МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) *Автоматизації і комп'ютерних систем*
Кафедра *Інформаційних систем*

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ ____*Форсюк А.В.*_____	_____ _____*Чумаченко С.М.*
(підпис) (прізвище та ініціали)	(підпис) (прізвище та ініціали)
«___» _____ 20__р.	«___» _____ 20__р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності____*122 «Комп'ютерні науки»*____
(код та назва спеціальності)
освітньо-професійної програми____*«Комп'ютерні науки»*_____
на тему: *Розроблення ВЕБ-додатку для проведення рекламних кампаній ТОВ «Пирятинський сирзавод»*

Виконав: здобувач__4_ курсу, групи_4___

_____*Янович Віктор Віталійович*_____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник __*Ліманська Наталія Володимирівна*____ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти___ _____ _____
(прізвище та ініціали) (підпис)

_____ _____
(прізвище та ініціали) (підпис)

_____ _____
(прізвище та ініціали) (підпис)

Рецензент _____ _____
(прізвище та ініціали) (підпис)

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.

Здобувач_____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _____*Автоматизація і комп'ютерних систем* _____
Кафедра _____*Інформаційних систем*_____
Освітній ступінь_____*Бакалавр*_____
Спеціальність_____122__*«Комп'ютерні науки»*
(код і назва)
Освітньо-професійна програма _____*«Комп'ютерні науки»*_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних систем

_____С.М. Чумаченко

"____" _____20___року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____*Янович Віктор Віталійович*_____
(прізвище, ім'я, по батькові)

1. Тема роботи «*Розроблення ВЕБ-додатку для проведення рекламних кампаній ТОВ «Пирятинський сирзавод»*»
керівник роботи_*Ліманська Наталя Володимирівна*, *к.т.н.*_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від №*248-кс* "*29*" *квітня 2021року*
2. Строк подання здобувачем роботи___*31.05.2021 року*_____
3. Вихідні дані до роботи___*ВЕБ-додаток для проведення рекламних кампаній*
_____
_____
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)__ *постановка задачі, логічна та фізична модель бази даних, база даних, опис процесу розроблення, опис реалізованих функцій, інструкція користувача, висновки* _____
5. Перелік графічного матеріалу
__ *функціональні моделі «AS-IS» та «TO-BE», логічної та фізичної моделі бази даних, згенерована модель бази даних, знімки екрану WEB додатку, фрагменти коду*_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Лiманська Н.В		
2	Лiманська Н.В		
3	Лiманська Н.В		
4	Лiманська Н.В		

7. Дата видачі завдання_____*25 березня 2021року*_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз процесу	05.03-07.03	Виконано
2	Розробка технічного завдання	08.03-11.03	Виконано
3	Реалізація задач автоматизації процесу вивчення технічних термінів	12.04-19.04	Виконано
4	Охорона праці	20.05-23.05	Виконано
5	Оформлення пояснювальної записки та створення презентації	24.05-28.05	Виконано

Здобувач _____ ____*Янович В.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ ____*Лiманська Н.В.*
(підпис) (прізвище та ініціали

АНОТАЦІЯ

Головною метою бакалаврської роботи є Розроблення web-додатку для проведення рекламної кампанії ТОВ «Пирятинський сирзавод».

Для реалізації WEB додатку розроблений макет сайту, в програмі Figma. Та реалізаія структури бази даних.

Об’єктом дослідження є запити для проведення рекламної кампанії.

Предметом дослідження є Web-додаток, який допомагає користувачам зареєструватися та замовляти послуги які надає додаток.

КЛЮЧОВІ СЛОВА: WEB-ДОДАТОК, АДМІНІСТРАТОР, БІЗНЕС-ЛОГІКА, ПОСЛУГИ, ТОВАР, ЗАМОВНИК, БД.

ANNOTATION

I am heading bachelor robotics є Development of web materials for the advertising campaign of Piryatinsky Sirzavod LLC.

For the implementation of WEB, the addition of the breakdowns, the layout of the site, in the Figma program. That implementation of the structure of the base danih.

Ob'kom research є written for an advertising campaign.

The subject of the doslidzhennya є Web-dodatok, which additional help koristuvacham zare зastruvatisya and replace the servants yaki nadaє dodatok.

KEYWORDS: WEB-DODATOK, ADMINISTRATOR, BIZNES-LOGIKA, SERVICES, PRODUCT, ZAMOVNIK, DB.

Зміст

ВСТУП

Web-додатки(сайти мобільні програми і тд.) відіграють дуже важливу роль в нашому повсякденному житті. Якщо для прикладу взяти рік 1990 не було смартфонів, обновлених комп'ютерів, щоб замовити рекламу потрібно було йти в рекламний центр залишати заяву на рекламу і очікувати відповідь, так же для купівлю якогось товару.

Зараз це стало набагато простіше, для цього потрібно лише смартфон або ПК і 5 хвилин вашого часу, щоб обробити замовлення в моєму web-додатку. Даний додаток може надати змогу замовити, рекламу свого товару або рекламу «Пирятинського сирзаводу» для кого і був зроблений цей web-додаток. Також на головній сторінці сайту, ви може побачити, акційні товари «Пирятинського сирзаводу». Також під мій web-додаток була організована CRUD система для управління користувачами. Яка надає змогу зареєструватися, залишити дані по продукту або залишити ваше замовлення. Обробляє ці замовлення адміністратор корегує їх та обробляє замовлення користувача.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ WEB ДОДАДКУ

1.1. Поняття веб-додатку, їх переваги, автоматизація інформатизація.

Користувачам ПК відомо, що таке додаток ОС Windows. Це програма, яка встановлюється на комп'ютер і працює в операційному середовищі ОС Windows. Текстові та графічні редактори, медіаплейери й поштові клієнти тощо.

Веб-додаток – розподілений додаток, в якому клієнтом виступає браузер, а сервером – веб-сервер .

Веб-додатки – це програми, написані скриптовою мовою (React, Node js, Javascript та ін.) або написані мовою високого рівня та відкомпільовані під відповідну ОС (C, C++ та ін.), які працюють на стороні веб-сервера та призначені для створення інтерфейсу між користувачем та веб-сайтом.

Отже, **веб-додаток** – це комп’ютерна програма, яка працює в браузері, як Microsoft Word працює в ОС Windows. Тому, для доступу до програми потрібні браузер та Інтернет. Зберігання та обробка інформації при такій організації обчислень відбувається на віддаленому сервері, а веб-переглядач служить програмою-клієнтом і призначеним для користувача інтерфейсом (рис. 1.1) [8].

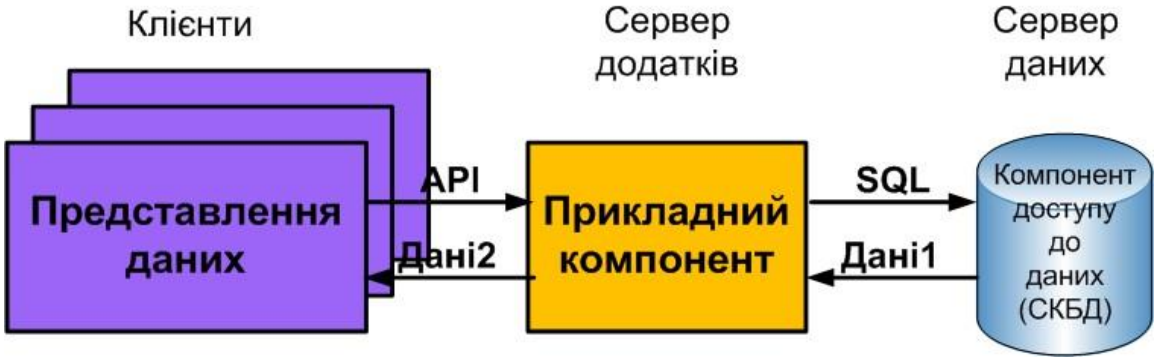


Рис.1.1. Схема роботи веб-додатку

Текстовий пакет Microsoft Office буде працювати тільки під Windows. А ось для веб-додатка операційна система встановлена на комп’ютері не має значення. Тому, що і операційною системою, і користувацьким інтерфейсом веб-додатка є браузер.

В результаті такої універсальності, постійний користувач веб-додатка може абсолютно без проблем працювати зі своєю улюбленою програмою на будь-якому зі своїх девайсів, починаючи з офісного стаціонарного комп'ютера, закінчуючи планшетом і смартфоном.

У прикладному відношенні – веб-додатки мають ту істотну перевагу: вже розроблено багато програм і сервісів, за допомогою яких будь-яка людина, не будучи програмістом і навіть просунутим користувачем, може створювати різні корисні програми для своєї зручності та розваги. Причому абсолютно безкоштовно.

Наприклад, всім відомий Gmail є повноцінним поштовим клієнтом, який робить все, що робить будь-який інший поштовий клієнт, запускається на вашому комп’ютері, і навіть трохи більше, має безліч додаткових функцій. Bloglines – веб-додаток для читання новин, яке безпосередньо конкурує із звичайними аналогічними застосуваннями, конкурує і виграє. Ці веб-додатки працюють на сервері, а їх користувацький інтерфейс (UI) відображається у вигляді веб-сторінок.

Хмарні технології

Хмарні технології – це парадигма, що передбачає віддалену обробку та зберігання даних [6].

Хмара – це деякий ЦОД (дата-центр, сервер) або їх мережа, де зберігаються дані та програми, що з’єднуються з користувачами через Інтернет (рис. 1.2).

Рис. 1.2. Дата центр

Хмарні технології дозволяють споживачам використовувати програми без установки їх на робочу машину і надають доступ до особистих файлів з будьякого комп'ютера, що має доступ в Інтернет. Ця технологія дозволяє значно ефективніше керувати, обробляти та працювати з інформацією за рахунок централізації цієї самої інформації.

Простим прикладом хмарних технологій є сервіси електронної пошти, наприклад, Gmail, Meta і т.д. Вам потрібно всього лише підключення до Інтернету, і Ви зможете відправити пошту, при цьому додаткового програмного забезпечення або сервера не потребуючи.

Хмарні технології являють собою загальний термін для всього, що включає в себе постачання послуги хостингу через Інтернет. Ці послуги, в цілому, можна поділити на три категорії:

- Програмне забезпечення як послуга (SaaS).

По моделі SaaS постачається апаратна інфраструктура і ПЗ, також розробник забезпечує взаємодію з користувачем через інтерфейсний портал. SaaS на даний момент є досить широко розповсюдженим. За SaaS можуть надаватись самі різноманітні послуги, від веб-пошти до управління

запасами, обробки БД. Перевагою такої моделі є те, що кінцевий користувач може вільно користуватись послугою з будь-якої точки світу.

• Платформа-як-сервіс (PaaS).

PaaS в хмарі визначається як набір програмних продуктів та засобів розробки, що розміщені на інфраструктурі провайдера. Розробники можуть створювати програми на платформі провайдера через Інтернет. PaaS провайдери можуть використовувати API, сайт-портали, шлюзи, або програмне забезпечення установлене на комп'ютері клієнта.

- Інфраструктура як послуга (IaaS).

IaaS являє собою віртуальний сервер instanceAPI для запуску, зупинки, доступу, налаштування своїх віртуальних серверів та систем збереження. IaaS дозволяє компанії платити саме за стільки потужностей, скільки їй необхідно.

Дану модель іноді називають «комунальні обчислення» [5].

На основі цього можна зробити висновки: веб-додаток є актуальним в наш час, це технологія, яка стрімко розвивається і покликана економити час та ресурси на створенні універсальних додатків (програм), які в свою чергу дають можливість працювати в будь-якому місці, у будь-який час та з будь-яких девайсів не залежно від операційної системи чи технічних характеристик.

1.2. Опис та особливості вивчення автоматизації роботи веб-додатків та прикладного програмного забезпечення

Прикладне програмне забезпечення

В основу роботи комп'ютерів покладено програмний принцип керування, який полягає в тому, що комп'ютер виконує дії за заздалегідь заданою програмою. Цей принцип забезпечує універсальність використання комп'ютера: у певний момент часу розв'язується задача відповідно до вибраної програми. Після її завершення у пам'ять

завантажується інша програма і т.д. Програма – це запис алгоритму розв'язання задачі у вигляді послідовності команд або операторів мовою, яку розуміє комп'ютер. Кінцевою метою будь-якої комп'ютерної програми є керування апаратними засобами [9].

Для нормального розв'язання задач на комп'ютері потрібно, щоб програма була налагоджена, не потребувала дороблень і мала відповідну документацію. Тому стосовно роботи на комп'ютері часто використовують термін програмне забезпечення (software), під яким розуміють сукупність програм, процедур і правил, а також документації, що стосуються функціонування системи оброблення даних.

Програмне та апаратне забезпечення у комп'ютері працюють у нерозривному зв'язку та взаємодії. Склад програмного забезпечення обчислювальної системи називається програмною конфігурацією. Між програмами існує взаємозв'язок, тобто багато програм працюють, базуючись на програмах нижчого рівня. Міжпрограмний інтерфейс – це розподіл програмного забезпечення на декілька пов'язаних між собою рівнів. Рівні програмного забезпечення являють собою піраміду, де кожен вищій рівень базується на програмному забезпеченні попередніх рівнів (рис. 1.3).

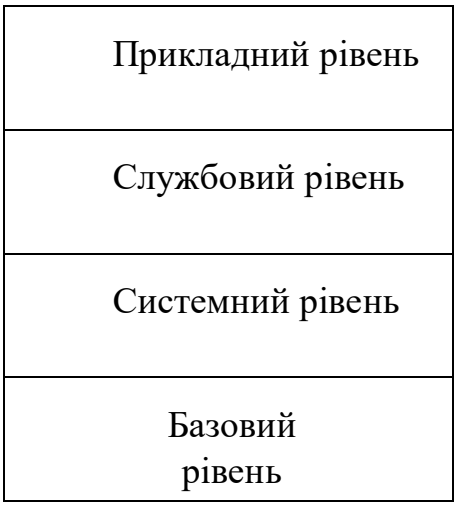


Рис.1.3 Структура програмного забезпечення

Базовий рівень

Цей рівень є найнижчим рівнем програмного забезпечення. Відповідає за взаємодію з базовими апаратними засобами. Базове програмне забезпечення міститься у складі базового апаратного забезпечення і зберігається у спеціальних мікросхемах постійного запам'ятовуючого пристрою (ПЗП), утворюючи базову систему введення-виведення BIOS. Програми та дані записуються у ПЗП на етапі виробництва і не можуть бути змінені в процесі експлуатації.

Системний рівень

Системний рівень – є перехідним. Програми цього рівня забезпечують взаємодію інших програм комп'ютера з програмами базового рівня і безпосередньо з апаратним забезпеченням. Від програм цього рівня залежать експлуатаційні показники всієї обчислювальної системи. При під'єднанні до комп'ютера нового обладнання, на системному рівні повинна бути встановлена програма, що забезпечує для решти програм взаємозв'язок із цим пристроєм. Конкретні програми, призначені для взаємодії з конкретними пристроями, називають драйверами.

Інший клас програм системного рівня відповідає за взаємодію з користувачем. Завдяки йому є можливість вводити дані у обчислювальну систему, керувати її роботою й отримувати результат у зручній формі. Це засоби забезпечення користувацького інтерфейсу, від них залежить зручність та продуктивність роботи з комп'ютером.

Сукупність програмного забезпечення системного рівня утворює ядро операційної системи комп'ютера. Наявність ядра операційної системи – є першою умовою для можливості практичної роботи користувача з обчислювальною системою. Ядро операційної системи виконує такі

функції: керування пам'яттю, процесами введення-виведення, файловою системою, організація взаємодії та диспетчеризація процесів, облік використання ресурсів, оброблення команд і т.д.

Службовий рівень

Програми цього рівня взаємодіють як із програмами базового рівня, так і з програмами системного рівня. Призначення службових програм (утиліт) полягає у автоматизації робіт по перевірці та налаштуванню комп'ютерної системи, а також для покращення функцій системних програм. Деякі службові програми (програми обслуговування) відразу додають до складу операційної системи, доповнюючи її ядро, але більшість є зовнішніми програмами і розширюють функції операційної системи. Тобто, у розробці службових програм відслідковуються два напрямки: інтеграція з операційною системою та автономне функціонування.

Технологія веб-додатка

По самій структурі додатку, вся програмна логіка сконцентрована в одному місці (на сервері), а користувацький інтерфейс доступний будь-якій людині у вигляді невеликої програми, яка існує з самого моменту зародження HTTP-мережі (мова йде про браузер). Для появи веб-додатків не вистачало тільки таких важливих технологій, як JavaScript і DOM.

Використання JavaScript у веб-додатках дуже сильно відрізняється від того, як він використовувався в 1999 р. Тоді технологія JavaScript була тільки класним способом змусити елементи сторінки реагувати на рух миші (що зараз всіх дратує), створювати сліди курсору на екрані, блимати текстом, міняти кольори. JavaScript-ом користувалися просто тому, що він був, а не тому, що він був необхідний. На якийсь момент почало здаватись JavaScript може зникнути. Використання JavaScript на сторінках стало поганим тоном у вебдизайні, після чого мову вже всі готові були списати з рахунків.

Сьогодні ж JavaScript – це зріла повноцінна мова. Зараз вона використовується саме для того, для чого і задумувалася, для виконання дій на стороні клієнта без зайвого звернення до віддаленого серверу.

Динамічна складова веб-сторінок завжди була слабкою частиною при створенні веб-додатків, і мабуть це єдина причина, чому веб-додатки тільки сьогодні стали набувати популярності. Сьогоднішній JavaScript повністю вирішує цю проблему. Тепер, зупинимось на питанні, навіщо створювати вебдодатки.

Отже, вище ми з'ясували, що вся програмна логіка програми знаходиться на сервері, на відміну від звичайного ПЗ, де логіка програми розташовується на комп'ютері кожного користувача. Так як є тільки одна робоча копія додатку, його набагато простіше поширювати серед користувачів. По суті про старий способі поширення додатка взагалі можна забути, оскільки користувач в реальності не отримує копії додатку, як раніше. Все, що одержує користувач, це інтерфейс програми (UI), тобто тільки те, що йому необхідно для роботи. Проблеми поширення веб-додатку не існує, оскільки отримати ви його можете в будь-який момент в будь-якому місці.

Для роботи користувачеві потрібна тільки одна програма і це браузер, набравши URL можете приступати до роботи.

Як правило, якщо користувач встановлює на своїй машині програму, йому доводиться брати на себе роль його адміністратора. Йому треба встановлювати програму, запускати, настроювати, лагодити, вирішувати проблеми, що виникають.

У випадку ж з веб-додатком, так як він розташовується на сервері, користувачеві немає необхідності турбуватися про це, на відміну від звичайного додатку. У ролі адміністратора виступає розробник додатку. Це додаткове навантаження на програміста, але якщо порівняти вартість

створення веб-додатку з вартістю утримання команди фахівців, які займаються установкою, підтримкою і налагодженням звичайних додатків на машинах користувачів, ви відразу ж побачите, що буде дешевше, не кажучи в ж про ефективність. З точки зору бізнесу набагато вигідно мати невелику команду програмістів, що працює в одному місці над одним додатком.

Додаток не вимагає нічого від користувача. В дійсності передбачається, що у користувача є браузер, який зможе працювати з додатком. Веб-додаток не пред’являє ніяких вимог до апаратної платформи. Це означає, що користувач зможе працювати з додатком на будь-якій операційній системі і не помітить ніякої різниці. Проблема підтримки різних версій у минулому. Як тільки виходить нова версія веб-додатку, всі без винятку користувачі її отримують буквально негайно, достатньо просто оновити сторінку в браузері. Знову ж таки, оскільки додаток знаходиться на сервері, існує тільки одна його копія у світі. Всі старі версії миттєво зникають, а користувач навіть не помічає, що у нього нова версія програми. Це також означає, що розробникам не треба піклуватися про підтримку старих версій програм і хвилюватися з приводу зворотної сумісності.

Стосовно використання пам’яті, користувачеві немає потреби завантажувати на свій комп’ютер весь додаток цілком, щоб почати з ним працювати. Навіть весь інтерфейс не обов'язково завантажувати. Досить завантажити тільки ту його частину, яка потрібна для виконання конкретної поточної задачі. Завдяки цьому веб-додатки невеликі за обсягом, швидко завантажуються і швидко відповідають на дії користувачів. Навіть найскладніший додаток завантажується всього за кілька секунд, і навіть менше, і то тільки, якщо канал зв’язку занадто повільний.

Так як на комп'ютері користувача нічого не встановлюється, користувач може працювати з додатком з будь-якого місця. «Будь-яке місце» буквально означає будь-яке місце на Землі.

1.3. Аналіз нинішнього стану web-додатків

Тема автоматизації обросла величезною кількістю стереотипів і міфів, які мають мало спільного з реальністю. Наприклад, що автоматизувати можна тільки роботу бухгалтера, що це необхідно тільки дуже великим підприємствам або що впровадження - дорогий процес, який не завжди окупається.

Реальна автоматизація веб додатків - це процес, при якому трудомістка ручна робота співробітників замінюється спеціальними програми. Таким чином, сучасні технології визволяють час персоналу для вирішення більш важливих завдань. У підприємництві зазвичай це необхідно у двох випадках:

- коли потрібно спростити рутинні процеси, без яких неможлива робота підприємства (звітність, бухгалтерський облік);
- коли потрібно автоматизувати процеси, спрямовані на розвиток замовлень та взаємодію з клієнтами(продажу, взаємодія з партнерами, постачальниками, клієнтами і т.д).

Виходячи з цього вже формуються цілі для автоматизації. Якщо в першому випадку це буде економія часу і ресурсів співробітників, а також скорочення помилок внаслідок людського фактора, то в другому - вплив на прибутковість бізнесу, поліпшення сервісу, залучення та утримання клієнтів і багато іншого. Сучасні програмні продукти для автоматизації здатні впоратися з будь-якою з перерахованих завдань. Важливим є те, що рішення підбирається індивідуально під запити конкретної компанії.

1.4. Розроблення функціональної моделі web-додатку

Для виявлення наявних недоліків у процесі автоматизації веб додатку потрібно провести аналіз процесу у такому вигляді, як він зараз існує. Для цього створюються функціональні моделі, а саме моделі AS-IS за допомогою методології IDEF0.

1.4.1 Опис методології IDEF0

Методологія IDEF0, особливості та прийоми застосування заснована на підході, розробленому Дугласом Т. Россом на початок 70-их років і отримала назву SADT (Structured Analysis & Design Technique - метод структурного аналізу і проєктування) [4]. Основи підходу і, як наслідок, методології IDEF0, складає графічна мова опису (моделювання) систем, що володіє наступними властивостями:

- Мова забезпечує точний і лаконічний опис модельованих об'єктів, зручність використання і інтерпретації цього опису;
- Мова полегшує взаємодію та взаєморозуміння системних аналітиків, розробників і персоналу досліджуваного об'єкта (фірми, підприємства), тобто служить засобом «інформаційного спілкування» великого числа фахівців і робочих груп, зайнятих в одному проєкті, в процесі обговорення, рецензування, критики та затвердження результатів;
- Мова пройшла багаторічну перевірку і продемонстрував працездатність як в проєктах ВПС США, так і в інших проєктах, що виконувалися державними та приватними промисловими компаніями;
- Мова легка і проста у вивченні і освоєнні.

Перераховані властивості мови зумовили вибір методології IDEF0 в якості базового засобу аналізу.

1.4.2 Опис функціональних схем (AS-IS)

Розроблена функціональна модель описує процеси вивчення фахової лексики іноземною мовою в тому вигляді, як вони здійснюються зараз (AS-IS). Контекстна діаграма наведена на рис.1 у додатку А.

Управління моделі:

- Правила та норми словників – це набір стандартів, по яким створюються та видаються словники;
- Правила правопису – регламентують орфографічне написання слів;

- План навчання – за допомогою плану навчання здійснюється організація навчального процесу.

Входи:

- Дані у паперовому вигляді – матеріали, з яких береться інформація для вивчення фахової лексики;
- Терміни – обрані технічні терміни з матеріалів для вивчення.

Виходи:

- Результат пошуку – впорядковані технічні терміни для вивчення фахової лексики.

Механізми:

- Користувач

Докомпонована модель наведена на рис.2 у додатку А.

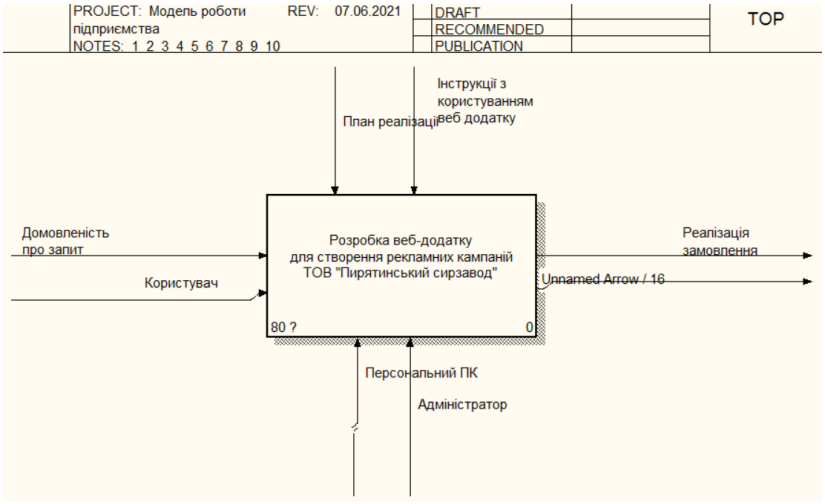
Дана модель являє собою трирівневу ієрархію упорядкованих і взаємозв'язаних діаграм, відображає існуючий порядок бізнес-процесів та має статус «AS-IS»– «як є».

На першому рівні декомпозиції діаграма складається з трьох блоків:

- Пошук матеріалів для вивчення
- Пошук термінів
- Упорядкування даних

Як можна побачити з наведених вище діаграм і їх описів, користувач (механізм) працює з паперовими матеріалами (вхід) для пошуку термінів для вивчення за певним планом навчання (управління).

Огляд процесів, що існують за допомогою моделі AS-IS здійснюється для того, щоб наочно побачити чи показати існуючі недоліки у процесі дослідження і чим не задовольняє нинішній рівень автоматизації.



1.5 Виявлені проблеми при дослідженні автоматизації web додатку

Автоматизація веб додатку може допомогти перенести рутинні завдання та їх облік у сервіси і додатки, аби зробити усі процеси більш прозорими та контрольованими. Якщо на підприємстві великий обсяг інформації, обробляти яку вручну (або наявними засобами автоматизації) стало неможливо (не встигаємо, помиляємося), то автоматизація — це вихід. Якщо ж потрібної інформації просто немає або вона недостовірна, рішення приймаються без урахування наявних даних, дії не узгоджені, то, швидше за все, не сформована сама система управління і автоматизація тут не допоможе. Тому, оцінивши ситуацію, потрібно зробити висновок, чи вирішить автоматизація виявлені проблеми.

Далі необхідно розібратися з бізнес-процесами, які протікають у вашій компанії. Щоб їх визначити і описати, досить відповісти на питання: “Що і для чого Ви робите всередині свого бізнесу, щоб він приносив прибуток?”

1.5.1 Виявлені недоліки

Отже, виявлено наступні проблеми:

- Оскільки більшість процесів проходить з використанням паперових матеріалів, уповільнюється пошук потрібних даних.
- Немає можливості швидко відсортувати, упорядкувати дані.
- Є велика ймовірність виникнення помилок при пошуку і упорядкуванню через людський фактор.

1.5.1 Задачі автоматизації web додатку

Для розв’язання виявлених проблем було вирішено автоматизувати процес замовлення реклам з використання web додатку.

Задачі для автоматизації:

- Зручний інтерфейс;
- Автоматичне упорядкування.
- Зручний вивід інформації.
- Можливість розміщення реклами.
- Можливість можливість отримання запитів в CRUD системі.
- Можливість слідкувати за користувачами.

1.6 Обґрунтування доцільності проектування й розроблення web додатку

Почнемо розгляд з визначення в вікіпедії "Web-додатки - клієнт-серверні додатки, в яких клієнтом виступають браузери, а сервером - веб-сервер на різних платформах. Логіка веб-додатків розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є кросплатформенними сервісами."

В визначенні ми бачимо, що логіка розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері , обмін інформацією відбувається по мережі.

Почнемо з того, як може бути розподілена логіка Web-додатки . Ми можемо на сервері зберігати тільки дані, а вся логіка роботи користувача буде зашита в користувальницький додаток, код якого може приходити з сервера якщо ми працюємо через браузер і може бути зашитий в додаток якщо ми працюємо з мобільного клієнта. А можемо віддавати користувачеві вже сформовану HTML сторінку з сервера і при кожній дії користувача її перебудовувати.

В сучасних рішеннях на сервері часто реалізується REST сервіс зберігання даних і частково бізнес алгоритмів. І до нього звертається браузерні додатки за тією чи іншою інформацією. Приклад такого рішення ви можете подивитися в нашому продукті CRM (Small Business) , який ми робили для нашого клієнта Select Sport. Там же в статті ви зможете перейти на демо версію.

Переважно всі дані зберігаються на сервері. Але можуть так само зберігатися локально як кеш браузер або в мобільному додатку якщо немає зв'язку або сервер з якоїсь причини недоступний.

1.6.1 Переваги Web-додатків

- Користувальницькі клієнти не залежать від обладнання і не вимагають налаштувань. Процес підключення нового користувача закінчується наданням йому посилання на додаток. І, в залежності від бізнес вимог, логіна і пароля, або в іншому випадку він може зареєструватися сам або додаток взагалі не вимагає користувальницької ідентифікації.
- Швидкість роботи - розроблені за такою архітектурою Web-додатки мають дуже високу продуктивність. Кількість одночасно працюючих користувачів може вимірюватися тисячами. Найчастіше готові десктопні рішення, а так само рішення із закладеною в свою архітектуру метамодель, мають складності з ростом користувачів і обробки даних.
- При створенні Web-додатки з нуля ви рятуєтеся від необхідності ліцензувати кожного клієнта окремо, а ноутбуки з Linux взагалі не зажадають витрат (крім самого заліза звичайно) на підключення нового робочого місця.

1.6.2 Недоліки Web-додатків

Як і для будь-якого рішення не можна оцінювати тільки його переваги, потрібно розуміти ризики, які пов'язані з прийняттям такого рішення.

При створенні Web-додатку з нуля ви частково прив'язується до компанії, розробника даного рішення. Перехід до іншого розробника буде пов'язаний з витратами на вивчення новим партнером архітектури рішення. Ну і звичайно ж багато часто чули: "Вони все зробили неправильно".

Часто, хоча не завжди, створення додатка з нуля коштує дорожче через необхідність створення типових об'єктів , які є в коробкових рішеннях. Іноді це виправдано, а іноді ні.

Висновок: Якщо брати до уваги то, що автоматизація web додатоку зменшить затрати часу на виконання тих чи інших замовлень, купівль товару. То я прийшов до висновку, що веб додаток є дуже зручним і надійним засобом надання послуг реклами.

Основними перевагами є можливість працювати як дома так і на іншому куточку землі.

1.7 Обґрунтування доцільності проєктування й розроблення системи для web додатку

Саме зараз люди почали користуватися web додатками саме для того, щоб звільнити від бумажної роботи, а всі дані оброблювати дистанційно за допомогою web додатків.

Із урахуванням цього вирішено створити систему, яка буде повністю задовольняти і виконувати поставлені задачі для автоматизації web додатку.

1.7.1 Концептуальна модель системи

Метою створюваної web-додатку для контролю та синхронізації інформації про послугу є надання швидкого доступу до всього товару та послуг який дає web-додаток, можливості перегляду акційних товарі та послуг реклами.

Знайдені в моделі AS-IS недоліки були виправлені та процеси, що існують, удосконалені при створенні нової CRUD системи та створення веб додатка.

Побудована система Crud процесу створення електронної системи вивчення, складається з таких процесів:

- Оформлення послуги.
- Опрацювання послуги.
- Виконання послуги.

Додаток заповнюється термінами, і має функції пошуку та збереження послуг які надаються, для яких входами є параметри пошуку і запити на пошук та збереження. Після виконання цих функцій, виводиться результат на екран.

1.8 Розрахунок економічного ефекту від впровадження системи

Інформаційна система підтримки діяльності служби адміністратора.

1. Ступінь новизни розроблюваних задач – "В" – використання типових проектних рішень за умови їх змін.

2. Група складності алгоритму – 2.

3. Узагальнені дані вхідної та вихідної інформації для служби головного механіка за видами вхідної та вихідної інформації табл. 5.1.

Таблиця 5.1. Узагальнені дані для вхідної та вихідної інформації служби адміністратора.

Вид інформації	П означення	К -сть наборів даних
Змінна інформація	ЗІ	m=4
Нормативно-довідкова інформація	НДІ	n=5
Банк (база) даних	БД	p=1

Обробка в режимі реального часу	РЧ	так
Забезпечення телекомунікаційної обробки даних і управління віддаленими об'єктами	ТОУ	ні

4. Витрати часу на систему призначену для підтримки діяльності служби адміністратора, а саме на розробку ескізного проекту (передпроектного дослідження) T_1 і технічного завдання T_2, будуть наступні (табл. 5.2).

Таблиця 5.2. Визначення витрат часу для служби адміністратора.

Вид системи	Стадія розробки системи	
	Передпроектне дослідження	Технічне завдання
	В	В
Управління технічною підготовкою виробництва.	T_1= 67	T_2 =31

5. Визначається базове значення витрат часу для стадій "Технічний проект" , "Робочий проект" і "Впровадження". Для цього використовуються дані табл. 5.3 - 5.5.

Таблиця 5.3. Технічний проект.

Кількість форм вхідної інформації	Кількість форм вихідної інформації				
	1	2	3 -4	5 -6	7 -9
1	30	36	43	49	56
2	42	50	59	68	75
3	49	61	72	83	93

4	57	71	94	96	108
5	64	79	91	104	118
6	70	85	100	114	128
7	75	90	107	123	138
8	79	97	115	130	145
9	84	103	120	137	155
10	88	109	126	145	163

Таблиця 5.4. Робочий проект.

Кількість форм вхідної інформації	Кількість форм вихідної інформації				
	1	2	3 -4	5 -6	7 -9
1	44	58	76	92	110
2	59	81	103	128	150
3	71	98	125	157	182

4	82	111	143	176	208
5	90	122	159	195	231
6	97	134	174	211	251
7	104	144	186	227	269
8	110	153	197	272	286
9	116	163	208	255	302
10	123	172	219	268	318

Таблиця 5.5. Впровадження.

Кількість форм вхідної інформації	Кількість форм вихідної інформації				
	1	2	3 -4	5 -6	7 -9
1	17	22	29	37	45
2	22	31	40	51	61
3	27	38	50	62	75
4	32	44	57	71	86
5	35	48	64	80	96

6	36	53	70	87	105
7	41	58	75	94	112
8	43	62	81	100	120
9	45	66	85	106	126
10	47	68	90	112	134

Вхідними даними для визначення є:

- кількість форм вхідної інформації $B_1 = 8$,

- кількість форм вихідної інформації $B_2 = 4$,

- базове значення витрат часу для стадій "Технічний проект":

$T_{Б3} = 115$;

- базове значення витрат часу для стадій "Робочий проект":

$T_{Б4} = 197$;

- базове значення витрат часу для стадій "Впровадження":

$T_{Б5} = 81$.

Базове значення витрат часу $T_Б$ коригується за допомогою поправочних коефіцієнтів для всіх стадій розробки інформаційної системи.

- Розрахунок витрат часу для стадії "Технічний проект" ($T_З$).

Коефіцієнт трудомісткості робіт $k_п$ визначається за (5.1) і з врахуванням коефіцієнтів табл. 5.6:

$$k_П = \frac{k_1 * m + k_2 * n + k_3 * p}{m + n + p} \qquad (5.1)$$

$$k_{П} = \frac{1.0*8 + 0.72*4 + 2.08*1}{8+4+1} = 0.997$$

Таблиця 5.6. Коефіцієнти k_1, k_2, k_3 для стадії "Технічний проект".

Вид використаної інформаії	Ступінь новизни
	В
k_1 (ЗІ)	1.0
k_2 (НДІ)	0.72
k_3 (БД)	2.08

Коефіцієнт ступеню новизни проекту, k_O, що враховує вид обробки інформації для трьох стадій розробки системи визначається з табл. 5.7.

Таблиця 5.7. Коефіцієнт ступеню новизни проекту, k_O для служби адміністратора.

Стадія розробки системи	Вид обробки	Ступінь новизни
		В
Технічний проект	РЧ	1.26
Робочий проект	РЧ	1.32
Впровадження	РЧ	1.21

Витрати часу для стадії “технічний проект” $T_З$ розраховуються за (5.2):

$$T_З = \mathrm{T_{БЗ}} * \mathrm{k_П} * \mathrm{k_O} = 115 * 0.997 * 1.26 = 144{,}47$$

(5.2)

- Розрахунок витрат часу для стадії "Робочий проект" (T_4) служби адміністратора ТОВ «Пирятинського сирзаводу».

Для визначення витрат часу на стадії "робочий проект" використовують формулу (5.3), де $k_П$ – коефіцієнт, що враховує вид використаної інформації і визначається за формулою (5.2.):

$$T_i = T_{Бi} * k_П * k_О * k_С, \qquad (5.3)$$

$$k_П = \frac{1.1*8 + 0.58*4 + 0.48*1}{8+4+1} = 0.892$$

Таблиця 5.8. Коефіцієнти k_1, k_2, k_3 для стадії "Робочий проект".

Вид використаної інформації	Група складності алгоритму	Ступінь новизни
		В
k_1 (ЗІ)	2	1.1
k_2 (НДІ)	2	0.58
k_3 (БД)	2	0.48

Коефіцієнт, що враховує вид обробки інформації на стадії "Робочий проект" табл. 5.8. Коефіцієнт складності контролю вхідної та вихідної інформації і визначається з таблиці 5.9. на стадії "Робочий проект" і "Впровадження".

Таблиця 5.9. Коефіцієнт складності контролю вхідної та вихідної інформації $k_С$.

Складність контролю вхідної інформації	Складність контролю вихідної інформації

	21	22
11	1.16	1.07
12	1.08	1.00

Складність контролю вхідної та вихідної інформації характеризується:

11 – вхідні дані і документи різного формату і структури, контроль здійснюється перехресно, тобто враховується зв’язок між показниками різних документів;

12 – вхідні дані і документи подібної форми і змісту, тобто здійснюється формальний контроль;

21 – друк документів складної багаторівневої структури, різної форми та змісту;

22 – друк документів подібної форми та змісту, виведення масивів даних на машині носії.

$$k_C = 1.16$$

Витрати часу T_4 вимірюються в людино-днях, розраховується за (5.3).

$$T_4 = T_{Б4} * k_П * k_O * k_C = 197 * 0.875 * 1.32 * 1.16 = 263.94$$

Поправочні коефіцієнти мають такі ж значення, як і при обрахунку T_4 табл. 5.9 і табл. 5.7.,

Для стадії визначення загальних витрат часу на "Впровадження" T_5 (люд-днів) використовують формулу (5.3):

$$T_5 = T_{Б5} * k_П * k_O * k_C = 81 * 0.875 * 1.21 * 1.16 = 99.48$$

Таким чином, загальні витрати людської праці на проектування системи за (5.4) складають:

$$T_\Sigma = T_1 + T_2 + T_3 + T_4 + T_5 \qquad (5.4)$$

$$T_\Sigma = 67 + 31 + 144.47 + 263.94 + 99.48 = 605.89 \text{ (люд-дн).}$$

Для дипломного проекту кількість робочих годин складає 530 із 7-годинним робочим днем, тому на розробку проекту виділено Ф, днів:

$$\Ф = \frac{530}{7} = 75 \quad .$$

Для дипломного проекту Ф = 75днів. Тоді визначаємо кількість місяців із розрахунку 25 робочих днів.

Кількість місяців на розробку, М:

$$\text{М} = \frac{\Ф}{25} = \frac{75}{25} = 3$$

Отже, для виконання такого проекту потрібно таку чисельність виконавців $Ч$, виконавців, обраховується за (5.5):

$$Ч = \frac{\text{Т}_\Sigma}{\Ф} = \frac{732}{75} \approx 10 \qquad (5.5).$$

Якщо прийняти, оплата розробника відділу АСУ ТОВ «Пирятинського сирзаводу» здійснюється в розмірі 2800 грн, то оплата праці всіх виконавців, яка підраховується за формулою (5.6.) складе:

$$V'_1 = Ч * \text{М} * \text{ЗП}_{\text{ПР}}, \qquad (5.6)$$

$$V'_1 = Ч * \text{М} * \text{ЗП}_{\text{ПР}} = 10 * 3 * 2800 = 84000 \text{ грн.}$$

Витрати, пов'язані з розробкою програми на ПК.

1. Розрахунок річного фонду часу роботи ПК.

Дійсний річний фонд часу ПК у годинах дорівнює числу робочих годин у році для оператора, за вийнятком часу на технічне обслуговування і ремонт ПК (в середньому 5год/міс + 6 роб.днів/рік).

$$T_{ПК} = 2000 - (6*8 + 5*12) = 1892$$ год.

Оскільки під час виконання дипломного проекту (роботи) студент в середньому витрачає 450 год. машинного часу, то величина фонду часу ПК за (5.7) дорівнює

$$T'_{ПК} = T_{ПК} * \frac{R}{T_{ОП}} \qquad (5.7)$$

$$T'_{ПК} = 1892 * \frac{450}{2000} = 425.7$$ год

2. Поточні витрати на експлуатацію V_1" .

Балансова вартість ПК вираховується за (5.8). $Ц_Р$ – ринкова вартість ПК, орієнтовно складає 4000 грн., $k_{УН}$ – коефіцієнт, що враховує витрати на установку і налагодження ПК і дорівнює 0.12.

$$Ц_{ПК} = Ц_Р * (1 + k_{УН}) = 4000 * (1 + 0.12) = 4480 \text{ грн} \qquad (5.8)$$

Амортизаційні відрахування використання ПК, $З_{АМ}$, обчислюються за (5.9), норма амортизаційних відрахувань, яка для ПК дорівнює $Н_А = 5$:

$$З_{АМ} = \frac{Ц_{ПК}}{H_А} \qquad (5.9)$$

$$З_{АМ} = \frac{4480}{5} = 896 \text{ грн.}$$

Витрати на електроенергію, споживану ПК, визначаються за (5.10), де

потужність ПК, $Р_{ПК} = 0.4$ кВт,

фонд корисного часу роботи ПК, $Т_{ПК}$ = 425.7 год,

вартість 1 кВт електроенергії для підприємств, $Ц_{ЕЛ}$ = 0.74 грн/кВт,

коефіцієнт інтенсивного використання ПК, А = 0.9.

Згідно з [ ] при 40-годинному робочому тижні кількість робочих годин 1 вересня - 7 грудня 2012 році фонд часу використання ПК для виконання дипломного проекту складає 547 год.

$$З_{ЕЛ} = Р_{ПК} * Т_{ПК} * Ц_{ЕЛ} * А = 0.4 * 547 * 0.74 * 0.9 = 145.72 \text{ грн}$$

(5.10)

$З_Р$ – витрати на поточний ремонт і технічне обслуговування ПК визначаються як 6% від балансової вартості ПК, $Ц_{ПК}$.

$$З_Р = Ц_{ПК} * 0.06 = 4480 * 0.06 = 268.8 \text{ грн.}$$

$З_{МАТ}$ – непрямі витрати, пов'язані з експлуатацією ПК, визначаються як 5% від балансової вартості ПК $Ц_{ПК}$.

$$З_{МАТ} = Ц_{ПК} * 0.05 = 4480 * 0.05 = 224 \text{ грн.}$$

Таким чином, маємо:

заробітна плата обслуговуючого персоналу (якщо роботи виконуються не на власному ПК); $З_{ОП}$ = 1680 грн, $З_{АМ}$ = 896 грн, $З_{ЕЛ}$ = 113.41 грн,

Поточні витрати на експлуатацію V_1", грн, визначаються за (5.11).

$$V_1'' = З_{ОП} + З_{АМ} + З_{ЕЛ} + З_Р + З_{МАТ} = 1680 + 896 + 145.72 + 268.8 + 224 = 321$$

(5.11)

Отже, загальні витрати на розробку програмного забезпечення комп'ютерної системи розраховуються за формулою (5.12) і складуть:

$$V_1 = V'_1 + V_1'' = 84000 + 3214.52 = 87214.52 \text{ грн.}$$

(5.12)

3. Витрати на придбання і установку ПК V_2.

Витрати на придбання і установку ПК (V_2) визначаються за (5.13) :

$$V_2 = Ц_{ПК} = 4480 \text{ грн.}$$

(5.13)

Якщо немає потреби в купівлі ПК, то ці витрати дорівнюють "0".

4. Витрати на підготовку приміщення V_3.

Ці витрати залежать від стану приміщення, де буде встановлюватися ПК. Так як пристосоване приміщення є, тому:

$$V_3 = 0 \text{ грн.}$$

5. Витрати на навчання персоналу V_4.

В середньому навчання персоналу триватиме 1 місяць, тому можна вважати, що:

$$V_3 = 2000 \text{ грн.}$$

6. Загальна вартість розробки і впровадження системи.

Загальна вартість розробки і впровадження системи V_Σ, вираховується за (5.14):

$$V_\Sigma = V_1 + V_2 + V_3 + V_4 = 87214.52 + 4480 + 0 + 2000 = 93694.52 \text{ грн.}$$

(5.14)

Оскільки норма амортизаційних втрат для комп'ютерних систем $H_A = 5$, то для обрахування річного економічного ефекту слід брати до розгляду величину (5.15)

$$V_P = \frac{V_\Sigma}{H_A} = \frac{93694.52}{5} = 18738.9 \text{ грн.}$$

(5.15)

Річний прибуток $П_Р$ від впровадження системи буде досягнуто за рахунок збільшення робочого часу обладнання і орієнтовно складатиме

10000 грн на рік. Коефіцієнт економічної ефективності розробки вираховується за (5.16):

$$К_{ЕФ} = \frac{П_Р}{V_Р} = \frac{10000}{18738.9} = 0.53$$

(5.16)

Термін окупності розробки дорівнює визначається за формулою (5.17).

$$T_{ОК} = \frac{1}{К_{ЕФ}} = \frac{1}{0.53} = 1.9$$

(5.17)

Таким чином, термін окупності інформаційної системи буде 1 рік і 9 місяців.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

2.1 Вибору засобів розроблення системи

У дослідженні було взято за основу стек технологій (HTML, CSS, Javascript). Ці технології відповідають за написання Web-додатку. Також для крад системи авторизації користувачів я використав стек(Node.js(фреймворк Express, React.js, MongoBD). Цей набір дозволяє самостійно реалізувати всі частини проєкту — як back-end, так і front-end.

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає *видові* у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Принципи роботи React Native в основному такі ж, як ReactJS, за винятком того, що він не маніпулює DOM через VirtualDom.

Він працює у фоновому процесі (який інтерпретує Javascript код написаний розробниками) безпосередньо на кінцевому пристрої і спілкується з нативною платформою. Очевидно, що Facebook виправив помилку, про яку Марк Цукерберг згадував в 2012 році. React Native взагалі не покладається на HTML, все написано на Javascript і залежить від нативних SDK.

HTML (англ. HyperText Markup Language — мова розмітки гіпертексту) — це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет. Браузери отримують HTML-документи з вебсервера або з локальної пам'яті й передають документи в мультимедійні вебсторінки. HTML описує структуру вебсторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Теги, такі як і безпосередньо вводять вміст на сторінку. Інші теги, такі як `<img />` `<input />` `<p>` оточують і надають інформацію про текст документа і можуть включати інші теги як піделементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані на мові сценаріїв, наприклад JavaScript, що впливає на поведінку та вміст вебсторінок. Включення CSS визначає вигляд і компонування вмісту. World Wide Web

Consortium (W3C), який супроводжує стандарти HTML та CSS, заохочує використання CSS над явним презентаційним HTML з 1997 року.

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

JavaScript (**JS**) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є *Раян Дал* (Ryan Dahl). Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то *node.js* надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа *Node.js* перетворила

JavaScript на мову загального використання з великою спільнотою розробників.

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плаґінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безкоштовно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

Можливості

- Інтеграція з системами управління версіями Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю побудови списку змін і відкладених змін
- Інтеграція з системами відстеження помилок
- Модифікація файлів .css, html, .js з одночасним переглядом результатів (Live Edit, в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»)

- Віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації
- Можливості Zen Coding і Emmet

Express.js, або просто **Express** — програмний каркас розробки серверної частини веб-застосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-застосунків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плаґінів, що підключаються.

Express є бекендом для програмного стека MERN, разом з базою даних MongoDB і каркасом React для фронтенду.

2.2.1 Призначення та цілі створення системи

Метою даного проєкту є створення web додатку та розширення можливостей для замовлення реклами.

Головним завданням є створення максимально зручного та зрозумілого інтерфейсу, який можна налаштувати для індивідуальних користувачів, запити на послуги.

Користувачем даного web додатку є **,** клієнт, та адмін, який є головним у даній системі.

Дана система допоможе власнику web додатку та crud системи гнучко упраляти данними які залишають користувачи в запитах, обробляти замовлення.

2.2.2 Вимоги до створюваної системи

Створювана система повинна мати простий інтерфейс для зручного користування. Дані мають бути подані у чіткій та зрозумілій формі для вивчення. Повинні бути реалізовані функції, що забезпечують доступ до матеріалу та навігацію.

- Вимоги щодо операційної системи — програмний продукт повинен функціонувати на найпопулярніших браузерах, серед яких зокрема:
  - Google chrome;
  - Firefox
  - Opera
  - Microsoft Edge
  - Safari
- Вимоги до інтерфейсу користувача — простий та незавантажений інтерфейс; зручна навігація по додатку; поле введення даних для пошуку, додаток повинен забезпечувати зручність та простоту взаємодії з користувачем.

- Вимоги до підключення до мережі – програмний продукт має мати постійне підключення до сервера для завантаження даних.

2.2.3 Функції, які має виконувати система

Основні функції, які повинен виконувати web додаток:

- Автоматичне упорядкування;
- Збереження потрібних даних про послугу;
- Зручний вивід інформації;
- Зручне редагування послуг;
- Створення запитів на послугу яку хоче вибрати користувач;
- Можливість редагувати свій профіль та вноси зміни;

2.2.4 Вхідні та вихідні дані системи

Вхідні дані розроблюваної системи:

- Дані про послугу
- Критерії запиту
- Запити на пошук
- Запит на збереження

Вихідні дані розроблюваної системи:

- Результати пошуку та впорядкування
- Збережені товари

Складена БД з унікальними id, назвами там їх тип данних

Для цього використовуємо MongoBD та Node.js для складання схеми

Таблиця 2.1. Структура таблиці «User»

№	Назва поля	Тип даних
1	**_id**	string
2	name	string
3	email	string
4	password	string
5	role	string
6	city	string
7	birhDate	Data

В кожного користувача має бути свій профіль замовлень.

Таблиця 2.1. Структура таблиці «Profile »

№	Назва поля	Тип даних
1	**_id**	string
2	name	string
3	email	string
4	password	string
5	role	string
6	city	string
7	birhDate	Data

```
const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const User = require("./user");

const ProfileSchema = new Schema( definition: {
  name: { type: String, required: true },
  gender: { type: String, required: true, default: "male" },
  birthDate: { type: Date, required: true },
  city: { type: String, required: true },
  userId: { type: mongoose.Types.ObjectId, required: true, ref: "User" },
});

module.exports = mongoose.model( name: "Profile", ProfileSchema);
```

```
const mongoose = require("mongoose");
const uniqueValidator = require("mongoose-unique-validator");
const Schema = mongoose.Schema;

const UserSchema = new Schema( definition: {
  username: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  role: { type: String, required: true, default: "user" },
  password: { type: String, required: true, minlength: 6 },
  profiles: [{ type: mongoose.Types.ObjectId, required: true, ref: "Profile" }],
});

UserSchema.plugin(uniqueValidator);

module.exports = mongoose.model( name: "User", UserSchema);
```

На даному етапі створюється об’єкт з типізованими данними які відповідають за зв’язок з БД.

РОЗДІЛ 3. Представлення опису розробки web-додатку

Для початку починаємо займатися розробкою UI частини тобто юзер інтерфейс. Почнемо з реєстрування на сайті.

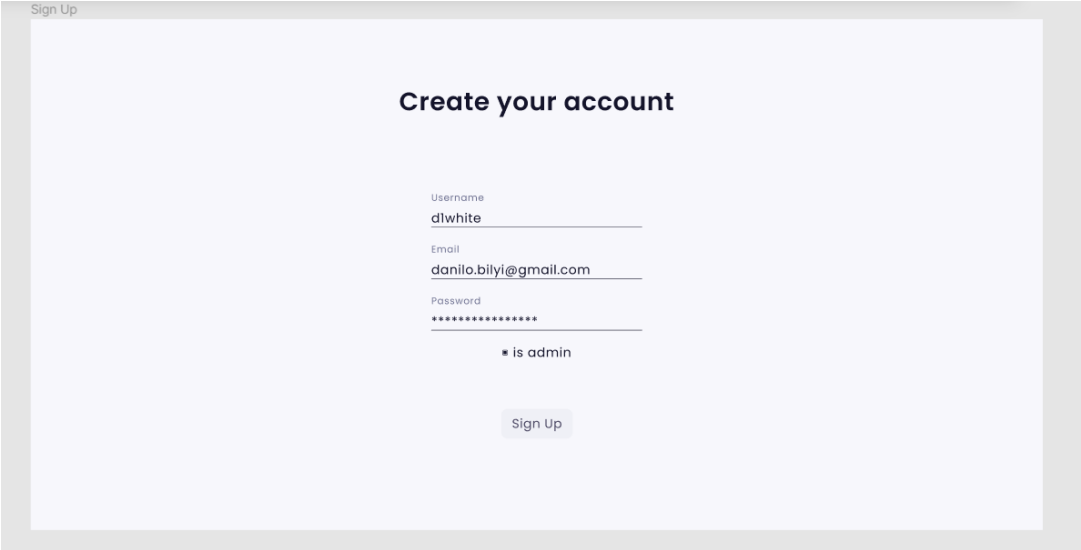


Рис. 1.1 Sing UP

Створюємо файл Sing_Up.components в форматі jsx

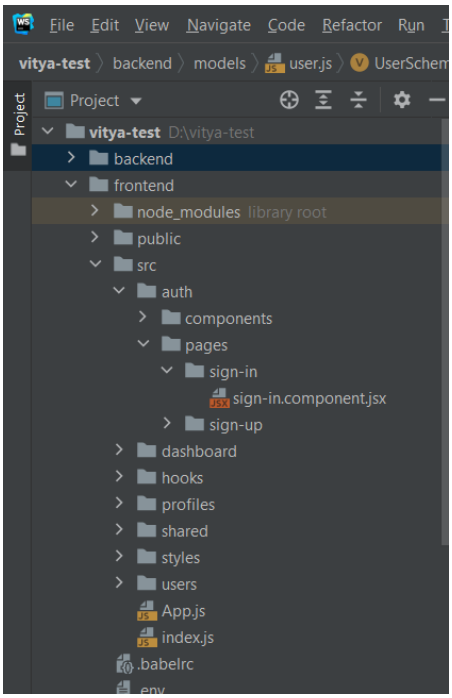


Рис 1.2 Дерикторія сайту

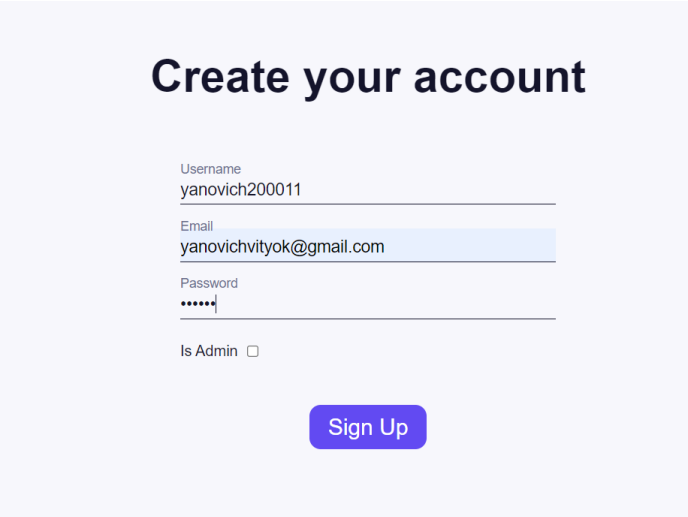


Рис. 1.1.2 Реєстрація

Перша сторінка реєстрації готова, але ці дані нікуди не відправляються.

І щоб реалізувати реєстрацію, потрібно сайт підключити до бази щоб отримати дані що користувач зареєструвався.

ПІДКЛЮЧЕННЯ ДО БД

Щоб підключитися до бд потрібно зареєструватися.

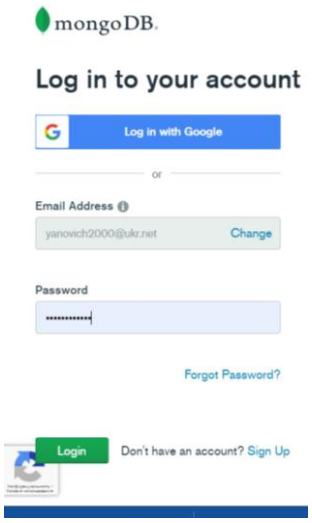


Рис 1.3 Реєстрація mongoBD

Далі створюємо базу данних і в цій базі потрібно створити Cluster який згенерує посилання на нашу БД там IP адресу

Рис.1.4 Створюємо Cluster

Чекаємо 1-3хвилини cluster готовий. Він згенерував посилання на БД.

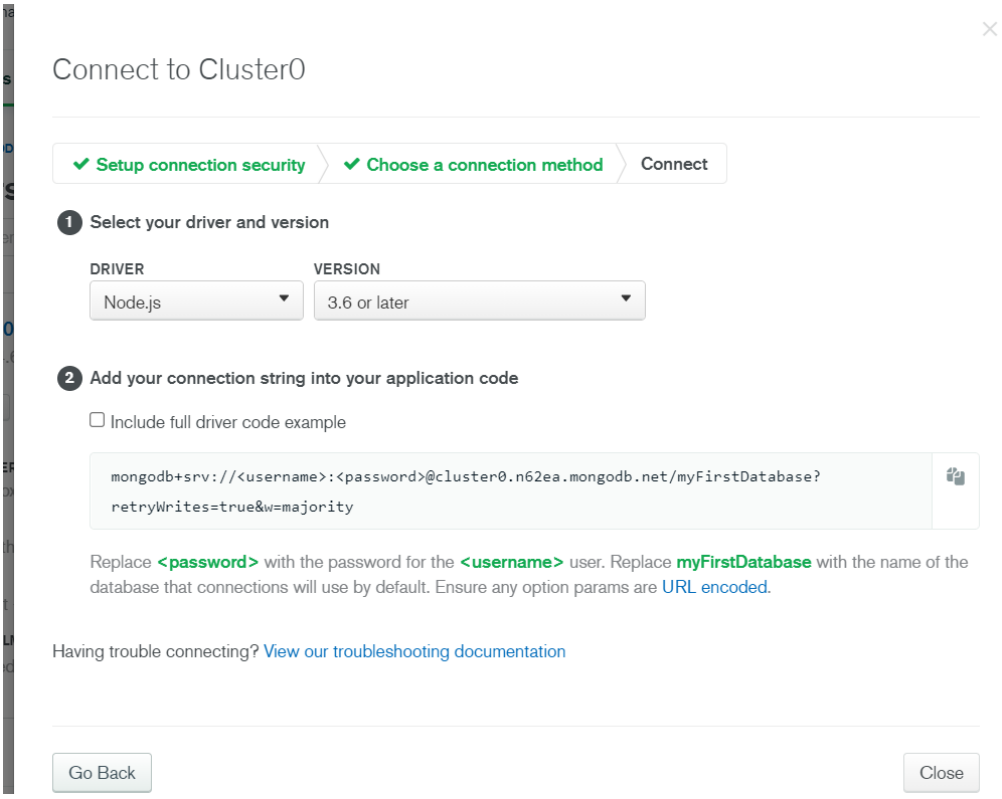


Рис.1.6 Підключення до БД.

Далі починаємо писати ряд залежностей при реєстрації. Ці залежності пишемо на Node.js тобто враховуємо всі помилки, які можуть бути при реєстрації.

```
const express = require("express");
const bodyParser = require("body-parser");
```

```
const mongoose = require("mongoose");
const cors = require("cors");

const HttpError = require("./models/http-error.modle");
const profilesRoutes = require("./routes/profiles.route");
const usersRoutes = require("./routes/users.route");

const app = express();

app.use(cors());

app.use(bodyParser.json());

app.options("*", cors());

app.use("/api/profiles", profilesRoutes);
app.use("/api/users", usersRoutes);

app.use((req, res, next) => {
  const error = new HttpError("Couldn't find this route", 404);
  throw error;
});

app.use((error, req, res, next) => {
  if (res.headerSent) {
    return next(error);
  }

  res.status(error.code || 500);
  res.json({ message: error.message || "Unexpected error occurred" });
});

mongoose
  .connect(

`mongodb+srv://${process.env.DB_USER}:${process.env.DB_PASSWORD}@cluster0.n62ea.mongodb
.net/${process.env.DB_NAME}?retryWrites=true&w=majority`,
    { useNewUrlParser: true, useUnifiedTopology: true }
  )
  .then(() => app.listen(5000))
  .catch((e) => console.log(e));
```

Ми провели підключення до бд.

І далі щоб зв’язати користувача з БД, використовуємо jwt token з унікальним Id.

```
const jwt = require("jsonwebtoken");

const HttpError = require("../models/http-error.modle");
const User = require("../models/user");

module.exports = (req, res, next) => {
  if (req.method === "OPTIONS") {
    return next();
  }
  try {
    const token = req.headers.authorization.split(" ")[1];
    if (!token) {
      throw new Error("Auth failed");
```

```
  }

  const decodedToken = jwt.verify(token, process.env.TOKEN_KEY);
  req.userData = { userId: decodedToken.userId };
  next();
 } catch (e) {
  return next(new HttpError("Auth failed", 401));
 }
};
```

Щоб перевірити Реєстрацію в терміналі, використовується команду yarn start

```
Microsoft Windows [Version 10.0.14393]
(c) Корпорация Майкрософт (Microsoft Corporation), 2016. Все права защищены.

D:\vitya-test\backend>yarn start
yarn run v1.22.10
$ nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
(node:1628) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
```

Реєстрація прошла успішно, але потрібно перевірити чи в бд з’явилися данні, про користувача, якій хоче зробити замолення.

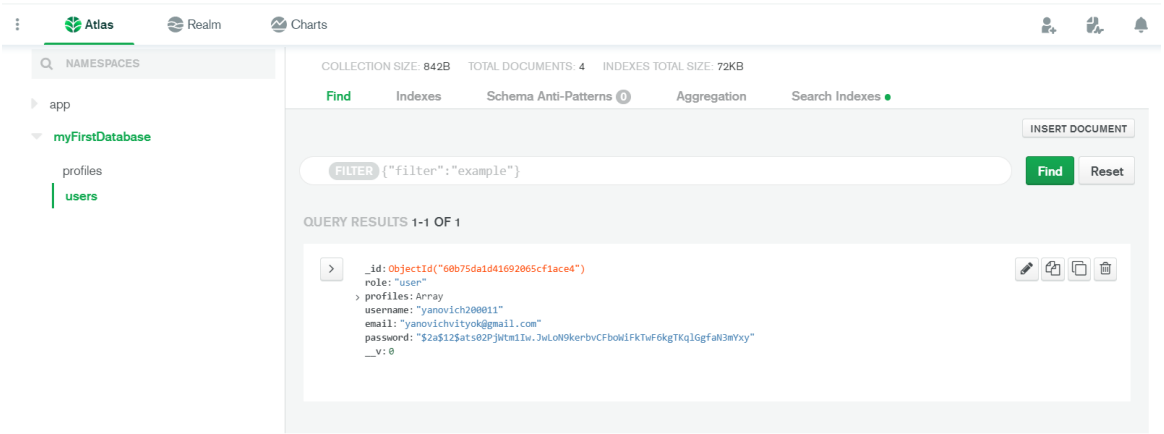
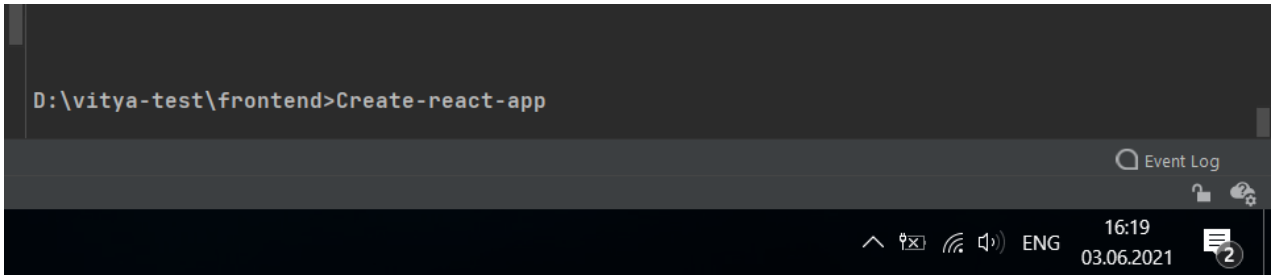


Рис 1.7 Данні які пришли з форми реєстрації

Так, реєстрація прошла успішно. Далі потрібно створити запити, які користувачи залишають розробникам.

3.2 Алгоритмізація та реалізація комплексу задач автоматизації

Щоб створити проект для початку його треба ініціалізувати: Create-react-app



Далі потрібно налаштувати роутинг нашого веб додатку :

```
const express = require("express");
const { check } = require("express-validator");

const router = express.Router();

const usersController = require("../controllers/users.controller");

router.get("/", usersController.getUsers);

router.post(
  "/signup",
  [
    check("username").not().isEmpty(),
    check("email").normalizeEmail().isEmail(),
    check("password").not().isEmpty().isLength({ min: 6 }),
  ],
  usersController.signup
);

router.post(
  "/login",
  [
    check("email").normalizeEmail().isEmail(),
    check("password").not().isEmpty().isLength({ min: 6 }),
  ],
  usersController.login
);

router.get("/:uid", usersController.getUserById);

router.patch("/:uid", usersController.updateUser);

router.delete("/:uid", usersController.deleteUser);

module.exports = router;
```

Налаштували роути для Користувачів :

```
const express = require("express");
const { check } = require("express-validator");
```

```
const router = express.Router();

const profilesController = require("../controllers/profiles.controller");
const dashboardController = require("../controllers/dashboard.controller");
const checkAuth = require("../middleware/check-auth");

router.use(checkAuth);

router.get("/user/:uid", profilesController.getProfilesByUserId);
router.get("/user/info", dashboardController.getInfo);

router.post(
  "/",
  [
    check("name").not().isEmpty(),
    check("city").not().isEmpty(),
    check("birthDate").isNumeric(),
  ],
  profilesController.createProfile
);

router.patch(
  "/:pid",
  [
    check("name").not().isEmpty(),
    check("city").not().isEmpty(),
    check("birthDate").isNumeric(),
  ],
  profilesController.updateProfile
);

router.delete("/:pid", profilesController.deleteProfile);

module.exports = router;
```

Налаштували Роути для Запитів

3.3.Розробка інтерфейсу

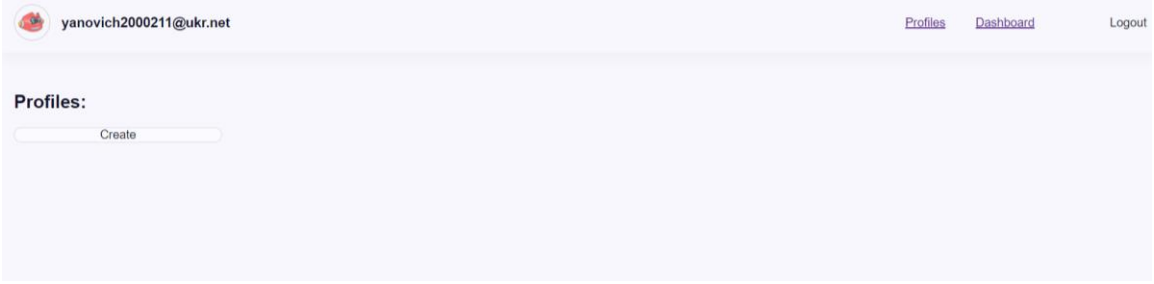
Після того, як налаштували роутинг, потрібно оформити розмітку для користувачів та запитів.

```
import React, { useEffect, useState, useContext } from 'react';
import { useRouteMatch, useParams } from 'react-router-dom';

import ProfilesGrid from '../components/profiles-grid/profiles-grid.component';
import ProfileCard from '../components/profile-card/profile-card.component';
import CreateProfile from '../components/create-profile/create-profile.component';

import { useHttpClient } from '../../hooks/http.hook';
import { ClipLoader } from 'react-spinners';
import { AuthContext } from '../../shared/context/auth.context';

const ProfilesPage = () => {
  const params = useParams();
  const auth = useContext(AuthContext);
  const [profiles, setProfiles] = useState([]);
  const { loading, error, sendRequest } = useHttpClient();
```

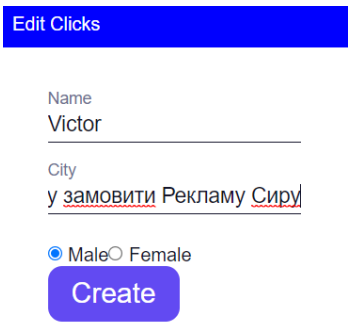
```
  const getUserProfiles = async () => {
    try {
      const res = await sendRequest(
        `http://localhost:5000/api/profiles/user/${params.id}`,
        'GET',
        null,
        {
          Authorization: `Bearer ${auth.token}`,
        }
      );
      setProfiles(res.profiles);
    } catch (e) {}
  };

  useEffect(() => {
    getUserProfiles();
  }, [sendRequest]);

  return (
    <div className={'container'}>
      {loading && <ClipLoader />}
      <h2>Profiles:</h2>
      <ProfilesGrid>
        {profiles && !loading && (
          <>
            {profiles.length !== 0 && (
              <>
                {profiles.map(profile => (
                  <ProfileCard
                    key={profile.id}
                    profile={profile}
                    profiles={profiles}
                    setProfiles={setProfiles}
                  />
                ))}
              </>
            )}
          </>
        )}
        <CreateProfile profiles={profiles} setProfiles={setProfiles} />
      </ProfilesGrid>
    </div>
  );
};

export default ProfilesPage;
```

Тепер, можемо залишити запит, нажавши на кнопку Create.

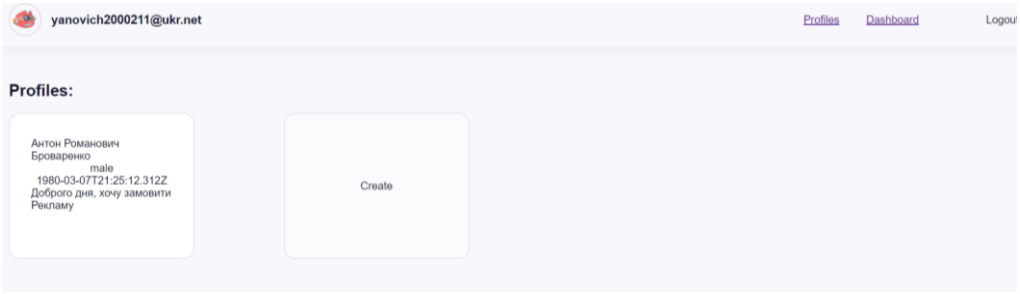
Так виглядає сторінка запитів і після залишеного запиту Адміністратор може його передивитися.



Далі вказуєм в полі name ім’я і в полі нижче залишаємо запит на замовлення.



Так, все працює, запит створений. Тепер, потрібно чекати, коли Адміністрато web-додатку передивиться запит.



Такоє, можемо залишити і інші запити. Запити зберігаються в проміжку місяць-два, після того, як запит був переглянутий.

Далі створюємо основу нашого веб додатку

Крок перший. Спочатку реалізовуємо розмітку сайту за допомогою HTML Документа.

```
<!DOCTYPE html>
<html lang="ru-RU">
<head>
   <meta name="viewport"
         content="width=device-width, height=device-height, initial-scale=1.0, maximum-scale=1.0, user-
scalable=no">
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>Моя компания - Лучшие товары и услуги в Интернете</title>

   <style type="text/css">
      :root {
         --fallback-font-header: "";
         --fallback-font-paragraph: ""
      }

      @font-face {
         font-family: 'Montserrat';
         font-style: normal;
         font-weight: 500;
         font-display: swap;
         src: local('Montserrat'), url('fonts/Montserrat-Medium.woff') format('woff');
      }

      @font-face {
         font-family: 'Montserrat';
         font-style: italic;
         font-weight: 500;
         font-display: swap;
         src: local('Montserrat'), url('fonts/Montserrat-MediumItalic.woff') format('woff');
      }

      @font-face {
         font-family: 'Montserrat';
         font-style: normal;
         font-weight: 900;
         font-display: swap;
         src: local('Montserrat'), url('fonts/Montserrat-Black.woff') format('woff');
      }

      @font-face {
         font-family: 'Montserrat';
         font-style: italic;
         font-weight: 900;
         font-display: swap;
         src: local('Montserrat'), url('fonts/Montserrat-BlackItalic.woff') format('woff');
      }
```

```
    </style>

    <link href="css/layouts.1fd853.min.css" rel="stylesheet" type="text/css">


</head>
<body id="top-body" class="site site_main-page " data-html-version="8450396804356098204101" data-
grid="12"
      data-ng-controller="BlocksController as blockCtrl" data-ng-init="moduleId = 2" data-lang="ru_RU"
style>


<div class="site__layout">
    <div id="container" class="site-container site-container_with-catalog-menu module main-page-module
">

        <header class="site-header js-sticky__header small-12">
            <div class="site-header__content">
                <div class="site-header__top-contacts">
                    <link href="css/topContacts.0f341d.min.css" rel="stylesheet" type="text/css">
                    <section id="topcontacts" data-block-id="3" data-ng-init="init(3)"
                             data-ng-controller="TopContactsController"
                             class="block-3 widget widget-block -nt-widget top-contacts nh-editor-panel"
                             data-ng-class="{'on-view': !blocks[3].isEdit, 'edit-item': blocks[3].editItem}">
                        <div class="top-contacts__inner">
                            <div data-ng-show="!blocks[3].isEdit" class="content-block" id="topcontacts-show">

                            </div>
                        </div>
                    </section>
                </div>
                <div class="site-header__bottom-section js-combo-box" combo-box="menu">
                    <div class="site-header-bottom">
                        <div class="site-header__logo small-8 inline-column large-3">
                            <div class="site-header__container">
                                <div class="small-2 medium-1 hide-for-large">
24 24">
                                        <defs/>
                                        <path d="M10.8 12.2L3 20l1.4 1.4 7.8-7.8 7.8 7.8 1.4-1.4-7.8-7.8 7.8-
7.8L20 3l-7.8 7.8L4.4 3 3 4.4l7.8 7.8z"/>
                                    </svg>
                                </div>
                            </div>
                            <nav id="main-menu" class="site-menu">
                                <div>
                                    <ul class="no-bullet site-menu__list clearfix">
                                        <li id="lnk-1" data-sort="0"
                                            class=" active disable-sort site-menu__item"
                                            data-loc="&amp;quot;mainpage&amp;quot;" data-label="Главная"><a
                                                href="https://hashtap.nethouse.ru/">
                                            Головна                                        </a></li>
                                        <li id="lnk-2" data-sort="1" class=" site-menu__item"
                                            data-label="Новости"><a href="https://hashtap.nethouse.ru/posts">
                                            Новини
                                        </a></li>

                                        class="quantity-items top-cart__quantity">
                                            0
                                    </span><span class="top-cart__arrow"><svg
xmlns="http://www.w3.org/2000/svg"
                                                                             fill="currentColor"
                                                                             viewBox="0 0 12 6"><path
                                        d="M6 6l5.2-6H.8L6 6z"/></svg></span></span></button>
```

```
                        </div>
                        <div class="js-combo-box__content_cart combo-box__content combo-
box__content_invert text-left">
                            <div class="combo-box__header">
                                <div class="combo-box__title">Корзина</div>
                                <div class="combo-box__close js-combo-box__close_cart">
                                    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
                                        <defs/>
                                        <path d="M10.8 12.2L3 20l1.4 1.4 7.8-7.8 7.8 7.8 1.4-1.4-7.8-7.8 7.8-
7.8L20 3l-7.8 7.8L4.4 3 3 4.4l7.8 7.8z"/>
                                    </svg>
                                </div>
                            </div>
                            <div class="small-cart site-header__small-cart ">
                                <div class="cart-is-empty">Корзина пуста</div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div></div>
    </header>


    <div id="content-anchor" class="site-content small-12">
        <div class="site-background" style></div>
        <section data-blocks-list id="content" class="js-blocks-list" data-ng-
controller="ContentController">

            <section class="slider widget ng-cloak on-view widget-block nh-editor-panel main-page-slider"
                data-widget-slider data-params="{&amp;quot;route&amp;quot;:&amp;quot;mainPage&amp;quot;}" data-
type="43"
                data-deleted data-entity-id="12">
                <link href="css/slider.3624bb.min.css" rel="stylesheet" type="text/css">

                    <div class="mr-block">
                        <div class="row">
                            <div class="mrb-small-18">
                                <div class="-gd-content user-inner slider__user-content"><p><span
                                        style="font-size: 60px; font-weight: bold;">Hashtap ДЛЯ РЕКЛАМНИХ
ВПРОВАДЖЕНЬ</span>
                                </p>
                                    <p><br></p>
                                    <p><span style="font-size: 20px;">На цьому сайті ви зможете
замовити рекламу своєї продукції. Та продукції Пирятинського Сирзаводу</span></p>
                                </div>
                            </div>
                        </div>
                        <div class="row">
                            <div class="mrb-small-18">
                                <div class="-gd-content">
                                    <div id="mrb-button-block"
                                        class="mrb-button-block theme-default-invert"
                                        style="text-align: center"><a href
                                                                      class="mrb-btn-large mrb-btn-item a"
                                                                      tabindex="0"><span
                                            class="mrb-btn-item-text">Відправити
замовлення</span></a></div>
                                </div>
                            </div>
```

```
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
            <div data-sorted-blocks class="js-sorted-blocks main-page-module__list module__list" data-
region="1"
                 data-module="2">
                <section data-widget-desc data-widget-container data-widget-id="6" data-widget-sort="1"
                        data-widget-
url="/widget?route=mainPage&amp;on_main=1&amp;type=3&amp;widget_id=6"

                                        <div class="services__order-button text-center"><a
                                                href="#order-form"
                                                class="button buttons button_for_services-list order-button
order">Оформити замовлення</a></div>
                                    </div>
                                </div>
                            </article>
                        </div>
                    </div>
                </div>
                <div class="widget__content widget__content_filled">
                    <div class="row">
                        <div data-ng-if="$ctrl.loaded" class="widget__view small-10 small-offset-1 column"
                            data-ui-view="services-view"></div>
                    </div>
                </div>

                </section>

            </section>

            <section data-widget-catalog data-widget-container data-widget-id="9" data-widget-sort="1"
                    data-widget-
url="/widget?route=mainPage&amp;on_main=1&amp;type=11&amp;widget_id=9"
                    data-region="1" data-widget-type="11"
                    class="widget-sort widget-block block-9 nh-editor-panel">


                <section id="products" data-widget-content data-ng-init="$ctrl.categoryId = null"
                        class="catalog on-view widget-block  -preview-size-250 catalog_light lazyload">
                    <link href="css/catalog.46f072.min.css" rel="stylesheet" type="text/css">
                    <link href="css/product.daeff1.min.css" rel="stylesheet" type="text/css">
                    <div class="widget">
                        <header class="widget__header row text-center widget__header_underlined
infoPlate">
                            <div class="small-12 column">
                                <h2><a href="https://hashtap.nethouse.ru/products">Каталог товарів</a></h2>
                            </div>
                        </header>
                        <div class="widget__content widget__content_filled">
                            <div data-ng-show="$ctrl.showContent()">
                                <article class="catalog__list catalog__list_250x330">
                                    <div class="catalog__product catalog__product_250x330">
                                        <div class="product-item  text-center medium-text-left"
                                            id="item50765612">
                                            <div class="product-item__content"><a
                                                    href="https://hashtap.nethouse.ru/products/50765612"
                                                    class="product-item__preview product-item__preview_250x330">
```

```
                                    <picture>
                                        <source
srcset="images/fb923d6857b14e2546237fb1aa45dd6cccb05d37_3.png 1x,
images/fb923d6857b14e2546237fb1aa45dd6cccb05d37_1.png 2x"
                                                type="image/webp">
                                        <source

                                    </picture>
                                </a>
                                    <div class="product-item__content">
                                        <div>
                                            <div class></div>
                                            <div class="product-item__link"><a

href="https://hashtap.nethouse.ru/products/50765613">Товар
                                                №2</a></div>
                                        </div>
                                        <div class="product-item__price -inline-group">
                                            <div class="product-item-price">4 900 ₽</div>
                                        </div>
                                    </div>
                                </div>
                                <div class="product-item__button product-item__button_card cart-btn js-
order-product button button_for_product-card js-cart-btn">
                                    <div class="cart-btn__text">В корзину</div>
                                </div>
                            </div>
                        </article>


                    </div>
                </div>
            </div>
            <div class="widget__content widget__content_filled">
                <div class="row">
                    <div data-ng-if="$ctrl.loaded" class="widget__view small-10 small-offset-1 column"
                        data-ui-view="catalog-view"></div>
                </div>
            </div>

        </section>


        <script type="text/javascript">
            if (typeof gtag != 'undefined') {
                gtag('event', 'view_item_list', {
                    "items": [{
                        "id": "50765612",
                        "name": "\u0422\u043e\u0432\u0430\u0440 \u21161",
                        "list_name": "main page",
                        "list_position": 1,
                        "price": "2000"
                    }, {
                        "id": "50765613",
                        "name": "\u0422\u043e\u0432\u0430\u0440 \u21163",
                        "list_name": "main page",
                        "list_position": 2,
                        "price": "4900"
                    }]
                });
            }
        </script>
```

```
                </section>

                <section data-widget-lead-form data-widget-container data-widget-id="10" data-widget-
sort="1"
                        data-widget-
url="/widget?route=mainPage&amp;on_main=1&amp;type=41&amp;widget_id=10"
                        data-region="1" data-widget-type="41"
                        class="widget-sort widget-block block-10 nh-editor-panel">


                    <section data-widget-content
                            class="-nt-widget -widget-lead-form lead-form lead-form_dark lazyload">
                        <link href="css/lead_form.3d3d36.min.css" rel="stylesheet" type="text/css">
                        <div class="widget">
                            <div class="content-block widget__content widget__content_filled">
                                <div class="row widget__inner" data-ng-show="$ctrl.showContent()">
                                    <div class="small-12 column">
                                        <header class="text-center widget__header ">
                                            <p class="widget__header--paragraph">
                                                Заповніть форму, для звязку
                                            </p>
                                        </header>



    <footer id="footer" class="site-footer site-footer_dark">
        <div class="row">
            <section class="small-12 column">
                <div class="-inline-group">
                    <div class="small-12 large-8 text-center medium-text-left">
                        <div id="footer-text" class="footer-text client nh-editor-panel">
                            <div id="footertext1" class="footer-text__about">
                                2021 © Моя кампанія
                            </div>
                            <div id="footertext2" class="footer-text__desc">Послуги інтернет замовлень та
реклам ТОВ "Пирятинський Сирзавод"</div>
                        </div>
                    </div>
                    <div class="small-12 medium-12 large-4 site-footer_indent">
                        <div class="text-center large-text-right site-footer__logo">
                            <div class="inline-block">

                            </div>
                        </div>
                    </div>
                </div>
            </section>
        </div>
    </footer>


</div>
<div id="for-explain"></div>
<input type="hidden" class="-currentFont" value="Montserrat">

<div id="vk_api_transport"></div>


</body>
</html>
```

Крок другий. Використовуючи CSS стилі, придаємо додатку основний вид.

```
.button {
    font-weight: 500;
    letter-spacing: -.00313rem;
    transition: all 0.2s ease-out;
    display: inline-block;
    cursor: pointer
}

.cart {
    width: 100%;
    background-color: transparent
}

.cart .select-dynamic-field select {
    background: #FFFFFF;
    color: #000;
    border: .0625rem solid #b2bcc3;
    font-size: .875rem;
    height: 2.5rem;
    border-radius: .25rem;
    padding-left: .625rem;
    color: #111;
    font-family: inherit;
    font-weight: 700;
    text-transform: uppercase;
    font-size: .75rem;
    line-height: 100%;
    margin: 0 0 1.6875rem;
    padding: 0 1.875rem 0 .9375rem;
    background-position: 103% center;
    background-position: right -1rem center
}

.cart .select-dynamic-field select::placeholder {
    color: #647886
}

.cart .select-dynamic-field select:hover {
    border-color: #1f61ff
}

.cart .select-dynamic-field select:focus {
    border-color: #054fff
}

.cart .dynamic-field-checkbox {
    margin: 0 0 1.6875rem
}

.cart input, .cart textarea {
    background: #FFFFFF;
    color: #000;
    border: .0625rem solid #b2bcc3;
    font-size: .875rem;
    height: 2.5rem;
    border-radius: .25rem;
    padding-left: .625rem;
    margin: 0 0 1.6875rem
```

```
}

.cart input::placeholder, .cart textarea::placeholder {
    color: #647886
}

.cart input:hover, .cart textarea:hover {
    border-color: #1f61ff
}

.cart input:focus, .cart textarea:focus {
    border-color: #054fff
}

.cart textarea {
    height: 100px;
    resize: vertical;
    line-height: 1.5
}
```

Зовнішній вид Навігації можна подивитися на Рис 1.9

```
.cart ul {
    list-style-type: none
}

.cart ul.errors {
    font-size: .875rem;
    text-transform: lowercase
}

@media screen and (min-width: 48em) and (max-width: 63.9375em) {
    .cart .select-dynamic-field select {
        font-size: 1.125rem;
        background-position: 102% center;
        background-position: right -1rem center
    }

    .cart input, .cart textarea {
        font-size: 1.125rem
    }
}

.cart .dynamic-form input[type="text"] {
    background: #FFFFFF;
    color: #000;
    border: .0625rem solid #b2bcc3;
    font-size: 1.125rem;
    height: 4rem;
    border-radius: .5rem;
    padding-left: 1rem
}

.cart .dynamic-form input[type="text"]::placeholder {
    color: #647886
}

.cart .dynamic-form input[type="text"]:hover {
    border-color: #1f61ff
}

.cart .dynamic-form input[type="text"]:focus {
    border-color: #054fff
```

```
}

.cart .dynamic-form textarea {
    font-size: .9375rem;
    background-color: #eaeaea;
    color: #888
}
```

Зовнішній Вид Головної сторінки Рис 1.10

```

.cart .dynamic-form label {
    padding-top: 0;
    padding-bottom: 0;
    padding-right: 0;
    margin: 0
}

.cart .dynamic-form dl {
    margin-bottom: 1.6875rem
}

.cart .dynamic-form dl:last-child {
    margin-bottom: 0
}

.cart .dynamic-form dt {
    margin: 0;
    font-weight: normal
}

.cart .dynamic-form dd {
    display: block;
    position: absolute;
    bottom: -1.3125rem;
    margin: 0;
    width: 100%
}

.cart .dynamic-form dd div {
    display: none;
    color: #ff0000;
    font-size: .75rem;
    line-height: 1.25;
    text-align: left
}

.cart .dynamic-form dd div.error {
    display: block
}

.cart input[type="text"], .cart textarea {
    margin: 0
}

.cart input[type="checkbox"] + label span {
    display: inline-block;
    vertical-align: middle;
    line-height: 1.5
}

@media screen and (min-width: 48em) and (max-width: 63.9375em) {
    .cart input[type="checkbox"] + label span {
```

```
        line-height: 1.3
    }
}

.cart__button_cancel {
    width: 100%;
    padding: 1rem;
    color: #3772ff;
    text-align: center;
    cursor: pointer;
    display: flex;
    align-items: center
}

.cart__button_cancel svg {
    fill: currentColor
}

@media screen and (max-width: 47.9375em) {
    .cart__button_cancel {
        justify-content: center;
        margin-top: 1rem
    }
}

@media screen and (min-width: 48em) {
    .cart__button_cancel {
        width: auto;
        padding: 1rem .625rem
    }
}

.cart-table {
    display: flex;
    flex-direction: column
}

.cart-table .total-result {
    font-size: .8125rem;
    line-height: 1.2;
    font-weight: 900;
    text-align: right;
    width: 100%;
    display: flex;
    align-items: center;
    justify-content: space-between
}
```

І закінчені стилі для Товарів та послуг + footer(підвал сайту) Рис 1.11, 1.12, 1.13

І за допомогою всіх ціх операцій отримує продукт.

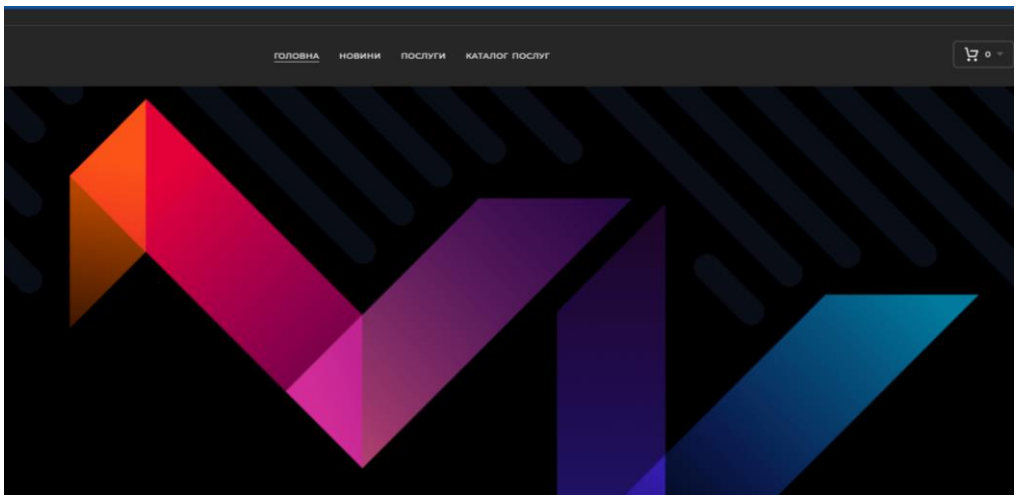


Рис 1.9 Навігація по додатку

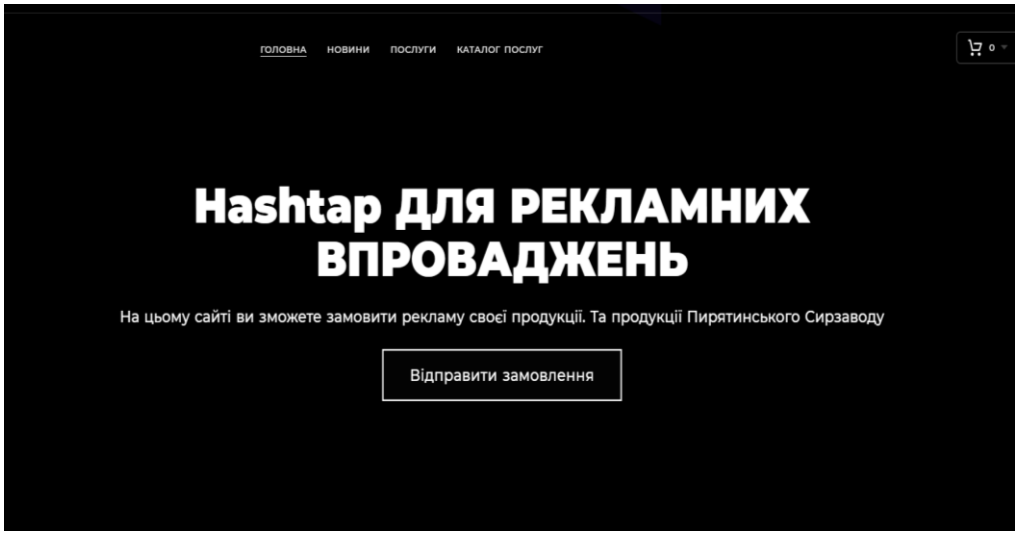


Рис.1.10 Головна сторінка

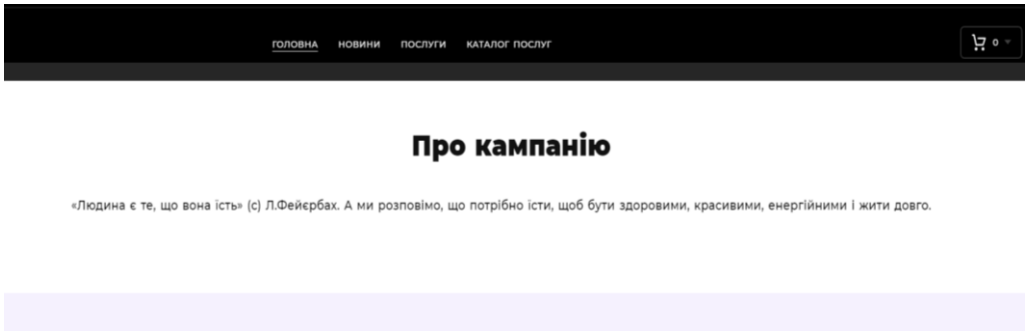


Рис.1.11 Новини

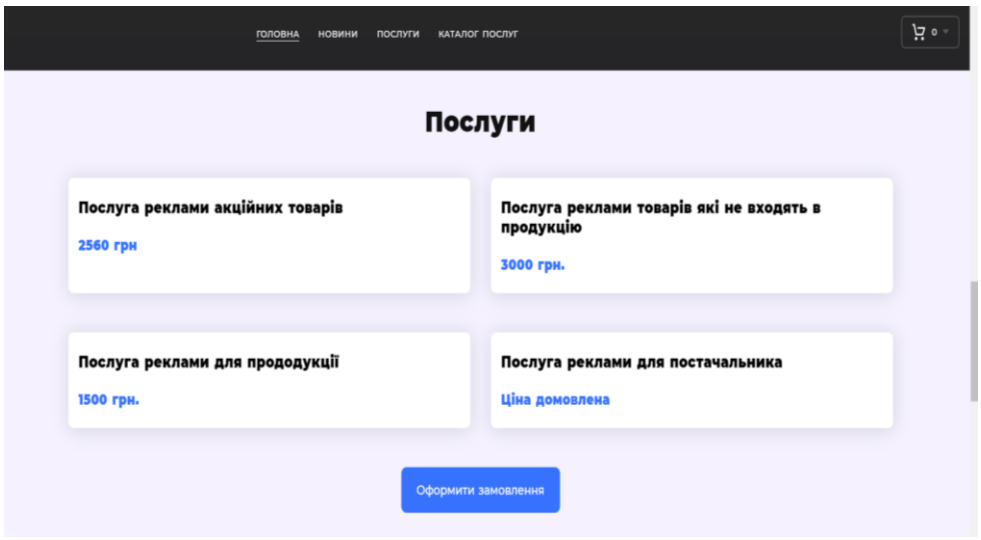


Рис. 1.12 Послуги які надає компанія

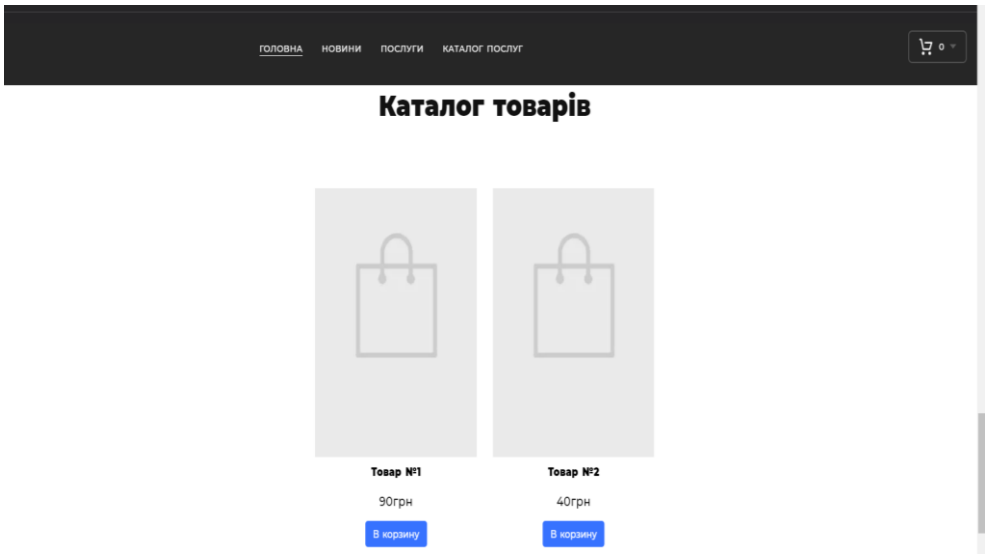
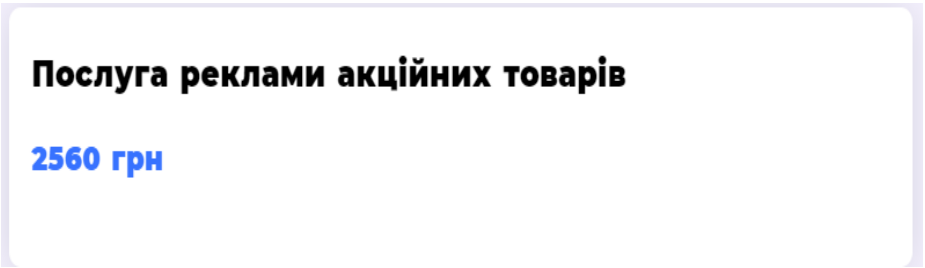


Рис.1.13 Каталог товарів

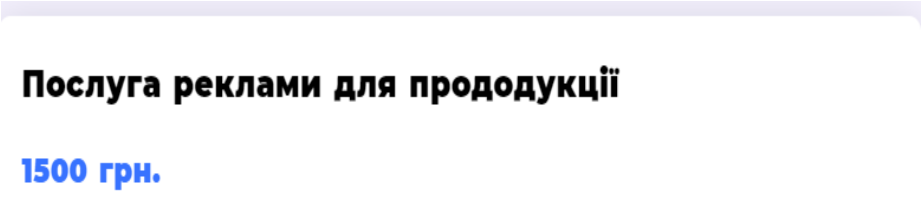
3.3.Інструкція користувача

В даному додатку є змого передивится послуги, які надає компанія.

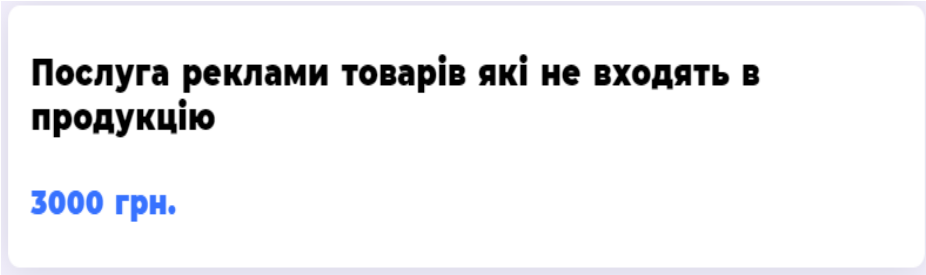
Послуга номер один :



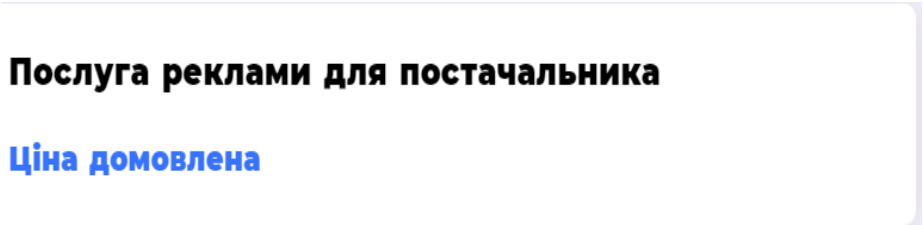
Послуга номер два :



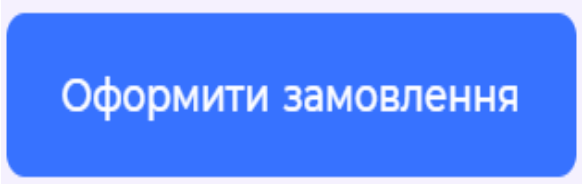
Послуга номер три :



Послуга номер чотири :

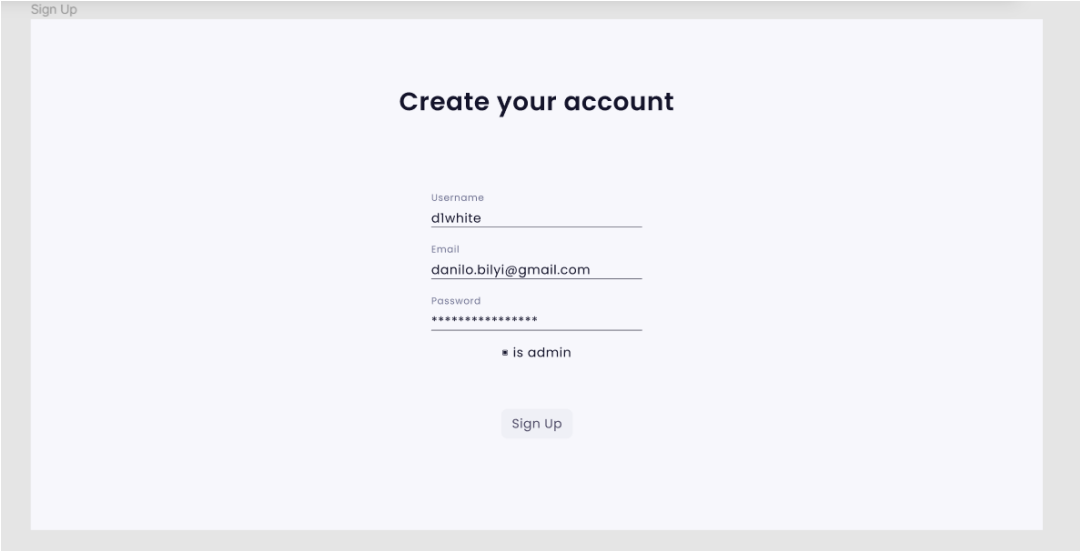


Користувач має оформити замовлення для цього потрібно натиснути на Кнопку :

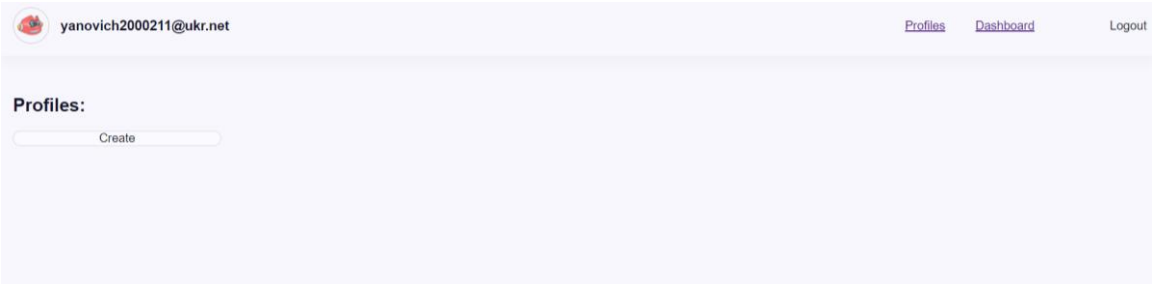


І після цього, користувач перейде на Crud систему, де він залишить це замовлення і протягом деякого часу або відразу, адміністратор прийме важе замовлення.

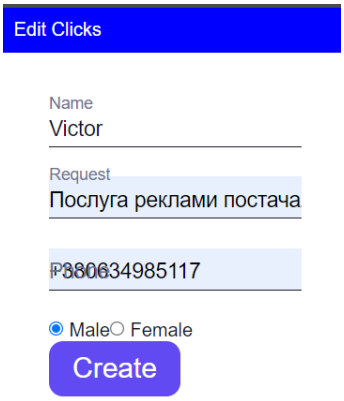
Крок перший: реєстрація



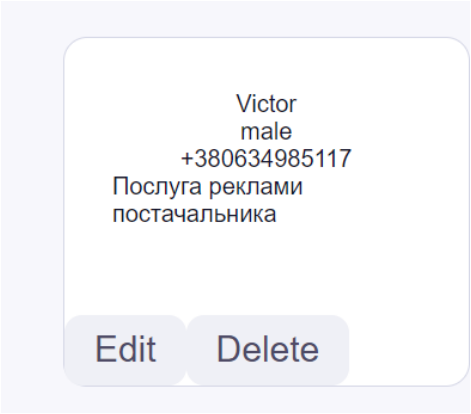
Крок другий: За допомоги кнопки Create клієнт створює запит, де вказує дану рекламу яку він замовив або продукт.



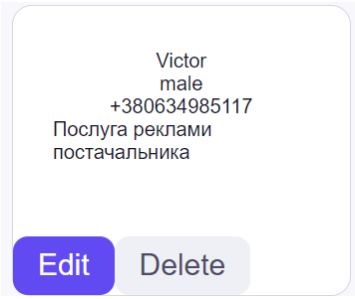
Крок три: створення запиту



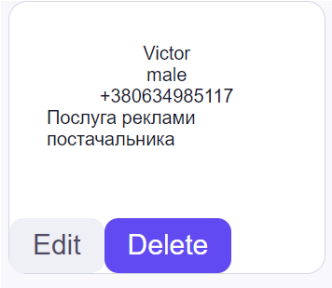
Крок чотири: перегляд запиту



Крок п’ятий: Редагування запиту

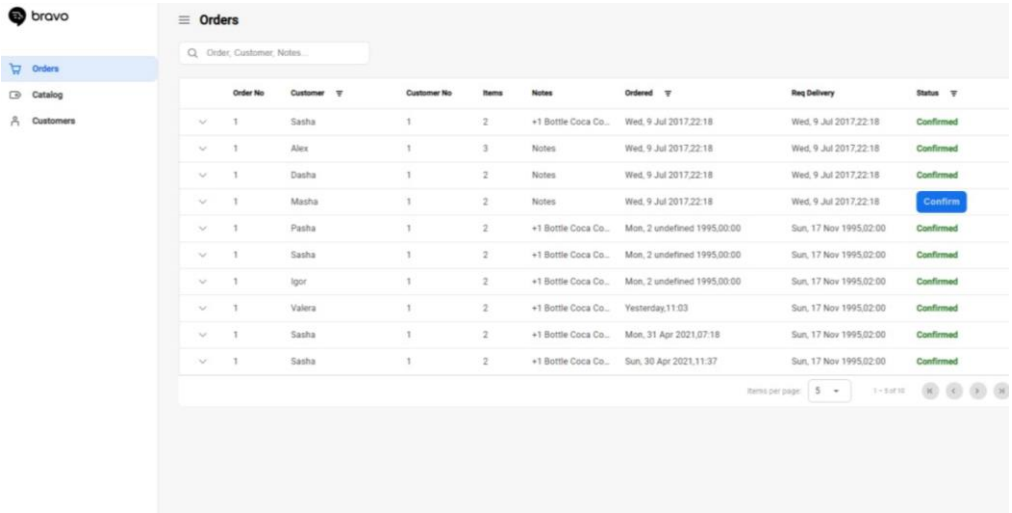


Крок шостий: Видалення запиту

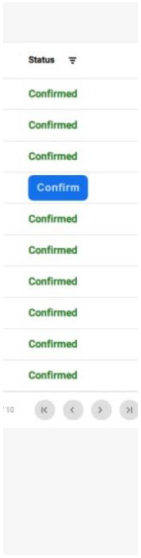


В запити також вказуємо контактний номер, для того, щоб адмін зміг з вами звязатися і обговорити деталі.

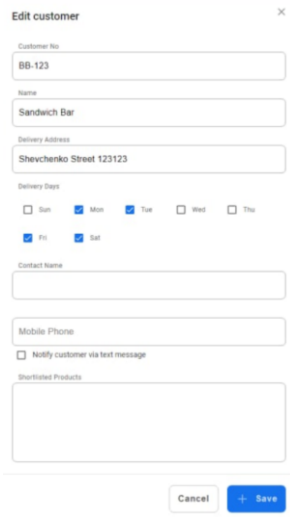
Також додана функція перегляду замовлень



Статус замовлення



Обновлена форма замовлення



3.4. Технічне та системне забезпечення розробки

Даний додаток абсолютно безпечний тому, як всі паролі захошовані за допомогою Node js + Express

```
let hashedPassword;
try {
  hashedPassword = await bcrypt.hash(password, 12);
} catch (e) {
  return next(
    new HttpError("Something went wrong, user not registered", 500)
  );
}
```

Також всі домовленності конфіденціальні, які оброблюються по телефону.

3.4.1 Обгрунтування технічних засобів

Всі вище використані інструменти для розробки є передовими на теперешній час. Також використовувати та маніпулвати складними процесами набагто простіше, ніж наприклад в 2010 році коли популярна була мова програмування PHP.

4. Охорона праці

Мета даного розділу – аналіз обраного приміщення на відповідність нормам охорони праці і безпеки в надзвичайних ситуаціях, оскільки ці питання мають першочергові значення Охорона праці є невід’ємним складником умов трудової діяльності.

Основні положення закріплено в Законі «Про охорону праці», в якому дано їй визначення - це система правових, соціально-економічних, організаційнотехнічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Комфортні та безпечні умови для працівника значно підвищують рівень його ефективності. Крім того, іноді вони виступають важливим фактором при виборі робочого місця, тому роботодавець має бути зацікавлений в створенні сприятливих умов та застосуванні сучасних засобів безпеки для своїх підлеглих. Працівник має право відмовитись від роботи поєднаної з небезпекою для життя або в умовах, що не відповідають нормам законодавства.

4.1. Шум у робочому приміщенні

Шум в робочому приміщенні негативно впливає на працездатність працюючих. Основними фізичними параметрами звуку, що нормуються є інтенсивність, звуковий тиск і частота коливань.

Нормування шумів, ультра- та інфразвуків здійснюється згідно ДСН 3.3.6.037-99. Відповідно до цих норм, рівень шуму не має перевищувати 50 дБ [37]. Можливий список джерел шуму список джерел шуму у нашому приміщенні:

- система охолодження ПЕОМ;
- принтер під час операцій друку.
- шум вуличного транспорту;
- кондиціонер

• система охолодження серверу;

Сумарний рівень інтенсивності звуку можна розрахувати за формулою: де T – робочий час протягом дня; ti – час надходження звуку від і-го джерела; Li – рівень звукового тиску і-го джерела. В (табл. 4.3) наведені джерела шуму, рівень звукового тиску та час дії протягом робочого дня 71 Таблиця 4.3 – Джерела шуму Джерело шуму Рівень шуму La, дБА Час дії шуму t, год Кількість джерел шуму N Зовнішній шум 35 8 1 Кондиціонер 34 4 1 Струменний принтер 60 0.5 1 Система охолодження ПК 32 8 2 Система охолодження серверу 40 8 1 Визначимо Lекв. еквівалентний рівень шуму за 8 робочих годин: Lекв = 10 ∗ lg 1 T ∑ ti ∗ 100.1∗La == 10 · lg 1 8 · (103.5 + 103.4 + 106 + 2 ∗ 103.2 + 104) ≈ 41 дБА Отже, максимально можливий рівень звукового тиску та рівень звуку на робочих місцях відповідає вимогам, так як в приміщеннях управління та робочих кімнатах допустимий еквівалентний рівень звуку менше норми.

4.2. Аналіз освітлення

Для приміщень, в яких робочі місця обладнано ПЕОМ, важливо організувати правильні умови освітлення. Нормування умов освітлення здійснюється згідно будівельних норм [38]. Освітлення приміщення здійснюється за допомогою штучного та природного освітлення. Згідно плану приміщення рис. 4.1, кабінет, який ми аналізуємо, має 3 вікна з лінійними розмірами: ширина – 1,6 м, висота – 1,8 м, відповідно площа кожного вікна – 2,88 м .

Вікна мають регульовані пристрої для відкривання та обладнані жалюзями з можливістю захисту працюючих від прямого попадання сонячних променів і регулювання рівня освітленості в приміщенні. 72 Стіни обклеєні світлими шпалерами, стеля білого кольору, у якості підлогового покриття використаний матовий ламінат з коефієнтом відбиття 0,3- 0,4.

Відблискування поверхонь обмежується за рахунок правильного вибору світильників та розташування робочих місць відносно джерел освітлення.

Яскравість відблисків на сучасних моніторах не перевищує 35 кд/м2 В досліджуваному приміщенні використовується система загального рівномірного штучного освітлення.

Мається два ряди світильників Л201Б 2х40- 0.3, у кожному з яких знаходиться по чотири лампи типу ЛБ-40. Їх технічні характеристики: • потужність – 40 Вт; • напруга на лампі – 103 В; • світловий потік: номінальний – 3120 лм, мінімальний – 2810 лм; • довжина лампи: без штирків – 1199.4 мм, із штирками – 1213.6 мм; • діаметр – 40 мм. План освітлення наведений на (рис. 4.2) Рисунок 4.2 – План освітлення 73 Освітлення достатньо рівномірно розподілено в приміщенні, завдяки комбінації штучного та природнього освітлення, уникаються різкі тіні, а системи його регулювання забезпечують такий стан протягом усього дня

4.3 Електробезбепа

Проаналізуємо приміщення на можливість ураження персоналу електричним струмом. Визначимо групу електронебезпечності даного приміщення.

Ознаки підвищеної небезпеки ураження електрострумом: 75

- наявність вологості;
- наявність температури більш ніж 35 ºС;
- наявність струмопровідного пилу;
- наявність струмопровідної підлоги;
- можливість одночасного дотику до корпусів чи струмопровідних елементів та до елементів, що мають зв’язок з землею. Ознаки особливої небезпеки ураження електрострумом:
- наявність особливої вогкості;
- наявність хімічно активного середовища.

Наше приміщення не має жодної ознаки особливої або підвищеної небезпеки ураження персоналу струмом. Тому за групою електронебезпечності воно відноситься до приміщень без підвищеної небезпеки ураження струмом. Споживачі електроенергії: 2 ПЕОМ, 2 дисплеї, 1 принтер, 4 світильники, 1 кондиціонер та 1 сервер. Кожне робоче місце

обладнане 4-ма розетками по 220 В, окремо існують розетки для кондиціонеру та серверу.

Всі прилади використовують саме цю напругу. Усі кабелі ізольовані. Заземлені конструкції захищені діелектричними сітками від випадкового дотику. Усе електроустаткування має апаратуру захисту від струму короткого замикання.

Лінія електромережі для живлення ЕОМ та периферійних пристроїв ЕОМ виконується як окрема групова три-провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників.

При виконанні робіт по ремонту і обслуговуванню ПЕОМ обслуговуючий персонал зобов'язаний керуватися “Правилами техніки безпеки при експлуатації електроустановок споживачами".

До роботи не допускаються особи, які не пройшли навчання з техніки безпеки. Даний кабінет задовольняє вимоги щодо електробезпеки у приміщенні, в якому встановлені ЕОМ, відображені в НПАОП 0.00-1.28-10.

Список літератури

1. Т. Пратт, М. Зелкович. Языки программирования. Разработка и реализация. 4-и издание. 688 с.
2. Н. Джосьютис. C++ стандартные библиотеки. 729с.
3. Д. Хорвиц. Unix системы от проєктирования до сопровождения Platinum Edition. 587 с.
4. У.Р. Стивенс. Unix разработка сетевых приложений. 1085 с.
5. Марти Холл, Лерри Браун. Программирование для WEB. библиотека профессионалов. 1260 с.
6. Джеймс Ф. Куроуз, Кит В. Росс. Компьютерные сети. Многоуровневая система интернета. 2-е издание. 764 с.
7. Роберт У. Себеста. Основные концепции языков программирования. 5-е издание. 668 с.
8. Ю. Вахалия. Unix изнутри. 843 с.
9. Дэвид Дж. Круглински, Скотт Уингоу, Джордж Шеферд. Программирование на Visual C++ 6.0. 5-е издание. 815 с.
10. А. Якобсон, Г. Буч, Дж. Рамбо. Унифицированный процесс разработки программного обеспечения. 492 с.
11. Основы программирования. 285 с.
12. Гради Буч. Объектно-ориентированное программирование. 514 с.
13. Rational XDE для Visual Studio .NET. 297 с.
14. Гради Буч. Объектно-ориентированное программирование с примерами применения на C++. 2-е издание. 558 с.
15. А.К. Гультяев. Microsoft Project 2002 русифицированная версия. Управление проєктом. 591 с.
16. И. Кузнецов. Создание отчетов в Crystal Reports учебный курс. 540 с.
17. Майкл Кэй. XSLT. Справочник программиста. 2-е издание.1008 с.
18. Брайен А. Уайт. Управление конфигурацией программных средств. Практическое руководство по Rational ClearCase. 265 с.
19. Ольга Здир. Microsoft Word 2003 русская версия. Учебный курс. 348 с.

20. Евангелос Петрусос Visual Basic .NET. 928 с.
21. В.А. Биллиг, И.Х. Мусикаев. Visual C++. 4 версия. 326 с.
22. Стивен Тилберт, Билл Маккарти. Самоучитель Visual C++. 490 с.
23. Альмонах программиста. Том 3. 311 с.
24. О.В. Співаковський, В.А. Крекнін. Лінійна алгебра. 144 с.
25. Хемди А. Таха. Исследование операций. 7-е издание. 901 с.