

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ
ОБЛІКУ ВИТРАТ НА ВІДРЯДЖЕННЯ ПРАЦІВНИКІВ ПІДПРИЄМСТВА
SOFTWARE IMPLEMENTATION OF AN APPLICATION FOR TRACKING
BUSINESS TRIP EXPENSES AT AN ENTERPRISE

Виконав: студент(ка) 4 курсу, групи КНТ-128
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Заїкіна К.Є.

(прізвище та ініціали)

Керівник Федорончак Т.В.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Заїкіної Карини Євгеніївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку для організації обліку витрат на відрядження працівників підприємства. Software Implementation of an Application for Tracking Business Trip Expenses at an Enterprise.

керівник проєкту (роботи) Федорончак Тетяна Василівна., к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 27 травня 2022 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Федорончак Т.В., доцент		
Нормоконтроль	Белова А.В., асистент		

7. Дата видачі завдання «28» березня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Заїкіна К.Є.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Федорончак Т.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 104 с., 25 рис., 10 табл., 3 дод., 17 джерел.

КЛІЄНТ-СЕРВЕРНИЙ ЗАСТОСУНОК, ОРГАНІЗАЦІЯ ОБЛІКУ ВИТРАТ НА ВІДРЯДЖЕННЯ, JAVASCRIPT, REACT, C#, .NET CORE, MICROSOFT SQL SERVER.

Об'єкт дослідження – облік витрат на відрядження працівників підприємства.

Предмет дослідження – вебзастосунки для організації обліку витрат на відрядження працівників підприємства.

Мета роботи – створення застосунку для організації обліку витрат на відрядження для користування на підприємстві.

Матеріали, методи та технічні засоби: середовища розробки Microsoft Visual Code, Microsoft Visual Studio 2017, мова програмування Javascript з використанням бібліотеки React для клієнтської частини, мова програмування C# з використанням платформа .NET Core для серверної частини та MSSQL для зберігання даних.

Результати. Створено програмний вебзастосунок для організації обліку витрат на відрядження працівників підприємства.

Висновки. Було виконано аналіз предметної області, розглянуто аналогічні існуючі методи та програмні засоби для вирішення завдання. Розроблено програмний продукт, який призначений для організації обліку витрат на відрядження працівників підприємства.

Галузь використання – проєкт може буде використано для малих та великих підприємства.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 104 pages, 25 figures, 10 table, 3 appendixes, 17 sources.

CLIENT-SERVER APPLICATION, ORGANIZATION OF ACCOUNTING, JAVASCRIPT, REACT, C#, .NET CORE, MSSQL.

The object of study is tracking business trips of employees of the enterprise.

The subject of study is web applications for tracking business trips of employees of the enterprise.

The purpose of the work is to create an application for tracking business trips of employees of the enterprise.

Materials, methods and technical tools: Microsoft Visual Code development environment, Microsoft Visual Studio 2017, Javascript programming language using React library for client part, C # programming language using .NET Core platform for server part and MSSQL for data storage.

Results. The software web application for tracking business trips of employees of the enterprise has been created.

Conclusions. The analysis of the subject area was performed, similar existing methods and software tools for solving the problem were considered. A software product has been developed, which is designed to organize the accounting of business trips of employees of the enterprise.

Scope – the project can be used for small and large enterprises.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	10
1.1 Опис предметної області	10
1.2 Огляд існуючих рішень проблеми	11
1.2.1 Програмний продукт «Smartfin.ua»	11
1.2.2 Програмний продукт «SAP Concur»	13
1.2.3 Програмний продукт «TravelPerk»	14
1.2.4 Порівняльна таблиця існуючих продуктів	16
1.3 Постановка завдання роботи	17
1.4 Висновки до розділу 1	17
2 Опис програми	18
2.1 Вибір основних засобів розробки	18
2.1.1 Мова розмітки гіпертексту — HTML	18
2.1.2 Каскадна таблиця стилів — CSS	18
2.1.3 Мова програмування Javascript	18
2.2 Вибір клієнтського середовища проєктування	19
2.2.1 Управління станом — Redux	20
2.2.2 Virtual DOM React	21
2.3 Вибір серверного середовища проєктування	21
2.4 Вибір бази даних	23
2.5 Вибір середовищ розробки	24
2.6 Архітектура програмного забезпечення	25
2.7 Вимоги до програмного забезпечення	26
2.8 Висновки до розділу 2	28
3 Розробка програми	29
3.1 Структура бази даних	29
3.2 Алгоритм функціонування застосунку	32

3.3 Структурна схема застосунок	32
3.3.1 Структурна схема клієнтської частини	33
3.3.2 Структурна схема серверної частини	34
3.4 Опис реалізованих функцій	35
3.4.1 Опис функцій клієнтської частини	35
3.4.2 Опис функцій серверної частини	37
3.5 Висновки до розділу 3	38
4 Експлуатація, тестування та експериментальне дослідження програми	39
4.1 Опис методики проведення тестування	39
4.2 Виконання програми	40
4.3 Повідомлення користувачу	47
4.4 Висновки до розділу 4	48
Висновки	49
Перелік джерел посилання	50
Додаток А Технічне завдання	52
Додаток Б Текст програми	57
Додаток В Слайди презентації	95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;
AWS – Amazon Web Services;
CSS – Cascading Style Sheets;
DBA – Database administrator;
DOM – Document Object Model;
HTML – HyperText Markup Language;
IDE – Integrated Development Environment;
JS – JavaScript;
MSSQL – Microsoft SQL Server;
MVC – Model-View-Controller;
SEO – Search Engine Optimization;
UI – User Interface;
VS – Visual Studio;
БД – база даних;
СКБД – система керування базами даних;
ПЗ – програмне забезпечення;
ПК – персональний комп'ютер.

ВСТУП

Сьогодні сучасні засоби автоматизації дозволяють компаніям істотно скорочувати час і гроші на вирішення найрізноманітніших бізнес-задач, в тому числі і таких вузькоспеціалізованих, як оформлення відряджень.

Стандартний процес оформлення відрядження робітника включає в себе заповнення відповідної заявки, бронювання опцій по поїзді, узгодження документації та бюджету з керівником, а також отримання авансового платежу на службові витрати. Після повернення з відрядження працівник також зобов'язаний підготувати пакет документів для звіту з понесених в ній витрат, узгодити його з бухгалтерією, і, в разі потреби, отримати грошову компенсацію.

Актуальність роботи полягає в тому, що кожен етап цього бізнес-процесу включає в себе безліч дій, які без належної оптимізації віднімають у всіх задіяних осіб масу робочого часу, і які стає практично неможливо проконтролювати належним чином. За підсумком все це виливається компанії в зайві витрати, і чим більше в ній відряджень, тим істотніше їх сума - ось чому автоматизація цього бізнес-процесу набуває все більшої популярності.

Створення застосунку для організації обліку витрат на відрядження значно скоротить час працівникам, буде доступний у будь-який час та з будь-якого пристрою (при наявності Інтернету)

Метою дипломної кваліфікаційної роботи є оцінка та аналіз існуючих засобів та програмна реалізація застосунку для організації обліку витрат на відрядження працівників підприємства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасному цифровому світі ми можемо в будь-який момент спілкуватися в різних часових поясах. Однак, коли справа доходить до укладання великої угоди або сприяння міцним і довготривалим діловим відносинам, ніщо не може замінити спілкування віч-на-віч.

Корпоративні подорожі все ще дуже важливі для всіх підприємств. Але багато організацій, здається, зіпсовані застарілою практикою та неефективною обробкою витрат. Це в кінцевому підсумку може призвести до втрати дорогоцінного часу та ресурсів. На щастя, сьогодні ці проблеми можна вирішити за допомогою програмного забезпечення для організації обліку витрат на відрядження. Розглянемо ряд визначень і спробуємо визначити ключові характеристики відрядження.

Відрядження – це подорожі, призначені спеціально для робочих цілей, виконання службового доручення керівника підприємства, працівником.

Облік витрат на відрядження — це відповідний інструмент для управління витратами на корпоративні подорожі.

Витрати на відрядження поділяються на:

- витрати на харчування;
- транспортні та проїзні витрати: квитки на потяг або літак, витрати на паркування, оренда автомобіля;
- проживання у відрядженнях;
- витрати на розваги (наприклад, ділові обіди з клієнтами) та подарунки клієнтам.

Звіт про витрати – це узагальнення всіх витрат, які здійснювались працівниками та керівниками у рамках їхньої тимчасового відрядження. Витрати підкріплені регулярними квитанціями, повинні відшкодовуватись підприємством. В результаті відшкодування витрат включаються в рахунки підприємства.

Основна мета програми:

- оформлення та узгодження заявок на відрядження;
- розрахунок витрат на відрядження та побудова звіту;
- зберігання даних про користувачів, заявок та довідників.

Отже, основна мета будь-якого застосунку для організації обліку витрат на відрядження в обслуговуванні всіх користувачів таким способом, який займе в них менше часу, та буде надавати всі необхідні можливості. Кінцевий результат використання програми полягає в підштовхуванні людей на повторне використання програми.

1.2 Огляд існуючих рішень проблеми

Для вирішення такої проблеми з'явилися вебзастосунки для організації обліку витрат на відрядження, які скорочують час працівникам та надають можливість лише в кілька кроків оформити заявку на відрядження.

1.2.1 Програмний продукт «Smartfin.ua»

Одним з аналогів розроблюваної програми є програма «Smartfin.ua» [1]. «Smartfin.ua» - це вебзастосунок для розрахунку витрат на відрядження. Це спеціальний застосунок зі зручним і інтуїтивно зрозумілим інтерфейсом. У ньому всього в кілька дій робітник формує заявку на відрядження: обирає працівника і тривалість ділової поїздки. Після збереження заявки, вона відображається в особистому кабінеті бухгалтера з розрахованими витратами, відповідно до заробітної плати працівника. Після цього повідомлення про створення заявки надсилається на електронну пошту працівника. Той переглядає заявку і погоджується з нею, якщо все оформлено правильно. На рисунку 1.1 наведено головну сторінку вебзастосунку програми «Smartfin.ua».

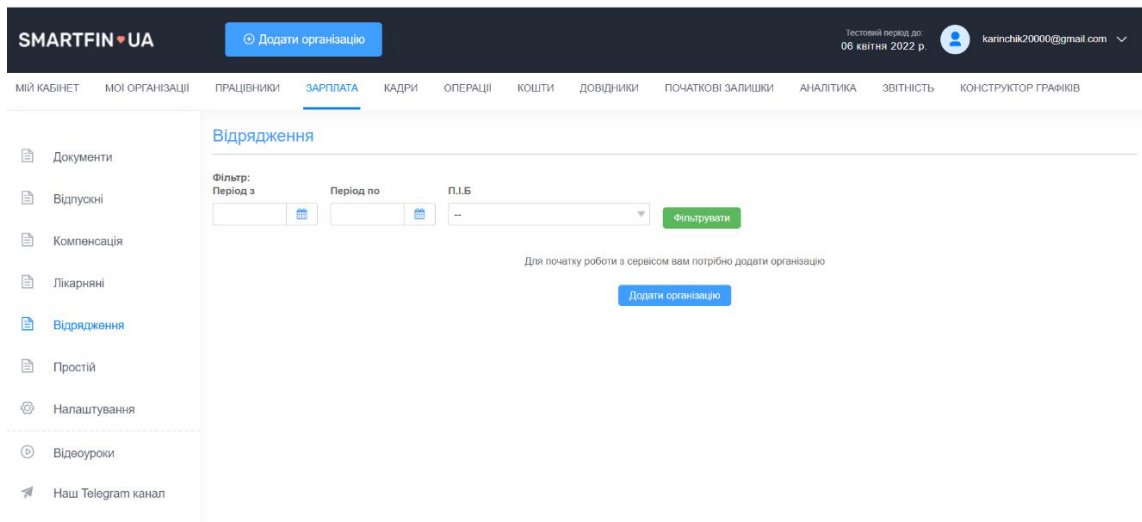


Рисунок 1.1 – Головна сторінка вебзастосунку «Smartfin.ua»

Переваги «Smartfin.ua»:

– мобільність. Програмний інструмент можна використовувати вдома, в дорозі, на роботі на смартфоні, планшеті або персональному комп'ютері - коли і де завгодно;

– інтегрованість. Якщо на підприємстві вже існує зручна ERP-система, тоді інструмент стане її доповненням, вдало впишеться в діючу архітектуру і не зажадає спеціальних хитрувань по налаштуванню і налагодженні. Для користування потрібно лише обрати відповідне підприємство;

– зовнішня взаємодія. Програму виконує необхідні функції. Можливість створення заявки, розрахунку витрат та подальше формування звітності.

Застосунок для розрахунку витрат на відрадженьня володіє важливою перевагою - з ним зручно складати авансовий звіт. Всі заявки на відрадженьня працівників зберігаються в системі і бухгалтер має можливість сформувати звіт за відповідним періодом або працівником в кілька дій прямо в вебзастосунку.

Головний і, можливо, єдиний недолік програми - платність. Безкоштовно «Smartfin.ua» допускається використовувати 3 дні, для одного підприємства плата 275 грн на місяць.

1.2.2 Програмний продукт «SAP Concur»

Наступним аналогом є «SAP Concur» [2]. «SAP Concur», призначений для керування подорожами, розроблений спеціально для відповідності їхнім потребам і вимогам. SAP concur має інструменти та мережу туристичних постачальників, які можуть знадобитися компанії, щоб реалізувати свою мету щодо подорожей та витрат [3].

Рішення може ефективно отримувати дані про подорожі з будь-якої точки світу, незалежно від того, де воно було заброньовано. Воно також об'єднує всі важливі дані на одній всеосяжній інформаційній панелі для легкого доступу та звітності. Програмне забезпечення також виконує процес керування поїздками таким чином, що не гарантує порушення існуючих правил подорожей [3].

З іншого боку, програмне забезпечення допомагає отримувати дані про витрати з кількох джерел і надає кращу інформацію для прийняття логічних рішень. Це полегшує дотримання політики та гарантує, що працівникам відшкодовуються витрати без затримок [3].

На рисунку 1.2 наведено користувацький інтерфейс програми «SAP Concur».

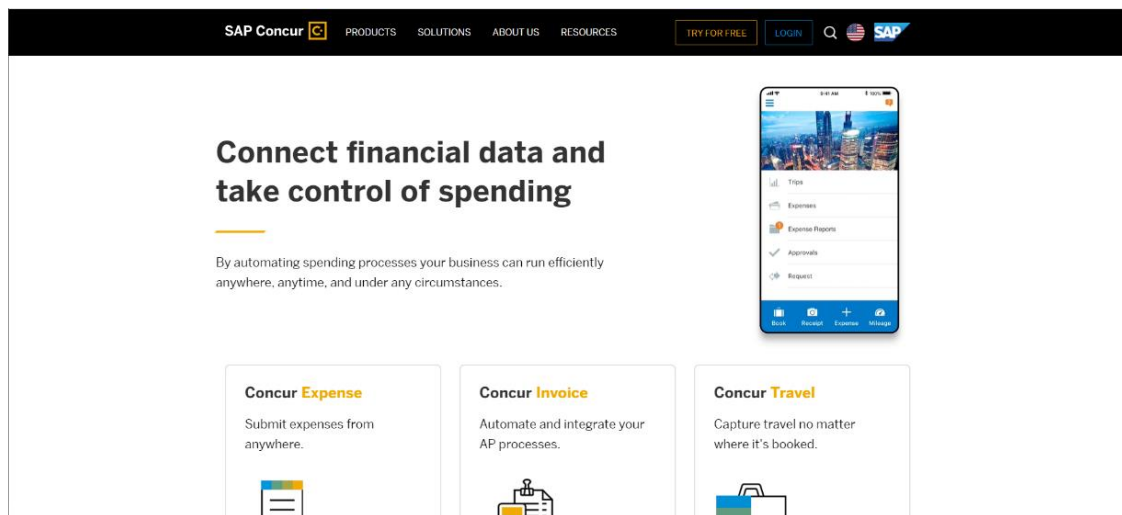


Рисунок 1.2 – Головна сторінка сайту «SAP Concur»

Переваги «SAP Concur»:

- запис даних про подорожі, незалежно від того, де вони заброньовані;
- перегляд зведених даних про подорожі на одній інформаційній панелі;
- легке бронювання авіа, залізничних рейсів, готелю чи автомобілю за допомогою одного інструменту онлайн-бронювання;
- дотримання легкої політики для робітників;
- доступ бронювати подорожі робітникам самостійно [3].

Недоліки «SAP Concur»:

- користування застосунком можливо лише за придбанням відповідного тарифного плану (безкоштовне користування лише 7 днів).

1.2.3 Програмний продукт «TravelPerk»

Ще одним аналогом є програма «TravelPerk» [4]. TravelPerk — об'єднує все, що стосується ефективного управління діловими подорожами, на одній надійній платформі. Інструмент можна використовувати для

багатьох цілей, від бронювання авіаквитків та розміщення до керування квитанціями на відрядження [3].

Програмне забезпечення робить неймовірно простим отримання звітів про витрати та корисну інформацію про витрати компанії на подорожі. Наявний мобільний застосунок, що робить процес бронювання значно зручнішим [3].

Застосунок надає доступ до, можливо, найбільшого переліку авіарейсів, готелів та орендованих транспортних засобів з усього світу. Крім того, TravelPerk автоматично встановлює політику подорожей, тому керівництву не доведеться турбуватися про дотримання вимог під час адміністрування відряджень. На рисунку 1.3 наведено користувацький інтерфейс програми «TravelPerk» [3].

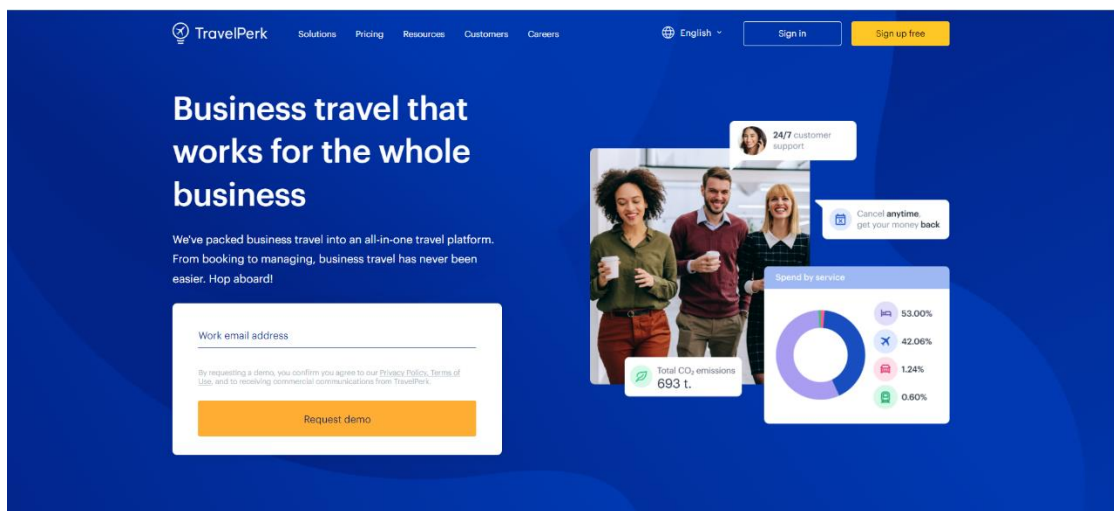


Рисунок 1.3 – Головна сторінка сайту «TravelPerk»

Сильні сторони «TravelPerk»:

- автоматичне створення та відстеження звітів про витрати;
- діюча інформація щодо показників звітності;
- автоматичне встановлення правил подорожей;
- доступ до величезного переліку авіарейсів, готелів та орендованих транспортних засобів [3].

Недоліком є те що програма платна та працює за тарифним планом, безкоштовна версія надає лише 3 можливості використання:

- зведене виставлення рахунків;
- звіти про подорожі;
- обмежений доступ до авіарейсів, готелів, потягів, автомобілів.

1.2.4 Порівняльна таблиця існуючих продуктів

Вище було наведено найбільш популярні на сьогоднішній день сервіси та програми, які дозволяють вести облік витрат на відрядження працівників підприємства. Але крім основних переваг, усі вони мають і недоліки. Наприклад, всі застосунки, вимагають вкладення коштів.

Також дійсно існують переваги характерні для всіх програм: всі програми мають простий та зручний інтерфейс, виконують багато функцій, таких як можливість контролювати витрати на відрядження, складання заявок на відрядження та формування авансового звіту витрат на відрядження.

В таблиці 1.1 проведено порівняння вже існуючих програмних продуктів.

Таблиця 1.1 — Порівняння програмних продуктів

	«Smartfin.ua»	«SAP Concur»	«TravelPerk»
Виконання основного функціоналу (створення заявок, формування звіту)	+	+	+
Багатоплатформність	+	+	+
Зручний інтерфейс	+	+	+
Платність	+	+	+
Підтримка української мови	+	-	-
Швидкість виконання функціоналу	+	-	+

Продовження таблиці 1.1

	«Smartfin.ua»	«SAP Concur»	«TravelPerk»
Адаптація під мобільні пристрої	+	+	+
Співпраця та доступ до серверів замовлення (бронювання готелів, автомобілів, та ін.)	-	-	+

1.3 Постановка завдання роботи

Оскільки завданням роботи є реалізація застосунку для організації обліку витрат на відрядження працівників підприємства, можна виділити такі задачі для виконання:

- проаналізувати предметну область;
- розробити функціональну структуру програми;
- проаналізувати логічну структуру програми;
- розробити структуру бази даних;
- розробити інтерфейс;
- обрати мову програмування та середовище розробки;
- виконати програмну реалізацію;
- провести тестування та виправлення помилок.

1.4 Висновки до розділу 1

В даному розділі було досліджено предметну область та проаналізовано аналоги вебзастосунків для обліку витрат на відрядження, визначено основні функції (створення заявки на відрядження, узгодження заявки, формування звіту за заявкою), які повинен мати програмний продукт. Також було наведено основні завдання розробки.

2 ОПИС ПРОГРАМИ

2.1 Вибір основних засобів розробки

2.1.1 Мова розмітки гіпертексту — HTML

HTML (Hypertext Markup Language) — це мова розмітки, яка використовується для оформлення інтерфейсної частини вебсторінок. HTML — це мова розмітки, яка поєднує гіпертекст з розміткою. Термін «гіпертекст» відноситься до сполучного посилання між вебсторінками. Мова розмітки використовується для створення архітектури текстової документації під тегами. Таким чином, вміст сторінки структурується певним чином у вигляді заголовків, підзаголовків, основного тексту, зображень тощо [5].

2.1.2 Каскадна таблиця стилів — CSS

CSS — це проста мова, розроблена для полегшення процесу адаптації вмісту макета на вебсторінці. Частину коду CSS можна легко підключити до HTML-коду, який автоматично призначає певні стилі та форматує вебсторінки [6].

2.1.3 Мова програмування Javascript

JavaScript — це текстова мова сценаріїв і програмування з відкритим кодом, яка використовується для підвищення інтерактивності вебсайтів. Відома своєю безпекою та переносимістю для роботи практично в будь-якій операційній системі, вона має можливість вставляти динамічний текст прямо в HTML вебсторінки та автоматично запускатися під час завантаження сторінки [7].

JavaScript використовується не тільки для написання інтерфейсу та серверного коду для серверів, щоб зробити вебсайти більш інтуїтивно зрозумілими, але також використовується як на стороні клієнта, так і на

стороні сервера вебсайту, що робить його дійсно універсальним інструментом, який використовується на різних пристроях і платформах [7].

Фреймворки та бібліотеки JavaScript стали поширеними у веброзробці, оскільки вони забезпечують простий спосіб створення компонентів JavaScript, найпопулярнішими серед яких є React, Vue, Svelte, Angular, Ember та інші.

2.2 Вибір клієнтського середовища проєктування

Фронтенд застосунку – це частина, з якою користувач безпосередньо взаємодіє. Зазвичай її називають «клієнтською стороною» програми [8].

Для розробки клієнтської частини програми було обрано бібліотеку ReactJS.

ReactJS — це бібліотека JavaScript, розроблена Facebook для розробки клієнтської частини застосунків в 2013 році. Вона використовується для створення односторінкових мобільних або вебзастосунків. Це потужна бібліотека, оптимальна для отримання даних, які швидко змінюються. Її основна функція - рендеринг даних [9].

Однак ReactJS — це полегшена бібліотека перегляду, яка підтримує лише компоненти інтерфейсу користувача. Тому для отримання повноцінної архітектури потрібна інтеграція з іншими технологіями та інструментами (наприклад, Flux і Redux) [9].

Переваги ReactJS:

- ReactJS використовує віртуальний DOM, що покращує роботу користувачів. Це також робить роботу розробника менш складною;
- JSX використовується в ReactJS, який дуже простий і легкий у освоєнні;
- немає необхідності в окремих файлах для логіки та розмітки в ReactJS;

- ReactJS — бібліотека з відкритим вихідним кодом, яку підтримує Facebook. Це бібліотека, яка постійно розвивається;
- прив’язка до одних даних робить код дуже стабільним;
- ReactJS також надає мобільне рішення, відоме як React Native;
- ReactJS швидший для візуалізації в порівнянні з іншими вебфреймворками;
- дружнє для SEO;
- має чудовий набір інструментів розробника [9].

2.2.1 Управління станом — Redux

Redux — це безкоштовна бібліотека керування станом, яка працює на інтерфейсі програм JavaScript, керуючи станом кожного компонента в інтерфейсі користувача. Бібліотека Redux полегшує поділ між кодом, який керує та зберігає дані в програмі, і кодом, який керує подіями та їх впливом на різні компоненти інтерфейсу програми [10].

React-Redux — це офіційна бібліотека прив’язки Redux UI для програм React, яка підтримується командою Redux [10].

Архітектура Redux зображена на рис. 2.1.

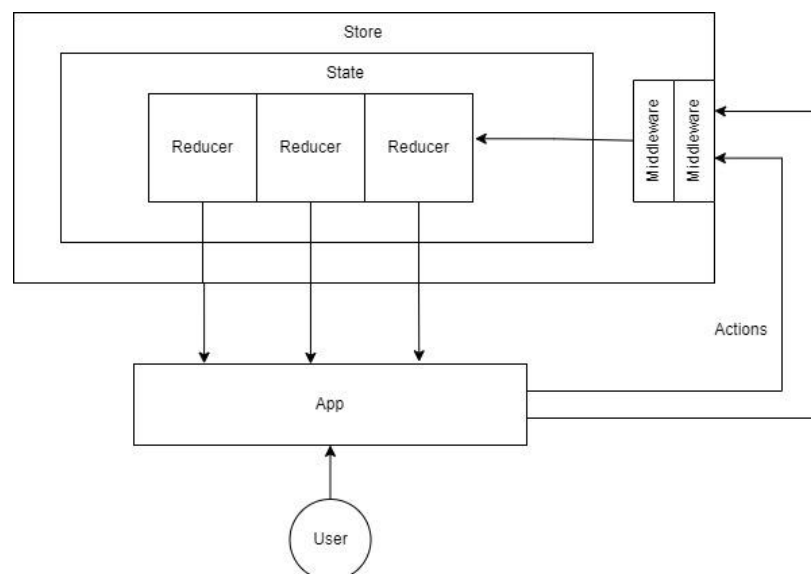


Рисунок 2.1 – Архітектура Redux

Store – зберігає всі дані, які використовує програма [11].

Middleware – надає можливість взаємодії з об'єктами Actions відправлені до Store перед тим, як вони відправляються до Reducer [11].

Reducer – функція, яка повертає деякі дані стану. Викликається дією type [11].

Actions – повідомляє Reducer, що потрібно змінити стан, містить type, додатково може містити payload (деякі дані) [11].

2.2.2 Virtual DOM React

DOM просто розшифровується як Document Object Model. DOM схожий на карту вашого інтерфейсу користувача (UI) на основі різних вузлів і представлений у вигляді деревоподібної структури даних [12].

Цей звичайний DOM оновлюється, щоб показувати зміни інтерфейсу переглядача щоразу, коли відбувається зміна на сторінці. З іншого боку, віртуальна DOM є - як впливає з назви - віртуальним представленням DOM [12].

Віртуальний DOM — це той, який спочатку оновлюється, коли стан вашої програми змінюється. Потім, коли в застосунку вноситься нова зміна, створюється новий знімок екрана віртуальної DOM і порівнюється з попереднім. Це триває з кожною зміною [12].

Якщо є багато оновлень DOM, це повільно та займає багато пам'яті. Саме через це ReactJS використовує віртуальний DOM [12].

2.3 Вибір серверного середовища проєктування

Бекенд також відомий як серверна частина вебсайту або програми. Він організовує та зберігає дані, а також забезпечує належне функціонування всього на клієнтській частині вебсайту [8].

Для реалізації серверної частини було використано мову програмування C# та платформу .NET Core.

C# — це універсальна мова програмування, яку можна використовувати для виконання різноманітних завдань. Вона використовується для створення вебзастосунків, десктоп застосунків, ігор, та іншого. C# підтримує різноманітні стилі програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. В результаті C# є потужною і адаптованою мовою програмування [13].

Синтаксис C# дуже схожий на C, Java або навіть JavaScript. Сам синтаксис досить простий в розумінні і читанні [13].

.NET існує протягом тривалого часу і за останні 15 років досить сильно змінився і покращився. Протягом років відзначаються такі прекрасні нововведення як MVC, узагальнення, LINQ, async / await оператори і багато іншого [13].

Одним з найбільш значущих плюсів C # в тому числі і .NET в цілому, є його багатогранність. Можна писати програми для ПК, вести веброзробку, створювати фонові сервіси або навіть мобільні застосунки. Крім того, щоб скомпонувати всі UI-коди разом, це, крім C #, всього лише трохи JavaScript'a (+TypeScript) [13].

Універсальність мови — досить вагомий плюс, так як ваш внесок в його вивчення може знайти застосування в широкому спектрі можливостей. Навички розробника дуже мультиплатформенні. Якщо побажає, розробник можете легко «перескочити» з розробки вебзастосунків на мобільні. Мабуть, це унікальна відміна від інших мов, заточених тільки під серверну частину [13].

В таблиці 2.1 наведено детальне порівняння мов програмування.

Таблиця 2.1 – Порівняння мов програмування

	JS+C#.Net	JS+PHP
Спосіб виконання	Компілюється	Інтерпретується
Багатопоточність	+	-
Асинхронність	+	-
Захист даних від sql-ін'єкцій	+	-
Колекції	Безліч різних колекцій	Тільки масиви
Типізація	+	-
Розмір проєктів	Придатний для великих проєктів	Кращий для невеликих проєктів
Швидкість бібліотек доступу до бази даних	Швидше	Повільніше
Обсяг коду	Трохи менш компактний	Компактний
Відлагодження	Легше проводити відлагодження	Складно проводити відлагодження
Середовище розробки	IDE високого класу	IDE середнього класу
Швидкість	Виконується на багато швидше, коли побудований в режимі випуску, оптимізований, кешований	Виконується досить швидко

2.4 Вибір бази даних

SQL Server — це система керування реляційною базою даних (RDBMS), розроблена та представлена Microsoft. Вона включає мову SQL і Transact-SQL (T-SQL), власну мову Microsoft з можливостями обробки винятків, оголошення змінної та збережених процедур [14].

2.5 Вибір середовищ розробки

Велику роль під час створення програмного забезпечення, крім інструментів для написання коду і розгортання системи, грають засоби для автоматизації процесу розробки.

У якості середовища (IDE) для розробки клієнтської частини було обрано – Visual Studio Code з набором відповідних плагінів.

Visual Studio Code — це безкоштовний, легкий і потужний редактор коду для Windows, Mac і Linux, заснований Electron/Chromium. Він має вбудовану підтримку JavaScript, TypeScript і Node.js і багату екосистему розширень, яка додає Intellisense, налагодження, підсвічування синтаксису для багатьох мов (C++, Python, Go, Java). Він працює без зайвих налаштувань [15].

Можливості Visual Studio Code:

- налагодження;
- розширення IntelliSense;
- вбудований термінал;
- редагування файлів проєкту;
- функції керування кодом;
- вбудована інтеграція Git [16].

Для розробки серверної частини було віддано перевагу Visual Studio версії 17.0.

Особливості Visual Studio:

- має хорошу функцію навігації;
- VS допомагає кращому пошуку, фільтрації та попередньому перегляду коду, над яким ви працюєте;
- має кращі налаштування і може включати сторонні розширення;
- можна створювати програми за допомогою HTML, JavaScript, C#, .NET (Xamarin), C++ тощо [17].

В таблиці 2.2 наведено детальне порівняння середовищ розробки.

Таблиця 2.2 – Порівняння середовищ розробки

	Visual Studio 2017	Visual Studio Code	IntelliJ Idea
Багатоплатформність	+	+	-
Швидкість модернізації	+	+	+
Наявність візуального конструктора	+	+	+
Підтримка репозиторіїв	+	+	+
Автозавершення	+	+	+
Розширення	+	+	+
Відлагодження програми	+	+	+
Навігація по проєкту	+	+	+
Робота з базами даних	+	+	-
Компілятор	+	+	+
Перевірка орфографії	+	+	+
Рефакторинг	+	+	+
Статичний аналіз коду	+	+	+
Портативність	-	-	-
Займає невелику кількість пам'яті	-	+	-

2.6 Архітектура програмного забезпечення

Для написання вихідного коду клієнтської частини застосунку було обрано мову JavaScript з використання UI бібліотеки React та інструментом управління станом Redux. Для реалізації серверу як вебсервіс було обрано Rest API, реалізований на мові C# та використано платформу .NET Core 2.2. Для системи управління базами даних було обрано Microsoft SQL Server.

Загальна архітектура програмного забезпечення зображена на рис. 2.2

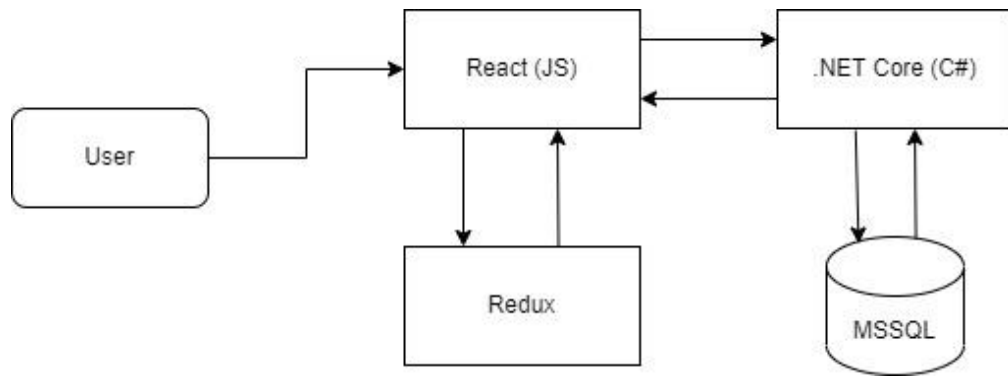


Рисунок 2.2 – Загальна архітектура ПЗ

2.7 Вимоги до програмного забезпечення

Програмний продукт повинен забезпечувати виконання нижче перерахованих функцій:

- аутентифікація та надання особі можливостей у відповідність до покладених йому прав в залежності від його ролі.

Для робітника:

- створення заявки на відрядження;
- перегляд узгодженої заявки з сумою відрядних та можливістю перегляду маршруту на карті.

Для бухгалтера:

- ведення нормативно-довідкової інформації (довідник статусів користувачів, довідник країн, довідник міст, довідник валют з поточним курсом, довідник лімітів по статусах);

- розрахунок тарифів на відрядження для кожного робітника, в залежності від його статусу на підприємстві (для розрахунку щоденних витрат, беремо початкове значення з довідника статусів (для кожного статусу різні показники витрат) та помножуємо на кількість днів відрядження, для транспортних витрат, беремо початкове значення та помножуємо на транспортний коефіцієнт з довідника міст, для мобільних, представницьких, непередбачених та витрат на проживання, значення беремо з довідника

стутусів, якщо робітник відправляється за кордон, кожний з тарифів ділимо на курс);

- відправка сформованої заявки на електронну пошту робітника;
- авансовий звіт за заявкою з можливістю завантаження.

Для адміністратора:

- додавання користувачів системи;
- видалення користувачів системи;
- коригування даних по користувачу.

Схема функцій користувача представлена на рис. 2.3

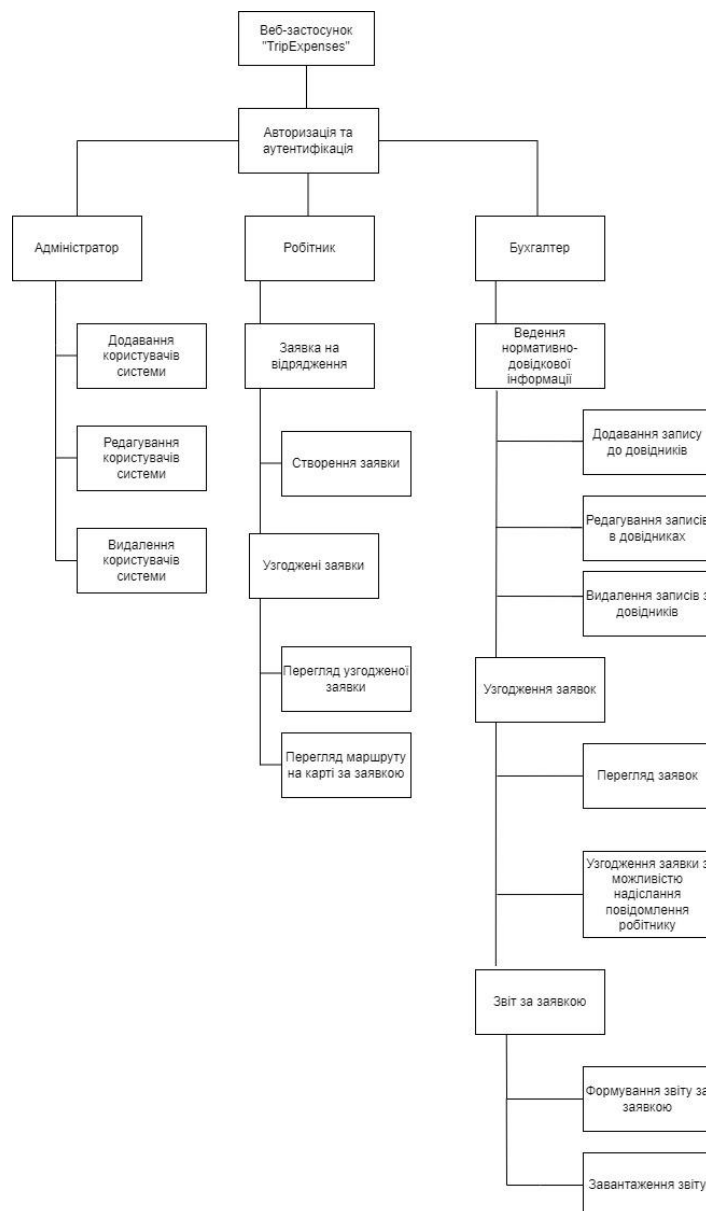


Рисунок 2.3 – Схема функцій користувача

При цьому ПЗ, що розробляється, повинно бути реалізовано у вигляді вебзастосунку, підтримувати українську мову та доступ до якого повинен бути можливий на будь-якому пристрої, у якому присутній доступ до мережі інтернет.

2.8 Висновки до розділу 2

У даному розділі було розглянуто основні інструменти для розробки вебзастосунку, такі як HTML, CSS, JavaScript, розглянуто відмінності між клієнтською та серверною частин програми.

Було представлено загальну архітектуру програмного забезпечення та сформульовано функціональні вимоги до розроблюваного застосунку.

Та після порівняння мов програмування було обрано JS (з використання React, Redux) + C#.Net для розробки ПЗ, так як воно набагато перевершує JS+PHP та інші мови програмування. Також було наведено детальне порівняння середовищ розробки Microsoft Visual Studio, Microsoft Visual Studio Code, в ході чого було виявлено, що обрані середовища розробки володіють багатьма фічами, та найкращі для розробки програмного забезпечення на мові Javascript та C#. Базою даних було обрано Microsoft SQL Server.

3 РОЗРОБКА ПРОГРАМИ

3.1 Структура бази даних

База даних проєкту складається з шести сутностей: країни, міста, курси валют, заявки користувачів, статуси користувачів, користувачі. Взаємодія між сутностями представлена на рисунку 3.1.

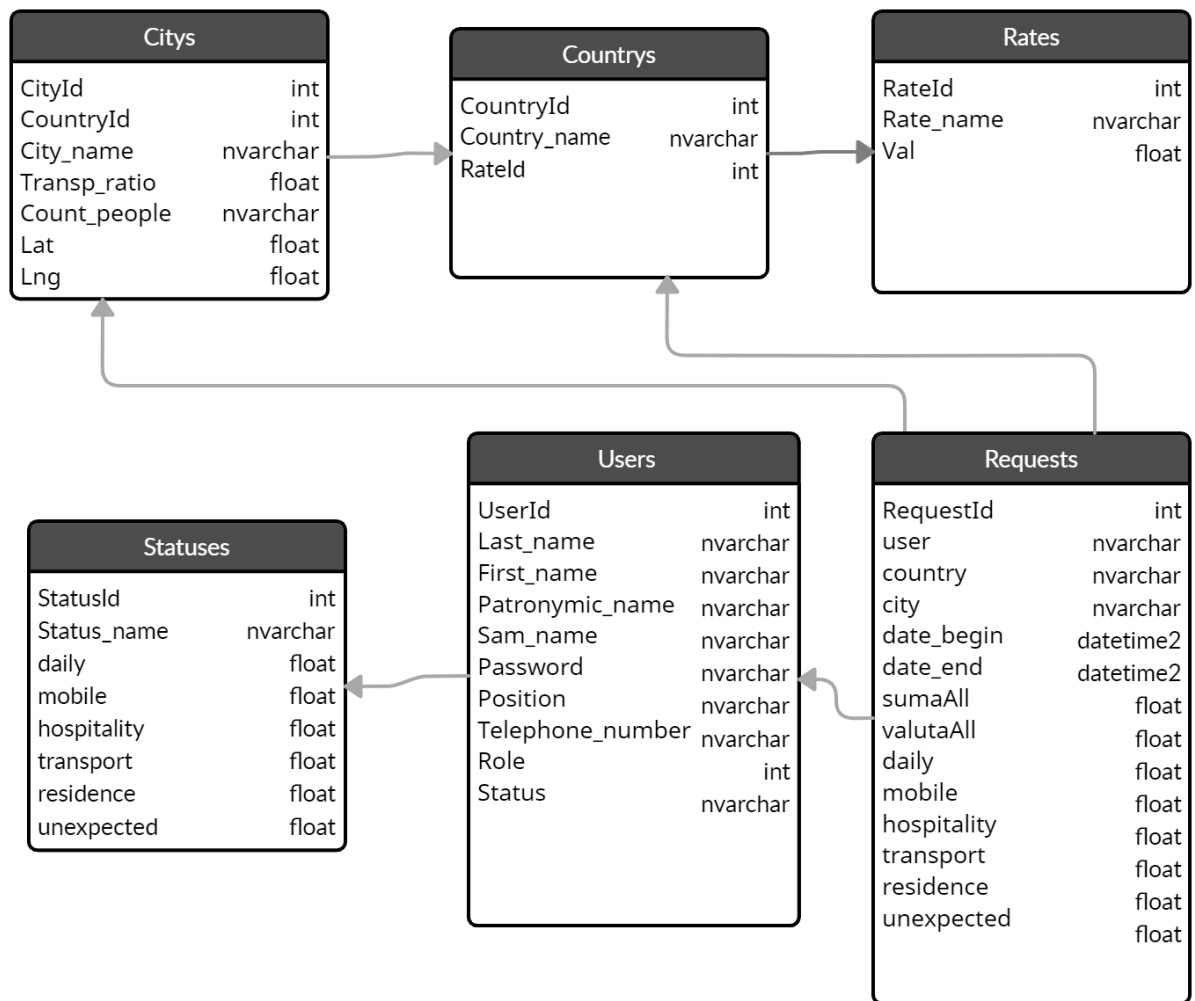


Рисунок 3.1 – Взаємодія між сутностями

Сутність Countrys (CountryId, Country_name, RateId) – сутність для таблиці бази даних країн.

Опис полів сутності та їх типу наведено у таблиці 3.1.

Таблиця 3.1 – Опис полів сутності Countrys

№	Назва поля	Тип даних	Розмір
1	CountryId	int	-
2	Country_name	nvarchar	MAX
3	RateId	int	-

Сутність Citys (CityId, CountryId, City_name, Transp_ratio, Count_people, Lat, Lng) – сутність для таблиці бази даних міст.

Опис полів сутності та їх типу наведено у таблиці 3.2.

Таблиця 3.2 – Опис полів сутності Citys

№	Назва поля	Тип даних	Розмір
1	CityId	int	-
2	CountryId	int	-
3	City_name	nvarchar	MAX
4	Transp_ratio	float	53
5	Count_people	nvarchar	MAX
6	Lat	float	53
7	Lng	float	53

Сутність Rates (RateId, Rate_name, Val) – сутність для таблиці бази даних курсів валют.

Опис полів сутності та їх типу наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності Rates

№	Назва поля	Тип даних	Розмір
1	RateId	int	-
2	Rate_name	nvarchar	MAX
3	Val	float	53

Сутність Requests (RequestId, user, country, city, date_begin, date_end, sumaAll, valutaAll, daily, mobile, hospitality, transport, residence, unexpected) – сутність для таблиці бази даних заявок.

Опис полів сутності та їх типу наведено у таблиці 3.4.

Таблиця 3.4 – Опис полів сутності Requests

№	Назва поля	Тип даних	Розмір
1	RequestId	int	-
2	user	nvarchar	MAX
3	country	nvarchar	MAX
4	city	nvarchar	MAX
5	date_begin	datetime2	7
6	date_end	datetime2	7
7	sumaAll	float	53
8	valutaAll	float	53
9	daily	float	53
10	mobile	float	53
11	hospitality	float	53
12	transport	float	53
13	residence	float	53
14	unexpected	float	53

Сутність **Statuses** (StatusId, Status_name, daily, mobile, hospitality, transport, residence, unexpected) – сутність для таблиці бази даних статусів користувачів.

Опис полів сутності та їх типу наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів сутності Statuses

№	Назва поля	Тип даних	Розмір
1	StatusId	int	-
2	Status_name	nvarchar	MAX
3	daily	float	53
4	mobile	float	53
5	hospitality	float	53
6	transport	float	53
7	residence	float	53
8	unexpected	float	53

Сутність **Users** (UserId, Last_name, First_name, Patronymic_name, Sam_name, Password, Position, Telephone_number, Role) – сутність для таблиці бази даних користувачів.

Опис полів сутності та їх типу наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів сутності Users

№	Назва поля	Тип даних	Розмір
1	UserId	int	-
2	Last_name	nvarchar	MAX
3	First_name	nvarchar	MAX
4	Patronymic_name	nvarchar	MAX
5	Sam_name	nvarchar	MAX
6	Password	nvarchar	MAX
7	Position	nvarchar	MAX
8	Telephone_number	nvarchar	MAX
9	Role	int	-
10	Status	nvarchar	MAX

3.2 Алгоритм функціонування застосунку

Взаємодія з застосунком починається зі сторінки авторизації. Користувачу необхідно вказати "Логін" та "Пароль" та увійти до системи. Відповідно до ролі користувача, буде надано основний функціонал програмного забезпечення. Робітник може створити заявку на відрядження та переглянути узгоджені заявки, для зручності також була додана карта для відображення маршруту та кількість витрат, які було йому надано відповідно до його статусу. Бухгалтер системи має доступ до всієї довідкової інформації, узгодження заявок та формування звітів за кожним працівником підприємства. Адміністратору надано доступ до бази користувачів (додавання, видалення, редагування інформації).

3.3 Структурна схема застосунку

В цьому підрозділі буде представлено структурні схеми клієнтської та серверної частини нашого проєкту. Структурну схему застосунку зображено на рисунку 3.2.

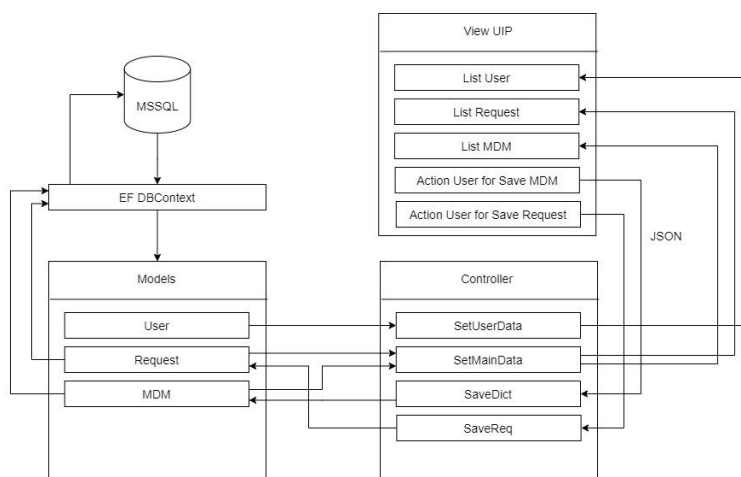


Рисунок 3.2 – Структурна схема застосунку

3.3.1 Структурна схема клієнтської частини

Структура директорій клієнтської частини проєкту представлена на рисунку 3.3.

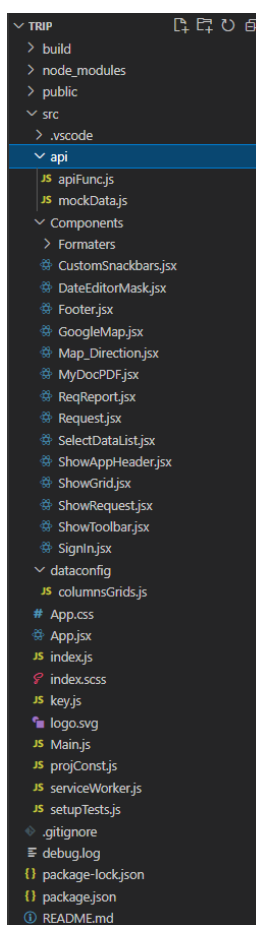


Рисунок 3.3 – Структура директорій клієнтської частини проєкту

Директорія `api` зберігає функції, які використовують найчастіше в компонентах, та мокові дані (тобто дані, з якими можна перевірити, як функціонує програми, на момент коли ще немає зв'язку з бекендом).

Директорія `Components` зберігає всі компоненти проєкту, вложена директорія `Formatters` містить компоненти для задання формату полям.

Директорія `dataconfig` містить компонент, який зберігає структуру всіх довідників, у вигляді об'єктів.

3.3.2 Структурна схема серверної частини

Структура директорій серверної частини проєкту представлена на рисунку 3.4.

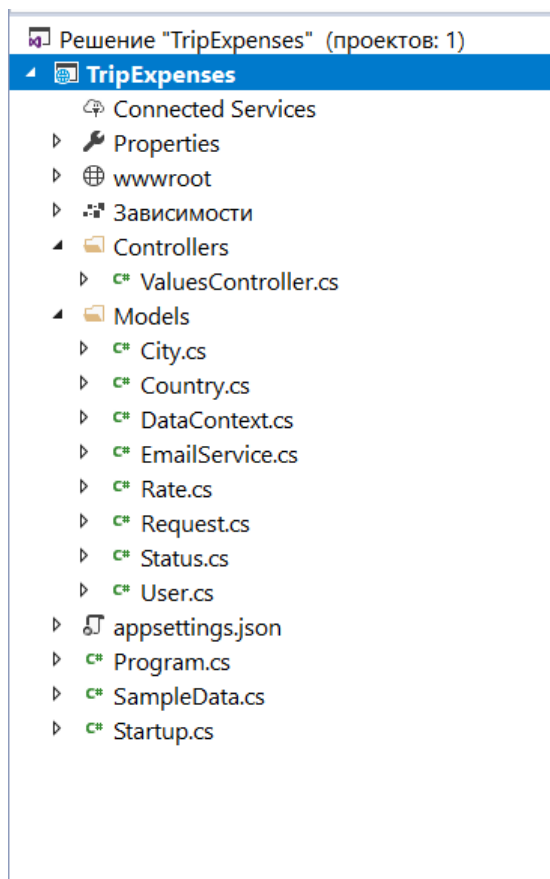


Рисунок 3.4 – Структура директорій серверної частини проєкту

Директорія `wwwroot` - призначена для представлення, тобто містить оптимізовану збірку клієнтської частини програми.

Директорія `controllers` призначена для файлів, які містять основну логіку кожного запиту. В тих файлах прописується всі змінні функції відносно до бази даних.

Директорія `models` призначена для моделей сутностей, які будуть в нашій БД. У файлі моделі сутності прописані всі властивості та їх типи даних. Models отримує дані від controllers, а потім передає дані у представлення.

3.4 Опис реалізованих функцій

В даному підрозділі розглянуто основні функції, які використовуються в застосунку. Розглянуто функції, що взаємодіють між користувачами та клієнтською частиною сайту та функції, що знаходяться на серверній частині та приймають запити з клієнту.

3.4.1 Опис функцій клієнтської частини

Основні функції, що викликаються під час взаємодії користувача із клієнтською частиною (файли `App.jsx` та `Main.js`):

- `mobileFlg()` – функція, що визначає, що встановлена мобільна версія застосунку;
- `alertWinOpen()` - функція, що вмикає модальне вікно, відображає успішність операції;
- `alertWinClose()` - функція, що закриває модальне вікно;
- `SetCurrentUserId()` - функція, що визначає id користувача, який увійшов до системи;
- `handle_Sign_User()` - функція, що визначає, який користувач зайшов до системи та отримує його дані;

- `onHandleAgreedReq()` - функція, що визначає, які саме заявки були узгоджені для робітника;
- `onHandleSave()` - функція для передачі та визначення, які саме дані потрібно зберегти;
- `onHandle_Email()` - функція, що визначає користувача та передає дані для подальшої відправки повідомлення;
- `onHandleAdd()` - функція, що виконує додавання запису до таблиці;
- `onHandleDelete()` - функція, що виконує видалення запису з таблиці;
- `onHandleUpdateGrid()` - функція, що визначає, які записи було відреаговано;
- `handleChange()` - функція, що надає можливість перемикатися по вкладкам у верхній панелі;
- `handleDictChange()` - функція, що надає можливість перемикатися по вкладкам довідників;
- `handle_Request_User_Submit()` - функція, що виконує створення заявки на узгодження робітником;
- `handle_Request_Bux_Submit()` - функція, що виконує збереження вже узгодженої заявки;
- `handle_Request_Close()` - функція, що виконує закриття форми про створенні нової заяви або узгодження заявки;
- `Handle_Seek_Req()` - функція, що виконує пошук по узгодженим заявкам за номером заявки, для подальшого формування звіту;
- `RequestNewRow()` - функція, що заповнює форму для створення заявки початковими значеннями;
- `RequestAgreedRow()` - функція, що визначає чи узгоджена обрана заявка;
- `loadUserRequest()` - функція, що визначає та відображає для робітника узгоджені заявки;
- `getServerData()` - функція для отримання даних з серверу;
- `handleSaveReq()` - функція для передачі даних про заявку на сервер;

- `handleSaveDict()` - функція для передачі змінених даних з таблиці на сервер;
- `handleEmailDict()` - функція для передачі даних для створення email повідомлення на сервер.

3.4.2 Опис функцій серверної частини

Функції, що викликаються під час взаємодії користувача із серверною частиною:

- `GetParamFromReq()` - функція, що визначає було створено нову заявку, чи відредаговано вже існуючу;
- `AddDb()` - функція, що додає новий запис до бази даних;
- `Updatedb()` - функція, що оновлює запис в базі даних;
- `DelDb()` - функція, що видаляє запис з бази даних;
- `SavetoDb()` - функція, що викликає додавання, редагування або видалення, в залежності від того, які дії виконав користувач;
- `GetUsersList()` - функція, що передає дані про користувачів на клієнтську частину програми;
- `GetMainData()` - функція, що передає основні дані (довідники, заявки) на клієнтську частину програми;
- `SaveDict()` - функція, що отримує та передає оновлені дані на клієнтську частину програми;
- `SaveReq()` - функція, що отримує та передає дані про створену або відредаговану заявку на клієнтську частину програми;
- `EmailDict()` - функція, що отримує дані для відправки email-повідомлення та викликає функцію `SendEmail()`, повертає (не)успішність операції;
- `SendEmail()` - функція, що виконує відправку повідомлення на поштову адресу робітника;

– Initialize() - функція, що виконує ініціалізацію бази даних початковими даними при першому запуску програми.

3.5 Висновки до розділу 3

У даному розділі було розглянуто основні сутності (Countrys, Citys, Rates, Requests, Statuses, Users) бази даних та наведено взаємодію між цими сутностями. Було розглянуто взаємодію з застосунком, також було продемонстровано структурну схему програми та структуру директорій, наведено основні реалізовані функції клієнтської та серверної частини програми.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис методики проведення тестування

Тестування буде проведене за таким сценарієм:

- перевірка модуля авторизації (успішність входження користувача в систему, успішне визначення ролі користувача, обробка ситуацій заповнення всіх полів форми та відсутності користувача в системі);
- перевірка модуля створення заявки на відрядження (успішність створення заявки, успішність збереження даних та відображення в системі бухгалтера);
- перевірка модуля перегляду узгоджених заявок для користувачів (коректне відображення узгоджених заявок для користувача, відображення коректної суми витрат на відрядження, відображення коректного маршруту на карті);
- перевірка модуля довідники (коректне заповнення всіх довідників, успішне додавання нового запису до довідника, успішне збереження даних в довіднику, успішне видалення запису з довідника);
- перевірка модуля узгодження заявок (коректне відображення заявок користувачів, коректний розрахунок всіх витрат на відрядження при встановленні статусу, коректний розрахунок суми разом у гривні, або валюті в залежності від країни, успішність збереження заявки після узгодження зі всіма вказаними даними, коректне надіслання повідомлення робітнику на пошту);
- перевірка модуля звіт за заявками (коректне відображення заявки після введення номеру заявки коректне завантаження заявки);
- перевірка модуля списку користувачів системи (коректне відображення даних користувачів системи, успішне додавання нового запису до таблиці, успішне збереження записів про користувачів, успішне видалення запису з таблиці).

4.2 Виконання програми

На початку програми користувач вводить свої дані в поле авторизації (рис. 4.1), програма обробляє та аналізує ці дані, та виконує пошук користувача в базі даних, якщо такого користувача немає, буде виведено повідомлення, про відсутність користувача в системі (користувачів до системи додає адміністратор).

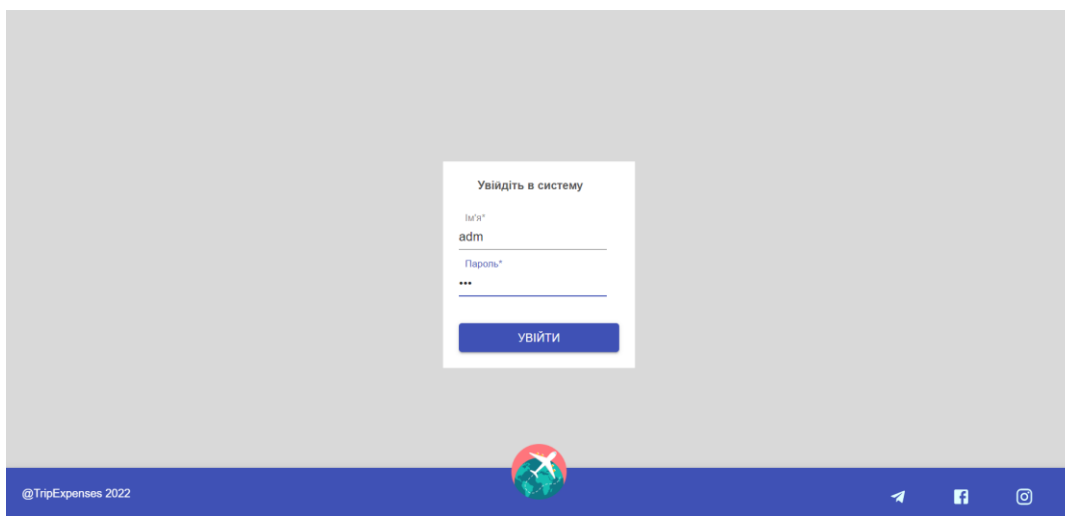


Рисунок 4.1 – Сторінка вебзастосунку для авторизації

Програма містить три ролі користувачів (адміністратор, бухгалтер, робітник), при першому використанні програмою в базі даних наявний лише один користувач (адміністратор), який повинен додати інших користувачів до системи, вказавши відповідну інформацію. Адміністратор має право додавати, видаляти та редагувати інформацію по користувачам (рис. 4.2).

Автоматизована система витрат на відрядження

Адміністратор системи

Вихід

КОРИСТУВАЧІ СИСТЕМИ

Сорокін Андрій
Олександрович

(Системний адміністратор)

Користувачі системи

+

Ключ	Прізвище	Ім'я	По-батькові	Аккаунт	Пароль	Роль (Співр.(1)/Бух.(2)/Адм.(3))	Телефон	Email	Статус	Посада
1	Сорокін	Андрій	Олександрович	adm	adm	3	+38068777720	sorokinol@gmail.com		Системний адміністратор
2	Іванов	Іван	Іванович	ivan	ivan	1	+380686789820	ivanovivan@gmail.com	C	Менеджер
3	Петров	Андрій	Олександрович	petr	petr	1	+380635878920	zaikinev@gmail.com	B	Консультант
4	Шаповалов	Карина	Данилівна	bux	bux123	2	+380687874520	karinchik20000@gmail.com		Бухгалтер

@TripExpenses 2022

Рисунок 4.2 – Сторінка вебзастосунку адміністратора системи

Коли користувачів було додано, робітник може увійти в систему та подати заявку на відрядження вказавши прізвище, ім'я та по-батькові, країну, місто та період відрядження (рис 4.3), після узгодження заявки бухгалтером, робітник може переглянути свою заявку у вкладці узгоджених заявок (рис. 4.4) та маршрут його подорожі на карті відповідно до заявки (рис. 4.5).

Автоматизована система витрат на відрядження										Користувач системи		Вихід
ПЕРЕГЛЯД УЗГОДЖЕНИХ ЗАЯВОК										СТВОРЕННЯ ЗАЯВКИ НА ВІДРЯДЖЕННЯ		Іванов Іван Іванович (Менеджер)

Заявка на відрядження

Співробітник
Іванов Іван Іванович

Країна Україна Місто Харків

Період з 03/05/2022 по 05/05/2022

СТВОРИТИ ЗАЯВКУ

Рисунок 4.3 – Сторінка вебзастосунку робітника системи зі створенням заявки на відрядження

Автоматизована система витрат на відрядження		Користувач системи				Вихід	
ПЕРЕГЛЯД УЗГОДЖЕНИХ ЗАЯВОК		СТВОРЕННЯ ЗАЯВКИ НА ВІДРЯДЖЕННЯ				Іванов Іван Іванович (Менеджер)	
Перегляд узгоджених заявок							
Співробітник		Країна	Місто	Дата початку	Дата кінця	Сума разом	Сума у валюті
Іванов Іван Іванович		Україна	Харків	03.05.2022	05.05.2022	3900	0

Рисунок 4.4 – Сторінка вебзастосунку робітника системи з переглядом узгоджених заявок

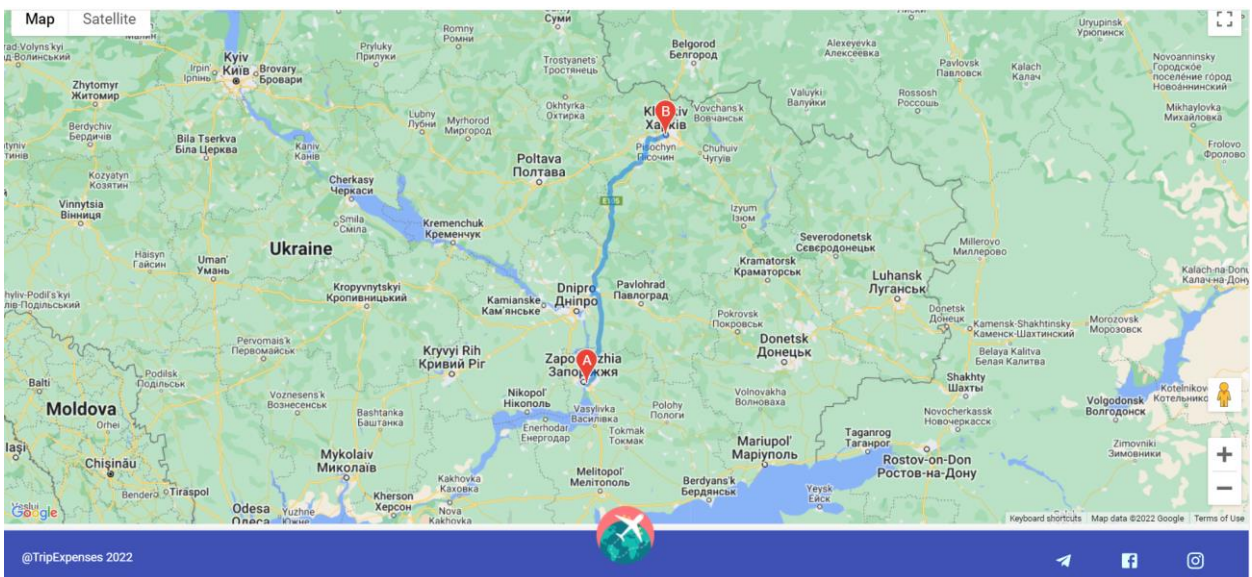


Рисунок 4.5 – Сторінка вебзастосунку робітника системи з переглядом маршруту відповідно до заявки

Бухгалтер системи може додавати, редагувати та видаляти записи в довідниках країн, міст, статусів користувачів, курсів валют та може переглядати довідник працівників підприємства (рис. 4.6). В довіднику міст наявна можливість перегляду місцезнаходження міст на карті (рис. 4.7). В довіднику статусів користувачів, знаходяться статуси та для кожного статусу визначені витрати (щоденні, мобільні, транспорті, проживання, представницькі, непередбачені). Довідник статусів користувачів необхідний при узгодженні заявок.

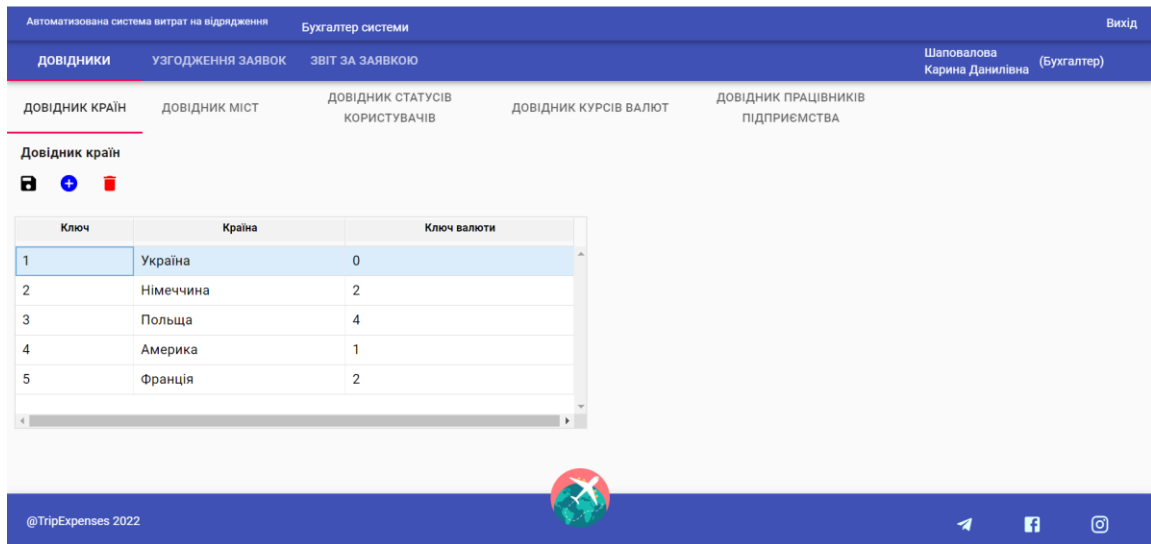


Рисунок 4.6 – Сторінка вебзастосунку бухгалтера системи з переглядом довідника країн

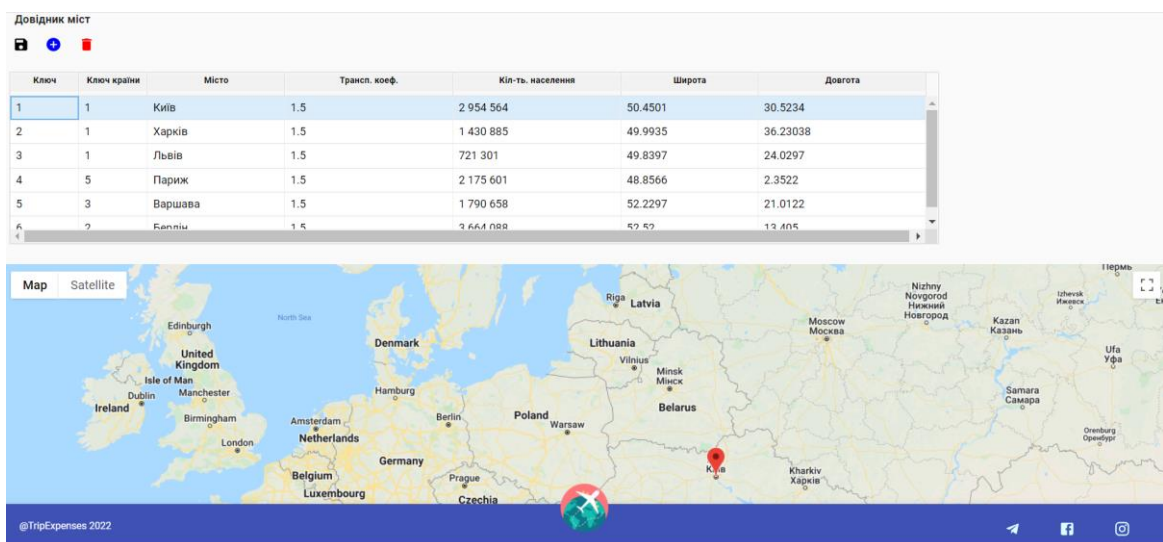


Рисунок 4.7 – Сторінка вебзастосунку бухгалтера системи з переглядом довідника міст

Також бухгалтер системи узгоджує заявки (рис. 4.8), які подають робітники підприємства, тарифи виставляються автоматично в залежності від статусу користувача (добові, мобільні, представницькі, транспортні витрати, проживання, непередбачені витрати). Коли статус користувача визначено, всі значення для витрат беруться з довідника статусів користувачів, для щоденних витрат, додатково помножується значення на кількість днів відрядження, для транспортних витрат, помножується на транспортний

коефіцієнт, якщо робітник відправляється за кордон, визначаються тарифи за курсом країни. Бухгалтер також може редагувати витрати на відрядження для кожного робітника.

Узгодження заявки

Заявка № 1

Співробітник: Іванов Іван Іванович Статус: С

Країна: Україна Місто: Харків

Період з: 03/05/2022 по: 05/05/2022

Ділові	600	Мобільні	300	Транспорт	1,500
Проживання	500	Представницькі	500	Непередбачені	500

Сума разом (гривня) - 3900 Сума разом (вал.) - 0

ЗБЕРЕГТИ ЗАЯВКУ СКАСУВАТИ ПОВІДОМЛЕННЯ НА ПОШТУ

Рисунок 4.8 – Сторінка вебзастосунку бухгалтера системи з узгодженням заявки працівника підприємства

Є можливість формування звіту за вказаною бухгалтером заявкою (рис. 4.9) та збереження цієї заявки у форматі PDF (рис. 4.10).

Звіт за заявкою

Вкажіть номер заявки: 1

Відряджений співробітник: Іванов Іван Іванович

Країна відрядження: Україна Місто відрядження: Харків

Період відрядження з: 03.05.2022 по: 05.05.2022

Витрати на відрядження:

Ділові	600	Мобільні витрати	300	Транспортні витрати	1500
Проживання	500	Представницькі витрати	500	Непередбачені витрати	500

Сума разом (гривня) - 3900 Сума разом (вал.) - 0

ЗБЕРЕГТИ У PDF

Рисунок 4.9 – Сторінка вебзастосунку бухгалтера системи з переглядом звіту за заявкою

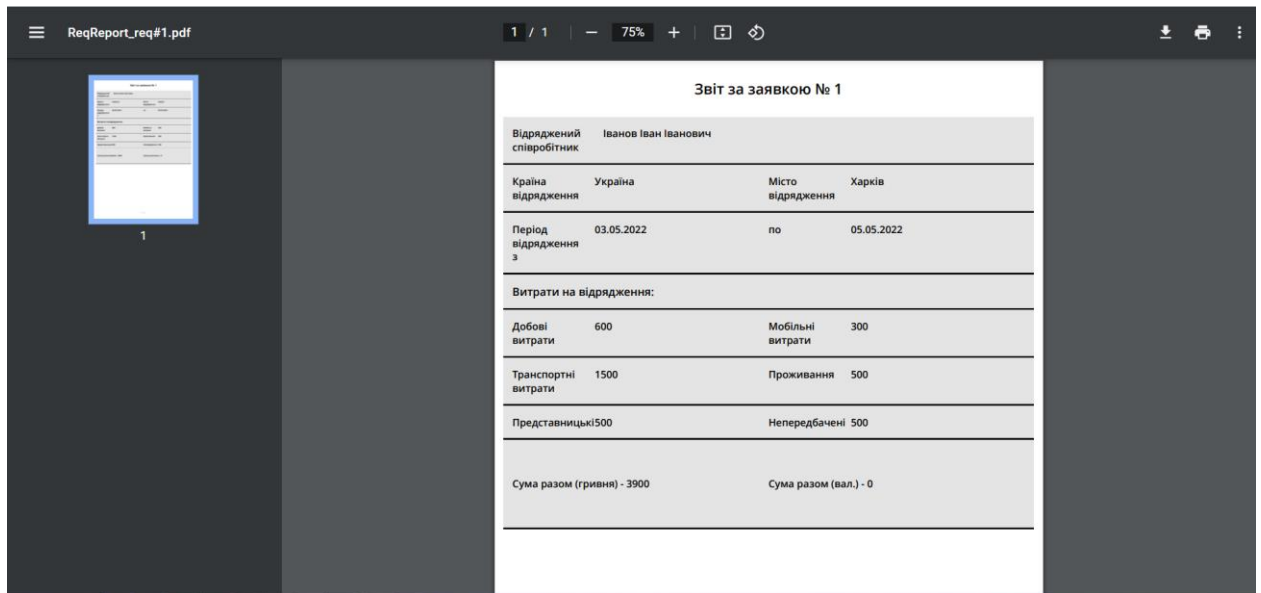


Рисунок 4.10 – Завантажений звіт за заявкою у форматі PDF

Також вебзастосунок адаптований під мобільні пристрої, що дозволяє користувачам взаємодіяти з вебзастосунком на будь-якому пристрої (рис. 4.11).

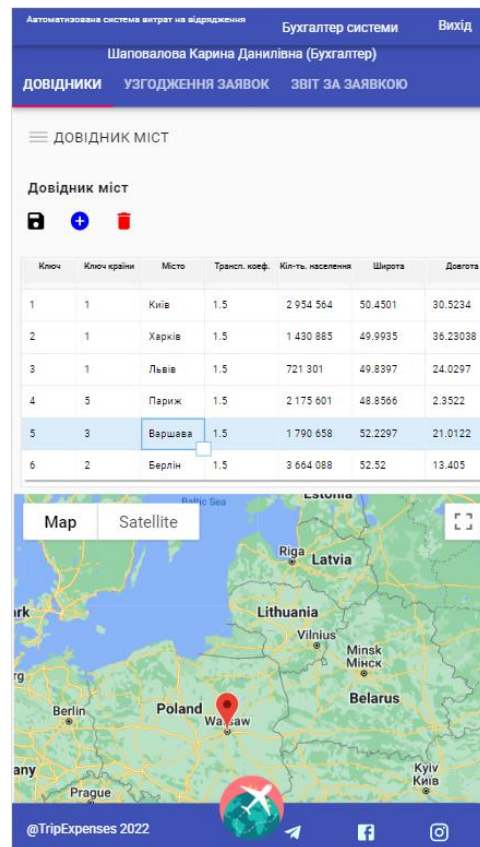


Рисунок 4.11 – Сторінка вебзастосунку бухгалтера системи у мобільній версії

Після тестування застосунку, було складено чек-лист, який наведено у таблиці 4.1.

Таблиця 4.1 – Чек-лист тестування

Вебзастосунок «TripExpenses»	Google Chrome	Microsoft Edge
Модуль авторизації та аутентифікації		
Входження користувача до системи	Passed	Passed
Визначення ролі користувача	Passed	Passed
Обробка заповнення всіх полів форми	Passed	Passed
Обробка відсутності користувача в системі	Passed	Passed
Модуль створення заявки на відрядження		
Створення заявки	Passed	Passed
Збереження та відображення заявки в системі бухгалтера	Passed	Passed
Модуль перегляду узгоджених заявок		
Відображення узгоджених заявок за певним користувачем	Passed	Passed
Відображення коректної суми витрат на відрядження за заявкою	Passed	Passed
Відображення маршруту за заявкою	Passed	Passed
Модуль довідники		
Коректне заповнення довідників	Passed	Passed
Додавання нового запису	Passed	Passed
Збереження нового запису	Passed	Passed
Видалення нового запису	Passed	Passed
Модуль узгодження заявок		
Відображення заявок користувачів	Passed	Passed
Коректний розрахунок витрат на відрядження	Passed	Passed
Коректний розрахунок загальної суми	Passed	Passed
Збереження заявки після узгодження	Passed	Passed

Продовження таблиці 4.1

Вебзастосунок «TripExpenses»	Google Chrome	Microsoft Edge
Відправка повідомлення робітнику на пошту	Passed	Passed
Модуль звіт за заявками		
Відображення заявки за певним номером	Passed	Passed
Завантаження звіту	Passed	Passed
Модуль списку користувачів системи		
Коректне відображення даних	Passed	Passed
Додавання нового запису	Passed	Passed
Збереження нового запису	Passed	Passed
Видалення нового запису	Passed	Passed

4.3 Повідомлення користувачу

Для того, щоб користувачу було легше взаємодіяти з системою, було створено декілька повідомлень.

Якщо користувач не заповнив одне з полів авторизації, буде відображено повідомлення про те, що поля повинні бути обов'язково заповнені (рис. 4.12). Якщо користувач вказав невалідні дані, буде відображено повідомлення про те, що користувач вказав невірні ім'я або пароль (рис. 4.13).

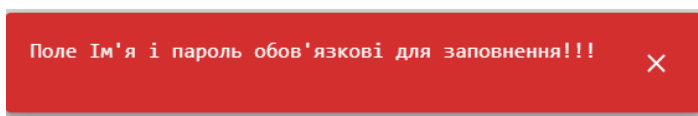


Рисунок 4.12 – Повідомлення про існуючий файл

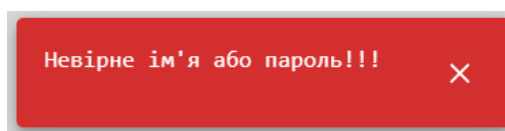


Рисунок 4.13 – Повідомлення про невалідність даних

При створенні нової заявки на відрядження спливає повідомлення про збереження заявки, якщо заявка була створена успішно (рис. 4.14). Також при відправці повідомлення робітнику про узгодження заявки бухгалтером спливає відповідне повідомлення (рис. 4.15).

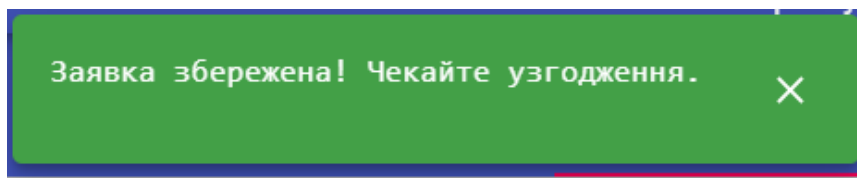


Рисунок 4.14 – Повідомлення про створення заявки

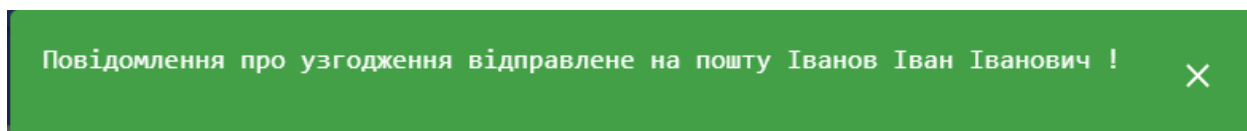


Рисунок 4.15 – Повідомлення про відправку повідомлення на пошту

4.4 Висновки до розділу 4

У цьому розділі було проведено тестування застосунка та продемонстровано взаємодію користувача з вебзастосунком. Було виявлено, що вебзастосунок виконує всі поставлені функції безпомилково. Також у випадку невалідності даних, при створенні заявки та відправці повідомлення на пошту, відображається відповідне повідомлення.

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи було проведено детальне ознайомлення із предметною областю та з найбільш важливими питаннями цієї проблемної області та найвагомішими рішеннями: розробка простого та зрозумілого застосунку з функціями розрахунку витрат на відрядження та побудовою звіту за заявкою. Було розглянуто аналоги застосунку, виділено основні переваги та недоліки застосунків, після чого було зрозуміло в яких аспектах розроблена програма повина перевершувати аналоги. На підставі цього були поставлені функціональні вимоги та наведено функціональну структуру у вигляді схеми.

Для реалізації ПЗ були обрані основні інструменти веброзробника, для клієнтської частини обрано JavaScript з використання бібліотеки React та Redux. Для реалізації серверної частини вибір припав на мову програмування C# з використанням платформи .NET Core. В якості інтегрованого середовища розробки програми для фронтенду було обрано – Visual Studio Code з набором відповідних плагінів, та для бекенду – Visual Studio версії 17.0. Для системи управління базами даних було обрано Microsoft SQL Server.

У розробленому програмному забезпеченні реалізовано створення та узгодження заявок, розрахунок витрат на відрядження, побудова та завантаження звіту, зберігання даних про користувачів, заявок, довідників, тощо. ПЗ було протестовано та перевірено на відображення коректних повідомлень про помилку.

Розроблене програмне забезпечення відповідає вимогам технічного завдання та може експлуатуватися за призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smartfin.ua [Electronic resource] / – Access mode: <https://smartfin.ua/>.
2. SAP Concur [Electronic resource] / – Access mode: <https://www.concur.com/en-us/travel-booking>.
3. Travel Management Software [Electronic resource] / – Access mode: <https://www.softwaretestinghelp.com/travel-management-software/>.
4. TravelPerk [Electronic resource] / – Access mode: <https://www.travelperk.com/>.
5. HTML [Electronic resource] / – Access mode: <https://www.investopedia.com/terms/h/html.asp>.
6. Thomas A. Powell HTML & CSS: The Complete Reference. – New York : The McGraw-Hill Companies, 2010. – 429 с.
7. Javascript [Electronic resource] / – Access mode: <https://www.bloomtech.com/article/what-is-javascript-used-for>.
8. Frontend and Backend [Electronic resource] / – Access mode: <https://lizard.global/blog/what-are-frontend-and-backend-in-app-development>.
9. React [Electronic resource] / – Access mode: <https://www.thirdrocktechkno.com/blog/why-choose-reactjs-for-your-next-project-features-and-benefits/>.
10. Redux [Electronic resource] / – Access mode: <https://www.makeuseof.com/how-to-use-redux/>.
11. Redux (Action, Reducers and Store) [Electronic resource] / – Access mode: <https://www.geeksforgeeks.org/introduction-to-redux-action-reducers-and-store/>.
12. Virtual DOM [Electronic resource] / – Access mode: <https://programmingwithmosh.com/react/react-virtual-dom-explained/>.
13. C# and .NET Core [Electronic resource] / – Access mode: <https://stackify.com/net-core-csharp-next-programming-language/>.

14. SQL-Server [Electronic resource] / – Access mode: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server>.
15. Visual Studio Code [Electronic resource] / – Access mode: <https://chromium.googlesource.com/chromium/src/+HEAD/docs/vscode.md>.
16. Можливості Visual Studio Code [Electronic resource] / – Access mode: <https://www.educba.com/what-is-visual-studio-code/>.
17. Visual Studio [Electronic resource] / – Access mode: <https://www.techclbr.com/2018/01/what-is-visual-studio-pros-and-cons-of-vs.html>.
18. Create React App [Electronic resource] / – Access mode: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app>.

ДОДАТОК А
Технічне завдання

Вступ

Вебзастосунок для організації обліку витрат може використовуватися тільки на підприємствах, щоб автоматизувати систему організації відряджень.

A.1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на дипломну кваліфікаційну роботу бакалавра на тему: «Програмна реалізація застосунку для організації обліку витрат на відрядження працівників підприємства». Затверджене наказом № 102 від 27 квітня 2022 р. по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Функціональне призначення: програма дозволяє організувати облік витрат на відрядження.

A.3 Вимоги до програми чи програмному виробу

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати:

- аутентифікація користувача.

Для робітника:

- створення заявки на відрядження;

- перегляд узгодженої заявки з сумою відрядних та можливістю перегляду маршруту на карті.

Для бухгалтера:

- ведення нормативно-довідкової інформації (довідник статусів користувачів, довідник країн, довідник міст, довідник валют з поточним курсом, довідник лімітів по статусах);
- розрахунок тарифів на відрядження для кожного робітника, в залежності від його статусу на підприємстві;
- відправка сформованої заявки на електронну пошту робітника;
- авансовий звіт за заявкою з можливістю завантаження.

Для адміністратора:

- додавання користувачів системи;
- видалення користувачів системи;
- коригування даних по користувачу.

A.3.2 Вимоги до надійності

Надійне функціонування системи має забезпечуватися наступними факторами:

- безперебійне живлення апаратно-технічних засобів;
- доступ до мережі Інтернет;
- необхідний рівень кваліфікації обслуговуючого персоналу.

Також, система повинна мати можливість оперативного відновлення працездатності при збоях.

A.3.3 Умови експлуатації

Для забезпечення експлуатації програмного забезпечення необхідно встановити наступні програмні застосунки:

- NodeJS;
- MSSQL;

- ОС: MacOS, Windows, Android, IOS;
- Браузер Google Chrome, Mozilla Firefox, Safari.

A.3.4 Вимоги до складу й параметрів технічних засобів

Мінімальні системні вимоги:

- ОЗУ: 2 ГБ;
- процесор: 1.86 GHz;
- система: багатоплатформна;
- ОС: MacOS, Windows, Android, IOS;
- доступ до Інтернету.

A.3.5 Вимоги маркування й упакування

Програма може бути записана на оптичному, жорсткому магнітному диску або USB-носії. На упаковці або відповідному пристрої має бути вказано назву роботи “Програмна реалізація застосунку для організації обліку витрат на відрядження працівників підприємства”.

A.3.6 Вимоги транспортування й збереження

Вимоги до транспортування та збереження повністю співпадають з вимогами до транспортування та збереження DVD-дисків.

A.4 Стадії та етапи розробки

Для роботи над дипломною кваліфікаційною роботою бакалавра заплановано виконання наступних етапів:

- а) аналіз проблеми;
- б) розробка архітектури програми;

- в) розробка програми;
- г) експлуатація, тестування та експериментальне дослідження програми;
- д) оформлення документації.

A.5 Порядок контролю й приймання

Дипломна кваліфікаційна робота бакалавра повинна бути виконана у відповідності з календарним планом. Дипломна кваліфікаційна робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Текст файлу index.jsx [18]

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.scss';
import Main from './Main';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<Main />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can
change
// unregister() to register() below. Note this comes with some
pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

Б.2 Текст файлу Main.jsx

```
import React,{Fragment,useState,useEffect} from 'react';
import App from './App';
import CustomSnackbars from "./Components/CustomSnackbars";
import { apiDataUrl,apiListUrl,apiSaveDictUrl,apiEmailDictUrl,apiSaveReqUrl,thisMock } from
'./projConst';
import {listdata,maindata} from './api/mockData';
import * as api from './api/apiFunc'
import { QueryParamProvider } from 'use-query-params';

function Main() {
  const [alertmessage, SetAlertmessage] = useState(0);
  const alertTypeMess = ['success', 'warning', 'error', 'info']
  const [userdata, SetUserData] = useState({});
  const [otherdata, SetMainData] = useState({});

  const alertWinClose = () => {
    SetAlertmessage({ open: false, mess: "", typemess: 'error'})
  };

  const getServerData = (apiUrl,param,thismock,mock,SetState) => {
    api.FetchLiteUrl(apiUrl, true, param, thismock, mock).then(onedata => {
      const { is_err, mess } = api.HandleFindError(onedata)
      if (is_err)
        SetAlertmessage({ open: true, mess: mess || ' Помилка при завантаженні даних з
сервера', typemess: 'error' })
      else
```

```

        if (typeof SetState === 'function') SetState(onedata)

    })
}
useEffect(() => {
    async function fetchData() {
        await getServerData(apiListUrl, {}, thisMock, listdata, SetUserData)
        await getServerData(apiDataUrl, {}, thisMock, maindata, SetMainData)
    }
    fetchData();
}, []);

const handleSaveReq = async (data) => {
    await getServerData(apiSaveReqUrl, data, false , maindata, SetMainData)
}

const handleSaveDict = async (data) => {
    //console.log('SaveDict = ', data)
    if (!api.isEmpty(data)&&data.table)
    {
        if (data.table == "User_List")
            await getServerData(apiSaveDictUrl, data, false , listdata, SetUserData)
        else
            await getServerData(apiSaveDictUrl, data, false , maindata, SetMainData)
    }
}

const handleEmailDict = async (data) => {
    if (!api.isEmpty(data))
    {

        await getServerData(apiEmailDictUrl, data, false , listdata, SetUserData)
    }
}

return (
    <Fragment>
        {api.isEmpty(userdata) && api.isEmpty(otherdata) &&
            <div>Завантаження даних з сервера...</div>
        }
        { userdata && otherdata &&
            <App userdata={userdata}
                maindata={otherdata}
                SaveDict={handleSaveDict}
                SaveReq={handleSaveReq}
                EmailDict={handleEmailDict}
            />
        }
        <CustomSnackbars
            useropen={alertmessage.open}
            userClose = {alertWinClose}

```

```

        uservar = {alertmessage.typemess}
        usermess = {alertmessage.mess}
      />
    </Fragment>
  );
}

export default Main;

```

Б.3 Текст файлу App.js

```

import React, { Fragment, useEffect, useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import "bootstrap/dist/css/bootstrap.css";
import PropTypes from 'prop-types';
import Typography from '@material-ui/core/Typography';
import Box from '@material-ui/core/Box';
import AppBar from '@material-ui/core/AppBar';
import Tabs from '@material-ui/core/Tabs';
import Tab from '@material-ui/core/Tab';
import List from '@material-ui/core/List';
import ListItem from '@material-ui/core/ListItem';
import ListItemText from '@material-ui/core/ListItemText';
import MenuItem from '@material-ui/core/MenuItem';
import Menu from '@material-ui/core/Menu';
import DehazeIcon from '@material-ui/icons/Dehaze';
import ShowGrid from "../Components/ShowGrid";

import "../App.css";
import { columnsCountryList,
  columnsCityList,
  columnsBuxUserList,
  columnsUserList,
  columnsStatusList,
  columnsRateList,
  columnsUserRequests,
  columnsBuxRequests
} from '../dataconfig/columnsGrids'

import CustomSnackbars from "../Components/CustomSnackbars";
import ShowAppHeader from "../Components/ShowAppHeader";
import SignIn from "../Components/SignIn";
import Request from "../Components/Request";
import ReqReport from "../Components/ReqReport";
import ShowRequest from "../Components/ShowRequest";

```

```

import * as api from "./api/apiFunc";
import Footer from "./Components/Footer";

const useStyles = makeStyles(theme => ({
  responsPerson: {
    margin: theme.spacing(),
    height: 900,
    width: "auto"
  },
  root: {
    flexGrow: 1,
    backgroundColor: theme.palette.background.paper,
  },
  apptitle: {
    display: 'flex',
    width: '100%'
  },
  apptitlemob: {
    display: 'block',
    width: '100%'
  },
  apptab: {
    minWidth: '80% !important'
  },
  apptabmob: {
    minWidth: '100% !important'
  },
  usertitle: {
    alignSelf: 'center',
    width: '60%',
    margin: '0 auto'
  },
  applabel: {
    marginLeft: '15px',
    marginTop: '10px'
  },
  apptabtitle: {
    margin: '10px'
  },
  content: {
    paddingLeft: -theme.spacing(5),
    textAlign: "center"
  },
  reqtitle: {
    paddingLeft: theme.spacing(3),
    marginBottom: 30,
    marginTop: 30,
    textAlign: "center",

```

```

    color: "#000",
    fontWeight: "bold",
    height: 5
  },
  title: {
    paddingLeft: theme.spacing(3),
    textAlign: "left",
    color: "#000",
    backgroundColor: "#d9d9d9",
    fontSize: "75%",
    fontWeight: "bold",
    height: 5
  },
  clgrid: {
    marginTop: 20,
    marginBottom: 20
  },
  refwaste: {},
  footer: {
    padding: '10px 20px',
    background: '#3f51b5',
    position: 'absolute',
    bottom: '0',
    width: '100%',
    color: 'white'
  }
}
}));

function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && (
        <Box p={3}>
          <Typography component={'div'}>{children}</Typography>
        </Box>
      )}
    </div>
  );
}

```

```

TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.any.isRequired,
  value: PropTypes.any.isRequired,
};

function App (props) {
  const classes = useStyles();
  const { userdata, maindata } = props

  const [alertmessage, SetAlertmessage] = useState(0);
  const [value, setValue] = useState(0);
  const [dictvalue, setDictValue] = useState(0);
  const [anchorEl, setAnchorEl] = useState(null);
  const [selectedIndex, setSelectedIndex] = useState(0);
  const [currentUser, SetcurrentUser] = useState({})
  const [curentRow,SetCurentRow] = useState({})
  const [city, SetCity] = useState([]);
  const [country, SetCountry] = useState([]);
  const [status, SetStatus] = useState([]);
  const [rate, SetRate] = useState([]);
  const [user, SetUser] = useState([]);
  const [request, SetRequest] = useState([]);
  const [agreed,SetAgreed] = useState(0);
  const [requestfind, SetRequestfind] = useState(null);
  const [picklists, SetPicklists] = useState([]);

  const [updatedData, SetUpdatedData] = useState([]);
  const [currentElem,SetcurrentElem] = useState({currentUserId:0,currentGroupId:null});
  const [thismobile, Setthismobile] = useState(api.thismobile());
  const [vismap,SetVisMap] = useState(false)

  const mobileFlg = () => {
    Setthismobile(api.thismobile())
    return (api.thismobile())
  }

  useEffect(() => {
    window.addEventListener('resize', function(e){
      mobileFlg()
    });
    return () => {
      window.removeEventListener('resize')
    }
  }, [])

  useEffect(() => {
    if (!userdata) return

```

```

    SetUser(userdata);
  }, [userdata])

  useEffect(() => {
    if (!maindata) return
    SetCountry(maindata.country_List)
    SetCity(maindata.city_List)
    SetStatus(maindata.status_List);
    SetRate(maindata.rate_List);
    SetRequest(maindata.requests);

    if (maindata.country_List) {
      const country_List = api.mutablePickList(maindata.country_List, 'countryId',
'country_name')
      SetPicklists([...picklists, country_List])
    }
  }, [maindata]);

  function allyProps(index) {
    return { id: `simple-tab-${index}`, 'aria-controls': `simple-tabpanel-${index}`, };
  }

  const alertWinOpen = (typemess, mess) => {
    SetAlertmessage({ open: true, mess , typemess})
  }
  const alertWinClose = () => {
    SetAlertmessage({ open: false, mess: "", typemess: 'error'})
  };

  const SetCurrentUserId = (id, fio) => {
    SetcurrentElem({...currentElem, currentUserId: id})
  }

  const handle_Sign_User = (user) => {
    SetcurrentUser({ ...user })
  }

  const onHandleAgreedReq = (name, row, pkid) => {
    SetAgreed(row[pkid])
  }

  const onHandleSave = (name, data) => {
    props.SaveDict({ table: name, data })
  }

  const onHandle_Email = (data) => {
    userdata.map((el) => {
      if (`${el.last_name} ${el.first_name} ${el.patronymic_name}` == data.user.trim()){

```

```

    const user = data.user
    props.EmailDict({ email : el.email, user })
    alertWinOpen('success', `Повідомлення про узгодження відправлене на пошту ${data.user}`)
  !`)
  }
  })
}

const onHandleAdd = (name, row) => {
  switch (name) {
    case 'country_List':
      SetCountry([...country,row])
      break
    case 'city_List':
      SetCity([...city,row])
      break
    case 'status_List':
      SetStatus([...status,row])
      break
    case 'rate_List':
      SetRate([...rate,row])
      break
    case 'User_List':
      SetUser([...user,row])
      break
  }
}

const onHandleDelete = (name, row, pkid) => {
  switch (name) {
    case 'country_List':
      SetCountry(country.filter(el => el[pkid] !== row[pkid]))
      break
    case 'city_List':
      SetCity(city.filter(el => el[pkid] !== row[pkid]))
      break
    case 'status_List':
      SetStatus(status.filter(el => el[pkid] !== row[pkid]))
      break
    case 'rate_List':
      SetRate(rate.filter(el => el[pkid] !== row[pkid]))
      break
    case 'User_List':
      SetUser(user.filter(el => el[pkid] !== row[pkid]))
      break
  }
}

```

```

const onHandleUpdateGrid = (updated) => {
  SetUpdatedData({updatedData : updated})
}

const handleChange = (event, newValue) => {
  setValue(newValue);
};

const handleDictChange = (event, newValue) => {
  setDictValue(newValue);
  setSelectedIndex(newValue);
};

const handle_Request_User_Submit = (action, req) => {
  alertWinOpen('success', 'Заявка збережена! Чекайте узгодження.')
  props.SaveReq({ action, ...req })
}

const handle_Request_Bux_Submit = (action, req) => {
  props.SaveReq({ action, ...req })
  SetAgreed(0)
}

const handle_Request_Close = (req) => {
  SetAgreed(0)
}

const Handle_Seek_Req = (id) => {
  const RowReq = request.filter(el => el.requestId == id)
  if (api.isNotEmptyArr(RowReq)) {
    SetRequestfind(RowReq[0])
  }
  else {
    SetRequestfind(null)
  }
}

const RequestNewRow = () => {
  return {
    requestId: 0, user: `${currentUser.last_name} ${currentUser.first_name}
${currentUser.patronymic_name} `,
    country: null, city: null, date_begin: null, date_end: null,
    summaAll: 0, valutaAll: 0, daily: 0, mobile: 0, hospitality: 0,
    transport: 0, residence: 0, unexpected: 0}
}

const RequestAgreedRow = () => {
  if (!api.isNotEmptyArr(request)) return null
  return request.filter(el => el.requestId == agreed)[0]
}

const loadUserRequest = () => {
  if (!api.isNotEmptyArr(request)) return null
  return request.filter(el => el.user.trim() == `${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name}` && el.summaAll !== 0 ||

```

```

el.valutaAll !== 0 ).sort((a,b)=>(new Date(b.date_begin) - new Date(a.date_begin)))
}

const ViewApprovedReq = () => <Fragment>
<div className={classes.applabel}><b>Перегляд узгоджених заявок</b></div>
<ShowGrid
  name='Requests'
  PrimaryId='requestId'
  columnsgrid={columnsUserRequests(thismobile)}
  data={loadUserRequest()}
  maxwidth={thismobile===false ? 3 : 2}
  mob={thismobile}
  readonly={true}
  picklist = {picklists}
  onSetId={SetCurrentUserId}
  onSaveData={onHandleSave}

  onHandleUpdateGrid={onHandleUpdateGrid}
  onHandleDelete={onHandleDelete}
  onHandleAdd={onHandleAdd}
  alertWin={alertWinOpen}
  googlemap={true}
  dicccity={city}
  num = {true}
/>
</Fragment>
const RequestNew = () => <Fragment>
  <div className={classes.reqtitle}>Заявка на відрядження</div>
  <Request
    current={currentUser}
    newreq={true}
    row={RequestNewRow}
    list={{ countryList: country, cityList: city }}
    onHandle_Submit={handle_Request_User_Submit}
    onHandle_Close={handle_Request_Close}
  />
</Fragment>
//-----
const AppBarRole1PC = () => <AppBar position="static">
  <div className={classes.apptitle}>
    <Tabs className={classes.apptab} value={value} onChange={handleChange} aria-
label="simple tabs">
      <Tab label="Перегляд узгоджених заявок" {...a11yProps(0)} />
      <Tab label="Створення заявки на відрядження" {...a11yProps(1)} />
    </Tabs>
    <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} `</div>
    <div className={classes.usertitle}>`${currentUser.position}`</div>

```

```

    </div>
  </AppBar>
  const AppBarRole1Mob = () => <AppBar position="static">
    <div className={classes.apptitlemob}>
      <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} (${currentUser.position})`</div>
      <Tabs className={classes.apptabmob} value={value} onChange={handleChange} aria-
label="simple tabs">
        <Tab label="Перегляд узгоджених заявок" {...a11yProps(0)} />
        <Tab label="Створення заявки на відрядження" {...a11yProps(1)} />
      </Tabs>
    </div>
  </AppBar>
  const AppBarRole1 = () => {
    if (!thismobile)
      return <AppBarRole1PC/>
    else
      return <AppBarRole1Mob/>
  }
  //-----
  const MenuDictionariesMob = () => {
    const options = [
      "ДОВІДНИК КРАЇН",
      "ДОВІДНИК МІСТ",
      "ДОВІДНИК СТАТУСІВ КОРИСТУВАЧІВ",
      "ДОВІДНИК КУРСІВ ВАЛЮТ",
      "ДОВІДНИК ПРАЦІВНИКІВ ПІДПРИЄМСТВА",
    ];
    const handleClickListItem = (event) => {
      setAnchorEl(event.currentTarget);
    };

    const handleMenuItemClick = (event, index) => {
      setSelectedIndex(index);
      setDictValue(index);
      setAnchorEl(null);
    };

    const handleClose = () => {
      setAnchorEl(null);
    };
    return <Fragment>
      <List component="nav" aria-label="Device settings">
        <ListItem
          button
          aria-haspopup="true"
          aria-controls="lock-menu"
          aria-label="when device is locked"

```

```

        onClick={handleClickListItem}
      >
        <DehazeIcon style={{color: '#bfbfbf'}} className='material-icons' />&nbsp;
        <ListItemText primary={options[selectedIndex]} />
      </ListItem>
    </List>
    <Menu
      id="lock-menu"
      anchorEl={anchorEl}
      keepMounted
      open={Boolean(anchorEl)}
      onClose={handleClose}
    >
      {options.map((option, index) => (
        <MenuItem
          key={option}
          disabled={index === 99}
          selected={index === selectedIndex}
          onClick={(event) => handleMenuItemClick(event, index)}
        >
          {option}
        </MenuItem>
      ))}
    </Menu>
  </Fragment>
}

const MenuDictionariesPC = () => <Fragment>
  <Tabs className={classes.apptab} value={dictvalue} onChange={handleDictChange} aria-label="simple tabs">
    <Tab label="Довідник країн" {...a11yProps(0)} />
    <Tab label="Довідник міст" {...a11yProps(1)} />
    <Tab label="Довідник статусів користувачів" {...a11yProps(2)} />
    <Tab label="Довідник курсів валют" {...a11yProps(3)} />
    <Tab label="Довідник працівників підприємства" {...a11yProps(4)} />
  </Tabs>
</Fragment>

const AppDictRole2 = () => {
  if (!thismobile)
    return <MenuDictionariesPC/>
  else
    return <MenuDictionariesMob/>
}

const AppBarRole2PC = () => <Fragment>
  <AppBar position="static">
    <div className={classes.apptitle}>
      <Tabs className={classes.apptab} value={value} onChange={handleChange} aria-label="simple tabs">

```

```

        <Tab label="Довідники" {...a11yProps(0)} />
        <Tab label="Узгодження заявок" {...a11yProps(1)} />
        <Tab label="Звіт за заявкою" {...a11yProps(2)} />
      </Tabs>
      <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} `</div>
      <div className={classes.usertitle}>`${currentUser.position}`</div>
    </div>
  </AppBar>
</Fragment>

const AppBarRole2Mob = () => <Fragment>
  <AppBar position="static">
    <div className={classes.apptitlemob}>
      <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} (${currentUser.position})`</div>
      <Tabs className={classes.apptabmob} value={value} onChange={handleChange} aria-
label="simple tabs">
        <Tab label="Довідники" {...a11yProps(0)} />
        <Tab label="Узгодження заявок" {...a11yProps(1)} />
        <Tab label="Звіт за заявкою" {...a11yProps(2)} />
      </Tabs>
    </div>
  </AppBar>
</Fragment>

const AppBarRole2 = () => {
  if (!thismobile)
    return <AppBarRole2PC/>
  else
    return <AppBarRole2Mob/>
}
//-----

const AppBarRole3PC = () => <Fragment>
  <AppBar position="static">
    <div className={classes.apptitle}>
      <Tabs className={classes.apptab} value={value} onChange={handleChange} aria-
label="simple tabs">
        <Tab className={classes.apptabtitle} label="Користувачі системи"
        {...a11yProps(0)} />
      </Tabs>
      <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} `</div>
      <div className={classes.usertitle}>`${currentUser.position}`</div>
    </div>
  </AppBar>
</Fragment>

```

```

const AppBarRole3Mob = () => <Fragment>
  <AppBar position="static">
    <div className={classes.apptitlemob}>
      <div className={classes.usertitle}>`${currentUser.last_name}
${currentUser.first_name} ${currentUser.patronymic_name} (${currentUser.position})`</div>
      <Tabs className={classes.apptabmob} value={value} onChange={handleChange} aria-
label="simple tabs">
        <Tab className={classes.apptabtitle} label="Користувачі системи"
        {...a11yProps(0)} />
      </Tabs>
    </div>
  </AppBar>
</Fragment>

const AppBarRole3 = () => {
  if (!thismobile)
    return <AppBarRole3PC/>
  else
    return <AppBarRole3Mob/>
}

return (
  <div className="approot">
    {api.isEmpty(currentUser) && <SignIn
      userList={user}
      onHandle_Sign_User={handle_Sign_User}
      alertWin={alertWinOpen}
    />}}
    { !api.isEmpty(currentUser) && request && <Fragment>
      <ShowAppHeader
        viewmode={currentUser.role}
        messageSimple={() => ("Просте повідомлення")}
        messagetitle="Автоматизована система витрат на відрядження"
        alertSimple={() => alertWinOpen("info", "Розширена інфо про сайт")}
        errorSimple={() => alertWinOpen("error", "Виведення помилки")}
        onExitApp={() => SetcurrentUser({})}
      />
      {currentUser.role == 1 && <Fragment>
        {AppBarRole1()}
        <TabPanel value={value} index={0}>
          {ViewApprovedReq()}
        </TabPanel>
        <TabPanel value={value} index={1}>
          {RequestNew()}
        </TabPanel>
      </Fragment>
      {currentUser.role == 3 && <Fragment>

```

```

{AppBarRole3()}
<TabPanel value={value} index={0}>
<div className={classes.applabel}><b>Користувачі системи</b></div>
    <ShowGrid
        name='User_List'
        PrimaryId='userId'
        columnsgrid={columnsUserList(thismobile)}
        data={user}
        maxwidth={2}
        minHeight={290}
        readonly={false}
        mob={thismobile}
        picklist = {picklists}
        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        alertWin={alertWinOpen}
    />
</TabPanel>
</Fragment>
{currentUser.role == 2 && <Fragment>
    {AppBarRole2()}
    <TabPanel value={value} index={0}>
        {AppDictRole2()}
        <TabPanel value={dictvalue} index={0}>
            <div className={classes.applabel}><b>Довідник країн</b></div>
                <ShowGrid
                    name='country_List'
                    PrimaryId='countryId'
                    columnsgrid={columnsCountryList(thismobile)}
                    data={country}
                    maxwidth={thismobile==false ? 1 : 3}
                    mob = {thismobile}
                    readonly={false}
                    picklist = {picklists}
                    onSetId={SetCurrentUserId}
                    onSaveData={onHandleSave}
                    onHandleUpdateGrid={onHandleUpdateGrid}
                    onHandleDelete={onHandleDelete}
                    onHandleAdd={onHandleAdd}
                    alertWin={alertWinOpen}
                />
            </TabPanel>
            <TabPanel value={dictvalue} index={1}>
                <div className={classes.applabel}><b>Довідник міст</b></div>
                    <ShowGrid

```

```

        name='city_List'
        PrimaryId='cityId'
        columnsgrid={columnsCityList(thismobile)}
        data={city}
        maxwidth={thismobile==false ? 3 : 2}
        mob = {thismobile}
        readonly={false}
        picklist = {picklists}
        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        alertWin={alertWinOpen}
        googlemap={true}
        num = {false}
    />
</TabPanel>
<TabPanel value={dictvalue} index={2}>
    <div className={classes.applabel}><b>Довідник статусів користувачів</b></div>
    <ShowGrid
        name='status_List'
        PrimaryId='statusId'
        columnsgrid={columnsStatusList(thismobile)}
        data={status}
        maxwidth={2}
        mob = {thismobile}
        readonly={false}
        picklist = {picklists}
        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        alertWin={alertWinOpen}
    />
</TabPanel>
<TabPanel value={dictvalue} index={3}>
    <div className={classes.applabel}><b>Довідник курсів валют</b></div>
    <ShowGrid
        name='rate_List'
        PrimaryId='rateId'
        columnsgrid={columnsRateList(thismobile)}
        data={rate}
        maxwidth={thismobile==false ? 1 : 3}
        mob = {thismobile}
        readonly={false}
        picklist = {picklists}

```

```

        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        alertWin={alertWinOpen}
    />
</TabPanel>
<TabPanel value={dictvalue} index={4}>
    <div className={classes.applabel}><b>Довідник працівників підприємства</b></div>
    <ShowGrid
        name='User_List'
        PrimaryId='userId'
        columnsgrid={columnsBuxUserList(thismobile)}
        data={user}
        maxwidth={2}
        readonly={true}
        mob = {thismobile}
        picklist = {picklists}
        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        alertWin={alertWinOpen}
    />
</TabPanel>
</TabPanel>
<TabPanel value={value} index={1}>
    <div className={classes.applabel}><b>Узгодження заявок</b></div>
    <ShowGrid
        name='Requests'
        PrimaryId='requestId'
        columnsgrid={columnsBuxRequests(thismobile)}
        data={request}
        maxwidth={2}
        mob={thismobile}
        readonly={true}
        agreed={true}
        agreement={true}
        minHeight={290}
        picklist={picklists}
        onSetId={SetCurrentUserId}
        onSaveData={onHandleSave}
        onHandleUpdateGrid={onHandleUpdateGrid}
        onHandleDelete={onHandleDelete}
        onHandleAdd={onHandleAdd}
        onHandleAgreedReq={onHandleAgreedReq}
    >

```

```

        alertWin={alertWinOpen}
      />
    </TabPanel>
    <TabPanel value={value} index={2}>
    <div className={classes.applabel}><b>Звіт за заявкою</b> </div>
      <ReqReport
        thismobile={thismobile}
        row={requestfind}
        onHandle_Seek_Req={Handle_Seek_Req}
      />
    </TabPanel>
  </Fragment>
  <ShowRequest
    open={agreed!==0}
    title={'Узгодження заявки'}
    message={`Заявка № ${agreed}`}
    onCloseModal={()=>SetAgreed(false)}
  >
    <Request
      current={currentUser}
      newreq={false}
      row={RequestAgreedRow()}

list={{countryList:country,cityList:city,statusList:status,rateList:rate}}
      onHandle_Submit={handle_Request_Bux_Submit}
      onHandle_Close={handle_Request_Close}
      onHandle_Email={onHandle_Email}
    />
  </ShowRequest>
</Fragment>
<CustomSnackbars
  useropen={alertmessage.open}
  userClose = {alertWinClose}
  uservar = {alertmessage.typemess}
  usermess = {alertmessage.mess}
/>
  <Footer></Footer>
</div>
);
}

export default App;

```

Б.4 Текст файла Program.cs

```
public class Program
{
    public static void Main(string[] args)
    {
        //CreateWebHostBuilder(args).Build().Run();
        var host = CreateWebHostBuilder(args).Build();
        using (var scope = host.Services.CreateScope())
        {
            var services = scope.ServiceProvider;

            try
            {
                var context = services.GetRequiredService<DataContext>();
                SampleData.Initialize(context);
            }
            catch (Exception ex)
            {
                var logger = services.GetRequiredService<ILogger<Program>>();
                logger.LogError(ex, "An error occurred seeding the DB.");
            }
        }
        host.Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

Б.5 Текст файла SampleData.cs

```
public class SampleData
{
    public static void Initialize(DataContext context)
    {
        if (!context.Rates.Any())
        {
            context.Rates.AddRange(
                new Rate { Rate_name = "USD", Val = 28.05 },
                new Rate { Rate_name = "EUR", Val = 34.25 },
                new Rate { Rate_name = "GBP", Val = 39.50 },
                new Rate { Rate_name = "PLN", Val = 7.90 }
            );
            context.SaveChanges();
        }
    }
}
```

```

    }
    if (!context.Statuses.Any())
    {
        context.Statuses.AddRange(
            new Status { Status_name = "A", daily = 5000, mobile = 1000, hospitality = 2000,
transport = 5000, residence = 1000, unexpected = 10000 },
            new Status { Status_name = "B", daily = 500, mobile = 300, hospitality = 1000,
transport = 2500, residence = 700, unexpected = 2000 },
            new Status { Status_name = "C", daily = 300, mobile = 300, hospitality = 500,
transport = 1000, residence = 500, unexpected = 500 },
            new Status { Status_name = "D", daily = 200, mobile = 100, hospitality = 100,
transport = 1000, residence = 300, unexpected = 200 }
        );
        context.SaveChanges();
    }
    if (!context.Countrys.Any())
    {
        context.Countrys.AddRange(
            new Country { Country_name = "Україна", RateId = 0 },
            new Country { Country_name = "Німеччина", RateId = 2 },
            new Country { Country_name = "Польща", RateId = 4 },
            new Country { Country_name = "Америка", RateId = 1 },
            new Country { Country_name = "Франція", RateId = 2 }
        );
        context.SaveChanges();
    }
    if (!context.Citys.Any())
    {
        context.Citys.AddRange(
            new City { CountryId = 1, City_name = "Київ", Transp_ratio = 1.5, Count_people = "2
954 564", Lat = 50.4501, Lng = 30.5234 },
            new City { CountryId = 1, City_name = "Харків", Transp_ratio = 1.5, Count_people = "1
430 885", Lat = 49.9935, Lng = 36.23038 },
            new City { CountryId = 1, City_name = "Львів", Transp_ratio = 1.5, Count_people = "721
301", Lat = 49.8397, Lng = 24.0297 },
            new City { CountryId = 5, City_name = "Париж", Transp_ratio = 1.5, Count_people = "2
175 601", Lat = 48.8566, Lng = 2.3522 },
            new City { CountryId = 3, City_name = "Варшава", Transp_ratio = 1.5, Count_people = "1
790 658", Lat = 52.2297, Lng = 21.0122 },
            new City { CountryId = 2, City_name = "Берлін", Transp_ratio = 1.5, Count_people = "3
664 088", Lat = 52.5200, Lng = 13.4050 }
        );
        context.SaveChanges();
    }
    if (!context.Users.Any())
    {
        context.Users.AddRange(

```

```

        new User { Last_name = "Сорокін", First_name = "Андрій", Patronymic_name =
"Олександрович",
                    Sam_name = "adm", Password = "adm", Position = "Системний адміністратор",
                    Telephone_number = "+380687777720", Email = "sorokinol@gmail.com", Role = 3
, Status = "" }
            );
            context.SaveChanges();
        }
    }
}

```

Б.6 Текст файлу Startup.cs

```

public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }
    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        string connection = Configuration.GetConnectionString("DefaultConnection");
        services.AddDbContext<DataContext>(options => options.UseSqlServer(connection));
        services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseHsts();
        }
        //app.UseHttpsRedirection();
        app.UseDefaultFiles();
        app.UseStaticFiles();
        app.UseMvc();
    }
}

```

Б.7 Текст файла City.cs

```
public class City
{
    public int CityId { get; set; }
    public int CountryId { get; set; }
    public string City_name { get; set; }
    public double Transp_ratio { get; set; }
    public string Count_people { get; set; }
    public double Lat { get; set; }
    public double Lng { get; set; }
}
```

Б.8 Текст файла Country.cs

```
public class Country
{
    public int CountryId { get; set; }
    public string Country_name { get; set; }
    public int RateId { get; set; }
}
```

Б.9 Текст файла Rate.cs

```
public class Rate
{
    public int RateId { get; set; }
    public string Rate_name { get; set; }
    public double Val { get; set; }
}
```

Б.10 Текст файла Status.cs

```
public class Status
{
    public int StatusId { get; set; }
    public string Status_name { get; set; }
    public double daily { get; set; }
    public double mobile { get; set; }
    public double hospitality { get; set; }
    public double transport { get; set; }
    public double residence { get; set; }
}
```

```

        public double unexpected { get; set; }
    }

```

Б.11 Текст файла User.cs

```

public class User
{
    public int UserId { get; set; }
    public string Last_name { get; set; }
    public string First_name { get; set; }
    public string Patronymic_name { get; set; }
    public string Sam_name { get; set; }
    public string Password { get; set; }
    public string Position { get; set; }
    public string Telephone_number { get; set; }
    public string Email { get; set; }
    public int Role { get; set; }
    public string Status { get; set; }
}

```

Б.12 Текст файла DataContext.cs

```

public class DataContext : DbContext
{
    public DbSet<Rate> Rates { get; set; }
    public DbSet<City> Citys { get; set; }
    public DbSet<Country> Countrys { get; set; }
    public DbSet<Status> Statuses { get; set; }
    public DbSet<User> Users { get; set; }
    public DbSet<Request> Requests { get; set; }

    public DataContext(DbContextOptions<DataContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }
}

```

Б.13 Текст файла EmailService.cs

```

public class EmailService
{
    public void SendEmail(string email, string subject, string message)
    {
        try

```

```

{
    var emailMessage = new MimeMessage();

    emailMessage.From.Add(new MailboxAddress("Бухгалтер сайта", "zaikineu@outlook.com"));
    emailMessage.To.Add(new MailboxAddress("", email));
    emailMessage.Subject = subject;
    emailMessage.Body = new TextPart(MimeKit.Text.TextFormat.Html)
    {
        Text = message
    };

    using (var client = new SmtpClient())
    {
        client.Connect("smtp-mail.outlook.com", 587, false);
        client.Authenticate("zaikineu@outlook.com", "Natali1977");
        client.Send(emailMessage);

        client.Disconnect(true);
    }
}
catch (Exception ex)
{
    string mes = ex.Message;
}
}
}

```

Б.14 Текст файлу ValuesControllers.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using TripExpenses.Models;

namespace TripExpenses.Controllers
{
    [Produces("application/json")]
    [Route("api/[controller]")]
    //[ApiController]
    public class ValuesController : Controller

```

```

{
    DataContext db;
    public ValuesController(DataContext context)
    {
        db = context;
    }
    private MyType GetParamFromReq<MyType>(Stream body)
    {
        string param_str = null;
        Object obj = null;

        using (StreamReader reader = new StreamReader(body, Encoding.UTF8))
        {
            param_str = reader.ReadToEnd();
        }

        if (!(param_str == null || string.IsNullOrEmpty(param_str)))
        {
            try
            {
                Stream ms = new MemoryStream(ASCIIEncoding.Default.GetBytes(param_str));
                DataContractJsonSerializer jsonFormatter = new
DataContractJsonSerializer(typeof(MyType));
                obj = jsonFormatter.ReadObject(ms);
                ms.Close();
            }
            catch //(Exception ex)
            {
                obj = param_str;
            }
        }

        return (MyType)obj;
    }
    private static string To_Upper(string str)
    {
        if (str.Length > 0) { return Char.ToUpper(str[0]) + str.Substring(1); }
        return "";
    }
    private void AddDb(string table, dynamic row)
    {
        switch (table)
        {
            case "User_List":
                User user = new User();
                foreach (dynamic field in row)
                {
                    switch (To_Upper(field.Name))

```

```

{
    case "Last_name":
        user.Last_name = field.Value;
        break;
    case "First_name":
        user.First_name = field.Value;
        break;
    case "Patronymic_name":
        user.Patronymic_name = field.Value;
        break;
    case "Sam_name":
        user.Sam_name = field.Value;
        break;
    case "Password":
        user.Password = field.Value;
        break;
    case "Position":
        user.Position = field.Value;
        break;
    case "Telephone_number":
        user.Telephone_number = field.Value;
        break;
    case "Email":
        user.Email = field.Value;
        break;
    case "Role":
        user.Role = (field.Value == "" ? 0 : field.Value);
        break;
    case "Status":
        user.Status = (field.Value == "" ? 0 : field.Value);
        break;
}
}
db.Users.Add(user);
db.SaveChanges();
break;
case "country_List":
    Country country = new Country();
    foreach (dynamic field in row)
    {
        switch (To_Upper(field.Name))
        {
            case "Country_name":
                country.Country_name = field.Value;
                break;
            case "RateId":
                country.RateId = (field.Value == "" ? 0 : field.Value);
                break;

```

```

    }
}
db.Countrys.Add(country);
db.SaveChanges();
break;
case "city_List":
    City city = new City();
    foreach (dynamic field in row)
    {
        switch (To_Upper(field.Name))
        {
            case "CountryId":
                city.CountryId = (field.Value == "" ? 0 : field.Value);
                break;
            case "City_name":
                city.City_name = field.Value;
                break;
            case "Transp_ratio":
                city.Transp_ratio = (field.Value == "" ? 0 : field.Value);
                break;
        }
    }
    db.Citys.Add(city);
    db.SaveChanges();
    break;
case "rate_List":
    Rate rate = new Rate();
    foreach (dynamic field in row)
    {
        switch (To_Upper(field.Name))
        {
            case "City_name":
                rate.Rate_name = field.Value;
                break;
            case "Transp_ratio":
                rate.Val = (field.Value == "" ? 0 : field.Value);
                break;
        }
    }
    db.Rates.Add(rate);
    db.SaveChanges();
    break;
case "status_List":
    Status stat = new Status();
    foreach (dynamic field in row)
    {
        switch (To_Upper(field.Name))
        {

```

```

        case "Status_name":
            stat.Status_name = field.Value;
            break;
        case "Daily":
            stat.daily = (field.Value == "" ? 0 : field.Value);
            break;
        case "Mobile":
            stat.mobile = (field.Value == "" ? 0 : field.Value);
            break;
        case "Hospitality":
            stat.hospitality = (field.Value == "" ? 0 : field.Value);
            break;
        case "Transport":
            stat.transport = (field.Value == "" ? 0 : field.Value);
            break;
        case "Residence":
            stat.residence = (field.Value == "" ? 0 : field.Value);
            break;
        case "Unexpected":
            stat.unexpected = (field.Value == "" ? 0 : field.Value);
            break;
    }
}
db.Statuses.Add(stat);
db.SaveChanges();
break;
}
}
private void UpdatedDb(string table, dynamic row)
{
    User user = null;
    Country country = null;
    City city = null;
    Rate rate = null;
    Status stat = null;
    foreach (dynamic par in row)
    {
        if (par.Name == "id") {
            int curId = par.Value;
            switch (table)
            {
                case "User_List":
                    user = db.Users.Where(u => u.UserId == curId).FirstOrDefault();
                    break;
                case "country_List":
                    country = db.Countrys.Where(u => u.CountryId == curId).FirstOrDefault();
                    break;
                case "city_List":

```

```

        city = db.Citys.Where(u => u.CityId == curId).FirstOrDefault();
        break;
    case "rate_List":
        rate = db.Rates.Where(r => r.RateId == curId).FirstOrDefault();
        break;
    case "status_List":
        stat = db.Statuses.Where(s => s.StatusId == curId).FirstOrDefault();
        break;
    }
}
if (par.Name == "updated")
{
    dynamic cur = par.Value;
    switch (table)
    {
        case "User_List":
            foreach (dynamic field in cur)
            {
                switch (To_Upper(field.Name))
                {
                    case "Last_name":
                        user.Last_name = field.Value;
                        break;
                    case "First_name":
                        user.First_name = field.Value;
                        break;
                    case "Patronymic_name":
                        user.Patronymic_name = field.Value;
                        break;
                    case "Sam_name":
                        user.Sam_name = field.Value;
                        break;
                    case "Password":
                        user.Password = field.Value;
                        break;
                    case "Position":
                        user.Position = field.Value;
                        break;
                    case "Telephone_number":
                        user.Telephone_number = field.Value;
                        break;
                    case "Email":
                        user.Email = field.Value;
                        break;
                    case "Role":
                        user.Role = field.Value;
                        break;
                    case "Status":

```

```

        user.Status = field.Value;
        break;
    }
}
db.SaveChanges();
break;
case "country_List":
    foreach (dynamic field in cur)
    {
        switch (To_Upper(field.Name))
        {
            case "Country_name":
                country.Country_name = field.Value;
                break;
            case "RateId":
                country.RateId = (field.Value == "" ? 0 : field.Value);
                break;
        }
    }
    db.SaveChanges();
    break;
case "city_List":
    foreach (dynamic field in cur)
    {
        switch (To_Upper(field.Name))
        {
            case "CountryId":
                city.CountryId = (field.Value == "" ? 0 : field.Value);
                break;
            case "City_name":
                city.City_name = field.Value;
                break;
            case "Transp_ratio":
                city.Transp_ratio = (field.Value == "" ? 0 : field.Value) ;
                break;
        }
    }
    db.SaveChanges();
    break;
case "rate_List":
    foreach (dynamic field in cur)
    {
        switch (To_Upper(field.Name))
        {
            case "Rate_name":
                rate.Rate_name = field.Value;
                break;
            case "Val":

```

```

        rate.Val = (field.Value == "" ? 0 : field.Value);
        break;
    }
}
db.SaveChanges();
break;
case "status_List":
    foreach (dynamic field in cur)
    {
        switch (To_Upper(field.Name))
        {
            case "Status_name":
                stat.Status_name = field.Value;
                break;
            case "Daily":
                stat.daily = (field.Value == "" ? 0 : field.Value);
                break;
            case "Mobile":
                stat.mobile = (field.Value == "" ? 0 : field.Value);
                break;
            case "Hospitality":
                stat.hospitality = (field.Value == "" ? 0 : field.Value);
                break;
            case "Transport":
                stat.transport = (field.Value == "" ? 0 : field.Value);
                break;
            case "Residence":
                stat.residence = (field.Value == "" ? 0 : field.Value);
                break;
            case "Unexpected":
                stat.unexpected = (field.Value == "" ? 0 : field.Value);
                break;
        }
    }
    db.SaveChanges();
    break;
}
}
}
}
private void DelDb(string table, dynamic row)
{
    switch (table)
    {
        case "User_List":
            foreach (dynamic field in row)
            {
                if (To_Upper(field.Name) == "UserId")

```

```

    {
        int fieldId = field.Value;
        User user = db.Users.Where(u => u.UserId == fieldId).FirstOrDefault();
        if (user != null)
        {
            db.Users.Remove(user);
            db.SaveChanges();
        }
    }
    break;
case "country_List":
    foreach (dynamic field in row)
    {
        if (To_Upper(field.Name) == "CountryId")
        {
            int fieldId = field.Value;
            Country countr = db.Countrys.Where(c => c.CountryId ==
fieldId).FirstOrDefault();
            if (countr != null)
            {
                db.Countrys.Remove(countr);
                db.SaveChanges();
            }
        }
    }
    break;
case "city_List":
    foreach (dynamic field in row)
    {
        if (To_Upper(field.Name) == "CityId")
        {
            int fieldId = field.Value;
            City city = db.Citys.Where(c => c.CityId == fieldId).FirstOrDefault();
            if (city != null)
            {
                db.Citys.Remove(city);
                db.SaveChanges();
            }
        }
    }
    break;
case "rate_List":
    foreach (dynamic field in row)
    {
        if (To_Upper(field.Name) == "RateId")
        {
            int fieldId = field.Value;

```

```

        Rate rate = db.Rates.Where(r => r.RateId == fieldId).FirstOrDefault();
        if (rate != null)
        {
            db.Rates.Remove(rate);
            db.SaveChanges();
        }
    }
    break;
case "status_List":
    foreach (dynamic field in row)
    {
        if (To_Upper(field.Name) == "StatusId")
        {
            int fieldId = field.Value;
            Status stat = db.Statuses.Where(s => s.StatusId == fieldId).FirstOrDefault();
            if (stat != null)
            {
                db.Statuses.Remove(stat);
                db.SaveChanges();
            }
        }
    }
    break;
}
}

private void SavetoDb(string table, string action, dynamic collect) {
    foreach (var row in collect)
    {
        switch (action)
        {
            case "added":
                AddDb(table, row);
                break;
            case "updated":
                UpdatedDb(table, row);
                break;
            case "deleted":
                DelDb(table, row);
                break;
        }
    }
}

[Route("[action]"), HttpPost]
public JsonResult GetUsersList()
{
    dynamic listdata = db.Users.ToList();
    return Json(listdata);
}

```

```

    }
    [Route("[action]"), HttpPost]
    public JsonResult GetMainData()
    {
        dynamic countrylist = db.Countrys.ToList();
        dynamic citylist = db.Citys.ToList();
        dynamic ratelist = db.Rates.ToList();
        dynamic statuslist = db.Statuses.ToList();
        dynamic requestlist = db.Requests.ToList();
        return Json(new { Country_List = countrylist,
                           City_List = citylist,
                           Rate_List = ratelist,
                           Status_List = statuslist,
                           Requests = requestlist
                        });
    }
    [Route("[action]"), HttpPost]
    public JsonResult SaveDict()
    {
        string param_str = "";
        string error_text = "";
        string Table = "";
        int is_error = 0;
        Dictionary<string, dynamic> Params = new Dictionary<string, dynamic>();

        if (Request.Body != null)
        {
            using (StreamReader reader = new StreamReader(Request.Body, Encoding.UTF8))
            {
                param_str = reader.ReadToEnd();
            }
        }
        var dict = JsonConvert.DeserializeAnonymousType(param_str, new Dictionary<string, object>());

        foreach (KeyValuePair<string, object> kp in dict)
        {
            if (kp.Key == "table")
            {
                Table = kp.Value.ToString();
            }
            else if (kp.Key == "data" && Params.Count() == 0)
            {
                Params = Params.Concat(JObject.Parse(kp.Value.ToString()).ToObject<Dictionary<string,
dynamic>>()).ToDictionary(x => x.Key, x => x.Value);
            }
        }
        foreach (KeyValuePair<string, dynamic> kp in Params)
        {

```

```

        if (kp.Value.Count > 0)
        {
            SavetoDb(Table, kp.Key, kp.Value);
        }
    }
    if (!(string.IsNullOrEmpty(error_text)))
    {
        is_error = 1;
        return Json(new { is_error, error_text });
    }
    if (Table == "User_List")
    {
        return Json(db.Users.ToList());
    }
    else
    {
        dynamic countrylist = db.Countrys.ToList();
        dynamic citylist = db.Citys.ToList();
        dynamic ratelist = db.Rates.ToList();
        dynamic statuslist = db.Statuses.ToList();
        dynamic requestlist = db.Requests.ToList();
        return Json(new
        {
            Country_List = countrylist,
            City_List = citylist,
            Rate_List = ratelist,
            Status_List = statuslist,
            Requests = requestlist
        });
    }
}

[Route("[action]"), HttpPost]
public JsonResult SaveReq()
{
    try
    {
        dynamic countrylist = db.Countrys.ToList();
        dynamic citylist = db.Citys.ToList();
        dynamic ratelist = db.Rates.ToList();
        dynamic statuslist = db.Statuses.ToList();
        AllParams reqData = GetParamFromReq<AllParams>(Request.Body);
        switch (reqData.action) {
            case "Insert":
                Request req = new Request();
                req.user = reqData.user;
                if (!String.IsNullOrEmpty(reqData.country)){
                    string country_name = db.Countrys.Where(c => c.CountryId ==
Int32.Parse(reqData.country)).FirstOrDefault().Country_name;

```

```

        req.country = country_name;
    }
    if (!String.IsNullOrEmpty(reqData.country))
    {
        string city_name = db.Citys.Where(c => c.CityId ==
Int32.Parse(reqData.city)).FirstOrDefault().City_name;
        req.city = city_name;
    }
    req.date_begin = DateTime.ParseExact(reqData.date_begin, "yyyy-MM-dd",
System.Globalization.CultureInfo.InvariantCulture);
    req.date_end = DateTime.ParseExact(reqData.date_end, "yyyy-MM-dd",
System.Globalization.CultureInfo.InvariantCulture);
    req.summaAll = reqData.summaAll;
    req.valutaAll = reqData.valutaAll;
    req.daily = reqData.daily;
    req.mobile = reqData.mobile;
    req.hospitality = reqData.hospitality;
    req.transport = reqData.transport;
    req.residence = reqData.residence;
    req.unexpected = reqData.unexpected;
    db.Requests.Add(req);
    db.SaveChanges();
    break;
case "Update":
    Request req2 = db.Requests.Where(r => r.RequestId ==
reqData.requestId).FirstOrDefault();
    if (req2 != null) {
        req2.summaAll = reqData.summaAll;
        req2.valutaAll = reqData.valutaAll;
        req2.daily = reqData.daily;
        req2.mobile = reqData.mobile;
        req2.hospitality = reqData.hospitality;
        req2.transport = reqData.transport;
        req2.residence = reqData.residence;
        req2.unexpected = reqData.unexpected;
        db.SaveChanges();
    }
    break;
}
dynamic requestlist = db.Requests.ToList();
return Json(new
{
    Country_List = countrylist,
    City_List = citylist,
    Rate_List = ratelist,
    Status_List = statuslist,
    Requests = requestlist
});

```

```

    }
    catch (Exception ex)
    {
        return Json(new { is_error = 1, error_text = ex.Message});
    }
}

[Route("[action]"), HttpPost]
public JsonResult EmailDict()
{
    string param_str = "";
    string error_text = "";
    string EmailAddress = "";
    string FIO = "";
    int is_error = 0;
    Dictionary<string, dynamic> Params = new Dictionary<string, dynamic>();
    if (Request.Body != null)
    {
        using (StreamReader reader = new StreamReader(Request.Body, Encoding.UTF8))
        {
            param_str = reader.ReadToEnd();
        }
    }
    var dict = JsonConvert.DeserializeAnonymousType(param_str, new Dictionary<string, object>());
    foreach (KeyValuePair<string, object> kp in dict)
    {
        if (kp.Key == "email")
        {
            EmailAddress = kp.Value.ToString();
        }
        else if (kp.Key == "user")
        {
            FIO = kp.Value.ToString();
        }
    }
    EmailService emailService = new EmailService();
    emailService.SendEmail(EmailAddress, "Заявка на відрядження узгоджена", "<div><b>Шановний " +
FIO + "! </b></div> <div>Ваша заявка за відрядження узгоджена.</div>");

    if (!(string.IsNullOrEmpty(error_text)))
    {
        is_error = 1;
        return Json(new { is_error, error_text });
    }
    return Json(db.Users.ToList());
}
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет "Запорізька політехніка"
Кафедра програмних засобів

Випускна кваліфікаційна робота бакалавра на тему:

**Програмна реалізація застосунку для організації
обліку витрат на відрядження працівників
підприємства**

Виконала:
ст. гр. КНТ-128

К.С. Заїкіна

Керівник:
к.т.н., доц.

Т.В. Федорончак

Рисунок В.1 – Слайд №1

— ■ ■ —

Об'єкт, предмет та мета роботи

- Об'єкт дослідження – облік витрат на відрядження працівників підприємства.
- Предмет дослідження – вебзастосунки для організації обліку витрат на відрядження працівників підприємства.
- Мета роботи – створення застосунку для організації обліку витрат на відрядження для користування на підприємстві.

Рисунок В.2 – Слайд №2

Завдання роботи

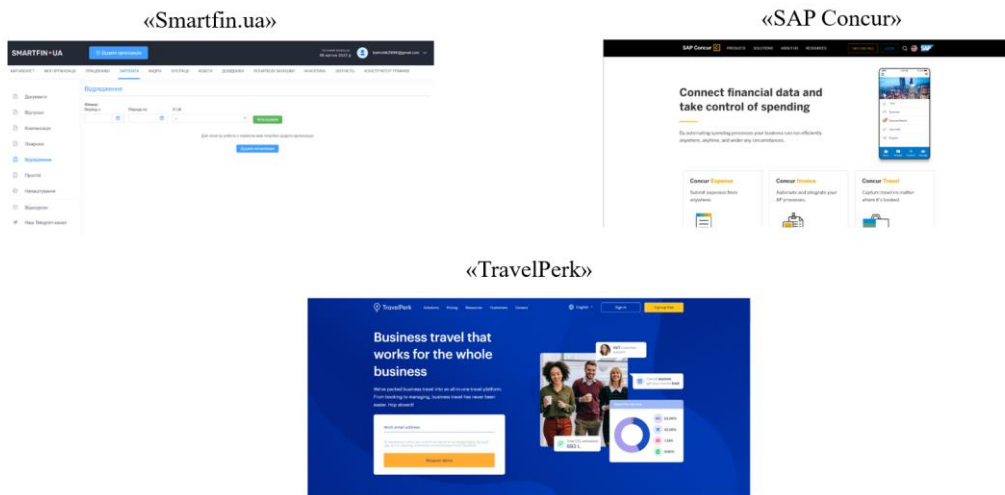
В процесі виконання роботи були поставлені такі задачі:

- проаналізувати предметну область;
- розробити функціональну структуру програми;
- проаналізувати логічну структуру програми;
- розробити структуру бази даних;
- розробити інтерфейс;
- обрати мову програмування та середовище розробки;
- виконати програмну реалізацію;
- провести тестування та виправлення помилок.

3

Рисунок В.3 – Слайд №3

Існуючі аналоги вирішення завдання



4

Рисунок В.4 – Слайд №4

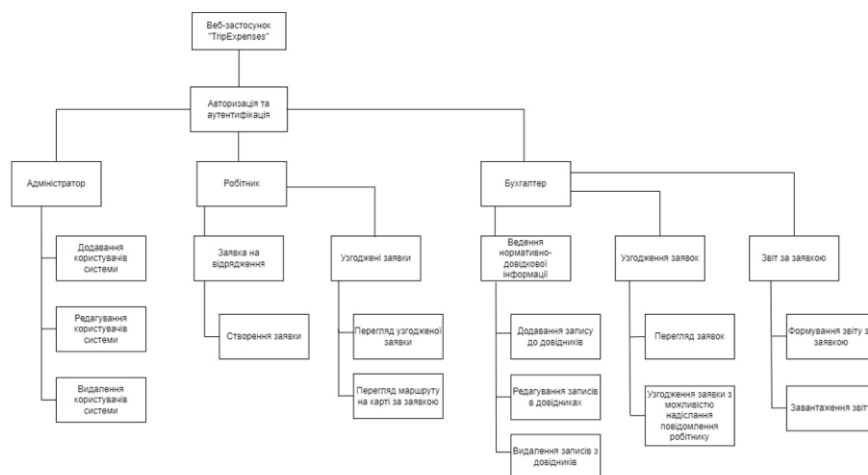
Порівняння аналогів

	«Smartfin.ua»	«SAP Concur»	«TravelPerk»
Виконання основного функціоналу (створення заявок, формування звіту)	+	+	+
Багатоплатформність	+	+	+
Зручний інтерфейс	+	+	+
Платність	+	+	+
Підтримка української мови	+	-	-
Швидкодія	+	-	+
Адаптивність під мобільні пристрої	+	+	+
Співпраця та доступ до серверів замовлення (бронювання готелів, автомобілів, та ін.)	-	-	+

5

Рисунок В.5 – Слайд №5

Схема функцій користувача



6

Рисунок В.6 – Слайд №6

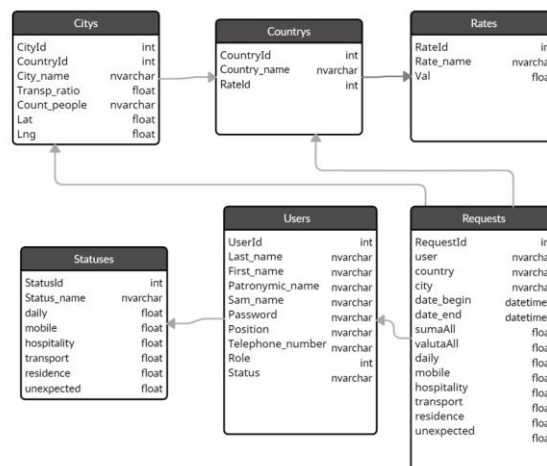
Модулі реалізованого застосунку

Модуль	Опис
Авторизація та аутентифікація користувача	За вказаним акаунтом та паролем визначення ролі користувача, та перехід до відповідної сторінки вебзастосунку.
Створення заявки на відрядження	Введення даних по працівнику, який відправляється у відрядження, країна і місто, в яку відправляється працівник та період відрядження.
Перегляд узгоджених заявок	Перегляд заявок, які узгоджені бухгалтером підприємства.
Довідники	Введення нормативно-довідкової інформації про країни, міста, статуси користувача, курсів валют, працівників підприємства.
Узгодження заявок	Розрахунок тарифів на відрядження для кожного співробітника, в залежності від його статусу на підприємстві з можливістю відправки повідомлення робітнику.
Звіт за заявками	Виведення основної інформації за заявкою відрядження та її завантаження.
Список користувачів в системі	Введення, видалення, редагування інформації по користувачам в системі.

7

Рисунок В.7 – Слайд №7

Схема бази даних



8

Рисунок В.8 – Слайд №8

Вибір мов програмування

	JS+C#.Net	JS+PHP
Спосіб виконання	Компілюється	Інтерпретується
Багатопоточність	+	-
Асинхронність	+	-
Захист даних від sql-ін'єкцій	+	-
Колекції	Безліч різних колекцій	Тільки масиви
Типізація	+	-
Розмір проектів	Придатний для великих проектів	Кращий для невеликих проектів
Швидкість бібліотек доступу до бази даних	Швидше	Повільніше
Обсяг коду	Трохи менш компактний	Компактний
Відлагодження	Легше проводити відлагодження	Складно проводити відлагодження
Середовище розробки	IDE високого класу	IDE середнього класу
Швидкість	Виконується на багато швидше, коли побудований в режимі випуску, оптимізований, кешований	Виконується досить швидко

9

Рисунок В.9 – Слайд №9

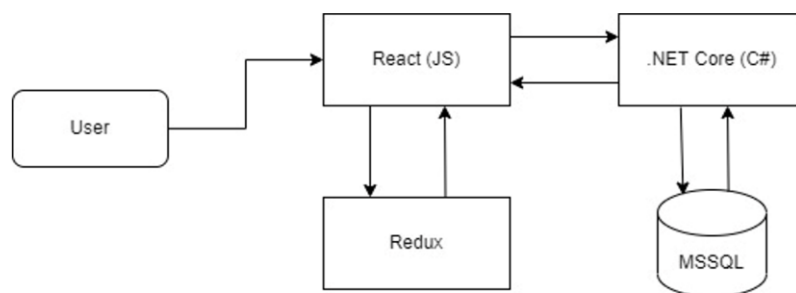
Вибір середовища розробки

	Visual Studio 2017	Visual Studio Code	IntelliJ Idea
Багатоплатформність	+	+	-
Наявність візуального конструктора	+	+	+
Підтримка репозиторіїв	+	+	+
Автозавершення	+	+	+
Розширення	+	+	+
Відлагодження програми	+	+	+
Навігація по проєкту	+	+	+
Робота з базами даних	+	+	-
Компілятор	+	+	+
Перевірка орфографії	+	+	+
Рефакторинг	+	+	+
Статичний аналіз коду	+	+	+
Портативність	-	-	-
Займає невелику кількість пам'яті	-	+	-

10

Рисунок В.10 – Слайд №10

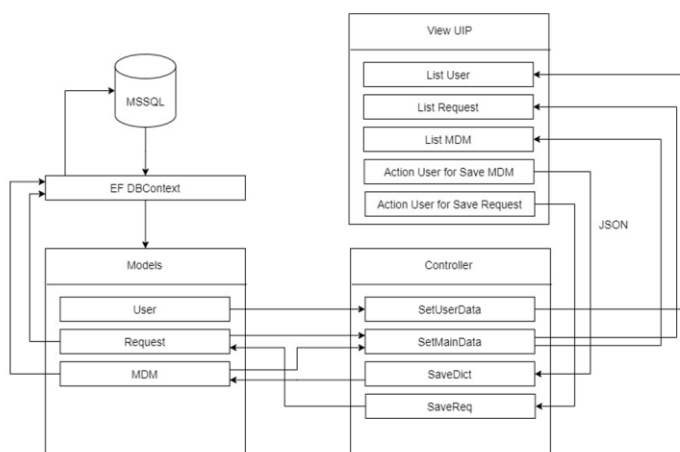
Схема архітектури вебзастосунку



11

Рисунок В.11 – Слайд №11

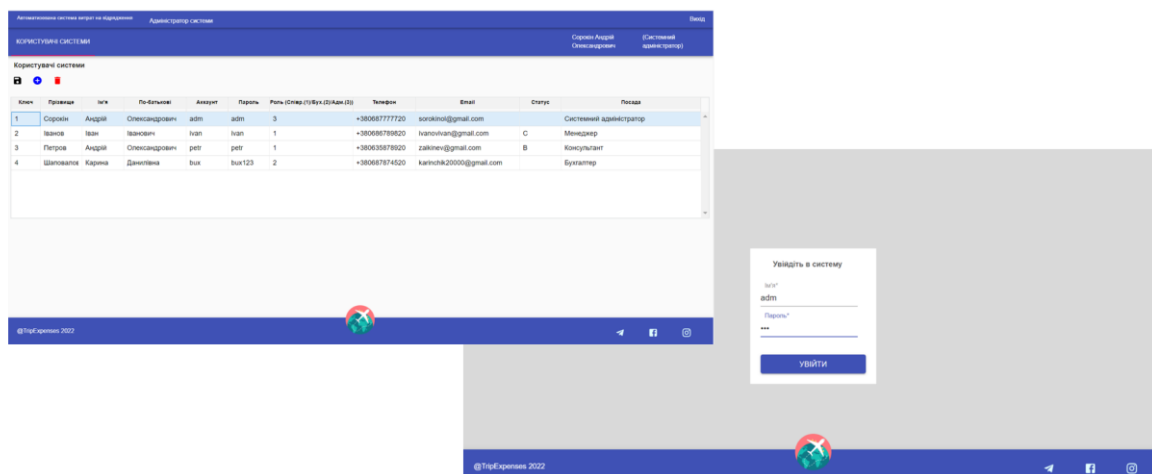
Структурна схема вебзастосунку



12

Рисунок В.12 – Слайд №12

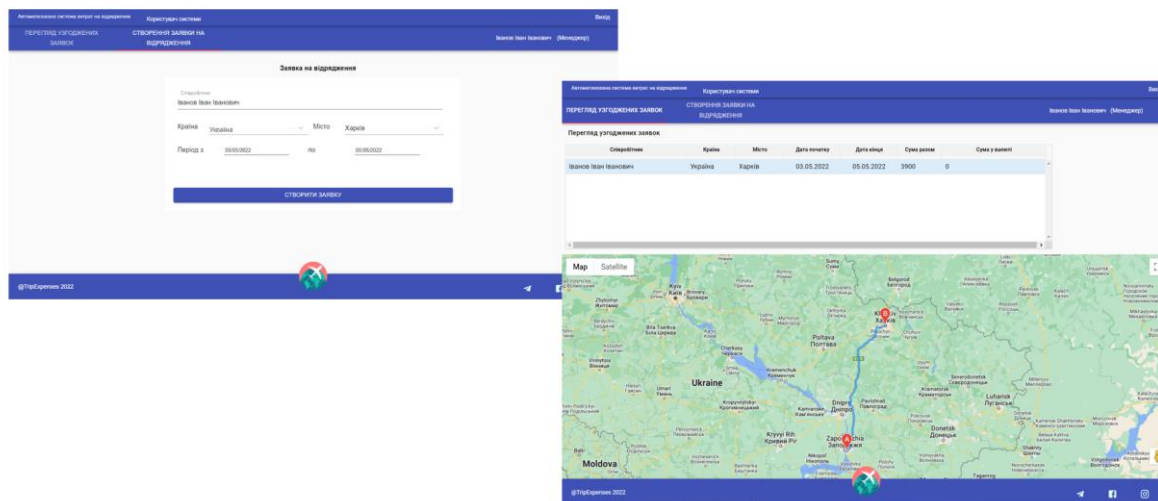
Сторінка авторизації та адміністратора системи



13

Рисунок В.13 – Слайд №13

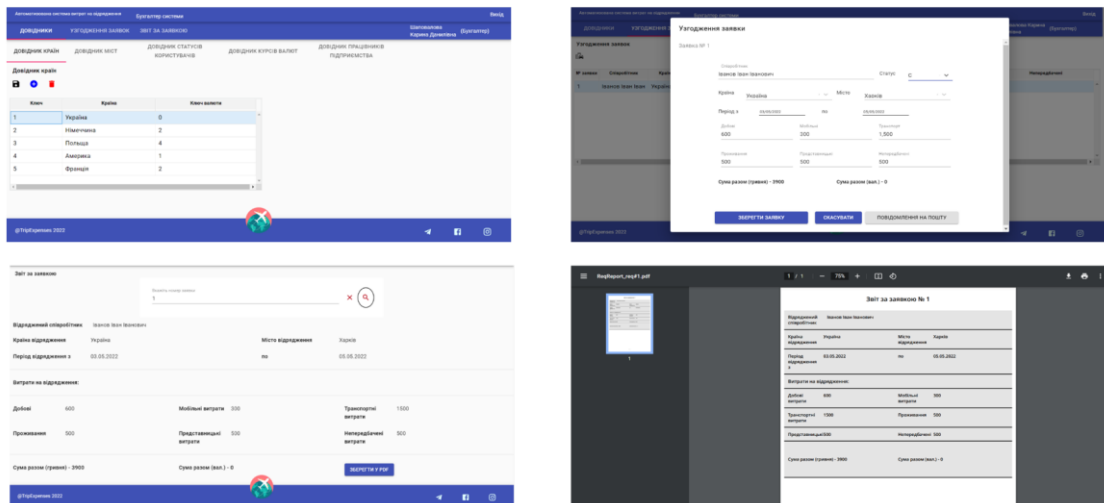
Сторінка робітника системи



14

Рисунок В.14 – Слайд №14

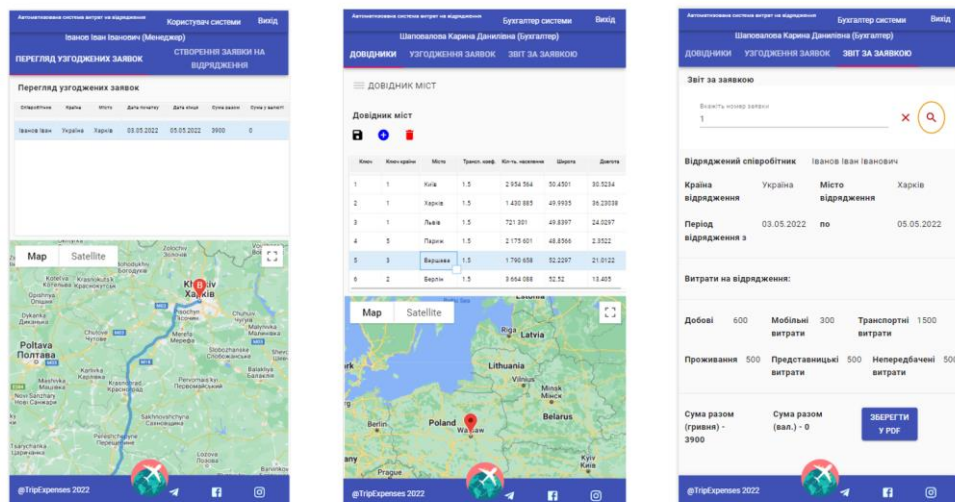
Сторінка бухгалтера системи



15

Рисунок В.15 – Слайд №15

Адаптація під мобільні пристрої



16

Рисунок В.16 – Слайд №16



Висновки

- В результаті виконання дипломної кваліфікаційної роботи було проведено детальне ознайомлення із предметною областю, розглянуто аналоги застосунку, виділено основні переваги та недоліки застосунків. На підставі цього були поставлені функціональні вимоги та наведено функціональну структуру у вигляді схеми.
- Для реалізації ПЗ були обрані основні інструменти веброзробника, для клієнтської частини обрано JavaScript з використання бібліотеки React та Redux, в якості інтегрованого середовища розробки – Visual Studio Code. Для реалізації серверної частини вибір припав на мову програмування C# з використанням платформи .NET Core, середовище розробки – Visual Studio версії 17.0. Для системи управління базами даних було обрано Microsoft SQL Server.
- У розробленому програмному забезпеченні реалізовано створення та узгодження заявок, розрахунок витрат на відрядження, побудова та завантаження звіту, зберігання даних про користувачів, заявок, довідників, тощо. ПЗ було протестовано та перевірено на відображення коректних повідомлень про помилку.
- Розроблене програмне забезпечення відповідає вимогам технічного завдання та може експлуатуватися за призначенням.