

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ ЗАКЛАД  
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут математики та інформаційних технологій  
Кафедра математики систем та інформатики

**Чень Ліянь**

**АНАЛІЗ ТА РОЗРОБКА ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ОНЛАЙН  
ТОРГІВЛІ**

Магістерська робота  
за спеціальністю 122 Комп’ютерні науки

Особистий підпис – \_\_\_\_\_

Науковий керівник – \_\_\_\_\_ д.т.н., професор Юрій КОЗУБ

В.о.зав. кафедри – \_\_\_\_\_ д.т.н., професор Юрій КОЗУБ

Полтава - 2025

## **АНОТАЦІЯ**

**Чень Ліянь. Аналіз та розробка веборієнтованої системи онлайн торгівлі.** Кваліфікаційна робота магістра за спеціальністю 122 «Комп'ютерні науки» Державний заклад «Луганський національний університет імені Тараса Шевченка», м. Полтава, 2025.

Актуальність роботи. На сьогоднішній день інтернет-торгівля є невід'ємною частиною суспільства, щодня в світі відбуваються тисячі різноманітних угод купівлі та продажу товарів або послуг через мережу інтернет. Для взаємодії з інтернет-магазинами користувачі, а також адміністраторам цих інтернет-магазинів використовують веббраузери, які в свою чергу використовують веб-додатки.

Мета і завдання роботи. Провести аналіз предметної області, розглянути існуючі варіанти рішень та провести вибір архітектури системи для розробки системи контролю та обліку товарів на складі інтернет магазину.

Загальна характеристика роботи. В роботі проведено порівняння існуючих систем інтернет-магазинів та розглянуто технології клієнт-серверних додатків. Проведено проектування системи, де розглянуто ряд вимог до системи, створено діаграми використання та послідовності взаємодії. Розроблена специфікація вимог до програмного забезпечення (SRS). Проведена реалізація програмного забезпечення клієнт-серверної архітектури на основі стеку angular framework.

Магістерська робота складається зі вступу, чотирьох розділів, висновків, списку використаної літератури, додатків, 2 таблиць та 14 рисунків.

**Ключові слова:** вебдодаток, Інтернет, Angular

## **ABSTRACT**

**Chen Liyan. Analysis and development of a web-based online trading system.**

Master's degree thesis in specialty 122 "Computer Science" State Institution "Luhansk Taras Shevchenko National University", Poltava, 2025.

**Relevance of the work.** Today, online commerce is an integral part of society, thousands of various transactions for the purchase and sale of goods or services take place every day in the world via the Internet. To interact with online stores, users, as well as administrators of these online stores, use web browsers, which in turn use web applications.

**Purpose and objectives of the work.** To analyze the subject area, consider existing solution options and select a system architecture for developing a system for controlling and accounting for goods in the warehouse of an online store.

**General characteristics of the work.** The work compares existing online store systems and considers client-server application technologies. The system is designed, where a number of system requirements are considered, usage diagrams and interaction sequences are created. A software requirements specification (SRS) is developed. The software of the client-server architecture based on the angular framework stack is implemented.

The master's thesis consists of an introduction, four sections, conclusions, a list of references, appendices, 2 tables and 14 figures.

**Keywords:** web application, Internet, Angular

## ЗМІСТ

ВСТУП.....	5
1. АНАЛІЗ ВИМОГ ДО РЕСУРСУ .....	6
1.1. Аналіз предметної області.....	6
1.2. Огляд існуючих рішень .....	13
1.2.1. AliExpress .....	13
1.2.2. Taobao.com .....	15
1.2.3. Rozetka.ua .....	17
1.2.4. OLX.....	18
1.3. Вибір архітектури системи.....	20
РОЗДІЛ 2. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ.....	27
2.1. TypeScript .....	27
2.2. Angular .....	30
РОЗДІЛ 3. ПРОЕКТУВАННЯ СИСТЕМИ.....	36
3.1. Визначення та аналіз вимог до системи .....	36
3.3. Моделювання системи.....	42
3.4. Розробка графічного інтерфейсу користувача .....	51
РОЗДІЛ 4. РЕАЛІЗАЦІЯ СИСТЕМИ.....	55
4.1. Схема роботи розробленої системи.....	55
ВИСНОВКИ.....	57
ПЕРЕЛІК ПОСИЛАНЬ .....	58
ДОДАТКИ.....	61

## ВСТУП

На сьогоднішній день тема покупок в інтернеті стає як ніколи актуальною. Кожного дня люди придбають та продають в інтернеті мільйони різноманітних товарів, асортимент яких необмежений і стосується будь-якої сфери людського життя.

Причини популярності інтернет-магазинів полягають в декількох моментах:

1. Небажання або відсутність можливості покинути дім
2. Неможливість дістати той чи інший товар в своєму місті
3. Можливість перевірити товар до оплати
4. Відгуки інших покупців щодо того чи іншого товару
5. Бажання спростити все, що можна спростити

Таким чином, через цей ряд причин, люди все частіше і частіше надають перевагу покупкам в інтернеті, аніж походам по реальним магазинам.

В зв'язку з підняттям попиту на покупку товарів в інтернеті, росте і пропозиція таких. Саме таким чином і з'явилося таке явище, як інтернет магазин.

Темою магістерської роботи є розробка системи автоматизованого обліку товарів інтернет-магазину, націленого на продаж різноманітних товарів, з метою дослідження інструментальних методів розробки подібного роду ресурсів.

## 1. АНАЛІЗ ВИМОГ ДО РЕСУРСУ

### 1.1. Аналіз предметної області

Вебдодаток — це програма, в якій клієнтом виступає браузер, а сервером — веб-сервер. Браузер належить до категорії тонких клієнтів, здатний відображати веб-сторінки та зазвичай є частиною операційної системи. Його оновлення і підтримка здійснюються постачальником ОС. Основна логіка веб-додатку знаходиться на сервері, тоді як браузер виконує роль інтерфейсу, відображаючи інформацію, отриману з сервера через мережу.

Однією з ключових переваг цього підходу є незалежність клієнтів від операційної системи, що робить вебдодатки універсальними та міжплатформеними. Завдяки такій універсальності та простоті розробки вебдодатки здобули популярність наприкінці 1990-х і на початку 2000-х років.

Підтримка стандартних функцій браузера дозволяє додаткам працювати незалежно від платформи клієнта. Замість створення окремих версій для Windows, Mac OS X, GNU/Linux чи інших систем, веб-додаток розробляється один раз і запускається на будь-якій платформі. Проте, різні реалізації HTML, CSS, DOM та інших стандартів у браузерах можуть створювати труднощі для розробників і технічної підтримки. Крім того, налаштування браузера користувачем (зміна шрифтів, кольорів, відключення сценаріїв тощо) може впливати на коректну роботу додатку.

Альтернативний підхід включає використання Adobe Flash або Java-апплетів для створення інтерфейсу. Більшість браузерів підтримують ці технології через плагіни, що спрощує їх використання. Вони надають програмістам більше контролю над інтерфейсом і допомагають уникати проблем із сумісністю браузерів. Проте різні реалізації Flash чи Java можуть створювати нові складнощі.

Зважаючи на схожість таких технологій із традиційними клієнт-серверними додатками ("товстими клієнтами"), виникають дискусії щодо їхньої належності до категорії веб-додатків. У таких випадках часто використовується термін "багатий інтернет-додаток" (Rich Internet Application).

Вебдодаток отримує запит від клієнта, виконує необхідні обчислення, формує вебсторінку та передає її клієнту через мережу за допомогою протоколу HTTP. Такий додаток також може виступати клієнтом для інших служб, наприклад, баз даних або інших вебдодатків, які розташовані на окремих серверах.

Яскравим прикладом вебдодатку є система управління вмістом статей у Вікіпедії. Величезна кількість користувачів може брати участь у створенні цієї онлайн-енциклопедії, використовуючи браузері своїх операційних систем (наприклад, Windows, GNU/Linux або інші), без необхідності завантажувати додаткові програми для роботи з базою даних статей.

Сьогодні популярність набуває новий підхід до розробки вебдодатків, відомий як Аґах. Завдяки Аґах вебсторінки оновлюються динамічно, без повного перезавантаження, завантажуючи лише необхідні дані з сервера. Це робить додатки більш інтерактивними, швидкими та зручними у використанні.

Для створення вебдодатків використовуються різноманітні технології і мови програмування, наприклад: PHP, ASP, ASP.NET, JSP, Java, CGI, Perl, Python, Ruby on Rails та інші. Ряд з них (PHP, Perl, Python) мають відкритий код, розповсюджуються вільно і можуть використовуватися практично на будь-яких вебсерверах, інші (ASP, ASP.NET, Java) — прив'язані до конкретних вебсерверів. На сьогодні існує безліч всіляких систем створення вебдодатків, серед яких можна виділити текстові редактори (Emacs, gedit, Notepad, TextEdit, UltraEdit, vi і т.п.); текстові редактори HTML-коду (т.з. «text-based HTML editors»: Alleycode HTML Editor, Aptana, Arachnophilia, BBEdit, Bluefish, Crimson Editor, CoffeeCup HTML Editor, EditPlus, Evrsoft 1st Page, HateML Pro, HotHTML, HTML-Kit, HTMLPad,

Macromedia HomeSite, Notepad++, NoteTab, PSPad, Quanta Plus, SCREEM, Siteaid, skEdit, Taco HTML Edit, TextMate, TopStyle, WebTide і т.п.); редактори WYSIWYG («What You See Is What You Get» або «що ви бачите, то ви і получите»)-редактори; досить могутні і зручні засоби створення і обробки вебдодатків: Adobe Contribute "Dreamweaver Lite", Adobe Dreamweaver – раніше називався Macromedia Dreamweaver, Adobe GoLive, Amaya, Blockstar, Bluevoda, HotDog, iWeb, Media Lab SiteGrinder, Microsoft Expression Web, Microsoft SharePoint Designer, Microsoft Visual Studio / ASP.NET Web Matrix, Microsoft Visual Web Developer, NetObjects Fusion, Nvu, Quanta Plus, Sandvox, SeaMonkey Composer, Softpress Freeway, RapidWeaver, WorldWideWeb і т.п.).

Додатки зазвичай діляться на логічні частини, звані "шарами", при цьому кожному шару призначається своя роль. Локальні додатки можуть складатися тільки з одного шару, який розміщується на комп'ютері клієнта, а вебдодатки за своєю природою слідуєть N -слойному підходу. Хоча можливі різні варіанти, найбільш поширеними є функції, які залежать три шари: шар уявлення ; шар бізнес-логіки ; шар доступу до даних (сховище). Кожен шар включає набір компонент (наборів класів), які виконують спеціальні функції.

Вебдодатки мають переваги у порівнянні з локальними програмами:

Це легкий доступ: будь-хто з доступом до Інтернету може використовувати вебдодаток без необхідності додаткових дій; простота розгортання: вебдодатки не потребують встановлення на комп'ютерах користувачів. Достатньо надати URL-адресу програми. У разі оновлення всі користувачі автоматично працюють із новою версією.

Крім того, вебдодаток має широку аудиторію досвідчених користувачів: завдяки популярності браузерів багато людей вже знайомі з роботою вебдодатків. Надійність і розвиток технологій: сучасні мережеві з'єднання та вебтехнології забезпечують стабільність і ефективність роботи додатків.



Поряд з перевагами необхідно зазначити недоліки вебдодатків:

У вебдодатках немає вбудованої підтримки постійного стану сесії, а також виникає затримка під час завантаження кожної сторінки. При кожному оновленні сторінки потрібно встановлювати HTTP-з'єднання, обробляти запит на сервері, отримувати відповідь і перезавантажувати сторінку в браузері. Це може призводити до переривань у роботі користувача. Набір елементів інтерфейсу обмежений можливостями мов програмування: HTML надає лише базові елементи управління для створення форм (текстові поля, радіокнопки, прапорці, випадаючі списки та кнопки), що обмежує можливості дизайну та функціональності додатків.

Інтернет-покупки - це форма електронної комерції, яка дозволяє споживачам безпосередньо купувати товари чи послуги у продавця через Інтернет за допомогою веббраузера. Споживачі знаходять товар, що цікавить, безпосередньо відвідуючи вебсайт роздрібної торгівлі або здійснюючи пошук серед альтернативних постачальників за допомогою торгової пошукової системи, яка відображає доступність та ціни одного товару у різних електронних торгових мереж. Станом на 2024 рік, клієнти можуть здійснювати покупки в Інтернеті за допомогою різних комп'ютерів та пристроїв, включаючи настільні комп'ютери, ноутбуки, планшетні комп'ютери, смартфони та розумні динаміки.

Інтернет-магазин викликає фізичну аналогію купівлі товарів чи послуг у звичайного роздрібного торгового центру чи торгового центру; цей процес називається інтернет-магазинами бізнес-споживач (B2C). Коли вебмагазин створений для того, щоб він міг купувати інший бізнес, цей процес називається інтернет-магазинами бізнес-бізнес (B2B). Типовий інтернет-магазин дозволяє замовнику переглядати асортимент товарів і послуг фірми, переглядати фотографії або зображення товарів, а також інформацію про технічні характеристики, опис продавця та ціни.

Інтернет-магазини зазвичай дозволяють покупцям використовувати функції "пошуку" для пошуку конкретних моделей, марок або предметів. Клієнти в Інтернеті повинні мати доступ до Інтернету та дійсний спосіб оплати, щоб завершити транзакцію, наприклад, кредитну карту, дебетову карту з підтримкою Interac або послугу, наприклад PayPal. Для фізичних продуктів (наприклад, книжок у м'якій обкладинці чи одягу) електронний пристрій відправляє продукцію замовнику; для цифрових продуктів, таких як цифрові аудіофайли пісень або програмного забезпечення, електронна пошта зазвичай надсилає файл замовнику через Інтернет. Найбільші з цих інтернет-корпорацій роздрібної торгівлі - Alibaba, Amazon.com та eBay.

Альтернативні назви діяльності - "електронний талінг", скорочена форма "електронний роздріб" або "електронний шопінг", скорочена форма "електронних покупок". Інтернет-магазин також може називатися електронним вебмагазином, електронним магазином, інтернет-магазином, вебмагазином, вітриною Інтернет-магазину та віртуальним магазином. Мобільна комерція (або m-commerce) описує покупку від вебсайту, оптимізованого для мобільних пристроїв, розробленого в Інтернеті, або програмного забезпечення ("додаток"). Ці вебсайти чи програми створені для того, щоб клієнти могли переглядати товари та послуги компанії на планшетних комп'ютерах та смартфонах.

Однією з найбільш ранніх форм торгівлі, що проводилася в Інтернеті, була обробка онлайн-транзакцій IBM (OLTP), розроблена в 1960-х роках, і вона дозволила обробляти фінансові операції в режимі реального часу. Комп'ютеризована система бронювання квитків, розроблена для американських авіакомпаній під назвою Напіваавтоматичне середовище для досліджень бізнесу (SABER), була однією з її прикладних програм. Тут комп'ютерні термінали, розташовані в різних туристичних агентствах, були пов'язані з великим комп'ютером IBM з основним кадром, який одночасно обробляв транзакції і

координував їх, щоб усі туристичні агенти мали доступ до однієї і тієї ж інформації одночасно.

Поява інтернет-магазинів, як ми знаємо сьогодні, розвивалася з появою Інтернету. Спочатку ця платформа функціонувала лише як рекламний інструмент для компаній, надаючи інформацію про свою продукцію. Це швидко перейшло від цієї простої утиліти до фактичної транзакції по магазинах в Інтернеті завдяки розробці інтерактивних вебсторінок і безпечних передач. Зокрема, зростання Інтернету як захищеного торгового каналу розвивалося з 1994 року, з першими продажами альбому Стінга «Ten Summoner's Tales». Вино, шоколадки та квіти незабаром пішли в трійку піонерських роздрібних категорій, що сприяло зростанню інтернет-магазинів. Дослідники встановили, що наявність продуктів, відповідних для електронної комерції, є ключовим показником успіху в Інтернеті. Багато з цих продуктів справились добре, оскільки це загальна продукція, яку покупцям не потрібно було чіпати та відчувати, щоб купувати. Але також важливо, що в перші дні покупців в Інтернеті було мало, і вони були з вузького сегмента: заможні, чоловіки, старше 30 років. Інтернет-покупки почали розвиватися з цих перших днів, і у Великобританії припадає значна частка відсотків (залежно від категорії товару, оскільки відсотки можуть змінюватися).

Маркетинг навколо цифрового середовища, поведінка покупців на покупців може не впливати та контролюватися торговою маркою та фірмою, коли вони приймають рішення про покупку, яке може стосуватися взаємодії з пошуковою системою, рекомендаціями, оглядами в Інтернеті та іншою інформацією. Завдяки швидкому відокремленню середовища цифрових пристроїв люди швидше використовують свої мобільні телефони, комп'ютери, планшети та інші цифрові пристрої для збору інформації. Іншими словами, цифрове середовище зростає на розум споживача та поведінку покупців. В умовах онлайн-торгівлі інтерактивне рішення може впливати на прийняття рішень щодо надання допомоги клієнтам.

Кожен клієнт стає все більш інтерактивним, і хоча онлайн-огляди клієнтів можуть впливати на поведінку інших потенційних покупців.

Згодом ризик та довіра також будуть двома важливими факторами, що впливають на поведінку людей у цифрових середовищах. Клієнт розглядає можливість перемикання між електронними каналами, оскільки на них впливає, головним чином, порівняння з офлайн-магазинами, що передбачає зростання ризику для безпеки, фінансових та виробничих ризиків. На прийняття рішення про купівлю можуть впливати три фактори, по-перше, люди не можуть перевірити, чи задовольняє товар їхні потреби та потреби до того, як вони отримають його. По-друге, клієнт може потурбуватися про послуги післяпродажного обслуговування. Нарешті, клієнт може побоюватися, що не може повністю зрозуміти мову, що використовується в електронних продажах. Виходячи з цих факторів, сприйняття клієнта ризиком може суттєво впливати на поведінку покупців в Інтернеті .

Інтернет-роздрібні торговці приділяють багато уваги аспекту довіри клієнтів, довіра - це ще один спосіб керування поведінкою клієнта в цифровому середовищі, який може залежати від ставлення та очікування клієнта. Дійсно, дизайн або ідеї компанії не можуть відповідати очікуванням замовника. Намір покупця купувати на основі раціональних очікувань та додатково впливає на емоційну довіру. Більше того, ці очікування можна встановити також щодо інформації про товар та перегляду інших.

Споживачі знаходять товар, що цікавить, відвідуючи вебсайт продавця безпосередньо або здійснюючи пошук серед альтернативних постачальників за допомогою торгової пошукової системи. Після того, як певний товар знайдений на вебсайті продавця, більшість інтернет-магазинів використовують програмне забезпечення для кошика, щоб споживач міг накопичувати кілька предметів та коригувати кількість, наприклад, наповнюючи фізичну кошик або кошик у звичайному магазині. Наступний процес "оформлення замовлення" (продовження

аналогій фізичного магазину), в якому збирається інформація про оплату та доставку, якщо це необхідно. Деякі магазини дозволяють споживачам підписатися на постійний онлайн-акаунт, так що частину або всю цю інформацію потрібно вводити лише один раз. Після закінчення транзакції споживач часто отримує підтвердження електронною поштою. Менш складні магазини можуть покладатися на споживачів, щоб зателефонувати або надіслати електронною поштою свої замовлення (хоча повні номери кредитних карток, дата закінчення терміну дії та Кодекс безпеки картки або банківський рахунок та номер маршруту не повинні прийматися електронною поштою з причин безпеки).

## 1.2 Огляд існуючих рішень

### 1.2.1. AliExpress

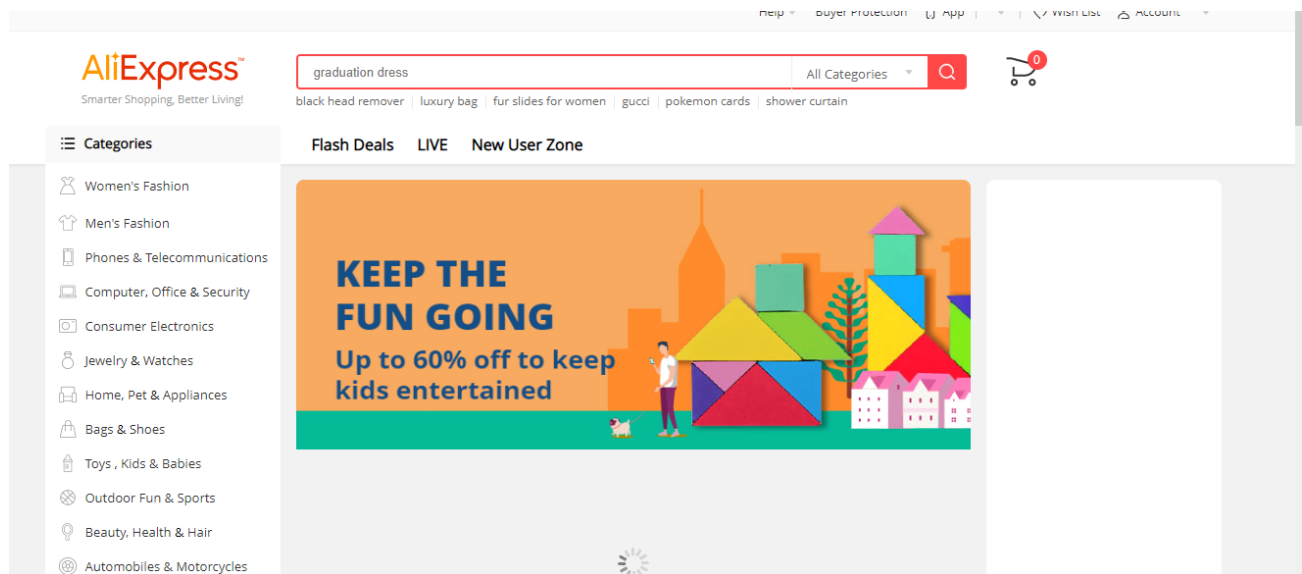


Рис. 1.1. Головна сторінка AliExpress

AliExpress - це інтернет-роздрібна послуга, що базується в Китаї і належить групі Alibaba. Створений у 2010 році, він складається з малого бізнесу в Китаї та інших місцях, таких як Сінгапур, які пропонують продукцію для міжнародних онлайн-покупців. Це сприяє малому бізнесу продавати клієнтам у всьому світі.

AliExpress здійснив порівняння з eBay, оскільки продавці є незалежними та використовують платформу, щоб пропонувати товари покупцям.

AliExpress стартував як портал купівлі-продажу бізнес-бізнесу. З цього часу вона розповсюдилася і на бізнес-споживач, і споживач-споживач, і хмарні обчислення, і платіжні послуги. Зараз AliExpress доступний мовами англійською, іспанською, голландською, французькою, італійською, німецькою, польською, португальською та російською мовами. Клієнти за межами країн для цих мов автоматично обслуговують англійську версію послуги. AliExpress часто використовується в магазинах електронної комерції, які використовують бізнес-модель дропшипа.

Продавцями на AliExpress можуть бути як компанії, так і фізичні особи. AliExpress відрізняється від Amazon тим, що діє лише як платформа електронної комерції та не продає продукцію безпосередньо споживачам. Це безпосередньо пов'язує китайський бізнес з покупцями. AliExpress відрізняється від компанії Alibaba-дочірньої компанії Таобао тим, що AliExpress орієнтований насамперед на міжнародних покупців.

AliExpress не дозволяє клієнтам у материковому Китаї купувати з платформи, хоча більшість підприємств роздрібної торгівлі самі є китайцями. На вебсайті пропонується популярна програма партнерського маркетингу, де партнери отримують винагороду за відправлення відвідувачів на сайт із комісією з продажів.

### 1.2.2. Taobao.com

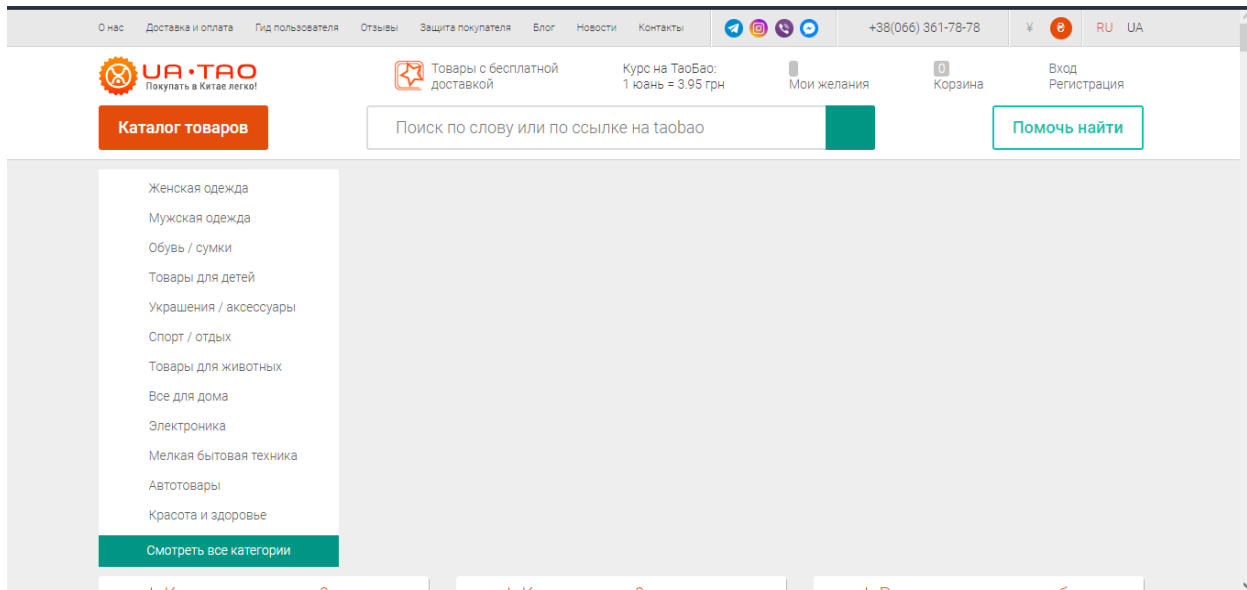


Рис. 1.2. Головна сторінка магазину Taobao

Taobao - це китайський інтернет-магазин для покупок, зі штаб-квартирою в Ханчжоу і належить Alibaba. Це найбільший у світі вебсайт електронної комерції та восьмий найбільш відвідуваний вебсайт за версією Alexa. Taobao.com зареєстрував 2003-04-21 з реєстратором домену Alibaba Cloud Computing (Beijing) Co., Ltd.

Вебсайт був заснований Alibaba Group у 2003 році. Taobao Marketplace сприяє роздрібній торгівлі споживачами (C2C), надаючи платформу для малого бізнесу та індивідуальних підприємців для відкриття інтернет-магазинів, які в основному обслуговують споживачів у китайськомовних регіонах (материковий Китай, Гонконг, Макао та Тайвань) та за кордоном, за рахунок яких здійснюється оплата через Інтернет-рахунки мобільних телефонів. Його магазини зазвичай пропонують експрес-доставку для клієнтів.

Маючи понад 1 мільярд списків товарів станом на 2016 рік, сукупний обсяг транзакцій Taobao Marketplace та Tmall.com у 2017 році досяг 3 трлн юанів, більше, ніж на всіх вебсайтах електронної комерції в США. The Economist

називає це "найбільшим інтернет-ринком в країні". Продавці можуть розміщувати товари на продаж або через фіксовану ціну, або на аукціоні. Аукціони складають невеликий відсоток угод. Більшість товарів - це нові товари, що продаються за фіксованими цінами. Покупці можуть оцінювати фон продавця за інформацією, доступною на сайті, включаючи рейтинги, коментарі та скарги.

Taobao Marketplace (раніше "Taobao") був запущений у травні 2003 року Alibaba після того, як eBay придбав Everynet, лідера інтернет-аукціонів у Китаї на той час, за 180 мільйонів доларів США і став головним гравцем на ринку електронної комерції в Китаї. Щоб протистояти розширенню eBay, Taobao запропонував безкоштовні списки продавців та представив функції вебсайтів, розроблені таким чином, щоб діяти в інтересах місцевих споживачів, такі як миттєві повідомлення для полегшення спілкування покупця-продавця та платіжний інструмент Alipay на основі ескроу. Як результат, Taobao став беззаперечним лідером ринку Китаю протягом двох років. Його частка на ринку зростає з 8% до 59% між 2003 та 2005 роками, тоді як eBay Китай знизився з 79% до 36%. eBay заклав свій сайт у Китаї у 2006 році.

У квітні 2008 року Taobao представив нещодавно виділену B2C-платформу під назвою Taobao Mall, щоб доповнити її ринок C2C. Торговий центр Taobao зарекомендував себе як місце для якісних товарних знаків для китайських споживачів. Компанія Taobao Mall запустила незалежний вебдомен Tmall.com і в листопаді 2010 р. Посилила свою увагу на вертикалі продуктів та вдосконаленням досвіду покупок. У червні 2011 року вона стала самостійним бізнесом і змінила китайське ім'я на Tian Mao (Tmall) у січні 2012 року. Станом на 2024 року це був восьмий найбільш відвідуваний вебсайт у Китаї.

У 2008 році Taobao сприяв загальному зростанню китайської індустрії торгових мереж через виконання стратегії "Big Taobao" з метою стати



постачальником послуг інфраструктури електронної комерції для всіх учасників ринку електронної комерції.

У жовтні 2010 року бета-версія Таобао запустила eТао як незалежну пошукову систему для інтернет-покупок, надаючи інформацію про товари та продавця з багатьох основних вебсайтів електронної комерції споживачів у Китаї. Інтернет-покупці можуть скористатися сайтом для порівняння цін у різних продавців та виявлення товарів, які купують. Згідно з вебсайтом групи Alibaba, eТао пропонує продукцію з Китаю Амазонки, Дангдангу, Гоме, Іхаодіана, Nike China та Vancle, а також Таобао і Tmall.

У травні 2011 року Alibaba Group відкрила роздрібний магазин в Пекіні під торговою маркою Таобао Mall. П'ятиповерховий салон меблів iFengChao 25000 квадратних метрів відкрився як доповнення до їх інтернет-магазинів.

### 1.2.3. Rozetka.ua

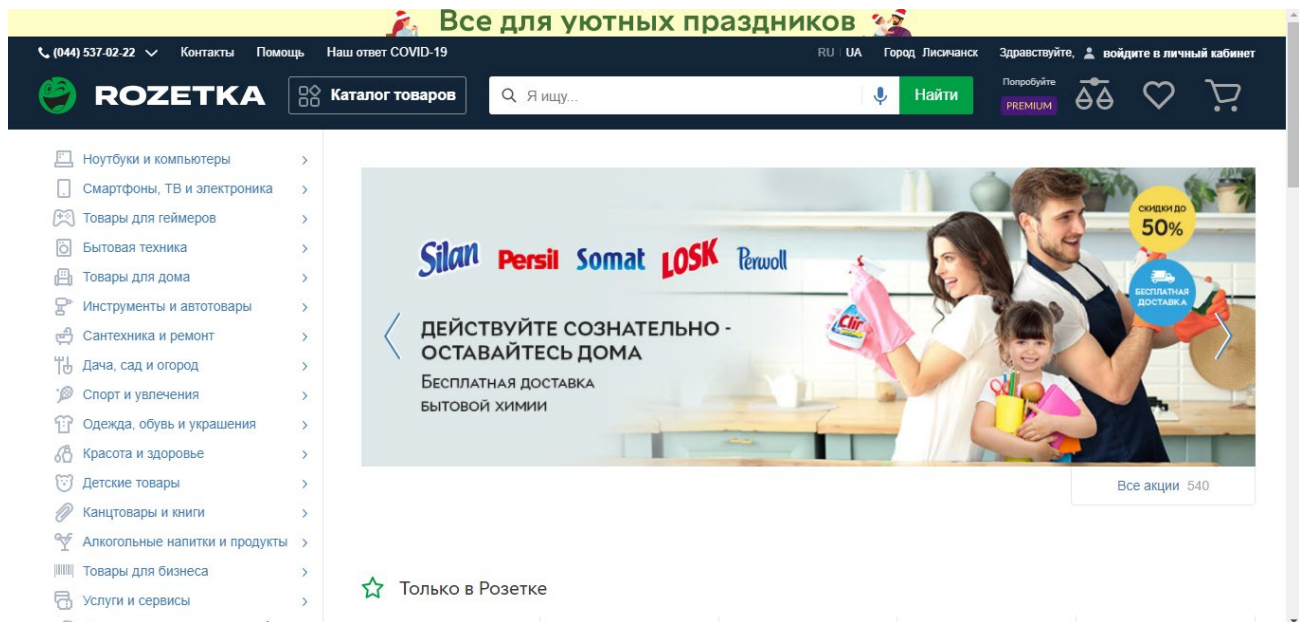


Рис. 1.3. Головна сторінка Rozetka

Rozetka.ua — український інтернет-магазин та маркетплейс, що з'явився 2005 року. Має відділення в Києві, Броварах та Одесі. Станом на серпень 2020 року сайт посідає 7 місце серед найвідвідуваніших в Україні.

З 2005 по 2016 року керівники компанії відмовлялись створити україномовну версію сайту, попри багаторічні скарги українських споживачів. Лише 2016-го року під громадським тиском було запущено тестову версію україномовного інтерфейсу, однак досі не всі товари мають український опис та характеристику.

Відсутність українського інтерфейсу не раз викликала претензії українських користувачів, які позивались до суду на магазин. Лише в кінці жовтня 2016 року, після кількох років суперечок та під тиском громадськості, магазин запустив тестову україномовну версію інтерфейсу, яка стала доступною лише деяким відвідувачам. Згодом, після закінчення тестування, ця версія стала доступною для всіх користувачів. Типово (за замовчуванням) встановлена російська версія, яку можна змінити на українську.

#### 1.2.4. OLX

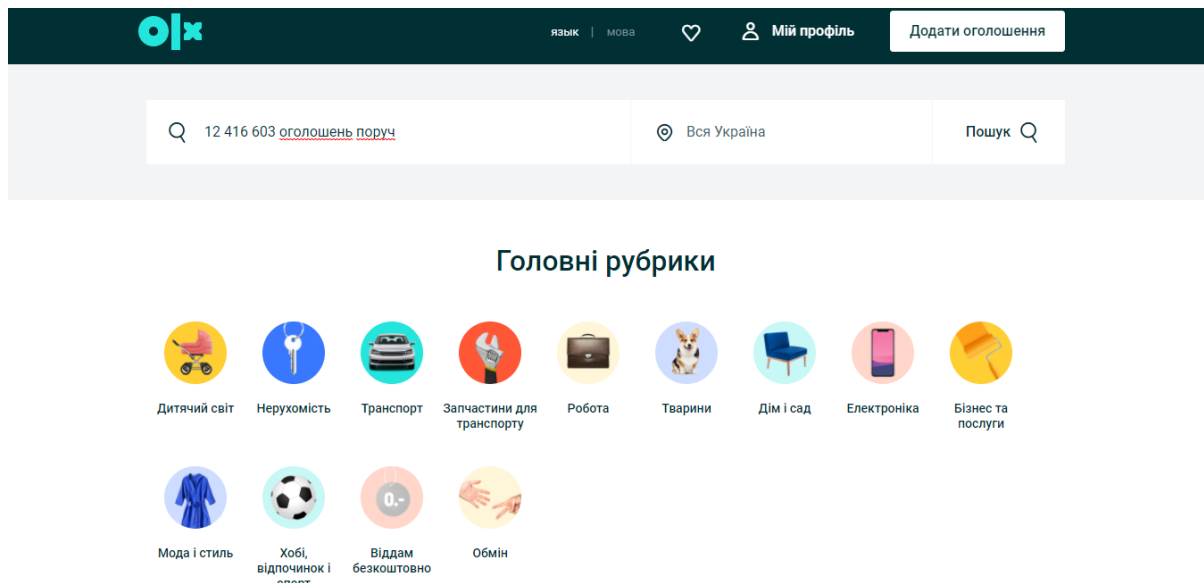


Рис. 1.4. Головна сторінка OLX

OLX (англ. OnLine eXchange) — це онлайн-платформа для оголошень, яка об'єднує користувачів для купівлі, продажу або обміну товарами та послугами. Оголошення розподіляються за категоріями, такими як: «Дитячий світ», «Нерухомість», «Транспорт», «Запчастини для транспорту», «Робота», «Тварини», «Дім і сад», «Електроніка», «Бізнес і послуги», «Мода і стиль», «Хобі, відпочинок і спорт», «Віддам безкоштовно» та «Обмін». OLX є найбільш популярним сервісом оголошень в Україні: щомісяця його відвідує кожен другий український інтернет-користувач.

*Історія.* OLX засновано у березні 2006 року підприємцями Фабрісом Гріндою та Алехандро Оксенфордом. Початково компанія активно працювала на ринку Індії, який на той час вважався одним із найбільш перспективних. У 2010 році OLX став частиною медіагрупи Naspers, а у 2012 році до команди приєднався Мартін Шіпбауер, нинішній виконавчий директор OLX Group. У цьому ж році платформа стала найбільшим онлайн-ресурсом оголошень в Індії, а у 2013 році — у Бразилії.

У квітні 2014 року платформи оголошень групи Naspers у Румунії, Болгарії, Казахстані, Білорусі, Угорщині та Польщі змінили назву на OLX. У вересні того ж року український сайт Slando також провів ребрендинг і приєднався до мережі OLX Group.

З вересня 2015 року українська платформа стала частиною міжнародної мережі OLX Group, яка працює у понад 40 країнах світу.

*Функціональність.* OLX надає платформу для купівлі та продажу товарів і послуг як приватним особам, так і бізнесу. Користувачі можуть реєструватися через мобільний телефон, електронну пошту або соціальні мережі, після чого додавати свої оголошення із детальним описом, фотографіями та контактними даними.

*Мобільні додатки та сервіси.* OLX використовує власні мобільні додатки для Android і iOS. У травні 2017 року платформа впровадила послугу доставки, реалізовану спільно з «УАРАУ» і «Нова пошта». Сервіс блокує кошти на карті покупця, списуючи їх лише після отримання товару, що підвищує безпеку угод. Подібні сервіси доступні на AliExpress, Amazon та eBay.

Таблиця 1.1 – Порівняльна характеристика вебсайтів аналогів

Характеристика	AliExpress	Tabao.com	Rozetka	OLX
Особистий кабінет	+	+	+	+
Наявність кошика	+	+	+	-
Фільтри пошуку	+	+	+	+
Зрозумілий дизайн	+	+/-	+	+
Кількість різних мов	12	1	2	2
Наявність слайдів	+	+	+	-

### 1.3 Вибір архітектури системи

Архітектура програмного забезпечення стосується фундаментальних структур програмної системи та дисципліни створення таких структур і систем. Кожна структура містить програмні елементи, відносини між ними та властивості як елементів, так і відносин. Архітектура програмної системи - це метафора, аналогічна архітектурі будівлі. Він функціонує як концепція системи та проекту, що розробляється, викладаючи завдання, необхідні для виконання проектними командами.

Архітектура програмного забезпечення полягає у прийнятті фундаментальних структурних виборів, які дорого змінити після їх впровадження. Вибір архітектури програмного забезпечення включає конкретні структурні

варіанти від можливостей розробки програмного забезпечення. Наприклад, системи, які керували ракетним апаратом Space Shuttle, вимагали бути дуже швидкими та дуже надійними. Тому потрібно вибрати відповідну мову обчислень у режимі реального часу. Крім того, для задоволення потреби в надійності вибір може бути зроблений з кількох надмірними та незалежно виготовленими копіями програми, а також запускати ці копії на незалежному обладнанні під час перехресної перевірки результатів.

Документація архітектури програмного забезпечення полегшує спілкування між зацікавленими сторонами, фіксує ранні рішення щодо дизайну високого рівня та дозволяє повторно використовувати компоненти дизайну між проектами.

Думки різняться щодо сфери архітектури програмного забезпечення:

- Макроскопічна структура системи: це відноситься до архітектури як до абстрагування вищого рівня програмної системи, що складається з набору обчислювальних компонентів разом із роз'ємами, що описують взаємодію між цими компонентами.

- Важливий матеріал - що б там не було: це стосується того, що архітектори програмного забезпечення повинні стосуватися тих рішень, які мають високий вплив на систему та її зацікавлених сторін.

- Те, що є основним для розуміння системи в її оточенні
- Те, що люди сприймають як важко змінити: оскільки проектування архітектури відбувається на початку життєвого циклу програмної системи, архітектор повинен зосередитись на рішеннях, які "повинні" бути правильними з першого разу. Дотримуючись такої лінії думки, питання архітектурного дизайну можуть стати не архітектурними, коли їх незворотність може бути подолана.

- Набір архітектурних дизайнерських рішень: архітектура програмного забезпечення не повинна розглядатися лише як сукупність моделей чи структур, але повинна включати рішення, що ведуть до цих конкретних структур, та

обґрунтування їх. Це розуміння призвело до значних досліджень управління знаннями архітектури програмного забезпечення.

Не існує різкого розрізнення між архітектурою програмного забезпечення та дизайном та вимогами (див. Відповідні поля нижче). Всі вони є частиною "ланцюжка інтенціональності" від намірів високого рівня до деталей низького рівня.

Архітектура програмного забезпечення демонструє наступне [17]:

- Безліч зацікавлених сторін: програмні системи повинні обслуговувати різні зацікавлені сторони, такі як керівники бізнесу, власники, користувачі та оператори. Усі ці зацікавлені сторони мають свої проблеми щодо системи. Врівноваження цих проблем та демонстрація їх вирішення є частиною проектування системи. Це означає, що архітектура передбачає вирішення різноманітних проблем та зацікавлених сторін і має багатодисциплінарний характер.

- Розділення проблем: усталений спосіб архітекторів зменшити складність - це відокремити проблеми, які зумовлюють дизайн. Документація щодо архітектури показує, що всі проблеми зацікавлених сторін вирішуються шляхом моделювання та опису архітектури з окремих точок зору, пов'язаних з різними проблемами зацікавлених сторін. Ці окремі описи називаються архітектурними видами

- На основі якості: класичні підходи до проектування програмного забезпечення (наприклад, Структуроване програмування Джексона) визначалися необхідною функціональністю та потоком даних через систему, але поточне розуміння полягає в тому, що архітектура програмної системи є більш тісною пов'язані з його якісними характеристиками, такими як відмовостійкість, зворотна сумісність, розширюваність, надійність, ремонтпридатність, доступність, безпека, зручність використання та інші подібні засоби. Проблеми зацікавлених

сторін часто перетворюються на вимоги щодо цих атрибутів якості, які по-різному називаються нефункціональними вимогами, позафункціональними вимогами, вимогами до поведінки або атрибутами якості.

- Повторювані стилі: як архітектура будівлі, дисципліна архітектури програмного забезпечення розробила стандартні способи вирішення проблем, що виникають. Ці "стандартні способи" називаються різними назвами на різних рівнях абстракції. Загальними термінами для повторюваних рішень є архітектурний стиль, тактика, довідкова архітектура та архітектурний зразок.

- Концептуальна цілісність: термін, введений Фредом Бруксом в «Міфічний чоловік-місяць» для позначення ідеї про те, що архітектура програмної системи являє собою загальне бачення того, що вона повинна робити і як це робити. Це бачення слід відокремити від його реалізації. Архітектор бере на себе роль "зберігача бачення", переконуючись, що доповнення до системи відповідають архітектурі, отже, зберігаючи концептуальну цілісність.

- Когнітивні обмеження: спостереження, вперше зроблене в документі 1967 року комп'ютерним програмістом Мелвіном Конвеєм, що організації, які проектують системи, обмежують виробляти конструкції, які є копіями структур зв'язку цих організацій. Як і щодо концептуальної цілісності, саме Фред Брукс познайомив її з широкою аудиторією, коли він цитував газету та ідею у своїй елегантній класиці «Міфічний чоловік-місяць», називаючи її «законом Конвея».

Модель – перегляд – контролер (зазвичай відомий як MVC) - це модель дизайну програмного забезпечення, яка зазвичай використовується для розробки інтерфейсів користувача, яка розділяє відповідну логіку програми на три взаємопов'язані елементи. Це робиться для того, щоб відокремити внутрішнє представлення інформації від способів подання та прийняття інформації від користувача. Цей вид візерунка використовується для проектування макета сторінки.

Традиційно використовується для графічних інтерфейсів користувачів настільних ПК (GUI), ця модель стала популярною для розробки вебдодатків. Популярні мови програмування, такі як JavaScript, Python, Ruby, PHP, Java, C # і Swift, мають MVC-рамки, які використовуються для розробки веб або мобільних додатків прямо з вікна.

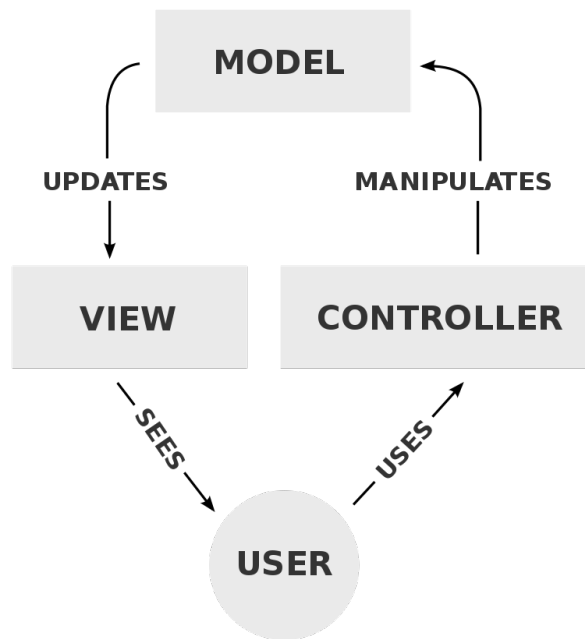


Рис. 1.5. Діаграма роботи MVC

### *Модель*

Центральний компонент візерунка. Це динамічна структура даних програми, незалежна від користувацького інтерфейсу. [5] Він безпосередньо керує даними, логікою та правилами програми.

### *Вид*

Будь-яке представлення інформації, наприклад діаграми, діаграми або таблиці. Можливі кілька переглядів однієї й тієї самої інформації, такі як гістограма для менеджменту та таблична таблиця для бухгалтерів.

### *Контролер*

Приймає введення та перетворює його в команди для моделі чи перегляду.



На додаток до поділу програми на ці компоненти, модель модель-перегляд – контролер визначає взаємодію між ними.

- Модель відповідає за управління даними програми. Він отримує введення користувача від контролера.
- Перегляд означає представлення моделі у певному форматі.
- Контролер відповідає на введення користувача та здійснює взаємодію з об'єктами моделі даних. Контролер отримує вхід, додатково перевіряє його, а потім передає вхід моделі.

Як і в інших моделях програмного забезпечення, MVC виражає «серцевину рішення» проблеми, дозволяючи пристосувати її до кожної системи. Конкретні конструкції MVC можуть суттєво відрізнятися від традиційного опису тут.

Незважаючи на те, що спочатку розроблявся для настільних обчислень, MVC отримав широке поширення як дизайн для світових вебдодатків на основних мовах програмування. Було створено декілька веббрамок, які застосовують схему. Ці рамки програмного забезпечення відрізняються за своєю інтерпретацією, головним чином тим, що обов'язки MVC розподіляються між клієнтом та сервером.

Деякі рамки веб MVC використовують тонкий клієнтський підхід, який розміщує майже всю логіку моделі, перегляду та контролера на сервері. Це відображено в таких структурах, як Django, Rails та ASP.NET MVC. У такому підході клієнт надсилає або запити гіперпосилання, або подання форми до контролера, а потім отримує повну та оновлену вебсторінку (або інший документ) з подання; модель існує повністю на сервері. Інші рамки, такі як AngularJS, EmberJS, JavaScriptMVC та Backbone, дозволяють компонентам MVC частково виконуватись на клієнті.

Оскільки MVC роз'єднує різні компоненти програми, розробники можуть працювати паралельно над різними компонентами, не впливаючи та не блокуючи

один одного. Наприклад, команда може розділити своїх розробників між передніми та задніми. Резервні розробники можуть спроектувати структуру даних та спосіб взаємодії з ними, не вимагаючи доповнення користувальницького інтерфейсу. І навпаки, розробники передумов можуть розробити та протестувати макет програми до того, як доступна структура даних.

Один і той же (або подібний) вигляд для однієї програми може бути відновлений для іншої програми з різними даними, оскільки представлення просто обробляє, як дані відображаються користувачеві. На жаль, це не працює, коли цей код також корисний для обробки даних користувача. Наприклад, DOM-код (включаючи власні абстракції програми до нього) корисний як для графічного відображення, так і для введення користувача. (Зверніть увагу, що, незважаючи на назву Document Object Model, DOM насправді не є моделлю MVC, оскільки це інтерфейс програми для користувача) [6].

Для вирішення цих проблем MVC (і схожі на нього візерунки) часто поєднуються з архітектурою компонентів, яка забезпечує набір елементів інтерфейсу. Кожен елемент інтерфейсу - це один компонент вищого рівня, який поєднує 3 необхідні компоненти MVC в єдиний пакет. Створюючи незалежні один від одного компоненти вищого рівня, розробники мають змогу швидко та легко використовувати компоненти в інших додатках.

## РОЗДІЛ 2. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

### 2.1. TypeScript

TypeScript - мова програмування з відкритим кодом, розроблена та підтримувана Microsoft. Це суворий синтаксичний набір JavaScript і додає до мови обов'язкове статичне введення тексту. TypeScript призначений для розробки великих програм та транскопіляцій у JavaScript. Оскільки TypeScript є набором JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript.

TypeScript може використовуватися для розробки програм JavaScript для виконання як на стороні клієнта, так і на стороні сервера (як у Node.js або Deno). Існує кілька варіантів транскопіляції. Може використовуватися перевірка TypeScript за замовчуванням, або компілятор Babel може бути викликаний для перетворення TypeScript в JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про типи існуючих бібліотек JavaScript, як і файли заголовків C ++, можуть описувати структуру існуючих об'єктних файлів. Це дає можливість іншим програмам використовувати значення, визначені у файлах, так, як якщо б вони були статично набраними сутностями TypeScript. Існують сторонні файли заголовків для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Також доступні заголовки TypeScript для основних модулів Node.js, що дозволяє розробляти програми Node.js в TypeScript.

Компілятор TypeScript сам пишеться в TypeScript і компілюється в JavaScript. Він ліцензований за ліцензією Apache 2.0. TypeScript включений як першокласна мова програмування в Microsoft Visual Studio 2013 Update 2 та новіших версій, поряд із C # та іншими мовами Microsoft. Офіційне розширення дозволяє Visual Studio 2012 також підтримувати TypeScript. Андер Хейлсберг,

провідний архітектор C # і творець Delphi і Turbo Pascal, працював над розробкою TypeScript [7].

Вперше TypeScript був оприлюднений у жовтні 2012 року (у версії 0.8) після двох років внутрішньої розробки в Microsoft. Незабаром після оголошення Мігель де Іказа похвалив саму мову, але розкритикував відсутність зрілої підтримки IDE, окрім Microsoft Visual Studio, яка в той час не була доступною для Linux та OS X. Сьогодні існує підтримка в інших IDE, зокрема в Eclipse, за допомогою плагіна, який надає Palantir Technologies. Різні текстові редактори, включаючи Emacs, Vim, Webstorm, Atom та власний код Visual Studio Microsoft, також підтримують TypeScript.

TypeScript 0.9, випущений у 2013 році, додав підтримку для дженериків. TypeScript 1.0 був випущений на конференції розробників Microsoft у 2014 році. Visual Studio 2013 Update 2 забезпечує вбудовану підтримку TypeScript.

У липні 2014 року команда розробників оголосила про новий компілятор TypeScript, який вимагає  $5 \times$  підвищення продуктивності. Одночасно вихідний код, який спочатку розміщувався на CodePlex, був переміщений до GitHub.

22 вересня 2016 року було випущено TypeScript 2.0; він ввів декілька функцій, включаючи можливість програмістів необов'язково запобігати присвоєнню змінним нульових значень, іноді їх називають помилкою в мільярд доларів.

TypeScript 3.0 був випущений 30 липня 2018 року, який містить багато мовних доповнень, таких як кортежі в параметрах спокою та вирази розповсюдження, параметри відпочинку з типами кортежу, загальні параметри відпочинку тощо.

TypeScript виникла з недоліків JavaScript для розробки масштабних додатків як у Microsoft, так і серед їх зовнішніх клієнтів. Проблеми, що стосуються роботи

зі складним кодом JavaScript, спричинили потребу в користувальницьких інструментах для полегшення розробки компонентів мови.

Розробники TypeScript шукали рішення, яке не порушило б сумісність зі стандартом та його міжплатформною підтримкою. Знаючи, що поточна стандартна пропозиція ECMAScript обіцяє майбутню підтримку програмування на основі класів, TypeScript базувався на цій пропозиції. Це призвело до компілятора JavaScript з набором синтаксичних розширень мови, суперсети на основі пропозиції, що перетворює розширення в звичайний JavaScript. У цьому сенсі TypeScript був попереднім переглядом того, чого слід очікувати від ECMAScript 2015. Унікальним аспектом, який не в пропозиції, а доданий до TypeScript, є не обов'язкове статичне введення тексту, що дозволяє статичний аналіз мови, що полегшує інструментарій та підтримку IDE.

TypeScript є суворим набором ECMAScript 2015, який сам по собі є набором ECMAScript 5, який зазвичай називають JavaScript. Таким чином, програма JavaScript також є дійсною програмою TypeScript, і програма TypeScript може безперешкодно споживати JavaScript. За замовчуванням компілятор орієнтований на ECMAScript 5, поточний переважаючий стандарт, але також може генерувати конструкції, що використовуються в ECMAScript 3 або 2015.

За допомогою TypeScript можна використовувати існуючий код JavaScript, включати популярні бібліотеки JavaScript та викликати код, сформований TypeScript з іншого JavaScript. Декларації типів для цих бібліотек надаються вихідним кодом [2].

*Середовища і редактори.*

- Microsoft надає плагін для Visual Studio 2012 та WebMatrix, повну інтегровану підтримку Visual Studio 2013, Visual Studio 2015 та підтримку базового текстового редактора для Emacs та Vim.

- Visual Studio Code - це (в основному) редактор вихідного коду з платформою з відкритим кодом, розроблений Microsoft на базі Electron. Він підтримує TypeScript на додаток до кількох інших мов і пропонує такі функції, як налагодження та інтелектуальне завершення коду.
- alm.tools - це хмара IDE з відкритим кодом для TypeScript, побудована за допомогою TypeScript, ReactJS та TypeStyle.
- JetBrains підтримує TypeScript з доповненням коду, рефакторингом та налагодженням його IDE, побудованих на платформі IntelliJ, таких як PhpStorm 6, WebStorm 6 та IntelliJ IDEA, , а також їх надбудови та розширення Visual Studio, ReSharper 8.1.
- У Atom є плагін TypeScript від Basarat з підтримкою завершення коду, навігації, форматування та швидкої компіляції.
- Онлайн Cloud9 IDE та Codenvy підтримують TypeScript.
- Плагін доступний для IDE NetBeans.
- Плагін доступний для IDE Eclipse (версія Kepler)
- TypEcs доступний для IDE Eclipse.
- Cross Cloud Cloud IDE Codeanywhere підтримує TypeScript.
- Webclipse Плагін Eclipse, призначений для розробки TypeScript і Angular 2.
- Angular IDE Автономний IDE, доступний через npm для розробки додатків TypeScript і Angular 2, з інтегрованою підтримкою терміналів.
- Tide - Інтерактивне середовище розробки TypeScript для Emacs.

## 2.2. Angular

AngularJS - це вебсистема з відкритим вихідним кодом, заснована на JavaScript, в основному підтримується Google та спільнотою осіб та корпорацій для вирішення багатьох проблем, що виникають при розробці програм на одній

сторінці. Він спрямований на спрощення як розробки, так і тестування таких додатків, забезпечуючи структуру для клієнтської архітектури модель-перегляд-контролер (MVC) та модель-перегляд-перегляд-модель (MVVM), а також компоненти, що часто використовуються в багатих Інтернет-додатках.

AngularJS - це пряма частина стека MEAN, що складається з бази даних MongoDB, рамки сервера вебдодатків Express.js, самого Angular.js та середовища виконання сервера Node.js. Версія 1.7.x працює на довгостроковій підтримці до 1 липня 2021 року. Після цієї дати AngularJS більше не оновлюватиметься, а натомість запропоновано Angular (2.0+).

Рамка AngularJS працює, попередньо прочитавши сторінку мови розмітки гіпертексту (HTML), на якій вбудовані додаткові спеціальні атрибути HTML. Кутова інтерпретує ці атрибути як директиви для прив'язки вхідних або вихідних частин сторінки до моделі, яка представлена стандартними змінними JavaScript. Значення цих змінних JavaScript можна встановити вручну в коді або отримати зі статичних або динамічних ресурсів JSON [11].

При створенні інтерфейсів користувача та підключення програмних компонентів у AngularJS використовується декларативний підхід для програмування. Для визначення ділової логіки програми використовується імперативне програмування. AngularJS розширює традиційний HTML для представлення динамічного контенту за допомогою двостороннього зв'язування даних, що дозволяє здійснювати автоматичну синхронізацію. Як результат, AngularJS децензує увагу на явній маніпуляції з об'єктною моделлю документа (DOM) з метою поліпшення перевірки та продуктивності [8].

Цілі проектування AngularJS включають:

- щоб від'єднати маніпуляцію DOM від логіки програми. На складність цього різко впливає спосіб структурування коду.

- для роз'єднання клієнтської програми з боку сервера. Це дозволяє паралельно прогресувати роботи з розробки та дозволяє повторно використовувати обидві сторони.
- надати структуру для побудови програми: від проектування інтерфейсу, через написання ділової логіки, до тестування.

AngularJS реалізує шаблон MVC для розділення компонентів презентації, даних та логіки. Використовуючи введення залежностей, Angular традиційно постачає сервіси на стороні сервера, такі як контролери, залежні від перегляду, до вебдодатків на стороні клієнта. Отже, значна частина навантаження на сервер може бути зменшена.

AngularJS використовує термін "сфера дії" таким чином, що схожий на основи інформатики.

Сфера застосування інформатики описує, коли в програмі діє певне прив'язка. Специфікація ECMA-262 визначає область як: лексичне середовище, в якому об'єкт функції виконується в вебскриптах на стороні клієнта; схожий на те, як визначається область застосування в обчисленні лямбда.

Як частина архітектури "MVC", область містить "Модель", а всі змінні, визначені в області, можуть отримати доступ до "Перегляду", а також "Контролера". Область поводить як клей і зв'язує «Вид» і «Контролер».

Завдання, яке виконується завантажувачем AngularJS, відбувається в три фази після завантаження DOM [4]:

1. Створення нового інжектора
2. Складання директив, що прикрашають DOM
3. Зв'язування всіх директив із сферою застосування

Директиви AngularJS дозволяють розробнику задавати користувацькі та багаторазові HTML-подібні елементи та атрибути, що визначають прив'язку даних



та поведінку компонентів презентації. Деякі з найбільш часто використовуваних директив:

`ng-animate`

Модуль забезпечує підтримку JavaScript, CSS3 переходу та анімаційних гачків анімації ключових кадрів CSS3 в рамках існуючих основних та спеціальних директив.

Оскільки атрибути `ng-*` не відповідають дійсності в специфікаціях HTML, дані `ng-*` можуть також використовуватися як префікс. Наприклад, і `ng-app`, і `data-ng-` додаток дійсні в AngularJS.

`ng-app` - Заявляє кореневий елемент програми AngularJS, згідно з яким директиви можуть використовуватися для оголошення прив'язок та визначення поведінки.

`ng-aria` - Модуль підтримки доступності загальних атрибутів ARIA.

`ng-bind` - Встановлює текст елемента DOM на значення виразу. Наприклад, `<span ng-bind = "name"> </span>` відображає значення "name" всередині елемента `span`. Будь-яка зміна змінної "ім'я" в області застосування програми негайно відображається в DOM.

`ng-class` - Застосовує клас, залежно від значення булевого виразу.

`ng-controller` - Визначає клас контролера JavaScript, який оцінює вирази HTML.

`ng-if` - Основна, якщо директива твердження, яка створює наступний елемент, якщо умови справджуються. Коли умова помилкова, елемент видаляється з DOM. Якщо це правда, клон складеного елемента повторно вставляється.

`ng-init` - Викликається один раз при ініціалізації елемента.

`ng-model` - Аналогічно `ng-bind`, але встановлює двостороннє зв'язування даних між представленням і областю.

`ng-model-options` - Забезпечує налаштування способів оновлення моделі.

`ng-repeat` - Створює кілька екземплярів елемента, для кожного об'єкта колекції.

`ng-show & ng-hide` - Показують чи ховають елемент залежно від значення булевого виразу. Показати та приховати досягається встановленням стилю відображення CSS.

`ng-switch` - Створює екземпляр шаблону з безлічі варіантів, в залежності від значення виразу.

`ng-view` - Базова директива, відповідальна за обробку маршрутів, які вирішують JSON перед наданням шаблонів, керованих зазначеними контролерами.

AngularJS був спочатку розроблений у 2009 році Мішко Гевери у компанії Brat Tech LLC [14] як програмне забезпечення, що стоїть на онлайновому сервісі зберігання даних JSON, яке було б коштуватиме мегабайт, для зручних для роботи підприємств. Це підприємство було розміщене у вебдомені "GetAngular.com".

Випуск 1.6 додав багато понять Angular до AngularJS, включаючи концепцію архітектури додатків на основі компонентів.

Подальші версії AngularJS просто називають Angular. Angular - це несумісне перезапис AngularJS на основі TypeScript.

AngularDart працює над Dart, який є об'єктно-орієнтованим, визначеним класом, єдиною мовою програмування успадкування, використовуючи синтаксис стилю C, який відрізняється від Angular JS (для якого використовується JavaScript) та Angular 2 / Angular 4 (для якого використовується TypeScript). Angular 4 випущений у березні 2017 року, при цьому версія рамки узгоджується з номером версії маршрутизатора, який він використовував. Angular 5 вийшов 1 листопада 2017 року. Основні вдосконалення в Angular 5 включають підтримку прогресивних вебдодатків, оптимізатор збірки та вдосконалення, пов'язані з

дизайном матеріалів. Angular 6 було випущено 3 травня 2018 року, Angular 7 було випущено 18 жовтня 2018 року, а Angular 8 було випущено 28 травня 2019 року. Angular дотримується стандартів семантичної версії, при цьому кожен головний номер версії вказує на потенційні порушення. Angular зобов'язався забезпечити 6 місяців активної підтримки для кожної основної версії, а потім 12 місяців довгострокової підтримки. Основні випуски є щорічно, з 1 до 3 незначних випусків для кожного головного випуску [19].

## РОЗДІЛ 3. ПРОЕКТУВАННЯ СИСТЕМИ

### 3.1. Визначення та аналіз вимог до системи

IEEE (Institute of Electrical and Electronics Engineers – Інститут інженерів електротехники і електроніки) Standard Glossary of Software Engineering Terminology визначає вимоги як:

- умови або можливості, необхідні користувачеві для вирішення проблем або досягнення цілей;
- умови або можливості, якими повинна володіти система або системні компоненти, щоб виконати контракт або задовольняти стандартам, специфікаціям або іншим формальним документам;

Бізнес – правило (business rule) – політика, припис, стандарт, правило або обчислювальна формула, яка визначає чи обмежує деякі сторони бізнес – процесів.

Бізнес – вимоги (business requirement) – обсяг інформації, який в сукупності описує потреба, яка ініціює один або більше проектів, покликаних надати рішення і отримати необхідний кінцевий бізнес – результат. Бізнес – вимоги включають бізнес – можливості, бізнес – цілі, метрики успіху, концепція і кордони і обмеження.

Обмеження (constraint) – накладається на доступні розробнику можливості дизайну або конструювання продукту. Інші типи обмежень можуть обмежити можливості, доступні для менеджерів проектів. Бізнес – правила часто накладають обмеження на бізнес – операції, а значить, на програмні системи.

Функціональна вимога (functional requirement) – опис поведінки системи в певних умовах.

Атрибут якості (quality attribute) – вид нефункціонального вимоги, що описує характеристику сервісу або продуктивності продукту. Приклади атрибутів якості: зручність і простота використання, легкість переміщення, легкість в експлуатації, цілісність, надійність, ефективність і стійкість до збоїв. У вимогах описані рамки атрибутів якості, до яких продукт демонструє бажані характеристики.

Користувацька вимога (user requirement) – мета і завдання, яку користувачі повинні мати можливість виконувати з системою, або положення про очікування користувачів про якість системи. Призначені для користувача вимоги зазвичай подаються у вигляді варіантів використання, призначених для користувача історій і сценаріїв.

Системне вимога (system requirement) – вимога верхнього рівня до продукту, що складається з багатьох підсистем, які можуть являти собою ПЗ або сукупність ПЗ і устаткування [14].

Вимоги для інтерфейсу зовнішнього пристрою (external interface requirement) – опис інтерфейсу між системою ПЗ і користувачем, іншою системою ПЗ або обладнанням.

Нефункціональна вимога (nonfunctional requirement) – опис властивих властивостей або характеристик, які система ПЗ повинна демонструвати, або обмеження, які вона повинна дотримуватися [14].

Вимоги до системи інтернет-магазину за різними типами інформації наведено у таблиці 3.1.

Таблиця 3.1 – Вимоги до системи тайм – трекеру

Тип вимоги	Вимоги до системи
Бізнес – правило	Неможливо продавати товар, якого немає на складі. Неможливо продати товару більше, ніж є на складі
Функціональні вимоги	<ul style="list-style-type: none"> <li>• контроль кількості товару</li> <li>• контроль наявності товару</li> <li>• контроль доступності товару</li> </ul>
Системні вимоги	Система повинна мати доступ до інтернету та підтримувати інструментальні засоби, використані при розробці
Вимоги користувачів	<p>Адміністратор:</p> <ul style="list-style-type: none"> <li>• доступ в систему за особистим логіном та паролем;</li> <li>• повернення товарів</li> <li>• оформлення замовлень</li> <li>• перегляд інформації про товари</li> <li>• додавання товарів</li> <li>• видалення товарів</li> <li>• контроль кількості товарів</li> </ul>
Нефункціональні вимоги	Повинна підтримуватись операційними системами сімейства Windows та UNIX.
Обмеження	<ul style="list-style-type: none"> <li>• Графічний інтерфейс повинен бути</li> </ul>

	інтуїтивно зрозумілим <ul style="list-style-type: none"> <li>• Масштабування сайту за розміром дисплея девайсу</li> </ul>
Атрибут якості	Система повинна контролювати кількість товару і забороняти оформлювати замовлення на товари, яких немає на складі

### 3.2. Розробка специфікації вимог SRS

Згідно рекомендованої методики складання специфікацій вимог до програмного забезпечення Інституту Інженерів з Електротехніки та Радіоелектроніки (Стандарт IEEE 830 – 1998) належним чином складена специфікація повинна [17]:

- замовникам програмного забезпечення точно описати, що вони хочуть отримати;
- постачальникам програмного забезпечення точно зрозуміти, що хоче замовник;
- розробити макет стандартної специфікації програмного забезпечення (SRS) для їх власних організацій;
- визначити формат і зміст конкретних специфікацій вимог до програмного забезпечення;
- розробити додаткові допоміжні документи, такі як контрольний лист для перевірки якості SRS або довідник упорядника SRS.

Система контролю кількості товару на складі направлена на запобігання помилкових замовлень, які принесуть збитки підприємству, яке утримує даний конкретний магазин. Головне завдання подібних систем — підняти ефективність

бізнес-процесів за рахунок контролю помилкових замовлень та відсутності необхідності в вирішенні конфліктних ситуацій, які базуються на цій проблемі.

Специфікація вимог до програмного забезпечення.

Введення.

Призначення.

Ця специфікація вимог до ПЗ описує функціональні та нефункціональні вимоги до інтернет-магазину та системи контролю кількості товару на складі. Цей документ призначений для підприємства, яке буде перевіряти та реалізовувати коректність роботи системи.

Область дії.

Система контролю товару на складі інтернет-магазину дозволить співробітникам вберегтися від необхідності вирішувати конфліктні ситуації, які можуть виникати в наслідок того, що клієнт замовляє і оплачує товар, якого немає на складі.

Аналогічні системи розглянуті за допомогою інтернет – ресурсів: [www.aliexpress.com](http://www.aliexpress.com), [www.amazon.com](http://www.amazon.com), [taobao.com](http://taobao.com).

Далі у документі специфікації буде надано інформацію про спеціалізацію продукту, його функціональні та специфічні вимоги, обмеження системи.

Розроблюваний продукт – це проста система контролю та обліку товарів на складі інтернет-магазину, яка контролює кількість, оповіщає про те, що товар закінчився на складі, або про те, що даний товар неможливо купити через те, що він закінчився, або його кількість недостатня для даного замовлення.

Інтерфейси користувача:

- система керується користувачем за допомогою комп'ютерної миші і клавіатури;
- система має інтуїтивно зрозумілий інтерфейс, який не потребує довідкового матеріалу;



- система представляє собою модуль інтернет магазину, який контролює наявність товарів

Апаратні інтерфейси – не виявлені.

Інтерфейси зв'язку – підтримує HTTP (Hyper Text Transfer Protocol – протокол передачі гіпер – текстових документів) протоколи.

Пам'ять – немає вимог.

Вимоги щодо адаптації місця використання – система адаптована під популярні ОС.

Функції продукту.

Система вмикається переходом по відповідному посиланню. Першим етапом початку роботи з системою є вхід до системи з певним логіном та паролем. Після перевірки правильності логіна та пароля користувачу будуть доступні всі можливості системи.

Характеристики користувача.

Клієнт – це користувач повинен мати базові навички роботи з віконними програмними застосунками.

Обмеження.

Користувача реєструє існуючий адміністратор.

Системою можливо користуватися на таких ОС як Windows або UNIX – сімейства. Обов'язково потрібне підключення до мережі Інтернет.

Специфічні вимоги.

Вимоги до зовнішніх інтерфейсів.

Інтерфейси користувача – користувач може увійти в систему лише під зареєстрованим логіном та паролем.

Апаратні інтерфейси – наявність комп'ютера.

Інтерфейси програмного забезпечення – наявність комп'ютера під керуванням ОС сімейства Windows, UNIX, UNIX – подібні системи.

Інтерфейси зв'язку.

Система повинна за допомогою Інтернет – з'єднання зберігати дані кожного користувача на віддаленому сервері.

Функціональні вимоги.

Клас користувачів – «Адміністратор»:

- Функціональна вимога 1 – Показ товарів при вході в систему;
- Функціональна вимога 2 – Робота з товарами(замовлення, повернення, додавання, зміна статусу доступності);

- Функціональна вимога 3 – Показ інформації про користувачів;

Вимоги до робочих характеристик:

- Кількість підключених до системи користувачів необмежена.
- Велика кількість одночасних користувачів, які роблять операції змін даних у базі даних.
- Підтримка запису та зберігання великої кількості інформації.
- Атрибути якості програмного забезпечення.
- Надійність – обов'язкова авторизація та аутентифікація користувачів.

### **3.3. Моделювання системи**

У багатьох проектах застосовуються методи об'єктно – орієнтованого аналізу, дизайну і розробки. Як правило, в предметної або робочої області об'єкти (objects) співвідносяться з об'єктами реального світу. Вони представляють окремі екземпляри, виведені із загального шаблону, який називається клас (class). В опису класів входять атрибути (дані) і дії, які можна виконувати з цими атрибутами. Діаграма класів (class diagram) – це графічний спосіб відобразити класи, ідентифіковані в ході об'єктно – орієнтованого аналізу, і взаємин між ними.

Стандартною мовою об'єктно – орієнтованого моделювання є Уніфікована мова моделювання (Unified Modeling Language, UML). UML в першу

чергу використовується для створення моделей дизайну. На рівні абстракції, який годиться для аналізу вимог, можуть використовуватися кілька моделей UML.

Діаграми класів, які відображають класи об'єктів предметної області: їх атрибути, поведінку і властивості, а також відносини між класами. Діаграма класів можуть також застосовуватися для моделювання даних, але в такому обмеженому застосуванні не повністю використовуються можливості діаграми класів.

Діаграми варіантів використання – для відображення відносин між діючими особами, що є зовнішніми по відношенню до систем, і варіантів використання, з якими вони взаємодіють.

Діаграми дій – відображає, як взаємодіють різні потоки або як ролі виконують певні дії, або моделюють потоки бізнес – процесів.

Діаграми (або машина) станів – для представлення різних станів, які можуть приймати об'єкти системи або даних, а також дозволених переходів між станами.

При графічному зображенні діаграм рекомендується дотримуватися наступних правил:

- кожна діаграма повинна бути закінченим поданням деякого фрагмента модельованої предметної області;
- представлені на діаграмі сутності моделі повинні бути одного концептуального рівня;
- вся інформація про сутності повинна бути явно представлена на діаграмі;
- діаграми не повинні містити суперечливої інформації;
- діаграми не слід перевантажувати текстовою інформацією;
- кожна діаграма повинна бути самодостатньою для правильної інтерпретації всіх її елементів;
- кількість типів діаграм, необхідних для опису конкретної системи, не є строго фіксованим і визначається розробником;

- моделі системи повинні містити тільки ті елементи, які визначені.

Діаграми прецедентів застосовуються для моделювання виду системи з точки зору прецедентів (або варіантів використання). Найчастіше це передбачає моделювання контексту системи, підсистеми або класу або моделювання вимог, що пред'являються до поведінки зазначених елементів.

Кожна така діаграма показує безліч прецедентів, акторів і відносини між ними.

У мові UML діаграми прецедентів якраз і дозволяють візуалізувати поведінку системи, підсистеми або класу, щоб користувачі могли зрозуміти як їх використовувати, а розробники – реалізувати відповідний елемент.

На діаграмі показані узагальнені варіанти використання програми користувачем, а також адміністратором (рис. 3.1).

При запуску програми користувач повинен увійти в систему, авторизуватися, якщо облікового запису не існує, то створити його.

Після того, як ви увійшли в систему користувачеві надається можливість використовувати функції програми. Серед можливостей використання системи існують такі: перегляд існуючих товарів, додавання товару в кошик, оформлення замовлення.

Серед можливостей адміністратора виділяється можливість перегляду інформації про користувачів, інформації про товари і їх кількість, можливість додавання та видалення товарів.

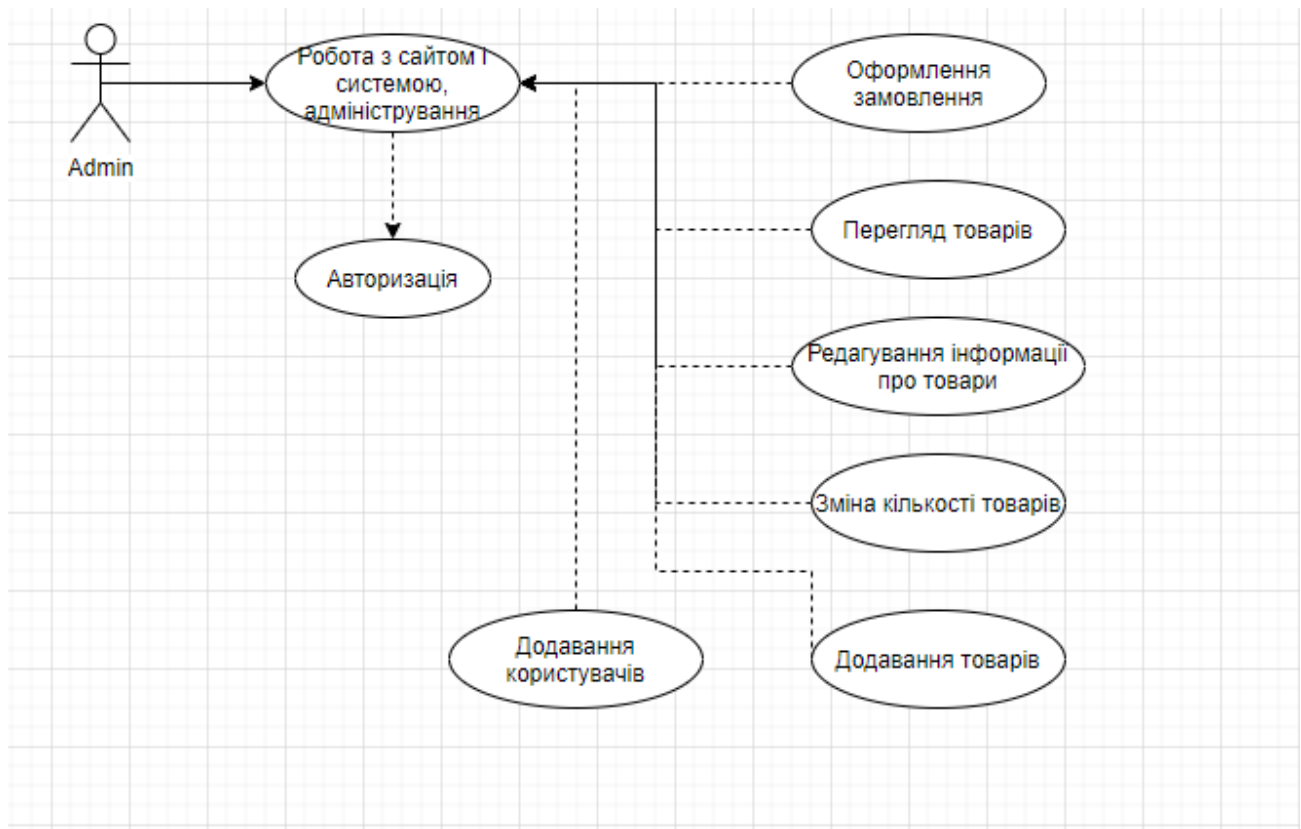


Рис. 3.1. Діаграма прецедентів

Діяльність (activity) (в контексті UML) – є сукупністю окремих обчислень, виконуваних автоматом, що призводять до деякого результату або дії (action).

Діаграма діяльності відображає логіку і послідовність переходів від однієї діяльності до іншої. Результат діяльності – зміна стану системи або повернення деякого значення.

Прецедент П1 - Авторизація користувача.

Головна послідовність: користувач переходить за посиланням та входить до системи за власним логіном та паролем. При успішному виконанні входу до системи надається ключ (token), який надає доступ до наступних даних у системі. Якщо користувач хоче вийти із системи, тоді потрібно натиснути кнопку

«Logout», буде опрацьована ситуація та знищено ключ доступу до системи (рис. 3.2).

Альтернативна невдала послідовність: коли клієнт вирішив увійти у систему, але така пара «логін – пароль» не існує, користувач буде проінформований про цю помилку авторизацію (рис. 3.3).

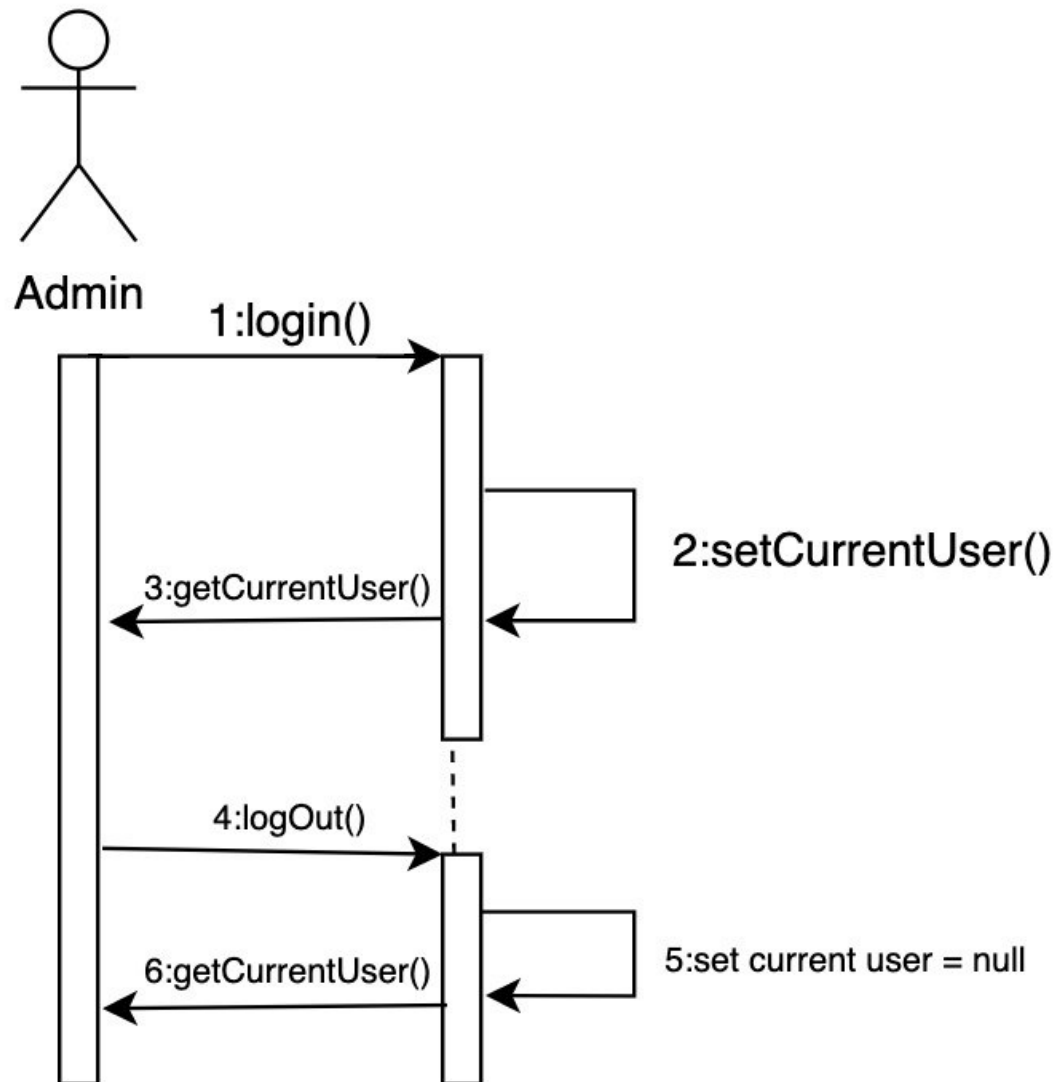


Рис. 3.2. Вхід в систему

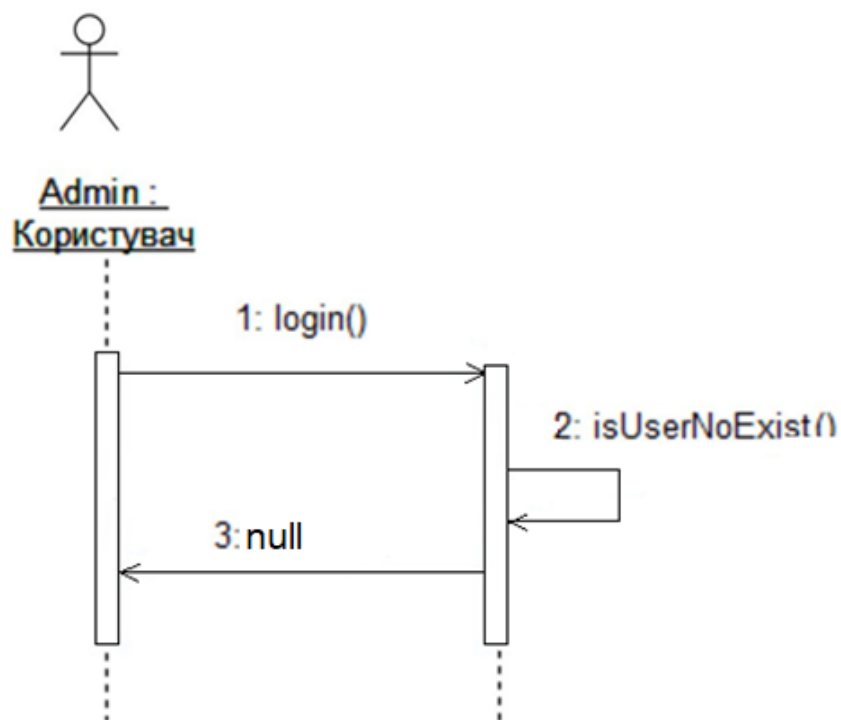


Рис. 3.3. Помилка при авторизації

Прецедент П2 — Діяльність адміністратора

Схема діяльності адміністратора представлена на рис.3.4.

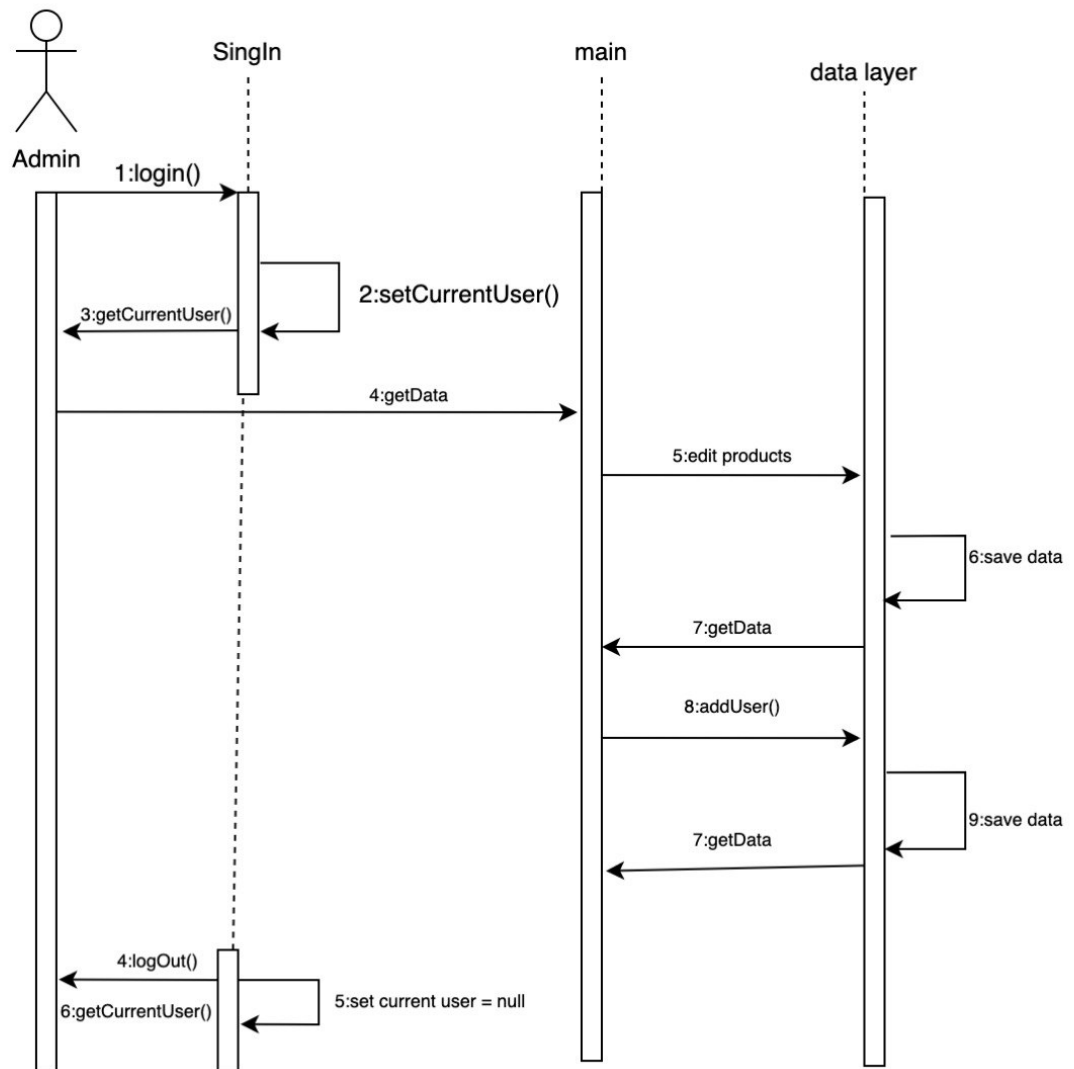


Рис. 3.4. Діяльність адміністратора

Прецедент ПЗ — Реєстрація користувача в системі

Схема реєстрації нового користувача в системі представлена на рис. 3.5.



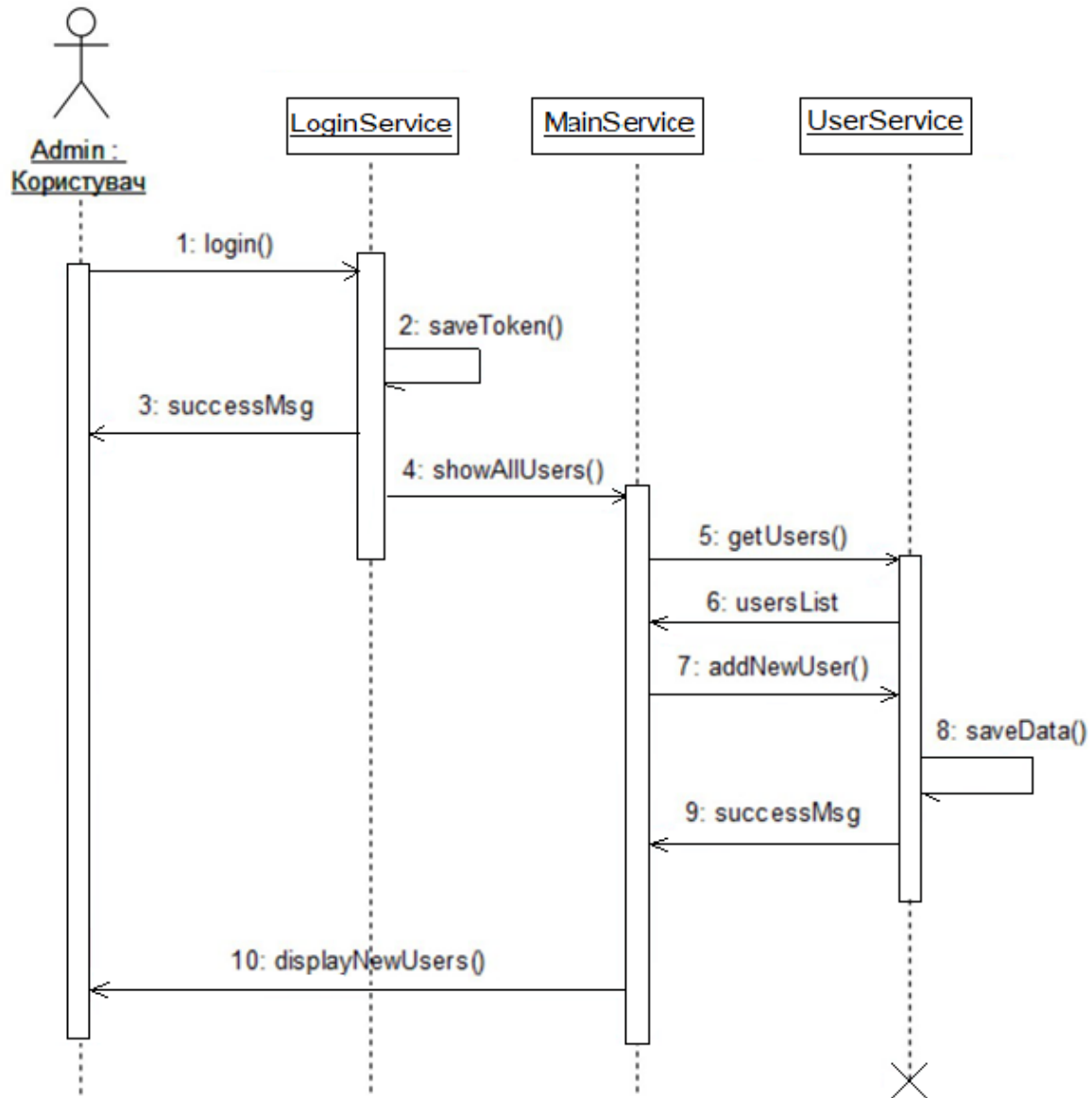


Рис. 3.5. Додавання нового користувача

Прецедент П4 — Додавання нового товару

Схема додавання нового товару представлена на рис. 3.6.

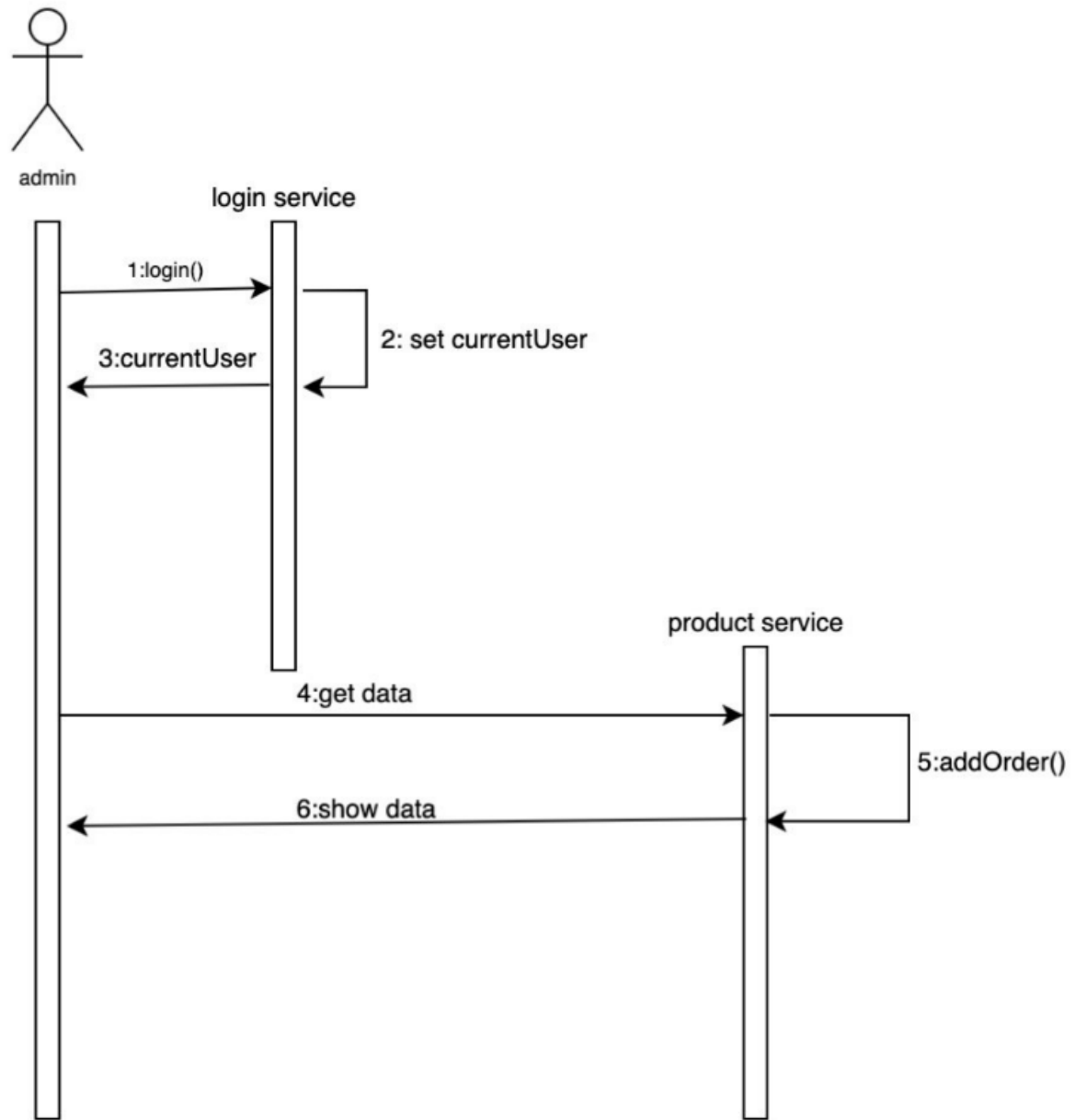


Рис. 3.6. Додавання нового товару

Найбільша перевага діаграм діяльності полягає в тому, що вони підтримують і стимулюють застосування паралельних процесів. Діаграми діяльності є цінним інструментом для подання логіки поведінки систем. Саме завдяки цьому вони представляють собою потужний засіб моделювання потоків робіт.

М. Фаулер, автор книги «UML. Основи», рекомендує будувати діаграми послідовностей не тільки при проектуванні, а й на етапі рефакторінга якщо потрібно замінити централізовану обробку децентралізованої (більш гнучкою), тому що цей вид діаграм є найкращим способом розподілу обов'язків. Також, вони часто використовуються при поясненні будь – яких деталей проекту іншим учасникам.

### **3.4. Розробка графічного інтерфейсу користувача**

Інтерфейс має важливе значення для будь – якої програмної системи і є невід'ємною її складовою, орієнтованої, перш за все, на кінцевого користувача. Саме через інтерфейс користувач судить про прикладну програму в цілому; більш того, часто рішення про використання прикладної програми користувач приймає по тому, наскільки йому зручний і зрозумілий призначений для користувача інтерфейс.

Інтерфейс – сукупність технічних, програмних і методичних (протоколів, правил, угод) засобів сполучення в обчислювальній системі користувачів з пристроями і програмами, а також пристроїв з іншими пристроями і програмами.

Інтерфейс користувача – набір методів взаємодії комп'ютерної програми і користувача цієї програми.

*Сучасні види інтерфейсу.*

Командний інтерфейс. Командний інтерфейс називається так тому, що в цьому виді інтерфейсу людина подає «команди» комп'ютера, а комп'ютер їх виконує і видає результат людині. Командний інтерфейс реалізований у вигляді пакетної технології та технології командного рядка.

WIMP – інтерфейс (Window Image Menu Pointer). Характерною особливістю цього виду інтерфейсу є те, що діалог з користувачем ведеться не за допомогою

команд, а за допомогою графічних образів – меню, вікон, інших елементів. Хоча і в цьому інтерфейсі подаються команди машині, але це робиться «опосередковано», через графічні образи. Цей вид інтерфейсу реалізований на двох рівнях технологій: простий графічний інтерфейс і «чистий» WIMP – інтерфейс.

SILK – інтерфейс (Speech Image Language Knowledge). Цей вид інтерфейсу найбільш наближений до звичайної, людської форми спілкування. В рамках цього інтерфейсу йде звичайний «розмова» людини і комп'ютера. При цьому комп'ютер знаходить для себе команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд він також перетворює в зрозумілу людині форму. Цей вид інтерфейсу найбільш вимогливий до апаратних ресурсів комп'ютера, і тому його застосовують в основному для військових цілей.

Інтерфейс WIMP характеризується різними особливостями.

Вся робота з програмами, файлами і документами відбувається у вікнах – певних окреслених рамкою частинах екрану.

Усі програми, файли, документи, пристрої та інші об'єкти представляються у вигляді значків – іконок. При відкритті іконки перетворюються у вікна.

Всі дії з об'єктами здійснюються за допомогою меню. Хоча меню з'явилося на першому етапі становлення графічного інтерфейсу, воно не мало в ньому головного значення, а служило лише доповненням до командного рядка. У чистому WIMP – інтерфейсі меню стає основним елементом управління.

Широке використання маніпуляторів для вказівки на об'єкти. Маніпулятор перестає бути просто іграшкою – доповненням до клавіатури, а стає основним елементом управління. За допомогою маніпулятора вказує на будь – яку область екрану, вікна або іконки, виділяють її, а вже потім через меню або з використанням інших технологій здійснюють управління ними.

Слід зазначити, що WIMP вимагає для своєї реалізації кольоровий растровий дисплей з високим дозволом і маніпулятор.

Також програми, орієнтовані на цей вид інтерфейсу, пред'являють підвищені вимоги до продуктивності комп'ютера, обсягом його пам'яті, пропускну здатності шини і т.п. Однак цей вид інтерфейсу найбільш простий у засвоєнні і інтуїтивно зрозумілий. Яскравим прикладом програм з графічним таким інтерфейсом є операційна система Microsoft Windows.

Для даної системи розроблено прототипи дизайну віконного типу. На рис. 3.7 зображено вікно входу в систему за паролем та імям користувача, для виконання входу потрібно ввести особистий логін та пароль й натиснути кнопку «SignIn».

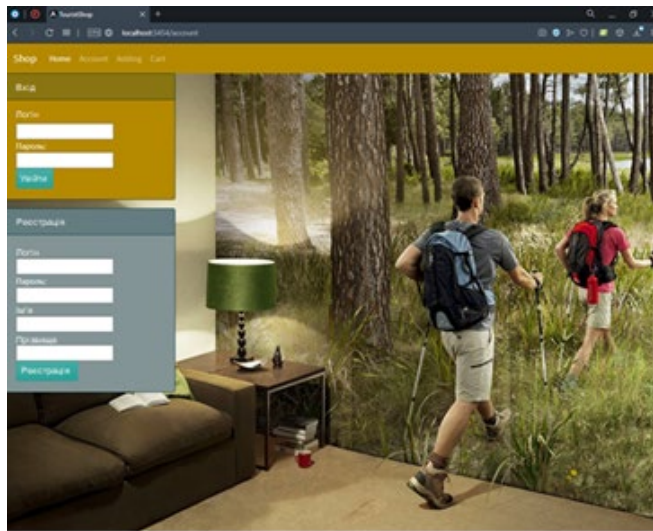


Рис. 3.7. Сторінка авторизації

Якщо логін та пароль правильні і вхід до системи виконано успішний, то користувачу стає доступна головна сторінка системи), на якій розташовані:

- Форма додавання продуктів(рис. 3.8)
- Список всіх продуктів з можливістю зміни кількості, замовлення і доступності(рис. 3.9)

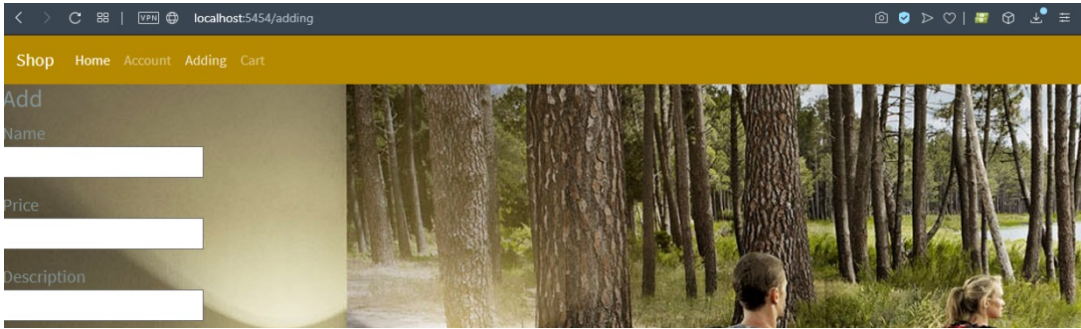


Рис. 3.8. Форма додавання продукту

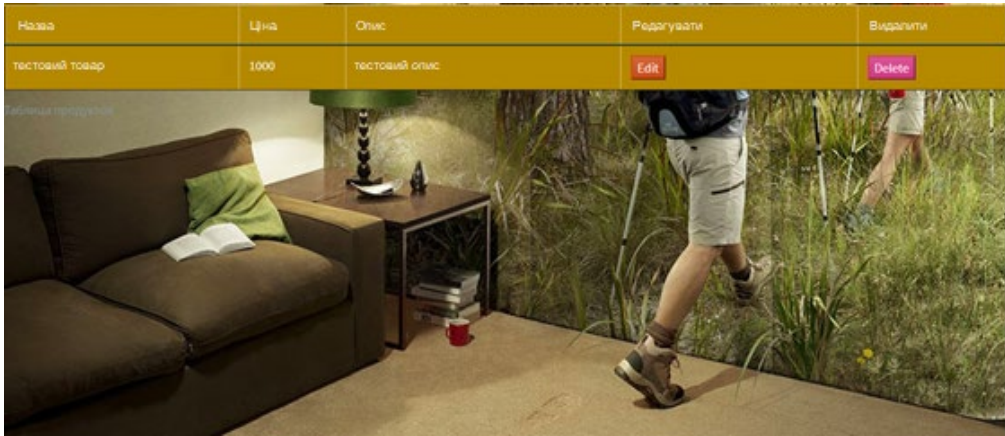


Рис. 3.9. Список усіх продуктів

## РОЗДІЛ 4. РЕАЛІЗАЦІЯ СИСТЕМИ

### 4.1. Схема роботи розробленої системи

Патерн (англ. design pattern) – повторимо архітектурна конструкція, що представляє собою рішення проблеми проектування в рамках деякого часто виникаючого контексту.

Патерн Model – View – Controller (MVC), відкритий у кінці 1970 – х, являє собою шаблон проектування архітектури програмного забезпечення, основним завданням якого є відділення функцій роботи з даними від їх подання [18].

Технологія MVC – це поділ логіки обробки запиту користувача і логіки подання інформації.

До винаходу цього шаблону, логіка обробки запиту користувача і логіка відображення були тісно переплетені в одному файлі. Це дуже ускладнювало підтримку і призводило до безлічі помилок у системі.

Перед розробниками патерну постала задача розробити архітектурне рішення, яке дозволяло б відокремити графічний інтерфейс від бізнес логіки, а бізнес логіку від даних. Таким чином, в класичному варіанті, MVC складається з трьох частин, які і дали йому назву (рис. 4.1).

Модель (Model) використовується для доступу і маніпулювання даними. У більшості випадків модель – це те, що використовується для доступу до сховища даних (наприклад, бази даних). Модель надає інтерфейс для пошуку даних, їх створення, модифікації і видалення зі сховища. У контексті патерну MVC модель є посередником між поданням і контролером.

Вкрай важливою рисою моделі є те, що технічно вона не має ніяких знань ні про те, що відбувається з даними в контролері і поданні. Модель ніколи не повинна робити чи очікувати будь – яких запитів в/з інших компонентів патерну.

В обов'язки представлення (View) входить відображення даних отриманих від Моделі. Однак, уявлення не може безпосередньо впливати на модель. Можна говорити, що уявлення отримує доступ «тільки на читання» до даних.

Представлення володіє наступними ознаками:

- у поданні реалізується відображення даних, які виходять від моделі будь – яким способом;
- у деяких випадках, уявлення може мати код, який реалізує деяку бізнес – логіку.

Контролер (Controller) – це остання частина зв'язки MVC. Завданням контролера є отримання даних від користувача і маніпуляція моделлю. Саме контролер, і тільки він, є тією частиною системи, яка взаємодіє з користувачем.

Контролер пов'язаний з одним поданням і однією моделлю, організовуючи таким чином односпрямований потік даних, контролюючи його на кожному етапі. Контролер починає свою роботу тільки в результаті взаємодії користувача з поданням, яке викликає відповідну функцію контролера.

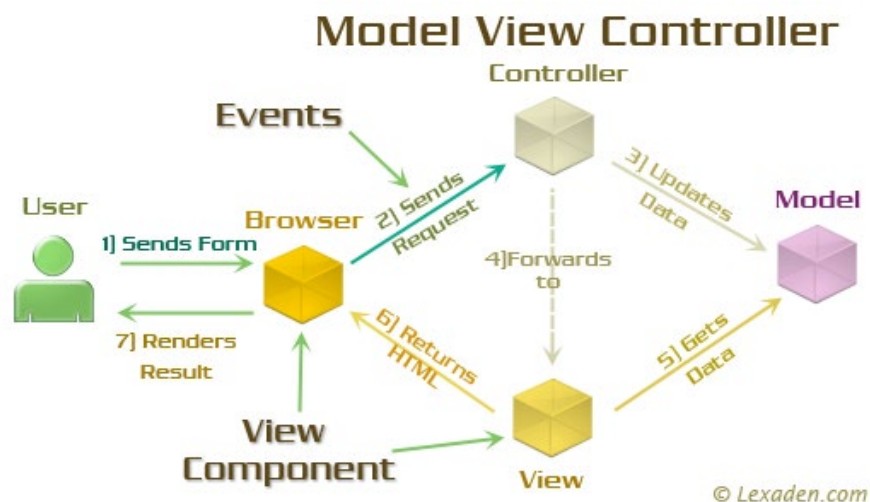


Рисунок 4.1. Схема взаємозв'язків компонентів патерну MVC



## ВИСНОВКИ

В результаті виконаної магістерської роботи розроблено систему контролю та обліку товарів на складі інтернет-магазину, призначену для використання на підприємствах, які потребують більш докладного слідкування за трафіком товарів.

У роботі вирішено наступні задачі:

1. Проведено порівняння існуючих систем інтернет-магазинів за основними параметрами та можливостями, розглянуто технології розробки клієнт-серверних застосунків.

2. Розроблено систему електронної торгівлі, запропоновано вимоги щодо атрибутів, властивостей та якостей системи. Проектування базується на застосуванні діаграм варіантів використання та послідовності взаємодії у системі, розроблено специфікацію вимог до програмного забезпечення (SRS) згідно діючої методики складання специфікацій вимог до програмного забезпечення, що рекомендується Інститутом Інженерів з Електротехніки та Електроніки. Обрано інтерфейс користувача віконного типу та розроблено прототипи дизайну згідно особливостей типу інтерфейсу.

3. Розроблено клієнт-серверний застосунок и на основі Angular та NodeJS.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Willard Legrand Bundy Biography [Електронний ресурс]. – Режим доступу: <http://bundymuseum.org/site3/about/the-history/willard-bundy-bio>. – Назва з екрана.
2. Петрова Ю.А. Фредерик Тейлор // Цикл статей – «Гуру менеджмента» / Ю.А. Петрова, О.С. Красова – Издательство «Ай Пи Эр Медиа», 2008. – 15 с.
3. Time Tracking Software – борцы с пожирателями времени. [Електронний ресурс]. – Режим доступу: [http://ko.com.ua/time\\_tracking\\_software\\_borcy\\_s\\_pozhiratelyami\\_vremeni\\_81133](http://ko.com.ua/time_tracking_software_borcy_s_pozhiratelyami_vremeni_81133). – Назва з екрана.
4. Time Tracker [Електронний ресурс]. – Режим доступу: [http://www.timetracker.biz/timetracker\\_function.html](http://www.timetracker.biz/timetracker_function.html). – Назва з екрана.
5. prima ERP | Блог. [Електронний ресурс]. – Режим доступу: [http://blog.ru.primaerp.com/2015/08/blog-post\\_26.html](http://blog.ru.primaerp.com/2015/08/blog-post_26.html). – Назва з екрана.
6. Лучшие MacOS программы для отслеживания рабочего времени для фрилансера [Електронний ресурс] – Режим доступу: <http://www.coolwebmasters.com/project-management/3313-time-tracking-mac-apps.html>. – Назва з екрана.
7. Трекеры времени для программиста [Електронний ресурс] – Режим доступу: <http://frey.su/programmer-time-tracker>. – Назва з екрана.
8. Тайм трекеры [Електронний ресурс] – Режим доступу: [http://www.onlineprojects.ru/tags/tajm\\_trekery](http://www.onlineprojects.ru/tags/tajm_trekery). – Назва з екрана.
9. 5 трекеров времени, которые помогут победить прокрастинацию [Електронний ресурс]. – Режим доступу: <https://lifel hacker.ru/2016/11/17/trekery-vremeni>. – Назва з екрана.
10. Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво" - Луцьк, 2013. Випуск №11

11. Electron [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Electron>. – Назва з екрана.
12. Требования к программным продуктам. [Электронный ресурс]. – Режим доступа: <http://www.dpgrup.ru/software-requirements.htm>. – Назва з екрана.
13. Non-functional requirement. [Электронный ресурс]. – Режим доступа: [https://en.wikipedia.org/wiki/Non-functional\\_requirement](https://en.wikipedia.org/wiki/Non-functional_requirement). – Назва з екрана.
14. Стандарт IEEE 830-1998. Методика составления спецификаций требований к программному обеспечению.
15. Буч Г. Язык UML. Руководство пользователя. – [2 – е изд] / Г. Буч, Д. Рамбо, И. Якобсон; пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.: ил. – ISBN 5 – 94074 – 334 – Х.
16. Фаулер М. UML. Основы, 3е издание. / М. Фаулер – Пер. с англ. – СПб: Символ\_Плюс, 2004. – 192 с.,ил.
17. Методы и средства разработки пользовательского интерфейса: современное состояние, Клещев А.С. , Грибова В.В. , 2001 [Электронный ресурс] □ Режим доступа <http://www.swsys.ru/index.php?page=article&id=765>. – Назва з екрана.
18. MVC pattern [Электронный ресурс] – Режим доступа: <http://ashep.org/2013/mvc-pattern-php-1/#.WQe3hNLyjtQ>. – Назва з екрана
19. Brown Ethan. Web Development with Node and Express. / Ethan Brown, US: O'Reilly Media, 2014 – ISBN 978-1-4919-4928-3.
20. Туловский А. Руководство разработчика Angular. / А. Туловский Электронная версия, 2013. – 106 с.
21. Програмні системи створення веб-сайтів, CMS – режим доступа: <http://www.znannya.org/?view=WebDev> – назва з екрану
22. OLX - Вікіпедія – режим доступа: <https://uk.wikipedia.org/wiki/OLX> - назва з екрану

23. Rozetka.ua – Вікіпедія – режим доступу:  
<https://uk.wikipedia.org/wiki/Rozetka.ua> - назва з екрану
24. Тузовський А.Ф. Проектування і розробка web додатків 2007 – режим доступу:  
[https://stud.com.ua/97571/informatika/proektuvannya\\_i\\_rozrobka\\_web-dodatkov](https://stud.com.ua/97571/informatika/proektuvannya_i_rozrobka_web-dodatkov) - назва з екрану
25. Доросинский Л.Р. Розробка інтернет додатків 2018 – режим доступу:  
[https://stud.com.ua/121236/informatika/rozrobka\\_internet-dodatkov](https://stud.com.ua/121236/informatika/rozrobka_internet-dodatkov) - назва з екрану
26. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. В. Дронов – режим доступу:  
[https://codernet.ru/books/css/html\\_5\\_css\\_3\\_i\\_web\\_2\\_razrabotka\\_sovremennykh\\_web\\_sajtov\\_dronov/](https://codernet.ru/books/css/html_5_css_3_i_web_2_razrabotka_sovremennykh_web_sajtov_dronov/) - назва з екрану
27. AngularJS / Вікіпедія [Електронний ресурс]. – Режим доступу:  
<https://uk.wikipedia.org/wiki/AngularJS> - назва з екрану
28. Офіційний сайт AngularJS [Електронний ресурс] - Режим доступу:  
<https://angularjs.org/> - назва з екрану
29. Angular и TypeScript. Сайтостроение для профессионалов. — СПб.: Питер, 2018. — 464 с.: ил. — (Серия «Библиотека программиста»). [Електронний ресурс] – Режим доступу: <http://sd.blackball.lv/books/17369?mode=read> – назва з екрану
30. Эффективный TypeScript: 62 способа улучшить код. — СПб.: Питер, 2020. — 288 с.: ил. — (Серия «Бестселлеры O'Reilly»). [Електронний ресурс] – Режим доступу: <http://sd.blackball.lv/books/17635?mode=read> – назва з екрану

## ДОДАТКИ

### Додаток 1. Main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
if (environment.production) {
  enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

### Додаток 2. app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core'
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

### Додаток 3. app-routing.module.ts

```
import { NgModule } from '@angular/core';
```

```

import { BrowserModule } from '@angular/platform-browser';
import { Routes, RouterModule } from '@angular/router';
import { AccountComponent } from './account.component/account.component';
import { AddingComponent } from './adding.component/adding.component';
import { HomeComponent } from './home.component/home.component';
import { NotFoundComponent } from './not-found.component/not-found.component';
import { ShopComponent } from './shop.component/shop.component';
const routes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'account', component: AccountComponent},
  { path: 'shop', component: ShopComponent},
  { path: 'adding', component: AddingComponent},
  //{ path: 'cart', component: CartComponent},
  { path: '**', component: NotFoundComponent },
];
@NgModule({
  declarations: [
    HomeComponent,
    NotFoundComponent,
    ShopComponent,
    AddingComponent,
    //CartComponent,
    AccountComponent,
  ],
  imports: [ BrowserModule, RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

#### Додаток 4. app.component.ts

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',

```

```

    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
  })
  export class AppComponent {
    title = 'TouristShop';
  }

```

## Додаток 5. app-component.html

```

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <a class="navbar-brand" routerLink="/">Shop</a>
  <button routerLink="/" class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarColor01">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" routerLink="/">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/account">Account</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/adding">Adding</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/cart">Cart</a>
      </li>
    </ul>
  </div>
</nav>
<router-outlet ></router-outlet>

```

**Додаток 6. account.component.ts**

```

import { Component, OnInit } from '@angular/core';
import { userModel } from '../models/user.model';
import { writeService } from '../services/writter.service';
@Component({
  selector: 'account-app',
  templateUrl: './account.component.html',
})
export class AccountComponent implements OnInit {
  admin: userModel;
  currentUser: userModel;
  users: userModel [];
  errorOpened: boolean = false;
  private _writeService : writeService;
  ngOnInit(): void {
    this._writeService = new writeService();
    this._initUsers();
  }
  public registration(username: string, password: string, firstName:string, secondName:string){
    if(!username){return;}
    if(!password){return;}
    let user: userModel = new userModel();
    user.username = username;
    user.password = password;
    user.firstName = firstName;
    user.secondName = secondName;
    this._addUser(user);
    this.currentUser = user;
    this._writeService.setCurrentUser(this.currentUser);
  }
  public login(username: string, password:string){
    let notFound = true;
    this.users.forEach(element => {

```



```

        if(element.username==username){
            if(element.password == password){
                this.currentUser = element;
                this._writeService.setCurrentUser(this.currentUser);
                notFound= false;
                return;
            }
        }
    });
    if(notFound){this.errorOpened = true;}
}

public closeError(){
    this.errorOpened = false;
}

public logout(){
    this.currentUser = null;
    this._writeService.setCurrentUser(this.currentUser);
}

private _addUser(user: userModel){
    this.users.push(user);
    this._writeService.setList(this.users, "users");
}

private _initUsers(){
    if(this._writeService.getCurrentUser() != null)
    {
        this.currentUser = this._writeService.getCurrentUser();
    }
    if(!this._writeService.getList("users")){
        this.users = [];
        this.users.push(this._initAdmin());
        this._writeService.setList(this.users, "users");
    }
    this.users = this._writeService.getList("users");
}

```

```

    }
    private _initAdmin(){
        this.admin = new userModel();
        this.admin.username = "admin123";
        this.admin.password = "qwerty";
        this.admin.firstName="admin";
        this.admin.secondName = "Main";
        this.admin.isAdmin = true;
        return this.admin;
    }
}

```

## Додаток 7. adding.component.ts

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { Routes, RouterModule } from '@angular/router';
import { AccountComponent } from './account.component/account.component';
import { AddingComponent } from './adding.component/adding.component';
import { HomeComponent } from './home.component/home.component';
import { NotFoundComponent } from './not-found.component/not-found.component';
import { ShopComponent } from './shop.component/shop.component';
const routes: Routes = [
    { path: '', component: HomeComponent},
    { path: 'account', component: AccountComponent},
    { path: 'shop', component: ShopComponent},
    { path: 'adding', component: AddingComponent},
    //{ path: 'cart', component: CartComponent},
    { path: '**', component: NotFoundComponent },
];
@NgModule({
    declarations: [
        HomeComponent,
        NotFoundComponent,

```

```

    ShopComponent,
    AddingComponent,
    //CartComponent,
    AccountComponent,
  ],
  imports: [ BrowserModule, RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

## Додаток 8.

home.component.ts

```

import { Component } from '@angular/core';
@Component({
  selector: 'home-app',
  templateUrl: './home.component.html',
})
export class HomeComponent {
}

```

home.component.html

```

<h3>Главная</h3>
<br/>
<div>
<p>
  <a routerLink="/shop">
    <button type="button" class="btn btn-success">
      Магазин
    </button>
  </a>
</p>
<p><a routerLink="/account"> <button type="button" class="btn btn-success">Личный
кабинет</button></a></p>

```

```

    <p><a routerLink="/adding"><button type="button" class="btn btn-success" >Добавление
товара</button></a></p>
    <p><a routerLink="/cart"><button type="button" class="btn btn-success" >Корзина</button></a></p>
  </div>

```

### Додаток 9. shop.component.ts

```

import { Component, OnInit } from '@angular/core';
import { userModel } from '../models/user.model';
import { productModel } from '../models/product.model';
import { writeService } from '../services/writter.service';
@Component({
  selector: 'shop-app',
  templateUrl: './shop.component.html',
})
export class ShopComponent implements OnInit {
  currentUser : userModel;
  products: productModel [];
  _writeService: writeService;
  cart: productModel [];
  ngOnInit(): void {
    this._writeService = new writeService();
    this._init();
  }
  public buy(item: productModel){
    this.cart.push(item);
    this._writeService.setList(this.cart, "cart");
  }
  private _init(){
    if(this._writeService.getCurrentUser()){
      this.currentUser = this._writeService.getCurrentUser();
    }
    if(!this._writeService.getList("products")){
      this.products = [];
    }
  }

```

```

    for (let index = 0; index < 5; index++) {
        this.products.push(this._initBaseProducts(index));
    }
    this._writeService.setList(this.products, "products");
}
this.products = this._writeService.getList("products");
if(!this._writeService.getList("cart")){
    this.cart = [];
    this._writeService.setList(this.cart, "cart");
}
this.cart = this._writeService.getList("cart");
}

```

```

private _initBaseProducts(index : number){
    let product = new productModel();
    product.name = "Product " + index;
    product.price = 20 + 10* index;
    product.description = "this is an initial product";
    return product;
}
}

```

## Додаток 10.

not-found.component.ts

```

import { Component } from '@angular/core';
@Component({
    selector: 'not-found-app',
    templateUrl: './not-found.component.html',
})
export class NotFoundComponent { }

```

not-found.component.html

```
<div class="alert alert-dismissible alert-danger">  
  <button type="button" class="close" data-dismiss="alert">&times;</button>  
  <strong>Упс!</strong> Страница не найдена. Вы можете вернуться на <a href="#" class="alert-  
link">главную</a> страницу.  
</div>
```