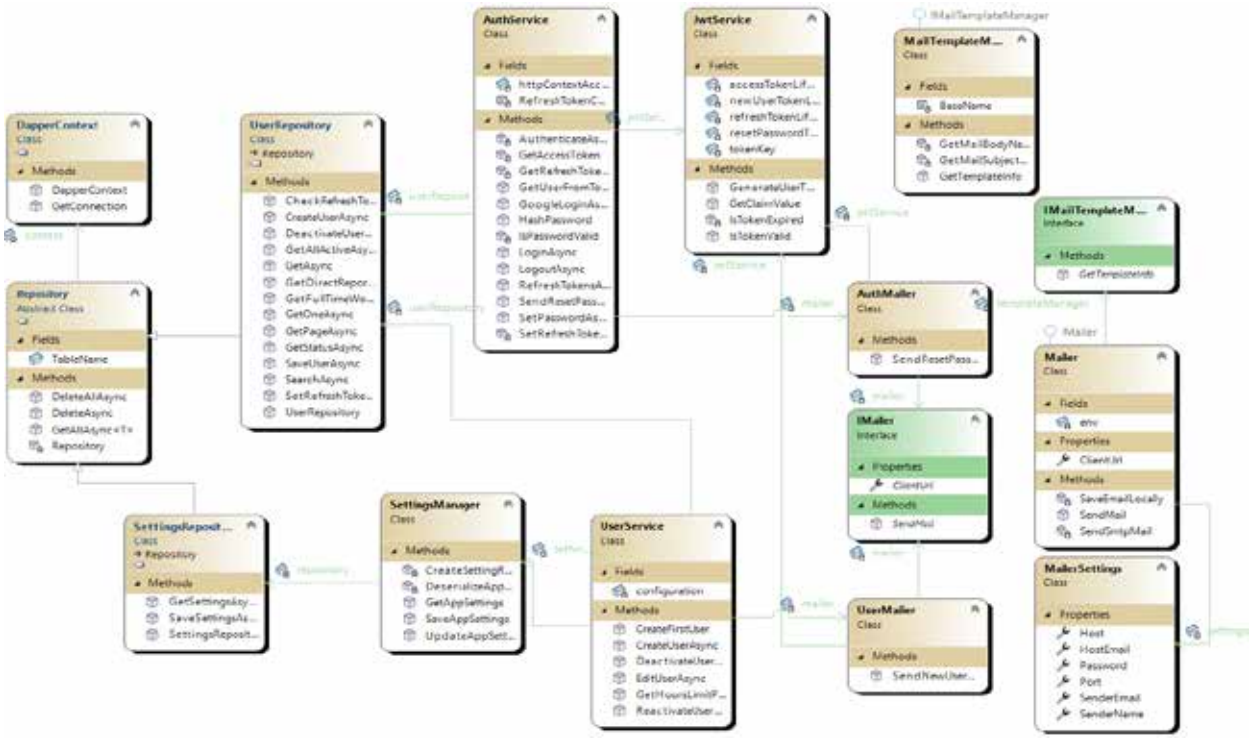

Рис. 1. Діаграма класів для модулю користувача

Такі класи, як **UserServive** виконують основні функції системи, до них звертаються класи що реалізовують API. UserService це клас для управління користувачами, що виконує логіку управління цими користувачами. Він використовує інші класи для реалізації алгоритмів, такі як UserRepository, SettingsManager чи UserMailer. Алгоритми класу визначають коли потрібно звернутися до бази даних, надіслати лист чи просто провести обрахунки.

JwtService – це клас, що лежить в основі автентифікації користувачів і містить методи для генерації, валідації та читання JWT токенів для користувачів. **AuthService** покладається на цей клас для автентифікації користувачів, включаючи управління токенами доступу, хешування паролів, встановлення cookie, встановлення токенів і паролів користувачів в базу даних за допомогою класу UserRepository, відправки листів за допомогою класу **AuthMailer**.

Для забезпечення усіх вимог, поставлених до вебзастосунку, потрібно спроектувати основні алгоритми роботи системи. Загальний алгоритм роботи системи можна описати як безперервну взаємодію користувача з вебінтерфейсом для, який в свою чергу використовуючи потоків даних сигналізує клієнтській логіці про необхідність створення запиту до серверу. Сервер, використовуючи GraphQL, обробляє запит використовуючи внутрішню бізнес логіку, за потреби використовує базу даних для діставання чи мутації даних. Результатом є надання відповіді клієнтській частинні застосунку після чого нові дані оновлюють стан системи і перемальовують інтерфейс.

Наведемо загальний алгоритм роботи на конкретному прикладі за допомогою діаграми активності – створення запиту на відпустку (рис. 2).

Розглянемо окремі фрагменти коду, що потребують детального огляду, звернувши увагу на статистично-аналітичний модуль. Він виконує логіку для отримання статистичних чи аналітичних даних. Розглянемо як відбувається обрахунок відпрацьованої кількісті днів.

1. Обраховується загальна кількість робочих днів у заданому діапазоні часу за допомогою методу GetWorkingDays(from, to).

2. Отримання даних про свята та скорочені дні:

2.1. Отримуються дати свят у вказаному діапазоні за допомогою репозиторію holidayRepository.

2.2. Обраховуються дні відпочинку на основі свят через метод GetDayOffDays(holidays, from, to).

2.3. Обраховуються скорочені дні через метод GetShortDays(holidays, from, to).

3. Отримання даних про відпустки, лікарняні та робочі сесії:

3.1. Отримуються ідентифікатори користувачів.

3.2. Викликаються відповідні репозиторії для отримання відпусток, лікарняних та робочих сесій у вказаному діапазоні часу:

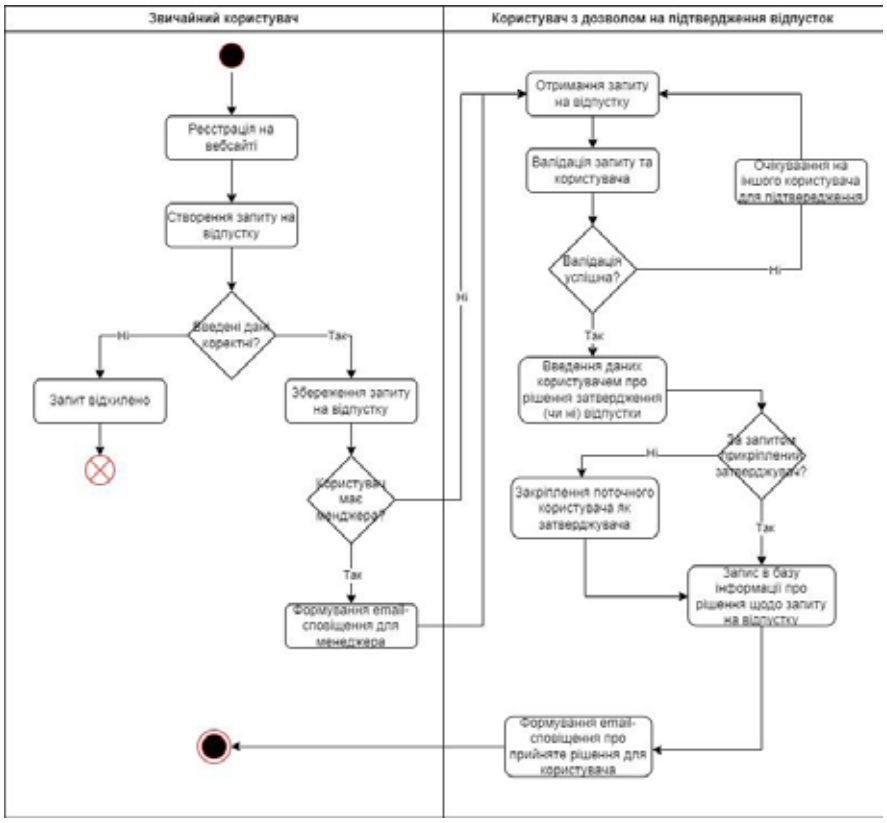

Рис. 2. Діаграма активності для створення і затвердження запиту на відпустку

4. Обрахунок індивідуальних зведень для кожного користувача:

4.1. Для кожного користувача обраховується кількість годин на день, враховуючи його ставку зайнятості.

4.2. Обраховується загальна кількість годин, які користувач має відпрацювати.

4.3. Віднімаються години на основі днів відпочинку та скорочених днів.

4.4. Віднімаються години на основі відпусток та лікарняних днів.

4.5. Обраховуються фактичні години, які користувач відпрацював.

4.6. Обраховується відсоток відпрацьованих годин від запланованих.

4.7. Створюється об’єкт IndividualHoursSummary для кожного користувача з розрахованими даними.

5. Повертається об’єкт HoursSummaryResult, що містить загальну кількість робочих днів, кількість святкових днів, кількість днів відпочинку та індивідуальні зведення для кожного користувача.

Таким чином, метод забезпечує детальний обрахунок відпрацьованих годин для кожного користувача з урахуванням святкових днів, відпусток та лікарняних, його годинної ставки та інших деталей як виключення годин, які вже були виключені під час відрахування усіх днів суботи та неділі.

Розглянемо клієнтську частину та деталі її реалізації. Підвантажування сторінок це певний механізм, який дозволяє зробити запит на отримання інформації із серверу, для функціонування сторінки. При цьому навігація не відбудеться доки запит не завершиться. Найбільшою перевагою такого підходу є майже повня ізоляція всієї логіки роботи навігації і підвантаження від компонентів, які займаються промальовуванням вебінтерфейсу. Зі сторони компонента, відбувається лише перехід за посиланням, в той час як система аналізує нове посилання, робить запит до серверу, якщо це потрібно, формує данні для сторінки і встановлює їх для загального стану системи, знаходить і промальовує потрібну сторінку. При цьому компонент не має жодної додаткової логіки, на момент завантаження він використовує дані, що вже будуть в стані системі.

В реалізації такого підходу було використано реактивне програмування. Маючи можливість поставити в стан програми будь-яку потрібну сторінку і дані, можемо не тільки ізолювати логіку від відображення, а й забезпечити актуальність даних. Розглянемо як саме визначаємо переадресацію на новий шлях, знаходимо потрібний обробник, робимо запит за потреби і оновлюємо стан сторінки. В цьому допоміг еріс, який є частиною бібліотеки Redux-Observable, що дає змогу керувати побічними ефектами в Redux за допомогою реактивного програмування RxJS.

```
export const loadPageStateEpic: Epic<LocationChangedAction> = (action$, state$, dependencies) =>
  action$.pipe(
    ofType(LOCATION_CHANGE),
    switchMap(({ payload: { location } }) => {
      const routes = routeBuilder.allRoutes;
      const possibleMatches = matchRoutes(routes, location.pathname);

      if (!possibleMatches || possibleMatches.length !== 1)
        return of(push(routeBuilder.routes.notFound));

      const match = possibleMatches[0];
      const pattern = possibleMatches[0].route.path;
      const handler = getPageHandler(pattern);

      const routeData: RouteData = {
        location,
        params: match.params,
        pattern,
      };

      return state$.pipe(
        first(({ general }) => !general.isInitialLoading),
        switchMap(state => handler(state, dependencies, routeData).pipe(
          concatMap(page => of(
            setUrlParams(match.params),
            setPageState(page),
            setPageLoading(false),
          )),
          startWith(setPageLoading(true)),
        )),
      );
    }),
  );
```

Розглянемо покроково дії, що виконує epic:

1. Epic підписується на action (подія в Redux), який відповідний за зміну шляху в url. Тобто як тільки шлях змінився, epic реагує на це і починає свою роботу, що гарно пояснює концепції реактивного програмування.
2. Action, що спричинив роботу цього epic, містить інформацію про новий url-шлях, за допомого чого маємо знайти його шаблон, наприклад «/user/profile/5» буде відповідати шаблону «/user/profile/:userId», де userId дорівнює 5.
3. Після знаходження потрібного шаблону, підготовлюємо дані для передачі його в обробник сторінки (pageHandler), про який розказано нижче. Дані включають актуальний стан системи, всю дані маршруту та залежності, такі як клас для взаємодії із API чи об'єкт для створення popup сповіщень.
4. Отримуємо потрібний обробник сторінки, який є унікальним для кожної сторінки. Зокрема, коли користувач переходить на сторінку створення відпустки, відпрацьовує наступний обробник:

```
export const vacationCreateHandler: PageHandler<VacationCreatePageState> = (_, { api }) => {
  return api.Request<VacationCreateStateResponse>(getVacationCreatePageStateRequest).pipe(
    map(({ data }) => ({
      pageName: PageName.vacationsRequest,
      balanceS: data.vacation.getVacationBalance,
      bookedDays: data.vacation.getBookedDays,
    })),
  );
};
```

Обробник отримав дані із серверу про наявний баланс відпусток користувача, створив новий об'єкт сторінки і повернув його до еріс. Використання цього механізму доводить наскільки корисними і зручними можуть бути концепції реактивного програмування. Наведемо для прикладу також інший еріс, який є відповідальним за підвантаження актуальної сесії користувача для відображення в компоненті для відслідковування робочих годин:

```
export const getTrackerInfoEpic: Epic = (_, state$, { api }) =>
  state$.pipe(
   filter(({ currentUser, tracker }) => currentUser.isLoggedIn && tracker.requireUpdate),
    exhaustMap(({ currentUser }) => makeTrackerRefreshRequest(api, currentUser.user!.id)),
  );

const makeTrackerRefreshRequest = (api: Api, userId: string) => api.Request<GetTrackerInfoResponse>(
  getTrackerInfoRequest,
  {
    userId,
    dayDate: formatDateOnlyString(dayjs()),
  },
).pipe(
  map(({ data: { workSession, user } }) => {
    const activeWs = workSession.getActive;
    const workedTime = workSession.getWorkedTimeForDay;
    const dayWorkTimeRate = user.getHoursPerDay;

    return loadTrackerInfo({
      activeWorkSession: activeWs,
      workedTimeForDayMS: workedTime,
      workRatePerDayMS: dayWorkTimeRate,
    });
  }),
);
```

1. Еріс підписується на загальний стан системи застосунку.
2. Еріс проводить фільтрацію стану, щоб визначити чи треба йому починати роботу. В даному випадку він почне роботи, якщо користувач увійшов у систему та трекер потребує оновлення даних актуальної робочої сесії.
3. Якщо стан пройшов фільтрацію, то еріс робить запит до серверу для отримання актуальної сесії користувача.
4. Після отримання відповіді від серверу, еріс встановлює дані в стан системи, що спричинить перемальовування компоненту для відслідковування робочого часу.

Опишемо окремі складові структури інтерфейсу розробленого додатку. Взаємодія користувачів із системою відбувається через вебзастосунок. Більшість функцій системи вимагають від користувача бути авторизованим, адже це платформа зав'язана на збиранні інформації для конкретної людини.

Головна сторінка – це перше, що бачить користувач, коли заходить до вебсайту. Одним із найважливіших елементів системи є трекер, який надає можливість користувачу засікати час, який потім буде збережений в системі (див. рис. 3).

Це один із ключових елементів, і доступ до нього має бути присутній завжди, тому було прийнято рішення розмістити його у вигляді кнопки, що завжди зафіксована на дні сторінки. При натисканні кнопки відбудеться відображення додаткової панелі, де буде відображені такі елементи як актуальна робоча сесія, кількість відпрацьованого часу за сьогодні, шкала для демонстрації потрібного відпрацьованого часу за сьогодні (див. рис. 4).

Для користувача з усім дозволами, є можливість налаштовувати систему відпвідно до власних потреб. Однією із найвідвідуваніших сторінок неодмінно буде сторінка із переліком усіх працівників (рис. 5).

Для відображення даних на цій та подібних цій сторінці використовували таблиці, для яких було додано велика кількість елементів для зручності, серед них пагінація, зміна кількості елементів на сторінці, сортування за колонками, зміна розмірів колонки, приховування колонок, зміна висоти рядка, пошук, кнопка авторозміру сторінки та розширені фільтри. Цих елементів куди більш достатньо аби забезпечити зручне користування таблицею із великою кількістю даних.

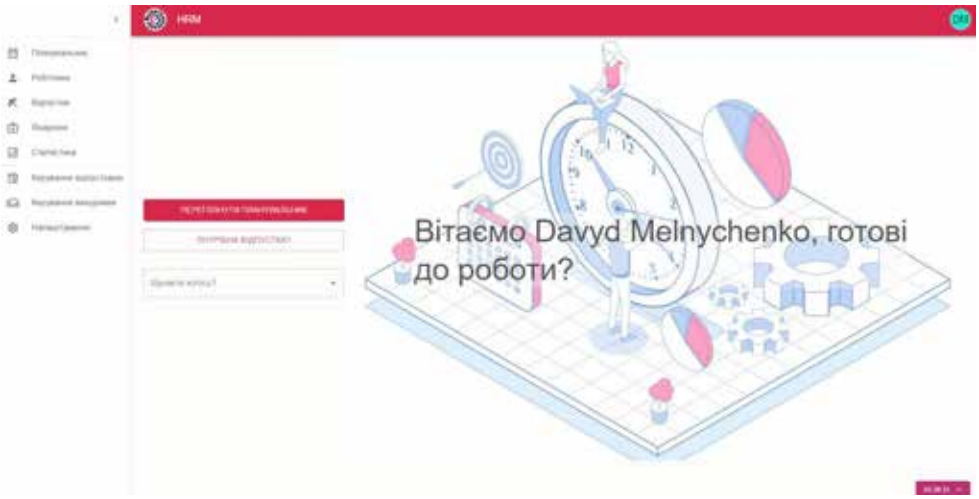

Рис. 3. Головна сторінка

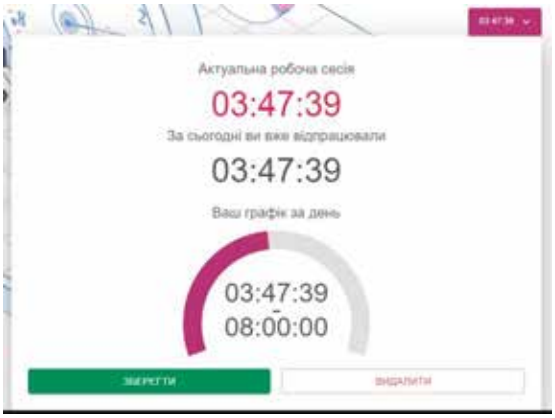

Рис. 4. Відображення відкритої зафіксованої панелі трекера

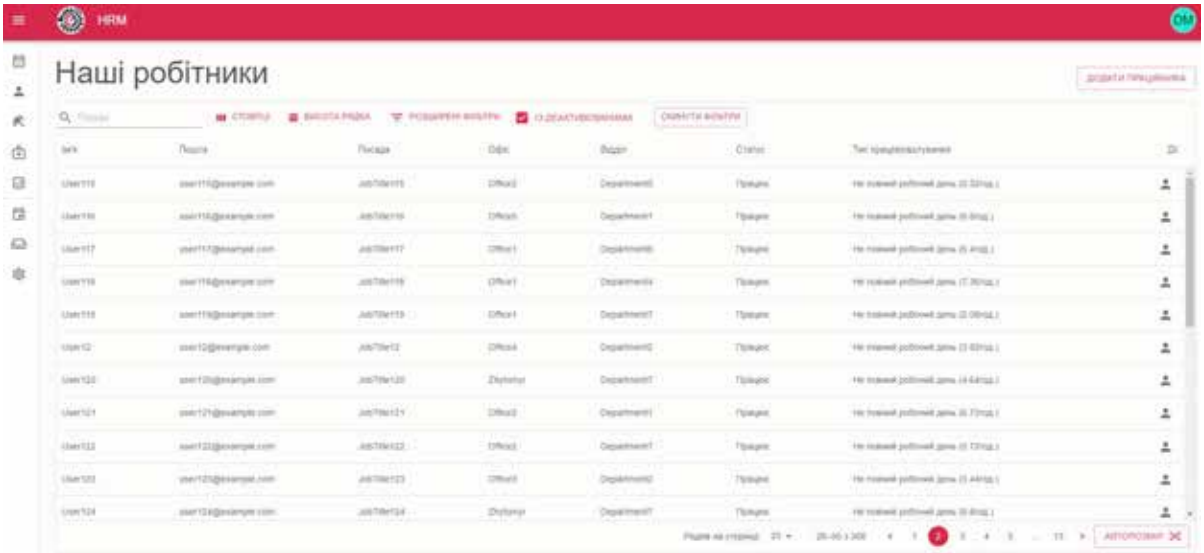

Рис. 5. Сторінка з переліком робітників і таблицею

Для додавання нового користувача в систему теж є окрема сторінка, яка містить форму для введення усіх полів із відповідною валідацією та іншими елементами. Якщо форму заповнює користувач із усіма дозволами, то на форму будуть присутні додаткові поля для встановлення чи редагування дозволів користувача що створюється чи редагується.

Розглянемо функціонал відпусток, яку можна поділити на звичайну сторінку та адміністративну (рис. 6). Сторінки виконані з використанням компонента таблиці, аналогічного до того, який знаходиться на сторінці користувачів. На сторінці відпусток користувач має змогу побачити власні запити на відпустку та їх статус, а також існує можливість відмінити запит поки менеджер не погодив відповідь на нього. На сторінці також вказується кількість часу відпустки, яку вдалося накопичити протягом усього часу, а також кількість зарезервованих днів, у випадку якщо є погоджені відпустки, час яких ще не настав.

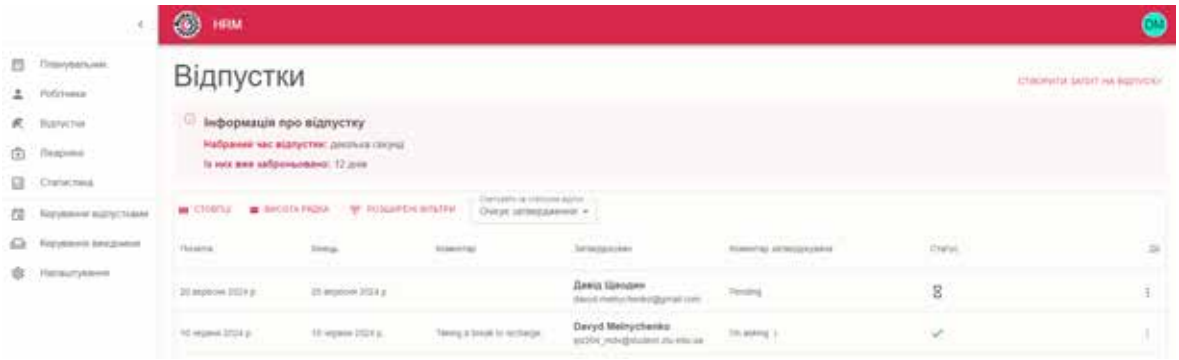

Рис. 6. Сторінка відпусток

Сторінка створення запиту на відпустку має аналогічний елемент інформації про зарезервований і накопичений час (рис. 7). А також додаткові поля валідація запиту, наприклад форма буде відображати попередження якщо користувач створює запит на відпустку, де кількість вказаних днів перевищує кількість накопичених днів. Для затвердження відпусток існує окрема сторінка, доступ до якої мають лише користувачі із відповідним дозволом, сторінка також представлена у вигляді таблиці. Для лікарняних була використана аналогічна схема з компонентом-таблицею із відмінністю у тому, що присутня лише одна сторінка, де користувачі без дозволу можуть лише переглядати інформацію, а з дозволом – редагувати, створювати та видаляти. У випадку із святами відмінність полягає у наявності форми для створення та редагування на самій сторінці поряд з таблицею.

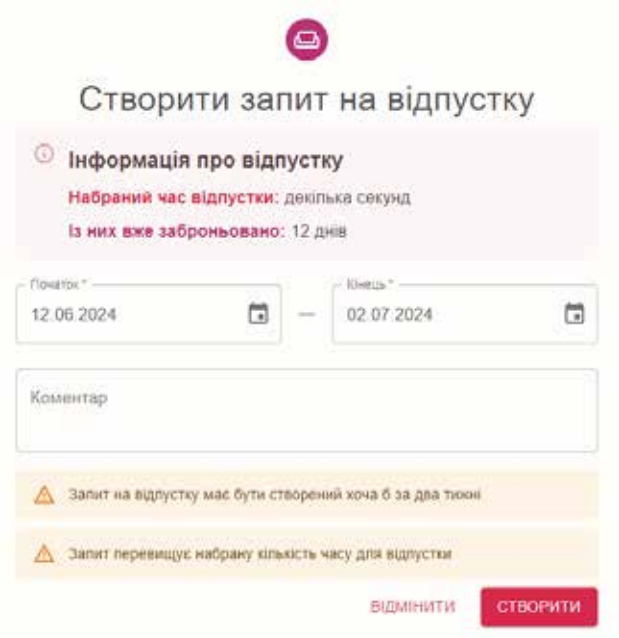

Рис. 7. Форма створення відпустки

Сторінка профілю містить дані про користувача у вигляді карток, поділених на групи для зручності. Меню керування, до складу якого входить наприклад редагування чи деактивація працівника. А також графік, який демонструє початок і кінець робочих сесій користувача протягом місяця і приблизний прогнозований час коли його найвірогідніше можна зустріти на роботі (рис. 8).

Розглянемо сторінку статистики, яка збирає статистичну інформацію про відпрацьовані години для працівників (рис. 9). Сторінка теж використовує компонент таблиці і деякі додаткові елементи, наприклад вибір місяця, загальна інформація про кількість робочих днів у місяці та вихідні в ньому. Зверху присутня кнопка завантаження звіту в вигляді excel-файлу. По кожному користувачу також можна переглянути детальну інформацію, яка подається у вигляді графіків та детального опису проведених обрахунків (рис. 10).

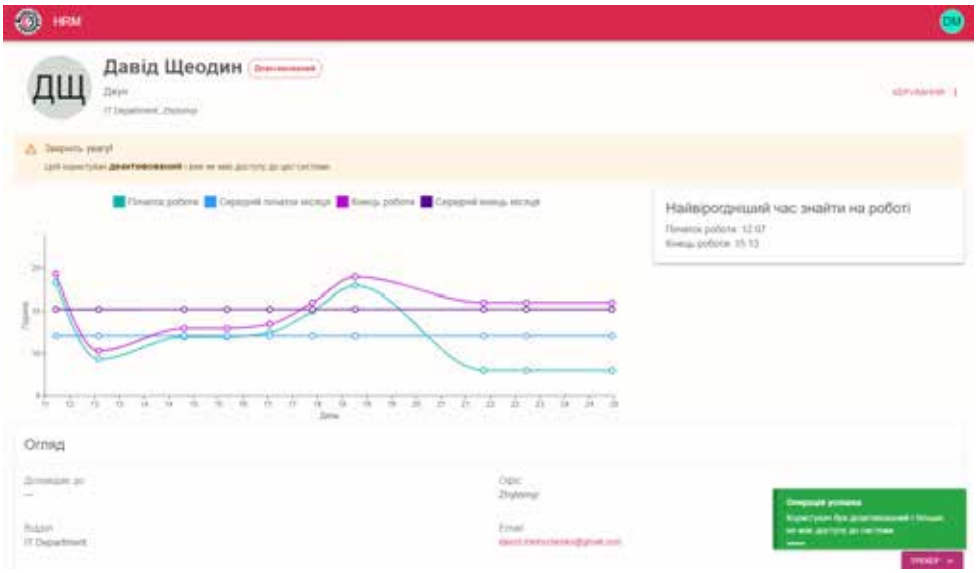

Рис. 8. Сторінка профілю працівника

Рис. 9. Сторінка статистики

Якщо користувач бажає отримати інформацію про робочий час, редагувати його чи звірити її із кимось іншим, він може скористатися сторінкою планувальника. Планувальник представляє собою календар, на якому відображають проведені робочі сесії, свята, відпустки та лікарняні. Календар підтримує перегляд окремого дня, тижня чи місяця, а також режим порядку денного (Agenda). Для зручності календар також підтримує зображення інших користувачів на календарі, що дає змогу порівнювати їх робочі сесії. Також користувачів можна відображувати на різному календарі, використовуючи розділений режим. При потребі користувач може сховати заплановані робочі сесії.

Для додавання інших користувачів, можна скористатися бокової панеллю, яка відображається над календарем на менших екранах (рис. 11–12).

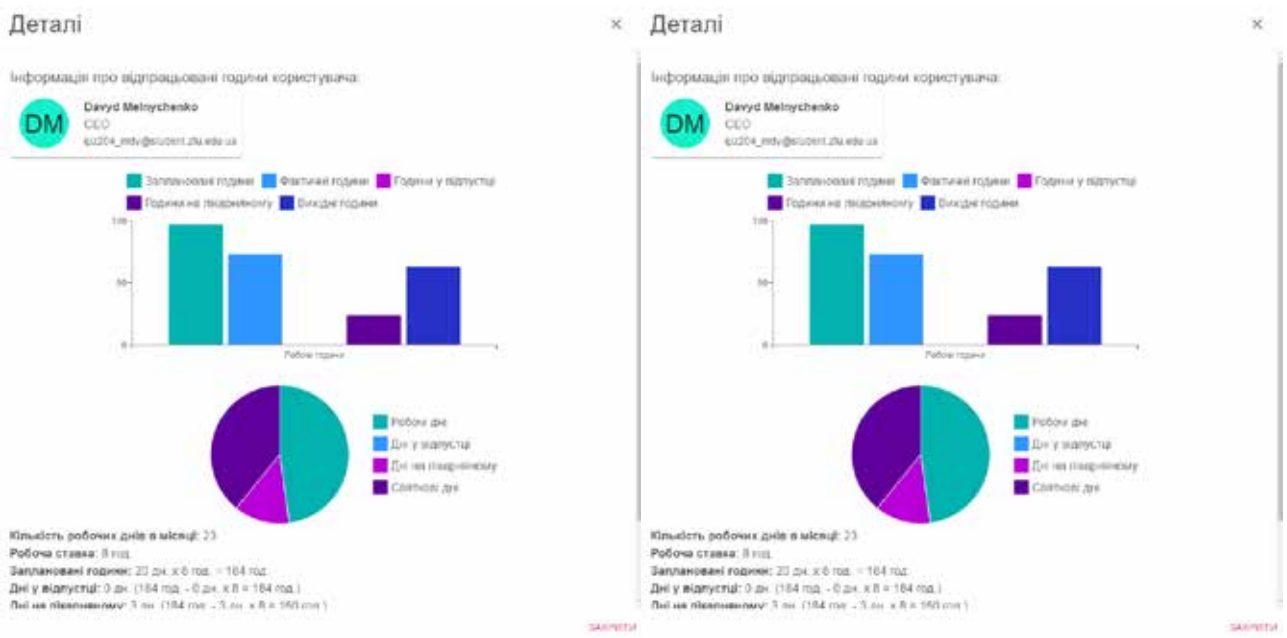

Рис. 10. Детальна статистика для одного користувача

Рис. 11. Сторінка планувальника в режимі перегляду місяця

Рис. 12. Сторінка планувальника в режимі порядку денного

Висновки

Результатом виконання роботи є розроблений і готовий до використання вебзастосунок для управління робочим часом з елементами системи управління персоналом, який забезпечує автоматизацію, оптимізацію та спрощення процесів обліку та планування робочого часу.

В результаті було створено масштабовану, потужну і надійну архітектурц, яка дозволяє проекту і надалі еволюціонувати у напрямку вебзастосунку для управління людським персоналом.

Розроблений вебзастосунок може бути впроваджений у діяльність підприємств різних галузей для автоматизації обліку та планування робочого часу. Це дозволить підвищити ефективність управління робочими процесами, знизити витрати часу на адміністративні процедури та покращити продуктивність працівників. Система також забезпечує точність даних, що сприяє оптимізації внутрішніх процесів компаній та створенню основи для подальшого розвитку повноцінної системи управління персоналом. Вебзастосунок може забезпечити централізоване місце для усієї інформації про робочий час, що полегшує доступ до цієї інформації та її аналіз.

До перспетив подальших досліджень відносимо можливість розширення функціоналу пропонованого додатку шляхом додавання функції автоматичного відстеження часу, яка би допомогла користувачам точніше відстежувати свій робочий час, інтеграції з іншими системами, такими як системи управління проектами або календарі, щоб надати користувачам більше контексту про свою роботу.

Список використаної літератури

1. SQL Sever. User-defined functions [Електронний ресурс]. Режим доступу до ресурсу: https://learn.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16.

2. Мельниченко Д.В. Вакалюк Т.А., Веб застосунок для керування робочим часом: огляд аналогів. Тези Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 13–17 травня 2024 року. Житомир, 2024. С. 71-73. https://conf.ztu.edu.ua/wp-content/uploads/2024/06/sekcija-4.pdf

3. Євтушенко Г. І., Дерев'янко В. М. Аналіз стану управління робочим часом та шляхи підвищення ефективності застосування «Тайм-менеджменту» в організації. Збірник наукових праць Національного університету державної податкової служби України. 2014.№ 1. С. 88-96. Режим доступу: http://nbuv.gov.ua/UJRN/znpnudps_2014_1_12

4. Писаревська Г. І. Використання тайм-менеджменту для підвищення ефективності управління персоналом. Науковий вісник Херсонського державного університету. Сер. : Економічні науки. 2016. Вип. 20(1). С. 148-153. Режим доступу: http://nbuv.gov.ua/UJRN/Nvkhdu_en_2016_20%281%29__38

5. Крикун О. О., Медяник Ю. Г. Вибір менеджером електронних інструментів для створення ефективної системи тайм-менеджменту. Проблеми сучасних трансформацій. Серія: економіка та управління, 11, 2024. DOI: 10.54929/2786-5738-2024-11-04-04

6. Технології тайм-менеджменту в управлінській діяльності державних службовців: монографія / Л.Л. Приходченко, Н.В. Піроженко, М.П. Кернова, І.М Синчак; під заг. ред Л.Л. Приходченко. Одеса: ОРІДУ НАДУ, 2021. 180 с.

7. Примак Т. Ю., Васильчук О. В. Тайм-менеджмент як інструмент підвищення ефективності діяльності туристичного підприємства. Ефективна економіка. 2019. № 12. DOI: 10.32702/2307-2105-2019.12.70

References

1. SQL Server. User-defined functions [Electronic resourse]. URL: https://learn.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16. [in English].

2. Melnychenko D.V., Vakaliuk T.A. (2024) Veb zastosunok dlia keruvannia robochym chasom: ohliad analohiv [Web application for work time management: review of analogues]. Tezy Vseukrainskoi naukovo-praktychnoi on-line konferentsii zdobuvachiv vyshchoi osvity i molodykh uchenykh, prysviachenoi Dniu nauky, 13–17 travnia 2024. Zhytomyr, pp. 71-73. [in Ukrainian].

3. Yevtushenko H.I., Derevianko V.M. (2014) Analiz stanu upravlinnia robochym chasom ta shliakhy pidvyshchennia efektyvnosti zastosuvannia «Taim-menedzhmentu» v orhanizatsii [Analysis of working time management and ways to increase the effectiveness of «Time management» in the organization]. Zbirnyk naukovykh prats Natsionalnoho universytetu derzhavnoi podatkovoi sluzhby Ukrainy, no. 1, pp. 88-96. [in Ukrainian].

4. Pysarevska H.I. (2016) Vykorystannia taim-menedzhmentu dlia pidvyshchennia efektyvnosti upravlinnia personalom [Using time management to increase the efficiency of personnel management]. Naukovyi visnyk Khersonskoho derzhavnoho universytetu. Ser.: Ekonomichni nauky, vol. 20(1), pp. 148-153. [in Ukrainian].

5. Krykun O.O., Medianyk Yu.H. (2024) Vybir menedzherom elektronnykh instrumentiv dlia stvorennia efektyvnoi systemy taim-menedzhmentu [Manager's choice of electronic tools for creating an effective time management system]. Problemy suchasnykh transformatsii. Seriia: ekonomika ta upravlinnia, no. 11. [in Ukrainian].

6. Prykhodchenko L.L., Pirozhenko N.V., Kernova M.P., Synchak I.M. (2021) Tekhnolohii taim-menedzhmentu v upravlinskii diialnosti derzhavnykh sluzhbovtsiv: monohrafiia [Time management technologies in the management activities of civil servants: monograph]. Odesa: ORIDU NADU. [in Ukrainian].

7. Prymak T.Yu., Vasylchuk O.V. (2019) Taim-menedzhment yak instrument pidvyshchennia efektyvnosti diialnosti turystychnoho pidpryiemstva [Time management as a tool to increase the efficiency of a tourist enterprise]. Efektyvna ekonomika, no. 12. [in Ukrainian].