Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерних систем і програмної інженерії
(назва факультету)
Кафедра інженерії програмного забезпечення
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: Розробка інформаційної веб платформи на основі технології C# для автоматизації документообігу на виробництві

Виконав: студент (ка) 6 курсу, групи СПм-62
спеціальності (напряму підготовки)
121 інженерія програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

	(підпис)	Гілюта М.І. (прізвище та ініціали)
Керівник	(підпис)	Ясній О.П. (прізвище та ініціали)
Нормоконтроль	(підпис)	Бойко І.В. (прізвище та ініціали)
Рецензент	(підпис)	Дмитроца Л.П. (прізвище та ініціали)

м. Тернопіль – 2019

Анотація

Робота містить 125 сторінок 5 таблиць і 14 рисунків , список літератури з 40 найменувань , 3 додатки

Актуальність теми полягає в тому, що сучасні транспортно-логістичні фірми все більше ресурсів різного роду виділяють на побудову чіткої та стабільної архітектури на базі інформаційних систем та інноваційних підходів до розробки даних у середині компанії та міжперсональних комунікаціях. В умовах швидкого економічного та технічного розвитку постає питання адаптації діючих систем до виконання фіксованих завдань з оптимізації та логістичної компетентності у фірмах з різною складністю і підходом до вирішення проблем і питань транспортування, разом з тим багато часу та інформації віддається на оптимальний пошук та розподіл маршрутного планування і витоків конструктивного забезпечення.

Об’єктом дослідження є пошук оптимальних шляхів для забезпечення скоординованої роботи логістичної компанії в умовах підвищеної складності вантажообігу та звернень водіїв з аналізом локального ресурсу для операційних систем типу Windows.

Метою роботи є побудови робочої архітектури для автоматизованої роботи з вантажообігу і скоординованої роботи операторів у системному аналізі рішень логістики з відходом до реалізації поширення закільцьованих маршрутів.

КЛЮЧОВІ СЛОВА: ПРОГРАМНА СИСТЕМ, ЛОГІСТИКИ, ТРАНСПОРТНІ ПЕРЕВЕЗЕННЯ, АВТОМАТИЗАЦІЯ

Abstract

The work contains 125 pages 5 tables and 14 figures, a list of literature of 40 titles, 3 appendices

The relevance of the topic is that modern transportation and logistics companies are increasingly allocating resources of all kinds to building a clear and stable architecture based on information systems and innovative approaches to data development within the company and interpersonal communications. In the context of rapid economic and technical development, the question arises of adapting existing systems to the execution of fixed optimization tasks and logistical competence in firms with different complexity and approach to solving problems and transport issues, while much time and information is devoted to optimal search and distribution of route planning. and leaks of structural support.

The object of the study is to find the best ways to ensure the coordinated operation of the logistics company in the context of increased complexity of turnover and appeals to drivers with local resource analysis for Windows operating systems.

The purpose of the work is to build a working architecture for automated work on truck traffic and coordinated work of operators in the systematic analysis of logistics solutions with the waste to the implementation of the distribution of ring routes

KEYWORDS: SOFTWARE, LOGISTICS, TRANSPORTATION, AUTOMATION

ЗМІСТ

Вступ

Стрімкий розвиток сфери логістичних транспортних перевезень спричиняє велику навантаженість вантажообігом і стимулює комп'ютерні технології долучатися до процесу вирішення питань доповнення виробництва та руху вантажів у метриці потоку і обігу товарів. Зі збільшенням навантаженості галузі виникає потреба у більш оптимізованих рішеннях та систематичних аналізах галузі із залучення сучасних технологій і ресурсів економічного зростання.

Питання оптимізації маршруту є ключовим у побудові плану на поїздку з вигідного кошторису та застосунку на всіх стадіях перевезення, але найбільше на початковій фазі , коли відбувається планування та розрахунок поїздки для водіїв. Є необхідність враховувати багато факторів , таких як витрати на пальне , часові обмеження водіїв, вигідний підбір локацій для загрузки та вигрузки товару з урахуванням роботи служб і сервісів сумісних та прилеглих до сфери вантажоперевезень. Одним з ключових завдань логістичної компанії є правильне проектування та розподіл маршруту і товарів між водіями і регулювання витрат пов'язаних із поїздками та транспортним обслуговуванням.

Модель продукту матиме аналітичне та графічне представлення , що можна буде візуалізувати дані про рух транспорту та виконати правки у відповідності до завдань які потрібно реалізовувати з тими ж заданими функціями розподілу , на які розрахована система. Таким чином ми побачимо можливі варіанти руху транспорту в послідовності до тимчасового або тривалого пожитку і створення таких моделей розвитку руху при яких буде відношення показників відстань затрати збігатися з потребами замовників і можливостями перевізників. З розвитком інформаційних технологій усе більше актуальності набувають програмні продукти з можливістю моделювання можливих варіантів розвитку транспортних розв'язків, які можуть бути

представлені у вигляді послідовної графічної інформації, яку в подальшому буде опрацьовувати оператор і приймати рішення для завчасного планування графіку руху водія. Завдяки успішному архітектурному та програмному плануванню і побудові стійкої стабільної системи аналізу руху транспорту логістична компанія має змогу значно заощадити витрати на максимізувати прибутки від своєї діяльності з додатковими ефективними накопиченнями у вигляді своєчасного виконання поставлених цілей і досягнення результативності у діловому світі. Отже , за мету було поставлено розробити систему , яка за допомогою алгоритмів і зручного інтерфейсу може подавати актуальну інформацію про стан руху водіїв і допоможе фірмі здійснювати ефективну діяльність у сфері логістичних перевезень.

В якості реалізації було вирішено виконати програмний продукт, з можливістю задання та накопичення відповідних параметрів, які будуть необхідними для роботи алгоритмів, на основі яких будуть будуватися графіки та діаграми руху , згідно з якими оператор може виконати якісну оцінку руху транспорту , та проектування суміжних маршрутів по визначенню яких буде можливо прокладати дорожні карти і реалізовувати комерційні потреби вантажообігу. Такий підхід до розробки моделі дозволить наочно проводити оцінку , що в свою чергу вирішить багато нетривіальних проблем пов’язаних із укладанням карт руху та оцінки фінансових ризиків при плануванні поїздки. Також завдяки швидкодії сучасного програмного проектування і комп’ютерної архітектури на базі рідкокристалічних процесорів буде підвищена точність при оцінці результатів.

Важливим елементом та хорошим проектним рішенням стане можливість локальної та віддаленої актуалізації та оновлення значення при підтримці ефективного коефіцієнту оцінки.

До моменту розробки автоматизованої системи обліку логістичної фірми існує декілька схожих програмних рішень, проте вони не мають можливості достатнього мобільного оновлення і застосування сучасних методів розробки та

підтримки систем забезпечення і оцінки, що відрізнялися б швидкодією і гнучкістю у можливостях побудови моделей руху та складанню можливих варіантів маршрутного транспортування . Саме з високої актуальності та відсутності готових програмних рішень для задоволення потреб ринку і було прийнято рішення взятися за розробку даного проекту в якості дипломної роботи.

Елементами новизни у даній роботі є алгоритми пошуку вантажів та руху транспорту з графічним представленням на карті і можливістю внесення змін до параметрів руху водія та динамічного задання вантажообігу в якості додаткового кошторисного додатку. Все це дозволяє з глибокою точністю і наочністю представляти та аналізувати результати роботи руху вантажного транспорту і приймати ефективні рішення для його покращення і реального показника спрощення в умовах швидкого розвитку.

В результаті виконання роботи було виконано веб-додаток за допомогою якого можна аналізувати вхідні дані , та завантажувати дані роботи на сервер і навпаки, отримувати дані про роботи водіїв у вигляді зручного та наочного файлу. Це дозволяє ефективно здійснювати керування фінансами і виконувати запити користувачів у різних галузях виробництва вантажообігу. Зручний графічний інтерфейс дозволяє швидко та зручно здійснювати взаємодію з користувачем або працівником і міняти задані параметри на актуальні , таким чином динамічно розвантажувати складні та нетипові проблеми руху .

Представлена нижче дипломна робота описує процес архітектури та основні етапи створення логістичного веб-додатку.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

Предметна область - це та сфера знань в якій буде відбуватися діяльність і , котра підлягатиме найбільшого впливу та інтересу під час дослідження. Предметна область для користувача є додаток адреси програмного продукту. Кожен її компонент характеризується значним кількісним впливом параметрів для яких визначена сутність їх взаємодії у вигляді картографічної схеми та користувачами , які мають можливість взаємодіяти із заданою множиною.

У якості предметної області вибирається задача планування маршруту поїздки водія і фінансового розподілу на основі графічного представлення та математичних моделей. Внаслідок цього завдання полягає у математичному та графічному моделюванні алгоритмів, які дозволять скласти план поїздки та визначити їх вартість з урахуванням таких факторів як відстань , якість і час.

Для отримання графічного представлення і математичного аналізу моделі планування поїздки необхідно проаналізувати графіки руху вантажного транспорту та дослідити ринок в умовах високої інтенсивності навантаження і встановити залежності між вантажообігом і заданими параметрами. Для цього було зроблено глибокий аналіз ринку вантажообігу та зібрано необхідні дані для аналізу.

Модель перевірена та проаналізована шляхом порівняльних методологічних підходів і математичним моделюванням.

1.1.1 Постановка задачі

Після аналізу предметної області , визначено розробити визначення основних функцій і систему планування картографічних систем та маршрутів із аналізом побудови маршруту та представленням графічного шляху подорожі для веб-додатку, яка включатиме :

-можливість вход

-можливість авторизації

-можливість додавання нового водія

- можливість редагування записів маршруту водія

-можливість введення параметрів , які включатимуть ціну, відстань , та швидкість доставки товару

-можливість побудови маршруту

-можливість резервного копіювання

- можливість побудови графічного представлення маршруту

- можливість аналізу витрат і планування фінансів

-можливість завантажувати дані про водіїв у форматі pdf

-використання реактивних технологій

-сервер, для обчислень оптимальних маршрутів

-застосування документно-орієнтованих баз даних у вигляді блоків

1.1.2 Пошук акторів та варіантів використання

Для моделювання обраної системи ключовим моментом буде розподіл її на виконавчі частини з допомогою розроблених функцій UML та пошуку акторів з урахуванням динамічної зміни сутностей системи. Такий підхід дозволить мати достовірні уявлення про поведінку системи в будь-який момент часу.

В рамках мови UML всі уявлення про модель складної системи фіксуються у вигляді спеціальних графічних конструкцій, що отримали назву діаграм прецедентів[1]. UML є представлена за допомогою концепції моделювання діаграм , яких налічується п’ять видів, що уможливлює використання складних характеристик динамічного концептивного виконання системи. Однією з найбільш важливих видів діаграм для моделювання є діаграма варіантів використання , яка показує розподіл системи за допомогою виділення окремих її сутностей або акторів, також відома як діаграма прецедентів (use case diagram) - вона є найбільш загальною і в той же час наочною моделлю представлення складної системи, яка містить в собі функціональний опис системи. [2] При цьому слід враховувати , що система може виступати як актор прецедентів у вигляді технічного засобу, людини, або іншого пристрою , який програміст визначить як узагальнену модель представлення прецедента, яка може слугувати основою впливу на систему , модельовану так, як визначить розробник. З іншого боку система чинить значний вплив з використання істинних джерел судження на яких грунтується підбір значень і подальша дієздатність системи з урахуванням таких факторів як ергономічність та динамічність сутностей акторів і визначає особливий набір послідовностей інструкцій, який виконує система при діалозі з системою.

Діаграма містить в собі абстракції щодо функціонування систем або їх часткового представлення у вигляді підсистем різної складності і таким чином в подальшому перебігу частково або загальним чином використовується більше однієї діаграми, які позначають певну функціональну частину механізму роботи системи.

Діаграми прецедентів використовується в наступних цілях:

- Узагальнений аналіз вимог до механізму роботи системи
- Абстрактне високорівневе представлення системи у вигляді діаграм
- Наочне та послідовне поєднання всіх компонентів системи
- Аналіз варіантів використання
- Супровід і поєднання усіх життєвих циклів проекту
- Оновлення змін і наочне представлення функціоналу роботи всієї системи продукта

Перед тим як проектувати та визначити діаграми системи використання слід визначити такі компоненти:

- функціональні особливості і характеристики системи, які мають бути представлені у вигляді варіантів використання
- власне актори або прецеденти
- відношення між акторами в системі та варіантами їх використання

Окремий варіант представлення характеризується своїм типом сутності і має представлення на діаграмі у вигляді еліпса всередині якого або біля нього , частіше всього під ним міститься його узагальнена скорочена назва у формі іменника або дієслова з поясненням до нього.

Мета з якою проектуються та визначаються сутності актори з варіантами використання полягає у тому, щоб визначити закінчений аспект або представлення певної частини сутності програмного проекту засобами діаграм та сутностей. В якості таких елементів сутностей може бути представлена будь-яка частина проекту програми, в якій відслідковується індивідуальний поведінковий патерн подібно системі або частини підсистемі сутності.

Актор - це поведінковий патерн, який виконується сутностями у вигляді ролі , яку приймають за частини елементу системи або підсистеми в залежності з

її специфікаціями і суміжними ланками виконання. Такі частини використовують ресурси механізму системи для досягнення своєї мети або вирішення індивідуальних задач. При цьому актори сутностей варіантів використання використовуються для узгодженої і злагодженої роботи значної кількості прецедентів ролей, які можуть грати інші користувачі або сутності в процесі робочої взаємодії та симуляції.

За допомогою засобів представлення мовою UML визначено такі види відношення між представленими сутностями акторів та варіантами їх використання, а саме :

Відношення залежності - таке відношення між сутностями, при якому зміна стану першої (незалежної) впливає на стан іншої (залежної) сутності.

Відношення асоціації - структурне відношення, що описує зв'язок, тобто з'єднання між об'єктами (з різними типами зв'язків). На основі асоціацій виконується навігація по елементах відношення.

Відношення узагальнення (агрегування) - відношення показує що один елемент (дочірній, частина) є структурною частиною іншого (батьківського, цілого) .Можливо композитне агрегування. Сутність є частина цілого і має з ним єдиний час життя.

Відношення реалізації - відношення показує що один елемент визначає контракт (зобов'язання), а інший виконує цей контракт.[3]

Для проектованої системи управління логістичного документообігу фірми головними актором є користувач. Його основні можливості з роботою системи можна відобразити у вигляді діаграм варіантів використання. Система розрахована як для вільного так і локального користування окремою фірмою або її частиною, тому і варіанти використання та архітектурні особливості підібрані належним чином. Згідно наших вимог додаток проектується для використання

як веб ресурс з можливістю маршрутного планування та проектування графічних карт маршрутів руху водіїв з можливістю їх подальшої корекції і експортуванням. У системі не планується жодних взаємодій між акторами користувачами тому можливо припустити, що рішення не володіє складними архітектурними особливостями в плані багатокористувацьких можливостей і значної кількості ролей. Система зосереджена на вузькій аудиторії користувачів , в основному це працівники логістичних фірм або їх власники, тому важливо врахувати унікальні особливості галузі логістики при проектуванні варіантів використання і побудові сценаріїв їх взаємодії із системою. В роботі із програмою будуть присутні два види користувачів: гість та авторизований користувач. Авторизований Користувач , має свій обліковий запис, перед попередньою реєстрацією в системі, до нього буде прикріплена відповідна кількість робочих фірм , з якими і буде відбуватися основна взаємодія, а гість буде відправлений на сторінку авторизації з можливістю реєстрації.

1.1.3 Опис ключових варіантів використання

Як ми уже розглянули в заданій системі будуть присутні два актори виконавці , в нашому проекті це буде Гість та Користувач. Гість представлений у системі як виконавець актор з сильно врізаними привілегіями та обмеженими функціональними можливостями, на відміну від актора Користувач, Гість не може додавати нових поїздок для водіїв, контролювати їх витрати, оформляти замовлення та позначати їх на мапі, розробляти плани витрат і позбавлений багатьох основних інструментів взаємодії. При стартовому запуску програми відбувається визначення типу користувача і відповідна класифікація до одного з варіантів використання застосунка , а саме до авторизованого чи неавторизованого користувача.

Для того, щоб отримати більш критичну оцінку корисності кожного виділеного персонажа, був складений графік випадків використання із зображенням ключових акторів.

На рисунку 1.1 зображено діаграму варіантів цих акторів.

Рисунок 1.1 - Варіанти використання

З обширним та детальним описом варіантів програмного використання можна ознайомитися в таблиці 1.1.

Кожне з полів "Логін" та "Вхід" використовує міркування про три різних використання, які дозволяють індивідуальним варіантами підтвердити рамки їх механізмів. Альтернативною автентифікації для використання, "Перегляд категорії витрат" включає додаткову активність, щоб побачити час та сукупність завершення всіх класифікацій витрат.

Таблиця 1.1 – Опис ключових варіантів використання системи

код	Найменування	Основний сценарій	Альтернативний сценарій
А1	Зміна локації	1. Користувач або Гість потрапляє на екран з налаштуваннями 2. Обирає у вікні з один з варіантів локації	
А2	Створити проект	1. Користувач або Гість відкриває екран створення проекта або вперше відкриває програму. 2. Вводить інформацію про проект 3. Натискає на кнопку створити проект 4.	2.a. Не заповнені обов’язкові поля 2. a.1. Виводиться повідомлення про те, яких даних не вистачає 2.b. Введено

		Користувач або Гість перенаправляється на екран загальної інформації про проект	невалідні дані Виводиться повідомлення про те, які дані невалідні
А3	Реєстрація за допомогою email і пароля	1. Гість переходить на екран реєстрації 2. Заповнює поля email, password, repeat password, country, zip code 3. Натискає кнопку “Sign up” 4. Гість авторизується в системі	3. a. Гість ввів не валідні дані 3.a.1. Виводиться повідомлення про те, що дані не валідні 3.b. Користувач з таким email вже зареєстрований 3.b.1. На екрані з’являєься вікно з помилкою

A 4	Реєстрація за допомогою Google	1. Гість переходить на екран реєстрації 2. Натискає кнопку “Sign up with Google” 3. З’являється вікно авторизації через сервіс Google 4. Гість авторизується в системі	3.а Гість відхилив запит. Виводиться повідомлення про те, користувач відхилив запит на вхід

З наведеної вище інформації у діаграм варіантів використання та таблиці можна детально ознайомитися із функціоналом системи та розподілом ролей між акторами і варіантами використання.

1.2 Проектування програмної системи

1.2.1 Вибір процесу розробки

Управління розробкою програмного забезпечення (англ. Software project management) — це довготривале мистецтво розробки, планування і керування

програмними проектами з проектуванням, особливий вид та управління проектами, в рамках якого відбувається розробка, реалізація, відслідковування і контроль за проектами з розробки програмного забезпечення [4].Системи та способи удосконалення включають використання різних методик для втілення цього підходу.

Для втілення цього починання було обрано еволюційну модель розвитку. На відміну від стійкої моделі, в моделі розвитку потреби встановлюються здебільшого попередньо і визначаються в кожній наступній ітерації.

Використання еволюційної моделі включає дослідження області, щоб обмірковувати потреби свого клієнта та знижувати ймовірність використання цієї моделі для реструктуризації. Ця модель використовується для вдосконалення базових та непрофільних рамок, де основною умовою є виконання ринкових потужностей. Незважаючи на те, необхідність не може бути вирішена оперативно та повністю. Відповідно, просуванню реалізує ітераційне вдосконалення шляхом її трансформаційного видозмінення з набуттям певної варіації рамкової моделі, яка перевіряє виконання потреб. Зрештою, така процедура є природно ітеративною, з повторними етапами просування, починаючи від змінених потреб і закінчуючи отриманням готового товару. У певному сенсі подібну модель можна віднести до звивистої моделі [5].

Просування цієї моделі є трансформативною моделлю прототипування протягом усього життєвого циклу вдосконалення програмування (рис. 1.2). У письмовій формі часто згадується як про швидку розробку додатків RAD [6].

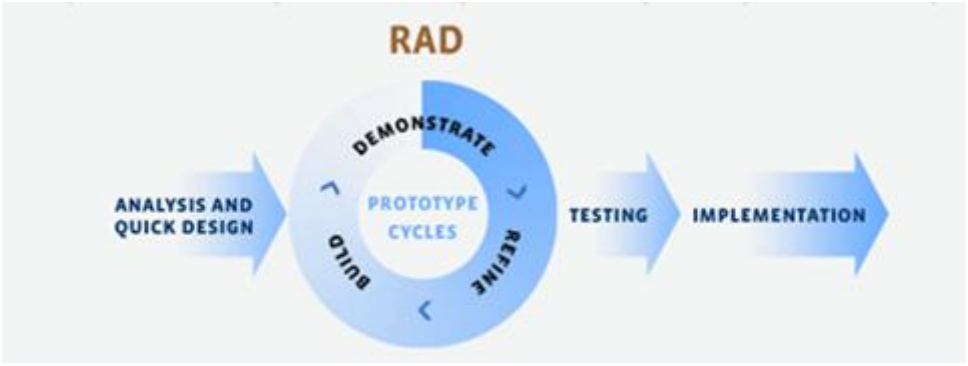


Рисунок 1.2 – Модель та розробки ПЗ у швидкому режимі

Ця модель має два важливих акценти просування розробки програми, структуру рамки та використання. Підтверджено, що він виконує всі практичні та корисні потреби з якими часто інші методики не мають змоги так оперативно і гнучко впоратися. Фундаментальною ідеєю цієї моделі є відображення окремих елементів та її стійке удосконалення з розвитком самої архітектурною відповідності для задоволення всіх визначень функціональним і не функціональним вимогам [7].

Принципові моменти еволюційної моделі ітерацій:

Mode поміркованість випадковості - раннє розпізнавання суперечок між потребами підприємства, моделями та використанням на першочергових завданнях; динамічні передумови організації та виконавців.влаштування потужної критики з боку групи зобов'язань до покупця, виготовлення товару, який справді відповідає його вимогам.

Швидкий випуск товару з мінімальною вартістю (MVP) та потужності для відправки товару та початку роботи багато раніше.

Принципові відмінності ітеративної моделі вдосконалення:

1. Issues Проблеми з архітектурою та накладні витрати. Під час роботи із передумовами та без створення всезагальної домовленості дизайн

програми може постраждати, і можуть знадобитися додаткові активи, щоб перенести його в готовий стан.

2. Немає фіксованого плану витрат і розподілу часу, і потрібне міцне включення клієнта в процес - для певних клієнтів це незадовільні умови співпраці з інженером, каскадна модель є більш кваліфікованою для них [8].

3. Загалом, ця модель конфігурації рамок продемонструвала, що вона є правильною для даної ситуації використання, виходячи з того, що в умовах, що зобов'язані активу, цілком доцільно виділяти час і грошові кошти, що розробляються. Так само ця методологія є дуже простою і корисною. Основні напрямки цієї стратегії включають адаптивність, простоту підтримки та зміни. Деяка частина техніки отримана з основних принципів інших відомих процедур просування (ітеративні та звивисті моделі). Ця модель стосується рамок, де

4. найбільш значущими є основні моменти, які слід негайно виставити для CASE-методів.

Обрана парадигма програмування - це методологія, заснована на статті (з огляду на детально розроблений дизайн) розвитку, що використовує особливості, які дозволяє адаптувати спосіб вдосконалення та використання цієї технології, будуючи різні сегменти для різних моделей. Цей світогляд дає змогу говорити зі структурою програми як про багато предметів, які, таким чином, об'єднуються в певний модуль. Тобто, створивши один клас чи модуль, ми можемо повторно використовувати його у плані та використанні іншої моделі чи рамки, як і при впровадженні результатів у різних випадках [9].

Вибраний підхід до вдосконалення дає змогу зменшити обсяг вихідного коду програми, зробити їх поступово простими та прискорити склад різних видів використання та тестів для них. Це також обмежує витрати на утримання та виконання програми.

Інженерія моделі передбачуваного фреймворку порівнює зі стилем програми клієнт-сервер. Ідея сервера клієнтів ототожнюється з взаємними ПК (серверами), які працюють із загальними активами та роблять ці активи доступними як підтримка своїх клієнтів. Системи ПК, засновані на ідеї "сервер клієнтів", дозволяють: реалізувати корисне управління активами ПК; розповсюджувати інформацію щодо її діяльності та процедури їх підготовки між різними робочими станціями та сервером.

В цілому такий підхід до розробки системи показує себе з сильної сторони і часто використовується в нетривіальних умовах , коли є обмеження на часовий та фінансовий ресурс. Такий план дає додаткові можливості в зручності і легкості розуміння та взаємозв’язку між учасниками робочого процесу. Також з таким видом реалізації можна досягнути легкої підтримки системи і внесення змін , що дозволить уникнути багатьох складнощів на подальших стадіях розробки і вдосконалення проекту , коли робота буде здійснюватися з обширними ітераціями та системами що складно структуризуються і поміщуються в базові концепції представлення. Така модель діяльності широко застосовується у системах з акцентом на функціональних особливостях, і які потребують швидкого наочного відображення в CASE-засобах.

Вибрана нами парадигма програмування є об’єктним підходом і своєю основою лягає на об’єктно-орієнтованій архітектур реалізації з втіленням модулності, що дозволяє широко розглядати системні і функціональні зміни при побудові різних складних частин моделі об’єднаних у певний модуль.[10]

Обраний підхід дозволяє значно пришвидшити розробку без додаткових текстових заглиблень і з максимально чітким бачення загальної концепції. При цьому важливу роль відіграє економія ресурсу та забезпечення гнучкості і свободи вибору під час реалізації складних частин.

1.2.2 Побудова UML діаграми діяльності програмної системи

Відображаючи поведінку досліджуваної системи, важливо не тільки описати шлях до зміни її станів, але крім того, щоб детально викласти основні моменти алгоритмічних та інтелектуальних адміністрацій, що використовуються в цих рамках у графічному представленні. Як правило, з цієї причини використовують блок-схеми розрахунків. Кожен такий план або блок-схема підкреслюють достовірність перевірки інших істотних або базових обмінів, які в цілому повинні мати ідеальні результати.

Передбачувані схематичні дії використовувались для виявлення завдань на мові UML. Система повинна бути розроблена спочатку, ми не можемо просто сісти і написати якийсь код. Під час його впровадження необхідно працювати в команді програмістів, бо одного точно не вистачає. Колектив повинен мати дуже гарне спілкування. Необхідно також реагувати на зміни у дорученні клієнта. Оцінити вартість системи на ранніх стадіях розвитку може бути дуже важко. UML - це один із інструментів, який допомагає нам вирішувати ці проблеми. Це допоможе у фазі аналізу, коли ми спілкуватимемось із клієнтом та дізнаємось, що ми насправді запрограмуємо. І це також допоможе нам на етапі проектування, коли ми розглянемо те, як ми програмуємо.

Таким чином, діаграми зміни станів діяльності можна вважати особливим випадком використання діаграм, Саме за допомогою їх можна використати засоби мови і особливості процедурного та синхронного методу управління діяльності внутрішніх процесів. Модель UML надає для цього необхідні терміни та семантичні засоби.[11]

У формі вираження UML діяльність (activity) формує таким чином певного роду сукупність , для проведення обчислень , що виконуються системою. При цьому слід зважати , що деякі обчислення та дії можуть і приводять до наслідків (actions). На діаграмах виражена логіка переходу від однієї суті діяльності до наступної, втім акцентується значна увага на результаті

обчислень та дій. Однак слід розуміти, що результат від роботи системи може призводити до змін у системі та поверненню значень.

Специфічний стан дії (action state) є спеціальним і особливим випадком використання стану з певними вхідними даними , і як мінімум, одним переходом. Цей перехід робить неявне припущення того що дія вже завершилася. Стан дії немає внутрішніх переходів, оскільки вони можуть бути тільки елементарними. Загальне використання стану дії розуміється як виконання певного конкретного потоку, процедури дії або керування.

Стан дії можна зобразити графічно за допомогою фігури, що нагадує прямокутник, у якого замість бічних сторін присутні дуги. Ця фігура містить всередині вираз, що повинен бути унікальним в рамках однієї діаграми.[12]

Перехід. При проектуванні діаграми діяльності варто користуватися лише тими переходами, що спрацьовують одночасно після завершення діяльності або опрацюванням певної дії. Цей перехід зміщує стан діяльності своєчасно , одразу після завершення дії. Такий перехід позначають суцільною лінією зі стрілками.

Графічно переходи між станами діяльності можна позначити у вигляді ромба. У цей ромб поміщається лише та стрілка від того стану дії із завершенням виконання якого керування перейде до однієї із гілок програми. Прийнято вхідні гілку сполучати із станом розгалуження верхньої або лівої вершини розгалуження. Для кожної з стрілок має бути передбачений два або більше виходи , до кожного з яких повинна бути вказана відповідна сторожова умова у вигляді булівського виразу.[13]

В більшості ситуацій діаграми діяльності відбуваються із об’єктами , які в той же час можуть ініціювати виконання певних дій у системі, або визначають ті чи інші випадки на противагу яких і створюються окремі реалізації варіанту використання у діаграмі. Дії ініціюють виклики , які можуть генерувати передачі

від одного об’єкта у графі до іншого. Оскільки в такому випадку об’єкти відіграють роль в процесі життєдіяльності , слід їх вказувати явними на діаграмі станів діяльності.

Для зображення графічних об’єктів застосовується прямокутник класу, з тієї новизною , що ім’я об’єкта слід підкреслити однією рискою. Зазвичай після імені і прямих дужках вказується характеристика стану об’єкта. З такими прямокутними формами об’єднуються діаграми класу об’єднання пунктирною рискою зі стрілкою. Така залежність показує стан об’єкта після виконання попередньої частини.[14]

Отже , можна говорити про те , що діаграми діяльності є демонстраційними частинами того , які можливості для використання програми є у користувача.

У системі можна виділити наступні форми взаємодії з користувачем :

1) авторизація
2) реєстрація
3) створення замовлення
4) додавання нового транспорту
5) видалення машини
6) створення нового маршруту
7) аналіз перевезень
8) редагування замовлення
9) редагування і аналіз витрат
10) створення і редагування карти маршрутів
11) зміна статусу водія

Дія “авторизація” та “реєстрація” дає можливість новому користувачеві взаємодіяти з системою і стати Користувачем, відповідно до свого статусу користувач має різні можливості доступу в системі.

“Створення замовлення” це можливість за допомогою якої зареєстрований користувач може створювати нові транспортні замовлення і вводити свої корективи у маршрутну гілку перевезень.

“Додавання нового транспорту” це можливість за допомогою якої зареєстрований користувач може додавати нові грузові автомобілі до списку уже готових.

“Видалення машини” дія за допомогою якої користувач може видаляти транспортні засоби з категорії транспорту.

“Створення нового маршруту” це дія за допомогою якої зареєстрований користувач може реалізувати створення власних дорожніх карт за допомогою системи навігації та мапи, ця можливість є невід’ємною частиною автоматизації транспортних перевезень і складає левову частку робочого процесу оператора.

“Аналіз перевезень” дозволяє Користувачеві створювати окремий файл з переліком перевізників та класифікувати їх за ефективністю і виконати розподіл і розрахунок перевізників.

“Редагування замовлення” функція для зареєстрованого користувача з допомогою якою можливо вносити зміни у доступні замовлення.

“Редагування і аналіз витрат” користувацька функція з допомогою якої можливо проводити повний аналіз витрат і вносити корективи у роботу фінансової сфери підприємства.

“Створення і редагування карт маршрутів” функція для запиту актора Користувач з переліком основних картографічних збережень і подальша правка їх в режимі реального часу.

“Зміна статусу водія” функція зареєстрованого користувача що дає можливість міняти статуси водіїв під час руху і таким чином керувати їх статусами в системі.

З детальним планом проектованої системи можна ознайомитись на рис.1.3

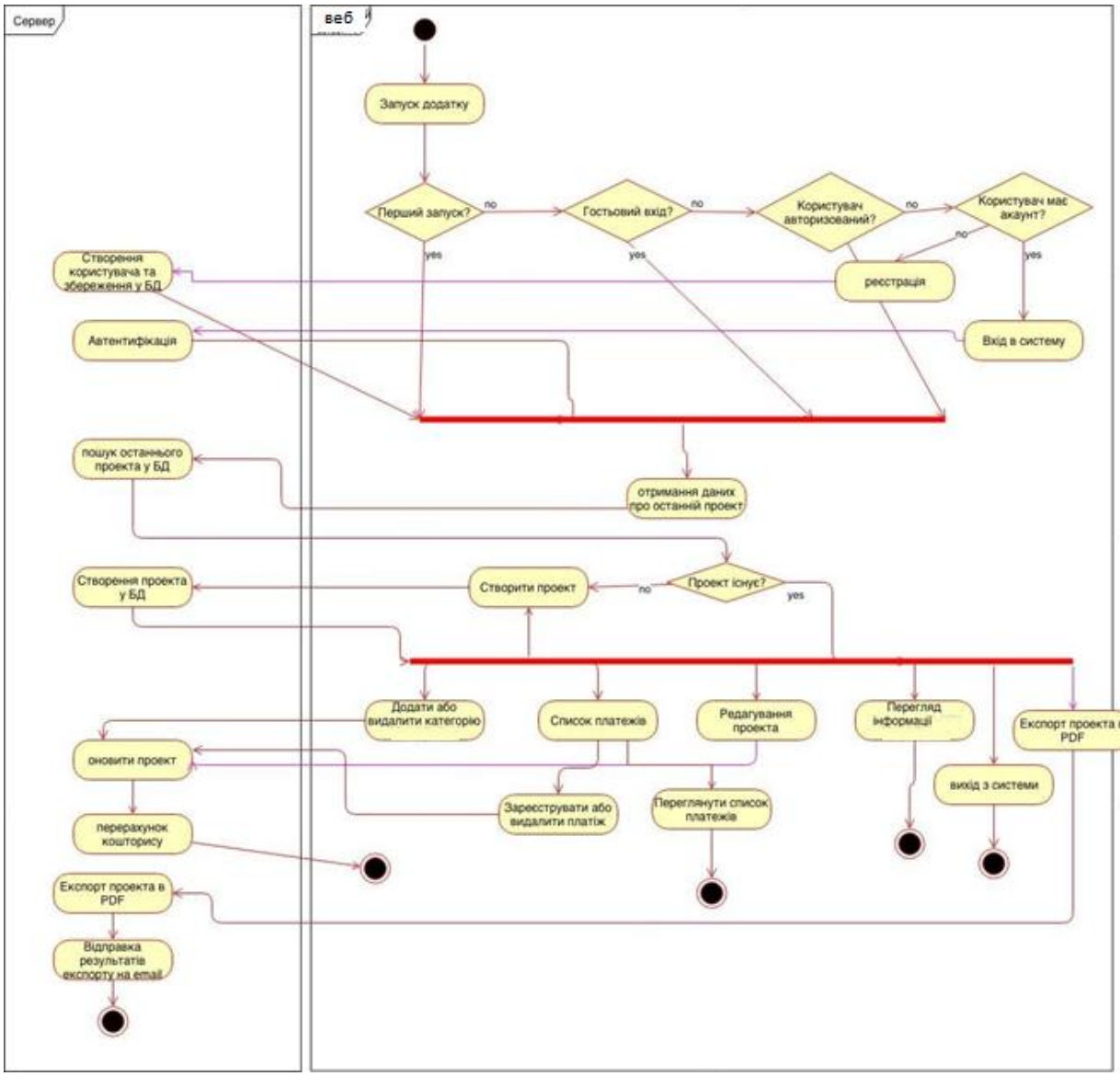


Рисунок 1.3 - план проектованої системи

Цей граф показує наскільки розгалуженими є можливі дії клієнта в додатку. Початок роботи є момент , коли користувач проводить спробу увійти до веб-додатку. Наступним етапом є визначення або ідентифікація користувача , на можливу реєстрацію в системі. Якщо ми маємо справу з незареєстрованим користувачем, тоді його буде перенаправлено на сторінку з можливістю реєстрації, якщо користувач є зареєстрованим , тоді проводиться авторизація і присвоєння статусу Користувач з повним розширенням усіх прав для користування ресурсом. Допоки користувач залишається незареєстрований в системі для нього буде створено гостьовий профіль , під яким він матиме можливість ознайомитись з головною сторінкою додатка та провести повну реєстрацію аккаунта.

В кінцевому результаті користувач буде мати можливість отримати дані з останнього проекту. Під час цього здійснюється запит до серваера, який надає можливість повернення даних . Якщо таких даних немає , тоді користувач має змогу внести нові дані для поїздки , після чого отримати нову базу даних з маршрутами і водіями.

1.2.3 Моделювання архітектури системи

Як було встановлено після планування технічної бази, проект за своїми характеристиками найдоцільніше створити на дизайні клієнт-сервер з інформаційною підтримкою.

Конструкція для побудови такого додатка передбачає 3 рівня взаємодії, споживач підключений до сервера , який послідовно має доступ до баз даних.

Така конструкція отримала гарний відгук та поширення із розвитком мережі

Клієнт - це елемент інтерфейсу (зазвичай графічний), який представляє первинний рівень, це дійсно додаток для кінцевих користувачів. У випадку

розробленої системи ця програма є додатком для операційних систем Windows. первинний рівень не повинен мати прямих посилань на інформацію (для вимог безпеки), не повинен бути завантажений базовою діловою логікою (для потреб кількісних показників) та підтримувати стан програми (для вимог надійності). первинний рівень буде прийнятий і, як правило, найкраща для логіки бізнесу: інтерфейс авторизації, алгоритми секретного запису, перевірка вхідних значень, прийнятність та відповідність формату, прості операції (сортування, групування, підрахунок) із знаннями, які вже завантажені на термінал

Сервер додатків знаходиться на другому рівні. Другий рівень зосереджений на більшості бізнес-логіки. Зовні фрагменти, експортовані до терміналів (див. вище), додатково як утримування процедур та тригерів, розміщених у межах третього рівня.

Інформаційний сервер забезпечує зберігання інформації і переноситься на третій рівень. це часто іноді є типовою відносною або об'єктно-орієнтованою СУБД. Якщо на третьому шарі може бути інформація, яка стоїть поряд із процедурами, тригерами та схемою, яка описує пристрій з точки зору відносної моделі, то другий шар будується як програмний системний інтерфейс, який пов'язує частини клієнта з логікою пристрою інформації.

У прямій конфігурації фізично сервер додатків часто підключається до інформаційного сервера на одному ПК, до якого один або додаткові термінали підключені по мережі.

У "правильній" (з точки зору безпеки, відповідальності, масштабування) конфігурації інформаційний сервер розміщується на завзятому ПК (або кластері), до якого один або додаткові сервери додатків підключені до мережі, до якої, у свою чергу, До мережі підключені термінали[15].

Розгортання програмного забезпечення – це сукупність дій, що роблять програмну систему готовою до використання [16]. Даний процес є частиною життєвого циклу програмного забезпечення.

Для підсумків дизайну системи на високому рівні ми схильні використовувати схему підготовки UML.

У межах специфікації UML підготовчі діаграми моделюють фізичну конструкцію системи. діаграми підготовки показують зв'язок між пакетною та апаратною частинами системи і, отже, фізичним розподілом процесу.

Діаграми розгортання, які, як правило, вбудовані в розділ реалізації, показують тіло вузлів у надзвичайно розподіленій системі, артефакти утримуються на кожному вузлі, а отже, частини та різні елементи, використані артефактами. Вузли представляють апаратні пристрої, такі як комп'ютери, датчики та принтери, так само як різні пристрої, що підтримують атмосферу виконання системи. способи комунікації та підготовки відносин моделюють зв'язки всередині системи [17].

Для правильного функціонування та виконання всіх вимог створені наступні системні вузли:

Amazon net Services Elastic Calculate Cloud - сервіс, який розміщує сервер та інформацію. Сервер працює на ОС Ubuntu Sixteen.6 і доступ до нього здійснюється за допомогою протоколу SSH. Сервер розгортає лоукостера з контейнерами, які можуть запускати інформаційні процеси MongoDB, сервер NodeJS, Nginx та Let's cypher SSL.

Docker може бути набором інструментів для управління ізольованими контейнерами операційної системи UNIX. longshoreman розширює набір інструментів LXC за допомогою додаткового розширеного API, який дозволяє керувати контейнерами на рівні ізоляції окремих процесів.

Серверна частина Meteor додатку є надбудовою Node.js сервера. Шифровий двигун MongoDB - сервер для баз даних MongoDB. найбільше системної інформації працює на такому сервері. Інформація використовується додатком NodeJS, який працює на самому сервері системи.

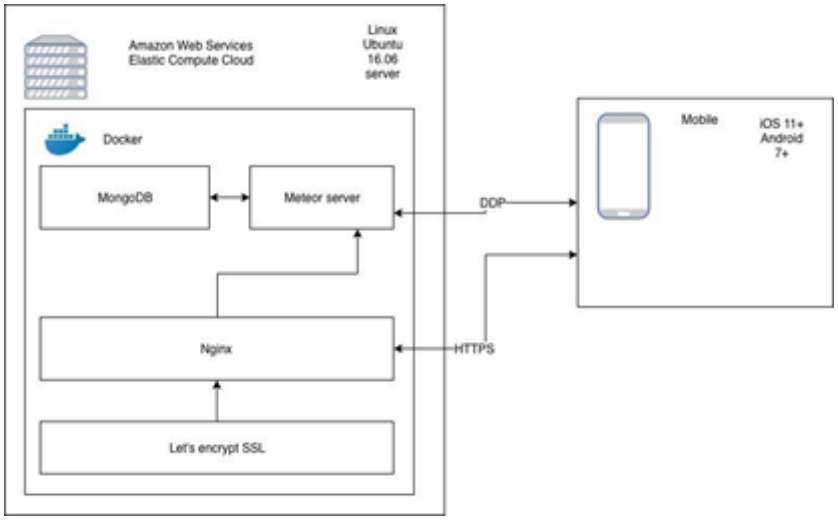
Клієнтський пристрій. Щоб діяти з системою, користувач повинен використовувати додаток ступеня асоційованого ступеня, який

встановлюється на інструмент, який працює на програмному забезпеченні iOS або автомат. Система підключається за допомогою протоколу зв'язку, покупець підключається до програми Meteor, що працює на першому сервері, а у відповідь покупець встановлює приналежність до DDP, тобто надбудову для додаткового ступеня для Websocket.

Після встановлення належності і, отже, ініціалізований розділ shopper, сервер спілкується через DDP.

Діаграму розгортання для розроблювальної системи зображено на рисунку 1.4.

рисунок 1.4 - діаграма розгортання системи



1.3 Конструювання системи

1.3.1 Вибір мови та середовища розробки

Як ви розумієте, розроблена система складається з багатьох частин, які рухаються. щоб полегшити події та підтримати додатки, коучинг та взаємодію розробників різних модулів, було налаштовано використовувати мову програмування JavaScript на найменших рівнях. Незважаючи на застаріле бачення, JavaScript не тільки може виконувати код у веб-переглядачі задля інтерактивності сайтів, однак додатково є мовою для впровадження сервера, насамперед через оточення Node.js, розроблене в Chrome Engine V8 Chrome.

JavaScript (JS) може бути динамічною, об'єктно-орієнтованою мовою програмування. Реалізація ECMAScript звичайна. найчастіше він використовується як частина браузера, яка дозволяє клієнтовому коду (виконуваному на пристрої кінцевого користувача) рухатися з користувачем, керувати браузером, асинхронно спілкуватися з сервером, змінювати структуру та вигляд чиста сторінка. JavaScript додатково використовується для серверного програмування (подібно до мов програмування Java та C #), розробки ігор, швидких та мобільних додатків, сценаріїв програм (таких як програми Adobe inventive Suite), в документах PDF тощо.

JavaScript оцінюється як зображення (набір об'єктно-орієнтованих), сценарій мови програмування з динамічною машинописом. Крім прототипування, JavaScript додатково частково підтримує альтернативні парадигми програмування (імперативні та частково функціональні) та декілька відповідних властивостей предмета, такі як динамічне та слабке введення тексту, автоматичне управління пам’яттю, успадкування зображень, функціонують як об’єкти.

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme [18].

JavaScript містить набір вбудованих операцій, які, власне кажучи, по суті не є функціями чи способами, так само, як колекція вбудованих операторів, що керують логікою виконання програм. Ця мова зараз є однією з усіх передових стандартних мов програмування в мережі. однак спочатку кілька кваліфікованих програмістів скептично ставились до мови, аудиторія якої - аматорські програмісти. поява міфічної модифікації справи та повернення уваги кваліфікованої громади до мови. Як результат, декілька практик JavaScript (включаючи тестування та налагодження) розробляються та вдосконалюються, створюються бібліотеки та рамки, а JavaScript розширюється на далекій стороні браузера.

Якщо це стосується успадкування, JavaScript має лише одну конструкцію: об'єкти. кожен об'єкт включає приховане властивість (іменоване [[прототип]]), яке містить відношення до іншого об'єкта, іменованого як модель. Ця модель, в свою чергу, має свою модель, і так далі, поки такий об'єкт Associate in Nursing не стане нульовим у вартості моделі. За визначенням, нульова сума не може бути моделлю, і вона є останньою в такому ланцюзі моделей.

Майже всі об’єкти в JavaScript є екземплярами Object, тобто початковою ланкою в ланцюзі вартості моделі.

Хоча це часто вважається одним із усіх слабких місць JavaScript, він насправді є додатково потужнішим, ніж класичний. як приклад, досить поширене застосування для створення класичної моделі, що підтримується модельною моделлю.

Обличчя споживача може бути мобільним додатком, який підтримує операційні системи Windows. в результаті смартфони та мобільні додатки

стали таким чином стандартними, чисті розробники шукають способи створити JavaScript для обробки мобільних додатків. Ця якість є світлодіодною для події багатьох рамок для цієї штучної мови, здатної виконувати власний код на мобільних пристроях. В даний час Кордова і Реакт Нативні - це головне стандартне рішення. Кордова підтримує мобільні платформи, такі як iOS, механічна людина та Windows Phone. З іншого боку, React Native підтримує механічну людину, операційні системи iOS. щоб прийняти рішення, яке рамки використовувати, ви хочете визначити їх переваги та недоліки.

Кордова з відкритим постачанням JavaScript для розробки гібридних додатків. Програми, що підтримують цю технологію, використовують WebView для розведення інтерфейсу програми. Кордова сумісно дозволяє розробникам використовувати новітні чисті технології, такі як HTML5, JS. Найбільш перевагою є те, що практично абсолютна однаковість коду для будь-якого програмного пакету. Недоліком є те, що ці додатки Кордова - це сайти, які вводять рідний інтерфейс. Як результат, є велике навантаження на обладнання, яке може знизити продуктивність додатків, і це не повинно бути просто.

React Native покладається на React (рамки JavaScript) для розробки інтерфейсів прикладного використання JavaScript. В основному, React Native не використовує WebView для відтворення інтерфейсу програми, як у Кордові. Весь інтерфейс React Native відображає рідні частини, які обробляють окремі потоки та обробляються з використанням графічного процесора замість апаратного забезпечення. Інтерпретатор JavaScript вбудований для бізнес-логіки, обробки та управління інтерфейсом, тоді як візуалізація займає рідну половину. Недоліком є те, що в деяких випадках ви хочете травмувати модулі, написані на Java та Objective-C. Перевага полягає в тому, що React Native може бути швидкою основою на React, яка добре зарекомендувала себе в спільноті розробників.

React дозволяє розробникам робити гігантські чисті програми, які використовують знання, які змінюються з часом, не перезавантажуючи сторінку. Її мета - бути швидким, простим, масштабованим. Реагує виключно обробляє комп'ютерну програму в додатках. це часто в залежності від видів у моделі Model-Controller (MVC) і може використовуватися разом із альтернативними бібліотеками JavaScript або в гігантських MVC-рамках, таких як AngularJS. Він може використовуватися з доповненнями на основі React, щоб вимагати догляду за елементами, а не інтерфейсом побудови інтернету для додатків.

В даний час React працює в Академії Хана, Netflix, Yahoo, Airbnb, Sony, Atlassian та ін.

Для системи нижче розробки необхідне просте використання та продуктивність. Тому технологія React Native була обрана для задоволення потреб кожного користувача, а також розробника.

Для основи сервера обрано фреймворк JavaScript під назвою Meteor.

Meteor – це платформа, побудована на основі Node.js для створення веб-додатків у режимі реального часу. Це прошарок між вашою базою даних та інтерфейсом програм, що стежить за їхньою синхронізацією.

Цю платформу було обрано через низку причин:

- програми написані на платформі Meteor є по-замовчуванню додатками реального часу;
- клієнтська і серверна частина розробляється за допомогою однієї мови - Javascript;
- велика швидкість розробки через наявність смарт-пакетів;
- нова технологія, яка стрімко розвивається.

В результаті всього цього, абстрагуючись від багатьох звичних неприємностей і підводних каменів розробки веб-додатків, ми маємо дуже потужну і водночас просту у використанні платформу.

Вибір на це фреймворк впав через факт, що дозволяє швидко розвивати Інтернет-програми, в результаті чого в майбутньому планується застосовувати цю техніку не лише для операційних систем iOS та автоматики, але додатково для браузера. . Метеор підходить для стартапів, і завдяки його вимірюваності він може бути з успіхом використаний для подій величезних масштабів. Також його перевагою є те, що серверний аспект фреймворку побудований на Node.js, тобто ви зможете використовувати сторонні пакети із загальнодоступного сховища NPM.

NPM (Node Package Manager) може бути менеджером пакунків для штучної мови JavaScript. Для виконання Node.js це менеджер пакунків за замовчуванням. Включає споживача команд, додатково згаданого як npm, аналогічно як веб-загальнодоступну та особисту інформацію, що називається реєстром npm. Реєстр доступний для споживачів, тому наявні пакети можуть переглядатися та шукатись через веб-сайт npm. Менеджером пакунків та реєстром керує npm, Inc. [17].

Microsoft Visual Studio Code (VS Cod), порівняно новий текстовий редактор, вільний від Microsoft, був обраний через середовище розробки. Він, як і Atom, покладається на двигун Cr, однак має свої відмінні варіанти, як IntelliSense, що випускається поза коробкою. [18]

Протягом останніх декількох років визнання коду Visual Studio стало дорослим на ринку відкритих продуктів IDE. Код VS був офіційно безкоштовним у 2015 році і наразі ним займається близько 35% усіх розробників, нарівні з опитуванням Stack Overflow 2018. [19]

Код VS підтримує JavaScript, Babel, CoffeeScript, матерії та багато іншого. Код VS забезпечує автоматичне завершення, аналіз коду на ходу, навігацію по коду, рефакторинг, налагодження та інтеграцію з системами управління

версіями. Дуже важливою перевагою інтегрованого середовища розробки коду VS є те, що робота надходить (включаючи рефакторинг коду JavaScript, що міститься у різноманітних файлах та папках проектів, аналогічно вбудованому HTML). Підтримується кілька вкладень (коли HTML-скрипт вбудований в HTML-документ, який вбудовує інший HTML-код у цей Javascript), такі конструкції підтримують правильний рефакторинг [20]

1.3.2 Вибір СКБД та опис її фізичної моделі

Для того, щоб система могла управляти бажаним складом знань, який буде містити всі знання, потрібно структурований підхід і розподіл. Документально-орієнтована інформація про грошову одиницю буде використовуватися, оскільки це тема знань, схожа на JSON.

Інформація може бути асортиментом бази даних, організованою відповідно до ідеї, яка описує характеристики цих даних, а отже, зв’язки між їх елементами; цей набір підтримує як мінімум одну програму. Загалом, інформація містить схеми, таблиці, представлення даних, зберігає процедури та різні об'єкти. інформація всередині інформації організована відповідно до моделі організації даних. Таким чином, сучасна інформація, крім самої інформації, містить їх опис і повинна містити припущення,як їх обробляти [21].

Документ не є відносним, а документоорієнтована інформація, внаслідок чого він дуже підходить для зберігання стратифікованих с знань і більш універсальний для запитів. Інформація, орієнтована на документи, може бути базою даних, під час якої інформація надається для збереження колекцій у файлах. вони будуть продумані далеко не аналог онлайн-таблиць баз даних, однак колекції можуть містити різні типи даних. Збір створюється з документів, що називаються записами, а записи містять стовпці, що називаються полями. хоча документи на асортимент також є дискреційними, для кращого поділу документи індексувати з однаковою структурою.

При розробці інформаційно-орієнтованої інформації важливою метою є формування передбачуваної схеми. Інформаційна схема полягає в тому, що структура системи бази даних, представлена дуже формальною мовою, підтримується системою спрямування (СУБД) і посилається на організацію інформації, щоб сформувати бази даних з розподілом. Формально інформаційна схема може представляти собою набір правил, що називаються обмеженнями цілісності.

Кінцевою метою створення схеми є зменшення можливої невідповідності знань, що зберігаються в межах інформації.

Система розробки використовує інформацію MongoDB. MongoDB - це асоційована система відкритого доступу, заснована на документах (СУБД), яка не потребує опису схеми таблиці. MongoDB займає чіткий сегмент між швидкими та масштабованими системами ключа / вартості та відносними СУБД, які є цілеспрямованими та простими.

Код MongoDB написаний на C ++ і поширюється під ліцензією AGPLv3.

MongoDB підтримує зберігання документів у форматі JSON, включає досить універсальну мову для запитів, виробляє індекси для численних атрибутів зберігання, ефективно забезпечує зберігання величезних бінарних об'єктів, підтримує робочі операції для зміни та додавання знань до інформації, додасть відповідність Парадигма карт / масштаб назад, підтримує реплікацію та стійку до відмов конфігурацію.

У MongoDB є цілісні інструменти для забезпечення шардингу (розподіл набору даних на серверах, що підтримуються певним ключем), поєднуючи те, що реплікація знань буде розроблена в горизонтальному масштабуванні кластеру зберігання даних, що не має однієї мети відмови (відмова будь-якого вузла не впливає на функціонування бази даних), підтримується автоматична відмова та передача вантажу з невдалого вузла. збільшення кластера або зміна одного сервера в кластер виконується без кінця інформацією, просто додаючи нові машини [22].

Розробляючи, автори виходили з вимоги до інформаційної спеціалізації, яка створювала їх можливість рухатися в сторону від принципу єдиного розміру. Зводячи до мінімуму лінгвістику процесу розгляду, ви отримали гнучкість для вирішення різноманітних проблем продуктивності, а горизонтальне масштабування змінюється на легше. Використовувана модель зберігання документів JSON / BSON - простіша в кодуванні, простіша в управлінні (в тому числі за допомогою використання передбачуваного "стилю без схем"), і тому внутрішня агломерація відповідних знань забезпечує додаткові підвищення продуктивності.

Нереляційний підхід досить зручний для формування баз даних, під час яких горизонтальне масштабування підказує підготовку на декількох машинах. гнучкість для забезпечення найбільш ефективних показників повинна існувати паралельно із підтримкою багато практичності, ніж використання пар ключових значень (у чистому вигляді).

Технологія баз даних повинна працювати всюди - від серверів користувачів та віртуальної машини до хмарних технологій [23]. відповідно до розробників, MongoDB повинен був заповнити проміжок між прямими сховищами ключових значень (швидкими та просто масштабованими) та величезними RSDBD (діаграмами та потужними запитами).

Основні можливості MongoDB:

- Документо-орієнтоване сховище (проста та потужна JSON -подібна схема даних)
- Досить гнучка мова для формування запитів
- Динамічні запити
- Повна підтримка індексів
- Профілювання запитів

- Швидкі оновлення «на місці»
- Ефективне зберігання двійкових даних великих обсягів, наприклад, фото та відео
- Журналювання операцій, що модифікують дані в БД
- Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг
- Може працювати відповідно до парадигми MapReduce

Система управління базами даних управляє наборами документів, подібних JSON, що зберігаються у двійковому типі у форматі BSON. MongoDB зберігає та витягує файли через дзвінки GridFS.

Як і різні СУБД, орієнтовані на документи (CouchDB тощо), MongoDB не є відносною СУБД.

Є розроблена і якісна документація, велика кількість прикладів і драйверів для поширених мов Java, C ++, C #, PHP, Python, Perl, Ruby.

Зразу випущено чітке повідомлення про те, що розряд MongoDB one.0 підготовлений для використання у виробництві як один хост, так і в з'єднанні master / slave. Код цього розвороту є стабільним і перевірений у промисловій експлуатації протягом одного року.

MongoDB повинен бути розгорнутий на щонайменше 2 серверах віктимізації Master / Slave реплікації, якщо це можливо. Це гарантує, що актуальна інформація з’являється на ринку один раз у всіх СУБД.

MongoDB є досить молодим продуктом, тому він стикається з помилками, новими можливостями тощо. Висока швидкість розвитку характерна (проект написаний не тільки волонтерами, але спільно організацією штатних людей) [24].

MongoDB є невід'ємною частиною метеорної рамки, внаслідок чого реактивність є однією з усіх найбільш варіантів цієї структури. насправді в

Метеорі є рішення для віктимізації різних баз даних, однак вони не практикуються в результаті, вони цього не хочуть, в результаті цього реактивність втрачається, а сам MongoDB може бути широко розповсюджена як ефективна інформація, яка добре підходить до розгадування проблем, властивих системам, як та, яка зараз розробляється.

Інформація MongoDB була обрана для системи подій. це інформація, орієнтована на документи, де б інформація зберігалася як документи у колекціях. Документоорієнтована модель MongoDB працює швидше і має більш високий масштаб порівняно з реляційними базами даних. використання бінарного формату зберігання інформації пришвидшить процеси, проте потребує додаткового місця.

1.3.3 Реалізація основних класів та методів

Після проведеного аналізу ми можемо виявити основні сутності предметної області, які відобразимо відповідним чином у базі даних. У базі даних будуть присутні наступні колекції: flottes, trajets

Детально розглянемо основні сутності системи. Оскільки, основним призначенням системи проектування транспортних перевезень, то потрібно виділити сутність Flottes, яка представляє елемент колекції flottes. Дана сутність містить у собі загальну інформацію, тобто id користувача, номер вантажного транспорту та тип локалізації та дату створення.

Основні методи даної колекції є :

- задання кінцевої дати поїздки за допомогою метода dateFin
- задання проміжної локації за допомогою метода destinationIntermediaire
- задання кінцевої точки маршруту за допомогою метода destinationFinale
- задання id вантажівки стандартними методами

Наступним елементом сутностей є Trajet в якому містяться дані про тривалість маршруту та User, в якому містяться дані про зареєстрованих користувачів.

За допомогою елементів цих колекцій можна сформувати систему з повноцінним набором даних в якій кожен метод зможе відокремлено виконувати свої функції і стабільно доповнювати розвиток виконання алгоритму автоматизації.

Основні види обчислень відбуваються на сервері і тому важливою є взаємодія між інтерфейсом користувача та алгоритмами підключеними до API google-maps в якому будуть відбуватися основні побудови маршрутів в нашому додатку .

Лістинг 1.1 - Побудова маршрутів

```
const distance = require('google-distance');
const Future = require('fibers/future');

Meteor.methods({
  calculDureeTrajet(dep, arr) {
    const future = new Future();
    distance.get(
      {
        origin: dep,
        destination: arr
      },
      (err, data) => {
        if (err) {
          future.return(console.log(err));
        }
        future.return(data.durationValue);
      }
```

```
  );
  return future.wait();
},
trajet(_id, depart, destinationIntermediaire, destinationFinale) {
  if (depart === destinationIntermediaire) {
    console.log('pas de destination intermédiaire (trajet direct)');
    const delai = Meteor.call('calculDureeTrajet', depart, destinationFinale);
    console.log('De ' + depart + ' à ' + destinationFinale + ' il y a ' +
      delai + ' secondes de trajet.');
    Meteor.call('insererTrajet', _id, depart, '', delai, destinationFinale);
  } else {
    console.log('il y a une étape intermediaire au trajet: ' +
      destinationIntermediaire);

    const delai1 = Meteor.call('calculDureeTrajet', depart,
      destinationIntermediaire);
    console.log('delai de: ' + depart + ' à ' + destinationIntermediaire +
      ': ' + delai1 + ' secondes');

    const delai2 = Meteor.call('calculDureeTrajet',
      destinationIntermediaire, destinationFinale);
    console.log('delai de: ' + destinationIntermediaire + ' à ' +
      destinationFinale + ': ' + delai2 + ' secondes');

    const delaiTotal = (delai1 + delai2);
    console.log('Temps total: ' + delaiTotal + ' secondes.');

    Meteor.call('insererTrajet', _id, depart, destinationIntermediaire,
      delaiTotal, destinationFinale);
```

```
    }
  },
  demandeTrajet(_id, depart, arrivee) {
    const localisation = Meteor.call('localisationCamion', _id);
    console.log('localisation initiale : ' + localisation);
    if (Meteor.call('disponibiliteCamion', _id)) {
      if (depart === arrivee) {
        console.log('depart et arrivee identiques');
      } else {
        console.log('demande de trajet de ' + depart + ' à ' + arrivee);
        Meteor.call('desactiverCamion', _id);

        Meteor.call('trajet', _id, localisation, depart, arrivee);
      }
    } else {
      console.log('flotte non disponible');
    }
  }
});
```

Система передбачає можливість авторизації. Для збереження інформації про користувача існує колекція users. Однією з особливостей є те, що дозволено мультисервісна авторизація за допомогою облікового запису в соціальній мережі Facebook, Google або за допомогою пари «електронна адреса – пароль». Основним ідентифікатором в даному випадку буде електронна адреса, адже вона є спільною для вищеперелічених сервісів авторизації. Таким чином, при вході за допомогою іншого сервісу, обліковий запис користувача буде доповнено новим способом входу. Метадані такі як

accessToken – ключ входу та expiresAt (до якого моменту дійсний ключ входу) будуть зберігатись і об’єкті services у вигляді «ключ-значення», де ключем буде виступати назва сервісу: facebook, google, password.

Також в обліковому записі користувача буде загальна інформація особи, яка буде зберігатись в об’єкті profile. В подальшому ця інформація може застосовуватись для ідентифікації особи при втраті доступу до облікового запису. До такої інформації відноситься наступне:

- name – ім’я та прізвище;
- email – електронна адреса;
- language

Ще одна особливість системи полягає в тому, що централізоване управління локалізацією інтерфейсу. Тобто доцільно записувати файли локалізації, динамічно їх на сервер, тоді споживча програма може передати оновлені файли після підключення до мережі. Це може створити просто виправлення помилок локалізації, не публікуючи оновлення для споживача - клієнт передасть все у фоновому режимі. щоб створити цей виконаний, був створений асортимент i18n (інтернаціоналізація). Поля під час цього асортименту: мова

- двобуквенний код країни з кроком з ISO 3166-1 alpha-2 та полем перекладу - документ локалізації JSON.

Виділяючи потреби системи, їх структуру та з'єднання, доцільно стилізувати Асоційована в діаграмі ER сестринської інформації.

Діаграма і зв'язок сутності, або асоційована діаграма з сестринською ER, може бути графічною ілюстрацією декількох сутностей, їх атрибутів та відносин. Давайте більш детально перевіримо основні ідеї діаграм ER.

Сутність може бути категорією об'єктів подібного роду, дані яких слід враховувати в межах моделі. кожна сутність повинна мати дільник в однині.

Атрибут Entity - це може бути атрибут, який називається, тобто декілька властивостей Associate in Nursing. Ім'я атрибута має бути іменником однини.

Секрет сутності - це додатковий набір атрибутів, значення яких спільно відмінні для кожного екземпляра сутності. Запасним є те, що видалення будь-якого атрибуту з ключа порушує його індивідуалізм. У асоційованої сестринської організації буде багато абсолютно різних ключів.

Комунікація може бути пов'язаною між двома суб'єктами. Одне утворення також пов'язане з іншим утворенням або з самим собою. З'єднання дозволяють одному суб’єкту шукати альтернативні сутності, пов'язані з ним. схематично асоціація розмежована лінією, що з'єднує 2 сутності. кожне посилання має 2 кінці та одне або два імені. Ім’я іноді виражається в Associate in Nursing невизначена форма дієслова: "have", "pripad" тощо. Кожне з назв відноситься до його закінчення спілкування. зазвичай імена не пишуться через їх прохідність.

Кожна зв'язок може мати один з типів зв’язку, зображених на рисунок 1.13.

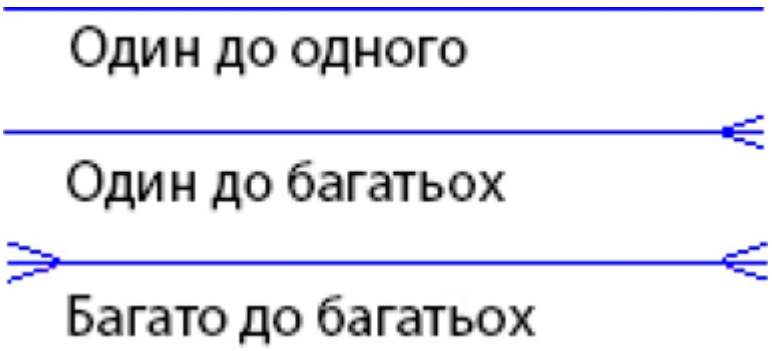


Рисунок 1.5 – типи зв’язку сутностей

Зв'язок типу один-до-одного означає, що один примірник першої сутності (лівої) пов'язаний з одним примірником другий сутності (правої).

Зв'язок один-до-одного найчастіше свідчить про те, що насправді ми маємо всього одну сутність, розділену на дві.

зв'язувальний пристрій для множин означає, що один екземпляр первинної сутності (зліва) пов'язаний з декількома екземплярами другого об'єкта (праворуч). це часто найперше звичайно використовуваний різновид спілкування. Ліва сутність (з "однієї" сторони називається материнською сутністю, правильна сутність (з "однієї" сторони) може бути дочірньою сутністю. Зв'язок "багато-багато" означає, що кожен екземпляр первинної сутності також може бути пов'язаний з декількома екземплярами другого об'єкта, і кожен екземпляр другого об'єкта також може бути пов'язаний з декількома екземплярами первинної сутності. Різноманітність комунікацій може бути тимчасовим типом спілкування, який допустимий на ранніх етапах розробки моделі. У майбутньому такий тип комунікації повинен бути замінений двома з'єднаннями один-на-багато, зробивши Associate in Nursing intermediate.

1.4 Використання програмної системи

1.4.1 Опис типових схем використання

При першому запуску користувачу буде запропоновано обрати можливість входу (див. рис. 1.16).

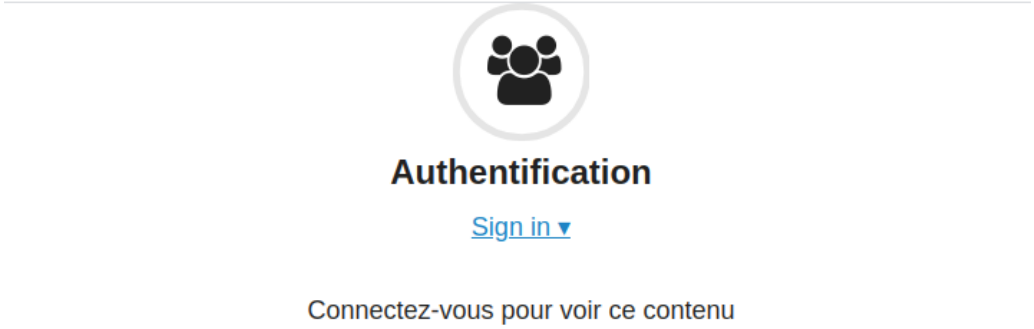


Рисунок 1.6 - перший запуск

Після відображення стартового екрану, користувачу буде запропоновано здійснити вхід, або пройти реєстрацію

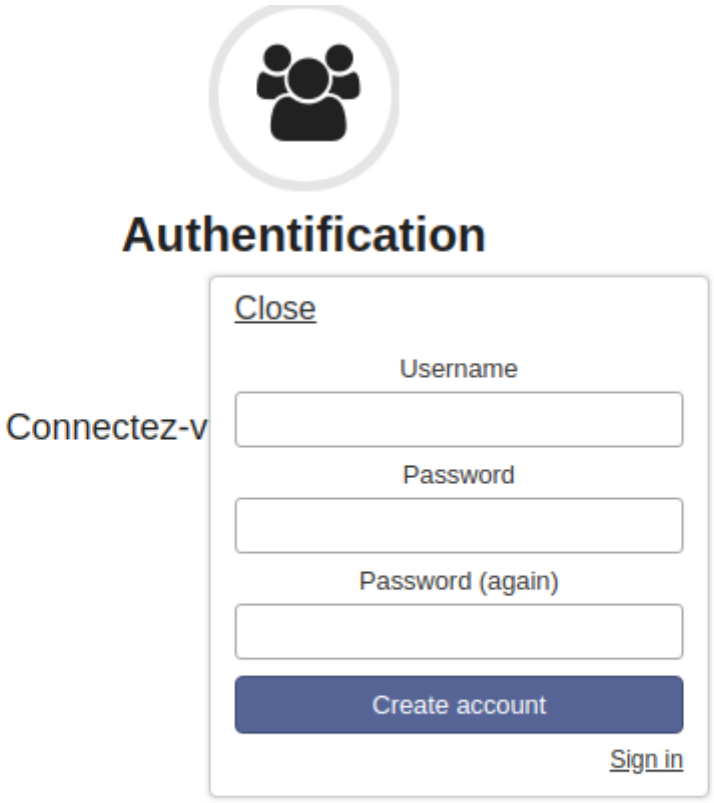


Рисунок 1.7 - реєстрація нового користувача

Після реєстрації новий користувач потрапляє на головну сторінку додатку, в якому йому буде запропоновано створити нову поїздку

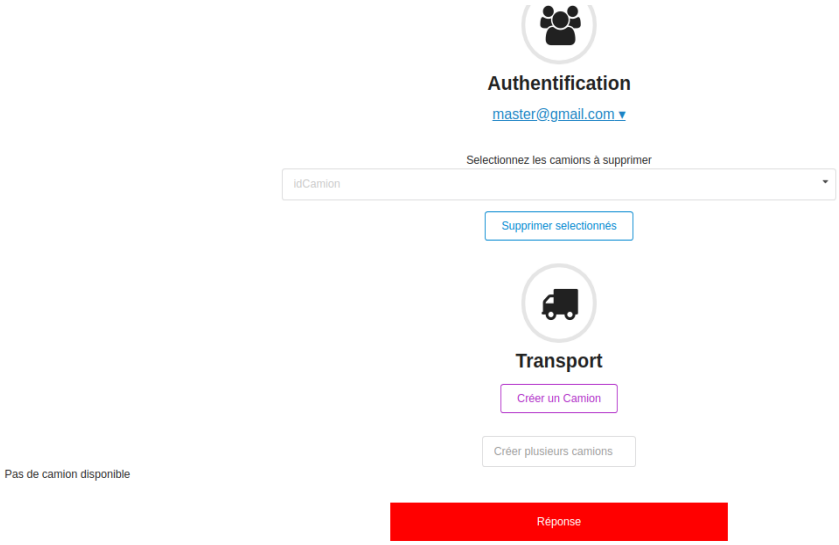


Рисунок 1.8 - головна сторінка додатку

Користувач має змогу додати новий вантажний автомобіль

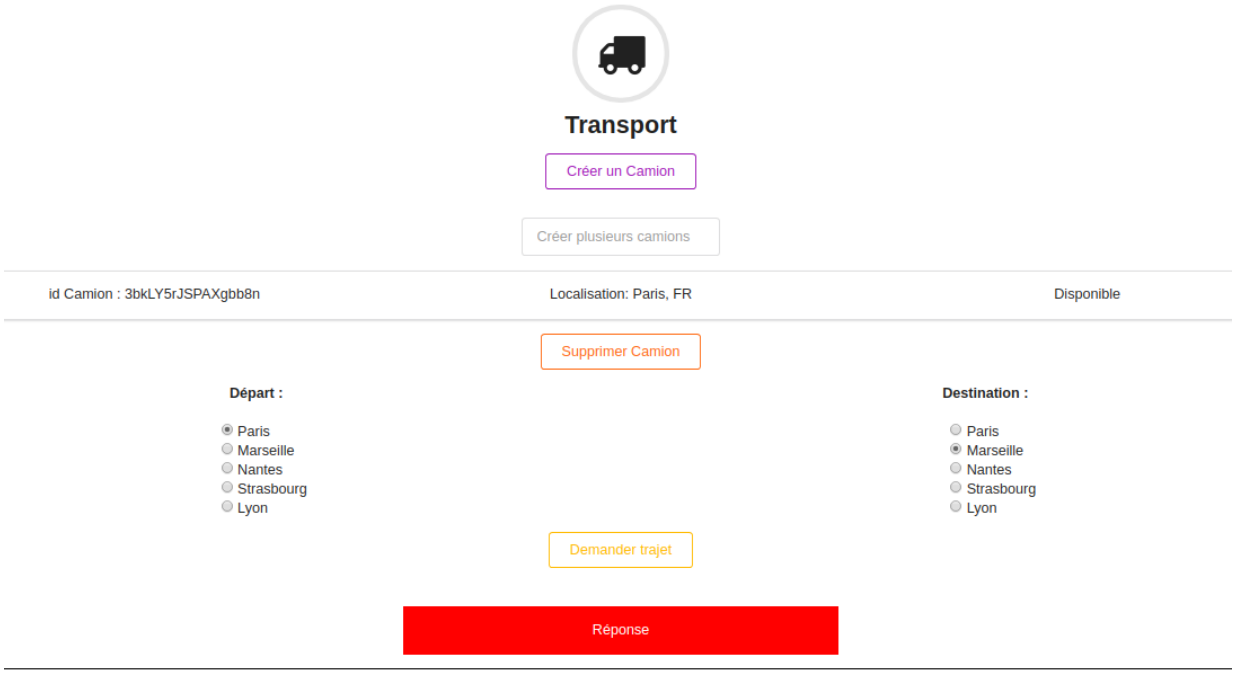


Рисунок 1.9 - Побудова маршруту

Після вибору маршруту можна здійснити поїздку в якій буде відображено основні і додаткові дані про маршрут з урахування відстані та часу

Рисунок 1.10 - відображення маршруту

Отже, в даному розділі було проведено розробку програмного забезпечення з аналізом вимог до програмної системи, проектуванням, що включало в себе вибір процесу розробки, моделювання архітектури системи. Було описано типові схеми використання системи.

Розроблена програмна реалізація відповідає усім поставленим вимогам, є сучасною та була написана відповідно до найкращих практик провідних компаній.

2. ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 План тестування

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

- відповідність вимогам, якими керувалися проектувальники та розробники;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з програмним забезпеченням та операційними системами;
- відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення баґів (помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників [28].Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). Процес розробки зазвичай передбачає коли та як буде відбуватися тестування.

Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах.

На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно [29].

2.2 Розробка тестів

Для тестування веб-сайтів було реалізовано unit-тести для перевірки правильності роботи серверних методів, та integration-тести для перевірки коректності роботи сценаріїв для виконання певних дій.

Модульне тестування — це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. У процедурному програмуванні модулем вважають окрему функцію або процедуру. В об'єктно-орієнтованому програмуванні — інтерфейс, клас. Модульні тести, або unit-тести, розробляються в процесі розробки програмістами та, іноді, тестувальниками білої скриньки (white-box testers).

Зазвичай unit-тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку [30].

Інтеграційне тестування — це фаза тестування програмного забезпечення, під час якої окремі модулі програми комбінуються та тестуються разом, у взаємодії. Інтеграційне тестування виконується після модульного тестування та перед верифікацією та валідацією ПЗ. Якщо розглядати цей процес як систему, то на вхід їй подаються модулі, які вже пройшли модульне тестування; потім модулі групуються в більші частини, виконуються тести передбачені планом, а на виході системи — інтегрована система, що готова до системного тестування [31].

Для тестування серверних методів було використано фреймворк Mocha. Mocha – фреймворк для тестування коду, написаного мовою JavaScript, що працює на основі node.js, підтримує роботу з браузерами, асинхронне тестування, звіти про покриття продукту тестами, а також роботу з будь-якими бібліотеками припущень [32].

Mocha може працювати з більшістю бібліотек припущень в JavaScript, включно з:

- should.js
- express.js
- chai
- better-assert
- unexpected

Як було зазначено, модульні тести призначені для тестування найменших частин програми, які можна протестувати. На лістингу 2.1 представлено реалізацію модульних тестів методів створення Trajets.

Лістинг 2.1 - тестування Flottes

```
// meteor test --driver-package=practicalmeteor:mocha
import {Meteor} from 'meteor/meteor';
import {resetDatabase} from 'meteor/xolvio:cleaner';
import {describe, it} from 'meteor/practicalmeteor:mocha';
import {expect} from 'meteor/practicalmeteor:chai';
import Trajets from '../imports/trajets';
const DDPCommon = Package['ddp-common'].DDPCommon;
if (Meteor.isServer) {
  describe('Fonctions relatives aux Trajets', () => {
    resetDatabase();
    const invocation = new DDPCommon.MethodInvocation({
      isSimulation: false,
      userId: 'dbAnEgsK22x5NAghP',
      setUserId: () => {},
      unblock: () => {},
      connection: {},
      randomSeed: Math.random()
    });
    it('insererTrajet: lise rajets ne dt pas ' +
      'être nulle', done => {
      DDP._CurrentInvocation.withValue(invocation, () => {
```

```
    Meteor.call('insererTrajet', '8Fd82Zw9pyMsE97yv', 'Paris, FR',
      'Marseille, FR', 25000, 'Nantes, FR');
  });
  expect(Trajets.find().fetch().length).to.not.equal(0);
  done();
});
it('dateFin: doit retourn un nom', done => {
  expect(Meteor.call('dateFin', Trajets.find().fetch()[0]._id))
    .to.be.a('number');
  done();
});
it('destinationIntermediaire: doit retourn un string', done => {
  expect(Meteor.call('destinationIntermediaire',
    Trajets.find().fetch()[0]._id)).to.be.a('string');
  done();
});
it('destinationFinale: doit retourn un string', done => {
  expect(Meteor.call('destinationFinale', Trajets.find().fetch()[0]._id))
    .to.be.a('string');
  done();
});
it('idCamion: idCamion do returne string', done => {
  expect(Meteor.call('idCamion', Trajets.find().fetch()[0]._id))
    .to.be.a('string');
  done();
});
it('terminerTrajet: terminé retourne true', done => {
  Meteor.call('terminerTrajet', Trajets.find().fetch()[0]._id);
  expect(Meteor.call('trajetTermine', Trajets.find().fetch()[0]._id))
  .to.equal(true);
  done();
```

```
    });
  });
}
```

Суть тесту полягає в тому, що послідовність дій користувача імітується програмно. На представленому лістингу, спочатку створюється шаблон редагування публікації. Далі здійснюється перевірка того, чи всі поля представлення коректно заповнені даними з моделі. Після цього, імітується зміни поля, та виконується підтвердження форми. Оскільки, після підтвердження викликається серверний метод, виклик якого є асинхронним, єдиний можливий варіант ззовні дочекатись змін, це використання setTimeout. Після того як мине одна секунда від підтвердження форми, з колекції витягується публікація, та виконується перевірка, чи поле було змінено.

Таким чином, було розглянуто методи тестування системи та реалізовано сценарії автоматичного тестування, а саме модульного та інтеграційного типів. Для цього використовувались такі бібліотеки як Jest, Enzyme, Mocha, та інші.

Усі використані інструменти використовуються та підтримуються провідними компаніями такими, як Facebook, Airbnb, Meteor development group, що дає впевненість у актуальності вибраних технологій та масштабованості. Було розглянуто декілька прикладів застосованих сценаріїв. Важливо пам’ятати, що автоматизоване тестування не є заміною ручного — швидше, це величезна підтримка для нього. В багатьох аспектах взаємодії користувача з програмою (зручне розташування кнопок, прості дії на кшталт авторизації і т.д.), присутність людини залишається незамінним «інструментом», який конкурентів не має. Тож найкраще поєднувати ручне та автоматизоване тестування — у пропорціях, які підійдуть проекту найкраще.

3 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

Метою цього розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності автоматизованої системи логістичних перевезень для системи Windows.

Для здійснення розрахунків використано реальні дані проведення науково-дослідної роботи, описано технологічний процес розробки із зазначенням трудомісткості кожної операції, визначено суму матеріальних затрат, та витрат на оплату праці основного і допоміжного персоналу, включно з відрахуванням на соціальні заходи, обчислено додаткові затрати, а також визначено економічну ефективність та термін окупності продукту.

3.1 Планування стадій та етапів проектування програмного забезпечення

Відповідно до ст. 1 Закону № 3792, комп'ютерна програма – це набір інструкцій у вигляді слів, цифр, кодів, схем, символів чи у будь-якому іншому вигляді, виражених у формі, придатній для зчитування комп'ютером, які приводять його в дію для досягнення певної мети або результату. В такому самому значенні розуміється цей термін і в контексті податкового обліку (пп. 8.2.2 Закону про прибуток). Статтею 8 Закону № 3792 передбачено, що комп'ютерні програми є об'єктом авторського права [33].

Діяльність з програмування – це процес створення (розробки) програми, який може бути представлений послідовністю таких кроків: формулювання вимог до програми та розробка алгоритму; кодування (запис алгоритму на

вибраній мові програмування); відлагодження; тестування; доопрацювання програми; розробка довідкової системи.

Проектування ПЗ складається з низки послідовних, цілеспрямованих, взаємозв’язаних та взаємообумовлених етапів, на які можна поділити весь часовий відрізок, що відводиться на розробку проекту (таблиця 3.1).

Таблиця 3.1 – Етапи розробки проекту

Найменування		Вид роботи	
стадії	Етапу	шифр	зміст роботи
1 Підгото-вча стадія	1.1 Вивчення стану предметної області	1.1.1	Дослідження проблеми
		1.1.2	Вибір платформи для реалізації проекту
		1.1.3	Вивчення та аналіз аналогічних розробок
		1.1.4	Економічне обґрунтування доцільності виконання проекту
	1.2 Розробка технічного завдання	1.2.1	Складання ТЗ
		1.2.2	Узгодження ТЗ із зацікавленими сторонами
		1.2.3	Складання плану та розрахунок розробки
2 Технічна пропозиція	2.1 Аналіз ТЗ та техніко-економічне обгрунтування проекту	2.1.1	Доведення техніко-економічного обгрунтування
		2.1.2	Аналіз ТЗ та визначення приоритетних аспектів розробки
		2.1.3	Доведення та уточнення загального обсягу робіт, строків виконання та витрат
3 Теорети-чна розробка	3.1 Теоретичне вивчення задачі	3.1.1	Дослідження технічних особливостей інформаційної системи
		3.1.2	Визначення переліку технологій, які використовуватимуться при розробці та мови програмування
		3.1.3	Визначення форматів даних та запитів і їх узгодження із розробниками проекту
		3.1.4	Розробка алгоритмів роботи програми на високому рівні
		3.1.5	Розробка структури систем забезпечення та схеми

			взаємодії її компонент
4 Практич-на реалізація	4.1 Реалізація окремих модулів ПЗ	4.1.1	Розробка алгоритмів роботи програми на низькому рівні
		4.1.2	Розробка окремих класів та компоновка їх у модулі
		4.1.3	Розробка графічного користувацького інтерфейсу для системи
	4.2 Відладка ПЗ	4.2.1	Автономна відладка окремих модулів системи
		4.2.2	Комплексна відладка ПЗ системи
		4.3.1	Розробка структурної схеми субмодуля

Продовження таблиці 3.1

Найменування		Вид роботи	
	4.3 Розробка субмодуля	4.3.2	Розробка функціональної схеми субмодуля
		4.3.3	Розробка алгоритму функціонування субмодуля
		4.3.4	Обгрунтування вибору елементного базису та розробка схеми
5 Доробка всього комплексу	5.1 Тестування всієї системи	5.1.1	Тестування системи у реальних умовах
		5.1.2	Доробка систем із урахуванням результатів тестування
	5.2 Підготовка документації по системі	5.2.1	Підготовка звіту про розробку системи
		5.2.2	Підготовка технічної документації
		5.2.3	Розробка довідкової системи, допоміжної і супроводжувальної документації, запис CD/DVD диска
6 Заключна стадія	6.1 Ознайомлення зацікавлених осіб з проектом	6.1.1	Підготовка презентації
		6.1.2	Демонстрація системи

Головна мета планування процесу розробки – це визначення необхідних ресурсів на всіх його етапах.

Традиційний процедурний підхід для розробки ПЗ в основі якого лежать алгоритми, процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію.

Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіати використання та покроковий, покомпонентний процес розробки.

Підготовча стадія, технічна пропозиція і заключна стадія при об’єктно-орієнтованому підході залишаються незмінними. Стадії теоретичної розробки,

практичної реалізації та доробки всього комплексу програмного забезпечення описані з точки зору специфіки об’єктно-орієнтованого підходу із врахування ЖЦ розробки ПЗ і представлені із уточненням фахівців, залучених у кожний робочий процес, та артефактів (таблиця 3.2) [34].

При створенні ПЗ за об’єктно-орієнтованим підходом необхідно залучити більшу кількість фахівців, більшу увагу приділити покроковій

покомпонентній розробці, що, можливо, збільшить робочий час і витрати. Однак, цей підхід значно скорочує витрати на подальший супровід і модернізацію, що приводить до зменшення загальних витрат.

Таблиця 3.2 - Робочі процеси життєвого циклу розробки ПЗ

№	Робочі процеси	Діяльності	Співробітники	Артефакти
1	«Визначення вимог»	Ідентифікація актантів (А) і варіантів використання (ВВ), Модель ВВ Описи (формалізація) ВВ і сценаріїв реалізації, Визначення пріоритетності ВВ Створення прототипів інтерфейсів користувача (ІК)	Системний аналітик Специфікатор ВВ	Модель ВВ Актанти Глосарій ВВ Прототип ІК
2	«Проектування»	Проектування архітектури, Визначення вузлів та мережевих конфігурацій Визначення систем та їх інтерфейсів Визначення підсистем та зв’язків між ними Визначення інтерфейсів підсистем Визначення архітектурно значущих класів та класів проектування, Визначення загальних механізмів проектування Проектування ВВ Проектування класів Визначення вимог до реалізації	Архітектор Інженер з ВВ	Модель про-ектування Модель розміщеня Опис арітектури Проекти реквізитів ВВ Класи про-ектування

3	«Реалізація»	Модель реалізації, Компоненти, Інтерфейси компонентів, Опис архітектури План збирання системи, Реалізація архітектури, Ітеративна реалізація класів Проектування з генерацією програмного коду, Збірка системи, Планування білдів Реалізація білдів і системи в цілому	Системний інтегратор Інженер-програміст	Модель реа-лізації Опис архітекту-ри План збірки Компоне-нти Підсист-еми реалізації Інтерфейси

Продовження таблиці 3.2

	«Тестування»	Модель тестування Тестові приклади Процедура тестування Тестові компоненти План і оцінка тестування Розробка тестів Тестування цілісності Тестування системи Оцінка результатів тестування	Інженер з тестування Системний тестер	Модель тестування Тестові приклади План тестування Оцінювання тестів

Терміни розробки залежать від замовника, від наданої необхідної інформації (зразків довідників, документів і т.і.), рівня та порядку оплати та інших умов, від замовлення, і можуть становити від 2 тижнів до 6 місяців і більше (за даними софтверних компаній).

Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки і становить від 200 до 5000 гривень (за даними аутсорсингових компаній). Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Для визначення загальної тривалості проведення робіт доцільно дані витрат часу по окремих операціях техпроцесу звести у таблицю (таблиця 3.3).

Таблиця 3.3 – Тривалість та трудоємність розробки та реалізації програмного проекту

Найменування роботи	Виконавець, посада, спеціальність	К-сть виконавців	Тривалість виконання роботи, дні
2	3	4	5
Дослідження предметної області,	Керівник		

вибір середовища для реалізації проекту, вивчення та аналіз аналогічних розробок	проекту	2	1
	Програміст		
Економічне обгрунтування доцільності виконання проекту, складання ТЗ, узгодження ТЗ із зацікавленими сторонами, складання плану та розрахунок розробки	Керівник проекту	2	1
	Програміст		
	Програміст		

Продовження таблиці 3.3

Доведення техніко-економічного обгрунтування, аналіз ТЗ та визначення приоритетних аспектів розробки, доведення та уточнення загального обсягу робіт, строків виконання та витрат	Керівник проекту Програміст	2	1
Теоретична розробка процедурний підхід	Програміст	1	3
Теоретична розробка об'єктно- орієнтований підхід	Системний аналітик, специфікатор ВВ, архітектор, інженер з ВВ	4	4
Практична реалізація процедурний підхід	Програміст	1	5
Практична реалізація об'єктно-орієнтований підхід	Системний інтегратор, інженер-програміст	3	7
Тестування програмного забезпечення	Інженер з тестування	1	3
Доробка всього комплексу програмного забезпечення	Програміст	1	4
Заключна стадія	Керівник проекту Програміст	2	1

Як для процедурного, так і для об'єктно-орієнтованого підходу, підготовча і заключна стадії та технічна пропозиція будуть виконуватися одинаково, теоретична розробка і практична реалізація для об'єктно-

орієнтованого підходу вимагатимуть залучення більшої кількості IT-спеціалістів, що, можливо, збільшить тривалість виконання роботи.

3.2 Розрахунок витрат на реалізацію проекту та оцінка економічної ефективності

Оцінка вартості комп’ютерних програм базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням

амортизації. Оподаткування заробітної плати програмістів підприємства-розробника, яким встановлено певний посадовий оклад, відбувається аналогічно оподаткуванню зарплати працівників інших галузей господарства.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, розробники часто йдуть на додаткові заходи їх стимулювання, які найчастіше втілюються у:

- певній системі преміювання (за виконання графіку розробки чи його дострокове виконання, оптимізацію етапів розробки тощо). Премії оподатковуються аналогічно витратам на оплату праці;

- авторській винагороді за кожну продану одиницю програми (роялті). Сума роялті не обкладається внесками до Пенсійного фонду та фондів соціального страхування, а використання авторських прав оформлюється відповідним договором (ліцензійний авторський договір).

Разом з тим до створення ПЗ можуть бути залучені також позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем:

- якщо виконавець за договором підряду не зареєстрований фізособою-підприємцем, то виплати за таким договором оподатковуються ПДФО за ставкою 15 %, також нараховуються (у розмірі 33,2 %) та утримуються (у розмірі 2 %) внески до Пенсійного фонду (п. 1 та п. 4 ст. 4 Закону № 400). Оподаткування провадиться у межах максимальної величини, що дорівнює 15 розмірам прожиткового мінімуму, встановленого для працездатних осіб;

- якщо ж виконавець за договором підряду є фізособою-підприємцем, яка сплачує єдиний податок чи знаходиться на загальній системі оподаткування, виплати, що здійснюються на його користь в рамках договору підряду, не оподатковуватимуться ПДФО.

Вважатимемо, що усі програмісти працюють у штаті підприємства-розробника і їм встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість і основна заробітна плата кожного учасника техпроцесу представлено у таблиця 3.4.

Таблиця 3.4 – Сума часткових заробітних плат

	Місячний оклад, грн.	Денна зар. плата, грн	Трудомісткість, людино-дні		Основна заробітна плата, грн.	
			Процедурний підхід	Об'єктно-орієнтований підхід	Процедурний підхід	Об'єктно-орієнтований підхід
Керівник проекту	18200	728	4	4	2912	2912
Інженер-програміст	15200	608	17	19	10336	11552
Інженер-тестувальник	12100	484	3	3	1452	1452
Дизайнер	7500	300	3	3	900	900
Всього			27 днів	29 днів	15600 грн.	16816 грн.

Визначимо витрати на основну заробітну плату. Вони складають суму заробітних плат за кожну із робіт. Витрати на основну зарплату для процедурного та об'єкт-орієнтованого підходів:

Додаткова заробітна плата обчислюється за формулою $ЗП_{дод}=ЗП_{осн}\cdot 0.2$.

Обчислимо витрати на додаткову зарплату для двох підходів:

$ЗП_{осн(п)}$ = 2912 + 10336 + 1452 + 900 = 15600 (грн.);

$ЗП_{осн(о)}$ = 2912 + 11552 + 1452 + 900 = 16816 (грн.);

Це сума часткових заробітних плат за кожну роботу, і вона приведена в таблиці 3.4.

Додаткова заробітна плата обчислюється за формулою:

$$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн}\ .$$

$$ЗП_{дод(п)} = 15600 \cdot 0.2 = 3120 \text{ (грн.)};$$

$$ЗП_{дод(о)} = 16816 \cdot 0.2 = 3363.2 \text{ (грн.)};$$

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$$ФЗП = ЗП_{осн} + ЗП_{дод}\ .$$

$$ФЗП_{п} = 15600 + 3120 = 18720 \text{ (грн.)};$$

$$ФЗП_{о} = 16816 + 3363.2 = 20179.2 \text{ (грн.)};$$

З цієї суми утримуються обов'язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов'язкове соціальне страхування та податкової соціальної пільги.

Таким чином обов'язкові відрахування на заробітну плату для працівників складають:

Утримання єдиного соціального внеску:

$$Відр_{ЄСВ1} = 0.036 \cdot ФЗП = 0.036 \cdot 18720 = 674 \text{ грн.};$$

$$Відр_{ЄСВ2} = 0.036 \cdot ФЗП = 0.036 \cdot 20179.2 = 726.5 \text{ грн.}$$

Податок на доходи фізичних осіб:

$Відр_{ПДФО1}$= 0.15 · (*ФЗП* – $Відр_{ЄСВ}$) = 0.15·(18720-674) = 2706.9 грн.;

$Відр_{ПДФО2}$= 0.15 · (*ФЗП* – $Відр_{ЄСВ}$) = 0.15·(20179.2-726.5) = 2917.9 грн. $Відр_1 = Відр_{ЄСВ1} + Відр_{ПДФО1}$ = 674 + 2706.9 = 3380.9 грн.;

$Відр_2 = Відр_{ЄСВ2} + Відр_{ПДФО2}$ = 726.5 + 2917.9 = 3644.4 грн.

Законом № 1621 підрозділ 10 розділу XX Перехідних положень Податковий кодекс України від 02 грудня 2010 року № 2755-VI зі змінами та

доповненнями (далі – ПКУ) доповнено пунктом 16, яким тимчасово, до 1 січня 2015 року, встановлено військовий збір.

Водночас Законом України від 28 грудня 2014 року № 71-VIII «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» (далі – Закон № 71), який набрав чинності з 01.01.2015, оподаткування військовим збором подовжено до набрання чинності рішенням Верховної Ради України про завершення реформи Збройних Сил України (пункт 16 підрозділ 10 розділу XX Перехідних положень ПКУ) [35].

Платниками збору в ромірі 1,5% від заробітної плати є особи, визначені пунктом 162.1 статті 162 цього ПКУ (підпункт 1.1 пункт 16 підрозділу 10 ПКУ) [36].

Військовий збір з фізичних осіб:

$$ВЗФО = 0{,}015 \cdot ФЗП$$

$$ВЗФО_1 = 0{,}015 \cdot 18720 = 280{,}8 \text{ грн.}$$

$$ВЗФО_2 = 0{,}015 \cdot 20179.2 = 302{,}7 \text{ грн.}$$

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%, для підприємців розмір єдиного внеску залежно від класу професійного ризику виробництва становить від 36,76 % до 49,70 %. Зокрема, видання програмного забезпечення – 36,77%) [37].

Нарахування на фонд оплати праці (*ФОП*):

$$ФОП_{ЄСВ} = 0.3677 \cdot ФЗП$$

$ФОП_{ЄСВ1}$ = 0.3677 · 18720= 6883.34 грн.;

$ФОП_{ЄСВ1}$ = 0.3677 · 20179.2= 7419.82 грн.

Всього витрат:

$B_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1}$ = 18720+ 6883.34 = 25603.34 грн.;

$B_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2}$ = 20179.2+ 7419.82 = 27599.02 грн.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{Bi} = q_i \cdot p_i$$

де q_i– кількість витраченого матеріалу i-го виду; p_i– ціна матеріалу i-го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{B\,i}$$

Проведені розрахунки занесемо у таблицю 3.5:

Таблиця 3.5 – Зведенні розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	SCRUM-дошка	шт	1	500,00	500,00
2	Папір для друку	листів	120	0,25	30,00
3	Чорнила для принтера	шт	1	65,00	65,00
4	Маркери	шт	4	14,00	56,00
Всього					651,00

Згідно постанов Національної комісії, що здійснює державне

регулювання у сфері енергетики (згідно Постанов Національної комісії, що здійснює державне регулювання у сфері енергетики (НКРЕ) від 22.01.2001р. №47 зі змінами і доповненнями, від 06.12.2002р. № 1358, від 22.10.2004р. № 1030, від 28.04.2016р. № 755 на підставі Закону України від 03.12.1999р № 1274-XIV "Про внесення змін до Закону України "Податкового кодексу України", згідно Закону України "Про міський електричний транспорт" від 29 червня 2004 року № 1914 – IY; також затвердженого постановою Кабінету

Міністрів України від 01.06.2011р. № 869, Порядку розрахунку роздрібних тарифів на електричну енергію, тарифів на розподіл електричної енергії (передачу електричної енергії місцевими (локальними) електромережами), тарифів на постачання електричної енергії за регульованим тарифом, затвердженого постановою Національної комісії, що здійснює державне регулювання в сфері енергетики та комунальних послуг (НКРЕКП) "Про ринкове формування роздрібних тарифів на електричну енергію, що відпускається для кожного класу споживачів, крім населення, на території України", ця сума становить 289,85 коп/кВт.г. [38].

Витрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_е = W \cdot T \cdot S$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії.S=2,90грн/кВт.г.

1 $З_{е1} = 1{,}5 \cdot 216 \cdot 2{,}9 = 939{,}6$ грн.;

2 $З_{е2} = 1{,}5 \cdot 232 \cdot 2{,}9 = 1009{,}2$ грн.

Розрахунок суми амортизаційних відрахувань.

Комп’ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Амортизація за час (період) використання обладнання (комп’ютера) складає долю затрат, які припадають на дослідницьку роботу (написання програми), і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}}$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %;

$T_{год}$ – річний фактичний час роботи обладнання по написанню програми, год.

$T_{фак}$

Таблиця 3.6 – Перелік обладнання, необхідного для розробки програми

Найменування	Кількість, шт.	Ціна за одиницю продукту, грн.	Сума, грн.
Комп’ютер	4	15775	63100
Принтер	1	2000	2000
Apple Developer Program Membership	1	3250	3250
Google Play Developer Membership	1	900	900
Хостинг	1	995	995
Доменне ім'я	1	366	366
Всього			70611

$$A_1 = \frac{70611 \cdot 0{,}6 \cdot 216}{2016} = 4539{,}3 \text{ грн.;} \qquad A_2 = \frac{70611 \cdot 0{,}6 \cdot 232}{2016} = 4875{,}52 \text{ грн.}$$

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$НВ = 0{,}5 \cdot ЗП_{осн}$.

$НВ_п = 0{,}5 \cdot 15600 = 7800$ грн.;

$НВ_о = 0{,}5 \cdot 16816 = 8408$ грн.

Проведемо розрахунок вартості створюваного програмного продукту.

Вартість продукції включає у собі собівартість і планований прибуток.

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво:

$$Св = В_{зп} + З_{мв} + З_{е} + А + НВ;$$

$Св_1 = 25603,3 + 651 + 939,6 + 4539,3 + 7800 = 38882,2$

(грн.)

$Св_2 = 27599,02 + 651 + 951,8 + 4875,52 + 8408 = 41891,74$ (грн.)

Приймемо прибуток на рівні 27%. Для нових інноваційних продуктів, що користуються високим попитом на ринку.

Отже, вартість розробленого програмного забезпечення:

$В_1 = Св_1 + 0,27 \cdot Св_1 = 38882,24 + 0,27 \cdot 38882,24 = 49380,44$ грн.;

$В_2 = Св_2 + 0,27 \cdot Св_2 = 41891,74 + 0,27 \cdot 41891,74 = 53202,51$ грн.

Економічна ефективність (E) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св}$$

де $П$ – прибуток, $П = В - Св$; $Св$ – собівартість.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку.

$E_п = (49380,44 – 38882,24) / 38882,24 = 0,27;$

$E_о = (53202,51 – 41891,74) / 41891,74 = 0,27.$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{}$$

E

У нашому випадку $T_{ок1}$= $T_{ок2}$ =1/0,27=3,7 років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Загальна вартість пропонованих робіт по розробці програмного продукту становить 18720 грн. для першого варіанту та 20179,2 грн. для другого. Оскільки ефективність для обидвох проектів відповідно до

встановленого рівня прибутку становить 0,27, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

3.3 Визначення витрат на супровід і модернізацію програмного продукту та уточнений аналіз ефективності вкладених інвестицій

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за процедурним методом 55% від початкових витрат, а за об'єктно-орієнтованим – 25%.

Собівартість модернізації:

$СвМ_п = 0,6 \cdot Св_п = 0,6 \cdot 38882,24 = 23329,35$ грн.,

$СвМ_о = 0,2 \cdot Св_о = 0,2 \cdot 41891,1 = 8378,22$ грн.

Для споживача вартість модернізації:

$М_п = 0,6 \cdot В_п = 0,6 \cdot 47719,7 = 29628,27$ грн.,

$М_о = 0,2 \cdot В_о = 0,2 \cdot 51710,7 = 10640,50$ грн.

Таким чином, уже після першої модернізації, загальні витрати на створення і супровід ПЗ для виробника за об'єктно-орієнтованим методом менші, ніж за процедурним, навіть якщо його собівартість є дещо дорожчою.

$ЗВ_{п\ (вир)} = 38882,24 + 23329,35 = 62211,59$ грн.,

$ЗВ_{о\ (вир)}$ = 41891,74 + 8378,22 = 50269,96 грн.

Як і для споживача:

$ЗВ_{п}$ = 49380,44 + 29628,27= 79008,71 грн.,

$ЗВ_{о}$ = 53202,51 + 10640,50 = 63843,01 грн.

Річна економія витрат за всіма можливими напрямами і додатковими витратами, пов'язаними з супроводом і тільки одноразовою модернізацією (у розрахунку на одиницю продукції) при об’єктно-орієнтованому методі порівняно із процедурним:

$\Delta C_{(вир)} = ЗВ_{1(вир)} - ЗВ_{2(вир)} = 62211,59 - 50269,96 = 11941,63$ грн,

$\Delta C = ЗВ_1 - ЗВ_2 = 79008,27 - 63843,01 = 15165,70$ грн,

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{ТВi} - \sum_{i=1}^{t} ІК_i \alpha_{ТВi};$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

$\alpha_{ТВi}$ – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(\quad)^n};$$
$$1+i$$

де i – ставка дисконтування або норма дисконту, $i = 0,27$;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$$\alpha_0 = 1, \quad \alpha_1 = \frac{1}{(1 + 0,27)} = 0,79$$

Вважатимемо, що обидва програмних продукта одинаково забезпечують

потреби і вимоги споживача, і тому придбання першої чи другої програми одинаково вплинуть на розмір його додаткових доходів на вкладений капітал, Тому приймемо цю величину за постійну, а порівняння дохідності двох проектів проведемо тільки за витратами.

$ЧПД_1' = ГП + 0{,}78 \cdot ГП - 47719{,}7$ $- 0{,}78 \cdot 28631{,}8 = 1{,}78\,ГП - 70052\,{,}5$ грн.,

$ЧПД_2' = ГП + 0{,}78 \cdot ГП - 51710{,}7$ $- 0{,}78 \cdot 10342{,}14 = 1{,}78\,ГП - 59777\,{,}57$ грн.

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об'єктно-орієнтованим підходом, становить 10274,93 грн. Отже, сучасну комп'ютеру програму доцільно виконувати по об'єктно-орієнтованій парадигмі,

Усі резутати розрахунків, приведених у даному розділі, зведені у таблицю 3.7

Таблиця 3.7 – Результати розрахунків

№ п.п.	Назва	Процедурний підхід	Об'єкто-орієнтова-ний підхід
1	Зарплата основна, грн	15600	16816
2	Зарплата додаткова, грн	3120	3363.2
3	Фонд заробітньої плати (1+2)	18720	20179.2
4	Обов'язкові відрахування на заробітну плату (4.1+4.2), грн	3480.9	3644.4
4.1	Утримання єдиного соціального внеску (3,6%), грн	674	726.5
4.2	Податок на доходи фізичних осіб (15%), грн	2806.9	2917.9
5	Військовий збір (1.5%), грн	280.8	302.7
6	Відрахування на ФОП (36,77%), грн	6883.34	7419.82
7	Разом на оплату праці (3+6), грн	25603.34	27599.02
8	Матеріальні витрати, грн	651	651

9	Електроенергія, грн	939.6	1009.2
10	Амортизація, грн	4539.3	4875.52
11	Накладні витрати, грн	7800	8408

Продовження до таблиці 3.7

12	Разом на ін. витрати (8+9+10+11)	13929.9	14943.72
13	Собівартість, грн	38882.24	41891.74
14	Прибуток, грн	10498.20	11310.77
15	Вартість розробленого ПЗ, грн	49380.44	53202.51
16	Економічна ефективність	0.27	0.27
17	Термін окупності, років	3.7	3.7
18	Собівартість модернізації, грн	23329.35	8378.22
19	Супровід і модернізація, грн	29628.27	10640.50

Загальна вартість пропонованих робіт по розробці програмного продукту становить 79008,71 грн. для першого варіанту та 63843,01 грн. для другого. Оскільки ефективність для обидвох проектів відповідно до встановленого рівня прибутку становить 0,27, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

При використанні об’єктно-орієнтовного підходу зменшується кількість працівників, які залучаються у проект, та зменшуються витрати на реалізацію проекту, але для підтримки проекту і його подальшої модернізації все ж в майбутньому потрібні набагато більші витрати. Для функціонального підходу потрібна більша кількість працівників, часу і коштів, але на модернізацію і підтримку в загальному потрібно менше ресурсів, і у висновку програма виконана по функціональній парадигмі стає дешевшою для споживача, і приносить економію ресурсів для розробників.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ

4.1 Охорона праці

Тема дипломної роботи передбачає створення автоматизованої системи проектування маршрутів з аналізом локального ресурсу для операційних систем типу Windows. Розробник системи в процесі розробки інтеграції експлуатує персональні електронно-обчислювані машини. Для розробки системи потрібне офісне приміщення, в якому будуть розміщуватись комп’ютеризовані робочі місця, тому потрібно спланувати приміщення відповідно до державних нормативно-правових актів та законів.

При роботі з комп'ютером користувач піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітні поля (діапазон радіочастот: ВЧ, УВЧ і СВЧ), рентгенівське та ультрафіолетове випромінювання, шум і вібрація, запиленість робочої зони, накопичення статичної електрики і інші.

Для зменшення впливу негативних ефектів, потрібно правильно розмістити обладнання та робочі місця в офісі. Список нормативно-правових актів, що регулюють дане питання, є досить широким. Так, обов’язки роботодавця щодо забезпечення працівникам комфортних та безпечних умов для здійснення роботи, а також права працівників на такі умови передбачено частиною 2 ст. 2 та ч. 1 ст. 21 КЗпП, а також ст. 13 Закону України «Про охорону праці».

В Законі України «Про охорону праці» вказано основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні [33].

Для оцінки відповідності робочого приміщення відповідно до актів потрібно заміряти його розміри. Так в приміщенні розміром 12 м^2 (4х3 м) і висотою 3,4 метра можна розмістити не більше двох комп’ютеризованих робочих місць (не менше 6 м^2 на одне робоче місце).

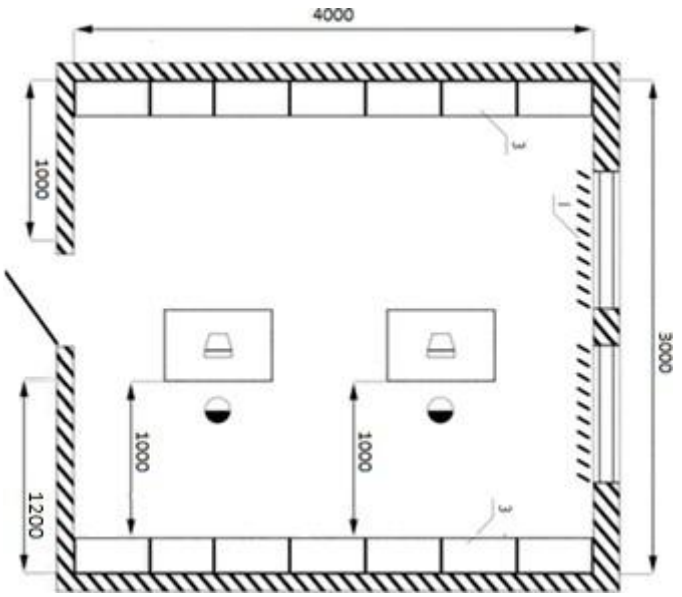
Окрім площі на одне робоче місце в нормативно правових актах вказано потрібний об’єм приміщення – 20 м^3. Об'єм приміщення в даному випадку становить 40,8 м^3 а об'єм, що припадає на одне робоче місце – 20,4 м^3. Таким чином норматив щодо об'єму приміщення на одне робоче місце виконується.

Планування розміщення комп'ютеризованих робочих місць у приміщенні проводимо з урахуванням таких вимог:

- робочі місця розміщуються на відстані не менше 1 м від стіни зі світловими прорізами (вікнами);
- відстань між бічними поверхнями має бути не меншою за 1,2 м;
- відстань між тильною поверхнею одного комп’ютеризованого робочого місця та екраном іншого не повинна бути меншою за 2,5 м;
- прохід між рядами робочих місць має бути не меншим за 1м.

Найкраще розмістити комп'ютеризовані робочі місця рядами вздовж стіни з вікнами. Це дасть змогу виключити дзеркальне відбиття на екрані джерел природного світла (вікон) та потрапляння останніх у поле зору операторів, що погіршує їх зорову роботу.

Наведемо план офісного приміщення для розробки програмного забезпечення з комп'ютеризованими робочими місцями (рис. 4.1):



1 – комп'ютеризоване робоче місце;

2 – сонцезахисні жалюзі;

3 – шафи для зберігання;

Рисунок 4.1 – План приміщення з комп'ютеризованими робочими місцями.

Параметри, за якими оцінюється мікроклімат, встановлюється відповідно до пори роки і категорії роботи. Для теплого періоду, температура повинна бути в проміжку +23 - +25 °С, вологість 40 - 60 % а швидкість повітря ≤ 0.1 м/с. Для холодного періоду температура повітря повинна бути в проміжку +22 - +24 °С. Всі показники задовольняють вимогам для робіт категорії легка 1а і є задовільними для здоров’я людини.

Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначається світловий потік, що падає на робочу поверхню.

Приміщення, в яких встановлені персональні комп’ютери, повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006.

Природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт

природною освітленості (КПО) не нижче ніж 1,5%. Розраховується КПО за методикою, викладеною в ДБН В.2.5-28-2006.

Штучне освітлення в приміщеннях з робочими місцями має здійснюватися системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення додатково встановлюються світильники місцевого освітлення). Зазначення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення.

Приміщення, має 2 вікна з лінійними розмірами: ширина – 1 м, висота – 1,8 м, відповідно площа кожного вікна – 1,8 м^2. Вікна мають регульовані пристрої для відкривання та обладнані жалюзі з можливістю захисту працюючих від прямого попадання сонячних променів і регулювання рівня освітленості в приміщенні. Відблискування поверхонь обмежується за рахунок правильного вибору світильників та розташування робочих місць відносно джерел освітлення. Яскравість відблисків на сучасних моніторах не перевищує 35 кд/м^2. В досліджуваному приміщенні використовується система загального рівномірного штучного освітлення. Мається 2 ряди світильників Л201Б 2х400.3, у кожному з яких знаходиться по чотири лампи типу ЛБ-40.

Споживачі електроенергії: 2 ПЕОМ, 2 дисплея, 4 світильника, 1 кондиціонер. Кожне робоче місце обладнане 4-ма розетками по 220 В. Заземлені конструкції захищені діелектричними сітками від випадкового дотику. Усе електроустаткування має апаратуру захисту від струму короткого замикання. Отже, можна зробити висновок, що кабінет задовольняє вимогам електробезпеки у приміщенні, в якому встановлені ЕОМ, відображені в НПАОП 0.00-1.28-10.

У приміщенні маються внутрішні джерела постійного шуму – вентилятори блоків ЕОМ, дисководи. Фактичний обмірюваний рівень шуму в робочій зоні задовольняє нормативному рівню шуму (не повинний

перевищувати 50 дБА згідно з ДСН 3.3.6.037- 99).

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

Приміщення відноситься до зони П-IIа згідно з ДНАОП 0.00-1.31-99 і до категорії пожежної небезпеки В. Горючі рідини, пил та волокна у приміщенні не використовуються і не виділяються.

Ймовірними причинами виникнення пожежі можуть бути несправність електрообладнання, короткі замикання внаслідок виходу з ладу електроустаткування, порушення правил протипожежної безпеки тощо.

4.2 Оцінка стійкості роботи об'єкту економіки до впливу вражаючих факторів ядерної зброї.

У сучасній війні об'єктами поразки будуть не тільки угруповання збройних сил, але й адміністративно-політичні центри, великі міста, об'єкти промисловості, енергетики, зв'язку та сільського господарства, що перебувають у глибокому тилу. Масоване застосування сучасних засобів ураження неминуче приведе до великих втрат серед населення, якщо не буде вжито необхідних заходів щодо його захисту. Для цього необхідно робити оцінку стійкості об'єктів економіки до впливу вражаючих факторів, засобів ураження, а саме ядерної зброї. Під стійкістю роботи об'єктів, розуміють їх здатність виконувати свої функції в умовах військового часу.

На стійкість роботи об'єктів народного господарства в воєнний час впливають наступні фактори:

– Надійний захист робітників та працівників від зброї масового ураження;

– Здатність інженерно-технічного комплексу об’єкту протидіяти у деякій мірі ударній хвилі, світловому випромінюванню та радіації;
– Захист об’єкта від дії вторинних вражаючих факторів (пожарів, вибухів, затоплень, отруєння СДОР);
– Надійність системи постачання об’єкту всім необхідним для виготовлення та випуску продукції (сировиною, паливом, електроенергією, водою та ін.);
– Стійкість та неперервність управління виробництвом та ЦО;
– Підготовка об’єкта до проведення рятувальних та невідкладних аварійно-відновлюваних операцій та робіт по відбудовуванню пошкодженого виробництва. [39]

Проведемо оцінку об’єкту економіки до впливу вражаючих факторів ядерної зброї, в якості досліджуваного об’єкта візьмемо модельний цех.

Вихідні дані для оцінки стійкості будівлі до впливу вражаючих факторів ядерного вибуху, наведено в таблиці 4.1.

Таблиця 4.1 – Вихідні дані для оцінки будівлі

№	Найменування	Умовні позначення та одиниці виміру	Числові характеристики
1	Потужність вибуху	q, кг	10000
2	Вид вибуху		Повітряний
3	Радіус міста	R , км	5,5
4	Віддалення об’єкта від центра міста (точки прицілу)	Rr, км	5,7
5	Імовірність максимального відхилення боєприпасів від точки прицілу	rвідх, км	0,7
6	Об’єкт розташований відносно міста по азимуту	β, град.	115

7	Рівень радіації на початок утворення радіоактивного зараження (3 год. після вибуху)	Р/год.	167
8	Коефіцієнт послаблення: кам’яних будівель	Кпосл.	-
	ПРУ	Кпосл.	50-100
	Коефіцієнт послаблення дерев’яних будівель	Кпосл.	-
9	Тривалість робочої зміни	12 годин	

Н

Таблиця 4.2 - Характеристику модельного цеху

№	Найменування	Характеристики
1	Тип будівель	Із металевим каркасом та бетонним наповненням
2	Дах (матеріал)	Черепиця червона
3	Дверні дерев'яні	Пофарбовані в білий колір
4	Віконне оправлення	Пофарбовані в білий колір
5	Конвеєрна прогумова тканина	Є
6	Мотлох х/б темного кольору	Є
	Крани та кранове обладнання	Є
7	Верстати: важкий, середній, легкий	Є
8	Електромережа- кабельні лінії	Підземні
9	Трубопровід	Заглиблений на 20 см.

Оцінка стійкості виконується в такій послідовності:

На плані міста, в межах якого розміщується об'єкт відмічається положення точки прицілу, і вона з'єднується прямою лінією з центром об'єкта. З точки прицілу в масштабі плану описується коло радіусом, який дорівнює можливому максимальному відхиленню ядерного боєприпасу $r_{відх}$. Точка перетину кола прямою, яка з'єднує точку прицілювання і центра об'єкта, приймається за можливий центр вибуху [40].

Обчислюємо відстань від об'єкта до найближчого ймовірного центра вибуху, Rx:

$$Rx = Rr - r_{відх} = 5,7 - 0,7 = 5 \text{ (км)}$$

Надмірний тиск боєприпасу потужністю 1000 кг на відстань 5 км до центра вибуху при повітряному вибусі. Він складає 30 кПа. Знайдене значення

ΔРф = 30 кПа і буде максимальним, оскільки воно відповідає випадку, коли центр вибуху буде на мінімальній відстані від об'єкта, $\Delta P_{фmax} = 30$ кПа.

Виділяємо основні елементи цеху металоконструкцій: будівля

цеху, в технологічному обладнанні - крани та кранове обладнання, верстати важкі; електромережа - кабельні лінії підземні; трубопровід наземний.

Межа стійкості об'єкта до ударної хвилі визначається за мінімальною межею стійкості всіх елементів, які входять до складу об'єкта.

Порівнюють знайдену межу стійкості об'єкта $\Delta P_{\phi lim}$ з очікуваним максимальним значенням надлишкового тиску $\Delta P_{\phi max}$. Якщо виявиться, що:

$\Delta P_{\phi lim} \geq \Delta P_{\phi max}$, то об’єкт стійкий до ударної хвилі

$\Delta P_{\phi lim} < \Delta P_{\phi max}$, не стійкий

Будівля цеху 20 кПа (межа стійкості) ≤ 30 кПа ($\Delta P_{\phi max}$) – елемент не стійкий до ударної хвилі.

Крани та кранове обладнання 30 кПа (межа стійкості) ≥ 10 кПа ($\Delta P_{\phi max}$) – елемент стійкий до ударної хвилі, 30 кПа = 30 кПа – елемент стійкий до ударної хвилі.

Трубопровід заглиблений: 250 кПа (межа стійкості) ≥ 30 кПа) – елемент стійкий до ударної хвилі.

$\Delta P_{\phi lim}$ = 20 кПа, а $\Delta P_{\phi max}$ = 10 кПа – об’єкт в цілому стійкий до впливу ударної хвилі.

Проведемо оцінку стійкості об’єкта до впливу світлового випромінювання ядерного вибуху.

В якості показника стійкості об'єкту до взаємодії світлового випромінювання приймається мінімальне значення світлового імпульсу, при якому може відбутись загорання матеріалів або конструкцій будівель або споруд, в результаті чого виникнуть пожежі на об'єкті.

Це значення світлового імпульсу прийнято вважать межею стійкості об'єкта до взаємодії світлового випромінювання ядерного вибуху $V_{св\ lim}$.

Світловий імпульс для боеприпасу потужністю 1000 кг на відстані 5 км. до центру вибуху про порівняному вибуху. Він становить $V_{св\ max}$ = 2900 кДж/м^2.

Таблиця 4.3 - Стійкість цеху до впливу світлового випромінювання

Цех	Коротка характеристика	Елементи, що - згоряють	Світловий імпульс що викликає згоряння. кДж/м2	Межа стійкості елементів цеху	Межа міцності цеху. Vсв lim	Доцільна межа міцності Vсв max
Дерево обробний	Бетонний	Дах : черепиця	1260	1260	420	2900
		Двері : пофарбовані в білий колір	1760	1760		
		Віконне оправлення : пофарбовані в білий колір	1760	1760		
		Мотлох х/б темного кольору	420	420		
		Конвеєрна прогумована тканина	840	840		

Так як, межа міцності цеху до світлового випромінювання $V_{св\ lim}$ = 420 кДж/м^2 , а максимальне світлове випромінювання $V_{св\ max}$ = 2900 кДж/м^2 , тому $V_{св\ max} > V_{св\ lim}$,отже об’єкт не стійкий до світлового випромінювання.

Визначаємо зону пожеж, в якій опиниться цех. Так як $V_{св\ max}$ =2900 кДж/м^2, робимо висновок, що цех у зоні суцільних пожеж.

На об’єкті при повітряному вибусі потужністю 1000 кг очікується $V_{св\ max}$= 2900 кДж/м^2 та $\Delta P_{фmax}$=30 кПа, що спричинює згорання елементів цеху. Необхідно підвищити стійкість роботи цеху. Застосувати вогнезахисну фарбу для дверей та віконних рам. Прибрати з цеху мотлох та конвеєрну прогумовану тканину замінити більш вогнетривким матеріалом, можливо замінити черепицю.

4.3 Висновок до розділу 4

У даному розділі були розглянуті умови праці для офісного приміщення, у якому програмістами розробляється та

експлуатується програмний продукт. Було проведено аналіз шкідливих та небезпечних виробничих чинників, наявних у даному приміщенні. Розміри приміщення та параметри робочих місць відповідають нормам чинного законодавства з охорони праці. Усі параметри приміщення відповідають необхідним нормативам.

ВИСНОВОК

В результаті виконання магістерської роботи була розроблена автоматизована система для документообігу логістичних перевезень. Були досягнуті такі цілі:

1. Розроблювана автоматизована система проектування будівельного кошторису із аналізом локального ресурсу для операційних систем типу Windows дозволяє проектувати поїздки з врахуванням таких категорій як місто та час. Це дозволить більш точно описувати проект користувача і проводити оцінку враховуючи всі вищеперераховані параметри.

2. Графічне представлення динамічного стану проекта. Це дозволить детально за допомогою діаграм побачити стан кожної з категорій поїздки, так як стан може мінятись незалежно один від одного і важливо мати візуальне бачення як проекта вцілому, так і окремих його частин.

3. Експорт вхідних і вихідних даних обчислення в форматі JSON. У разі виникнення помилки обчислення це дозволить відслідкувати альтернативні та вийняткові ситуації при обчисленнях.

4. При розробці системи оцінки вартості житлового будівництва була використана документно - орієнтовна база даних. Це дозволить пришвидшити роботу системи оскільки така база даних найкраще підходить

для формування більш гнучких запитів та для зберігання документів зі складною ієрархією.

5. Сервер, який буде здійснювати обчислення, та здатний витримувати великі навантаження. Це дозволить централізувати ядро обчислень для того, щоб уникнути проблем з підримкою багатоверсійності бази даних та шару API.

ПЕРЕЛІК ПОСИЛАНЬ

1.Канонічні діаграми мови UML — Студопедія [Електронний ресурс] – 2018 – Режим доступу: https://studopedia.com.ua/1_42750_kanonichni- diagrami-movi-UML.html

2. Rumbaugh K. The unified modeling language reference manual [Text] / J. Rumbaugh, I. Jacobson, G. Booch – Boston. : Addison Wesley Longman Inc., 1999 – p 26. - ISBN 0-201-30998-X.

3.http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?inpopup=true&id=44416

4.Stellman, Andrew; Greene, Jennifer (2005). Applied Software Project Management. O'Reilly Media. ISBN 978-0-596-00948-9

5. ДСТУ 2844-94 Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення

6. Rapid Application Development Model - RAD Model [Електронний ресурс]. – Режим доступу: https://www.testingexcellence.com/rapid- application-development-rad/

7. Martin, James (1991). Rapid Application Development. Macmillan. ISBN 0- 02-376775-8.

8. 01м - Моделі і методи проектування інформаційних систем Проектування інформаційних систем Тема 1 - Проектування ІС [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151203140326/165292/index.html

9. Beck, Kent (2000). Extreme Programming Explained. Addison Wesley. pp. 3–7. ISBN 0201616416.

10. Об’єктно-орієнтована парадигма

[Електронний ресурс]. – Режим доступу: https://stud.com.ua/97382/informatika/obyektno_oriyentovana_paradigma

11. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual. Addison Wesley Longman Inc. ISBN 0-201- 30998-X.

12. Rumbaugh K. The unified modeling language reference manual [Text] / J. Rumbaugh, I. Jacobson, G. Booch – Boston. : Addison Wesley Longman Inc., 1999 – p 228. - ISBN 0-201-30998-X.

13. Unified Modeling Language Superstructure Specification, version 2.1.1. Document formal/2007-02-05, Object Management Group, February 2007.

14. ISO/IEC 2382:2015, Information technology — Vocabulary — Part 1: Terms and definitions: «database: collection of data organized according to a conceptual structure describing the characteristics of these data and the relationships among their corresponding entities, supporting one or more application areas»

15.Distributed Application Architecture [Електронний ресурс] – 2006 – Режим доступу: http://java.sun.com/developer/Books/jdbc/ch07.pdf.

16.Software deployment [Електронний ресурс] – 2018 – Режим доступу: https://goo.gl/VfNcvg

17. Eurostat - Tables, Graphs and Maps Interface (TGM) table [Електронний ресурс] – 2018 – Режим доступу: https://ec.europa.eu/eurostat/tgm/table.do?tab=table&init=1&plugin=1&language=en&pcode=tec00120

18. ECMAScript Language Overview (PDF). 2007-10-23. с. 4. Процитовано 2009-05-03.

19. About npm [Електронний ресурс]. – Режим доступу: https://docs.npmjs.com/about-npm/

20. Використовуємо VS Code для Веб-розробки [Електронний ресурс]. – Режим доступу: http://it-ua.info/news/2016/02/09/vikoristovumo-vs-code- dlya-veb-rozrobki.html

21. ISO/IEC 2382:2015, Information technology — Vocabulary — Part 1: Terms and definitions: «database: collection of data organized according to a conceptual structure describing the characteristics of these data and the relationships among their corresponding entities, supporting one or more application areas»

22. MongoDB [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/

23. Stellman, Andrew; Greene, Jennifer (2005). Applied Software Project Management. O'Reilly Media. ISBN 978-0-596-00948-9.

24. 01м - Моделі і методи проектування інформаційних системПроектування інформаційних систем Тема 1 - Проектування ІС [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151203140326/165292/index.html

25. Rapid Application Development Model - RAD Model [Електронний ресурс]. – Режим доступу: https://www.testingexcellence.com/rapid- application-development-rad/

26. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York, et al: John Wiley and Sons, Inc. c. 480. ISBN 0- 471-35846-0.

27. Лайза Криспин, Джанет Грегори Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — M. : «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series). — 1000 прим. — ISBN 978- 5-8459-1625-9.

28. Unit Testing Guidelines from GeoSoft [Electronic resource] – Mode of access: http://geosoft.no/development/unittesting.html.

29. Martyn A Ould & Charles Unwin (ed), Testing in Software Development, BCS (1986), p71. Accessed 31 Oct 2014

30. Mocha testing [Електронний ресурс]. - https://github.com/mochajs/mocha

31. Закон України «Про авторське право і суміжні права» від 23.12.1993

№ 3792-XII. – [Електронний ресурс]. – Режим доступу: http://portal.rada.gov.ua/5.6.12

32. Методичні рекомендації по виконанню розділу техніко-економічного обґрунтування дипломних робіт студентами технічних спеціальностей напряму підготовки 8.05010302 «Інженерія програмного забезпечення» освітньо-кваліфікаційного рівня «Магістр» / Укладачі: Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

33. Закон України «Про оподаткування прибутку підприємств» від 28.12.1994 № 334/94-ВР. – [Електронний ресурс]. – Режим доступу: http://portal.rada.gov.ua/6

34. Закон України «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» від 28 грудня 2014 року № 71-VIII. – [Електронний ресурс]. – Режим доступу: http://portal.rada.gov.ua/

35. Закон України «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» від 28 грудня 2014 року № 71-VIII. – [Електронний ресурс]. – Режим доступу: http://portal.rada.gov.ua/

36. Закон України «Про міський електричний транспорт» від 29 червня 2004 року № 1914 – IY. – [Електронний ресурс]. – Режим доступу: http://portal.rada.gov.ua/

37. Охорона праці [Електронний ресурс]. – Режим доступу: http://www.geum.ru/next/art-203058.leaf-3.php

38. Охорона праці [Електронний ресурс]. – Режим доступу: http://www.geum.ru/next/art-203058.leaf-3.php

39. Методичні вказівки до виконання магістерської роботи освітнього рівня “магістр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид- во ТНТУ імені Івана Пулюя, 2016 – 26 с.

40. Comparing React Native to Cordova | Toptal [Електронний ресурс].– Режим доступу: https://www.toptal.com/mobile/comparing-react-native-to- cordova

Додаток А – Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Факультет інформаційних систем

Кафедра “ Програмної інженерії ”

ЗАТВЕРДЖУЮ
Завідувач кафедрою
програмної інженерії
“___”_____2019р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання проекту

«Розробка інформаційної веб платформи на основі технології C#
для автоматизації документообігу на виробництві»

Керівник роботи: док. техн.
наук, проф. Ясній О.П.
“___”_____2019р.

_____

Виконавець: студент групи
СПм-62 Гілюта М.І.
“___”_____2019р.

_____

Тернопіль 2019

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019-2020 н. р.

Тема проекту: «Розробка інформаційної веб платформи на основі технології C# для автоматизації документообігу на виробництві».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Предметом розробки є автоматизована система логістичних перевезень із аналізом локального ресурсу для операційних систем типу Windows, для створення якої було використано серверний фреймворк Meteor, об’єктно- орієнтовану мову програмування JavaScript та C#. В якості СКБД для даної системи було обрано MongoDB.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ 4

4.1 Функціональні характеристики

Програмний продукт має забезпечити можливість наступних операцій:

– користувач – має можливість створювати, редагувати, видаляти проект. Крім того він може переглядати, видаляти та реєструвати нові маршрути. Також повинен мати можливість завантажувати вхідні і вихідні дані.

4.2 Технічні вимоги

Система повинна бути написана мовою програмування Javascript. В якості СКБД використати MongoDB. В якості системи контролю версій використовувати Git/Bitbucket. Середовище розробки Microsoft VSCode.

Функціональні вимоги:

Користувачі системи повинні мати змогу виконувати наступні функції:

- реєстрація та авторизація для користувача;
- створення, редагування, видалення проекту;
- додавання або видалення категорії витрат;
- можливіть завантаження вхідних і вихідних даних.
- вхідна інформація отримується шляхом:
- введення інформації користувачами.

Вихідна інформація:

- виводиться у вигляді графіків;
- подається у форматі PDF з можливістю завантаження

Система повинна працювати на пристроях на базі операційних систем типу Windows

Мінімальні вимоги для використання: оперативна пам'ять від 1024 Мб, об’єм пам’яті па накопичувачі: від 100 МБ; дисплей: 1280 х 1024.

4.3 Програмна сполучність

Система повинна: коректно функціонувати під керуванням Windows 10.

4. СТАДІЇ РОЗРОБКИ

– аналіз предметної галузі;

– вибір інструментів;

– розробка прикладного програмного забезпечення;

– тестування програмного продукту;

– оформлення супровідної документації;

– здача продукту.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

– технічне завдання;

– пояснювальна записка.

Лістинг В.1 – компонент створення проекта

```
import {Template} from 'meteor/templating';
import {Meteor} from 'meteor/meteor';
import Trajets from '../../imports/trajets';

Template.trajet.events({
  'click #avancement'() {
    Meteor.call('dateFin', this._id, (err, dateFin) => {
      if (err) {
        document.getElementById('reponse').innerHTML = err;
      } else if (dateFin < Date.now()) {
        document.getElementById('reponse').innerHTML = 'trajet terminé';
        Meteor.call('destinationFinale', this._id, (err, arrivee) => {
          if (err) {
            document.getElementById('reponse').innerHTML = err;
          }
          Meteor.call('idCamion', this._id, (err, idCamion) => {
            if (err) {
              document.getElementById('reponse').innerHTML = err;
            }
            Meteor.call('changerLocalisationCamion', idCamion, arrivee);
            Meteor.call('activerCamion', idCamion);
            Meteor.call('terminerTrajet', this._id);
          });
        });
      } else {
        const ms = dateFin - Date.now();
        const s = ms / 1000;
        const min = s / 60;
        const min2 = min % 60;
        const sec = s % 60;
        const h = min / 60;
        document.getElementById('reponse').innerHTML = ('il reste ' +
        Math.floor(h) + ' heure(s) ' + Math.floor(min2) + ' minute(s) ' +
        Math.floor(sec) + ' seconde(s) de trajet');
      }
    });
  }
});
```

```
const interval = 5 * 60 * 1000; // 5 minutes
Meteor.setInterval(() => {
  console.log('Verification toutes les 5 minutes');
  let i;
  for (i = 0; i < Trajets.find().fetch().length; i++) {
    const idTrajet = Trajets.find().fetch()[i]._id;
    Meteor.call('dateFin', idTrajet, (err, dateFin) => {
      if (err) {
        console.log(err);
      } else if (dateFin < Date.now()) {
        document.getElementById('reponse').innerHTML = 'trajet terminé';
        Meteor.call('destinationFinale', idTrajet, (err, arrivee) => {
          if (err) {
            console.log(err);
          }
          Meteor.call('idCamion', idTrajet, (err, idCamion) => {
            if (err) {
              document.getElementById('reponse').innerHTML = err;
            }
            Meteor.call('changerLocalisationCamion', idCamion, arrivee);
            Meteor.call('activerCamion', idCamion);
            Meteor.call('terminerTrajet', idTrajet);
          });
        });
      } else {
        const ms = dateFin - Date.now();
        const s = ms / 1000;
        const min = s / 60;
        const min2 = min % 60;
        const sec = s % 60;
        const h = min / 60;
        document.getElementById('reponse').innerHTML = ('il reste ' +
        Math.floor(h) + ' heure(s) ' + Math.floor(min2) + ' minute(s) ' +
        Math.floor(sec) + ' seconde(s) de trajet');
      }
    });
  }
}, interval);
```

Лістинг В.2 – компонент детальної інформації про транспорт

```
import {Template}
from
'meteor/templating';
                        import {Meteor} from 'meteor/meteor';


                        Template.camion.events({
                          'click #delete'() {
                            Meteor.call('disponibiliteCamion', this._id, (err, res) => {
                              if (err) {
                                console.log(err);
                              }
                              if (res === true) {
                                Meteor.call('supprimerCamion', this._id);
                              } else {
                                console.log('impossible de supprimer un camion non disponible');
                              }
                            });
                          }
                        });


                        Template.demanderTrajet.events({
                          'submit form'(event) {
                            event.preventDefault();
                            const origine = event.target.Depart.value;
                            const destination = event.target.Destination.value;
                            Meteor.call('disponibiliteCamion', this._id, (err, res) => {
                              if (err) {
                                console.log(err);
                              }
                              if (res === true) {
                                if (origine === destination) {
                                  console.log('depart et destination identiques');
                                } else {
                                  Meteor.call('demandeTrajet', this._id, origine, destination);
                                  Meteor.call('changerLocalisationCamion', this._id, 'Trajet en
                        cours');
                                }
                              } else {
                                console.log('flotte non disponible');
                              }
```

```
    });
  }
});
```

Додаток В - CD Диск з програмним кодом

УДК 004.422.83

М.І. Гілюта ст. гр. СПм-62

О.П. Ясній д-р. техн. наук, проф. каф. мат. методів в інженерії

Тернопільський національний технічний університет імені Івана Пулюя

Проблеми розвитку вантажної логістики та впровадження інформаційних технологій у галузі

Становлення логістики в Україні характеризується прогресивною динамікою, про що свідчить національний індекс її ефективності (Logistics Performance Index – LPI) . Цей показник вперше було визначено у 2007 році. У 2015 році уже вп’яте проведено моніторинг тенденцій і ступеня розвитку логістики серед 160 країн світу, де Україна посіла 80 місце (2,74 бали), тоді як у 2010 році вона була 102-ою у рейтингу (2,57 бали) [1]. Великий попит породжує значну кількість пропозицій на ринку вантажної логістики , з розвитком самого ринку в різних країнах не менш важливою є сфера суміжна і це є сфера інформаційних технологій. Логістика фінансує близько 15% надходжень до бюджету від виробничої сфери і забезпечує близько 40% вітчизняного ринку послуг [1]. Тому постає питання актуального та своєчасного розвитку галузі інформаційних технологій з урахуванням логістичних потреб галузі. Сфера вантажної логістики є чи не найпотужнішою з усіх доступних видів перевезення і займає левову долю ринку. Такий складний та розвинений ринок потребує виважених і стабільних програмних рішень, які змогли б виконувати функції з обробки маршрутних даних і слугувати надійною опорою для усіх учасників ринку. На сьогодні з представлених в галузі програмних продуктів є багато неякісного і застарілого програмного забезпечення, як за технічними так і часовими параметрами, тому актуальним буде проект розробки сучасного

програмного продукту з урахуванням усіх складних і часто унікальних запитів логістичних фірм, який зможе доповнити галузь , зробивши її більш ефективною , безпечною та прибутковою. Новий програмний продукт має мати такі якості як наочність , мобільність і стабільність у виконані поставлених завдань. З можливістю доповнення маршрутних карт і визначення найбільш оптимальних шляхів вантажоперевезень.
Метою роботи є створення програмного продукту який зможе вирішити проблеми неякісного та застарілого устаткування на фірмах логістичного спрямування і підвищить ефективність їх роботи і задоволення від своєї діяльності . В подальшому це може бути виражено у накопичені значної кількості позитивних змін в економіці країни та людей.

Список використаної літератури:

1. Logistics Performance Index. Website of World Bank. URL: https://lpi.worldbank.org/report
2. Pro-Consulting Аналітика ринків. Фінансовий консалтинг. URL: http://pro-consulting.ua/