МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ
на здобуття освітнього ступеня бакалавра

на тему: «**РОЗРОБКА ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ СКЛАДСЬКИМИ ЗАПАСАМИ: МОДУЛЬ АДМІНІСТРАТОР»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Погонець Назарій Ігорович

Керівник:
Красюк Богдан Віталійович

Рецензент:
Гаврильчик Леонід Сергійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)

Протокол № ____ від « ___ » _____ 2022 р.

Острог, 2022

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*

Кафедра: *економіко-математичного моделювання та інформаційних технологій*

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА

«____» _____20__ р.

ЗАВДАННЯ

на кваліфікаційну роботу/проєкт студента

Погонця Назарія Ігоровича

1. Тема роботи Розробка додатку для автоматизації управління складськими запасами: модуль адміністратор.

керівник роботи/проєкту Красюк Богдан Віталійович.

Затверджено наказом ректора НаУОА від 29 жовтня 2021 року №110.

2. Термін здачі студентом закінченої роботи/проєкту: 03 червня 2022 року.

3. Вихідні дані до роботи/проєкту: постановка задачі, аналіз аналогів, вихідні дані.

4. Перелік завдань, які належить виконати: опис предметного середовища; огляд наявних аналогів; постановка задачі; опис архітектури рішення, що розробляється; опис коду та інтерфейсу додатку; тестування.

5. Перелік графічного матеріалу: UML-діаграма додатку, схема розподілу роботи між студентами.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Красюк Б. В.	01.12.2021р.	01.12.2021р.
2	Красюк Б. В.	01.12.2021р.	01.12.2021р.
3	Красюк Б. В.	01.12.2021р.	01.12.2021р.

7. Дата видачі завдання: 01.12.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Постановка технічного завдання	до 01.12.2021	
2.	Розробка архітектури проекту, погодження функціоналу	до 20.12.2021	
3.	Розробка бази даних	до 08.01.2022	
4.	Розробка макетів інтерфейсу веб застосунку	до 24.02.2022	
5.	Розробка серверної частини проєкту	до 15.05.2022	
6.	Розробка клієнтської частини проєкту		
7.	Попередній захист кваліфікаційної роботи/проєкту	до 01.06.2022р.	
8.	Здача кваліфікаційної роботи/проєкту на кафедру	03.06.2022 р.	

Студент: _____ Назарій ПОГОНЕЦЬ

Керівник кваліфікаційної роботи/проєкту: _____ Богдан КРАСЮК

АНОТАЦІЯ

кваліфікаційної роботи/проєкту
на здобуття освітнього ступеня бакалавра

Тема: Розробка додатку для автоматизації управління складськими запасами: модуль адміністратор

Автор: Погонець Назарій Ігорович

Науковий керівник: Красюк Богдан Віталійович

Захищена «……»…… 20__ року.

Короткий зміст праці:

Кваліфікаційна робота/проєкт на тему « Розробка додатку для автоматизації управління складськими запасами: модуль адміністратор» присвячена розробці Web-додатку, створеного за допомогою мови програмування JavaScript. Актуальність обраної для кваліфікаційної роботи теми, пов’язана з потребою підприємств у недорогому програмному забезпеченні, яке спростить роботу для працівників, та створить зручність для клієнтів. У процесі роботи використано такі сучасні засоби як: мова програмування JavaScript, фреймворк Vue.js, система керування базами даних MySQL, інтегроване середовище розробки Microsoft Visual Studio Code.

Qualification work on "Development of an application for automation of warehouse inventory management: administrator module" is devoted to the development of a Web-application created using the JavaScript programming language. The relevance of the topic chosen for the qualification work is related to the need of companies for inexpensive software that will simplify the work for employees and create convenience for customers. In the process of work such modern tools were used as: JavaScript programming language, Vue.js framework, MySQL database management system, Microsoft Visual Studio Code integrated development environment.

_____

ЗМІСТ

ВСТУП

В даний час управління будь-якою діяльністю неможливо без аналізу великого обсягу інформації та її обробки за допомогою комп'ютерів. Використання обчислювальної техніки в різних областях діяльності людини пройшло великий шлях, який визначався не тільки розвитком власне техніки, а й розвитком принципів і методів обробки інформації як з точки зору областей застосування, так і з точки зору широти використання.

Зі створенням в 80-х роках персональних комп'ютерів відбулося не тільки збільшення комп'ютеризованих робочих місць, а що більш важливо, зміна вимог до програмного забезпечення, яке використовувалося в сфері управління та інших. Програмне забезпечення тепер не повинно вимагати спеціально підготовленого оператора і має бути зрозуміло фахівцеві в предметній області, який користується комп'ютером як інструментом.

Крім того, інформація, з якої ми працюємо тепер, розподіляється між різними комп'ютерами і для доступу до «чужих» даних використовуються локальні мережі, які прийшли на зміну багатотермінальним системам.

Ще одним важливим, а останнім часом, можливо, найбільш важливим аспектом використання персональних комп'ютерів стало розвиток глобальних мереж і їх використання не в режимі пошти, а робота в режимі реального часу. Завдяки розвитку телекомунікацій і засобів зв'язку став можливим доступ до величезних накопичених за століття знань з використанням сучасних інформаційно-пошукових систем. Цей аспект діяльності надзвичайно важливий у науковій та навчальній роботі, підвищенні кваліфікації. Ринок комп’ютерної індустрії є чи не наймасштабнішою частиною світового ринку.

У зв’язку з карантинними обмеженнями та на жаль військовим станом в країні дуже постраждала торгівля звичайних магазинів, однак, карантин та війна масово повели бізнес в інтернет. А економічна криза, що розгоряється тільки підігріває цей

процес. Кількість онлайн торговців зростає з кожним днем і схоже, що в найближчому майбутньому ця тенденція збережеться. Відповідно торговцям потрібно продавати свої товари, які будуть розміщуватися на складах, а наша програма в свою чергу контролюватиме продукцію на складах.

Створили CRM систему для контролю продукції на складах підприємства. Проект має назву «**MyWarehouse**». Ми використали для front-end частини Vue, так як ми хотіли створити SPA(single page application) web-додаток (Односторінковий додаток — це веб-додаток або веб-сайт, який використовує єдиний HTML-документ як оболонку для всіх веб-сторінок.)

Обрали саме цей фреймворк так як він є найсучаснішим та активно розвивається. Також він має готову збірку Vue-cli, що спрощує нам як програмістам роботу з налаштуванням проекту,а саме з Web-паком.

Також при ініціалізації проекту, ми вирішили вибрати з запропонавоного vue cli наступні параметри, а саме для написання стилів SCSS - так як вважаємо його зручним та Vuex для того, щоб злегкістю працювати з локальною і динамічною БД, яка добре підтримується на Vue. А також для взаємодії з нашим сервером вирішили використати axios, так як він має чудову базу методів, для взаємодій з сервером.

Для написання back-end частини ми вирішили використовувати Node js. Виришили вибрати саме його, так як вважали його для нас найзручнішим. Для створення сервера вирішили використати Koa js. Koa надає мінімальний інтерфейс для створення програм, має багато різних бібліотек, які з легкістю підключаються та використовуються з ним. Це дуже маленька структура, яка надає необхідні інструменти для створення програм та досить гнучка. Саме за домогою Koa ми створили нашу Api. Базу даних вирішили створювати SQL`овську, а саме MySql. За допомогою пакета MySql2 ми злегкістю змогли підключити наше АПІ до бази даних, та злегкістю отримувати з нею дані, використовуючи Koa. Також для зручності постворювали на back частині моделі таблиць, які лежали в нашій БД, так як більшість запитів, ми робили використовуючи Koa, а для його роботи, необхідні були ці моделі.

Сама архітектура складалась з моделі та маршрутера, в якому створювалась логіка запитів до БД.

Для перевірки наших запитів, ми використовували PostMan, так як ним легко і швидко можна було перевірити коректність нашого АПІ.

Адміністратор матиме змогу створювати склади по Україні, добавляти та видаляти продукцію на складах, переглядати / підтверджувати реєстрацію нових клієнтів, погоджувати, або ж відмовляти їх замовлення, та блокувати небажаних користувачів системи.

В свою чергу можливості ролі користувача набагато скромнішими. Користувач після реєстрації в системі має змогу лише замовляти потрібну йому продукцію з обраного ним складу.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного становища

Розробку програмного забезпечення було розділено на декілька етапів- back-end та front-end. Кожен з цих процесів вимагав використання певного програмного середовища.

Для розробки серверної частини та інтерфейсу використовувалась програма Visual Studio Code

Visual Studio Code

Visual Studio Code (VS Code) - це легкий, але потужний редактор вихідних кодів, який працює на робочому столі та доступний для Windows, macOS та Linux. Вона підтримує JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов.

Редактори коду дозволяють зручно та легко розробляти інтерфейси програм, чи веб-сайтів, мають настроюваний зовнішній вигляд, підсвічування синтаксису, можливість додавання сторонніх плагінів для покращення взаємодії з користувачем, що дає високі переваги над типовими текстовими процесорами.

Переваги Visual Studio Code:

- Безкоштовний.
- Велика кількість плагінів.
- Динамічний набір тексту.
- Підтримка Git.
- Легка вага – редактор не потребує значних ресурсів операційної системи.

GitHub

Так як розробка проекту виконувалась у команді, для зручності роботи було вибрано систему контролю версій GitHub.

GitHub – один з найбільших веб-сервісів для спільної розробки програмного забезпечення та контролю версій.

Весь код ми зберігали на конкретному репозиторії, до якого мав доступ кожен з учасників команди.

Тестування Api

В якості програмного забезпечення для тестування серверної частини та окремих функцій було вирішено використати Postman.

Postman- це свого роду швейцарський ніж, який дозволяє створювати і виконувати запити.

Postman було обрано, оскільки він є найпопулярнішим із всіх можливих рішень.

Переваги Postman:

• інтуїтивно-зрозумілий і простий у використанні, не вимагає якогось складного налаштування або знання мов програмування; безкоштовний;

• підтримує різні API (REST, SOAP, GraphQL);

• розширюється під будь-які потреби за допомогою Postman API;

• легко інтегрується в CI/CD за допомогою Newman – консольної утиліти для запуску тестів;

• запускається на будь-яких ОС;

• підтримує ручне та автоматизоване тестування;

• зібрав навколо себе велике ком'юніті, де можна знайти відповіді будь-які питання.

Тестувальнику цей інструмент дозволяє:

• надсилати запити та отримувати відповіді;

• зберігати запити до папок та колекцій;

• змінювати параметри запитів;

• змінювати оточення (dev, test, production);

• виконувати автотести, використовуючи Collections runner, зокрема за розкладом;

• імпортувати та експортувати колекції запитів та набори тестів, щоб обмінюватися даними з колегами.

За допомогою Postman тестувальник може:

- складати та надсилати HTTP-запити до API;
- створювати колекції (набір послідовних запитів) та папки запитів для скорочення часу тестування;
- змінювати параметри запитів (наприклад, ключі авторизації та URL);
- змінювати оточення для запитів (наприклад, на тестовому стенді, локально або на сервері);
- додавати під час виклику API контрольні точки (фіксацію моменту передачі);
- проводити автоматизоване тестування API з колекції запитів за допомогою Collection Runner.

Для роботи із серверами програма використовує протокол HTTP. Тестувальник відправляє тестові запити від клієнта на сервер та отримує відповідь, чи є помилка в роботі API. Postman доступний як програми для Windows, Linux і macOS, а також у web-інтерфейсі (для його роботи потрібно встановити програму Postman Desktop Agent).

Методи Postman

Найчастіше у роботі API використовується архітектура RESTful. У цій архітектурі є чотири стандартні методи запитів до серверів HTTP:

POST — створення об'єкта та надсилання даних на сервер;

GET - отримання інформації із сервера;

PUT – оновлення об'єкта;

DELETE – видалення об'єкта.

У Postman можна протестувати запити за кожним методом: його потрібно вибрати вкладці запиту. Після надсилання запиту тестувальник отримує відповідь як код статусу HTTP. Усього таких статусів 40 у п'яти категоріях; кожен код допомагає зрозуміти, чи правильно працює API.

1.2 Стек технологій:

Back-end:

1. Node.js
2. Koa.js
3. MySql

Front-end:

1. Vue (Vue cli в ролі зборки)
2. Vuex
3. SCSS
4. Axios

Node.js - це серверна платформа, обгорнута мовою JavaScript для створення масштабованих програм, керованих подіями. Це бентежить навіть досвідчених програмістів, оскільки традиційне середовище JavaScript завжди було на стороні клієнта - у браузері користувача або в додатку, який спілкується з сервером. JavaScript не розглядався, коли мова йде про сервер, який відповідає на запити клієнта, але саме це надає Node.js.

Node.js написаний не на JavaScript (він написаний на C ++), але він використовує мову JavaScript як інтерпретаційну мову для обробки запиту / відповіді на стороні сервера. Іншими словами, Node.js запускає автономні програми JavaScript. Перевага полягає в тому, що програмісти можуть використовувати свої поточні, хоч і на стороні клієнта, знання програмування і починати кодування з Node.js набагато легше.

Що таке Koa?

Koa надає мінімальний інтерфейс для створення програм. Це дуже маленька структура (600 LoC), яка надає необхідні інструменти для створення програм та досить гнучка. На npm є безліч модулів для Koa, які можна безпосередньо підключити до нього. Koa можна розглядати як ядро express.js без жодних накручень.

Чому Koa?

Koa має невелику площу (600 лок) і є дуже тонким шаром абстракції над вузлом для створення додатків на стороні сервера. Він повністю підключається і має величезну спільноту. Це також дозволяє нам легко розширювати Koa і використовувати його відповідно до наших потреб. Він побудований за допомогою передової технології (ES6), яка дає йому перевагу перед старими платформами, такими як Express. Pug (раніше відомий як Jade) - це коротка мова для написання HTML-шаблонів.

Виробляє HTML

Підтримує динамічний код

Це одна з найпопулярніших шаблонних мов, що використовуються в Koa.

MySQL – це популярний сервер баз даних, що використовується у різних додатках. SQL означає мову структурованих запитів - (S)tructured (Q)uery (L)anguage, яку MySQL використовує для комунікації з іншими програмами. Крім того, MySQL має власні розширені функції SQL для того, щоб забезпечити користувачам додатковий функціонал. У цьому документі ми розглянемо як провести початкову установку MySQL, налаштувати бази даних та таблиці та створити нових користувачів.

Vue

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків монолітів, Vue створено придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня представлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. Технічно Vue.js визначена як ViewModel шар шаблону MVVM. Вона з'єднує модель

і представлення у двосторонньому зв’язуванні даних. Мета надати переваги швидких зв'язувань даних і складних уявлень компонентів з API, як прості так і зрозумілі. Бібліотека не є повномасштабним фреймворком, вона є всього лише рівнем уявлення. Можна використовувати її як окремо, для швидкого прототипування, або змішувати і поєднувати з іншими бібліотеками для кастомізації інтерфейсу користувача.

Vuex

Vuex - патерн керування станом + — бібліотека для управління станом вебзастосунків, вона досить проста та легко інтегрується з Vue. Він служить централізованим сховищем даних для всіх компонентів додатка з правилами, що гарантують, що стан може бути змінено лише передбачуваним чином. Vuex інтегрується з офіційним розширенням vue-devtools (opens new window), надаючи "з коробки" такі просунуті можливості, як "машину часу" для налагодження та експорт/імпорт зліпків стану даних.

SASS — це метамова на основі CSS, призначена для збільшення рівня абстракції CSS коду та спрощення файлів каскадних таблиць стилю.

SASS надає більше можливостей і свободи при написанні CSS для **створення сайтів**. Це як програмна мова всередині CSS. Ви можете використовувати щось подібне до функцій зі змінними, логічно структурувати ваш код (структурованість стилів і класів).

SASS має два синтаксиса. Новий основний синтаксис відомий як "SCSS" (SassyCSS). Нами було вирішено, використовувати SCSS синтаксис, тому що він більш читабельний і зрозумілий. Синтаксис SCSS нічим не відрізняється від синтаксису CSS. Але при цьому він має масу корисних можливостей, які можуть полегшити життя верстальщика.

Axios

Axios — це сучасний і популярний HTTP-клієнт на основі обіцянок, який ми можемо використовувати для виконання HTTP-запитів. Він підтримує всі сучасні браузери, включаючи IE, і працює в обох браузерах (клієнтах JS) і на платформі Node.js.

Axios має надійний набір функцій:

- Пропонує широкий спектр підтримки браузера, включаючи старі
- Дозволяє користувачам робити XMLHttpRequests з браузерів і HTTP- запити від Node.js для взаємодії з серверами
- Дозволяє перехоплювати запити та відповіді
- Дозволяє користувачам встановлювати час очікування відповіді
- Забезпечує захист від XSRF на стороні клієнта
- Підтримує Promise API
- Дозволяє скасувати запит
- Дозволяє трансформувати дані запитів і відповідей
- Пропонує автоматичне перетворення даних JSON.

Реалізували **middleware's** та **request interceptors**.

Middleware's (Проміжне програмне забезпечення). Служба проміжного програмного забезпечення Axios HTTP Простий сервіс проміжного програмного забезпечення axios HTTP для спрощення перехоплення (і тестування перехоплювачів) HTTP-запитів, зроблених через Axios.

Що це? Це служба HttpMiddlewareService, яка керує стеком програмного забезпечення проміжного шару і підключається до екземпляра axios. Проміжне ПЗ - це просто об'єкти або класи, що складаються з простих методів для різних точок життєвого циклу запиту. Він працює або з глобальними аксіомами, або з локальним екземпляром.

Чому б не використати перехоплювачі? Використання перехоплювачів axios робить код тісно пов'язаним з axios та його складніше тестувати. Цей сервісний модуль проміжного програмного забезпечення: пропонує більше функцій (наприклад, onSync) більш слабкий зв'язок з axios дійсно легко тестувати класи проміжного програмного забезпечення Це покращує читання та можливість повторного використання у стратегії централізованого підключення.

Request interceptors (Запит перехоплювачів).

Загалом, перехоплювачі Axios — це не що інше, як функція, яка викликається для кожного зробленого HTTP-запиту та відповіді, отриманої вашою програмою.

Таким же чином, перехоплювач Axios — це функція, яка викликається Axios для оновлення/трансформації кожного запиту перед його пересиланням та перевірки/фільтрації відповіді перед її поверненням. Коротше кажучи, ви можете перехопити запити або відповіді до того, як вони будуть оброблені до того часу або перехопленні.

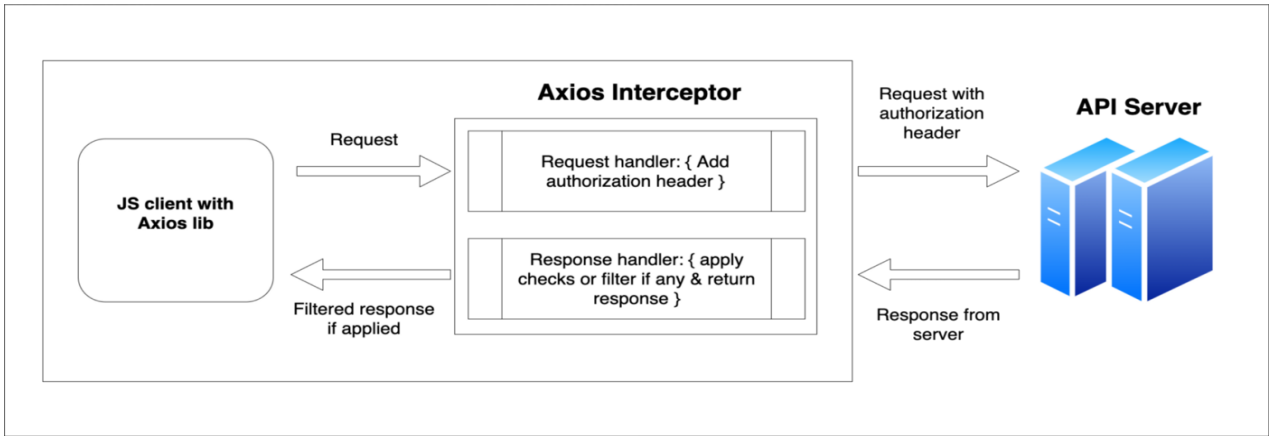


Рис. 1.1 Перехоплення запитів і відповідей за допомогою Axios

SPA

Single Page Application – скорочено SPA, у перекладі російською мовою означає “Додаток однієї сторінки”. Тобто SPA - це web-додаток, розміщений на одній web-сторінці, яка для забезпечення роботи завантажує весь потрібний код разом із завантаженням самої сторінки. Додатки такого типу з'явилися порівняно недавно, з

початком ери HTML5 та SPA є типовим представником додатків на HTML5. Як ми знаємо, HTML5 це щось інше як HTML + CSS3 + JavaScript + [кілька нових тегів]. Таким чином, SPA - це програми, написані на мові JavaScript.

Складові SPA

Принципи будь-якого фреймворку який реалізує парадигму SPA повинні дотримуватися наступних понять та визначень:

- SPA підтримує клієнтську навігацію. Всі "ходіння" користувача по модулях-сторінках однозначно фіксуються в історії навігації, причому навігація при цьому є "глибокою", тобто якщо користувач скопіює та відкриє посилання на внутрішню модуль-сторінку в іншому браузері або вікні, він потрапить на відповідну сторінку.
- SPA розміщується на одній web-сторінці, отже, все необхідне для роботи сайту (порталу) скрипти та стилі повинні бути визначені в одному місці проекту – на єдиній web-сторінці.
- SPA зберігає постійно стан (важливі змінні) роботи клієнта (клієнтського скрипта) в кеші браузера або Web Storage.
- SPA завантажує всі скрипти, що потрібні для старту програми при ініціалізації web-сторінки.
- SPA поступово підвантажує модулі на вимогу.

Якщо коротко, то SPA - це web-програма, розміщена на одній сторінці, яка для забезпечення роботи завантажує всі javascript-файли (модулі, віджиті, контролі і т.д.), а також файли CSS разом із завантаженням самої сторінки.

1.3 Огляд наявних аналогів

1С

Система програм «1С:Підприємство» складається з технологічної платформи (ядра) та розроблених на її основі прикладних рішень («конфігурацій»). Така архітектура системи принесла їй високу популярність, оскільки забезпечує відкритість прикладних рішень, їхню функціональність і гнучкість, короткі терміни впровадження, високу продуктивність, масштабованість від одного до десятків тисяч робочих місць, роботу в режимі «хмарного» сервісу та на мобільних пристроях.

Області застосування

Гнучкість платформи дозволяє застосовувати «1С:Підприємство 8» у найрізноманітніших областях:

автоматизація виробничих та торгових підприємств, бюджетних та фінансових організацій, підприємств сфери обслуговування тощо.

підтримка оперативного управління підприємством;

автоматизація організаційної та господарської діяльності;

ведення бухгалтерського обліку з кількома планами рахунків та довільними вимірами обліку, регламентована звітність;

широкі можливості для управлінського обліку та побудови аналітичної звітності, підтримка багатовалютного обліку;

вирішення завдань планування, бюджетування та фінансового аналізу;

розрахунок зарплати та управління персоналом та інші галузі застосування.

BAS - програма призначена для ведення бухгалтерського, управлінського, фінансового обліку на підприємстві та управління всіма аспектами його діяльності. Особливістю системи є те що вона в основній своїй масі експлуатується на

підприємствах України і має адекватну для свого ринку вартість, європейські системи такого класу як правило коштують у кілька разів дорожче. Незважаючи на невисоку вартість за всіма параметрами програма має конкурентоспроможну функціональність і переваги за рахунок наявності в системі модулів бухгалтерського і податкового обліку які розроблені для всіх країн в яких вона використовується. Завдяки практично монопольному становищу на ринку програма BAS має якісну технічну та методичну підтримку від фірм партнерів яких в Україні налічується близько 500. Розробники програми BAS контролюють якість послуг, що надаються фірмами партнерами, проводять заходи для їх навчання і сертифікації з метою підвищення якості послуг і максимального задоволення бажань замовників. Для користувачів які хочуть впроваджувати програми BAS самостійно існує величезна кількість методичної літератури російською та українською мовою, велика кількість різноманітних навчальних курсів та відеоматеріалів, також є можливість отримати деякі версії програм для навчання і експлуатації безкоштовно.

При створенні програми BAS розробники особливу увагу приділяли технологіям для експлуатації програми засобами мережі Інтернет і тепер крім роботи в режимі розподіленої інформаційної бази (РІБ) з'явилася можливість повноцінної роботи програм BAS в режимі Тонкого клієнта і Web клієнта.

Система BAS добре підходить для споживачів різного розміру бізнесу. Так версія BAS Базова поставляється для малих підприємств з можливістю експлуатації на одному робочому місці і безкоштовним оновленням, підтримкою,. Версія ПРОФ програм BAS має можливість розширяться додатковими робочими місцями на одне, п'ять, десять, двадцять, п'ятдесят, сто, п'ятсот робочих місць, також версія ПРОФ може модифікуватися, розширяться новим функціоналом для максимальної відповідності з потребами організації. Вартість ПРОФ версій програм на 5 робочих місць знаходиться в діапазоні від 11000 - 20000 грн., Щоб оновлювати ПРОФ програму і отримувати консультаційну підтримку необхідно придбати у фірми партнер ІТС (інформаційно-технічне супроводження) вартість на 12 місяців близько 9900 грн. Для великих підприємств фірма розробила систему класу ERP в яку

включений функціонал для управління всіма процесами на підприємстві, вартість системи на 10 робочих місць близько 200000 грн.

Програма BAS може працювати як у файловому режимі бази даних так і Клієнт-серверному варіанті, для компаній які використовують 10 і більше користувачів краще використовувати Клієнт-серверний варіант програми. Для організації роботи програми в Клієнт-серверному варіанті крім ліцензій на Основну програму і додаткові робочі місця необхідно придбати ліцензію на Сервер BAS і вибрати один з декількох варіантів платних або безкоштовних SQL серверів, платні: Microsoft SQL Server, Oracle Database, безкоштовні PostgreSQL, IBM DB2. Робота в Клієнт-серверному варіанті забезпечує більш швидкий доступ до даних і кращий захист інформації в порівнянні з файловим варіантом.

Система BAS підтримує роботу з широким спектром торгового обладнання: фіскальні реєстратори, ваги, ПОСТ термінали, сканери штрих-кодів, датчики обліку відвідувачів.

Програма BAS складається з двох компонентів: Середовище розробки (Конфігуратор) і Конфігурації (прикладні рішення).

1.4 Постановка задачі

Створення CRM системи для контролю продукції на складах підприємства. Проект має назву «**MyWarehouse**». Серверна частина проекту була розроблена за допомогою Node.js, а клієнтська частина за допомогою Vue.js. Для функціонування бази даних даного проекту було використано MySql.

Адміністратор:

- створює нових користувачів і надсилає їм запрошення в систему

- самостійно користувач отримати доступ до системи не може

- адміністратор створює та редагує всю пов’язану з складами та замовленнями інформацію

- адміністратор додає та видаляє користувачів системи, підтверджує або ж відхиляє замовлення

Клієнт:

- виконує реєстрацію в додатку

- отримує інформацію про склади та продукцію

- має можливість замовлення певного продукту з вибраного складу

2. Роль адміністратора, роль клієнта.

а. сторінка клієнта

На сторінці клієнта відображається інформація про його поточні

замовлення (доставки) з описовою інформацією

3. Сторінка для користувачів з інформацією про склади підприємця, з

можливістю перейти на сторінку конкретного складу та створити замовлення.

4. Сторінка для редагування особистого аккаунта.

Була створена наступна UML-діаграма.

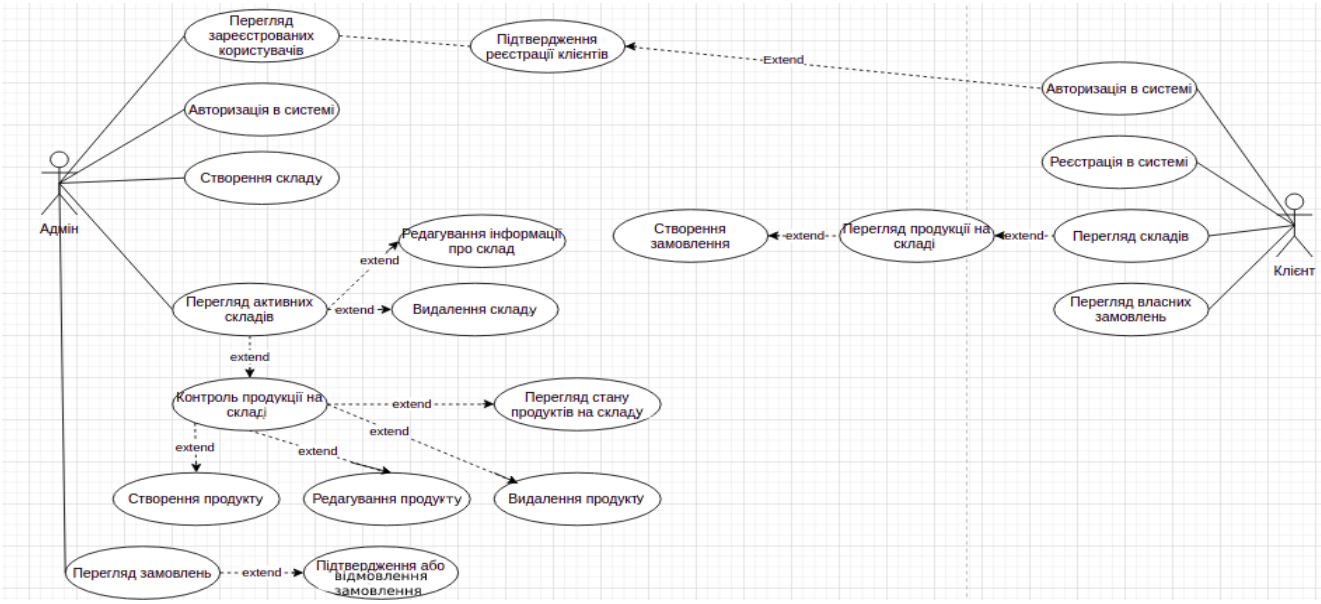
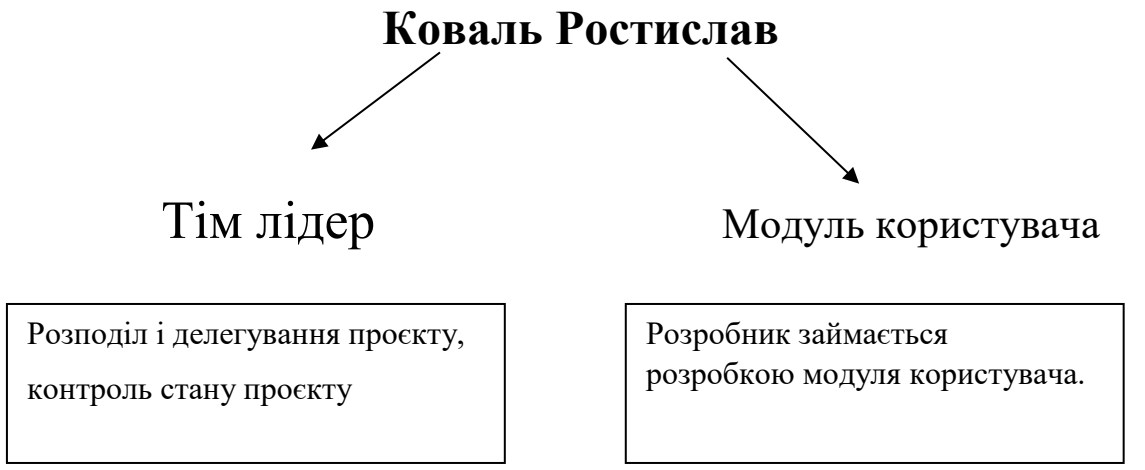
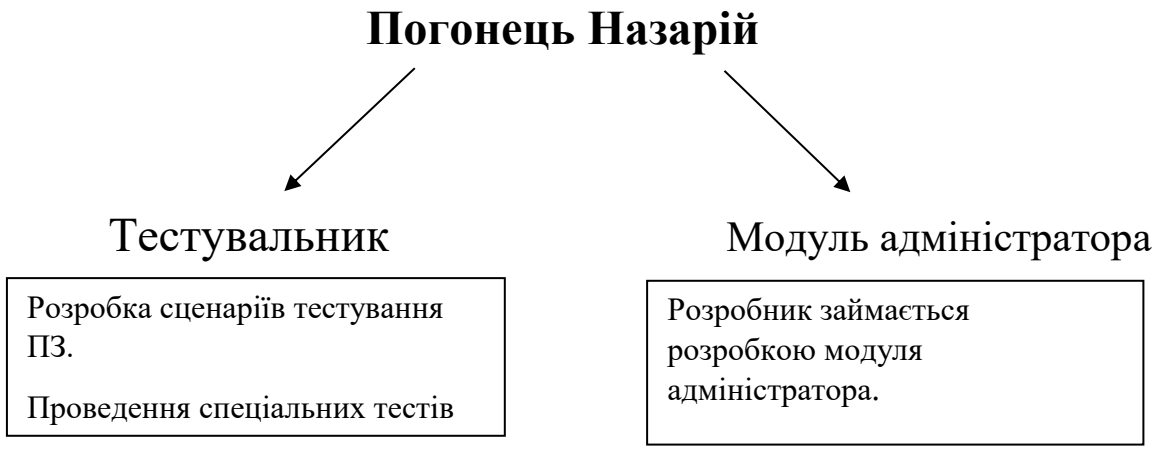


Рис.1.2 UML-діаграма проєкту.

1.5 Відповідальні особи





1.6 Організація розробки

Представимо що ми команда розробників, які взяли замовлення на розробку проекту (в нашому випадку це керівник в лиці університету). На початку розробки, ми обговорили з керівником те, як ми будемо працювати в продовж розробки проекту, в кінці обговорення ми обрали методологію scrum. Познайомимось детальніше про цю методологію.

Скрам — це фреймворк управління, згідно з яким одна чи декілька кросфункціональних команд створюють продукт інкрементами, тобто, поетапно. В команді може бути близько семи людей.

У скрамі є система ролей, подій, правил і артефактів. У цій моделі за створення й адаптацію робочих процесів відповідають команди.

Як працює структура скраму.

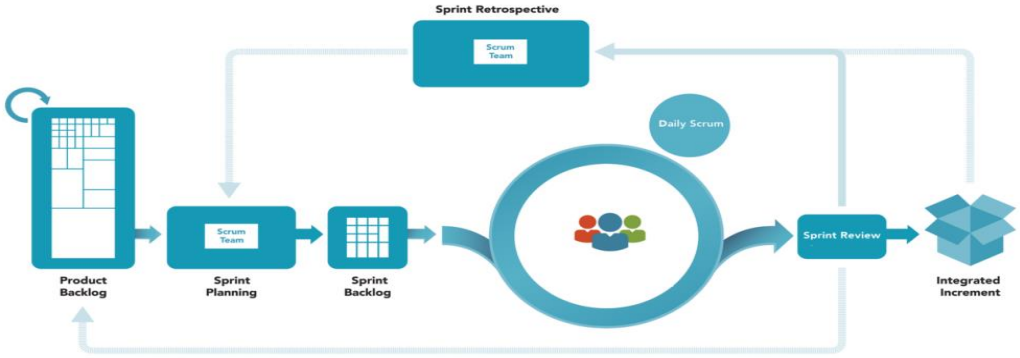


Рис.1.3 схема scrum

Скрам як фреймворк управління проєктами базується на тому, що самоорганізовані команди постачають закінчені продукти у фіксовані терміни, які також називаємо спринтами. Щоб успішно застосовувати скрам, потрібно використовувати його структуру. Вона складається з ролей, подій, правил і артефактів.

Ролі в скрамі

У скрамі існує три ролі, що разом утворюють скрам-команду.

1. Власник продукту — апологет продукту, який повністю розуміє його цінність для бізнесу. Ця людина доносить потреби замовника і стейкхолдерів до розробників, але не відповідає за технічний бік процесу. Власник продукту також відповідає за історії користувачів і визначає їх пріоритетність.
2. Розробники виконують всі технічні задачі. Вони кросфункціональні, їхня кросфункціональність залежить від області роботи. Так чи інакше, розробники

завжди відповідають за створення беклогу спринту, забезпечення якості відповідно до визначення готового, адаптацію плану щодо цілі спринту і власні експертні зони відповідальності.

3. Скрам-майстер виступає фасилітатором роботи скрам-команди. Скрам-майстер допомагає власнику продукту і розробникам виконувати роботу без перешкод і відволікаючих факторів. Уся комунікація людей з-поза команди з командою розробки відбувається через скрам-майстра. (Часом скрам-команди взаємодіють у форматі скраму скрамів, коли скрам-майстри команд мають власні окремі зустрічі).

Події в скрамі

Існує п’ять типів скрам-подій:

1. Спринт (Sprint) — самісіньке серце скраму, де ідеї набувають цінності. Всередині спринту виконується вся робота, необхідна для досягнення цілі продукту, в тому числі планування спринту, дейлі скрами, огляд і ретроспектива спринту.
2. Планування спринту (Sprint Planning) — в ньому беруть участь усі члени скрам-команди. На цій зустрічі відбувається презентація продукту. Кожен член команди може висловитися про те, що цікавить або турбує його. Під час зустрічі встановлюються пріоритети й оцінюються терміни.
3. Дейлі скрам (Daily Scrum) — скрам-події, які проходять щодня під час спринтів. Вони короткі (до 15 хвилин) і призначені для планування денного розкладу розробників. На цих зустрічах обговорюють робочі труднощі, пояснюють незрозумілі історії користувачів. Дейлі є обов’язковим для розробників. Присутність скрам-майстра необов’язкова.
4. Огляд спринту (Sprint Review) — демонстрація діючого продукту, розробленого протягом спринту. Ця подія відбувається наприкінці спринту і призначена в першу чергу для того, щоб детально продемонструвати досягнення стейкхолдерам.

5. Ретроспектива спринту (Sprint Retrospective) — щось схоже на розтин. Це обговорення того, як команда спрацювала протягом спринту, і пошук, як підвищити якість її роботи в майбутньому.

На додаток до цих подій під час спринту команди можуть проводити також уточнення беклогу (Backlog Refinement) — обговорювати елементи беклогу й готуватися до наступного спринту. В рамках цієї зустрічі можна обговорити пріоритетність елементів і розділити елементи беклогу на дрібніші складові.

Артефакти

Артефакти скраму — це робота, яку потрібно виконати, щоб завершити проєкт або спринт. Завдяки їм інформація про проєкт залишається прозорою для всіх, хто над ним працює.

Існує три обов’язкові/основні артефакти у скрамі — беклог продукту, беклог спринту й інкремент. Вони необхідні, щоб постачати програмне забезпечення, яке буде цінним для ваших замовників. Є й необов’язкові артефакти, які, втім, можуть полегшити життя вашої команди (наприклад, берн-даун чати).

Артефакти спринту і їх компоненти — це:

- Беклог продукту — всі необхідні дії, пов’язані з користувацькою і технічною сторонами проєкту.
- Беклог спринту — сукупність всіх задач, які потрібно виконати протягом ітерації спринту. Його отримують із беклогу продукту під час планування спринту.
- Інкремент — це сума всіх елементів беклогу продукту, виконаних під час спринту, а також цінність інкрементів усіх попередніх спринтів. До закінчення спринту новий інкремент має бути готовий, тобто працездатний і відповідний визначеним раніше критеріям готовності.

- Елемент беклогу продукту — частина роботу, яку слід виконати протягом ітерації спринту. Зазвичай розбивається на декілька малих задач.
- Ціль спринту — те, що треба зробити, щоб виконати елемент беклогу продукту.
- Бьорн-даун чат спринту — робота, що залишається до повного виконання задач спринту. Бьорн-даун чат може мати висхідний або низхідний характер, залежно від того, з чим стикається команда при виконанні задачі. Він слугує не звітом про прогрес команди, а методом подолання труднощів і підтримки активності.

Висновки до розділу 1

Отже, ми ознайомилися з предметним середовищем. Визначилися з конкретними завданнями які будемо виконувати. Вирішили що дана програма буде досить актуальна у сьогодення, так як у зв’язку з карантинними обмеженнями та на жаль військовим станом в країні дуже постраждала торгівля звичайних магазинів, однак, карантин та війна масово повели бізнес в інтернет. А економічна криза, що розгоряється тільки підігріває цей процес. Кількість онлайн торговців зростає з кожним днем і схоже, що в найближчому майбутньому ця тенденція збережеться. Ознайомилися з аналогами програми. Так, як малі підприємства не мають великих коштів для оплати ліцензій програм 1С та BASS, ми можемо запропонувати свій варіант, який вони будуть використовувати в своїх потребах.

Створили UML-діаграму проєкту для того, щоб легше орієнтуватися з розробкою проекту. Бачимо головний функціонал ролі адміністратора та клієнта.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Проектування системи (проектування бази даних)

Створення бази даних

Використали SQL код для створення бази даних. Основні властивості бази даних: orders, products, users, warehouses.

Проміжні властивості: id, title, count, price, status, recipient_city, warehouse_id, owner_id, data, product_id, charasterictic.

Опис всіх властивостей знаходиться в папці Models.

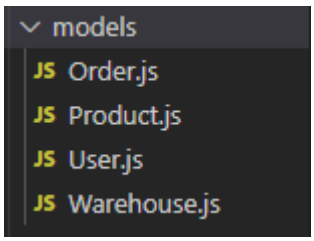


Рис. 1.4 Перелік властивостей

Властивість **order**

Має такі поля:

id – номер замовлення.

title – назва продукту.

count – сума.

price – ціна

status – статус замовлення.

Respient_city – місто одержувача

Warehouse_id – ідентифікатор складу

Data- дата замовлення

Product_id- номер продукту

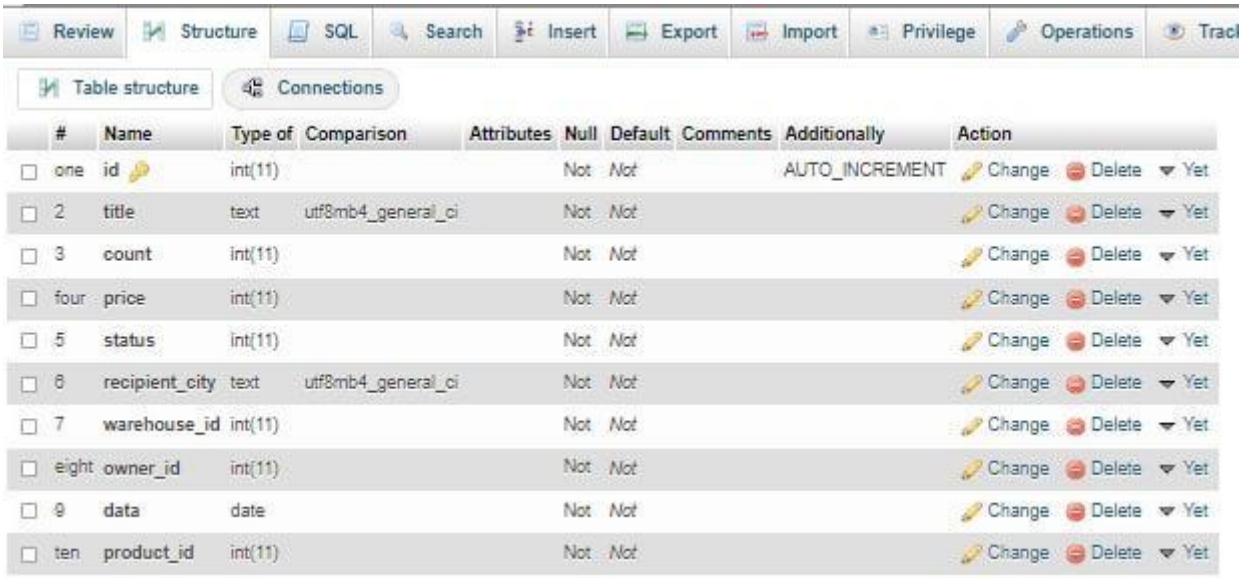


Рис. 1.5 Структура таблиці order.

Властивість **products**

Має такі поля:

id – номер замовлення.

title – назва продукту.

count – сума.

price – ціна

Warehouse_id – ідентифікатор складу

Рис. 1.6 Структура products

Властивість **users**

Має такі поля:

Id- номер замовлення.

Email- електронна пошта.

Password- пароль.

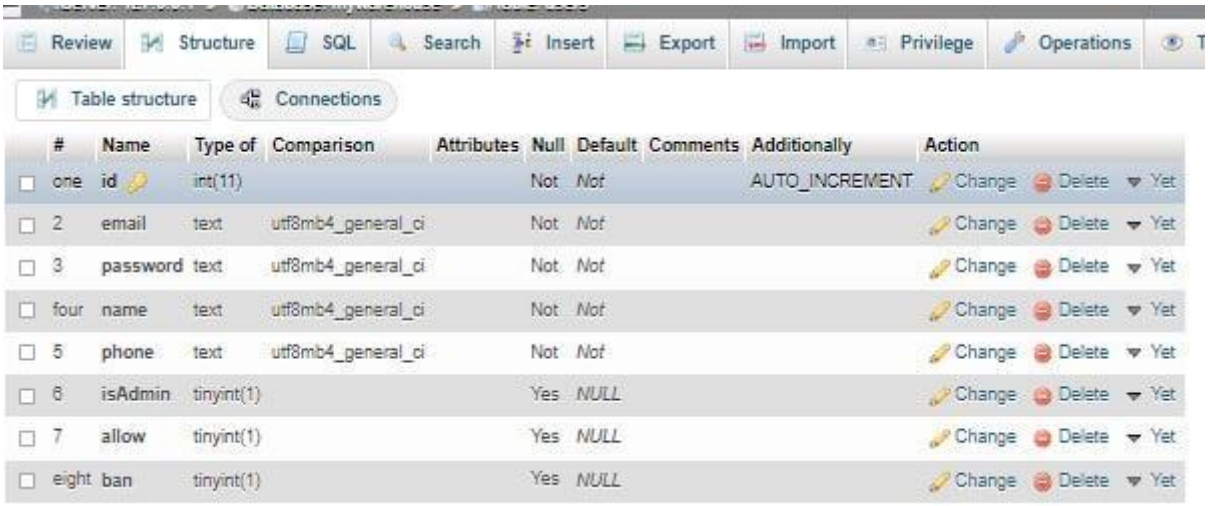
Phone- номер телефону.

Name- ім’я.

isAdmin- надавання прав адміна.

Allow- дозволено.

Ban- заборонено



#	Name	Type of	Comparison	Attributes	Null	Default	Comments	Additionally	Action
one	id	int(11)			Not	Not		AUTO_INCREMENT	Change Delete Yet
2	email	text	utf8mb4_general_ci		Not	Not			Change Delete Yet
3	password	text	utf8mb4_general_ci		Not	Not			Change Delete Yet
four	name	text	utf8mb4_general_ci		Not	Not			Change Delete Yet
5	phone	text	utf8mb4_general_ci		Not	Not			Change Delete Yet
6	isAdmin	tinyint(1)			Yes	NULL			Change Delete Yet
7	allow	tinyint(1)			Yes	NULL			Change Delete Yet
eight	ban	tinyint(1)			Yes	NULL			Change Delete Yet

Рис. 1.7 Структура users

Властивість **warehouses**

Має такі поля:

Id – номер

Town- місто

Region- область

Rent- орендна плата

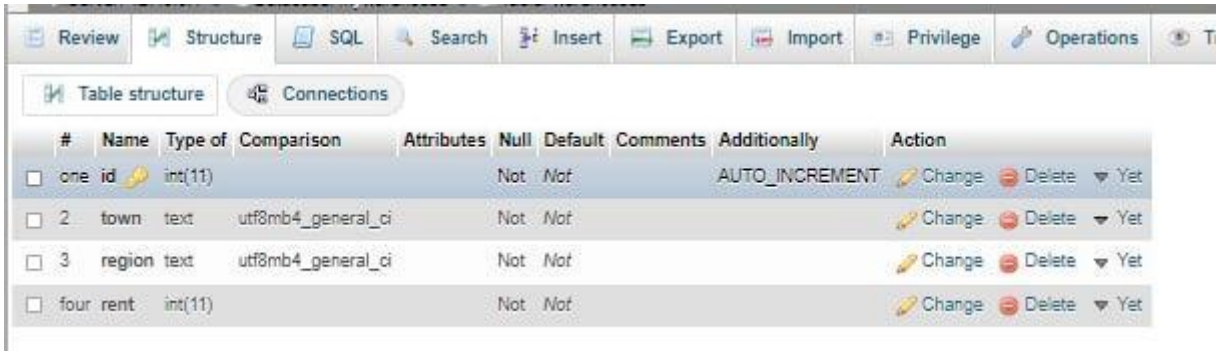


Рис. 1.8 Структура warehouses.

З’єднуємо нашу базу даних з основною частиною проекту за допомогою MySQL2.

Проект MySQL2 є продовженням MySQL-Native . Код аналізатора протоколу було переписано з нуля, а API змінено, щоб відповідати популярним mysqljs/mysql . Команда MySQL2 працює разом з командою mysqljs/mysql , щоб виділити спільний код і перемістити його в організацію mysqljs . MySQL2 здебільшого сумісний із API з mysqljs і підтримує більшість функцій. MySQL2 також пропонує ці додаткові функції. З MySQL2 ми також отримуємо підготовлені оператори. З підготовленими операторами MySQL не потрібно готувати план для кожного запиту, це призводить до кращої продуктивності. Використання пулів підключень - пули підключень допомагають скоротити час, витрачений на підключення до сервера MySQL, за рахунок повторного використання попереднього з’єднання, залишаючи їх відкритими замість того, щоб закриватися, коли ми закінчимо з ними. Це покращує затримку запитів, оскільки ми уникаємо всіх накладних витрат, пов’язаних із встановленням нового з’єднання.

Висновок до розділу 2

Отже в цьому розділі ми проаналізували предметне середовище, для себе дізналися як правильно спілкуватися з майбутніми користувачами нашого продукту, та як в майбутньому обслуговувати їх. Також створили базу данних SQL. Виділили переваги цієї мови - незалежність від конкретної СУБД, Наявність стандартів, Декларативність, а також недоліки - невідповідність реляційній моделі даних (рядки що повторюються, невизначені значення (null), явна вказівка порядку стовпчиків зліва направо), складність, відхилення від стандартів, складність роботи з ієрархічними структурами.

Розділ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

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

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.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та

правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

JavaScript має низку властивостей об’єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об’єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам, — функції як об’єкти першого класу, об’єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає мові додаткову гнучкість.

Vue - це прогресивний фреймворк для створення інтерфейсів користувача. На відміну від фреймворків-монолітів, Vue створено придатним для поступового впровадження. Його ядро насамперед вирішує завдання рівня представлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових програм (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

3.2 Опис програмної реалізації

Front-end

Single-page application, SPA

Односторінковий застосунок (*single-page application, SPA*), також відомий як **односторінковий інтерфейс** (*single-page interface, SPI*) це вебзастосунок чи вебсайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

В односторінковому застосунку весь необхідний код - HTML, JavaScript, та CSS - завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з вебсервером.

Розуміння основи MVVM. скорочено від Model-View-ViewModel. Це, по суті, покращена версія MVC. MVVM повинен абстрагувати стан і поведінку View, давайте відокремимо інтерфейс користувача View і бізнес-логіку. Звичайно, все це було зроблено ViewModel для нас: він може отримати дані з Моделі і допомогти розібратися з бізнес-логікою, пов'язаною з View, через необхідність відображати контент.

Говорячи про цю модель MVVM, ми маємо поговорити про інфраструктуру MVC.

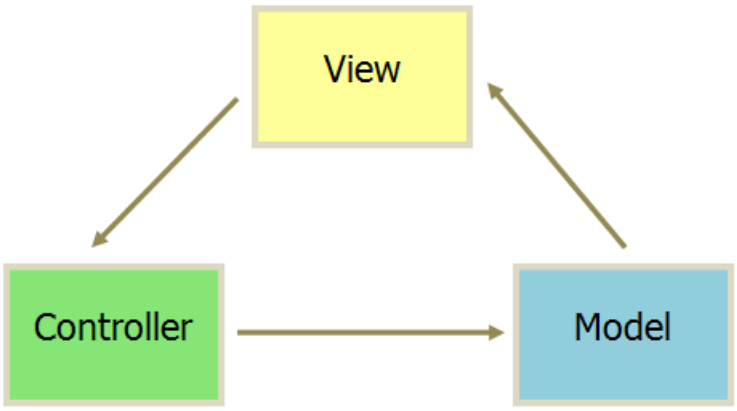


Рис.1.9 Модель MVVM

Вся інтерфейсна сторінка поділяється на View, Controller, Model та зміни уявлення. Відповідь передається в модель (джерело даних) через Controller, а дані у поданні змінюються джерелом даних.

Весь процес виглядає як хмара, бізнес-логіка розміщується в моделі, а логіка рендерингу сторінок - у виставі, але при практичному використанні виникає проблема: тобто інфраструктура MVC дозволяє View і Model безпосередньо взаємодіяти.

Іншими словами, оскільки обсяг бізнесу між View і Model продовжує зростати, існуватимуть залежності, з якими важко впоратися, подібно до павутини, повністю відхиляючись від «відкритого і закритого принципу», який має слідувати розробка.

Зіткнувшись із цією проблемою, з'явився фреймворк MVVM, у якому є дві основні відмінності від фреймворку MVC:

1. Поділ даних та подання

2. Керування переглядом даних, розробникам потрібно дбати лише про зміни даних, операції DOM інкапсульовані.

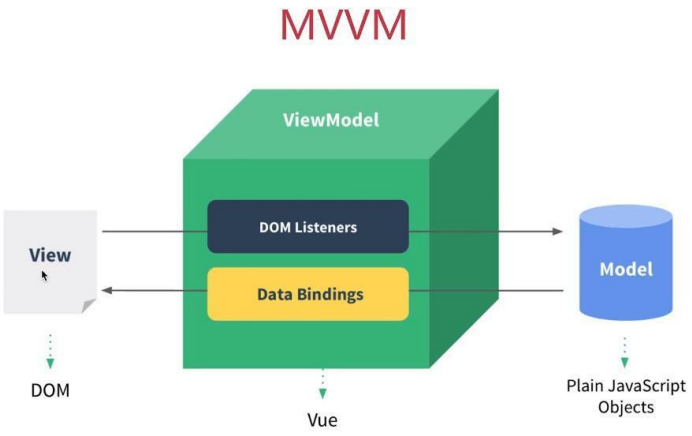


Рис.2.0 MVVM

Ви можете бачити, що MVVM посилається на View, Model, View-Model, View прив'язує події до Model через прослуховувачі DOM View-Model, а Model керує View через прив'язки даних. За даними, View-Model служить сполучним мостом.

Back-end

В області серверної розробки Model-View-Controller є одним із найбільш обговорюваних шаблонів проектування у світі розробки серверної частини. Архітектура MVC спочатку була включена в два основні середовища веб-розробки – Struts та Ruby on Rails.

Основні поняття:

Шаблон проектування в розробці програмного забезпечення – це метод вирішення проблеми, що часто виникає при розробці програмного забезпечення.

Проектування моделі вказує, який тип архітектури ви використовуєте для вирішення проблеми або розробки моделі.

Існує два типи моделей проектування: архітектура моделі 1, архітектура моделі 2 (MVC).

Що таке архітектура MVC у серверній частині?

Проекти моделей, засновані на архітектурі MVC в серверній частині, слідують шаблону проектування MVC і відокремлюють логіку програми від інтерфейсу користувача при розробці програмного забезпечення. Як випливає з назви, шаблон MVC має три шари:

- Модель – представляє бізнес-рівень програми.
- Перегляд – визначає подання програми.
- Контролер – управляє потоком програми.

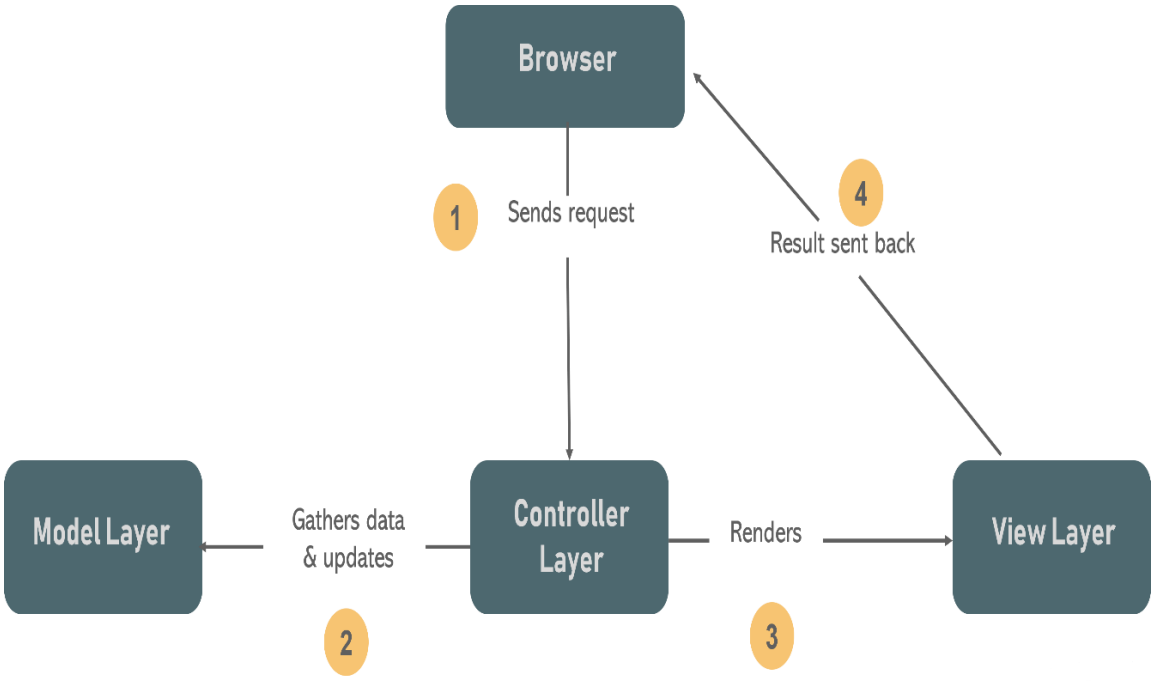


Рис.2.1 Модель MVC

У контексті програмування модель складається з простих класів, представлення відображає дані, а контролер складається із сервлетів. Цей поділ призводить до того, що запити користувача обробляються таким чином:

1. Браузер на клієнті надсилає запит сторінки контролеру, що є на сервері.
2. Контролер виконує дію на виклик моделі, тим самим виймаючи необхідні дані у відповідь на запит.
3. Потім контролер передає отримані дані у виставу.
4. Подання відображається та відправляється назад клієнту для відображення у браузері.

Поділ програмного додатку на ці три окремі компоненти є гарною ідеєю з низки причин.

Переваги архітектури

Архітектура MVC пропонує безліч переваг для програміста при розробці додатків, які включають:

- Декілька розробників можуть працювати з трьома шарами (Модель, Вид та Контролер) одночасно.
- Забезпечує покращену масштабованість, яка доповнює здатність програми зростати.
- Оскільки компоненти мають низьку залежність один від одного, їх легко підтримувати.
- Модель може бути повторно використана декількома уявленнями, що забезпечує можливість повторного використання коду.
- Прийняття MVC робить програму більш виразною і простою для розуміння.
- Розширення та тестування програми стає легким.

3.3 Керівництво користувача

Для того щоб реалізувати вибрану нами тему, ми вирішили створити роль адміністратора та користувача програми.

Отже, для початку що користувачам пропонується пройти авторизацію.

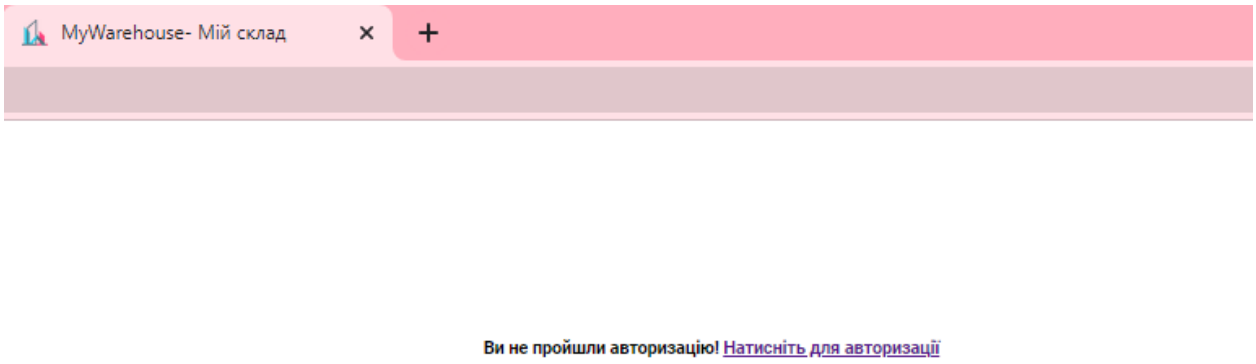


Рис. 2.2 Пропонується пройти авторизацію

Далі переходячи по посиланню «Натисніть для авторизації» з’являється вікно авторизації. На цій сторінці користувач може ввести свої особисті дані (електронна скринька та пароль), які він створював при реєстрації нового кабінету та увійти у свій кабінет. В разі введення некоректних даних з’явиться помилка про неправильно введені дані.

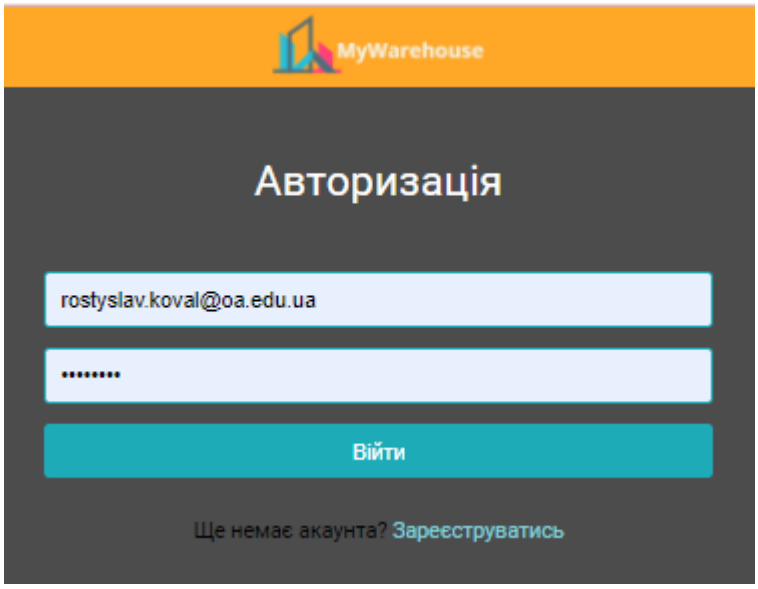


Рис. 2.3 Авторизація в системі

Якщо користувач ще немає акаунту, то йому пропонується перейти по посиланню та зареєструватись в нашій системі.

Для вдалої реєстрації прийдеться ввести свої дані, такі як: свою електронну скриньку, вигадати пароль, який будете застосовувати для входу в свій електронний кабінет, написати своє ім’я та номер телефону (щоб в разі потреби мати змогу зв’язатися з клієнтом та знати як до нього звертатися).

Реєстрація буде невдалою якщо:

1) користувач введе некоректні дані електронної скриньки, тоді введенні вже неправильні дані будуть підкреслені червоним курсивом та буде вимагати ввести коректний Email;

2) пароль не буде сходитись з попередньо створеним;

3) Ви не введете ім’я;

4) довжина номеру телефону повинна складатися з 10 цифр.

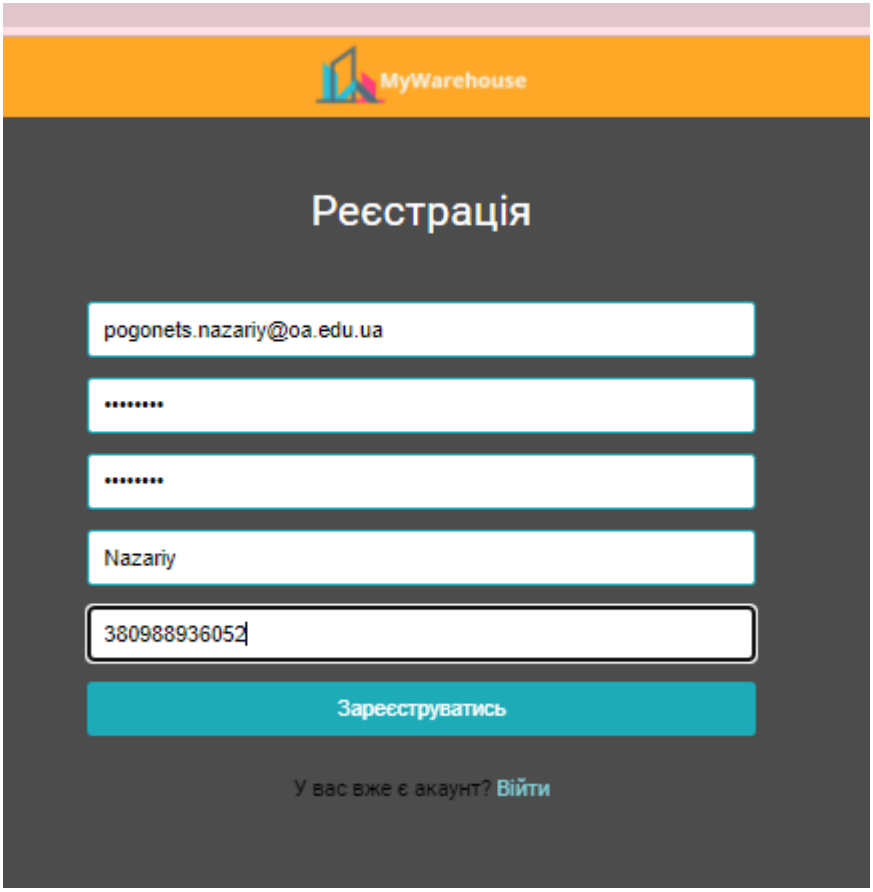
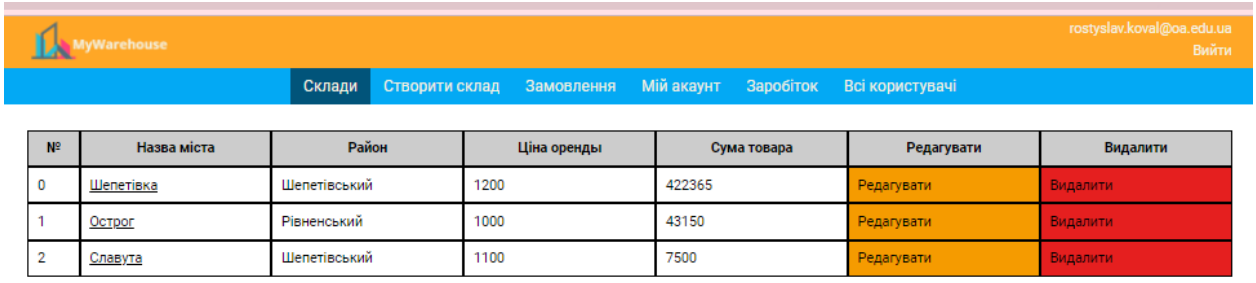


Рис.2.4 Реєстрація в системі.

Можливості адміністратора

Після того як користувач увійшов в свій акаунт (в даному випадку він має роль адміністратора), він може побачити інформацію про свої склади.



№	Назва міста	Район	Ціна оренды	Сума товара	Редагувати	Видалити
0	Шепетівка	Шепетівський	1200	422365	Редагувати	Видалити
1	Острог	Рівненський	1000	43150	Редагувати	Видалити
2	Славута	Шепетівський	1100	7500	Редагувати	Видалити

Рис.2.5 Сторінка «Склади».

Користувач може редагувати його або видалити (за умови, якщо він є адміністратором).

Також всі користувачі незалежно від ролі можуть перейти на сам склад і побачити товар, який він містить.

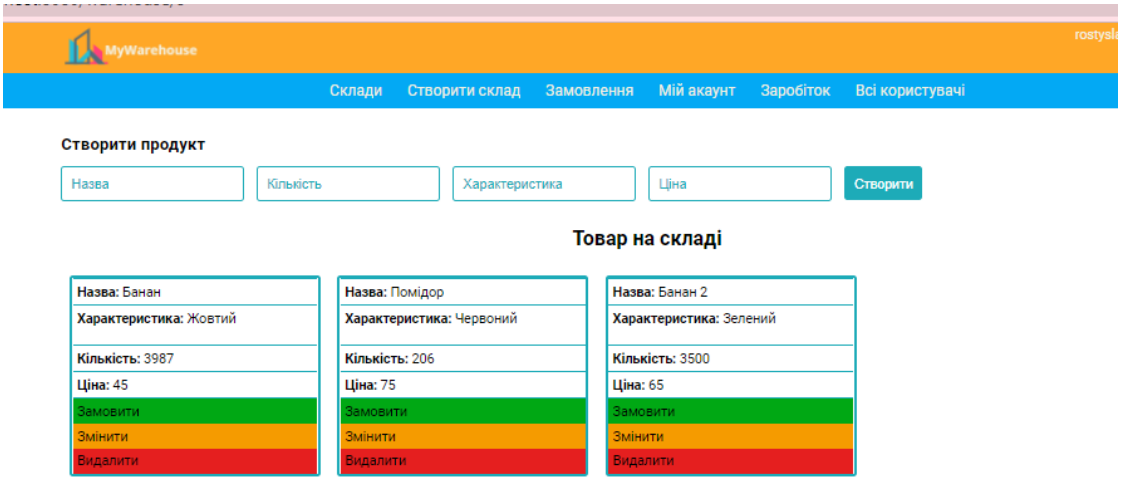


Рис.2.6 Товари на складі.

Так як наразі ми маємо роль адміністратора, можна створити новий продукт у цьому ж вікні, заповнивши відповідну форму, редагувати створений товар, видаляти його або зробити замовлення.

Адміністратор може створювати новий склад вказавши місто де буде знаходитись склад, район та ціну оренди.

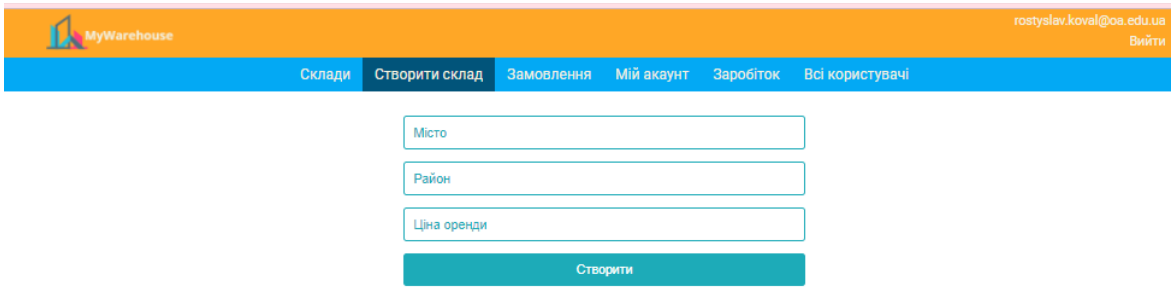
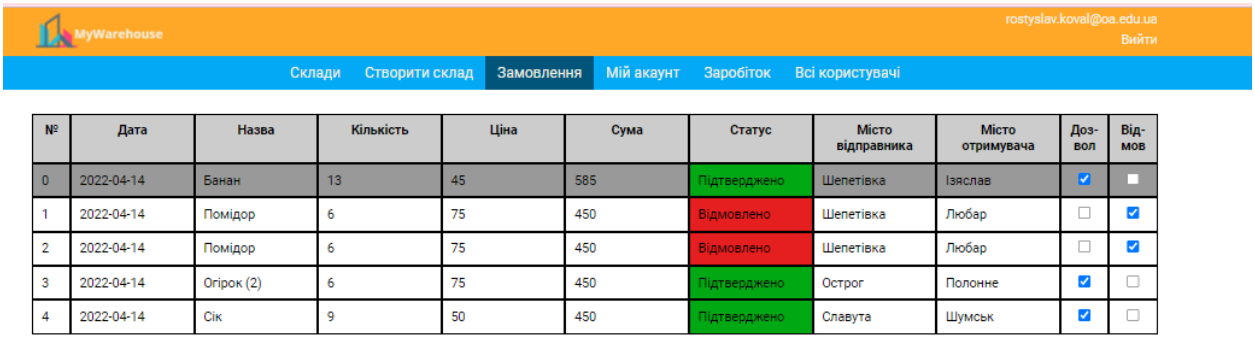


Рис.2.7 Створення складу

Створивши замовлення, воно з’явиться у всіх замовленнях.

Ми бачимо замовлення, яке щойно було створено. Тепер маючи роль адміністратора, ми можемо вирішити його статус, підтвердивши або відмовивши йому. Оскільки ми

маємо роль адміністратора, ми також можемо бачити заробіток з наших складів та бачити всіх користувачів, редагувати їх, а також створювати нові склади.



№	Дата	Назва	Кількість	Ціна	Сума	Статус	Місто відправника	Місто отримувача	Дозвол	Відмов
0	2022-04-14	Банан	13	45	585	Підтверджено	Шепетівка	Ізяслав	☑	☐
1	2022-04-14	Помідор	6	75	450	Відмовлено	Шепетівка	Любар	☐	☑
2	2022-04-14	Помідор	6	75	450	Відмовлено	Шепетівка	Любар	☐	☑
3	2022-04-14	Огірок (2)	6	75	450	Підтверджено	Острог	Полонне	☑	☐
4	2022-04-14	Сік	9	50	450	Підтверджено	Славута	Шумськ	☑	☐

Рис.2.8 Список замовлення.

Є також сторінка для редагування персональної інформації (в разі зміни деяких даних, щоб можна було мати актуальну інформацію про користувача).

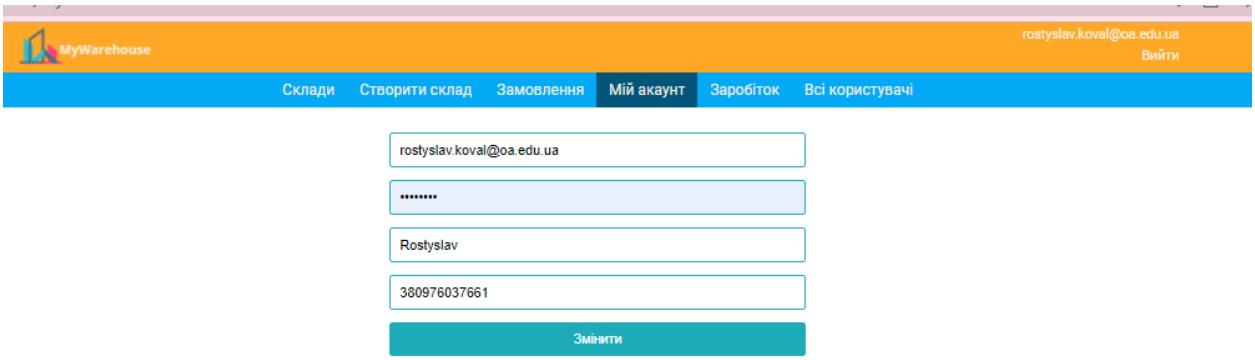
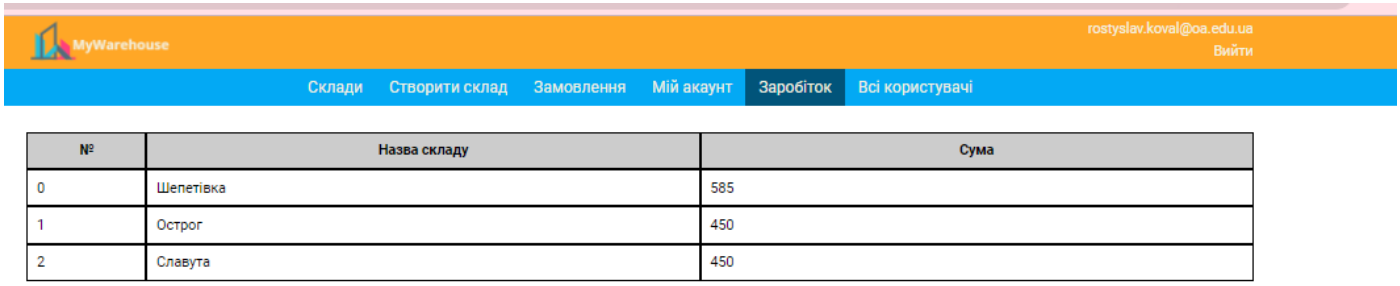


Рис.2.9 Персональна інформація сторінки користувача

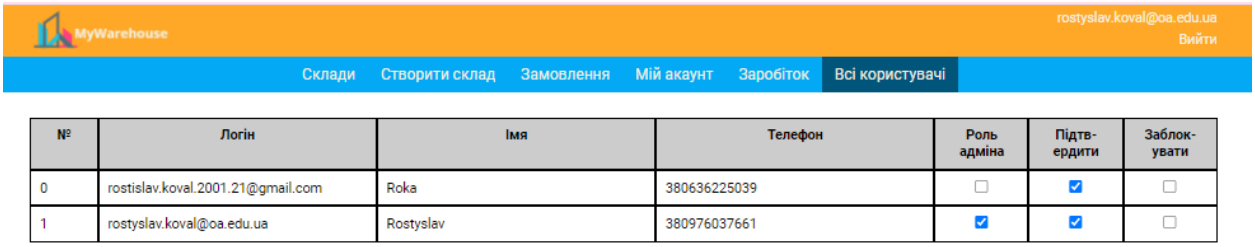
В розділі «Заробіток» можна дізнатись загальну суму доходу зі складу.



№	Назва складу	Сума
0	Шепетівка	585
1	Острог	450
2	Славута	450

Рис.3 Загальний заробіток з складів.

В розділі « Всі користувачі» адміністратор має змогу переглянути інформацію про всіх користувачів системи, а саме «логін, ім’я, номер телефону, та можна надати роль адміна, підтвердити акаунт користувача, або ж заблокувати його.



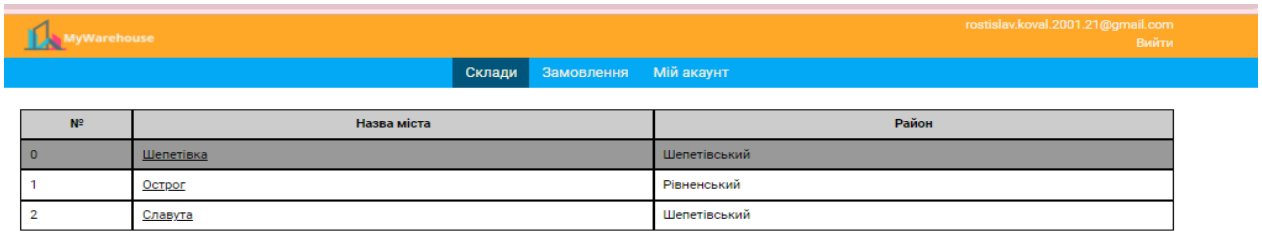
№	Логін	Імя	Телефон	Роль адміна	Підтв-ердити	Заблок-увати
0	rostislav.koval.2001.21@gmail.com	Roka	380636225039	☐	☑	☐
1	rostyslav.koval@oa.edu.ua	Rostyslav	380976037661	☑	☑	☐

Рис.3.1 Розділ «Всі користувачі»

Можливості користувача.

Можливості користувача є скромнішими як адміністратора, однак всі необхідні можливості для покупця присутні.

Користувач може переглядати усі наявні склади та обирати йому потрібний.



№	Назва міста	Район
0	Шепетівка	Шепетівський
1	Острог	Рівненський
2	Славута	Шепетівський

Рис.3.2 Наявні склади.

Обравши потрібний користувачу склад, може переглядати продукцію на складі, бачити назву, характеристику, кількість, ціну та має можливість замовити продукцію (після замовлення надходить запит адміністратору про нове замовлення).

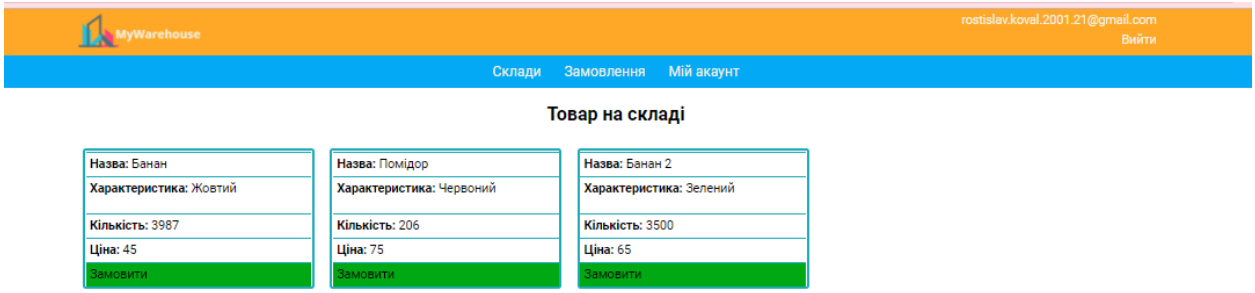


Рис.3.3 Товари на складі.

При оформленні замовлення слід вказати кількість та місто отримувача. При замовленні необхідної кількості продукції виводиться загальна сума замовлення.

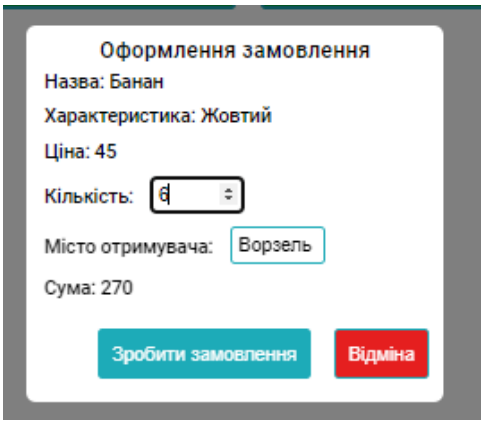
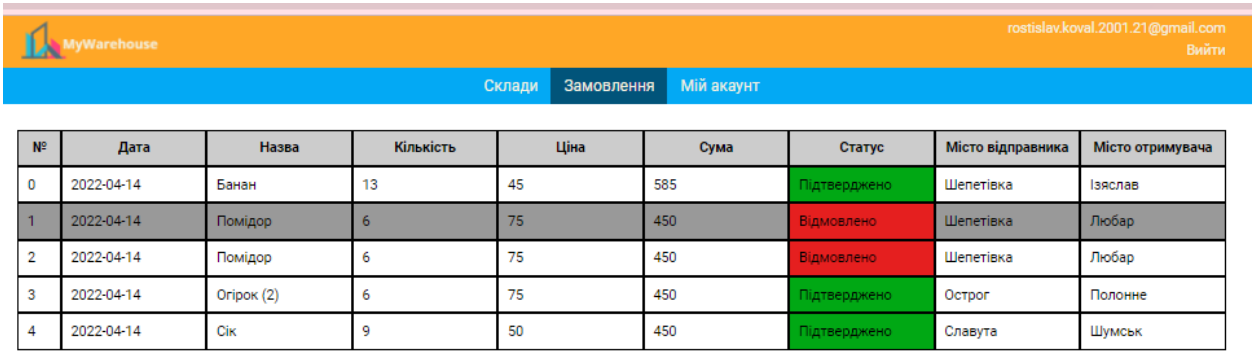


Рис. 3.4 Оформлення замовлення.

Також можна переглядати усі замовлення які зробив користувач. А саме: дату коли було замовлено, назву продукції, кількість, ціну, суму, статус, місто відправника та місто одержувача.



№	Дата	Назва	Кількість	Ціна	Сума	Статус	Місто відправника	Місто отримувача
0	2022-04-14	Банан	13	45	585	Підтверджено	Шепетівка	Ізяслав
1	2022-04-14	Помідор	6	75	450	Відмовлено	Шепетівка	Любар
2	2022-04-14	Помідор	6	75	450	Відмовлено	Шепетівка	Любар
3	2022-04-14	Огірок (2)	6	75	450	Підтверджено	Острог	Полонне
4	2022-04-14	Сік	9	50	450	Підтверджено	Славута	Шумськ

Рис.3.5 Замовлення здійснені з акаунту.

Є також сторінка для редагування персональної інформації (в разі зміни деяких даних, щоб можна було мати актуальну інформацію про користувача).

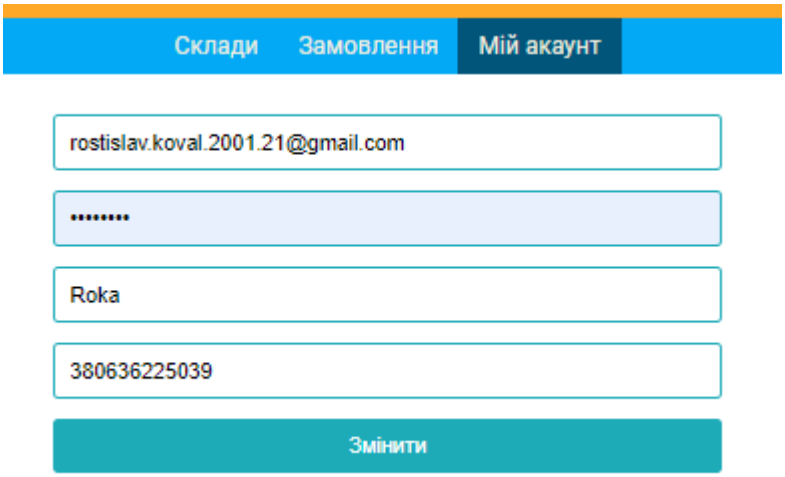


Рис.3.6 Персональна інформація користувача, її редагування.

Зверху зі правої сторони є інформація з якого акаунту був виконаний вхід, як в користувача так і в адміністратора є можливість в будь-який час вийти зі свого акаунту, натиснувши на кнопку «вийти».

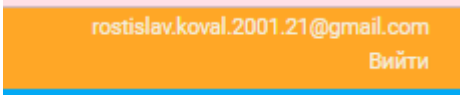


Рис.3.7 Акаунт з якого був виконаний вхід та вихід.

Тестування

Після завершення розробки ми приступили до тестування, використовували як автоматизоване, так і мануальне.

Відмінності між ручним та автоматичним тестуванням

Нижче наведено пояснення ручного та автоматичного тестування:

- Автоматичне тестування передбачає використання інструментів тестування Ручне тестування потребує втручання людей для тестування. Тоді як ручне тестування вимагає кваліфікованої робочої сили, тривалого часу та витрат.
- Автоматичне тестування економить час, витрати та робочу силу. Коли записується, автоматизований набір тестів простіший в експлуатації.
  Деякі види тестів, як тестування спеціальних дій та тестування мавп, більше підходять для ручного виконання, і будь-який запит можна перевірити вручну. Автоматизовані тести пропонуються лише для стабільних систем і в основному використовуються для тесту регресії.

Інтеграційне тестування

Виходячи із відмінностей між модульним тестування і системним тестуванням, інтеграційне тестування є перехідним етапом між представленням програми у вигляді окремих модулів у вигляд повністю функціональної системи.

Інтеграційне тестування – вид тестування, при якому на відповідність вимог перевіряється інтеграція модулів, їх взаємодія між собою, а також інтеграція підсистем в одну загальну систему. Для інтеграційного тестування використовуються компоненти, вже перевірені за допомогою модульного тестування, які групуються у множини. Дані множини перевіряються відповідно до плану тестування, складеним для них, а об’єднуються вони через свої інтерфейси.

Так як модулі поєднуються між собою за допомогою передбачених реалізацією інтерфейсів і в процесі тестування у нас немає потреби розглядати внутрішню структуру компонентів, можна стверджувати, що інтеграційне тестування виконується методом «чорного ящику».

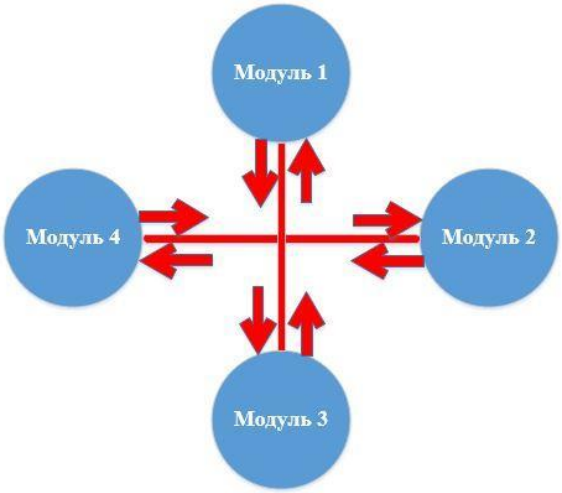


Рис.3.8 Зображення об’єкту тестування

Об’єкт тестування виділений червоним кольором.

Існує декілька підходів до інтеграційного тестування:

Знизу вгору. Спочатку збираються і тестуються модулі найнижчих рівнів, а потім по зростанню до вершини ієрархії. Даний підхід вимагає готовності усіх зібраних модулів на всіх рівнях системи.

Зверху вниз. Даний підхід передбачає рух з модулів високого рівня вниз. При цьому використовуються заглушки для тих модулів, які знаходяться нижче за рівнем, але включення яких до тесту ще не відбулося.

Великий вибух. Всі модулі всіх рівнів збираються разом, а потім тестується. Даний метод економить час, але вимагає ретельного опрацювання тест кейсів.

При автоматизації тестування використовується Система безперервної інтеграції. Принцип її дії в наступному:

1) Система безперервної інтеграції проводить моніторинг системи контролю версій.

2) При зміні джерела коду в репозиторії проводиться оновлення локального сховища.

3) Виконуються необхідні перевірки і модульні тести.

4) Вихідні коди компілюються в готові виконувані модулі.

5) Виконуються тести інтеграційного рівня.

6) Генерується звіт про тестування.

Це дозволяє тестувати систему відразу після внесення змін, що істотно скорочує час виявлення та виправлення помилок.

Тестування вручну - це форма тестування програмного забезпечення, у якій тести виконуються вручну без використання приладів автоматизації. Найпримітивнішим з усіх видів тесту є ручне тестування та допомагає користувачам виявляти помилки у програмній системі. Будь-які свіжі програми потрібно протестувати вручну, перш ніж автоматизувати тестування. Це тестування потребує більших зусиль, але воно необхідне для перевірки доцільності автоматизації. Тестувальник підготує тестовий документ, що описує комплексний та систематичний підхід до тестування програмного забезпечення. Тестові примірники охоплюють майже 100% запланованої реалізації програмного забезпечення. Це трудомісткий тест, оскільки ручні тести включають повні тестові примірники. Існують недоліки у розрізненні між реальними та бажаними результатами. Потім розробник програмного забезпечення виправляє недоліки. З метою забезпечення виправлення несправностей тестер оцінює дефекти. Цель цього тестування - переконатися, що додаток не містить дефектів і помилок добре працює, щоб надати клієнтам відмінну якість роботи.

Висновок до розділу 3

У цьому розділі ми ознайомились з програмною та технічною частиною нашого проекту, описали основну структуру Front-end та Back-end. Також у цій частими ми коротко розповіли про функціонал нашого проекту, та методи тестування з яким ми знайомились, для того щоб протестувати наш проект.

Загальний висновок

Отже, ми розробили програму кваліфікаційної роботи темою якої була: «Розробка додатку для автоматизації управління складськими запасами». Поставлені перед собою вимоги ми виконали, а саме:

Першою вимогою є авторизація адміністратора в систему з подальшим контролем користувачів, або ж авторизація користувача.

Адміністратор має перед собою такі права, як:

- створювати нових користувачів і надсилати їм запрошення в систему

- самостійно користувач отримати доступ до системи не може

- адміністратор може створювати та редагувати всю пов’язану з складами та замовленнями інформацію

- адміністратор додає та видаляє користувачів системи, підтверджує або ж відхиляє замовлення

В свою чергу клієнт має більш обмежені права:

- виконує реєстрацію в додатку

- отримує інформацію про склади та продукцію

- має можливість замовлення певного продукту з вибраного складу

Ми гадаємо, що дана програма є досить актуальною у сьогодення, так як кількість онлайн торговців зростає з кожним днем. Причиною цього всього може слугувати дана ситуація в країні, а саме воєнний стан та карантинні обмеження. Людям потрібно контролювати продукцію на складах та замовляти потрібні їм речі не виходячи з дому, щоб залишатися у безпеці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Положення про проведення практики здобувачів вищої освіти НаУОА.
2. Положення про навчально-методичне забезпечення в НаУОА.
3. Шаховська Н.Б., Голощук Р.О. Алгоритми і структури даних . Посібник. Львів. 2008. 245 с.
4. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.
5. Альфред Ахо. Структуры данных и алгоритмы / [Альфред Ахо, Джон Э. Хопкрофт, Джеффри Д.Ульман]. М.: Вильямс, 2007. 400 с.
6. Вирт Н. Алгоритмы и структуры данных. М., 2012. 272 с.

Чисельні методи : навчальний посібник / В. М. Задачин, І. Г. Конюшенко. Х. : Вид. ХНЕУ ім. С. Кузнеця, 2014. 180 с.

7. Будай А. Дизайн-паттерны – простіше простого. Львів, 2012. 90 с.
8. Швець. О. Занурення в патерни проектування. Refactoring.guru. 397 с.
9. Glenford J. Myers, Corey Sandler, Tom Badgett. The Art of Software Testing. 2011. 256 p.
10. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.
11. Вакалюк Т.А. Технології тестування програм. Навчально-методичний посібник для студентів напряму 6.040302 Інформатика. Житомир: Вид-во ЖДУ, 2013. 96 с.
12. Бер Бібо, Єгуда Кац. jQuery. Детальний посібник з просунутому JavaScript. - М.: Символ-Плюс, 2009. - 168 с.
13. Вайк А. JavaScript. Енциклопедія користувача. - К.: DiaSoft 2003. - 204 с.
14. Вільямсон Х. Універсальний Dynamic HTML. - СПб.: Пітер, 2001. – 185 с.
15. Кожем'якін А.А. HTML та CSS у прикладах. створіння Web-сторінок. - Мінськ: Альтекс-А, 2004. - 255с.

16. Колісніченко Д. Н. Joomla 1.5. Керівництво користувача. - М.: Діалектика, 2009. - 265 с.
17. Матросов А.В. HTML 4.0 у оригіналу. - СПб.: BHV, 2004. 7. Мейєр Е. А. CSS. Каскадні таблиці стилів. Детальний посібник. - М.: Символ-Плюс, 2008. - 235 с.
18. Томсон Л. Веллінг Л. Розробка Web-додатків на PHP та MySQL.- М.: Вільямс, 2009. - 330 с.
19. Хабібуллін І. Самовчитель XML. - СПб.: BHV, 2003.
20. Акоста Н. Внутрішній світ World Wide Web. - К.: DiaSoft, 2005. – 180 с.
21. Айзекс С. Dynamic HTML. Секрети створення інтерактивних Web - сторінок. - СПб.: BHV, 2001. - 255 с.
22. Боуен Річ. Apache.Настільна книга адміністратора. - К.: DiaSoft, 2002. – 350 с.
23. Зельдман Дж. Web-дизайн за стандартами. - М: НТ Прес, 2005. – 260 с.
24. Крейн Д., Паскарелло Е., Джеймс Д. Ajax у дії. - М.: Вільямс, 2006. – 195 с.
25. Матвієнко, О.В. Internet-технології: проектування Webсторінки: Навчальний посібник для студентів вузів / О. В. Матвієнко, І. Л Бородкіна. - 2-е вид., перероб. та дод. . – К.: ЦНЛ, 2004. – 154 с.