Київський національний торговельно-економічний університет

Кафедра цифрової економіки та системного аналізу

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Веб система управління підприємством електронної комерції»

Студента 2м курсу, 1 групи,
051 Економіка,
цифрова економіка

Кравченка Івана
Володировича

підпис студента

Науковий керівник:
кандидат економічних наук,
доцент кафедри цифрової економіки та системного аналізу

Кулаженко Володимир
Валерійович

підпис керівника

Гарант освітньої програми:
доктор фізико-математичних наук, професор кафедри цифрової економіки та системного аналізу

Гамалій Володимир
Федорович

підпис гаранта

Київ 2020

Київський національний торговельно-економічний університет

Факультет обліку, аудиту та інформаційних систем
Кафедра цифрової економіки та системного аналізу
Освітній ступінь магістр
Спеціальність 051 «Економіка»
Спеціалізація «Цифрова економіка»

Затверджую
Зав. кафедри _____Роскладка А. А.
«15» січня 2020р.

Завдання
на випускну кваліфікаційну роботу (проект) студентці

Кравченку Івану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Веб система управління підприємством електронної комерції»
Затверджена наказом ректора від *«02» грудня 2019 р. № 4145*
2. Строк здачі студентом закінченої роботи *05 листопада 2020 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: вивчення бізнес-процесів та особливостей розробки веб-системи управління підприємством електронної комерції на основі сучасних технологій веб-програмування.
Об'єкт дослідження: веб-система управління підприємством електронної комерції.
Предмет дослідження: процес моделювання та створення веб-системи управління підприємством електронної комерції.
4. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Кулаженко В. В.	*15.01.2020*	*15.01.2020*
2	Кулаженко В. В.	*15.01.2020*	*15.01.2020*
3	Кулаженко В. В.	*15.01.2020*	*15.01.2020*

5. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Тенденції розвитку електронної комерції

1.2 Принципи функціонування електронної комерції

1.3 Аналіз бізнес-процесів підприємства електронної комерції

Висновки до розділу 1

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

2.1 Створення операційної-моделі підприємства

2.2 Стан та проблеми управління підприємством електронної комерції

2.3 Проектування веб-системи управління підприємством

Висновки до розділу 2

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

3.1 Інструменти та методи розробки веб-системи управління

3.2 Програмна реалізація веб-системи

3.3 Аналіз результатів розробки веб-системи управління підприємством електронної комерції

Висновки до розділу 3

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.12.2019*	*01.12.2019*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.01.2020*	*15.01.2020*
3	*Вступ*	*01.06.2020*	
4	*Розділ 1. Теоретичні основи управління підприємством електронної комерції*	*25.06.2020*	
5	*Розділ 2. Моделювання та аналіз управління підприємством електронної комерції*	*02.09.2020*	

6	*Підготовка статті у збірник наукових статей магістрів*	*07.09.2020*	
7	*Розділ 3. Реалізація системи управління підприємством електронної комерції*	*19.10.2020*	
8	*Висновки*	*02.11.2020*	
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*05.11.2020*	
10	*Попередній захист випускної кваліфікаційної роботи*	*20.11.2020*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*23.11.2020*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*25.11.2020*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

7. Дата видачі завдання «15» січня 2020 р.

8. Керівник випускної кваліфікаційної роботи (проекту) Кулаженко В. В.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Гамалій В. Ф.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент-дипломник Кравченко. І. В.

(прізвище, ініціали, підпис)

АНОТАЦІЯ

У магістерському дослідженні уточнено зміст понять аналіз бізнес-процесів, інформаційна система. Розглянуто основні етапи розробки інформаційної-системи управління підприємством електронної комерції. Вивчено бізнес-процеси підприємства, і на основі цього створено модель бізнес-процесів. Розроблено модель веб-системи та спроектовано логічну модель бази даних. Визначено методику та інструметарій створення веб-системи управління. Розроблено систему-управління на основі сучасних технологій веб-програмування: JavaScript (React.js, Node.js, Express.js), MongoDB, HTML та CSS. Оцінено економічний ефект від запровадження веб-системи управління на підприємтсві електронної комерції.

Ключові слова: інформаційна модель, інформаційна система, бізнес-процеси, аналіз бізнес-процесів, електронна комерція, економічний вплив, моделювання бізнес-процесів, підприємство, веб-додаток, СУБД, BPD, BPMN, веб-технології, веб-сайт, мови програмування.

ABSTRACT

The master`s research specified sense of terms business-process analysis, information system. It was considered the main development stages of a management information system of an e-commerce enterprise. In the work had studied business-processes of an enterprise and then created a business process model. It was developed the web-system-model and designed the database logic model. The method and instruments of creating the management web-system were determined. The work shows developed the management system based on modern web-programming technologies: JavaScript (React.js, Node.js, Express.js), MongoDB, HTML та CSS. The research assessed an economic influence from introducing the management web-system in an e-commerce enterprise.

Keywords: information model, information system, business-processes, business-process analysis, e-commerce, economic influence, business-process modeling, enterprise, web-application, DBMS, BPD, BPMN, web technologies, web site, programming languages.

ЗМІСТ

ВСТУП

У час стрімкого розвитку електронної комерції, зростає кількість інтернет-магазинів та можливостей для торгівлі. Постає нагальна необхідність у розробці системи управління через використання інформаційних систем та бізнес-моделювання діяльності економічних об’єктів.

Інформаційна система дозволяє визначити логічні та господарчі зв’язки у такому економічному об’єкті, як інтернет-магазин. Так, у цій роботі буде представлено як за допомогою сучасних технологій програмування можна створити автоматизовану систему для збору та обробки інформації для такого об’єкта як підприємство електронної комерції.

Актуальність роботи полягає у необхідності підвищення ефективності управління підприємством та зменшення навантаження на працівників. Так, інформаційна система управління інтернет-магазину реалізує автоматизований збір даних, їхню обробку та зберігання у базі даних. Кількість підприємств електронної комерції швидко зростає, виникає нагальна потреба спростити роботу управлінцям з метою досягнення економічного ефекту через створення відповідного програмного забезпечення, яке буде реалізовувати інформаційну систему управління. Для реалізації подібної системи на сьогоднішній день найкращим засобом є застосування веб-технологій на противагу настільним системам у зв’язку з їхньою доступністю та універсальністю.

Розкриттю даної проблематики присвячені праці українських дослідників, зокрема: Пурський О.І.[15], Сосновська О.О.[18], Свидрук І.І.[16], Георгіаді Н.Г[9], Плескач В.Л.[14] Цікавими є напрацювання й досвід зарубіжних учених Айнін С.[24], Блок М., Ар'є С.[27] та інших.

Мета даного магістерського дослідження полягає у вивченні бізнес-процесів та особливостей розробки веб-системи управління підприємством електронної комерції на основі сучасних технологій веб-програмування.

Об’єктом дослідження є веб-система управління підприємством електронної комерції.

Предметом дослідження є процес моделювання та створення веб-системи управління підприємством електронної комерції.

Методи застосовані у процесі дослідження:

1. Метод порівняння
2. Аналітичний метод
3. Метод абстракції
4. Метод експертних оцінок
5. Метод моделювання
6. Бізнес-моделювання та бізнес-аналіз
7. Метод діджиталізації інформації

Відповідно до зазначеної мети виникає необхідність вирішення таких **завдань**:

- Розглянути сутність поняття, принципи та сучасні тенденції розвитку електронної комерції.
- Здійснити бізнес-аналіз та бізнес-моделювання процесів підприємства електронної комерції.
- Зрозуміти особливості проектування баз даних на основі схеми інформаційної системи.
- Визначити інструменти та методи розробки веб-системи.
- Створити веб-систему управління на основі JavaScript, HTML та CSS.
- Проаналізувати вплив від впровадження розробленої веб-системи на стан підприємства електронної комерції.

Методологічною основою праці є технічна документація з мов програмування та фреймворків, правила бізнес-моделювання.

Практичне значення роботи полягає у розробці систем управління за допомогою веб-технологій для ефективного управління економічним об’єктом. Електронна комерція надає споживачам багато переваг у вигляді наявності товарів за меншими цінами, більш широкого їх вибору та економить час.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.Тенденції розвитку електронної комерції.

Світ електронної комерції стає все більш конкурентоспроможним. Щоб випереджати конкурентів, потрібно постійно проводити моніторинг тенденцій електронної комерції. Незалежно від зрілості магазину на даний момент, при ігноруванні існуючих тенденцій в електронній комерції, підвищується ризик відставання від конкурентів. Потрібно продовжувати дивитися вперед, щоб забезпечити майбутній успіх. На межі 2020-2021 рр., необхідно знати ці тенденції, зокрема прогресивні, щоб ними скористатися. Ось чому так важливо, щоб тенденції в електронній комерції були своєчасно проаналізовані та сприйняті. Роблячи це, можна розвивати свій бренд електронної комерції і випереджати конкурентів.

Водночас не існує стандартного визначення терміна «e-commerce», і різні організації визначали і тлумачили його по-різному. Під електронною комерцією розуміють «виробництво, розповсюдження, маркетинг, продаж чи доставку товарів та послуг електронними засобами» [24]. Азіатсько-Тихоокеанське економічне співробітництво («АТЕС») прийняло грунтовне і широке визначення електронної комерції, щоб включати всю господарську діяльність, що здійснюється за допомогою комбінації електронних комунікацій та технологій обробки інформації [25]. Економічна та соціальна комісія Організації Об'єднаних Націй для Азії та Тихого океану («ЮНЕСКАП») також дає визначення поняттю «електронна комерція» як «процесу використання електронних методів та процедур для ведення всіх форм ділової діяльності» [27].

Якщо поглянути на основні сегменти «e-commerce», хоча трансакції B2B відіграють важливу роль на ринку електронної комерції, частка доходів від електронної комерції у країнах, що швидко розвиваються, генерується B2C. Електронна комерція надає споживачам багато переваг у вигляді наявності товарів за меншими цінами, більш широкого їх вибору та економить час. Люди можуть купувати товари одним клацанням кнопки миші, не виходячи зі свого будинку чи офісу. Аналогічно онлайн-сервіси, такі як: банківська діяльність, придбання

квитків (включаючи авіакомпанії, автобуси, залізниці), оплата рахунків, бронювання готелів тощо, мають величезну користь для клієнтів. Інтернет-бізнес, зокрема фінансові послуги, подорожі, розваги та купівля продуктів, швидше за все, зростатиме. Електронна комерція розвивалася різними формами взаємовідносин у межах бізнес-процесів. Це могло бути у вигляді електронної реклами, електронної платіжної системи, електронного маркетингу, електронної служби підтримки клієнтів та електронного замовлення та доставки.

Серед бізнес-спільноти зростає обізнаність про можливості, які пропонує електронна комерція. Легкість доступу до мережі Інтернету та навігації - це найважливіші фактори, які призведуть до швидкого сприйняття Інтернет-торгівлі. Безпечні та захищені способи оплати є також важливими у зв'язку з необхідністю винаходити та популяризувати інновації, такі як мобільна комерція. Електронна комерція - це новий майданчик для зв'язку зі споживачами та проведення транзакцій. Віртуальні магазини працюють цілодобово, 7 днів на тиждень. Багато віртуальних роздрібних торговців представляють одну компанію, а інші представляють консорціум компаній.

Мар'ям Мохсін виокремила 10 трендів електронної комерції на найближчі роки[22]:

1. Зростання онлайн-продажів буде невпинним.

Продажі онлайн постійно зростають і з поважних причин. Інтернет-покупки - це одна з найпопулярніших онлайн-активностей. За прогнозами дослідниці, продажі збільшаться з 1,3 трлн у 2014 році до 4,5 трлн у 2021 році. Це величезна цифра. Це означатиме триразове зростання протягом 7-річного періоду. Див. Рис 1.1.

Рис 1.1. Зростання онлайн продажів[22]

Зі зростанням популярності магазинів електронної комерції все більше людей звертається до інтернет-магазинів. Таке збільшення кількості інтернет-магазинів можна пояснити низкою факторів. Один з головних - це, ймовірно, рівень комфорту, який надається покупцям у мережі Інтернет. Також спостерігається зростання довіри покупців до продавців в Інтернеті під час купівлі товарів, а також через покращений UX (user experience) роботи на веб-сайті.

2. Екологічна тематика впливає на покупців

Зелене споживання зростає, і брендам потрібно вживати заходів. Половина цифрових споживачів кажуть, що екологічні проблеми впливають на їхні рішення щодо придбання товарів і послуг. Підприємства електронної комерції повинні прагнути до створення більш стійких практик. У наш час люди стають більш свідомими у ставленні до довкілля, ніж будь-коли раніше, і це цілком правильно. Інтернет-бізнесу потрібно займатися цими питаннями більше та переконатися, що їхня практика є екологічною.

3. Зростання продажів із мобільних пристроїв

Помітним є зростання мобільної торгівлі. З 2016 року продажі, здійснені через мобільні пристрої, зросли на 15%. До кінця 2021 року 73% продажів електронної комерції відбуватиметься на мобільному пристрої. Ці цифри не можна ігнорувати. Поліпшення досвіду електронної комерції для мобільних клієнтів може стати величезною можливістю для бізнесу.Див. Рис.1.2.

Рис 1.2. Зростання мобільної торгівлі[22]

4. Зростання голосової комерції

Голосові покупки зростають серед користувачів мережі Інтернет. Так, у 2017 році 13% власників розумних динаміків у США здійснювали покупки голосом. За прогнозами їхня кількість до 2022 року зросте до 55%. Загальна сума витрат на голосові покупки також зросла й у Великобританії. Ця тенденція електронної комерції стала популярною головним чином з 2014 року, коли Amazon представила свій смарт-спікер - Echo. Хоча голосовий шопінг ще знаходиться на ранній стадії розвитку, ця статистика демонструє нам, що він стане найбільш популярним у найближчі роки.

5. Зростаюча роль соціальних медіа

Кількість покупців із соціальних мереж також швидко збільшується. З впровадженням кнопки «Купити» у Facebook та Instagram Checkout соціальні медіа відіграють значну роль у сфері електронної комерції.

Соціальні медіа змінили спосіб нашого повсякденного життя, включаючи й те, як ми купуємо товари. Це прекрасна можливість для брендів почати думати про те, як поліпшити своє становище в соціальних медіа, що є чудовою платформою для того, щоб бренди могли бути розкриті. Оскільки споживачі витрачають більше часу на різні типи соціальних медіа, компанії з електронної комерції можуть

отримати допомогу від Instagram-менеджерів, щоб збільшити шанси на виявлення їхньої цільової аудиторії.

6. Повернення QR-кодів

Ще одна послуга, яка зростає разом із електронною комерцією, - це мобільні платежі, які стали нормою. Дослідження переконують, що понад третина населення в мережі Інтернет використовувала мобільні платежі у 2018 році.

І, схоже, один конкретний спосіб оплати, QR-коди, повертається. Вперше винайдені у 1994 році і лише на початку 2010-х QR-коди стали популярними. Але проблеми, пов'язані з пошкодженням кодів, означали, що їхня популярність триватиме недовго.

За останні кілька років QR-код, здається, відновлює свою популярність, особливо в галузі електронної комерції. Згідно з дослідженням GlobalWebIndex, 40 відсотків усіх користувачів Інтернету сканують QR-код на своїх мобільних телефонах щомісяця (GlobalWebIndex, 2019). Існує і зворотна кореляція за віком - з підлітками та молодими людьми, які використовують QR-коди частіше, ніж ті, хто є у старших вікових групах.

7. Роль штучного інтелекту

Світові витрати роздрібної торгівлі на штучний інтелект (AI) до 2022 року досягнуть 7,3 мільярда доларів на рік, що перевищує прогнозовані 2 мільярди доларів у 2018 році. Це трапляється, коли у роздрібній торгівлі орієнтуються на нові засоби, щоб підвищити персоналізацію для споживачів. Це дослідження підтверджує нам, що роздрібні продавці готові вкладати значні кошти в інструменти, які допоможуть їм покращити свої послуги для клієнтів та нададуть їм конкурентну перевагу.

Ці інструменти AI варіюються від автоматизованих маркетингових платформ, які оснащені для створення своєчасних пропозицій, до чатів, які миттєво відповідають на запити клієнтів. Інші сфери, де AI буде корисним для роздрібної торгівлі, включають оптимізовані AI ціни та дисконтування, а також прогнозування попиту. Див. Рис.1.3.

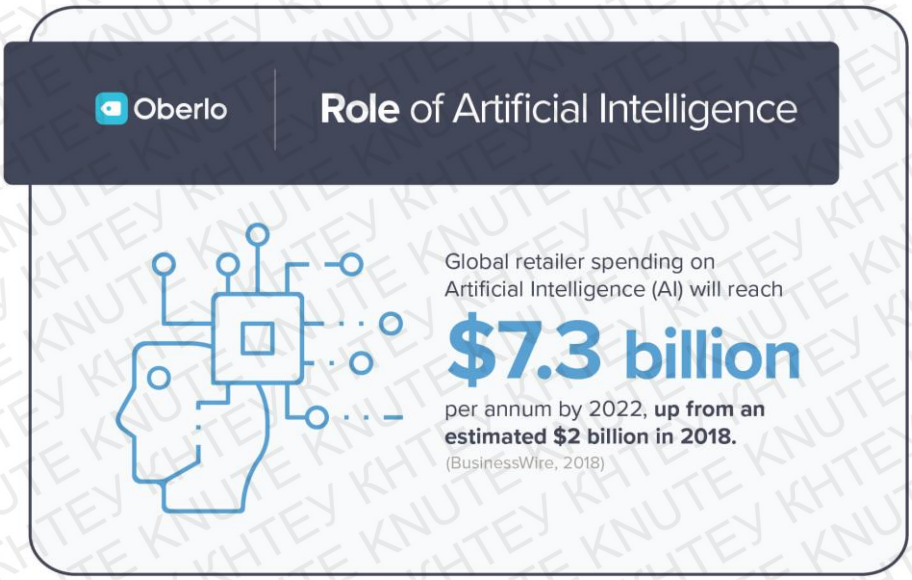


Рис 1.3. Зростання штучного інтелекту[22]

8. Доповнена реальність трансформує, як ми купуємо

До 2022 року понад 120 000 магазинів будуть використовувати технології Augmented Reality (AR), пропонуючи більш багатий досвід покупки. Поглинання AR у цьому секторі здійснюватиметься робочою силою в роздрібній торгівлі та інтернет-покупцями. Одна з головних проблем, яка виникає у людей при закупівлі в Інтернеті, - неможливість побачити товар з перших рук. Технологія AR допомагає подолати цей розрив та надає змогу інтернет-покупцям краще візуалізувати товари, що їх цікавлять.

9. Персоналізація – це майбутнє

Понад 50% покупців стверджують, що для них є важливим персоналізований досвід в Інтернеті. Крім цього, 74% маркетологів вважають, що персоналізація має «сильний» або «екстремальний» вплив на просування і поглиблення відносин із клієнтами.

Персоналізація досвіду покупок в Інтернеті - це ключ до задоволення вимог клієнтів. Люди, які здійснюють купівлю в Інтернеті, хочуть допомоги у тому, щоб знайти потрібні їм товари, вони цінують більш персоналізований досвід. Внаслідок поганих рекомендацій щодо товарів, інтернет-покупці можуть взагалі уникати певних магазинів.

10. Візуальна комерція на підйомі

Серед труднощів в управлінні магазином електронної комерції визначають необхідність продавати свій товар споживачам, які не мають шансів на фізичну взаємодію з запропонованим товаром. Тут вступає в дію візуальна комерція.

Іншими словами, візуальна комерція - це наступне покоління звичайних статичних зображень. Це вимагає маркетингу на цілком іншому рівні, оскільки замість того, щоб просто використовувати світлини або зображення товарів для рекламування бізнесу, візуальна комерція робить крок вперед, включаючи інші типи візуальних засобів. Зокрема такі, як засоби масової інформації, що створюються самими споживачами - інтерактивний зміст із залученням відео, що, як уже згадувалося раніше, створює доповнену реальність. Див. Рис.1.4.

Рис 1.4. Зростання ринку розпізнавання зображень[22]

Візуальна комерція повільно, але впевнено стає невід'ємною частиною електронної комерції, про що свідчить зростання глибинного навчання, що стоїть за нею. Сюди входить ринок розпізнавання зображень, який повинен збільшитися з 20,19 мільярдів доларів у 2018 році до 81,88 мільярда доларів до 2026 року – при цьому відмічають загальний річний темп зростання в 19,6 відсотка [22].

Таким чином, електронна комерція прямує до повного покриття людей різною продукцією і послугами, прагне забезпечення якомога більшого кола покупців та зацікавлених осіб. Торгівля стає доступною як ніколи, і на сьогодні ми вже не прив’язані до набору товарів на полицях магазинів, а можемо замовляти усе,

що потрібно, з будь-якого кінця світу, долаючи географічні та часові обмеження. Зростання різноманіття методів оплати, технологій представлення товарів та послуг також сприяють цьому та збільшують привабливість для покупців саме онлайн-майданчиків. Усе перелічене зменшує витрати реалізації для національних та міжнародних ринків, що призводить до зниження цін для самих покупців.

1.2 Принципи функціонування електронної комерції.

На сьогоднішній день для подальшого розвитку та функціонування бізнесу необхідно максимально розширити базу потенційних клієнтів, зробити канали збуту більш доступними. Для виконання цих завдань необхідна електронна комерція. Успішний запуск бізнесу в мережі Інтернет вимагає дотримання певних правил і знання принципів e-commerce.

Дослідник К. Келлі [38] сформулював основні принципи функціонування електронної економіки:

Принцип єдиного системного зв'язку. Персональні комп'ютери й інші комп'ютерні пристрої пов'язані між собою через телекомунікації і утворюють всесвітню мережу.

Принцип повноти. В електронній економіці цінність товару/послуги зумовлена різноманітністю пропозицій. Це означає, що чим більше товарів у мережі, тим ціннішими вони стають. Проте цей принцип суперечить відомим 28 аксіомам, які відбивають відповідні закономірності традиційної економіки (перша аксіома: цінність визначається рідкістю товару, оскільки його кількість обмежена; друга аксіома: надмірне виробництво товарів призводить до значної втрати його цінності).

Принцип експоненти – розвиток електронної економіки відбувається експонційно, що пов'язано з нелінійним характером збільшення кількості її елементів.

Принцип зростаючого ефекту. Прихід в електронну економіку нових учасників призводить до збільшення розмірів мережі. Завдяки збільшенню обсягів мережі Інтернету до неї потрапляє все більша кількість бізнесменів. Зрештою збільшується обсяг продажу товарів (послуг), який призводить до зростання обсягу

отримуваного прибутку учасника бізнес-процесів.

Принцип «безоплатності». В електронній економіці цінність товару (послуги) прямо пропорційна масштабу його поширення. Тому зростання кількості наданих користувачам копій (наприклад, програмних продуктів) призводить до збільшення і цінності кожної з них. Продаючи варіанти продукту, які у майбутньому модернізуються, і додаткове сервісне обслуговування до нього, Інтернет-компанія може постійно і цілком достатньо заробляти. При цьому вона продовжує безкоштовно поширювати початкову версію продукту. У зв'язку з цим існують такі правила:

1. Необхідно поставляти на Інтернет-ринок безкоштовні послуги, продукти для розширення кола майбутніх покупців продукту, що модернізується.

2. Пропонуючи один продукт безкоштовно, інші продукти легше продавати.

3. Для формування в перспективі потрібного підприємцеві обсягу попиту на продукт потрібно пропонувати зацікавленим покупцям у безкоштовному користуванні початкову версію цього продукту.

Принцип лояльності. Сутність цього принципу полягає у тому, що прихильність покупців до певної Інтернет-компанії зумовлює водночас і застосування мережі і мережевих платформ. Якщо в традиційній економіці рівень якості життя кожного громадянина здебільшого залежить від ефективності функціонування національної економіки, то в Інтернеті добробут громадянина визначається рівнем процвітання мережі. З цього дійшли висновку: для забезпечення максимально високого рівня життя кожного громадянина необхідно всіляко сприяти розширенню й удосконаленню мережі і можливостей у ній працювати.

Принцип глобалізації. Електронна економіка – це сукупність тісно пов'язаних між собою ринків у світовому масштабі. Географічне розташування Інтернет-компаній не має принципового значення. Будь-який бізнес у мережі розповсюджується практично миттєво по всіх країнах світу. З такою ж швидкістю з'являються і конкуренти, що пов'язано зі зростанням різних ризиків. Могутнім американським Інтернет-компаніям, які займаються бізнесом у сфері

телекомунікації, дуже серйозну конкуренцію складають аналогічні компанії Європейського Союзу [19].

Отже, якщо ви хочете, аби ваш e-commerce досяг успіху, вам необхідно забезпечити зручність для клієнта, ваш сайт має запам’ятовуватися легко. Також ви маєте знати цільову аудиторію своїх покупців. Необхідно дотримуватися вищезазначених принципів для забезпечення зростання вашої мережі, бо саме її розмір визначає ваш дохід. Необхідно не забувати, що з інтернетом зникають кордони і продаж власного продукту може здійснюватися у всьому світі.

1.3 Аналіз бізнес-процесів підприємства електронної комерції.

Аналіз бізнес-процесів (BPA) - це методологія для розуміння стану здоров'я різних операцій у бізнесі для підвищення ефективності процесів. Це спеціалізований метод у широкому контексті управління бізнес-процесами для аналізу того, чи відповідають поточні процеси своїм цілям. Використання аналізу бізнес-процесів допомагає визначити згубні елементи в операції та визначити, як подолати перешкоди [28].

Випадки, коли необхідно використовувати BPA:

- Невідомі проблеми, такі як регулярні затримки або посилення скарг клієнтів.
- Зацікавленим сторонам у процесах незрозуміло, як провести процес.
- Перед тим, як запровадити автоматизацію, необхідно переконатися, що процес оптимізований.
- Команда хоче замінити процес новою версією.

Аналіз бізнес-процесів відповідає 4-кроковому плану:

- Визначте процес. Перший крок - вибір процесу «AS-IS» для аналізу та визначення зацікавлених сторін, які ним займаються. Переконайтеся, що маєте чіткі межі початку та зупинки для процесу.
- Збір інформації про процес. Далі, необхідно зібрати якомога більше інформації про процес, щоб зрозуміти проблеми, з якими він стикається, цілі, сферу вдосконалення та інші цілі аналізу.
- Проаналізуйте «AS-IS» процес. Виконайте план аналізу бізнес-процесів.

Підійдіть до нижньої частини ідентифікованого процесу, визначте процес у блок-схемах та інших діаграмах та виміряйте його ефективність.

- Розробіть план «ТО ВЕ». Нарешті, скористайтеся аналізом, щоб скласти рекомендації щодо того, як повинен виглядати процес «ТО ВЕ». Вказати на вимоги, запропонувати ресурси та зміни, визначити часовий графік тощо[28].

Бізнес-процеси «за результатами» поділяють на основні, обслуговуючі, управлінські та розвитку. Визначимо бізнес-процеси та зберемо інформацію про них. Див. Таблицю 1.1.

Таблиця 1.1

Основні бізнес процеси

Закупівля	
Вхід	інформація про замовлення підприємством товарів та грошові кошти для закупівлі товарів
Вихід	закуплені товари від постачальників
Ресурси	інформація про затверджений перелік товарів, інформація про постачальників товарів, персонал складу, транспорт, грошові кошти у вигляді заробітної плати відповідних працівників, кошти на оплату закуплених товарів.
Управлінський механізм	договори із постачальниками
Виконавець	менеджер із закупок
Керівник	менеджер із закупок
Власник	генеральний директор
Споживач	менеджер та працівники складу
Розміщення на складі	

Продовження табл. 1.1

Вхід	надходження товарів в магазин від постачальників
Вихід	відвантажені, перевірені за накладними товари на складі
Ресурси	інформація про заплановане прибуття товарів, працівники складу, грошові кошти у вигляді заробітної плати відповідних працівників, складське приміщення
Управлінський механізм	регламент прийняття та зберігання товарів на складі, документація товарно-транспортними накладними, накладними на тару, сертифікати якості продукції, правила товарного сусідства
Виконавець	працівники складу
Керівник	менеджер складського приміщення
Власник	генеральний директор
Споживач	адміністратори онлайн-майданчика
Просування товарів	
Вхід	товари на складі та інформація про асортимент
Вихід	представлення інформації про товари на онлайн-майданчику, реклама в інтернеті, sms-розсилка та фізична реклама

Продовження табл. 1.1

Ресурси	інформація про товари, приміщення з робочим обладнанням та робочими місцями для працівників, кошти на проведення маркетингових заходів, персонал з маркетингу
Управлінський механізм	маркетингова стратегія підприємства
Виконавець	адміністратори та менеджер із маркетингу
Керівник	менеджер із маркетингу
Власник	генеральний директор
Споживач	сторони купівлі-продажу, тобто магазин та покупець магазину
Оплата товару клієнтом	
Вхід	покупець магазину зі сформованим набором товарів для покупки
Вихід	одержання грошових коштів накладеним платежем або на банківський рахунок магазину
Ресурси	інформація про товари, приміщення з робочим обладнанням та робочими місцями для працівників, особисті кошти покупця та безпосередньо товари магазину
Управлінський механізм	порядок ведення розрахункових операцій в національній валюті України
Виконавець	адміністратори магазину
Керівник	менеджер із продажів

Продовження табл. 1.1

Власник	генеральний директор
Споживач	сторони купівлі-продажу, тобто магазин та покупець магазину
IT-підтримка	
Вхід	інформація про необхідність технічного забезпечення, програмні баги
Вихід	адміністрування сайту, обслуговування техніки підрозділів підприємства, рефакторинг, виправлення багів
Ресурси	техніка підприємства, засоби для ремонту, грошові кошти на виплату заробітної плати системному адміністратору, програмне забезпечення
Управлінський механізм	інструкція до техніки, нормативно-правові акти про безпечне поводження з технікою, документація до ПЗ
Виконавець	системний адміністратор, фріланс програмісти
Керівник	системний адміністратор
Власник	системний адміністратор
Споживач	структурні підрозділи підприємства, покупці
Процес продажу	

Закінчення табл. 1.1

Вхід	інформація про асортимент та інформація від клієнта
Вихід	створене замовлення та отримані кошти від клієнта
Ресурси	інформація про товари, приміщення з робочим обладнанням та робочими місцями для працівників, товари розміщенні на складі, персонал з маркетингу та працюючі на складі
Управлінський механізм	договори з продажу
Виконавець	адміністратори та менеджери із продажу
Керівник	менеджер із продажу
Власник	генеральний директор
Споживач	сторони купівлі-продажу, тобто магазин та покупець магазину

Керуючі бізнес процеси визначені та перелічені у Додатку А.

Висновки до розділу 1.

Під поняттям «e-commerce» розуміють «процес використання електронних методів та процедур для ведення всіх форм ділової діяльності». Водночас електронна комерція, як і вся Інтернет-мережа, зростають за експонентою, що в свою чергу робить ринок для інвестицій досить привабливим. IT постійно створює засоби, які потім стають частиною електронної комерції, наприклад заміна мобільним шопінгом настільного. Це, в свою чергу, змушує підприємців постійно тримати руку на пульсі трендів та новинок, аби не втратити покупців та прибутки.

Для побудови онлайн-майданчика, необхідно дотримуватися принципів за якими розвивається e-commerce, при цьому може виникнути складність через трансформаційні процеси, які викликані розвитком онлайн-простору та

переміщенням бізнесів із фізичного світу в цифровий. Також треба коректно спроектувати дизайн-сайт, аби споживач не відмовився від послуг, та аби сайт вдалося швидко знайти в мережі.

З метою зробити наше підприємство та веб-систему ефективними у дослідженні проаналізували бізнес-процеси, а також виявили можливі перешкоди в роботі підприємства. За результатами аналізу було виокремлено 4 групи бізнес-процесів. Зібрані дані необхідні для подальшого проектування бізнес-моделі підприємства та для автоматизації його роботи. Під час аналізу бізнес-процесів ми переконалися, що процеси відповідають поставленим цілям, а також загальній меті підприємства – мати прибуток та зайняти свою нішу на ринку.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

2.1 Створення операційної-моделі підприємства.

Моделювання бізнес-процесів (BPM) в управлінні бізнес-процесами та інженерії систем - це діяльність, яка представляє процеси підприємства, щоб поточний процес міг бути проаналізованим, удосконаленим та автоматизованим. BPM, як правило, виконують бізнес-аналітики, які надають експертизу з дисципліни моделювання; експертами з предметів, які мають спеціалізовані знання про модельовані процеси; або частіше командою, що складається з обох. Крім того, модель процесу може бути отримана безпосередньо з журналів подій за допомогою інструментів майнінгу процесів [2].

Моделювання бізнес-процесів ще корисно й тим, що:

- Для створення візуальних моделей процесів - документації, керованої Word, часто недостатньо, щоб працівники розуміли спосіб виконання процесу. Візуальні уявлення допомагають забезпечити всебічну картину.
- Вирівнювання операцій - за будь-якої нової бізнес-стратегії, щоб підтримувати послідовність процесів після змін, потрібно з'ясувати, як залишатися в межах загальної організаційної стратегії. Аналізи також проводяться для виявлення проблемних місць, неефективності та з метою забезпечення гнучкості процесу.
- Поліпшення комунікації процесів - комунікація є ключовим фактором для всіх наступних завдань: формалізації існуючих процесів (які були колись неофіційними знаннями), створення послідовних процесів, усунення здогадок у ділових правилах, робота з винятками, забезпечення дотримання регуляторних норм, забезпечення відповідальними діловими людьми та підтримка нових ініціатив.
- Підвищення операційної ефективності - процеси моделювання сприяють оптимізації, дозволяючи моделювати та ілюструвати необхідні вдосконалення. Це скорочує час циклу та сприяє кращому використанню ресурсів.

- Отримайте конкурентну перевагу - процес краще в цілому, коли він постійно удосконалюється та узгоджується зі стратегіями свого бізнесу. Ця ефективність дозволяє компанії рухатися вперед, щоб бути кращою за конкуренцією [8].

Для того, щоб побудувати правильно працюючу модель потрібно керуватися наступними правилами:

1. Перед тим, як щось запустити, необхідно побудувати графічну модель, ретельно її обміркувавши.
2. Якщо не опрацьовували модель до її фактичного запуску, необхідно бути готовим до внесення змін у процесі роботи.
3. Важливо бути готовим до перевірки моделі в реальності.
4. Перш ніж внести зміни в процесі роботи треба прописати їх у модель і виконати всі попередні пункти.

Оскільки в основі кожного бізнесу орієнтуються на клієнта, ми повинні вибудувати нашу бізнес-модель навколо нього. На основі взаємодії підприємства і клієнта необхідно визначити бізнес-процеси, перелічені в попередньому розділі: процес продажу, розміщення на складі, просування товарів, оплата клієнтом.

Для створення графічної схеми необхідно використати нотацію BPMN. Ця нотація є стандартом для моделювання бізнес-процесів, що надає графічну нотацію для визначення бізнес-процесу у вигляді «Діаграми бізнес-процесу» (Business Process Diagram, BPD). Така діаграма ґрунтується на представленні бізнес-процесу у вигляді блок-схеми, що семантично схожа на діаграму діяльності.

Метою BPMN є підтримка моделювання та управління бізнес-процесами. При чому єдина модель бізнес-процесу повинна бути зрозумілою для всіх користувачів (зацікавлених осіб). Водночас, нотація дає можливість визначати складну семантику бізнес-процесів. Для спрощення розуміння та використання стандарту пропонується розбити елементи нотації на два рівні: базових елементів нотації та елементи, що дають можливість визначити всі (технічні) деталі бізнес-процесу [13]. Див. Рис. 2.1.

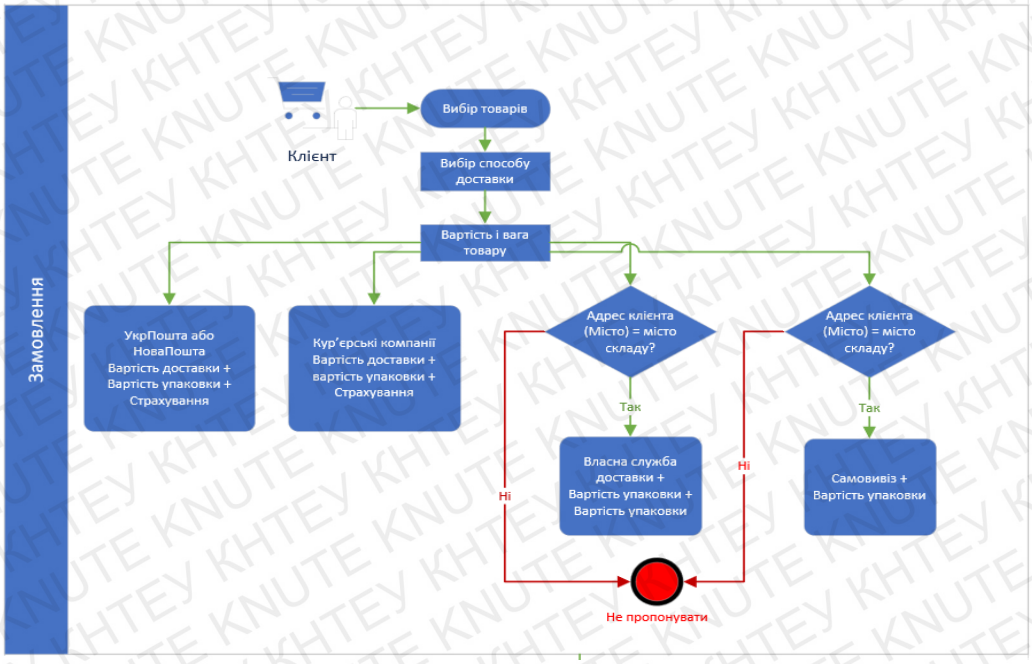


Рис 2.1 Схема процесу вибору товарів та продажу[Авторська розробка]

На першому етапі клієнт має обирати товари, а потім відібрати оптимальний варіант доставки. Після цього формується вартість доставки. На другому етапі проходить вибір оплати замовлення, спосіб оплати залежить від виду доставки. Див. Рис.2.2.

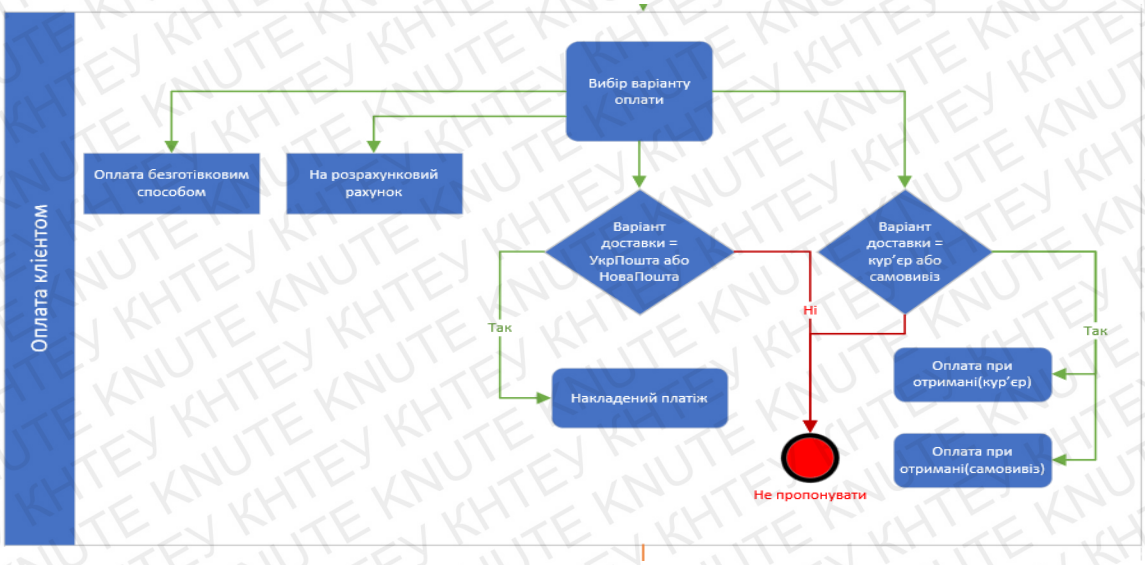


Рис 2.2 Схема процесу оплати клієнтом[Авторська розробка]

На третьому етапі, проходить сама оплата та підтвердження замовлення. Див. Рис. 2.3.

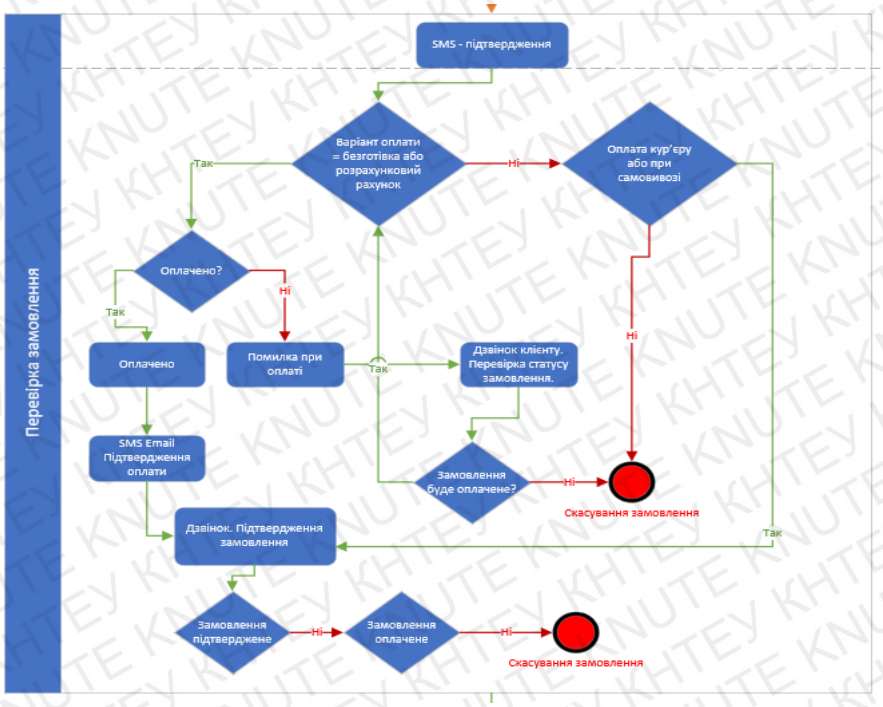


Рис 2.3 Схема підтвердження замовлення[Авторська розробка]

На останньому етапі відбувається видача замовлення зі складу, його доставка та закриття виконаного замовлення. Див. Рис. 2.4.

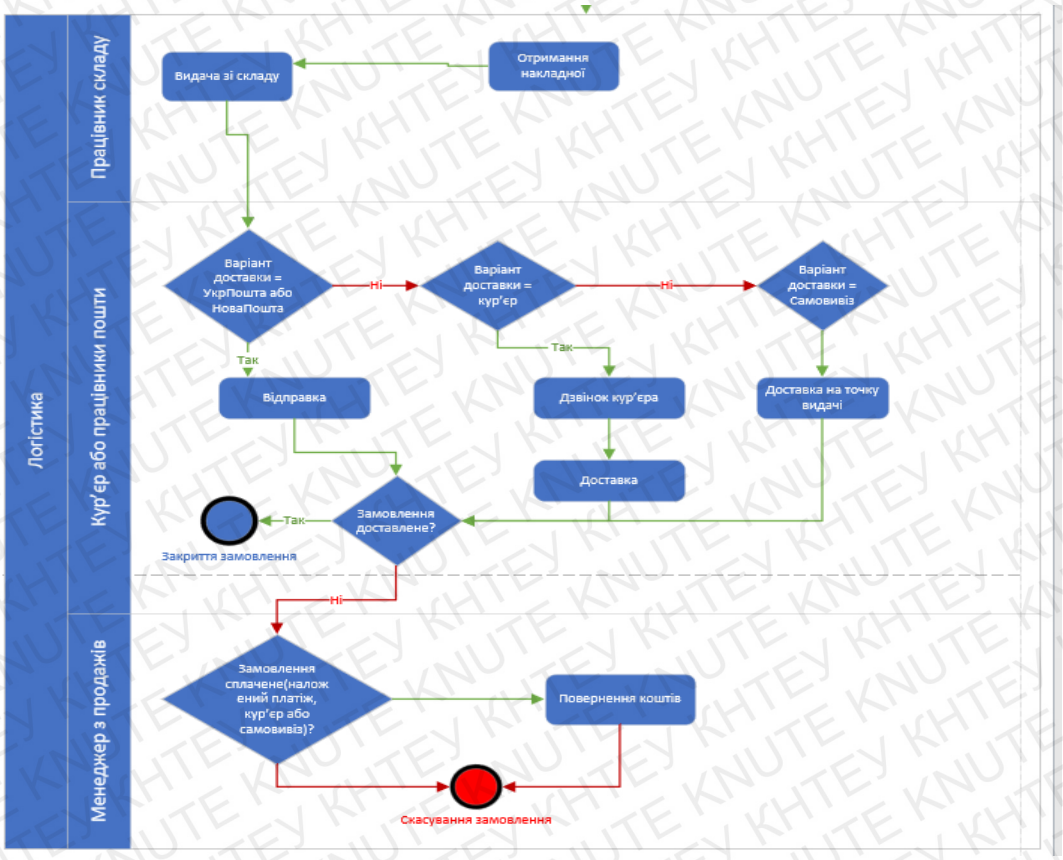


Рис 2.4 Схема логістики[Авторська розробка]

Якщо на попередніх етапах процеси відбуваються автоматично внаслідок дій

клієнта у веб-системі, то наприкінці процесу беруть участь вже й працівники різних відділів. Так, спочатку мають відбутися дії працівників складу, потім служби доставки, а у випадку неотриманого замовлення, за роботу береться й менеджер з продажів.Відповідно до побудованої моделі, при моделюванні веб-системи необхідно розробити застосунки для обліку замовлень, де будуть передбачені поля для вибору способів доставки та оплати. Також необхідно впровадити облік клієнтів. У системі має бути представлена можливість змінювати статус замовлень та повідомляти про це покупців.

2.2 Стан та проблеми управління підприємством електронної комерції.

Як і у будь-якому бізнесі, керівники стикаються із проблеми при управлінні і в електронній комерції, яка не стала виключенням. Див. Рис. 2.5.

Рис 2.5 Схема чинників, що зумовлюють виникнення проблем в управлінні підприємствами електронної торгівлі [15]

Нерозвиненість електронних засобів платежу як наслідок загальної непоширеності та непопулярності серед населення банківських платіжних карток, незважаючи на їх суттєве зростання в останні роки, значною мірою гальмує розвиток електронної торгівлі та використання її можливостей для задоволення платоспроможного попиту споживачів. Щодо методів оплати, то на базі експертного опитування (див. Рис. 1.2) можна стверджувати, що, незважаючи на

популярність безготівкової форми розрахунку, основна маса респондентів найчастіше здійснює оплату через кур'єра (близько 49%). Це свідчить про наявність недовіри до безготівкової форми розрахунку внаслідок відсутності законодавства, яке б захищало права споживачів. Значною перепоною у поширенні електронної торгівлі виступають національні традиції, менталітет вітчизняних споживачів.

Основна маса українців під час вибору товару прагне отримати якнайбільше інформації про нього, в т. ч. органолептичну (потримати в руках, оглянути, зважити всі «за» і «проти»), що дуже часто мінімізується саме в електронних магазинах і зумовлює обмеженість асортименту товарів, що реалізуються через них. Недобросовісні конкуренти, шахраї, які користуються неосвіченістю населення щодо отримання й оплати товарів та послуг в електронних магазинах в умовах недосконалої законодавчої бази, підривають імідж електронної комерції, формують відповідне ставлення у населення до придбання товарів у мережі Інтернет. Електронний магазин повинен висвітлювати контактні дані, проводити маркетингові заходи для тощо, щоб споживач міг пересвідчитися в існуванні магазину, якості пропонованих ним товарів та послуг, пропонувати різні варіанти розрахунків за товари, організовувати за необхідності післяпродажне обслуговування покупців, ініціювати наявність «книги відгуків» про електронний магазин, стимулювати формування лояльності у покупців тощо.

Однією з найважливіших проблем управління підприємствами електронної торгівлі в Україні є відсутність чіткого законодавства, яке однозначно описує використання електронно-цифрового підпису і засобів захисту інформації, а також законів, які б зазначали правові засади функціонування підприємств електронної торгівлі. Це, у свою чергу, супроводжується демпінгом цін, недобросовісною конкуренцією тощо. Напрям інформаційно-правової підтримки процесу електронної торгівлі повинен забезпечувати учасників електронної торгівлі повною інформацією щодо правових засад здійснення операцій у цій галузі, в тому числі – щодо регламентації порядку купівлі-продажу товарів саме в інтернет-магазині та порядку вирішення спорів між сторонами угоди (наприклад, питання

щодо укладення договорів, відповідальності сторін, податків, мита тощо). Цей аспект повинен реалізовуватися у двох частинах інформаційно-правового забезпечення (ІПЗ): внутрішньому (ІПЗ усередині країни) та міжнародному (ІПЗ на рівні міжнародних відносин) [24].

Розглядаючи світовий досвід, дослідники визначили ті фактори, що впливають на управління [15]:

- місце укладання е-угоди;
- підтвердження факту укладання е-угоди;
- адреса веб-представництва постачальника (продавця);
- повернення товарів, робіт і послуг, отриманих дистанційно;
- пост-гарантійне обслуговування товарів, робіт і послуг, отриманих через ІКТ;
- інформаційна безпека фінансових даних споживачів;
- неповноцінне використання міжнародних електронних платіжних систем;
- гарантування законності угод;
- оподаткування операцій в електронному вигляді;
- захист прав споживачів;
- захист даних (у тому числі персональних) тощо.

З вищезазначеного зрозуміло, що електронній комерції необхідна регуляція, однак залежно від моделі державного контролю у підприємства можуть виникнути проблеми при управлінні.

Для прикладу розглянемо американську та європейську моделі.

Американська модель передбачає практично повну відмову держави від втручання у сферу е-торгівлі з метою максимізації вигоди від використання потенціалу мережі для економіки. Сутність американської моделі регулювання е-торгівлі полягає в тому, щоб створити інституційні умови для її просування та пріоритетного розвитку відповідної інфраструктури на території США. Саме тому там знаходяться найбільші е-майданчики («Amazon», «eBay») і платіжні системи

(«PayPal») з мільярдними оборотами [26]. Європейська модель передбачає тотальну регламентацію та реєстрацію суб'єктів е-торгівлі та їхніх угод. Прикладом такої діяльності може слугувати введення провідними європейськими країнами (Німеччиною, Францією і Швейцарією) загальнодоступного реєстру добросовісних продавців, зареєстрованих у податкових органах. На сайтах з товарними пропозиціями продавці зобов'язані вказувати ідентифікаційний номер своєї державної реєстрації. Зазначимо, що європейська модель не зовсім виправдовує себе у сфері C2C (споживач для споживача), оскільки європейська торговельна інформаційна інфраструктура не є світовим лідером в е-комерції через інституційні обмеження. Виграють від неї в основному великі товаровиробники і роздрібні торговельні мережі за рахунок обмеження зовнішньої конкуренції. Таким чином, управління процесами в європейський країнах обійдеться дорожче, а відповідно створює нерівні умови для гравців на ринку.

Варто також відзначити, що з ростом очікувань клієнтів та попиту на дані, маркетологи з електронної комерції знаходять більше даних, ніж вони можуть обробити. Прагнучи краще співпрацювати з існуючими клієнтами (і виявляти нових), бренди поспішають купувати нові технології, які можуть збирати дані про поведінку та залучення клієнтів. Накопичення даних та технологій, вимагає великого обсягу ресурсів, а відповідно вимагає значних вкладень. Замість цього необхідно вкладати кошти у всеохоплюючі рішення, які охоплюють не тільки маркетинг, а інші потреби.

Отже, основними проблеми в управлінні є прогалини в законодавстві, які можуть викликати проблеми в операційній діяльності, нерозвиненість фінансової сфери, необізнаність населення та зациклення на одній ділянці роботи.

2.3 Проектування веб-системи управління.

Інформаційна система управління - сукупність інформації, економіко-математичних методів і моделей, технічних, програмних, інших технологічних засобів і фахівців, призначена для обробки інформації та прийняття управлінських рішень.

Використовуючи оперативну інформацію, отриману під час функціонування ІС, керівник може спланувати і збалансувати ресурси фірми; прорахувати і оцінити результати управлінських рішень; налагодити оперативне управління собівартістю продукції, ходом виконання плану, використанням ресурсів тощо.

Інформаційні системи управління дозволяють:

- здійснювати збір, зберігання і оперативний доступ до облікової інформації фірми;
- забезпечувати зростання продуктивності праці, скорочення невиробничих втрат;
- підвищувати ступінь обґрунтованості і своєчасність прийнятих рішень;
- домагатися зростання ефективності управління за рахунок повного уявлення та своєчасного надання необхідної інформації керівникам усіх рівнів управління з єдиного інформаційного фонду;
- погоджувати рішення, що приймаються на різних рівнях управління і в різних структурних підрозділах. ІС складається з декількох підсистем, кожна з яких реалізує свою задачу і функцію.

Для обробки типових задач з боку великої кількості користувачів постає необхідність в автоматизації. В нашому випадку дії у користувацькій частині веб-системи повинні автоматично призводити до їх відображення в адміністративній частині.

Автоматизовані інформаційні системи (АІС) - це системи із застосуванням сучасних засобів автоматизованої обробки даних, економіко-математичних та інших методів для регулярного розв’язування задач управління виробничо-господарською діяльністю підприємства.

Корпоративні АІС використовуються для автоматизації всіх функцій управління фірмою чи корпорацією, яка має територіальну відокремленість підрозділів, філій, відділів, офісів тощо.

Для створення інформаційної системи необхідно спочатку створити модель, де б відображалися бізнес-процеси та інформаційні потоки.

Інформаційна модель – це модель, що описує інформаційні процеси або містить інформацію про властивості і стан об'єктів, процесів, явищ [8]. Див. Рис. 2.6.

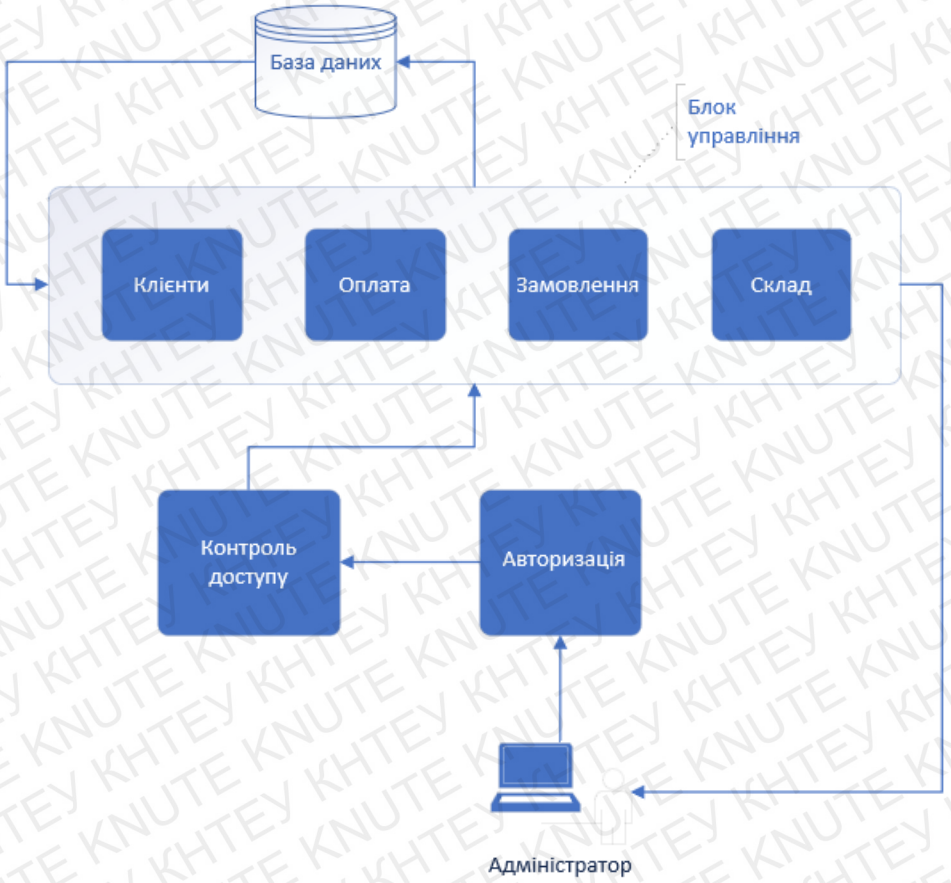


Рис 2.6 Схема моделі веб-системи[Авторська розробка]

За управління буде відповідати адміністратор сайту або менеджер з продажів. Для захисту системи від стороннього впливу необхідно створити модуль авторизації та організувати контроль доступу на основі ролей. Блок управління буде містити 4 модулі: модуль управління замовленнями, клієнтами, оплатою та управління складом.

На основі інформаційної моделі ми створимо базу даних на СУБД MongoDB. MongoDB — документо-орієнтована система управління базами даних (СУБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СУБД, функціональними і зручними у формуванні

запитів [4]. Вона має задовольняти умови відображені у моделі, це і контроль доступу (розмежування доступу користувачів системи), швидка відповідь на запити користувачів через веб-додаток та безпека даних. Див. Рис. 2.7.

Рис 2.7 Логічна модель БД створена на основі моделі веб-системи[Авторська розробка]

Варто зазначити, що MongoDB – це нереляційна база даних. Відповідно ключові поля у різних колекціях не є зв'язаними. Однак, на моделі зазначені зв'язки, так як вони створюються у самому програмному коді нашої системи. Така модель була обрана задля більш легкої масштабованості та більшої гнучкості. Наприклад, після змін набору атрибутів ми зможемо додати атрибути тільки для нових записів БД. Оскільки кортежі у MongoDB представлені об'єктами JSON, то у нас є можливість зберігати масиви та об'єкти за атрибутами.

Головною колекцією є Orders, адже кортежі цієї таблиці необхідні для створення записів у інших таблицях. Також варто відмітити створення колекції Payments, яка дублює деякі атрибути колекції Orders. Це зроблено для контролю бухгалтерського обліку, адже у нашій системі передбачені різні методи оплати, і кошти необов'язково перераховується відразу після створення замовлення.

Отже, ми створили модель інформаційної системи управління на основі бізнес-процесів та потоків інформації зазначених у підрозділі 2.1 та базу даних на основі MongoDB.

Висновки до розділу 2.

Таким чином, створено модель бізнес-процесів нашого інтернет-магазину, на основі бізнес-процесів розглянутих у першому розділі. Така модель необхідна для розуміння чіткості дій та відсутності проблем із тлумаченням правил. Для керівників це дає змогу легше проводити зміни та поліпшувати ефективність управління. У моделі було розглянуто кілька варіантів доставки та оплати, а також випадки для скасування замовлень.

Серед чинників, які впливають на підприємства електронної комерції є прогалини у законодавстві, бюрократичні процедури, нерозвиненість ринкової інфраструктури, необізнаність населення. Серед моделей регуляції вирізняються американська та європейська. Для американської характерно створення інституційних умов для просування е-бізнесу, а для європейської – сувора регламентація бізнес-діяльності в інтернеті.

Моделювання веб-системи управління базувалося на бізнес-процесах та інформаційних потоках на підприємстві. Мета системи автоматизувати їх та забезпечити облік інформації. Важливою складовою є контроль доступу до блоку управління. Блок складається з таких частин: модуль замовлень, оплати, клієнтів та складу. Для бази даних вибрана СУБД - нереляційна система MongoDB, яка дає кращі можливості для гнучкості проекту та його масштабування у порівнянні із реляційними аналогами.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПІДПРИЄМСТВОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

3.1. Інструменти та методи розробки веб-системи управління.

З переходом до прикладного рівня, розробки електронного майданчика, з'являється необхідність дотримання правил формування сайту, адже від нього залежить наскільки швидко інвестиції окупляться. На сьогоднішній день існують різні програмні підходи щодо створення інформаційних систем. Це в свою чергу веб, розробка настільних рішень, створення децентралізованих систем тощо.

Чому найкращим рішенням буде створення веб-системи?

По-перше, доступ до нашої інформаційної системи мають всі, незалежно від того, де люди знаходяться. По-друге, система має просте адміністрування, адже керувати можуть люди і без спеціальної освіти. Це спричинено специфікою процесів електронної комерції. По-третє, розробка рішення не займає багато часу, дивлячись на складність та розмір інформаційної системи. Враховуючи перелічені аргументи, необхідно визначити, яку архітектуру ми будемо використовувати. В нашому випадку найкраще підходить REST API.

REST (скор. Від англ. Representational State Transfer – «передача стану представлення») - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгодженим набором обмежень, що враховуються при проектуванні розподіленої інформаційної-системи [6]. У певних випадках (інтернет-магазини, пошукові системи, інші системи, засновані на даних) це призводить до підвищення продуктивності і спрощення архітектури. У широкому сенсі компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів у Всесвітній павутині. Докладніше це йдеться у Додатку Б.

REST API – дає змогу за допомогою CRUD-операцій синхронізувати нашу систему із сторонніми сервісами і створити додатки на різних платформах. Крім цього, розширення REST не викличе проблем у системі, адже необов'язково використовувати новостворені функції.

Коли, ми визначилися з програмним підходом постає питання: «Які технології веб ми маємо застосовувати?»

На сьогоднішній день основу веб-технологій для створення сучасних веб-додатків складає комбінація таких технологій як: HTML, CSS, JavaScript (бек-енд на платформі Node.js та фронт-енд на React.JS).

Тому розглянемо кожну технологію окремо:

HTML (від англ. HyperText Markup Language - «мова гіпертекстової розмітки») - стандартизована мова розмітки документів у Всесвітній павутині. Більшість веб-сторінок містить опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами; отриманий у результаті інтерпретації форматований текст відображається на екрані монітора комп'ютера або мобільного пристрою.

HTML - документ складається з дерева HTML-елементів і тексту. Кожен елемент позначається в вихідному документі початковим (відкриває) і кінцевим (закриває) тегом (за рідкісним винятком). Початковий тег показує, де починається елемент, кінцевий - де закінчується. Закриваючий тег утворюється шляхом додавання слеша / перед ім'ям тега: <ім'я тега> ... </ ім'я тега>. Між початковим і закриваючим тегами знаходиться вміст - тега-контент. Поодинокі теги не можуть зберігати в собі вмісту безпосередньо, воно прописується як значення атрибута, наприклад, тег <input type = «button» value = «Кнопка»> створить кнопку з текстом «Кнопка» всередині. Теги можуть вкладатися один в одного, наприклад, <p> <i> Текст </ i> </ p>. При вкладенні слід дотримуватися порядку їх закриття (принцип «матрьошки»), наприклад, такий запис буде невірною: <p> <i> Текст </ p> </ i>.

JSX, або JavaScript XML (TSX для typescript), є розширенням синтаксису мови JavaScript [31]. Подібно до зовнішнього вигляду HTML, JSX надає спосіб структурувати візуалізацію компонентів за допомогою синтаксису, знайомого багатьом розробникам.

Каскадні таблиці стилів (CSS) - це мова сторінок стилів, що використовується для опису презентації документа, написаного на мові розмітки. Найчастіше використовується для встановлення візуального стилю веб-сторінок та користувацьких інтерфейсів, написаних у HTML та XHTML, мова може бути застосована і до будь-якого XML-документа, включаючи простий XML, SVG і

XUL, і застосовується до рендерингу з HTML і JavaScript. CSS є основою технології, що використовується більшістю веб-сайтів для створення візуально привабливих веб-сторінок, користувацьких інтерфейсів для веб-програм і користувацьких інтерфейсів для багатьох мобільних додатків [29].

JavaScript (часто просто JS) - це легка мова програмування, що інтерпретується або є JIT-компільованою, об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache CouchDB. JavaScript - це прототипно-орієнтована, мультипарадигмна мова з динамічною типізацією, яка підтримує об'єктно-орієнтований, імперативний і декларативний (наприклад, функціональне програмування) стилі програмування [3].

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та привернула увагу професійної спільноти до мови, а подальші модифікації мови внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

TypeScript - це мова програмування, розроблена та підтримувана корпорацією Microsoft. Це строгий синтаксичний набір JavaScript, що додає додаткову статичну типізацію до мови. TypeScript призначений для розробки великих додатків та перекомпіляції в JavaScript. Оскільки TypeScript є надмножиною JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript[39].

Node.js - це середовище виконання з відкритим вихідним кодом, міжплатформене, внутрішнє середовище JavaScript, яке виконує код JavaScript поза веб-браузером. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера -

запуску сценаріїв на стороні сервера для створення динамічного вмісту веб-сторінки до того, як сторінка буде надіслана до веб-браузера користувача. Отже, Node.js представляє парадигму "JavaScript скрізь" [34], що об'єднує розробку веб-додатків навколо однієї мови програмування, а не різних мов для серверних та клієнтських сценаріїв.

Хоча «.js» є стандартним розширенням імені файлу для коду JavaScript, назва «Node.js» не посилається на певний файл у цьому контексті, а є просто назвою продукту. Node.js має керовану подіями архітектуру, здатну до асинхронного вводу-виводу. Ці шляхи проектування спрямовані на оптимізацію пропускної здатності та масштабованості у веб-додатках з великою кількістю операцій введення /виведення, а також для веб-додатків у режимі реального часу (наприклад, програм спілкування в реальному часі та браузерних ігор) [41].

Express.js, або просто Express - це внутрішня веб-програма для Node.js, випущена як безкоштовне програмне забезпечення з відкритим кодом під ліцензією MIT. Призначена для створення веб-додатків та API. Її називали де-факто стандартною серверною структурою для Node.js [32].

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript з відкритим вихідним кодом, інтерфейс, для побудови користувальницьких інтерфейсів або компонентів інтерфейсу. Підтримується Facebook та спільнотою окремих розробників та компаній. React можна використовувати як основу при розробці односторінкових або мобільних додатків. Основним завданням React є візуалізація даних у DOM, але його також можна розширити багатьма бібліотеками, такими як response-router або material-ui [5].

Отже, ми розглянули стек, який використовується для розробки системи-управління і складається із JavaScript(Node.js, React.Js, Express), HTML і CSS. За основу додатку був вибраний REST API для розширення можливостей управління.

3.2. Програмна реалізація веб-системи.

Система управління складається із серверної частини та клієнтської. Серверна частина буде відповідати за отримання та обробку даних через бізнес-логіку. Клієнтська буде відображати веб-інтерфейс, через який користувачі

системи будуть виконувати адміністрування даних та переглядати їх. Зв'язок між частинами виконується за допомогою AJAX-запитів.

Для початку ми створили структуру проекту. Потім виконали команду «npm init», вона проініціалізує проект та створить файл package.json, в якому містяться налаштування. Див. Рис. 3.1.

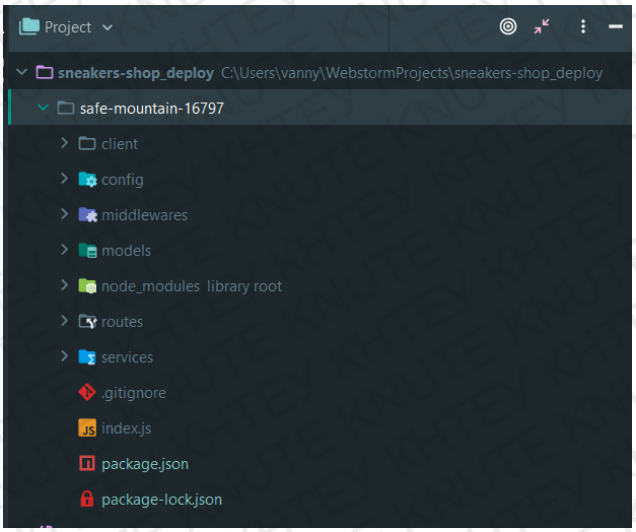


Рис 3.1 Скріншот структури проекту[Авторська розробка]

Далі записуються у файл необхідні дані для розробки репозиторію та налаштування запуску додатку. Див. Рис. 3.2.

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "engines": {
    "node": "^8.11.3",
    "npm": "^5.6.0"
  },
  "scripts": {
    "start": "node index.js",
    "server": "nodemon index.js",
    "client": "npm run start --prefix client",
    "dev": "concurrently \"npm run server\" \"npm run client\"",
    "heroku-postbuild": "NPM_CONFIG_PRODUCTION=false npm install --prefix client && npm run build --prefix client"
  },
  "author": "vannyy01",
  "license": "ISC",
  "dependencies": {
    "awesome-debounce-promise": "^2.1.0",
    "axios": "^0.18.0",
    "concurrently": "^3.6.1",
    "cookie-session": "^2.0.0-beta.3",
    "express": "^4.16.4",
    "lodash": "^4.17.11",
    "mongoose": "^5.10.11",
    "nodemon": "^1.19.0",
    "passport": "^0.4.0",
    "passport-google-oauth": "^2.0.0",
    "passport-google-oauth20": "^2.0.0",
    "react-async-hook": "^3.6.2",
```

Рис 3.2 Скріншот налаштування кореня системи та серверної частини[Авторська розробка]

Після цього була розроблена серверна частина. Вона починається зі створення корінного файлу index.js.

Лістнинг файлу index.js.

```
const express = require('express'); const mongoose = require('mongoose');
const cookieSession = require('cookie-session'); const passport = require('passport');
const keys = require('./config/keys'); require('./models/User');
require('./models/Order'); require('./services/passport');
mongoose.connect(keys.mongoDBConnect);
const app = express();
app.use(
  cookieSession({
    maxAge: 30 * 24 * 60 * 60 * 1000,
    keys: [keys.cookieKey]
  }));
app.use(passport.initialize()); app.use(passport.session()); app.use(express.json());
require('./routes/auttRoutes')(app); require('./routes/orderRoutes')(app);
require('./routes/commodityRoutes')(app);
if (process.env.NODE_ENV === 'production') {
  app.use(express.static('client/build'));
  const path = require('path');
  app.get('*', (req, res) => {
   res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
  })}
const PORT = process.env.PORT || 5000;
app.listen(PORT);
```

Для авторизації ми використовуємо cookies-сесію та Google+ API. Для підключення сервісів Google, спочатку потрібно активувати потрібні сервіси у Google developer console. Далі за допомогою бібліотеки passport.js ми будемо реєструвати та авторизувати клієнтів. (див. Додаток В) У файлі authRoutes.js ми

прописали маршрути для авторизації, виходу, та маршрути для управління клієнтами. (див. Додаток Г)

Далі створюються маршрути для обробки запитів, налаштування із ключами доступу до бази даних та сервісів Google, для безпеки додається prod та dev ключі у виключення Git. Також додається середовище для контролю доступу користувачів користувачів (код нижче, 20 – код адміністратора, 10 – користувача, 0 - заблокований). Нижче навединий лістинг файлу requireLogin.js.

Лістнинг коду requireLogin.js

```
module.exports = (req, res, next) => {
  if (req.user.role !== 20)
    return res.status(401).send({error: 'You must log in to continue.'});
  next(); };
```

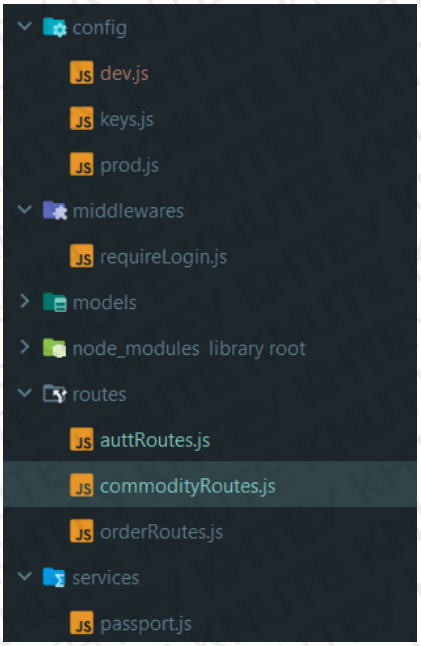


Рис 3.3 Скріншот маршрутів серверної частини[Авторська розробка]

Маршрути діляться на три категорії: замовлення, товари, користувачі. Частина з них відкрита, а частина вимагає валідації. Нижче наведено код commodityRoutes.js. маршрутів товарів.

Лісттинг коду commodityRoutes.js

```
const mongoose = require('mongoose');
const Commodity = mongoose.model('commodities');
const requireLogin = require('../middlewares/requireLogin');
```

```
module.exports = app => {
   app.get('/api/commodity/:to', async (req, res) => {
      await Commodity.find({}, function (err, comms) {
        if(err)      res.status(404).send(err);
        else res.send(comms);
}).limit(Number.parseInt(req.params.to);   });
          app.get('/api/commodity/get/:id', async (req, res) => {
              await Commodity.findById(req.params.id, function (err, comm) {
                if (err)
        res.status(404).send('Cannot get the goods list!');
                else {
        res.status(200).send(comm);
                } });
           });
        app.put('/api/commodity/edit/:id',requireLogin, async (req, res) => {
         await Commodity.updateOne({_id: req.params.id}, req.body, {upsert: true},
        function (err, comm){
             if (err){
              res.status(404).send(`Cannot update the good with _id: ${req.params.id}.
              Error: !${err}`);
        }      else {
        res.status(200).send(comm);
              }});
        });};
```

Для запитів до бази даних потрібно використати бібліотеку Mongoose і створенні на її основі моделі. Модель містить у собі назви полів та типи даних, які мають співпадати перед перетворенням у JSON. Приклади моделей наведені у Додатках Д, Е, Ж.

Далі перейдемо до клієнтської частини веб-додатку. В директорії client потрібно спочатку виконати команду «npx create-react-app-ts sneakers-shop-deploy»,

таким чином встановиться готовий каркас додатка на typescript. Потім здійснюється налаштування проекту у package.json директорії client. Налаштування tslint.json та tsconfig.json залишаємо незмінними. Після цього вносимо зміни у файл index.tsx, де підключаємо доступ до даних з усіх рівнів клієнтської частини через redux-state.

Лістнинг коду index.tsx

```
import * as React from 'react'; import * as ReactDOM from 'react-dom';
import {applyMiddleware, createStore} from "redux";
import App from './components/App';import reduxThunk from 'redux-thunk';
import reducers from './reducers'; import {Provider} from 'react-redux';
const store = createStore(reducers, {}, applyMiddleware(reduxThunk));
ReactDOM.render(<Provider store={store}><App/></Provider>,
document.getElementById('root'));
```

У файлі налаштувань потрібно прописати цілі proxy, це необхідно аби маршрути за вказаними цілями переадресовувалися із клієнтської частини на серверну.

Рис 3.4. Скріншот з налаштування клієнтської частини[Авторська розробка]

Далі необхідно розбити візуальну складову на частини, спільною частиною буде шапка сайту (NavBar), а наша система-управління буде розташована за маршрутом «/admin/*», доступ до якої буде відбуватися через захищений маршрут.

Код захищеного маршруту (файл ProtectedRoute.tsx)

```
import * as React from "react";
import {Redirect, Route, RouteProps} from "react-router-dom";
export interface ProtectedRouteProps extends RouteProps {
  authenticationPath: string;
  isAuthenticated: boolean;
}
export default class ProtectedRoute extends Route<ProtectedRouteProps> {
public render() {
    let redirectPath: string = '';
    if (!this.props.isAuthenticated) {
      redirectPath = this.props.authenticationPath;
    }
      if (redirectPath) {
const renderComponent = () => (<Redirect to={{pathname: redirectPath}}/>);
        return <Route {...this.props} component={renderComponent}/>;
}
      return <Route {...this.props}/>
  }}
```

Лістнинг коду відображення контенту (файл App.tsx)

```
public render() {
    return (
      <BrowserRouter>
        <div>
          <NavBar/>
          <div>
            {this.props.auth !== null &&
```

```
            <Switch>
              <Route exact={true} path="/" component={Landing}/>
              <ProtectedRoute
                authenticationPath="/auth/google"
                isAuthenticated={!!this.props.auth}
                path='/admin'
                component={AdminModule}
              />
            </Switch>
            }
          </div>
        </div>
      </BrowserRouter>
    )  }
```

Далі ми створили маршрути до модулів системи управління, це – склад, клієнти, замовлення, платежі. Див. Рис. 3.5.

```
<main className={classes.content}>
    <Switch>
        <Route path="/" exact={true} component={Default}/>
        <Route path="/admin/goods" exact={true} component={Goods}/>
        <Route path="/admin/goods/create" component={CreateGood}/>
        <Route path="/admin/goods/edit/:commID" component={Edit}/>
        <Route path="/admin/users" exact={true} component={Users}/>
        <Route path="/admin/users/create" component={CreateUser}/>
        <Route path="/admin/users/edit/:userID" component={EditUser}/>
        <Route path="/admin/orders" component={Default}/>
        <Route path="/admin/payments" component={Default}/>
    </Switch>
</main>
```

Рис 3.5 Скріншот маршрутів системи управління[Авторська розробка]

Потім створюються CRUD для усіх модулів. Приклад коду списку товарів наведений у Додатку К.

У списку створюються посилання для редагування та перегляду повної інформації про товар. Код форми редагування товару можна переглянути у Додатку Л.

Для взаємодії із серверною частиною використовується actions – функції для AJAX-запитів, а також reducers – для керування state додатка. Відповідно до методів http використовуються запити GET, PUT, POST. Нижче наведено код reducers та actions.

Лістнинг коду index.tsx (директорії reducers)

```
export const fetchGoods = (to: number) => async (dispatch: any) => {
const res = await axios.get(`/api/commodity/${to}`);
dispatch({type: FETCH_GOODS, payload: res.data})
};
export const fetchGoodByID = (id: string) => async (dispatch: any) => {
const res = await axios.get(`/api/commodity/get/${id}`);
dispatch({type: FETCH_GOOD, payload: res.data})
};
export const updateGood = (good: ShoeInterface, callback: () => void) => async
(dispatch: any) => {
const res = await axios.put(`/api/commodity/edit/${good._id}`, good);
dispatch({type: UPDATE_GOOD, payload: res.data, callback: callback()});
};
```

Лістнинг коду GoodsReducer.tsx

```
type AuthAction =  FetchGoodAction | FetchGoodsAction | UpdateGoodAction;
type StateType = ShoeInterface[] | [];
export const goodsReducer = (state: StateType = [], action: AuthAction) => {
  switch (action.type) {
    case FETCH_GOODS:
      if (action.payload.length > 0 && state.length > 0) {
return state[0]._id === action.payload[0]._id ? state : [...state,
...action.payload];
      }
      return action.payload;
    case FETCH_GOOD:
```

```
            return action.payload;
        case UPDATE_GOOD:
      return [];
  default:
    return state; }};
```

Отже, перелічені розробки дозволяють керувати підприємством електронної комерції через веб-інтерфейс, завдяки використанню серверної частини написаної на Node.js та express.js, де використовується організація доступу, взаємодія із СУБД MongoDB, робота з клієнтами через Google+ API та використання бізнес-логіки при обробці запитів. Через клієнтську частину адміністратори здатні проводити CRUD-операції через використання React.js та AJAX-запитів.

3.3. Аналіз результатів розробки веб-системи управління підприємством електронної комерції.

У результаті проведеної роботи було створено систему управління інтернет-магазином. Для входу до системи необхідно ввести URI «/admin». Див. Рис. 3.6.

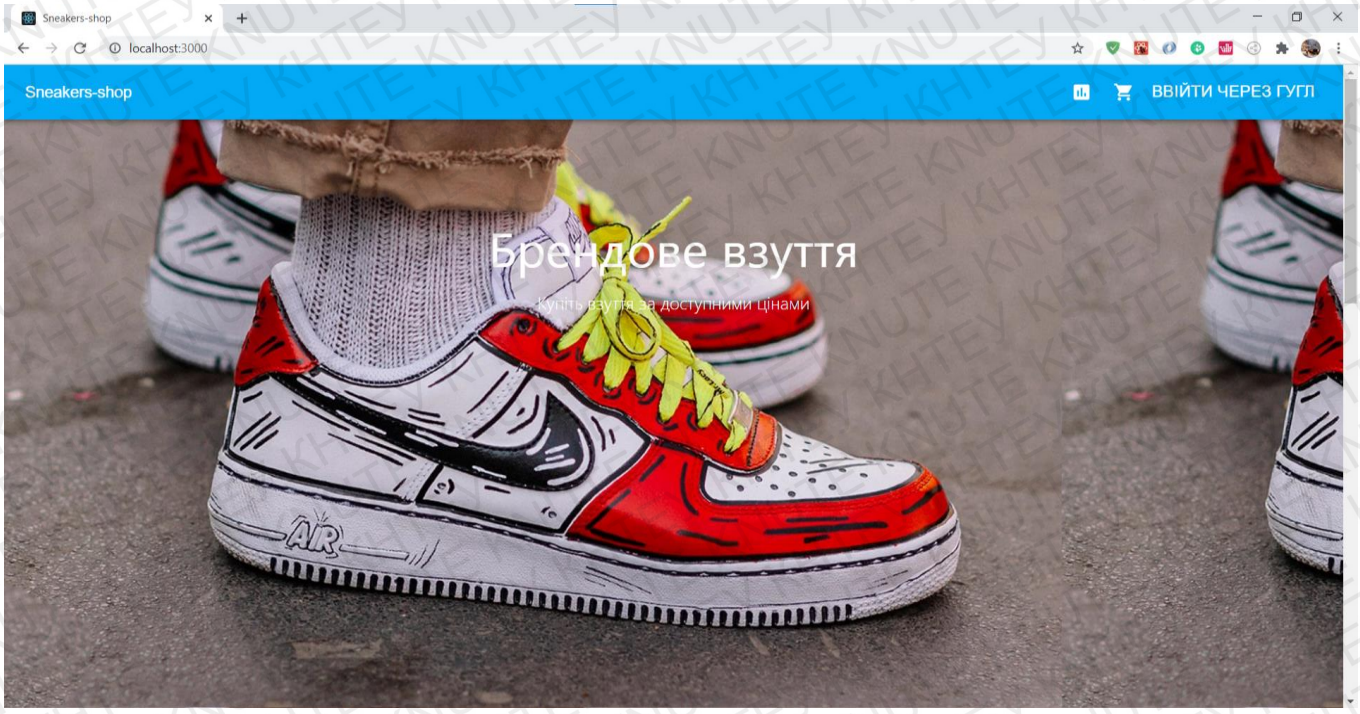


Рис. 3.6 Головна сторінка інтернет-магазину[Авторська розробка]

Для того, щоб авторизуватися необхідно натиснути кнопку «ввійти через гугл». Тут відбувається вхід через Google+ API. Далі вибираємо потрібний акаунт. Див. Рис. 3.7.

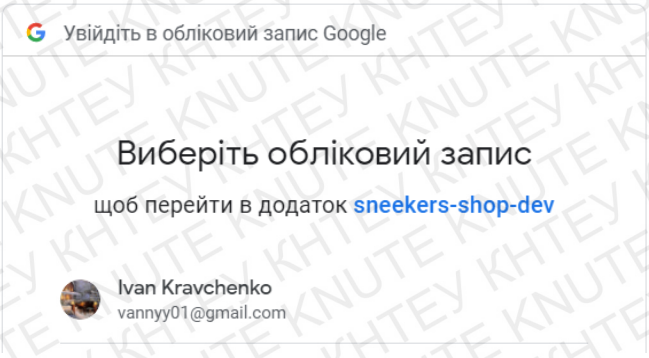


Рис. 3.7 Авторизація у системі[Авторська розробка]

Якщо у користувача немає ролі 20, його буде направлено на головну сторінку. У самій системі ми бачимо чотири модулі: склад, користувачі (клієнти), замовлення, оплата. Перейдемо до модуля складу. Навігація відбувається через бокову панель. Див. Рис. 3.8.

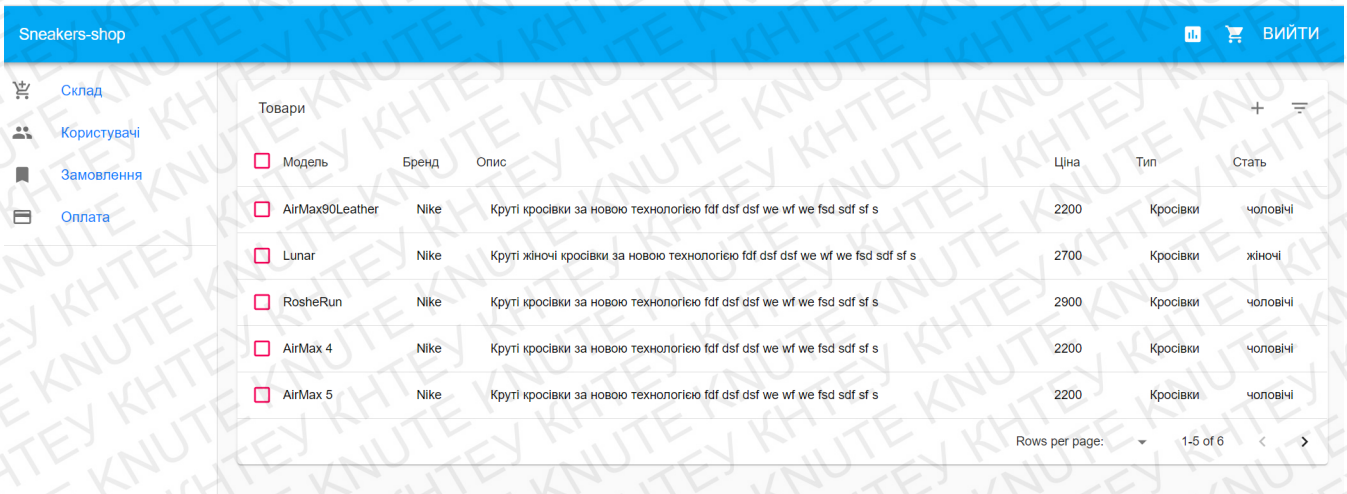


Рис. 3.8 Модуль керування товарами[Авторська розробка]

У модулі користувачів ми бачимо перелік користувачів. Ми можемо вибрати усіх або кількох користувачів, щоб їх видалити. Для редагування інформації про користувача потрібно вибрати його зі списку. Див. Рис. 3.9.

Рис 3.9 Модуль керування клієнтами[Авторська розробка]

Натиснувши кнопку «Створити новий запис» на панелі модулю користувачів можна додати нового користувача(клієнта) до системи. Статус користувача додаємо через поле «Роль», а до інших полів додана валідація. Див Рис.3.10.

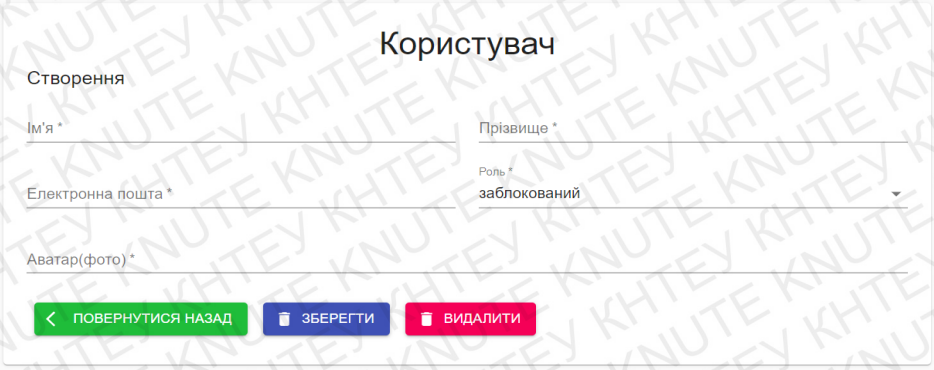


Рис 3.10 Форма створення користувача[Авторська розробка]

У формі товару ми бачимо повну інформацію про товар. Тут ми можемо внести зміни до товару. До полів форми додана валідація, це дає змогу уникнути небажаних типів даних та занадто короткої інформації. Див. Рис. 3.11.

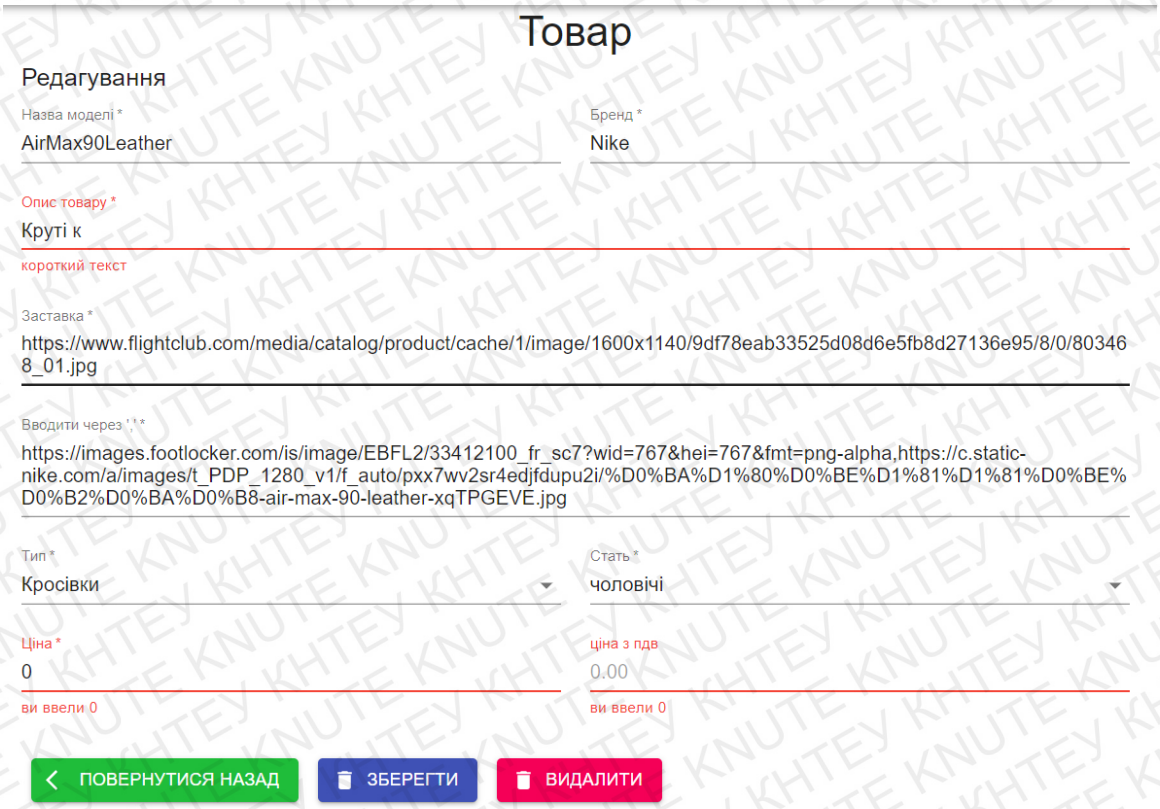


Рис 3.11 Форма редагування товару[Авторська розробка]

Коли все відповідає правилам, натискаємо «Зберегти Зміни». Див. Рис. 3.12.

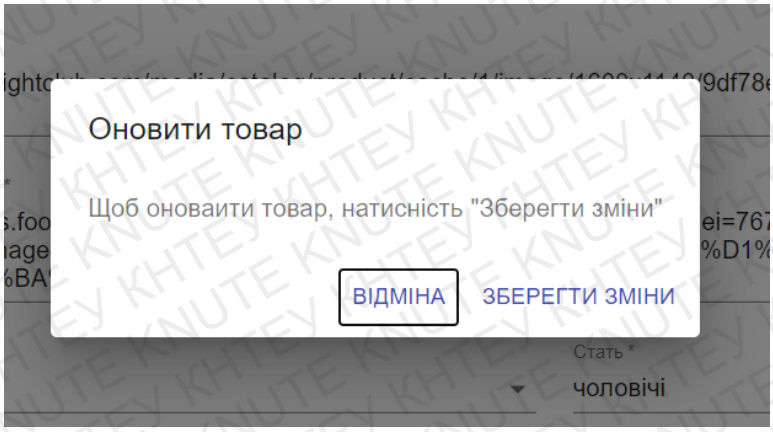


Рис 3.12 Підтвердження оновлення товару[Авторська розробка]

Так, як система виконує функцію оптимізації витрат підприємства та зменшення рутинної роботи працівників, вона призводить до позитивного економічного ефекту. Для наочного представлення ефекту приведемо приклад витрат підприємства, якби він був представлений оффлай та онлайн.

За рахунок переносу бізнесу в онлайн вдалося скорроти орендну площу зі 100 до 30 м^2, а також зменшити кількість працівників на 1 у пункті видачі. В обох випадках необіхдно купити техніку. Комп'ютер має термін амортизації у 5 років, а принтер у 7. Не залежно від способу ведення діяльності необхідно мати РРО. Розробка інтернет-магазину робиться власними силами, для його запуску необхідно взяти в оренду сервер. Якщо в оффлайн випадку витрати на рекламу йдуть на розвішування оголошень та роздачу флаєрів, то в онлайні ми купумо таргетовану рекламу за місцем розташування магазину. На випадок винекнення труднощів закладено 10000 гривень на техпідтримку. Див таблицю 3.1.

Таблиця 3.1

Витрати підприємтсва

Стаття витрат	Оффлайн-модель	Онлайн-модель	Витрати за рік оффлайн	Витрати за рік онлайн	Різниця
Оренда	100 м^2	30 м^2	180000	54000	126000
Робітники	3	2	504000	336000	168000
Комп'ютер/5	1	1	4000	4000	0

Лазерний принтер/7	1	1	700	700	0
РРО і каса	1	1	3600	3600	0
Розробка інтернет-магазину	1	1	0	0	0
Оренда сервера	1	1	2000	2000	0
Маркетинг	+	+	30000	30000	0
Техпідтримка	-	+	0	10000	10000
Всього			717300	425300	292000

Таким чином, ми продемонстрували в роботі функції покладені на систему управління підприємством електронної комерції. Було продемонстровано, що має робити адміністратор інтернет-магазину. Також, система може бути розширена, структура дозволяє мати більше 4 модулів. У підсумку запуск інтернет-магазину дає змогу зекономити 292000 гривень або 40,7%.

Висновки до розділу 3.

Таким чином, ми розглянули технології веб-програмування, що застосовуються у нашій системі управління. Це HTML, CSS, JavaScript(Node.js, React.js). Було визначено, що система буде реалізована на основі REST API, це дасть змогу розширювати систему без виникнення проблем для клієнта.

Реалізація системи, відбувається через розбиття проекту на дві частини. У першій, в корінній директорії, розміщується індексний файл сервера. Поряд з ним розміщуються директорії з файлами, які обслуговують бізнес-логіку на стороні сервера та роботу з базою даних MongoDB. Клієнтська частина розміщується у директорії client і відповідає за представлення даних, їхню валідацію та роботу інтерфейсу.

У результаті створено систему-управління, яка доступна за URI «/admin». У меню ми можемо побачити 4 модулі: склад, користувачі (клієнти), замовлення, оплата. В модулях ми можемо проводити CRUD-операції. Контроль доступу до системи відбувається через запровадження ролей у користувачів. У підсумку веб-

система дозволяє керувати підприємством будь-де, де є підключення до мережі Інтернет.

Також варто відзначити, що розроблена система дозволяє виконувати керування модулями незалежно від місця перебування, виконувати персоналізовані продажі, інформувати покупців, виконувати аналіз даних в реальному часі. Усе перелічене має позитивний вплив на динаміку продажів та управлінські рішення. Повний код проекту можна побачити за https://github.com/vannyy01/sneakers-shop_deploy [40], а робочу версію за адресою: https://safe-mountain-16797.herokuapp.com/ [36].

Економічний ефект запровадження системи управління дає змогу суттєво скоротити витрати (на 40,7%), особливо на оренду приміщень та найманих працівників. Також система дає додаткові можливості для збільшення продажів та спрощення роботи працівників.

ВИСНОВКИ

У магістерському дослідженні опрацьовано теоретичні напрацювання з розглядуваної проблеми і з’ясовано та уточнено сутність поняття «електронної комерції», під якою розуміємо виробництво, розповсюдження, маркетинг, продаж чи доставку товарів та послуг електронними засобами. Поділяємо також поглиблене визначення «електронної комерції» Азіатсько-Тихоокеанського економічного співробітництва («АТЕС»), що включає в нього і всю господарську діяльність, що проводиться за допомогою комбінації електронних комунікацій та технологій обробки інформації.

Серед принципів електронної комерції виокремлюємо такі: принцип лояльності, принцип «безоплатності», принцип експоненти, єдиного системного зв’язку, принцип стійкого зростання, принцип глобалізації.

Опрацьовано методологію аналізу бізнес-процесів (BPA) для розуміння стану здоров'я різних операцій у бізнесі для підвищення ефективності процесів. Цей спеціалізований метод у широкому контексті управління бізнес-процесами застосовували для аналізу того, чи відповідають поточні процеси своїм цілям. Здійснений аналіз використання бізнес-процесів допоміг визначити згубні елементи в операції та визначити, як подолати існуючі перешкоди. Під час здійснення аналізу бізнес-процесів визначили 4-кроковий план: визначили процес, здійснили збір інформації про процес, проаналізували «AS-IS» процес, розробили план «TO BE».

Моделювання бізнес-процесів (BPM) в управлінні бізнес-процесами та інженерії систем розглядали як діяльність, яка представляє процеси підприємства, щоб поточний процес міг бути проаналізованим, удосконаленим та автоматизованим. Для створення графічної схеми використали нотацію BPMN. Ця нотація є стандартом для моделювання бізнес-процесів, що надає графічну нотацію для визначення бізнес-процесу у вигляді «Діаграми бізнес-процесу» (Business Process Diagram, BPD). У підсумку була сформована модель, яка показує як відбуваються процеси на підприємстві електронної комерції.

Для створення інформаційної системи спочатку створили модель, де відображаються бізнес-процеси та інформаційні потоки. Інформаційна модель – це модель, що описує інформаційні процеси або містить інформацію про властивості і стан об'єктів, процесів, явищ.

На основі інформаційної моделі створили базу даних на СУБД MongoDB. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СУБД, функціональними і зручними у формуванні запитів[18]. Варто зазначити, що MongoDB – це нереляційна база даних. Відповідно ключові поля у різних колекціях не є зв’язаними. Така модель була обрана задля більш легкої масштабованості та більшої гнучкості.

У роботі розглянули стек, який використовується для розробки системи-управління і складається із JavaScript(Node.js, React.Js, Express), HTML і CSS. За основу додатку був вибраний підхід REST API для розширення можливостей управління. Система управління складається із серверної частини та клієнтської. Серверна відповідає за отримання та обробку даних через бізнес-логіку. Через клієнтську частину адміністратори здатні проводити CRUD-операції через використання React.js та AJAX-запитів.

Створені розробки дозволяють керувати підприємством електронної комерції через веб-інтерфейс, завдяки використанню серверної частини написаної на Node.js та express.js, де використовується організація доступу, взаємодія із СУБД MongoDB, робота з клієнтами через Google+ API та використання бізнес-логіки при обробці запитів. Ця система дозволяє слідувати за потребами підприємства електронної комерції, автоматизує управління, надає можливості приєднання до сторонніх сервісів та

Також було проаналізовано вплив створення системи-упраління на витрати підприємтсва. В підсумку визначено, що при переході в онлайн підриємтсво має змогу скоротити витрати на понад 40% за рахунок зменшення видатків на оренду приміщення та утримання штату працівників при додаткових витратах на хостинг та підтримку веб-сайту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України Про електронну комерцію. [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/675-19/paran221#n221

2. BPMN [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/BPMN

3. JavaScript [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/JavaScript

4. MongoDB [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/MongoDB

5. React.js Початок роботи даних [Електронний ресурс]. Режим доступу https://uk.reactjs.org/docs/getting-started.html

6. REST [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/REST

7. База даних [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/База_даних

8. Інформаційна модель. Алгоритми [Електронний ресурс]. Режим доступу: https://sites.google.com/site/informatica1kurs/informatika/lekciie/infor macijna-model-algoritmi

9. Інформаційні системи управління: сутність, види, функції, принипи побудови / Н.Г Георгіаді // Lviv Polytechnic National University Institutional Repository- 2008р. [Електронний ресурс]. Режим доступу: http://ena.lp.edu.ua

10. Кулак Н.В. Формування та принципи функціонування електронних систем торгівлі / Н.В. Кулак // Державне управління: удосконалення та розвиток. – 2013. – № 4 – С. 19 –26. [Електронний ресурс]. Режим доступу : http://www.dy.nayka.com.ua/?op=1&z=773

11. Гармідер Л.Д., Орлова А.В. Особливості розвитку вітчизняної електронної комерції. Європейський вектор економічного розвитку. 2015. № 1 (18) с. 58-65

12. Максимова Т. С. Використання електронної комерції роздрібними торговельними підприємствами / Т. С. Максимова, Д. В. Сорочан // Торгівля і ринок України: Тематичний збірник наукових праць. – 2010. – Випуск 29. – С. 273-279.

13. Маловічко С. В. Конвергентна система управління електронною торгівлею підприємств: Вісник Хмельницького національного університету № 4 Т. 1/ Хмельницький: ХНУ, 2015. с.112-116

14. Плескач В.Л. Проблеми розвитку електронної комерції в Україні / В.Л. Плескач, Т.Г. Затонацька, Л.В. Олексюк // Економіка України. – 2017. – № 11. – С. 73-84.

15. Пурський О. І. Функціональна модель Web-підприємства з мережею Інтернет-магазинів / О. І. Пурський, Д. П. Мазоха, І. О. Жарій // Проблеми економіки. - 2015. - № 2. - С. 166-171. [Електронний ресурс]. Режим доступу: http://nbuv.gov.ua/UJRN/Pekon_2015_2_25

16. Свидрук І.І. Формування систем управління в підприємствах електронної торгівлі : дис. на здобуття наукового ступеня канд. екон. наук : 08.00.04 / Свидрук І. І. Львів, 2007. С. 229

17. Система керування базами даних [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Система_керування_базами_даних

18. Сосновська О. О., Хамула О. Г. Сучасні проблеми управління підприємствами електронної торгівлі в Україні: Інфраструктура ринку/ Причорноморський Науково-дослідний Інститут Економіки та Інновацій, Одеса, 2018. – 209-213с.

19. Тардаскіна Т.М. Електронна комерція / Тардаскіна Т.М., Стрельчук Є.М., Терешко Ю.В. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 244 с.

20. Хамула О.О. Завдання та проблеми розвитку сфери електронної торгівлі в Україні як середовища функціонування системи реалізації книжкової продукції / Наукові записки : наук.-техн. зб. Львів : УАД, 2015. № 2 (51). С. 63–69.

21. Ховрак Інна Вікторівна Електронна комерція в Україні: Переваги та недоліки/ ЕКОНОМІКА, ФІНАНСИ, ПРАВО 4’2013 С.16-20

22. Maryam Mohsin. 10 Ecommerce Trends That You Need to Know in 2020 [Infographic] [Електронний ресурс] Режим доступу: https://www.oberlo.com/blog/ecommerce-trends

23. A Complete Breakdown for Business Process Analysis [Електронний ресурс] Режим доступу: https://kissflow.com/bpm/business-process-analysis/

24. Ainin, S. and Jaffar, N., E-Commerce Stimuli and Practices in Malaysia. PACIS Proceedings: Association for Information Systems AIS Electronic Library (AISeL). 2003. 201p.

25. Arie, S., Dadong W., & Caroline, B. Financial EDI over the Internet: a Case Study. Working Paper CITM-WP-1006: Fisher Center for Information Technology & Management, University of California in Berkeley. 1995. 123p.

26. Beginners Guide to Business Process Modeling [Електронний ресурс] Режим доступу: https://www.smartsheet.com/beginners-guide-business-process-modeling

27. Block, M., Yves, P. & Arie, S. On the Road of Electronic Commerce- a Business Value Framework,Gaining Competitive Advantage and Some Research Issues. Working Paper: Fisher Center for Information Technology and Management, University of California, Berkeley. 1996. 52p.

28. Business process modeling. [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/Business_process_modeling

29. Cascading Style Sheets. [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

30. Computer Engineering and Intelligent Systems: Emerging Trend of E-o Vol 2, No.5 [Електронний ресурс] Режим доступу: www.iiste.org, 2011 p.17-20

31. Draft: JSX Specification [Електронний ресурс] Режим доступу: https://facebook.github.io/jsx/

32. Express.js [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/Express.js

33. Mongoose. Getting Started [Електронний ресурс] Режим доступу: https://mongoosejs.com/docs/index.html

34. Node.js [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/Node.js

35. React (web framework) [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/React_(web_framework)

36. Sneakers-shop [Електронний ресурс] Режим доступу: https://safe-mountain-16797.herokuapp.com/

37. The MongoDB 4.4 Manual [Електронний ресурс] Режим доступу: https://docs.mongodb.com/manual/

38. Top 10 e-commerce Design Principles for a Successful Website [Електронний ресурс] Режим доступу: https://www.devsaran.com/blog/top-10-e-commerce-design-principles-successful-website

39. TypeScript [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/TypeScript

40. vannyy01/sneakers-shop_deploy [Електронний ресурс] Режим доступу: https://github.com/vannyy01/sneakers-shop_deploy

41. What You Need To Know About Node.js [Електронний ресурс] Режим доступу: https://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/

Додатки

Додаток А

Керуючі бізнес-процеси

Управління підприємством	
Вхід	інформація про всі бізнес-процеси, що відбуваються на підприємстві
Вихід	ефективна, керована діяльність підприємства
Ресурси	інформація про всі бізнес-процеси, що відбуваються на підприємстві, управлінський персонал, грошові кошти для оплати їхньої діяльності, елементи діяльності підприємства, якими необхідно керувати
Управлінський механізм	закони України та підзаконні акти ведення підприємницької діяльності, міжнародні законодавчі акти, Господарський та Податковий Кодекси України і так далі
Виконавець	генеральний директор
Керівник	генеральний директор
Власник	генеральний директор
Споживач	всі функціональні та лінійні служби, над якими здійснюється управління

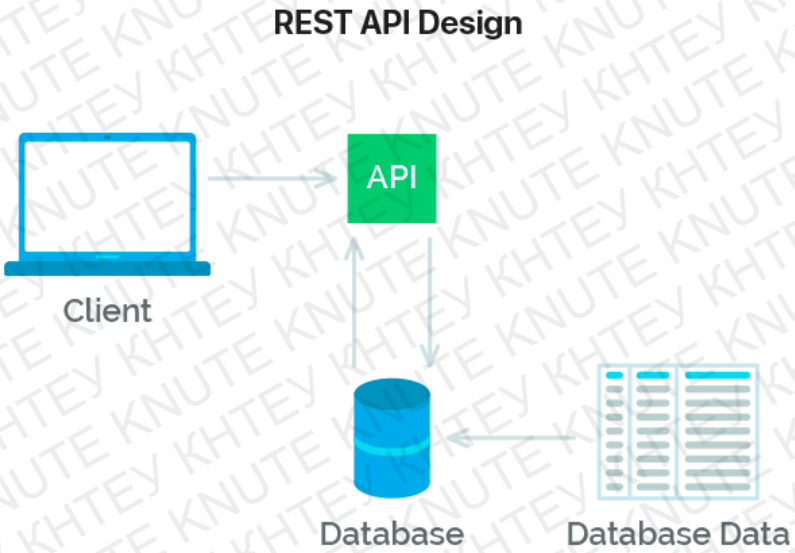
Стратегічне управління	
Вхід	інформація про всі бізнес-процеси, гроші та трудові ресурси
Вихід	координація, організація, аналіз, оптимізація, вибір стратегії
Ресурси	інформація про поточний стан роботи підприємства, звітна інформація минулих періодів, сучасні дослідження у сфері стратегічного управління, довідкові видання про кон'юнктуру ринку, державні нормативно-правові акти, які впливають на діяльність підприємства, ліцензії, думки споживачів про продукцію підприємства, тощо
Управлінський механізм	закони України та підзаконні акти ведення підприємницької діяльності, міжнародні законодавчі акти, Господарський та Податковий Кодекси України і так далі
Виконавець	генеральний директор
Керівник	генеральний директор
Власник	генеральний директор
Споживач	всі функціональні та лінійні служби, над якими здійснюється управління

Фінансове управління	
Вхід	інформація про фінансовий стан підприємства, гроші та трудові ресурси, бухгалтерські зобов'язання
Вихід	сформований кошторис видатків для забезпечення діяльності підприємства відповідно до обраної стратегії, оплачені бухгалтерські зобов'язання
Ресурси	інформація про стратегію підприємства, працівники фінансового відділу підприємства, грошові кошти на виплату заробітної плати працівникам фінансовому відділу, приміщення з обладнанням та робочими місцями, положення та стандарти бухгалтерського обліку та інші нормативно-законодавчі акти, працівники бухгалтерії (головний бухгалтер, бухгалтер)
Управлінський механізм	договори з кредиторами та інвесторами, політика ведення бухгалтерського та податкового обліку, закон України про бухгалтерський облік та нормативні документи Міністерства фінансів України, постанова Кабінету Міністрів України і Національного банку України, положення (стандарти)

	бухгалтерського обліку
Виконавець	працівники бухгалтерського відділу
Керівник	головний бухгалтер
Власник	головний бухгалтер
Споживач	всі функціональні та лінійні служби, над якими здійснюється управління, всі партнери підприємства, перед якими є бухгалтерські зобов'язання

Додаток Б

Схема роботи REST API



Додаток В

Лістинг коду авторизації та реєстрації passport.js

```
const mongoose = require('mongoose');
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const keys = require('../config/keys');
const User = mongoose.model('users');
passport.serializeUser((user, done) => {
  done(null, user.id)
});
passport.deserializeUser((id, done) => {
  User.findById(id).then(user =>
    done(null, user)
  );
});
passport.use(
  new GoogleStrategy({
    clientID: keys.googleClientID,
    clientSecret: keys.googleClientSecret,
    callbackURL: '/auth/google/callback',
    proxy: true
  },
  async (accessToken, refreshToken, profile, done) => {
    const existingUser = await User.findOne({googleID: profile.id});
    if (existingUser) {
      return done(null, existingUser);
    }
    const newUser = await new User({googleID: profile.id,
      email: profile.emails[0].value}).save();
    done(null, newUser);
  })
);
```

Додаток Г

Лістинг коду маршрутів для роботи з користувачами authRoutes.js

```
const passport =
require('passport');
const requireLogin =
require('../middlewares/requireLogin');
const mongoose =
require('mongoose');
const User =
mongoose.model('users');

module.exports = (app) => {
  app.get('/auth/google',

passport.authenticate('google',
    {
      scope: ['profile', 'email']
    }) );
  app.get('/auth/google/callback',

passport.authenticate('google'),
    (req, res) => {
      res.redirect('/');
    }
  );
  app.get('/api/logout', (req, res)
=> {
    req.logout();
    res.redirect('/');
  });
  app.get('/api/current_user', (req,
res) => {
    if(req.user)
      res.send(req.user);
    else
      res.send('');
  });
  app.get('/api/users',
requireLogin, async (req, res) => {
    await User.find({}, function
(err, users) {
      if (err)

res.status(404).send('Cannot get the
users list!');
      else
        res.send(users);
    }).select(['googleID', '__id',
'email', 'role']).limit(10);
  });
};
```

Додаток Д

Лістинг коду моделі товарів СУБД MongoDB

```
const mongoose = require('mongoose');
const {Schema} = mongoose;
const Size = require('./Size');

/**
 * Base schema for other instances
 * @param props
 *                                  @returns
{module:mongoose.Schema}
 */
const commoditySchema = new
Schema({
   title: {type: String, unique: true},
   brand: String,
   description: String,
   price: Number,
   mainImage: String,
   images: [String],
   sizes: [Size],
   type: String,
   sex: String,
});
mongoose.model('commodities',
commoditySchema);
module.exports = commoditySchema;
const mongoose = require('mongoose');
const {Schema} = mongoose;

const sizeSchema = new Schema({
   sizeValue: Number,
   count: Number
});

module.exports = sizeSchema;
```

Додаток Е

Лістинг коду моделі замовлень СУБД MongoDB

```
const mongoose = require('mongoose');
const {Schema} = mongoose;
const commoditySchema = require('./Commodity');

const orderSchema = new Schema({
  goods: [commoditySchema],
  count: Number,
  sum: Number,
  _user: { type: Schema.Types.ObjectId, ref: 'User'},
  dateOrder: Date,
  dateSent: Date
});
mongoose.model('orders', orderSchema);
```

Додаток Ж

Лістинг коду моделі користувачів СУБД MongoDB

```
const mongoose = require('mongoose');
const {Schema} = mongoose;

const userSchema = new Schema({
  googleID: {type: String, unique: true},
  email: {type: String, unique: true},
  role: {type: Number, default: 10}
});

mongoose.model('users', userSchema);
```

Додаток К

Лістнинг коду Goods.tsx

```
import * as React from 'react'; import {connect} from "react-redux";
import {fetchGoods} from "../../actions"; import GridView from '../GridView';
import {ShoeInterface} from "../../actions/types";
interface PropsInterface {
  goods: ShoeInterface[] | [],
  fetchGoods: (to: number) => void
}
interface HeadCell {
  disablePadding: boolean;
  id: keyof ShoeInterface;
  label: string;
  numeric: boolean;
}
const headCells: HeadCell[] = [
  {id: 'title', numeric: false, disablePadding: true, label: 'Модель'},
  {id: 'brand', numeric: false, disablePadding: true, label: 'Бренд'},
  {id: 'description', numeric: false, disablePadding: true, label: 'Опис'},
  {id: 'price', numeric: true, disablePadding: false, label: 'Ціна'},
  {id: 'type', numeric: false, disablePadding: true, label: 'Тип'},
  {id: 'sex', numeric: false, disablePadding: true, label: 'Стать'},
];
class Goods extends React.Component<PropsInterface, any> {
  public componentDidMount() {
    this.props.fetchGoods(6);
  }
  public render() {
    if (this.props.goods.length > 0 && Array.isArray(this.props.goods)) {
      for (const item of this.props.goods) {
```

```
            delete item.images;
            delete item.mainImage;
            delete item.sizes;
        }
        return (
            <React.Fragment>
<GridView idField="_id" route='/admin/goods/edit' data={this.props.goods}
headCells={headCells}
title="Товари"/>
            </React.Fragment>
)}
    return <div>Loading...</div> }}
const mapStateToProps = ({goods}: any) => ({goods});
export default connect(mapStateToProps, {fetchGoods})(Goods);
```

Додаток Л

Лістинг коду форми редагування товарів

```
import * as React from 'react';
import Grid from '@material-ui/core/Grid';
import MenuItem from '@material-ui/core/MenuItem';
import TextField from '@material-ui/core/TextField';
import Paper, {PaperProps} from '@material-ui/core/Paper';
import Typography from '@material-ui/core/Typography';
import {createStyles, Theme} from "@material-ui/core";
import withStyles from "@material-ui/core/styles/withStyles";
import {RouteComponentProps} from 'react-router-dom';
import Button from "@material-ui/core/Button";
import DeleteIcon from '@material-ui/icons/Delete';
import ArrowBack from '@material-ui/icons/ArrowBackIos';
import {connect} from 'react-redux';
import {fetchGoodByID, updateGood} from "../../actions";
import {ShoeInterface} from "../../actions/types";
import Snackbar from "@material-ui/core/Snackbar";
import MuiAlert, {AlertProps} from '@material-ui/lab/Alert';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogContentText from '@material-ui/core/DialogContentText';
import DialogTitle from '@material-ui/core/DialogTitle';
import Draggable from 'react-draggable';

function Alert(props: AlertProps) {
   return <MuiAlert elevation={6} variant="filled" {...props} />;
}
```

```
function PaperComponent(props: PaperProps) {
  return (
    <Draggable handle="#draggable-dialog-title"
cancel={'[class*="MuiDialogContent-root"]'}>
      <Paper {...props} />
    </Draggable>
  );
}
const styles = (theme: Theme) => createStyles({
  alert: {
    marginTop: theme.spacing(7)
  },
  button: {
    margin: theme.spacing(1)
  },
  paper: {
    marginBottom: theme.spacing(3),
    marginTop: theme.spacing(3),
    padding: theme.spacing(2),
    [theme.breakpoints.up(600 + theme.spacing(3) * 2)]: {
      marginBottom: theme.spacing(6),
      marginTop: theme.spacing(6),
      padding: theme.spacing(3),
    },
    width: "60em"
  }
});
interface PathParams {
  commID: string
}
```

```
interface PropsType extends RouteComponentProps<PathParams> {
  classes: { alert: string, button: string, paper: string },
  fetchGoodByID: (id: string) => void,
  good: ShoeInterface,
  updateGood: (good: ShoeInterface | {}, callback: () => void) => void
}
interface StateType {
  showAlert: boolean,
  showDialog: boolean,
  formErrors: { title: string, brand: string, description: string, mainImage: string, images:
string, type: string, sex: string, price: string },
  formValid: boolean
  good: ShoeInterface | {},
  isValid: { titleValid: boolean, brandValid: boolean, descriptionValid: boolean,
mainImageValid: boolean, imagesValid: boolean, typeValid: boolean, sexValid: boolean,
priceValid: boolean },
}
const sexes = [
  {
    label: 'чоловічі',
    value: 'чоловічі',
  },
  {
    label: 'жіночі',
    value: 'жіночі',
  }
];
const types = [
  {
```

```
      label: "Кросівки",
      value: "Кросівки"
    },
    {
      label: "В'єтнамки",
      value: "В'єтнамки",
    },
    {
      label: "Туфлі",
      value: "Туфлі",
    }
];
class AddressForm extends React.Component<PropsType, StateType> {
  constructor(props: PropsType) {
    super(props);
    this.state = {
      showAlert: false,
      showDialog: false,
      formErrors: {
        title: '',
        brand: '',
        description: '',
        mainImage: '',
        images: '',
        type: '',
        sex: '',
        price: ''
      },
      formValid: true,
      good: {},
```

```
            isValid: {
                titleValid: true,
                brandValid: true,
                descriptionValid: true,
                mainImageValid: true,
                imagesValid: true,
                typeValid: true,
                sexValid: true,
                priceValid: true
            },
        }
    }
    public componentDidMount() {
        this.props.fetchGoodByID(this.props.match.params.commID);
        this.setState({good: this.props.good})
    }
    public componentDidUpdate(prevProps: Readonly<PropsType>, prevState:
Readonly<StateType>, snapshot?: any): void {
        if (JSON.stringify(prevProps.good) !== JSON.stringify(this.props.good)) {
            this.setState({good: this.props.good})
        }
    }
    public render() {
        if (this.state.good.hasOwnProperty('_id')) {
            // @ts-ignore
            const {title, brand, description, mainImage, images, type, sex, price} =
this.state.good;
            return <Paper className={this.props.classes.paper}>
                <Typography component="h1" variant="h4" align="center">
                    Товар
```

```
</Typography>
<React.Fragment>
  <Typography variant="h6" gutterBottom={true}>
    Редагування
  </Typography>
  <form noValidate={true} onSubmit={event => this.handleSubmit(event)}>
    <Grid container={true} spacing={3}>
      <Grid item={true} xs={12} sm={6}>
        <TextField
          required={true}
          id="title"
          name="title"
          label="Назва моделі"
          fullWidth={true}
          autoComplete="title-name"
          value={title}
          onChange={event => this.handleChange(event, 'title')}
          helperText={this.state.formErrors.title}
          error={this.state.formErrors.title.length > 0}
        />
      </Grid>
      <Grid item={true} xs={12} sm={6}>
        <TextField
          required={true}
          id="brand"
          name="brand"
          label="Бренд"
          fullWidth={true}
          autoComplete="brand-name"
          value={brand}
```

```
        onChange={event => this.handleChange(event, 'brand')}
        helperText={this.state.formErrors.brand}
        error={this.state.formErrors.brand.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12}>
      <TextField
        required={true}
        id="description"
        name="description"
        multiline={true}
        rowsMax={4}
        label="Опис товару"
        fullWidth={true}
        autoComplete="description-name"
        value={description}
        onChange={event => this.handleChange(event, 'description')}
        helperText={this.state.formErrors.description}
        error={this.state.formErrors.description.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12}>
      <TextField
        required={true}
        id="mainImage"
        name="mainImage"
        label="Заставка"
        multiline={true}
        fullWidth={true}
        autoComplete="mainImage-name"
```

```
        value={mainImage}
        onChange={event => this.handleChange(event, 'mainImage')}
        helperText={this.state.formErrors.mainImage}
        error={this.state.formErrors.mainImage.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12}>
      <TextField
        required={true}
        id="images"
        name="images"
        label="Вводити через ','"
        fullWidth={true}
        multiline={true}
        autoComplete="mainImage-name"
        value={images}
        onChange={event => this.handleChange(event, 'images')}
        helperText={this.state.formErrors.images}
        error={this.state.formErrors.images.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12} sm={6}>
      <TextField
        required={true}
        id="type"
        name="type"
        label="Тип"
        fullWidth={true}
        autoComplete="type-name"
        select={true}
```

```
        value={type}
        onChange={event => this.handleChange(event, 'type')}
        helperText={this.state.formErrors.type}
        error={this.state.formErrors.type.length > 0}
      >
        {types.map((option) => (
          <MenuItem key={option.value} value={option.value}>
            {option.label}
          </MenuItem>
        ))}
      </TextField>
    </Grid>
    <Grid item={true} xs={12} sm={6}>
      <TextField required={true}
          id="sex" name="sex" label="Стать" fullWidth={true}
          autoComplete="sex-name" select={true} value={sex}
          helperText={this.state.formErrors.sex}
          error={this.state.formErrors.sex.length > 0}
      >
        {sexes.map((option) => (
          <MenuItem key={option.value} value={option.value}>
            {option.label}
          </MenuItem>
        ))}
      </TextField>
    </Grid>
    <Grid item={true} xs={12} sm={6}>
      <TextField
        required={true}
        id="price"
```

```
        name="price"
        label="Ціна"
        fullWidth={true}
        autoComplete="shipping postal-code"
        value={price}
        onChange={event => this.handleChange(event, 'price')}
        helperText={this.state.formErrors.price}
        error={this.state.formErrors.price.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12} sm={6}>
      <TextField
        required={false}
        disabled={true}
        id="price_vat"
        name="price_vat"
        label="ціна з пдв"
        fullWidth={true}
        autoComplete="price_vat-name"
        value={(price * 1.2).toFixed(2)}
        helperText={this.state.formErrors.price}
        error={this.state.formErrors.price.length > 0}
      />
    </Grid>
    <Grid item={true} xs={12}>
      <Button
        variant="contained"
        color="inherit"
        style={{backgroundColor: "#1fbd3a", color: "#fff"}}
        startIcon={<ArrowBack/>}
```

```
            className={this.props.classes.button}
            onClick={event => this.handleComeBack()}
          >
            Повернутися назад
          </Button>
          <Button
            type="submit"
            variant="contained"
            color="primary"
            startIcon={<DeleteIcon/>}
            className={this.props.classes.button}
          >
            Зберегти
          </Button>
          <Button
            variant="contained"
            color="secondary"
            startIcon={<DeleteIcon/>}
            className={this.props.classes.button}
          >
            Видалити
          </Button>
        </Grid>
      </Grid>
    </form>
    <Dialog
      open={this.state.showDialog}
      onClose={event => this.handleClose("cancel")}
      PaperComponent={PaperComponent}
      aria-labelledby="draggable-dialog-title"
```

```
            >
              <DialogTitle style={{cursor: 'move'}} id="draggable-dialog-title">
                Subscribe
              </DialogTitle>
              <DialogContent>
                <DialogContentText>
                  Щоб оноваити товар, натисність "Зберегти зміни"
                </DialogContentText>
              </DialogContent>
              <DialogActions>
                <Button autoFocus={true} name="cancel" onClick={event =>
this.handleClose("cancel")}
                        color="primary">
                  Відміна
                </Button>
                <Button name="save" onClick={event => this.handleSave()}
color="primary">
                  Зберегти зміни
                </Button>
              </DialogActions>
            </Dialog>
            <Snackbar anchorOrigin={{vertical: 'top', horizontal: 'center'}}
open={this.state.showAlert}
                      autoHideDuration={6000} className={this.props.classes.alert}
                      onClose={event => this.handleClose("alert")}>
              <Alert onClose={event => this.handleClose("alert")}
severity="error">Виправте помилки!</Alert>
            </Snackbar>
          </React.Fragment>
        </Paper>
```

```
    }
    return <div>Loading...</div>;
  }

  protected handleComeBack = (): void => {
    this.props.history.goBack();
  };

  protected handleChange = (event: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>, label: keyof ShoeInterface) => {
    const newState: ShoeInterface | {} = this.state.good;

    const {name, value} = event.target;
    if (label === "images") {
      newState[label] = value.split(",");
      this.setState({good: newState}, () => {
        this.validateField(name, newState[label]);
      });
      return;
    }
    if (label === "price") {
      newState[label] = value !== '' ? Number.parseInt(value) : 0;
      this.setState({good: newState}, () => {
        this.validateField(name, newState[label]);
      });
      return;
    }
    newState[label] = value;
    this.setState({good: newState}, () => {
      this.validateField(name, newState[label]);
```

```
    });
  };

  protected handleSubmit = (event: React.FormEvent<HTMLFormElement>): void =>
{
    event.preventDefault();
    if (!this.state.formValid) {
      this.setState({showAlert: true});
      return;
    }
    this.setState({showDialog: true});
  };
  protected handleClose = (name: "cancel" | "alert" | "save"): void => {
    switch (name) {
      case "cancel":
        this.setState({showDialog: false});
        break;
      case "alert":
        this.setState({showAlert: false});
        break;
      default:
        alert('Збережено');
        console.log(this.state.good);
        break;
    }
  };
  protected handleSave = (): void => {
    this.props.updateGood(this.state.good, () =>
      this.props.history.push('/admin/goods')
    );
```

```
};
protected validateField = (fieldName: string, value: any): void => {
  const fieldValidationErrors = this.state.formErrors;
  const {isValid} = this.state;
  switch (fieldName) {
    case 'title':
      if (typeof value !== 'string') {
        fieldValidationErrors.title = 'некоректний текст';
        isValid.titleValid = false;
        break;
      }
      if (value.length < 2) {
        fieldValidationErrors.title = 'занадто коротка назва';
        isValid.titleValid = false;
        break;
      }
      fieldValidationErrors.title = '';
      isValid.titleValid = true;
      break;
    case 'type':
      if (typeof value !== 'string') {
        fieldValidationErrors.type = 'некоректний тип';
        isValid.typeValid = false;
      }
      break;
    case 'brand' :
      if (typeof value !== 'string') {
        fieldValidationErrors.brand = 'некоректний текст';
        isValid.brandValid = false;
        break;
```

```
    }
    if (value.length < 2) {
      fieldValidationErrors.brand = 'занадто коротка назва';
      isValid.brandValid = false;
      break;
    }
    fieldValidationErrors.brand = '';
    isValid.brandValid = true;
    break;
  case 'description':
    if (typeof value !== 'string') {
      fieldValidationErrors.description = 'некоректний текст';
      isValid.descriptionValid = false;
      break;
    }
    if (value.length < 15) {
      fieldValidationErrors.description = 'короткий текст';
      isValid.descriptionValid = false;
      break;
    }
    fieldValidationErrors.description = '';
    isValid.descriptionValid = true;
    break;
  case 'sex':
    if (typeof value !== 'string') {
      fieldValidationErrors.sex = 'некоректний текст';
      isValid.sexValid = false;
      break;
    }
    fieldValidationErrors.sex = '';
```

```
        isValid.sexValid = true;
        break;
    case 'mainImage':
        if (typeof value !== 'string') {
            fieldValidationErrors.mainImage = 'некоректний текст';
            isValid.mainImageValid = false;
            break;
        }
        if (value.length < 6) {
            fieldValidationErrors.mainImage = 'занадто коротка назва';
            isValid.mainImageValid = false;
            break;
        }
        fieldValidationErrors.mainImage = '';
        isValid.mainImageValid = true;
        break;
    case 'images':
        if (!Array.isArray(value)) {
            fieldValidationErrors.images = 'некоректний тип даних';
            isValid.imagesValid = false;
            break;
        }
        if (value.join('').length < 6) {
            fieldValidationErrors.images = 'додайте зображення';
            isValid.imagesValid = false;
            break;
        }
        fieldValidationErrors.images = '';
        isValid.imagesValid = true;
        break;
```

```
      case 'price':
        if (Number.parseInt(value) === 0 || value === '') {
          fieldValidationErrors.price = 'ви ввели 0';
          isValid.priceValid = false;
          break;
        }
        if (isNaN(Number.parseInt(value))) {
          fieldValidationErrors.price = 'некоректний тип даних';
          isValid.priceValid = false;
          break;
        }
        fieldValidationErrors.price = '';
        isValid.priceValid = true;
        break;
      default:
        break;
    }
    this.setState({
      formErrors: fieldValidationErrors,
      isValid
    }, this.validateForm);
  };

  protected validateForm = (): void => {
    let isValid = true;
    for (const key in this.state.isValid) {
      if (!this.state.isValid[key]) {
        isValid = false;
      }
    }
```

```
        this.setState({formValid: isValid});
    }
}
const mapStateToProps = ({goods}: { goods: ShoeInterface }) => ({good: goods});
export default connect(mapStateToProps, {fetchGoodByID,
updateGood})(withStyles(styles)(AddressForm));
```