РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: веб-додаток для реалізації послуг підприємства з виготовлення та продажу пакувальної продукції.

Мета кваліфікаційної роботи: розробити зручний веб-додаток для підприємства, що виготовляє та реалізує пакувальну продукцію, основними функціями якого є надання інформації про послуги та можливості онлайн замовлення різного виду продукції.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в підвищенні ефективності роботи підприємства за рахунок автоматизації ведення онлайн замовлень, скорочення витрат часу потенційного покупця на пошук і замовлення певного виду продукції, що реалізується наведеним підприємством.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, в зв'язку з необхідністю впровадження сучасних інформаційних технологій в торгівлі та їх розповсюдження в мережі Інтернет для забезпечення успішної діяльності підприємства та підвищення інтересу на пропоновані товари.

Список ключових слів:ВЕБ-ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, ВИРОБНИЦТВО, УПАКОВКА, БАЗА ДАНИХ, ПРОЄКТУВАННЯ, ПРОГРАМУВАННЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: a web application for the implementation of enterprise services for the manufacture and sale of packaging products.

The purpose of the qualification work: to develop a convenient web application for the company that manufactures and sells packaging products, the main functions of which are to provide information about services and opportunities to order different types of products online.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is to increase the efficiency of the enterprise by automating online ordering, reducing the time spent by potential buyers to find and order a certain type of product sold by the company.

The relevance of the topic of qualification work is determined by the high demand for such developments, due to the need to introduce modern information technologies in trade and their dissemination on the Internet to ensure the success of the enterprise and increase interest in the products offered.

List of keywords: WEB APPLICATION, INFORMATION SYSTEM, PRODUCTION, PACKAGING, DATABASE, DESIGN, PROGRAMMING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – бази даних;

ОС – операційна система;

ПК – персональний комп'ютер;

СКБД – система керування базами даних;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

HTML – Hyper Text Markup Language, мова розмітки гіпертекстових документів;

MVC – Model-View-Controller, Модель-Вид-Контролер.

ЗМІСТ

ВСТУП

Метою кваліфікаційної роботи бакалавра є розробка зручного веб-додатку для підприємства, що виготовляє та реалізує пакувальну продукцію, основними функціями якого є надання інформації про послуги та можливості онлайн замовлення різного виду виготовленої продукції.

Дана робота запропонована для впровадження на реальному підприємстві «ПакСтандарт», яке займається виробництвом різної пакувальної продукції, зокрема виробляє коробки різних розмірів і форм.

Для розробки кожної конкретної коробки необхідно створювати технологічну карту - описовий документ, який містить в собі всю необхідну інформацію про виріб - розміри, точне креслення, опис матеріалів, марки картону, кольру картону тощо. Даний документ створюється організацією і затверджується із замовником. Після цього починається виробництво виробів.

Спочатку замовлення дана організація брала тільки по телефону або особисто, просто записуючи на папір. Даний програмний продукт (і особистий кабінет зокрема) розроблений для впорядкування інформації про замовлення, а також для спрощення і автоматизації процесу замовлення товару.

Для виконання завдання даної роботи необхідно виконати наступні етапи:

1. Дослідити предметну область, в якій буде проводитися розробка. Навести опис процесу роботи підприємства, для якого розробляється програмне забезпечення.
2. Виконати аналіз та опис використовуваних інструментів розробки: основні концепції та аналіз їх переваг та недоліків, порівняння з іншими існуючими аналогами.
3. Визначити необхідні функції програмного забезпечення та детально описати їх роботу.
4. Описати деталі проектного рішення.
5. Протестувати та продемонструвати результати роботи.

Практичне значення роботи полягає в підвищенні ефективності роботи підприємства за рахунок автоматизації ведення онлайн замовлень, скорочення витрат часу потенційного покупця на пошук і замовлення певного виду продукції, що реалізується наведеним підприємством.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, в зв'язку з необхідністю впровадження сучасних інформаційних технологій в торгівлі та їх розповсюдження в мережі Інтернет для забезпечення успішної діяльності підприємства та підвищення інтересу на пропоновані товари.

Дана робота запропонована для впровадження на підприємстві «ПакСтандарт», але після деяких перетворень зможе бути запропонована також іншим підприємствам, метою яких є осучаснення бізнесу завдяки створенню веб-додатку для забезпечення електронної торгівлі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Актуальність розробки

У сучасному інформаційному суспільстві кожна стабільна торгова компанія повинна мати власний інтернет-магазин, який забезпечить інформаційну підтримку існуючого бізнесу і збільшить рівень продажів. За допомогою електронної торгівлі фірми вирішують такі завдання, як представлення компанії в мережі Інтернет, розширення потенційної аудиторії споживачів, підтримка бренду, підвищення впізнаваності, інформування громадськості та ін.

Електронна торгівля - це торгівля через мережу за допомогою комп'ютерів покупця і продавця товару. Предметом електронної торгівлі може бути будь-який товар, послуга, нерухомість, банківський продукт і т.п. Цінність електронної торгівлі для покупців полягає в тому, що вона значно економить час покупця на пошук і покупку потрібного йому товару. Для продавця цінність електронної торгівлі полягає в потенційній можливості охопити своєю торгівлею незліченну кількість покупців.

Розвиток мережі Інтернет призвело до різкого зростання цієї технології торгівлі серед усіх торговельних фірм і громадян. Інтернет стимулював розвиток електронної торгівлі на рівні окремого господарюючого суб'єкта. Малі підприємства та громадяни отримали можливість вести свої комерційні угоди та інші операції в оперативному електронному режимі - в режимі реального часу онлайн. Режим онлайн - це режим роботи банкомату, коли обмін інформацією між банком і центром відбувається постійно і всі транзакції за рахунком виконуються в реальному масштабі часу, тобто банківські проводки здійснюються «день в день». Інтернет дозволяє знизити витрати на проведення торгових та інших угод, тому користувачі Інтернет стали переводити свої дані в

цифрову форму.

Оперування цифровою інформацією в комп'ютерних мережах розширює можливості бізнесу. Будь-яку інформацію можна уявити і зберегти у вигляді ланцюжка біт. Біт - це одиниця кількості інформації в двійковій системі числення. А основною одиницею в сучасних ЕОМ є байт. Електронна торгівля створює нову форму організації торгових підприємств - віртуальні магазини - і постійно під впливом конкуренції пропонує нові товари і послуги для реалізації в віртуальному магазині.

Віртуальний магазин - це реалізоване в мережі Інтернет представництво шляхом створення веб-сервера для продажу товарів і пов'язаних з ними послуг іншим користувачам мережі Інтернет. Віртуальний магазин - це співтовариство територіально роз'єднаних співробітників магазина (продавців, касирів) і покупців, які можуть спілкуватися і обмінюватися інформацією виключно через електронні засоби зв'язку при повній (або мінімальній) відсутності особистого прямого контакту. Віртуальний магазин - це торгова площадка в Інтернеті. Він працює за технологією, схожою з роботою традиційного магазину.

Покупець товару є користувачем мережі Інтернет. Він входить через комп'ютер на сервер віртуального магазину, тобто в сервер продавця товару. Потім покупець переглядає на своєму комп'ютері сторінки сервера з метою отримання інформації про товар. При наявності в магазині великого асортименту товарів їх розміщують на окремих сторінках сервера як спеціалізовані і однорядні товари. Це розміщення дозволяє покупцеві повну і чітку інформацію про кожен товар (зовнішній вигляд, якість, призначення і ціна тощо). Перш ніж зробити остаточний вибір покупки товару, покупець може оглянути товар з усіх боків, проконсультуватися з продавцем через комп'ютер або по телефону, попросити продавця продемонструвати товар і дії і т.п. Все це відбувається на різних територіях: покупець отримує інформацію сидячи у себе вдома на дивані, продавець описує і демонструє якість товару та інші його характеристики, не виходячи з магазину.

Вибравши потрібний йому товар, покупець переходить (за вказаним URL) на іншу сторінку сервера і робить заявку, тобто, замовлення на товар. Замовлення на товар виробляється електронною поштою у формі відправки директору або продавцеві віртуального магазину цифрового запиту на затвердження заявки і подальше оформлення. До запиту можна приєднати веб-сторінку з описом обраного товару.

1.1.2. Дослідження предметної галузі

Для виконання кваліфікаційної роботи було проведено дослідження предметної галузі об’єкта розробки.

Даний веб-додаток розробляється для підприємства «ПакСтандарт», що виробляє різну пакувальну продукцію (коробки різних форм і розмірів).

Аналіз роботи підприємства показав наступні етапи в процесі продажу пакувальної продукції:

1. У фірму звертається клієнт (зазвичай таке звернення відбувається за допомогою телефонного дзвінка) і розмовляє з менеджерами підприємства про замовлення пакувальної продукції певного типу. Цей тип може бути стандартним (які мають свій номер по ГОСТу), або клієнт може замовити новий тип коробки спеціальної форми. Також клієнт може вказати свої побажання з приводу матеріалу картону і гофри, кольору коробки, малюнка на ній і ін.

2. Підприємство приймає замовлення, збирає дані про клієнта та виготовляє технологічну карту виробу. Клієнти оформляють замовлення на юридичні особи. Таких юридичних осіб може бути кілька у одного і того ж клієнта.

3. Технологічна карта надсилається клієнту і затверджується чи ні (в цьому випадку вона редагується). Також клієнтові оголошується вартість і строки виготовлення.

4. Після затвердження починається виробництво продукції.

5. Після виробництва, готова продукція відправляється клієнту службами доставки або за допомогою самовивозу.

1.1.3. Функціональні опис веб-додатку

В принципі роботи веб-додатку підприємства «ПакСтандарт», що виробляє різну пакувальну продукцію, є можливість замовлення товару за необхідними параметрами на сайті, його виготовлення на підприємстві та доставці продукту в потрібне місце призначення.

Для того, щоб проект торгівлі через Інтернет став успішний, крім навичок ведення бізнесу, наявності стартового капіталу, домовленостей з постачальниками, а також системи обробки замовлень, необхідно:

— написання продуктивної системи керування контентом, розрахованого на модульність і модернізацію;

— привабливий і зручний інтерфейс як для користувацької частини, так і частини адміністрування;

— оптимізація сайту у пошукових системах для збільшення кількості клієнтів, рекламні кампанії.

Типовий варіант інтернет-магазину складається із наступних функціональних частин: каталог товарів, пошукова система, віртуальна корзина користувача, реєстраційна форма, форма відправлення замовлення (рис. 1.1).

Рис. 1.1. Типовий алгоритм замовлення товару в інтенет-магазині

Каталог являє собою складну і багаторівневу структуру даних, яка повинна простим і зрозумілим способом виконувати упорядкування товарів. Простіше за все такий каталог представити у вигляді дерева об'єктів, верхній рівень якого складається зі списку розділів. Розділи можуть містити підрозділи або посилання на конкретний товар і т.д. Таке впорядкування просто необхідно для зручного і швидкого пошуку і замовлення товарів.

Пошукова система є обов'язковим елементом динамічного каталогу й реалізується на стороні сервера. Незважаючи на те, що каталог забезпечує упорядкування та угруповання даних, пошукова система дає користувачеві можливість швидкого пошуку інформації, що особливо важливо в тому випадку, коли каталог являє собою досить розгалужену структуру даних з великою кількістю розділів, підрозділів і товарів, користувач погано уявляє в якому розділі може перебувати цікавить його товар і чи є він в каталозі взагалі.

Пошукова система в деяких випадках дозволяє значно скоротити кількість переходів між сторінками каталогу для доступу до інформації, що цікавить.

Особливість реалізації пошуку в Інтернеті полягає в тому, що тут відбувається вибірка всіх записів, які задовольняють умовам запиту (даний механізм пошуку я називаю пошук з надлишком). У разі великої вибірки даних вивід результатів пошуку здійснюється посторінково для того, щоб відвідувачам не доводилося довго чекати завантаження всієї вибірки, яка може включати в себе сотні, тисячі і більше записів. Як правило, відвідувачі не переглядають всі сторінки вибірки, обмежуючись двома або трьома. Тому даний механізм пошуку в багатьох випадках працює вкрай повільно і неефективно. Однак він дозволяє здійснити вибірку однакових товарів від різних постачальників, порівняти їх параметри між собою і вибрати оптимальний варіант.

Користувацький кошик представляє собою деякий масив даних, який служить для зберігання замовленого користувачем товару.

Реєстраційна форма для введення персональних даних користувачів. Надалі ця інформація використовується для їх ідентифікації між сеансами роботи з інтернет-магазином. Дана інформація може зберігатися як на стороні сервера, так і на стороні клієнта.

Форма відправки замовлення служить для введення контактної інформації замовника і відправки її та замовлення на електронну скриньку організації.

1.2. Призначення розробки та галузь застосування

Розвиток комп'ютерних інформаційних систем і телекомунікаційних технологій привели до формування нового виду економічної діяльності - електронної комерції, або електронного бізнесу. Електронний бізнес є особливою формою бізнесу, що реалізується в значній мірі за допомогою комп'ютеризації процесів виробництва, продажу і розподілу товарів і послуг.

Електронні магазини істотно зменшують витрати виробника, заощадивши на утриманні звичайного магазину, розширюють ринки збуту і розширюють можливість покупця - купувати будь-який товар в будь-який час в будь-якій країні, в будь-якому місті, в будь-який час доби, в будь-який час року. Це дає електронним магазинам значні переваги перед звичайними магазинами.

Основними операціями, що відбуваються в електронному магазині це вибір товару, його замовлення, обробка та виконання.

Процес оформлення замовлення на підприємстві ще і досі часто відбувається «на папірцях», які часто губляться. Таким чином для підприємства можуть мати місце випадки втрати клієнтів. Тому для поліпшення якості процесу роботи і підвищення зручності оформлення замовлення, як для потенційних замовників так і для менеджерів підприємства, було прийнято рішення про створення зручного веб-додатку для підприємства «ПакСтандарт»,, який не тільки би представляв підприємство в мережі Інтернет, а й дозволяв користувачам оформляти замовлення в електронному вигляді, запобігаючи цим ряд проблем «паперових замовлень», в тому числі і їх втрату.

Для цього рішення варто було б всі дані зберігати у базі, звідки менеджери могли б контролювати їх, а замовники могли б відстежувати стан своїх заявок прямо на сайті.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка веб-додатку засобами фреймворку Vue.js для підприємства з виготовлення пакувальної продукції» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою кваліфікаційної роботи бакалавра є розробка зручного веб-додатку для підприємства, що виготовляє та реалізує пакувальну продукцію, основними функціями якого є надання інформації про послуги та можливості онлайн замовлення різного виду виготовленої продукції.

Дана робота запропонована для впровадження на реальному підприємстві «ПакСтандарт», яке займається виробництвом різної пакувальної продукції, зокрема виробляє коробки різних розмірів і форм.

Для виконання даної роботи поставлені наступні завдання:

1. Вивчити основи програмування за технологією прогресивного JavaScript фреймворку для розробки призначених для користувача інтерфейсів - Vue.js.

2. Дослідити предметну область для розробки веб-додатку з використанням даного фреймворку.

3. Виконати проектування і реалізацію відповідного веб-додатку.

4. Розробити дизайн користувальницького інтерфейсу.

5. Розробити програмне забезпечення відповідно до результатів аналізу предметної області і вимог.

Основним завданням веб-додатку є:

1. Надання користувачам можливості перегляду інформації про підприємство.

2. Забезпечення покупцю можливості вибору потрібного товару або замовлення його виготовлення за необхідними параметрами. Замовлення пакувальної продукції можливо певного типу. Цей тип може бути стандартним (які мають свій номер по ГОСТу), або клієнт може замовити новий тип коробки спеціальної форми. Також клієнт може вказати свої побажання з приводу матеріалу картону і гофри, кольору коробки, малюнка на ній і ін.

3. Оформлення заявки на його виготовлення та покупку має здійснюватись з використанням сучасних онлайн технологій.

4. Адміністрування сайту і управління замовленнями.
5. Ведення обліку клієнтів і редагування бази даних системи.

Дана інформаційна система дозволить вирішити задачу вибору та виготовлення продукції, перегляд короткого опису інформації про товар, онлайн замовлення товарів, також існує можливість додавати, редагувати і видаляти інформацію про матеріали, продукцію, користувачів, оперативно обробляти замовлення і отримувати звіт про ведення замовлень.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблена система повинна забезпечувати наступні функції:

– користувач, потрапляючи на сайт, повинен отримувати чітку інформацію про те, які товари пропонує підприємство, як він зможе оплатити замовлення, які умови і терміни виробництва, доставки і т.д .;

– необхідно створити для користувача зручний і швидкий пошук необхідного йому товару, адже не всі мають необмежений доступ в інтернет, і багато оплачують його по годинах. Та й виснажливий перегляд каталогів мало кому до душі;

– всі товари (пакувальна продукція), пропоновані на сайті, повинні бути або в наявності, або доступними для виробництва зі зазначених покупцем матеріалів.

Розроблена база даних системи повинна містити в собі дані про матеріали, продукцію, замовлення, а також коротку інформацію про замовників, визначати ролі та можливі дії для певних груп користувачів.

У режимі користувача дана інформаційна система повинна забезпечувати:

– реєстрацію або авторизацію користувача;
– вивід інформації про товари та послуги;
– пошук товару та матеріалів;
– можливість онлайн замовлення товарів за власними технічними

характеристиками;

У режимі менеджера дана інформаційна система повинна забезпечувати:

– перегляд бази замовлень.
– можливість управління замовленнями.
– перегляд звіту про продажі;
– адміністрування бази даних;
– можливість редагування всіх даних про товари, користувачів, і т.д

1.5.2. Вимоги до інформаційної безпеки

Для надійної роботи системи необхідно:

1. Використовувати ліцензійне програмне забезпечення на сервері.
2. Здійснювати захист від вірусів на сервері.
3. Здійснювати захист від несанкціонованого доступу.
4. Застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні.
5. Здійснювати контроль даних, що вводяться користувачами.
6. Автоматично завершувати сеанс роботи з користувачем у разі тривалої перерви його активності.

1.5.3. Вимоги до складу та параметрів технічних засобів

Інтерфейс веб-додатку повинен бути розрахований:

– на різні розділювачі здатності монітору, як мінімум 1024x768 пікселів;

– на різні версії браузерів, в тому числі дуже застарілих, наприклад Internet Explorer 6;

– основна частина інтерфейсу не повинна потребувати установки розширень типа Adobe Flash чи Microsoft Silverlight.

Треба враховувати той факт, що не у всіх користувачів може бути швидке інтернет-з'єднання чи дешевий (безлімітний) тарифний план, тому елементи інтерфейсу, такі як файли зображень, бібліотеки сценаріїв повинні бути оптимізовані, або від їх використання краще відмовитись.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування веб-додатку необхідна наявність наступного встановленого програмного забезпечення:

- операційна система Windows XP/ Vista / 7;
- MySQL 5.5.
- веб-браузер.

Для роботи з розробленою системою необхідний один з перерахованих веб-браузерів:

- IE 5.5 і вище;
- Firefox 0.8 і вище;
- Safari 1.2.4 і вище;
- Netscape 7.1 і вище;
- Mozilla 1.4 і вище;
- Opera 7 і вище.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Розроблена система забезпечує наступні функції:

– користувач, потрапляючи на сайт, отримує чітку інформацію про те, які товари пропонує підприємство, як він зможе оплатити замовлення, які умови і терміни виробництва, доставки і т.д .;

– створено зручний і швидкий пошук необхідного товару;

– всі товари (пакувальна продукція), пропоновані на сайті, або є в наявності, або доступні для виробництва зі зазначених матеріалів.

Розроблена база даних системи містить в собі дані про матеріали, продукцію, замовлення, а також коротку інформацію про замовників, визначає ролі та можливі дії для певних груп користувачів.

Розроблена система дозволяє вирішити такі проблеми як:

– вивід бази даних наданих товарів і послуг;

– автоматизація процесу підбору необхідного виду продукції;

– можливість онлайн реалізації та обробки замовлень;

– вивід списку замовлень;

– зберігання інформації про товари, послуги, користувачів, замовленнях;

– формування звітів про продажам;

– можливість редагування бази даних.

2.2. Опис застосованих математичних методів

Математичні методи для проектування та реалізації даного веб-додатку не використовуються.

2.3. Опис використаних технологій та мов програмування

2.3.1. Інструменти розробки інтерфейсу користувача

Для розробки призначеного для користувача інтерфейсу були використані такі інструменти як мову розмітки гіпертексту html, таблиці стилів css і мову програмування JavaScript, зокрема його фреймворк Vue.js.

2.3.1.1. Скриптова мова JavaScript

JavaScript - об'єктно-орієнтована скриптова мова програмування. Є діалектом мови ECMAScript. JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

На JavaScript зробили вплив багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легкою для використання непрограммістами. Через використання в веб-розробці. JavaScript став найпопулярнішою у світі мовою програмування.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але реалізоване в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними об'єктно-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, анонімні функції, замикання - що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript в порівнянні з мовою Сі має корінні відмінності:

– об'єкти, з можливістю інтроспекції;
– функції як об'єкти першого класу;
– автоматичне приведення типів;
– автоматичне прибирання сміття;

– анонімні функції.

JavaScript використовується в клієнтській частині веб-додатків клієнт-серверних програм, в якому клієнтом виступає браузер, а сервером - веб-сервер, що мають розподілену між сервером і клієнтом логіку. Обмін інформацією у веб-додатках відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки являються поміжплаформенними сервісами.

На сьогоднішній день підтримку JavaScript забезпечують сучасні версії всіх найбільш часто використовуваних браузерів. В Internet Explorer, Mozilla Firefox, Safari, Google Chrome, Opera є повна підтримка третьої редакції ECMA-262. При цьому в Mozilla Firefox зроблена спроба здійснення підтримки четвертої редакції специфікації, а першим браузером, у якому з'явилася неповна підтримка специфікації 3.1, з'явився Internet Explorer 8

Синтаксис мови JavaScript багато в чому нагадує синтаксис Сі і Java, семантично ж мова набагато ближче до Self, Smalltalk або навіть Ліспу. У JavaScript:

— всі ідентифікатори регістру;

— в назвах змінних можна використовувати літери, підкреслення, символ долара, арабські цифри;

— назви змінних не можуть починатися з цифри.

Для оформлення однорядкових коментарів використовуються / /, багатострокові і однострокові коментарі починаються з / * і закінчуються * /.

Структурно JavaScript можна представити у вигляді об'єднання трьох чітко помітних одна від одної частин:

— ядро (ECMAScript);

— об'єктна модель браузера (Browser Object Model або BOM (de));

— об'єктна модель документа (Document Object Model або DOM).

Якщо розглядати JavaScript у відмінних від браузера оточеннях, то об'єктна модель браузера і об'єктна модель документа можуть не підтримуватися.

Об'єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежної від мови інтерфейсу документа. На противагу цьому ряд авторів знаходять BOM і DOM тісно взаємопов'язаними.

Ядро ECMAScript не є браузерною мовою і насправді в ньому не визначаються методи введення і виведення інформації. Це скоріше основа для побудови скриптових мов. Специфікація ECMAScript описує типи даних, інструкції, ключові і зарезервовані слова, оператори, об'єкти, регулярні вирази, не обмежуючи авторів похідних мов в розширенні їх новими складовими.

Об'єктна модель браузера - браузероспеціфічна частина мови, що є прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - керування вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом BOM. Об'єктна модель браузера на даний момент не стандартизована, проте специфікація знаходиться в розробці WHATWG та W3C.

Крім управління вікнами, в рамках об'єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей:

- управління фреймами;
- підтримка затримки у виконанні коду та зациклення з затримкою;
- системні діалоги;
- управління адресою відкритої сторінки;
- управління інформацією про браузері;
- управління інформацією про параметри монітора;
- обмежене управління історією перегляду сторінок;
- підтримка роботи з HTTP cookie.

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно з документом DOM можна поставити у відповідність дерево об'єктів, що володіють рядом властивостей, які дозволяють виробляти з них різні маніпуляції:

— отримання вузлів;

— зміна вузлів;

— зміна зв'язків між вузлами;

— видалення вузлів.

Розташування всередині сторінки. Щоб додати JavaScript-коду на сторінку, можна використовувати теги <script> </ script>.

Розташування всередині тегу. Специфікація HTML описує набір атрибутів, використовуваних для завдання обробників подій.

Відділення від розмітки. У наведеному прикладі при натисненні на посилання функція confirm ('Ви впевнені?'); Викликає модальне вікно з написом «Ви впевнені?», А return false; блокує перехід за посиланням. Зрозуміло, цей код буде працювати тільки якщо в браузері є і включена підтримка JavaScript, інакше перехід за посиланням відбудеться без попередження.

Винесення в окремий файл. Є й третя можливість підключення JavaScript - написати скрипт в окремому файлі, а потім підключити його за допомогою конструкції

<script type="text/javascript" src="http://Шлях до файлу, у якому знаходиться сценарій"></script>

Атрибути тегу script. Тег script, широко використовуваний для підключення до сторінки JavaScript, має кілька атрибутів.

— обов'язковий атрибут type для вказівки MIME-типу вмісту;

— необов'язковий атрибут src, що приймає в якості значення адреса до файлу зі скриптом;

— необов'язковий атрибут charset, використовуваний разом з src для вказівки використовуваної кодування зовнішнього файлу;

— необов'язковий атрибут defer, який використовується для того, щоб показати, що скрипт не генерує ніякого вмісту (що означає, зокрема, те, що в цьому скрипті відсутній виклик document.write ()).

При цьому атрибут language (language = "JavaScript"), незважаючи на його активне використання, відноситься до не рекомендованим (deprecated), відсутній в DTD, тому вважається некоректним.

2.3.1.2. Мова каскадних таблиць стилів CSS

CSS - технологія опису зовнішнього вигляду документа, написаного мовою розмітки. Створення сучасного інтерфейсу для програмного забезпечення Інтернет неможливе без застосування цієї технології розмітці.

Переважно використовується як засіб оформлення веб-сторінок у форматі HTML і XHTML, але може застосовуватися з будь-якими видами документів у форматі XML, включаючи SVG і XUL.

CSS для відображення сторінки можуть бути взята з різних джерел:

1. Стилі автора (інформація надана автором сторінки) у вигляді:

— зовнішні таблиці стилів, тобто окремого файлу. css, на який робиться посилання в документі;

— вбудованих стилів - блоків CSS всередині самого HTML-документа;

— Inline-стилів, коли в HTML-документі інформація стилю для одного елемента вказаний в налаштуваннях style.

2. Користувальницькі стилі. Локальний CSS-файл, вказаний користувачем в настройках браузера, перевизначають авторські стилі, і вживаний до всіх документах.

Стилі складаються зі списку правил. Кожне правило, у свою чергу, складається з одного або більше селектор, розділених комами, і блоку визначень. Блок визначення оточеного фігурними дужками, і складається з набору властивостей і їх значень.

Стандарт CSS визначає пріоритети, у порядку яких застосовуються правила стилів, якщо для якогось елементу підходять властивості декількох правил одночасно (або, в рідкісних випадках, в одному правилі є однойменні

властивості). Це називається "каскадом", в якому для властивостей розраховуються пріоритети або "ваги", що робить результати передбачуваними.

Пріоритети розраховуються таким чином (від більшого до меншого):

1. властивість задано за допомогою !Importan.

2. Стиль прописаний прямо в тегу.

3. Кількість ідентифікаторів (# id) в селекторі (чим більше, тим більше пріоритет).

4. Кількість класів (. class) і псевдоклассов (: pseudoclass) в селекторі.

5. Кількість імен тегів в селекторі.

Крім того, має значення відносний порядок розташування властивостей - властивість, вказане пізніше, має пріоритет.

CSS-верстка. До появи CSS оформлення веб-сторінок здійснювалося безпосередньо усередині вмісту документа. Проте з появою CSS стало можливим принципове розділення змісту і представлення документа. За рахунок цього нововведення стало можливим легке застосування єдиного стилю оформлення для маси схожих документів, а також швидка зміна цього оформлення.

Переваги:

1. Декілька дизайнів сторінки для різних пристроїв перегляду. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на КПК і стільниковому телефоні меню буде слідувати за вмістом.

2. Зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. У цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і можуть бути закешувані.

3. Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл.

4. Додаткові можливості оформлення. Наприклад, за допомогою CSS-розмітки можна зробити блок тексту, який решта тексту буде обтікати (наприклад для меню) або зробити так, щоб меню було завжди видно при прокручуванні сторінки.

Недоліки:

Різного відображення верстки в різних браузерах (особливо застарілих), які по різному інтерпретують одні й ті ж дані CSS.

Часто зустрічається необхідність на практиці виправляти не тільки один CSS-файл, але і теги HTML і код PHP, які складним і ненаглядним способом пов'язані з селектори CSS, що іноді зводить нанівець простоту застосування єдиних файлів стилів і значно подовжує час редагування та тестування.

CSS Framework, (також Web design framework) - це заздалегідь підготовлена css-бібліотека, створена для спрощення роботи верстальника, швидкості розробки і виключення максимально можливого числа помилок верстки (проблеми сумісності різних версій браузерів і т.д.). Так само як і бібліотеки скриптових мов програмування, CSS-фреймверку, зазвичай мають вид зовнішнього .css-файлу, "підключаються" до проекту (додаються в заголовок веб-сторінки), дозволяючи не досвідченому в тонкощах верстки програмісту або дизайнерові правильно створити xhtml-макет .

2.3.1.3. Фреймворк Vue.js

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

Основною концепцією Vue є компоненти. Ця абстракція дозволяє збирати великі програми з маленьких «шматочків». Вони являють собою придатні до повторного використання об'єкти. Майже будь-який інтерфейс можна уявити як дерево компонентів (рис. 2.1).

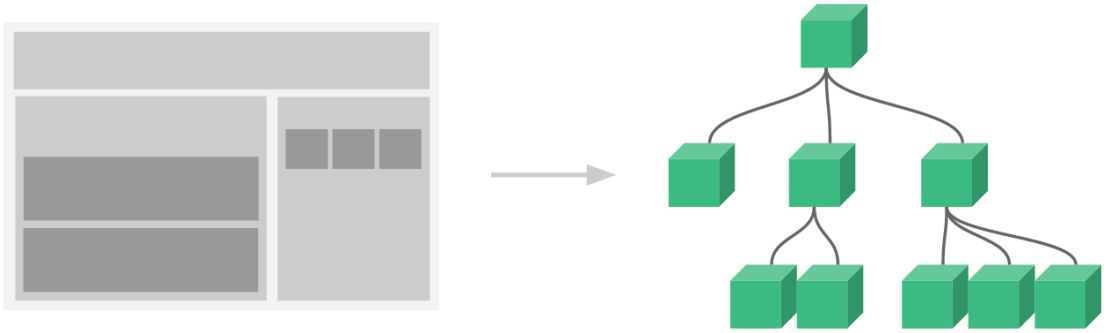

Рис. 2.1. Приклад дерева компонентів на сайті.

У Vue компонент - це, по суті, екземпляр Vue з попередньо встановленими опціями.

Vue.js створювався з оглядкою на кращі практики існуючих технологій. З React.js команда Vue запозичила ідею віртуального DOM. Цей підхід виключає пряму взаємодію з вузлами інтерфейсу. Початкова робота ведеться з його легковагій віртуальної копією (virtual DOM). І тільки після цього зміни застосовуються до реальних вузлів інтерфейсу. Паралельно відбувається порівняння реального DOM дерева і його віртуальної копії. Таким чином, виявляється різниця і перемальовується тільки те, що зазнало змін.

З Angular Vue.js запозичив двостороннє зв'язування (two-way data binding) - це створення двостороннього зв'язку між елементами сторінки і моделлю. Змінюючи дані на рівні уявлення, змінюється модель, а змінюючи дані на рівні моделі, змінюється уявлення.

Ядро Vue.js, подібно React, містить лише необхідний функціонал для роботи з інтерфейсом. Тому воно компактно, легко інтегрується з іншими технологіями, в тому числі з jQuery і навіть може використовуватися замість нього (для розробки простих інтерфейсів).

За замовчуванням, Vue використовує стандартні html, js і css для створення своїх компонентів. Однак легко можна підключити й інші технології (наприклад, препроцесори sass, less). Для цього потрібно всього лише встановити відповідні модулі node і додати атрибут до відповідного розділу однофайловий компонента.

Відносно особливостей розробки на даному фреймворку, відмінною рисою є розміщення коду всього компонента в одному файлі. Коли люди пишуть в Vue, вони використовують стиль «компонент в одному файлі». Це файл з суфіксом .vue, що складається з 3 частин (css, html і js) для кожного компонента.

Це зручне поєднання технологій. Досить зручно зберігати інформацію про один компонент в одному файлі. До того ж, у такого стилю є побічний ефект - програмісти намагаються підтримувати стисліcть компонентів. Якщо js, css та html компонента займають занадто багато рядків, то це означає, що можливо необхідно створити більше модулів.

Зручним доповненням є те, що стилі компонента можна інкапсулювати, додавши атрибут scoped. Цей спосіб стилізації компонентів уявлення в багатьох випадках зручніше, ніж підходи написання css в js в інших великих фреймворками.

Переваги Vue.js:

1. Бібліотека проста і функціональна. Для того щоб розібратися в ній, потрібен мінімальний багаж знань.

2. Вимоги до стека відсутні, тому Vue.js можна використовувати на будь-якому проекті.

3. Фреймворк легкий. Це економить час завантаження сторінок і надає інші переваги.

4. Висока швидкість розробки. Завдяки використанню будь-яких шаблонів і доступності документації, більшість виникаючих проблем вирішуються досить швидко.

5. Можливість знайти і підключити до проекту практично будь-якого

розробника, хто хоч трохи знайомий з фронтенд розробкою. Низький поріг входження дозволяє працювати з фреймворком, як фронтендщікам, так і бекендщікам.

6. Vue дозволяє створювати функціональні додатки, які відповідають усім сучасним стандартам, при мінімальному підключенні нових ресурсів і, власне, дешевше.

Недоліки Vue.js:

1. Робота над станом додатку відбувається "під капотом". У масштабних проектах це може вилитися в неочевидність роботи компонентів, що часто створює проблеми з налагодженням і ефективної розробкою.
2. Компонентний підхід у Vue.js не такий гнучкий, як в React.
3. Система рендеринга React більш функціональна. Вона надає більше можливостей для налагодження.
4. Шаблонізація React значно гнучкіша (JS vs DSL).

Проаналізувавши всі плюси і мінуси фреймворка, можна зробити висновок, що Vue.js реалізує всі сучасні підходи до розробки web-UI і є легким в освоєнні, гнучким і високо інтегрованим зі сторонніми технологіями фреймворком.

2.3.2. Засоби розробки СКБД

Інформаційна система розробленого веб-додатку містить в собі базу даних, яка реалізована за допомогою СКБД MySQL 5.5.34

MySQL - вільна система управління базами даних (СКБД). MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненої Sun Microsystems, що здійснює розробку і підтримку програми. Поширюється під GNU General Public License або під власною комерційною ліцензією. Крім цього розробники створюють функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

Максимальні розміри таблиць:

- MYSQL 3.22 : до 4 Гб;
- MYSQL 3.23+ : До 8 мільйонів терабайт. (2 ^ 63).

Розмір таблиці обмежений її типом. У загальному випадку тип MYISAM обмежений граничним розміром файлу у файловій системі операційної системи. Наприклад в NTFS цей розмір теоретично може бути до 32 ексабайт. У разі INNODB одна таблиця може зберігатися в декількох файлах, що представляють єдиний табличний простір. Розмір останнього може досягати 64 терабайт.

MySQL портована на велику кількість платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS / 2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003, WinCE, Windows Vista і Windows 7. Існує також порт MySQL до OpenVMS. Важливо відзначити, що на офіційному сайті СУБД для вільного завантаження надаються не тільки вихідні коди, а й відкомпілювалися і оптимізовані під конкретні операційні системи готові виконувані модулі СУБД MySQL.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис структури веб-додатку

Програмний комплекс містить велику кількість елементів, модулів, допоміжних бібліотек, сценаріїв. Загально програмний комплекс можна поділити на три основні частини:

1. Частина для користувачів (клієнтів) – безпосередньо сайт, в якому, власне, користувач переглядає товар, читає інформацію та робить замовлення.

2. Адміністративна частина, до якої є доступ тільки працівникам сайту (адміністратор та менеджер), в цій частині обробляються замовлення, додається та редагується доступний в магазинах товар, який представлений у користувацькій частині, та редагується деякі основні параметри сайту.

3. База даних, у якій зберігаються параметри сайту, база товарів, клієнтів та замовлень, з базою даних взаємодіють як частина для користувачів, так і адміністративна частина.

Користувацька частина – це інтерфейс сайту для користувачів. При заході на веб-сайт компанії запускається файл index.html, структура якого показана на рис. 2.2.

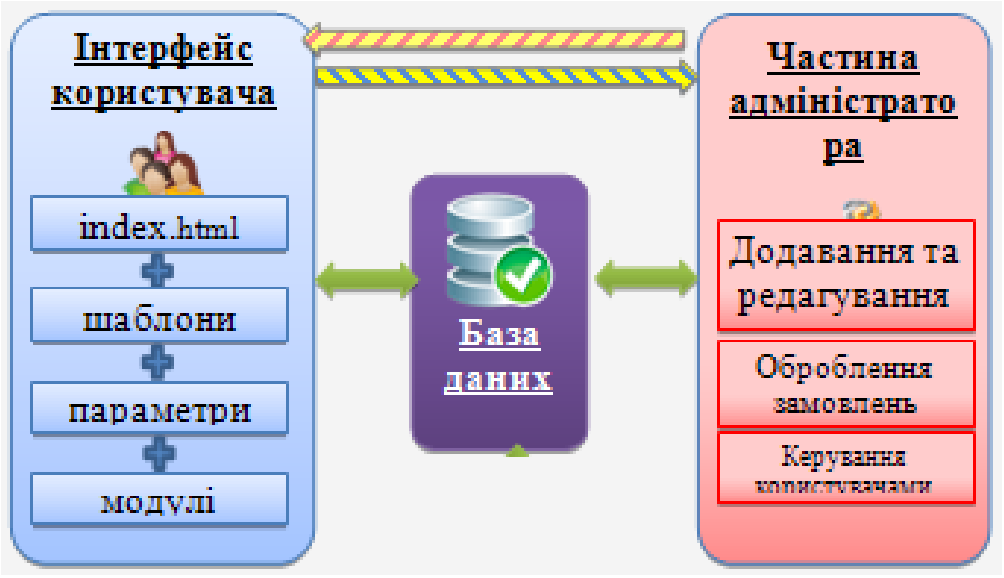

Рис. 2.2. Загальна структура інтернет-ресурсу

Файл index.html містить код, який спершу ініціює з’єднання з базою даних та завантажує з неї параметри сайту. Далі йде частина коду, яка завантажує файли функцій і робить їх доступними для всіх сторінок сайту. Також йде завантаження файлів дизайну, які відповідають за інтерфейс, ці файли знаходяться за шляхом /design/style.css, а вони, в свою чергу, завантажують необхідні графічні зображення, параметри кольорів, шрифтів, фону та параметри розташування елементів на сторінці. Також завантажуються шаблони сторінок та файли допоміжних модулів, які виводять модулі зліва та з права від основної частини.

Система керування контентом будується за схемою Модель-Вид-Контролер (MVC, Model-View-Controller). Модель представляє систему, вигляд відповідає за відображення моделі, а контролер обробляє дані, що поступають від користувача.

Описання файлу index.html надано на рис. 2.3.

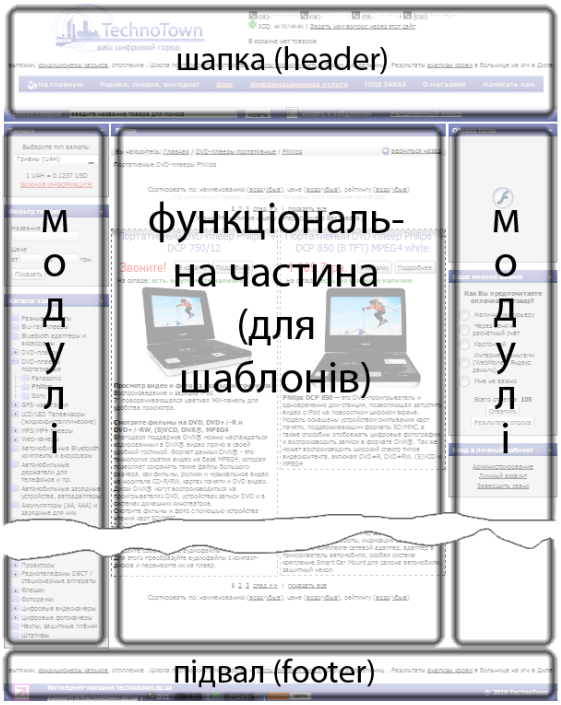

Рис. 2.3. Загальна структура файлу файл index.html

2.4.2. Функціональне проєктування системи

Організація «ПакСтандарт» займається виробництвом різної пакувальної продукції, зокрема виробляє коробки різних розмірів і форм.

Для розробки кожної конкретної коробки необхідно створювати технологічну карту - описовий документ, який містить в собі всю необхідну інформацію про виріб - розміри, точне креслення, опис матеріалів, марки картону, квітів картону, і т.д. Даний документ створюється організацією і затверджується із замовником. Після цього починається виробництво виробів.

Спочатку замовлення дана організація брала тільки по телефону або особисто, просто записуючи на папір. Даний програмний продукт (і особистий кабінет зокрема) був розроблений для впорядкування інформації про замовлення, а також для спрощення і автоматизації процесу замовлення товару.

При розробці даної системи враховані наступні характеристики:

— користувачеві надається повна інформації про товари, що доступні для покупки та виготовлення в заданих параметрах;

— користувачеві надається можливість онлайн замовлення товарів;

— адміністратор має можливість додавати, редагувати і видаляти певні види товарів;

— можливість перегляду, статусу та обліку замовлень.

Головне меню веб-додатку складається з наступних пунктів: Головна, Новини, Прайси, Контакти та Особистий кабінет.

Основними розділами особистого кабінету є такі сторінки: «Оформити замовлення», «Історія замовлень», «Довідник виробів», «Архів виробів», «Мої компанії», «Особисті дані».

Структурна схема системи наведена на рис. 2.4.

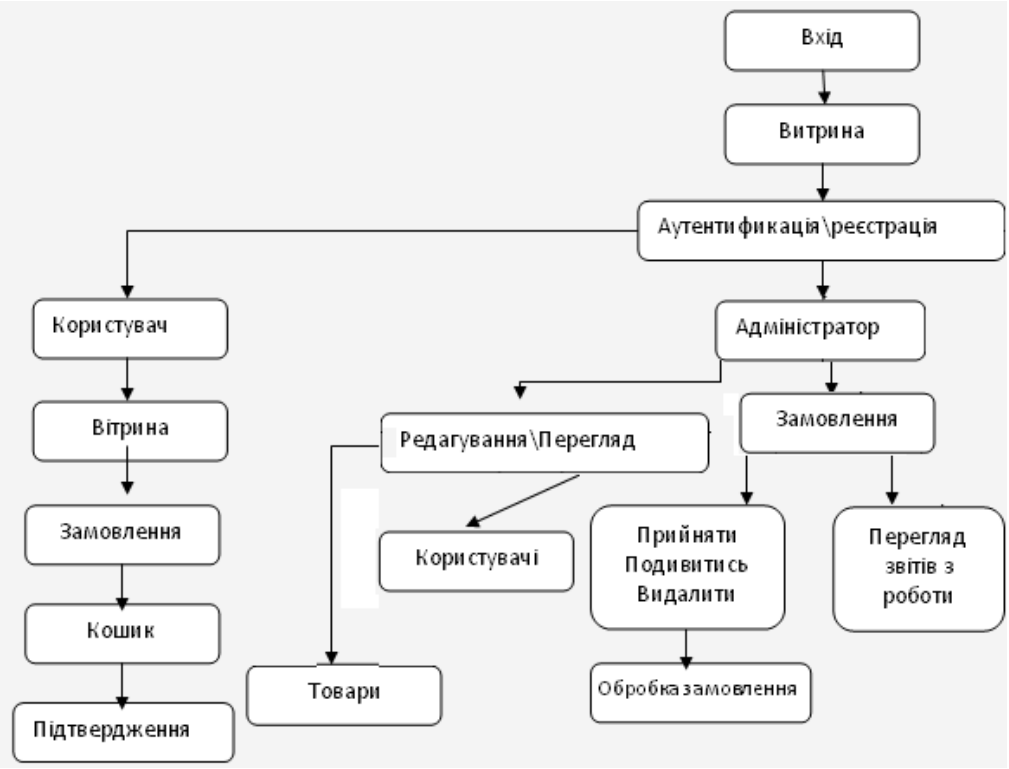

Рис. 2.3. Структурна схема веб-ресурсу

Згідно з вимогами, особистий кабінет даного веб-додатку повинен надавати користувачеві ряд функцій, що наведені на рис. 2.5:

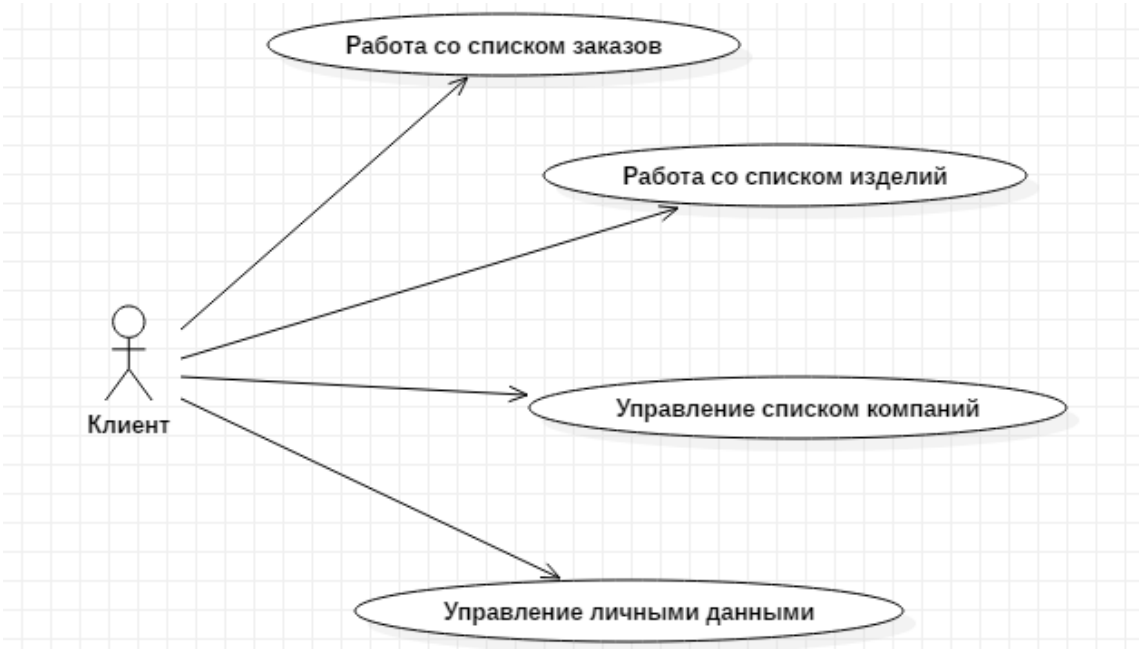

Рис. 2.5. Діаграма варіантів використання для особистого кабінету клієнта підприємства.

Деталі цих функцій описані нижче.

1. Робота зі списком замовлень.

Особистий кабінет надає основний і найважливіший функціонал - можливість оформлення замовлення. Замовити можна товар з готового списку виробів з затвердженими технологічними картами або ж залишити заявку на новий вид виробу й замовити його після створення технологічної карти. Особистий кабінет надає можливість переглядати список замовлень, а також кожне замовлення окремо, додавати, переглядати, редагувати і видаляти замовлення.

2. Робота зі списком виробів. Як було написано вище, одним з початкових етапів виробництва виробів є вибір або створення технологічної карти. Особистий кабінет надає можливість залишити заявку на виробництво технологічної карти для будь-якого типу пакувального виробу, а також стежити за станом обробки цієї заявки. Коли технологічна карта буде розроблена і узгоджена, можна буде залишати замовлення на дану продукцію.

Також користувач має доступ до перегляду вже створених і затверджених технологічних карт. Крім того, для користувача передбачена можливість поміщати в архів застарілі технологічні карти.

3. Управління списком компаній. Кожен зареєстрований користувач в теорії може бути представником декількох компаній (юридичних осіб) або ж взагалі замовляти коробки особисто для себе (як фізична особа). Тому було вирішено додати можливість користувачеві мати власний список компаній і функціонал для їх управління. Зокрема, користувачеві надається можливість додавати, редагувати, переглядати і видаляти елементи списку компаній.

4. Управління особистими даними. Користувач має можливість переглядати і редагувати свої особисті дані на окремо відведеній для цього сторінці кабінету.

На рис. 2.6 наведено послідовність додавання замовлення клієнта до інформаційної системи додатку.

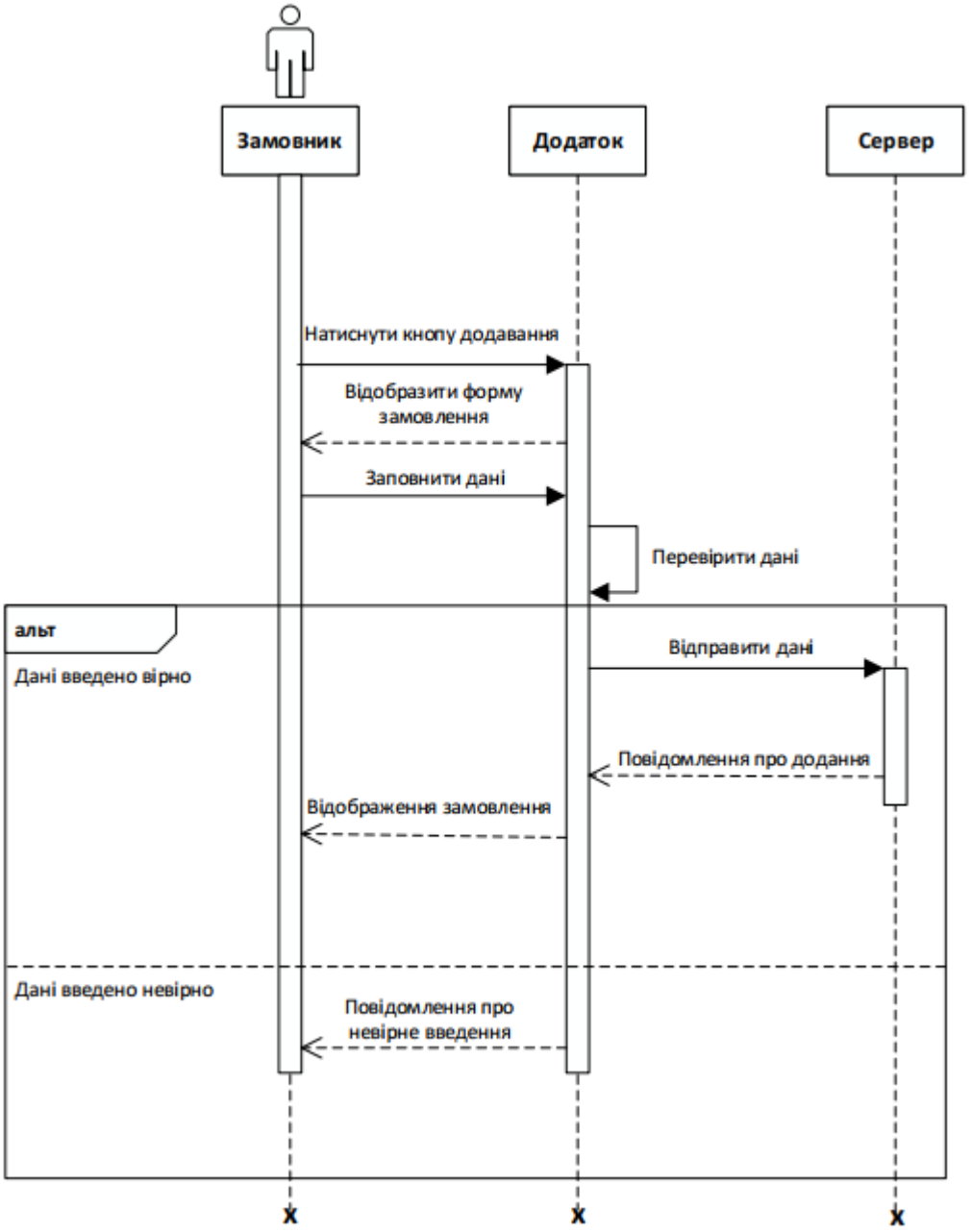

Рис. 2.7. Діаграма варіантів (додавання замовлення)

На рис. 2.8. зображено процедуру прийняття замовлення клієнта на виконання адміністратором (продавцем).

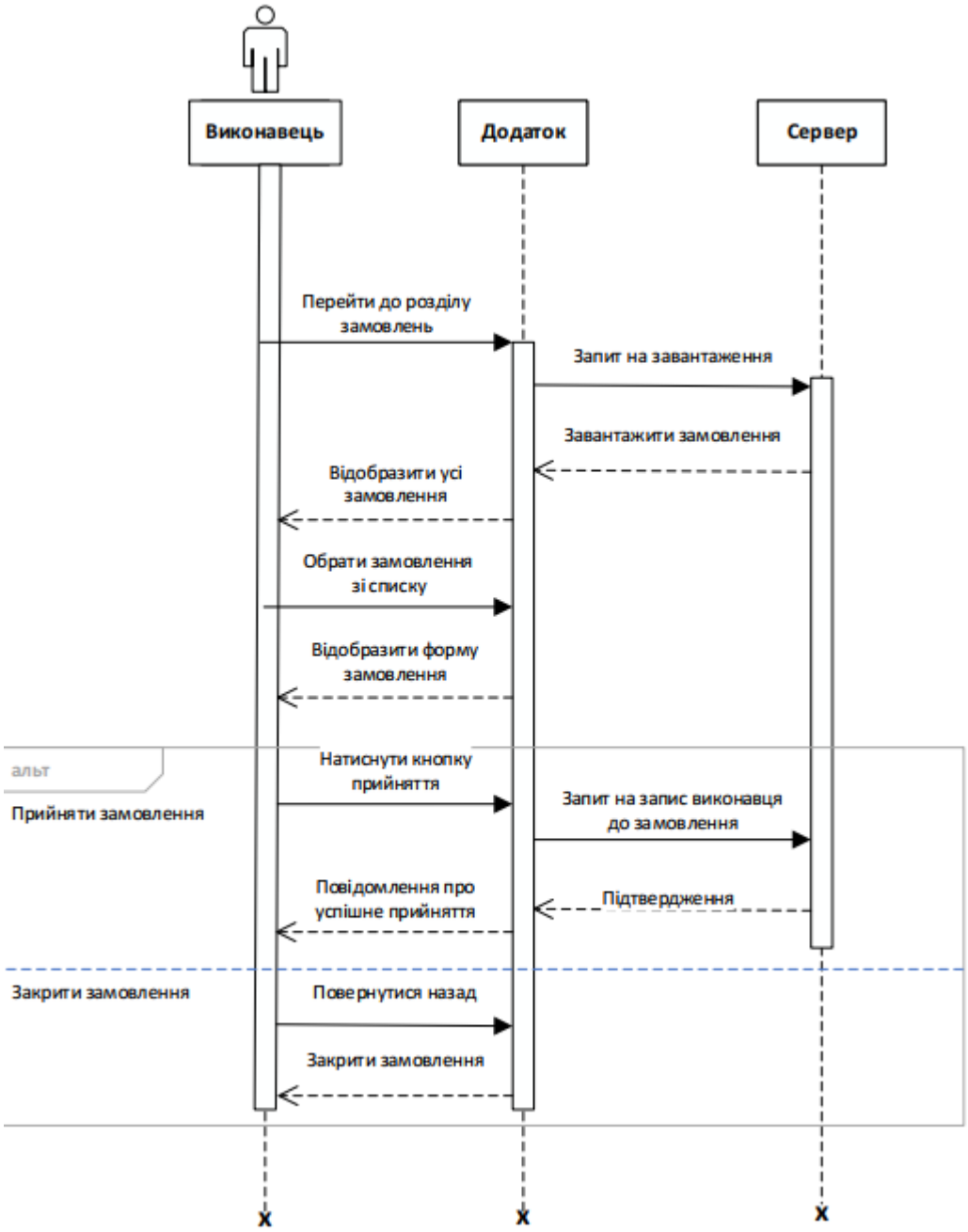

Рис. 2.8. Діаграма варіантів (виконання замовлення)

На рис. 2.9. представлена більш детальна блок-схема функціонування програми з урахуванням всіх Ролей і Дій користувачів.

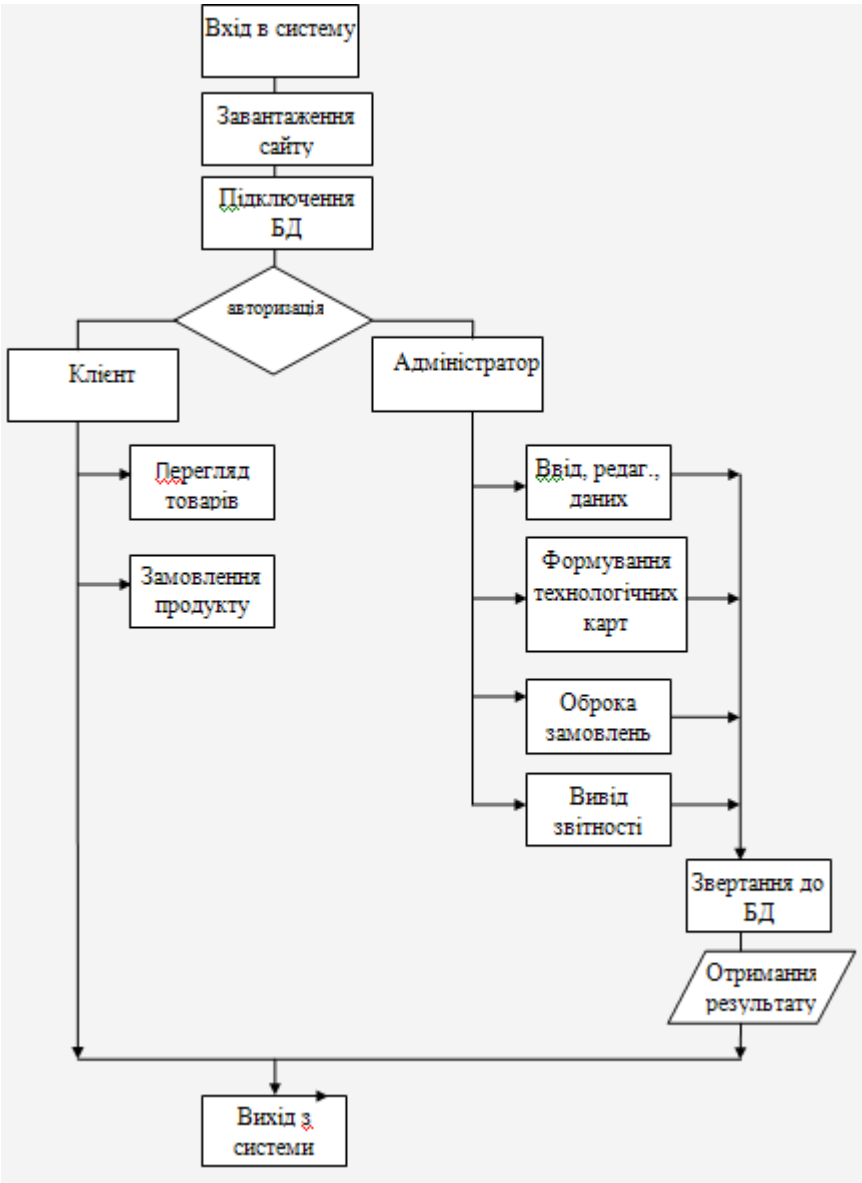

Рис. 2.9. Схема функціонування системи

2.4.3. Проектування бази даних

Розроблений додаток зберігає інформацію, що використовується для роботи, у своїй основі даних під ім'ям PakStandart. У розробленій інформаційній системі використовуються наступні структури: список товарів (objects), каталог технологічних карт (types), список користувачів (users), список ролей (role), список замовлень (orders). База даних складається з 7 таблиць, структура БД представлена на рис. 2.10.

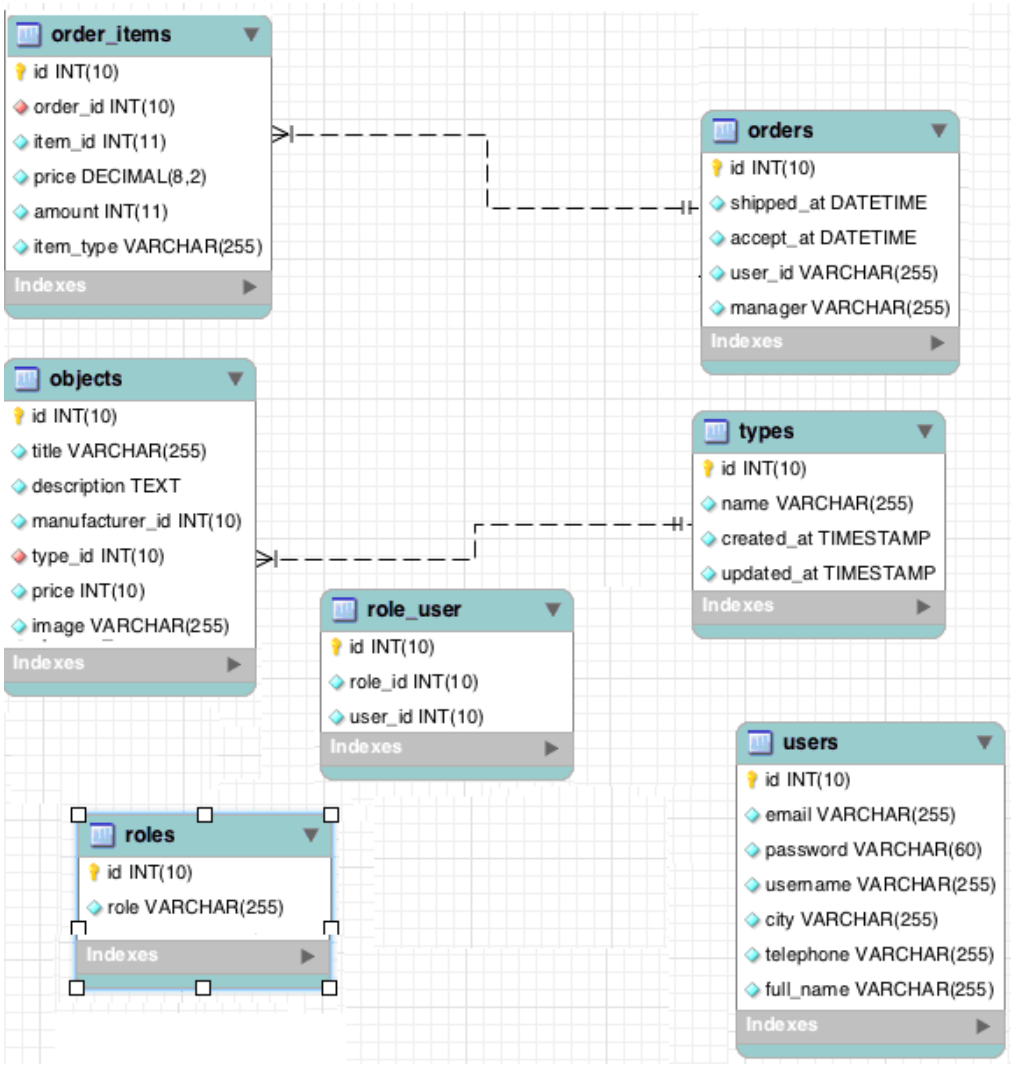

Рис. 2.10. Структура БД системи

Таблиця objects містить поля id, title (назва товару), description (склад, характеристики), price (ціна), image (шлях до файлу з обкладинкою), а також зв'язок з таблицею types. Таким чином, в таблиці objects вказується поле type_id, яке відповідає id в таблиці types (рис. 2.11).

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии | Опер

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(10)		UNSIGNED	Нет	Нет	AUTO_INCREMENT
☐	2	title	varchar(255)	utf8_unicode_ci		Нет	Нет	
☐	3	description	text	utf8_unicode_ci		Нет	Нет	
☐	4	type_id	int(10)		UNSIGNED	Нет	Нет	
☐	5	price	int(10)		UNSIGNED	Нет	Нет	
☐	6	image	varchar(255)	utf8_unicode_ci		Нет	Нет	

Рис.2.11. Зображення таблиці «Товар»

Таблиця types містить поля id, name (назва технологічної карти), і представлена на рис. 2.12:

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
☐	2	name	varchar(255)	utf8_unicode_ci		Нет	*Нет*	

Рис.2.12. Зображення таблиці «Технологічні карти»

Таблиця users містить поля id, email (адреса електронної пошти), password (пароль), username (ім'я користувача), city (місто), telephone (телефон), full_name (повне ім'я) (рис. 2.13):

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
☐	2	email	varchar(255)	utf8_unicode_ci		Нет	*Нет*	
☐	3	password	varchar(60)	utf8_unicode_ci		Нет	*Нет*	
☐	4	username	varchar(255)	utf8_unicode_ci		Нет	*Нет*	
☐	5	city	varchar(255)	utf8_unicode_ci		Нет	*Нет*	
☐	6	telephone	varchar(255)	utf8_unicode_ci		Нет	*Нет*	
☐	7	full_name	varchar(255)	utf8_unicode_ci		Нет	*Нет*	

Рис. 2.13.Зображення таблиці «Користувачі»

Таблиця roles містить поля id, role (роль користувача, статус) (рис. 2.14):

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
☐	2	role	varchar(255)	utf8_unicode_ci		Нет	*Нет*	

Рис. 2.14. Зображення таблиці «Ролі»

Таблиця role_user призначена для зв'язку таблиць roles і users, оскільки вони не пов'язані, між ними повинна бути багато-до-багатьох, а методів реалізації такого роду зв'язку при розробці БД не існує. Тому нова таблиця

містить поля id, а також зв'язок з таблицями roles і users. Таким чином, в таблиці role_user вказується поле role_id, яке відповідає id в таблиці roles і поле user_id, яке відповідає id в таблиці users (рис. 2.15).

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
2	role_id	int(10)		UNSIGNED	Нет	*Нет*	
3	user_id	int(10)		UNSIGNED	Нет	*Нет*	

Рис. 2.15. Зображення таблиці «Ролі користувачів»

Таблиця orders містить поля id, shipped_at (дата і час надходження замовлення), accept_at (дата і час обробки замовлення), user_id (зв'язок з id користувача, в таблиці користувачів), manager (ім'я компанії) (рис. 2.16):

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
2	shipped_at	datetime			Нет	*Нет*	
3	accept_at	datetime			Нет	*Нет*	
4	user_id	varchar(255)	utf8_unicode_ci		Нет	*Нет*	
5	manager	varchar(255)	utf8_unicode_ci		Нет	*Нет*	

Рис. 2.16. Зображення таблиці «Замовлення»

Таблиця order_items містить поля id, item_id (номер товару), price (ціна), amount (кількість), item_type (розділ технологічні карти), (рис. 2.17), а для зв'язку з таблицею orders вказується поле order_id, яке відповідає id в таблиці orders .

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Привилегии

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(10)		UNSIGNED	Нет	*Нет*	AUTO_INCREMENT
☐	2	order_id	int(10)		UNSIGNED	Нет	*Нет*	
☐	3	item_id	int(11)			Нет	*Нет*	
☐	4	price	decimal(8,2)			Нет	*Нет*	
☐	5	amount	int(11)			Нет	*Нет*	
☐	6	item_type	varchar(255)	utf8_unicode_ci		Нет	*Нет*	

Рис. 2.17. Зображення таблиці «Тіло замовлення»

2.4.4. Програмування системи

2.4.4.1. Програмування БД

Для створення БД системи, виконуємо SQL запити:

```
Структура таблицы `objects`
CREATE TABLE IF NOT EXISTS `objects` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`description` text COLLATE utf8_unicode_ci NOT NULL,
`manufacturer_id` int(10) unsigned NOT NULL,
`type_id` int(10) unsigned NOT NULL,
`price` int(10) unsigned NOT NULL,
`image` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
 KEY `objects_type_id_index` (`type_id`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=12 ;

Структура таблицы `order_items`
```

```
CREATE TABLE IF NOT EXISTS `order_items` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`order_id` int(10) unsigned NOT NULL,
`item_id` int(11) NOT NULL,
`price` decimal(8,2) NOT NULL,
`amount` int(11) NOT NULL,
`item_type` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
KEY `order_id_index` (`order_id`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=9 ;
```

Структура таблицы `role_user`

```
CREATE TABLE IF NOT EXISTS `role_user` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`role_id` int(10) unsigned NOT NULL,
`user_id` int(10) unsigned NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
KEY `role_user_role_id_index` (`role_id`),
KEY `role_user_user_id_index` (`user_id`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=10 ;
```

Структура таблицы `types`

```
CREATE TABLE IF NOT EXISTS `types` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`),
  UNIQUE KEY `types_name_unique` (`name`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=11 ;

Структура таблицы `users`
CREATE TABLE IF NOT EXISTS `users` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`password` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
`username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`city` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`telephone` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`full_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
UNIQUE KEY `users_email_unique` (`email`),
UNIQUE KEY `users_username_unique` (`username`),
KEY `users_telephone` (`telephone`),
KEY `users_city` (`city`),
KEY `users_full_name` (`full_name`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
```

```
AUTO_INCREMENT=3 ;
    CREATE TABLE IF NOT EXISTS `objects` (
    `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
    `title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
    `description` text COLLATE utf8_unicode_ci NOT NULL,
    `manufacturer_id` int(10) unsigned NOT NULL,
    `type_id` int(10) unsigned NOT NULL,
    `price` int(10) unsigned NOT NULL,
    `image` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
    `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
    `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
    PRIMARY KEY (`id`),
    KEY `objects_type_id_index` (`type_id`)
    )
    ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=12 ;

    Структура таблицы `order_items`
    CREATE TABLE IF NOT EXISTS `order_items` (
    `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
    `order_id` int(10) unsigned NOT NULL,
    `item_id` int(11) NOT NULL,
    `price` decimal(8,2) NOT NULL,
    `amount` int(11) NOT NULL,
    `item_type` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
    `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
    `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
    PRIMARY KEY (`id`),
    KEY `order_id_index` (`order_id`)
    )
```

```
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=9 ;

Структура таблицы `role_user`
CREATE TABLE IF NOT EXISTS `role_user` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`role_id` int(10) unsigned NOT NULL,
`user_id` int(10) unsigned NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
KEY `role_user_role_id_index` (`role_id`),
KEY `role_user_user_id_index` (`user_id`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=10 ;

Структура таблицы `types`
CREATE TABLE IF NOT EXISTS `types` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
UNIQUE KEY `types_name_unique` (`name`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=11 ;

Структура таблицы `users`
```

```
CREATE TABLE IF NOT EXISTS `users` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`password` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
`username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`city` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`telephone` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`full_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`),
UNIQUE KEY `users_email_unique` (`email`),
UNIQUE KEY `users_username_unique` (`username`),
KEY `users_telephone` (`telephone`),
KEY `users_city` (`city`),
KEY `users_full_name` (`full_name`)
)
ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=3 ;
```

2.4.4.2. Програмування графічного інтерфейсу

Логіка фремворка Vue.js побудована на створенні компонентів, які можна використовувати кілька разів в різних місцях сайту. Компонентами є елементи, до яких компілятор Vue прикріплює деякий поведінку. Компоненти дозволяють інкапсулювати код і потім використовувати його багато разів в різних частинах програми.

Кожен компонент Vue.js міститься в окремому файлі. У кожному з цих файлів визначена розмітка, стилі і скрипти компонента.

Даний фреймворк допускає можливість зберігання стилів в окремих файлах, і в даному проекті ця можливість була використана - практично всі стилі зберігаються в окремій папці assets / css, а не в файлі компонентів.

Для спрощення розробки призначеного для користувача інтерфейсу була використана бібліотека Bootstrap. Bootstrap - це інструментарій з відкритим вихідним кодом для розробки за допомогою HTML, CSS і JS. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Основними перевагами Bootstrap, які були використані в даній роботі, є: система сіток (розбиття сторінки на колонки і ряди для зручного вибудовування елементів), типографіка (спеціальні шрифти для різних ситуацій і станів додатку), адаптивність для різних типів пристроїв (даний сайт має планшетні і мобільний версії).

Як вже говорилося вище, логіка Vue.js побудована на компонентах. Кожна сторінка сайту складається з набору таких компонентів або зовсім може бути одним компонентом. Для навігації між сторінками сайту (а також компонентами) використовується спеціальний інструмент - Vue Router.

Vue Router - це офіційна бібліотека маршрутизації для Vue.js. Вона глибоко інтегрується з Vue.js і дозволяє легко переходи між сторінками. Включає наступні можливості:

- вкладені маршрути/уявлення;
- модульна конфігурація маршрутизатора;
- доступ до параметрів маршруту, query, wildcards;
- анімація переходів уявлень на основі Vue.js;
- зручний контроль навігації;
- автоматичне проставляння активного css класу для посилань;
- режими роботи html5 history або hash, з авто-перемиканням в IE9;
- налаштована поведінка прокрутки сторінки.

Головне меню веб-додатку складається з наступних пунктів: Головна, Новини, Прайси, Контакти та Особистий кабінет.

Основними розділами особистого кабінету є такі сторінки: «Оформити замовлення», «Історія замовлень», «Довідник виробів», «Архів виробів», «Мої компанії», «Особисті дані». Саме між ними відбувається навігація, керована роутером.

Фрагмент файлу-роутера даного додатку представлений в додатку А.

2.4.4.3. Розробка інтерфейсу прикладного програмування

При створенні даного веб-додатку неодноразово необхідно отримувати і відображати дані з серверних API. Зокрема отримувати інформацію з бази даних про замовлення, технологічних картах, компаніях і навіть про користувача. Крім отримання, сайт надає можливість змінювати, додавати і видаляти деяку інформацію.

API, або інтерфейс прикладного програмування, є програмним посередником, який дозволяє двом додаткам спілкуватися один з одним. API часто надає дані, які інші розробники можуть використовувати в своїх власних додатках, не турбуючись про бази даних або відмінностях у мовах програмування. Розробники часто отримують дані з API в форматі JSON, які вони інтегрують в інтерфейсні програми.

Існує кілька способів використання API у Vue.js, але найбільш популярним рішенням є використання axios, заснованого на Promise HTTP-клієнта. Саме цей інструмент був використаний в даному проекті.

Дана бібліотека надає методи як GET, PUT, DELETE, PATCH і POST, які відповідно дозволяють зчитувати, оновлювати (замінювати), видаляти, оновлювати (модифікувати) і створювати. Наприклад, нижче наведено фрагмент коду, що відправляє дані на сервер при вході в особистий кабінет:

```
login ({commit}, {user, $ router}) {
   axios.post ( '/ auth / login /', user)
```

```
                .then (req => {
                  //console.log('login success ', req);
                  if (! req.data.token) {
                    delete localStorage.token;
                    delete localStorage.person_uuid;
                    delete localStorage.person_companies_count;
                    delete localStorage.person_name;
                    return commit ( 'set_auth', false);
                  }
                 localStorage.setItem ( 'token', req.data.token);
                  localStorage.setItem ( 'person_uuid', req.data.person_uuid);
                  localStorage.setItem('person_companies_count',
req.data.person_companies_count);
                  localStorage.setItem ( 'person_name', req.data.person_name);
                  commit ( 'set_auth', true);
                  commit ( 'login_name', localStorage.getItem ( 'person_name'));
                  commit ( 'uuidPersonBack', localStorage.getItem ( 'person_uuid'));
                  if (req.data.person_companies_count === 0) {
                    $ Router.replace ( '/ profile / add-companies')
                  } Else {
                    $ Router.replace ( '/ profile / order-history')
                  }
                    })
                .catch (error => {
                  console.error (error);
                  delete localStorage.token;
                  delete localStorage.person_uuid;
                  delete localStorage.person_companies_count;
                  delete localStorage.person_name;
                  commit ( 'set_auth', false);
```

```
            commit ( 'set_error', true);
            setTimeout (function () {
              commit ( 'set_error', false)
            }, 6000);
            });
    },
```

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідна інформація для роботи з веб-додатком інформаційної системи зберігається в БД та містить такі дані:

- інформація про товар та їх параметри, за якими вони виготовляються;
- інформація про користувача, компанію замовника;
- інформація про замовлення.

В результаті роботи програми може бути отримана наступна вихідна інформація, що надається за запитами з БД

- інформація про вміст БД (товари, технологічні карти, користувачі).
- інформація про замовлення.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки програми використовувався комп’ютер з наступними параметрами:

- процесор Dual Core Intel чи AMD з частотою 2.8 GHz;
- оперативна пам'ять 4GB ОЗУ;
- відеокарта nVidia GeForce 8600/9600GT, ATI/AMD
- Radeon HD2600/3600;
- DirectX версії 8.0c;

– жорсткий диск 32 GB.

2.6.2. Використані програмні засоби

Інформаційна система являє собою веб-додаток, для розробки призначеного для користувача інтерфейсу були використані такі інструменти: мова розмітки гіпертексту html, таблиці стилів css і мова програмування JavaScript, зокрема його фреймворк Vue.js. База даних складена за допомогою СКБД MySQL 5.5.34. Робота системи забезпечується під управлінням ОС MS Windows 7.

2.6.3. Виклик та завантаження програми

Завантаження інформаційної системи здійснюється з сервера. Запуск інформаційної системи проводиться переходом по посиланню http://pakstandart.ua. З'являється головне вікно веб-додатку інформаційної системи.

2.6.4. Опис інтерфейсу користувача

Після завантаження головного вікна веб-додатку, на екрані користувача з’являється головна сторінка (рис. 2.18).

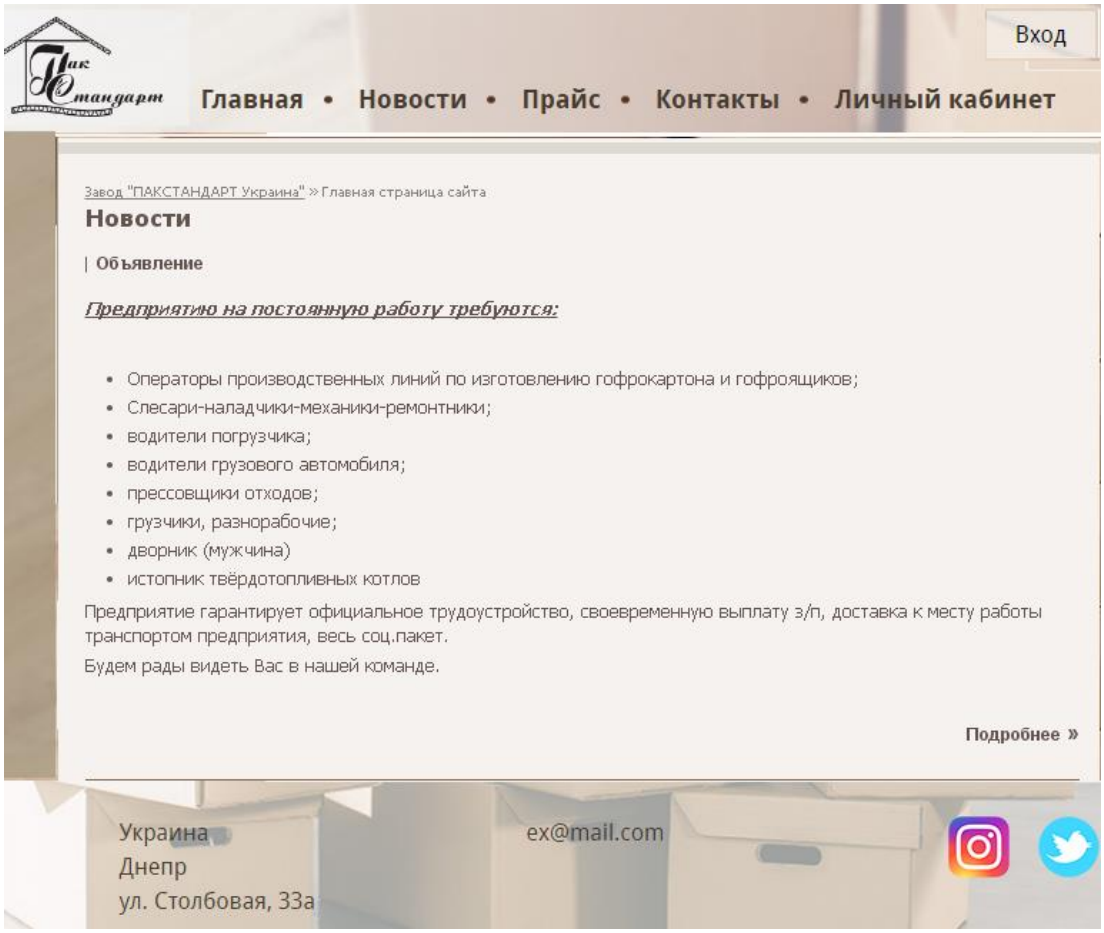

Рис. 2.18. Головна сторінка сайту

Головне меню веб-додатку складається з наступних пунктів: Головна, Новини, Прайси, Контакти та Особистий кабінет.

Користувач – потенційний покупець має можливість ознайомитися з інформацію про підприємство (рис. 2.19), контакти, переглянути новини та ознайомитися з асортиментом продукції та прасом (рис. 2.20, 2.21),

Завод "ПАКСТАНДАРТ Украина" » о компании

о компании

Предприятие занимается производством изделий из гофрокартона любых размеров и конфигураций.

Ассортимент изготавливаемой продукции разнообразен: гофрокартон, гофролотки, гофротрейсы, гофроящики от простых до сложной конфигурации и комплектующие к ним прокладки, обечайки, решетки. На изделия из гофрокартона может наноситься одно-, двухцветная печать в виде логотипа предприятия или любая другая информация.

Вся продукция отвечает стандартам и требованиям ГОСТов.

Предприятие реализует продукцию по всей Украине.

Товары, упаковываемые в гофротару:

- автомобильные детали;
- бумага и изделия из бумаги;
- бытовая химия;
- игрушки;
- книги;
- кондитерские изделия;
- консервированные продукты в жестяных и стеклянных банках;
- краски;
- лекарства;
- ликеро-водочные изделия;
- макаронные изделия;
- масло и маргарин;
- мебель;
- метизы;
- мясные продукты;
- напитки безалкогольные и алкогольные;
- нефтепродукты;
- обувь;
- одежда и галантерея;
- парфюмерия и косметика;
- приборы;
- скобяные изделия;
- спорттовары;
- табачные изделия;
- фрукты, ягоды свежие и сушенные.

Соотношение цены и качества Вас приятно удивит. Для постоянных клиентов действует система скидок, отсрочка платежа.

Ждем Ваших заказов!

Рис. 2.19. Інформація про підприємство

Рис. 2.21. Інформація про асортимент

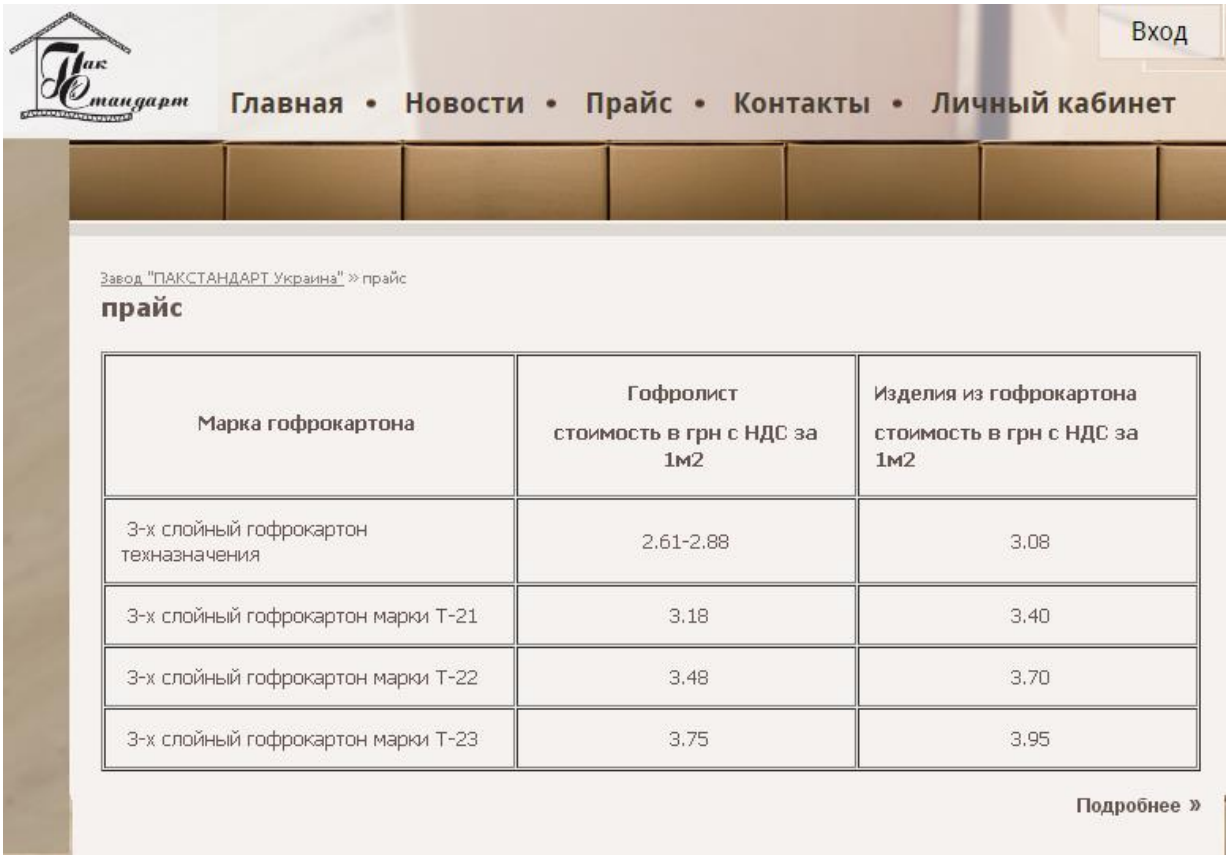

Рис. 2.21. Прайс лист товарів та матеріалів

Для входу в систему для здійснення замовлення або адміністрування системи, на головній формі передбачена кнопка «Вход», яка дозволяє зареєструватися або авторизуватися (рис. 2.22, 2.23).

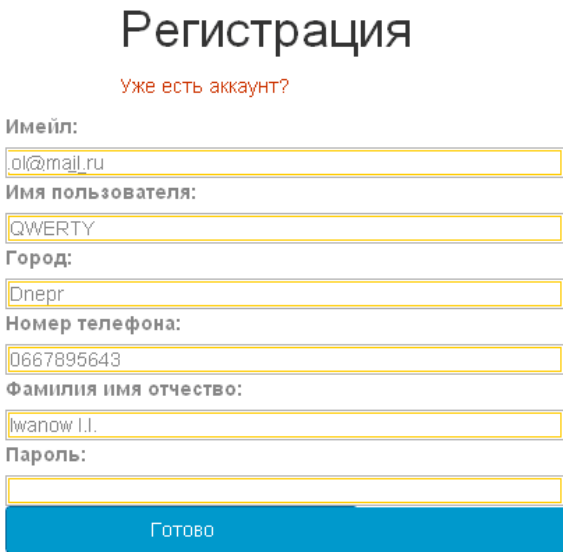

Рис. 2.22. Форма реєстрації користувача

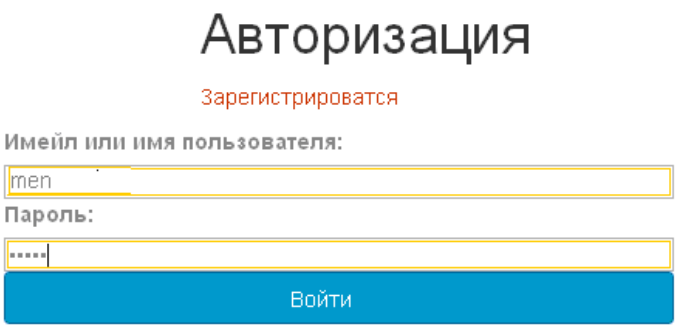

Рис. 2.23. Авторизація в системі

Після успішної авторизації, користувачу стає доступним функціонал особистого кабінету користувача (рис. 2.24).

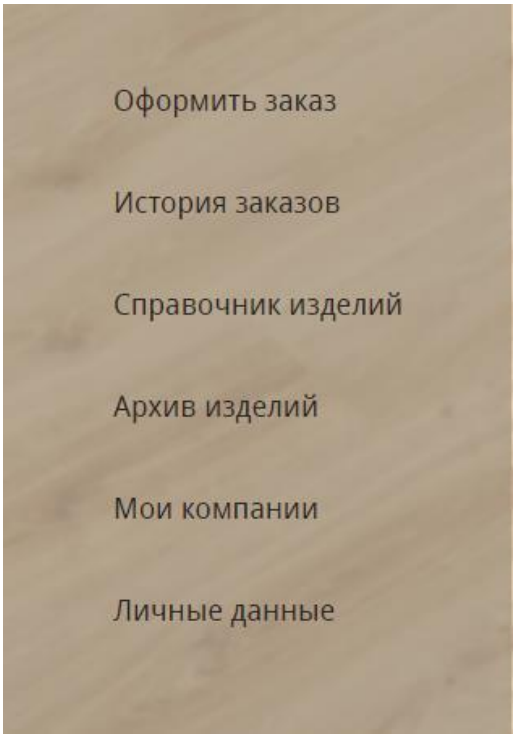

Рис. 2.24. Меню особистого кабінету користувача

Основними розділами особистого кабінету є такі сторінки: «Оформити замовлення» (рис. 2.25), «Історія замовлень» (рис. 2.26), «Довідник виробів» (рис. 2.27), «Архів виробів» (рис. 2.28), «Мої компанії» (рис. 2.29, 2.30), «Особисті дані» (рис. 2.31).

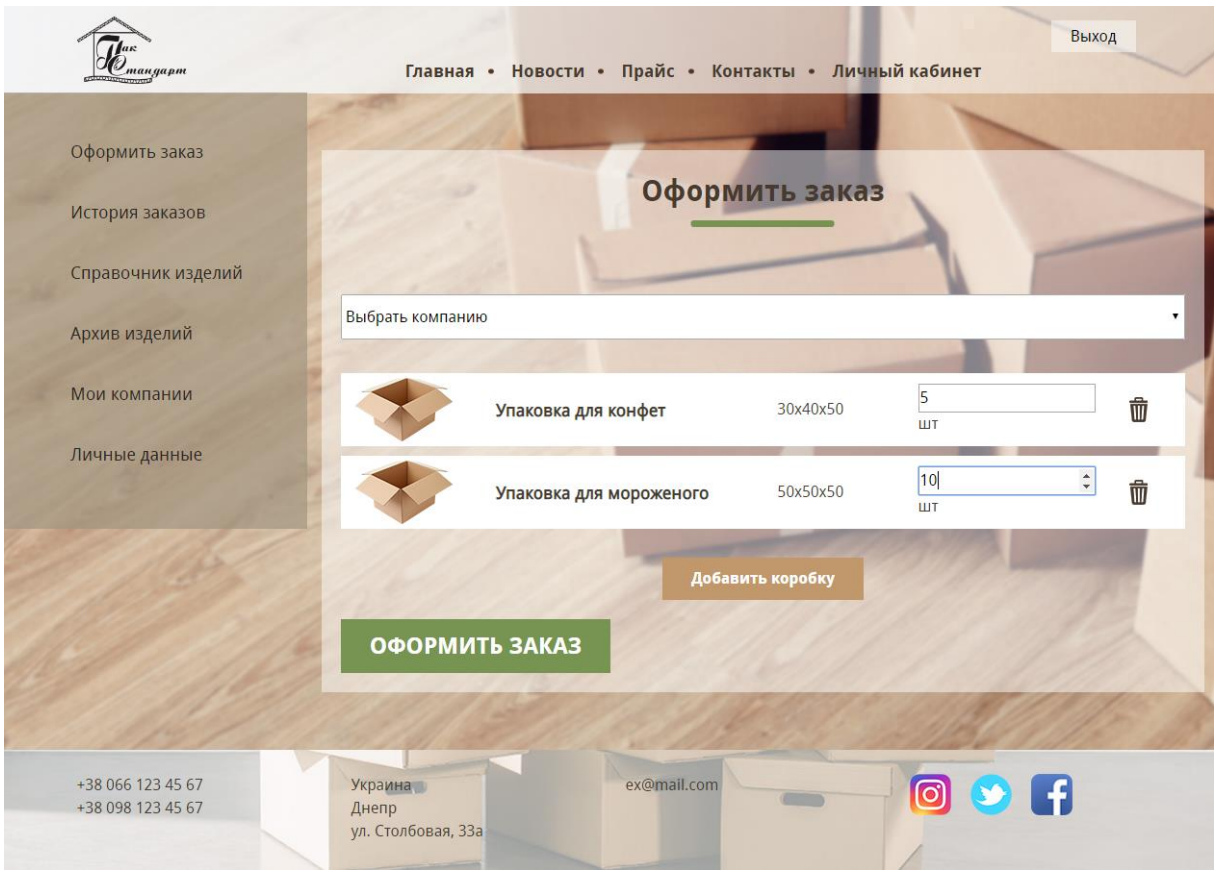

Рис. 2.25. Сторінка «Оформити замовлення»

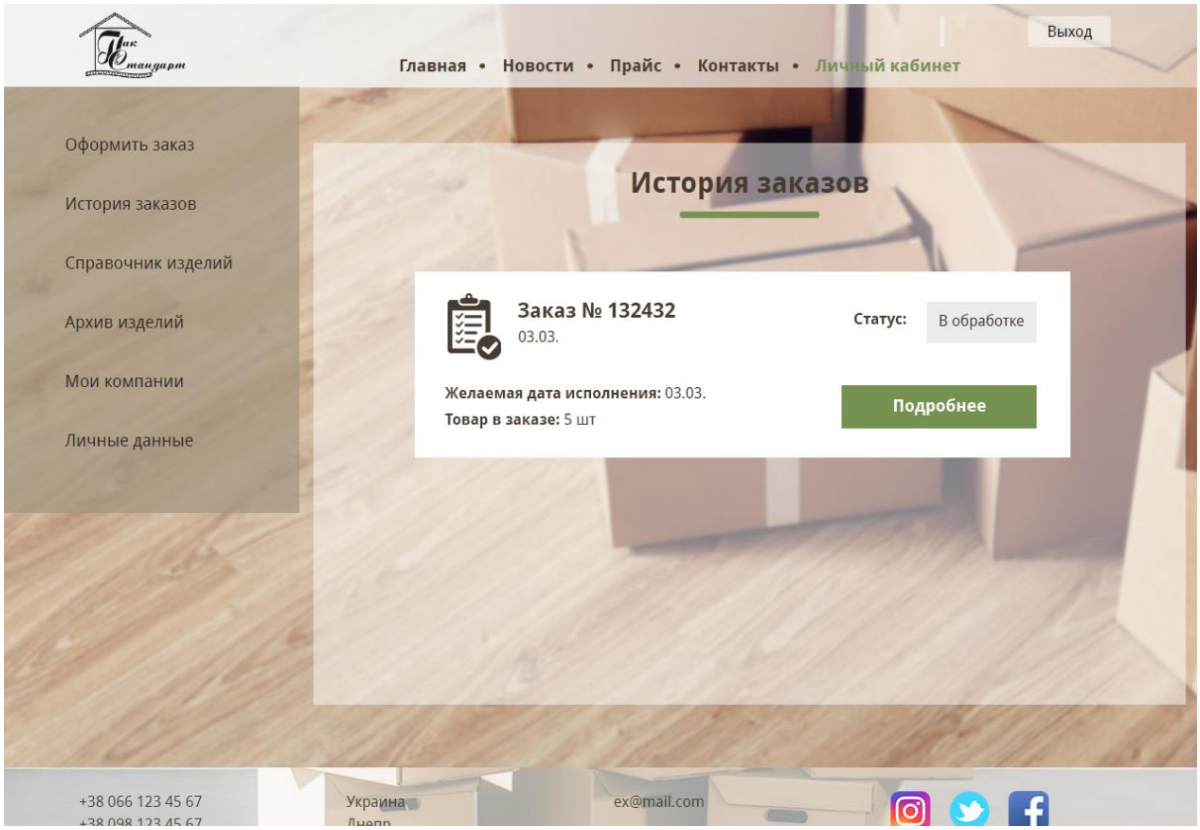

Рис. 2.26. Сторінка «Історія замовлень»

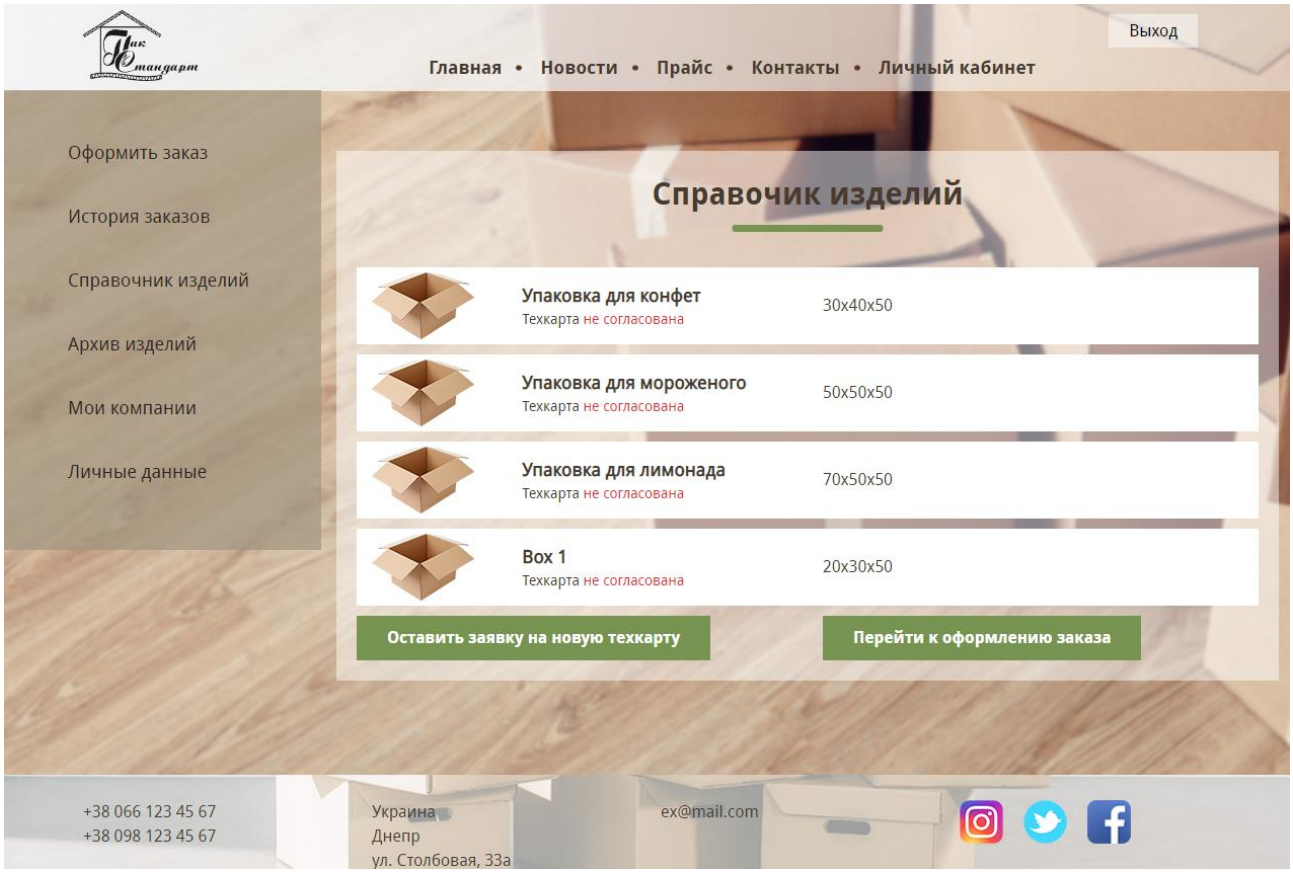

Рис. 2.27. Сторінка «Довідник виробів»

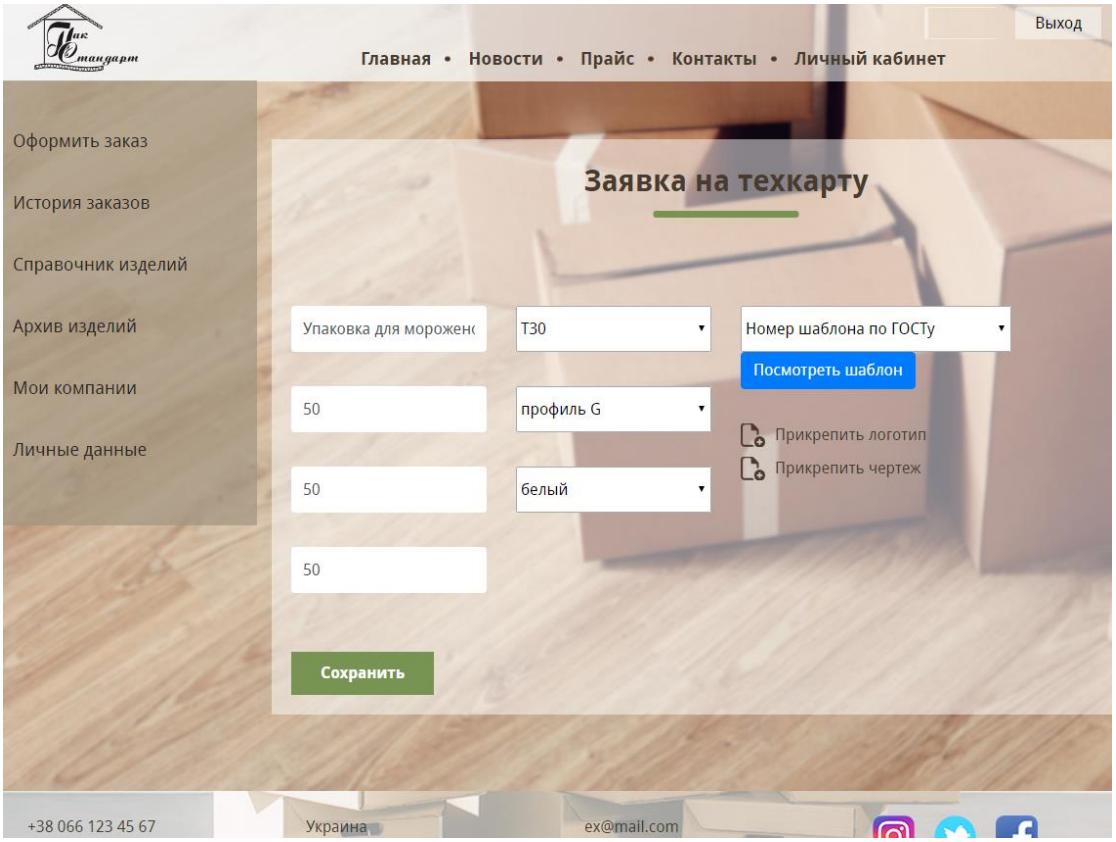

Рис. 2.28. Сторінка «Замовлення на технологічну карту»

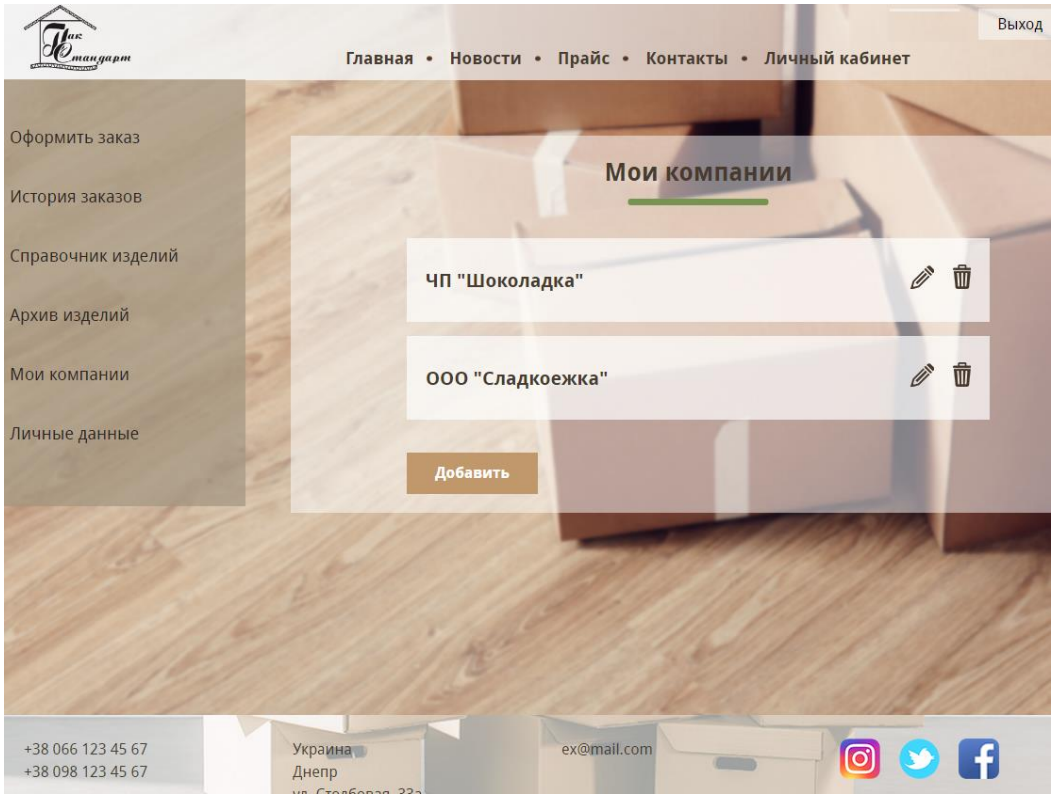

Рис. 2.29. Сторінка «Мої компанії»

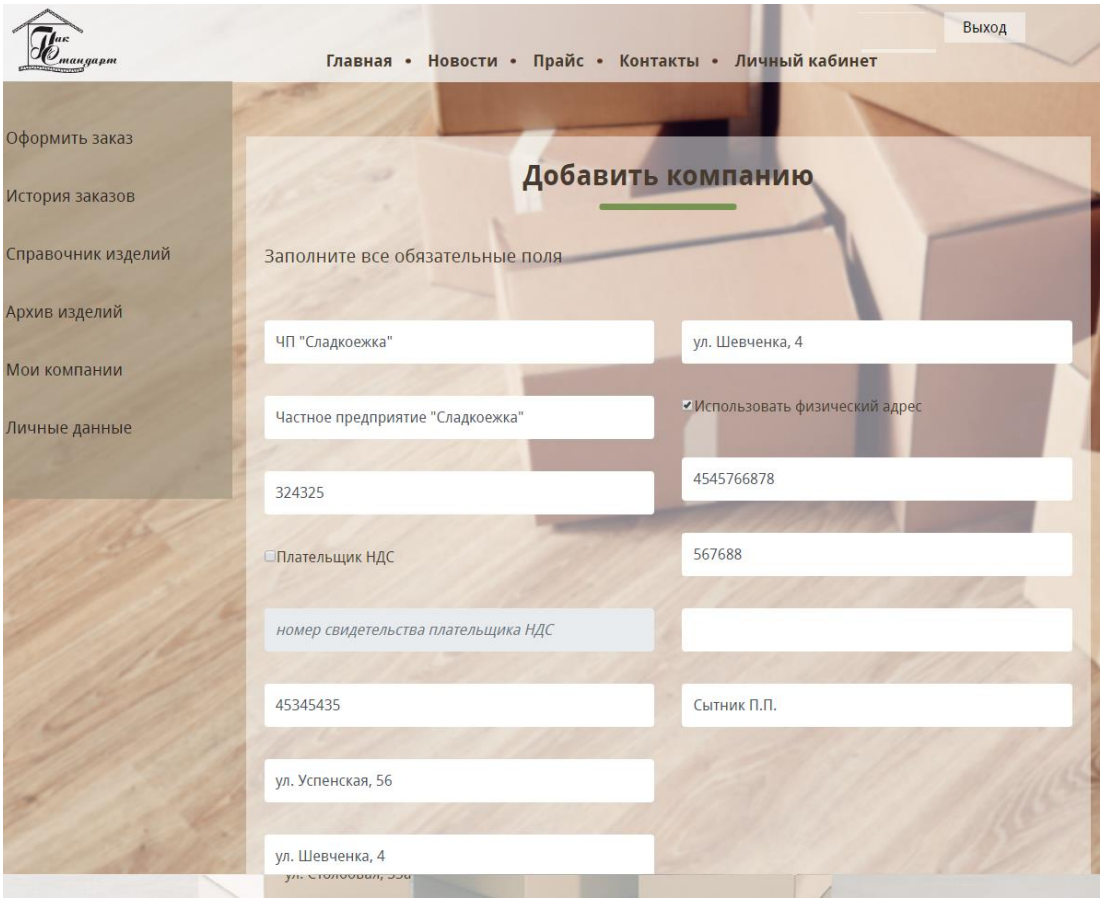

Рис. 2.30. Сторінка «Додати компанію»

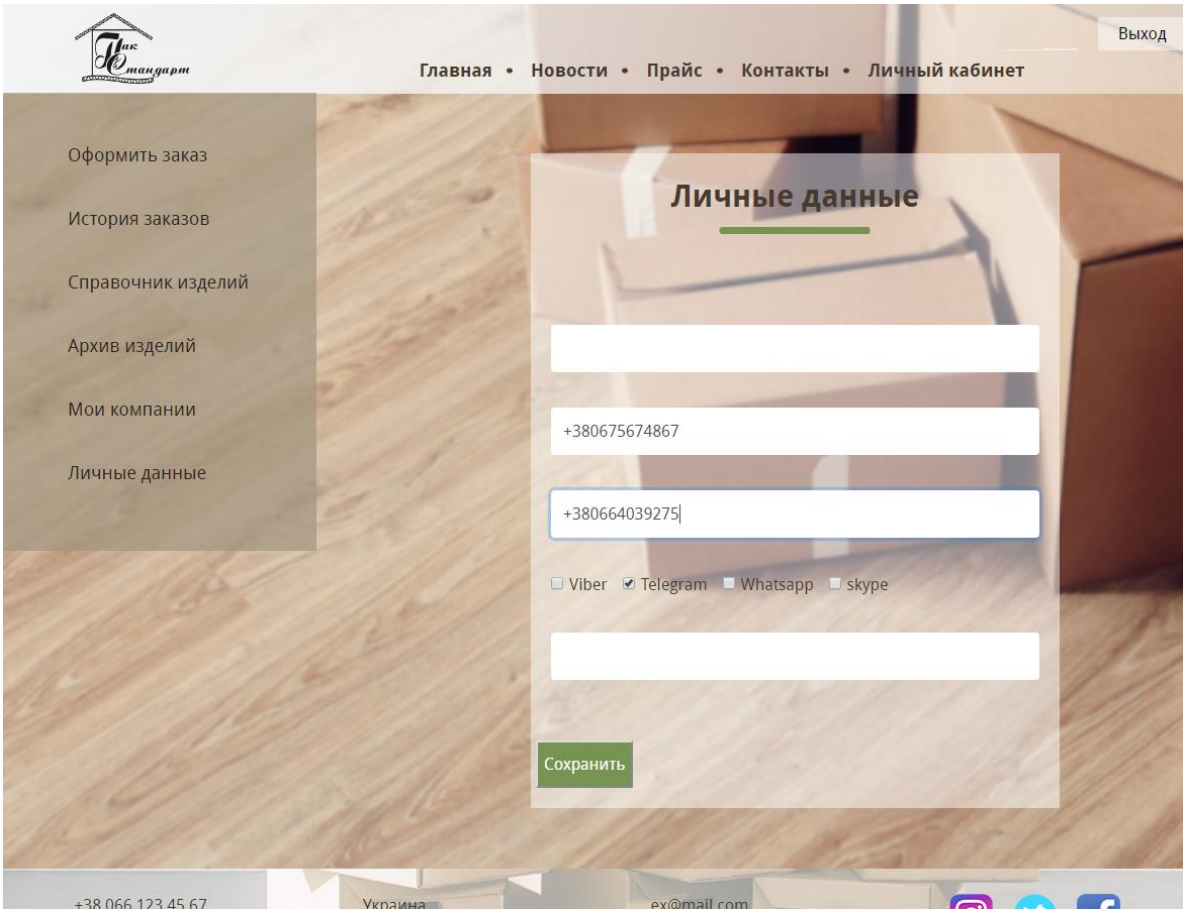

Рис. 2.31. Сторінка «Особисті дані»

Якщо користувач зареєстрований в системі як адміністратор, то після успішної аутентифікації йому стають доступними функції роботи з базою даних системи: редагування, додавання та видалення даних про вироби та користувачів системи, а також робота з замовленнями. На рис. 2.32 наведено приклад обробки отриманих замовлень на виготовлення продукції.

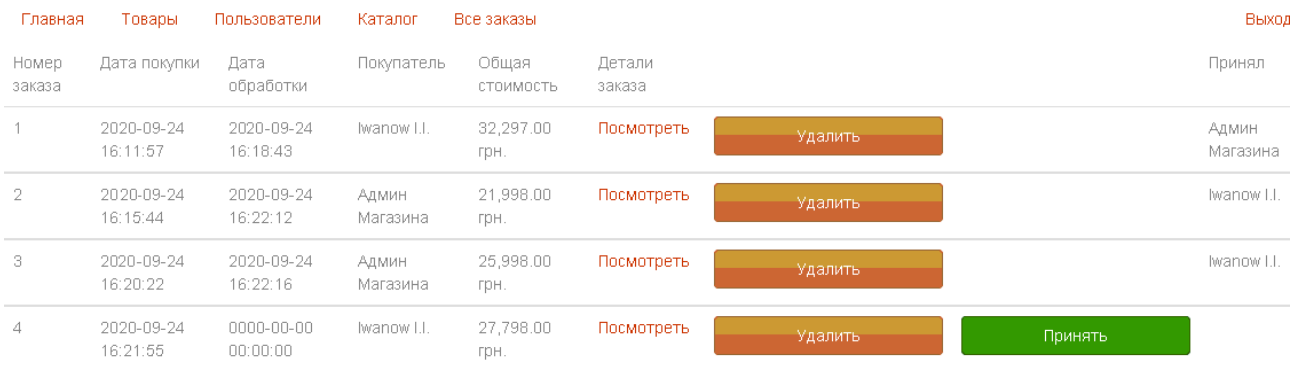

Рис. 2.32. Пункт меню «Замовлення»

Пункт меню «Користувачі» (рис. 2.33, 2.34) дозволяє переглянути інформацію про всіх користувачів системи, видаляти їх із системи, редагувати дані, а також призначати їм права доступу, призначаючи відповідні статуси.

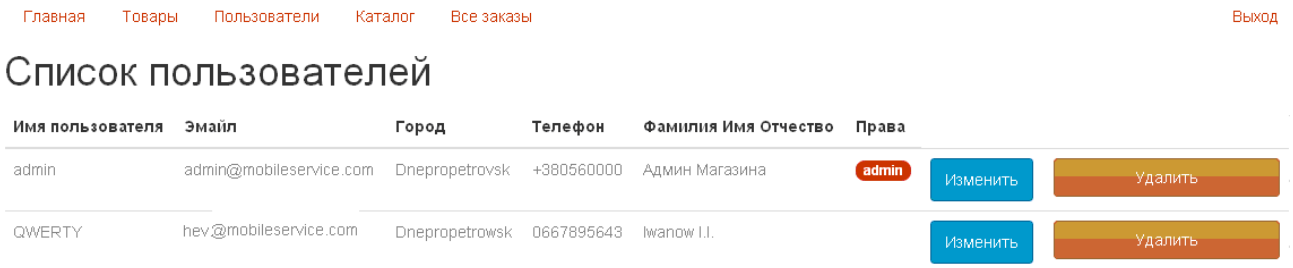

Рис. 2.33. Пункт меню «Користувачі»

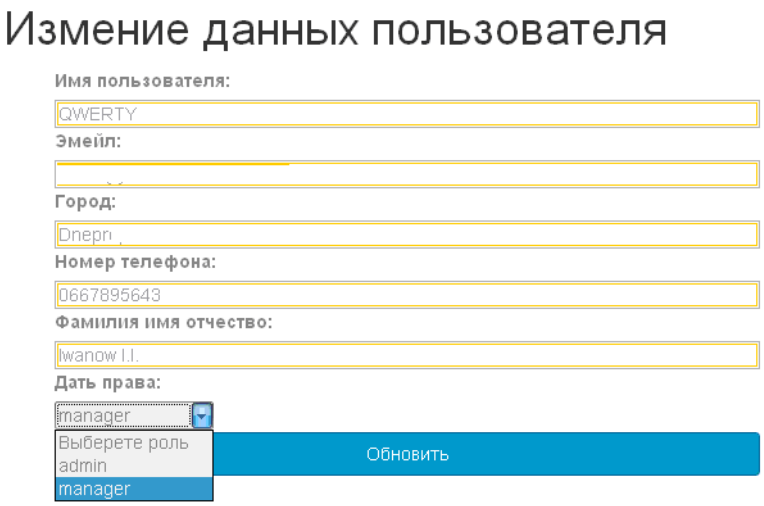

Рис. 2.34. Форма для зміни даних користувача

Пункт меню «Вихід» дозволяє вийти з поточного сеансу і завантажити головне вікно веб-додатку.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 800;
- коефіцієнт складності програми - 1,5;
- коефіцієнт корекції програми в ході її розробки - 0,05;
- годинна заробітна плата програміста, грн / год - 40.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де:

to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

tu - витрати праці на дослідження алгоритму рішення задачі;

ta - витрати праці на розробку блок-схеми алгоритму;

tn - витрати праці на програмування по готовій блок-схемі;

тотл - витрати праці на налагодження програми на ЕОМ;

tд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p)\text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 800 \cdot 1{,}5 \cdot (1 + 0{,}05) = 1260 \text{ людино-годин.} \qquad (3.3)$$

Витрати праці на вивчення опису задачі *tu* визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}\text{ , людино-годин,} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{1260 \cdot 1{,}2}{80 \cdot 0{,}8} = 23\text{, людино-годин.} \qquad (3.4)$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.5)$$

$$t_a = \frac{1260}{20 \cdot 0{,}8} = 78 \text{ людино-годин.} \qquad (3.6)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.7}$$

$$t_n = \frac{1260}{25 \cdot 0{,}8} = 63 \text{ людино-годин.} \tag{3.8}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.9}$$

$$t_{отл} = \frac{1260}{4 \cdot 0{,}8} = 393 \text{ людино-годин.} \tag{3.10}$$

за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \tag{3.11}$$

$$t_{отл} = 393*1{,}2 = 472{,}5 \tag{3.12}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.13}$$

де *tдр* - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.14}$$

$$t_{др} = \frac{1260}{15 \cdot 0{,}8} = 105 \text{ людино-годин,} \tag{3.15}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.16}$$

$$tdo = 0{,}75 \cdot 105 = 78 \quad (3.17)$$

$$t_{\partial} = 105 + 78 = 183 \text{ людино-годин.} \quad (3.18)$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 23 + 78 + 63 + 472 + 183 = 871 \text{ людино-годин.} \quad (3.19)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.20)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.20)$$

де:

t - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 871 \cdot 40 = 34865 \text{ грн.} \quad (3.21)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.22)$$

де:

тотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 7 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 472 \times 7 = 3307 \text{ грн.} \tag{3.23}$$

$$Кпо = 34865 + 3307 = 38172 \text{ грн.} \tag{3.24}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \tag{3.25}$$

де:

Bk - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні *Fp*=176 годин).

$$T = \frac{871}{1 \cdot 176} = 4{,}9 \text{ міс.} \tag{3.26}$$

Визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (38172 грн.) та розраховано час на його створення (4,9 міс).

ВИСНОВКИ

Метою кваліфікаційної роботи бакалавра є розробка зручного веб-додатку для підприємства, що виготовляє та реалізує пакувальну продукцію, основними функціями якого є надання інформації про послуги та можливості онлайн замовлення різного виду виготовленої продукції.

Інформаційна система являє собою веб-додаток, для розробки інтерфейсу користувача були використані такі інструменти: мова розмітки гіпертексту html, таблиці стилів css і мова програмування JavaScript, зокрема його фреймворк Vue.js. База даних складена за допомогою СКБД MySQL 5.5.34. Робота системи забезпечується під управлінням ОС MS Windows 7.

В ході виконання даної роботи були вивчені особливості фреймворка для розробки користувальницького інтерфейсу Vue.js і таких додаткових бібліотек як BootstrapVue, Vue Router, Axios і ін.

В роботі була досліджена предметна область розробки - система роботи підприємства з виробництва та продажу пакувальних виробів.

Були визначені вимоги на дизайн користувальницького інтерфейсу та програмне забезпечення відповідно до результатів аналізу предметної області і вимог замовника.

Як результат було створено і реалізовано веб-додаток та інтерфейс особистого кабінету для сайту організації по виробництву пакувальної продукції «ПакСтандарт».

Дана робота запропонована для впровадження на реальному підприємстві «ПакСтандарт», яке займається виробництвом різної пакувальної продукції, зокрема виробляє коробки різних розмірів і форм.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (38172 грн) та розраховано час на його створення (4,9 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. А. Горев, С. Макашарипов, Р. Ахаян. Эффективная работа с СУБД; 1997. – 321с.

2. Антони Синтес. Освой самостоятельно объектно-ориентированное программирование за 21 день. Sams Teach Yourself Object-Oriented Programming in 21 Days. - М.: «Вильямс», 2002. - С. 672.

3. Аруп Нанда, Стивен Фейерштейн Oracle PL/SQL для администраторов баз данных; 2008. – 308 с.Кевин Луни, Боб Брила Oracle 10g. Настольная книга администратора баз данных – Лори: 2008. – 752с.

4. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

5. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2021.

6. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

7. Дж. Боуман, С.Эмерсон, М.Дарновски. Практическое руководство SQL; 2000. – 321 с.

8. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

9. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

10. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

11. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

12. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

13. Офіційний сайт середовища розробки Visual Studio Code URL : https://code.visualstudio.com/docs дата звернення: 15.03.2021.

14. Никсон Р., - Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. - СПб.:Питер, 2011. - 496 с.:ил. - (Серия «Бестселлеры O'Reilly»)

15. Microsoft SQL Server. Эффективная работа Вишневский Алексей, серия: Эффективная работа, Изд.: Питер 2009 г.

16. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. / В.Дронов. – СПб.: БХВ-Петербург, 2011. – 416 с.

17. HTML и CSS. Путь к совершенству. / Б. Хеник – СПб.: Питер, 2011. – 336 с.

18. HTML и XHTML. Подробное руководство. Шестое издание. / Ч. Муссиано, Б. Кеннеди. – Издательство: Символ-Плюс, 2011. – 752 с.

19. JavaScript: Подробное руководство (Definitive Guide). / D. Flanagan. - Издательство: Символ-Плюс, 2008. – 992 с.

ДОДАТОК А

КОД ПРОГРАМИ

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title>frontend</title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but frontend doesn't work properly without JavaScript enabled. Please
enable it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

App.vue

```
<template>
  <div id="app">
    <Navigation></Navigation>

    <div class="content-wrapper">
      <router-view></router-view>
    </div>

    <Footer></Footer>
  </div>
</template>

<script>
  import Navigation from './components/Navigation.vue'
  import Footer from './components/Footer.vue'
export default {
  name: 'app',
  components: {
    Navigation,
    Footer
  },
  data(){
    return{
    }
  },
  beforeCreate() {
    this.$store.dispatch('checkToken');
```

```
  }
}
</script>
<style>
  #app{
    display: flex;
    flex-direction: column;
    height: 100%;
  }
</style>

main.js
import Vue from 'vue'
import App from './App.vue'
import Vuelidate from 'vuelidate'
import BootstrapVue from 'bootstrap-vue'
import store from './store'
import router from './router'
import axios from './backend/vue-axios'
import vSelect from 'vue-select'
import VueTheMask from 'vue-the-mask'
Vue.component('v-select', vSelect)

import './assets/css/droidsans.css'
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'
import './assets/css/main.css'
import './assets/js/custom'
import 'vue-select/dist/vue-select.css';

Vue.config.productionTip = false;

Vue.use(Vuelidate)
Vue.use(BootstrapVue)
Vue.use(VueTheMask)

new Vue({
  store,
  router,
  axios,
  render: h => h(App),
}).$mount('#app');

Router/index.js
import Vue from 'vue'
import Router from 'vue-router'
import Home from '@/pages/Home'
import Price from '@/pages/Price'
import News from '@/pages/News'
import Profile from '@/pages/Profile'
import Contacts from '@/pages/Contacts'
import Registration from '@/pages/auth/Registration'
```

```
import Login from '@/pages/auth/Login'
import checkout from '@/pages/profile/checkout'
import myCompanies from '@/pages/profile/my-companies'
import addCompanies from '@/pages/profile/add-companies'
import editCompany from '@/pages/profile/edit-company'
import orderHistory from '@/pages/profile/order-history'
import editProfile from '@/pages/profile/edit-profile'
import card from '@/pages/profile/card'
import archive from '@/pages/profile/archive'
import catalog from '@/pages/profile/catalog'

Vue.use(Router);

export default new Router({
  mode: 'history',
  routes: [
    {
      path: '/',
      name: '',
      component: Home
    },
    {
      path: '/price',
      name: 'price',
      component: Price
    },
    {
      path: '/news',
      name: 'news',
      component: News
    },
    {
      path: '/contacts',
      name: 'contacts',
      component: Contacts
    },
    {
      path: '/reg',
      name: 'reg',
      component: Registration
    },
    {
      path: '/login',
      name: 'login',
      component: Login
    },
    {
      path: '/profile',
      name: 'profile',
      component: Profile,
      children:[
        {
```

```
        path: 'checkout',
        name: 'checkout',
        component: checkout
      },
      {
        path: 'companies',
        name: 'companies',
        component: myCompanies
      },
      {
        path: 'add-companies',
        name: 'addCompanies',
        component: addCompanies
      },
      {
        path: 'edit-company/:id',
        name: 'editCompany',
        component: editCompany
      },
      {
        path: 'edit-profile',
        name: 'editProfile',
        component: editProfile
      },
      {
        path: 'order-history',
        name: 'orderHistory',
        component: orderHistory
      },
      {
        path: 'catalog',
        name: 'catalog',
        component: catalog
      },
      {
        path: 'card',
        name: 'card',
        component: card
      },
      {
        path: 'archive',
        name: 'archive',
        component: archive
      }
    ]
  }
 ]
})
```

Navigation.vue

```
<template>
 <header class="header">
```

```
<div class="container">
  <div class="row align-items-center justify-content-between">
    <div class="col-lg-2 col-sm-4 col-5">
      <a href="/" class="header__logo"></a>
    </div>
    <div class="col-lg-10 col-sm-8 col-7">
      <div id="nav-icon3" @click="menuShow = !menuShow" :class="{ open: menuShow }">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
      </div>

      <nav class="menu-mobile" :class="{ open: menuShow }">
        <ul class="menu__list menu__list_auth clearfix">
          <li
            class="menu__item menu__item_button"
            v-for="link in navAuth"
            :key="link.title"
            v-if="link.auth === auth"
            @click="menuShow = false"
          >
            <router-link
              class="button__header button__header_entry"
              :to="`${link.url}`"
            >{{ link.title }}</router-link>
          </li>
        </ul>
        <ul class="menu__list clearfix">
          <li
            class="menu__item"
            v-for="link in nav"
            :key="link.title"
            @click="menuShow = false"
          >
            <router-link class="menu__link" :to="`${link.url}`">{{ link.title }}</router-link>
          </li>
        </ul>
      </nav>

      <nav class="menu">
        <ul class="menu__list menu__list_auth clearfix">
          <li
            class="menu__item menu__item_button"
            v-for="link in navAuth"
            :key="link.title"
            v-if="link.auth === auth"
          >
            <router-link
              class="button__header button__header_entry"
              :to="`${link.url}`"
            >{{ link.title }}</router-link>
```

```
          </li>
        </ul>

        <ul class="menu__list menu__list_auth clearfix">
          <li class="menu__item menu__item_button" v-if="auth === true">
            <span class="button__header button__header_name">
              {{ personeName }}
            </span>
          </li>
          <li
            class="menu__item menu__item_button"
            v-for="link in navAuthLogin"
            :key="link.title"
            v-if="link.auth === auth"
          >
            <span class="button__header button__header_entry"
                @click="exit(link)">
              {{ link.title }}
            </span>
          </li>

        </ul>

        <ul class="menu__list clearfix">
          <li class="menu__item"
              v-for="link in nav"
              :key="link.title"
              v-if="link.auth === 'both' || link.auth === auth">
            <router-link class="menu__link" :to="`${link.url}`">{{ link.title }}</router-link>
          </li>
        </ul>

      </nav>
    </div>
  </div>
 </div>
 </header>
</template>

<script>
export default {
 name: "Navigation",
 data() {
  return {
   menuShow: false
  };
 },
 computed: {
  personeName() {
   return this.$store.getters.personeName;
  },
  auth() {
```

```
      return this.$store.getters.auth;
    },
    nav() {
      return this.$store.getters.nav;
    },
    navAuth() {
      return this.$store.getters.navAuth;
    },
    navAuthLogin() {
      return this.$store.getters.navAuthLogin;
    }
  },
  methods: {
    exit(link){
      if(link.title == 'Выход'){
        this.$store.dispatch('logout');
        this.$router.replace('/')
      }
    }
  }
};
</script>

<style scoped>
</style>
```

Profile.vue

```
<template>
  <section class="section section_profile profile">
    <div class="container-fluid">
      <div class="row no-gutter">
        <div class="col-md-3">
          <NavigationProfile></NavigationProfile>
        </div>
        <div class="col-md-9">
          <div class="profile__content-section">
            <router-view></router-view>
          </div>
        </div>
      </div>
    </div>
  </section>
</template>

<script>
  import NavigationProfile from '@/components/Navigation-profile.vue'

  export default{
    name: "profile",
    components:{
      NavigationProfile
    },
```

```
    data(){
      return{}
    }
  }
</script>
```

Footer.vue

```
<template>
  <footer class="footer">
    <div class="container">
      <div class="row">
        <div class="col-md-3">
          <ul class="footer__list">
            <li class="footer__item">
              +38 066 123 45 67
            </li>
            <li class="footer__item">
              +38 098 123 45 67
            </li>
          </ul>
        </div>
        <div class="col-md-3">
          <ul class="footer__list">
            <li class="footer__item">
              Украина
            </li>
            <li class="footer__item">
              Днепр
            </li>
            <li class="footer__item">
              ул. Столбовая, 33а
            </li>
          </ul>
        </div>
        <div class="col-md-3">
          <a href="#" class="footer__mail">
            ex@mail.com
          </a>
        </div>
        <div class="col-md-3">
          <div class="social">
            <a href="#" class="social__item social__item_instagram"></a>
            <a href="#" class="social__item social__item_twitter"></a>
            <a href="#" class="social__item social__item_fb"></a>
          </div>
        </div>
      </div>
    </div>
  </footer>
</template>
```

```
<script>
  export default {
    name: 'Footer'
  }
</script>

<style scoped>
</style>
```

Navigation-profile.vue

```
<template>
  <nav class="nav-profile">
    <ul>
      <li>
        <a href="/profile/checkout">
          Оформить заказ
        </a>
      </li>
      <li>
        <a href="/profile/order-history">
          История заказов
        </a>
      </li>
      <li>
        <router-link
                class=""
                :to="'/profile/catalog'"
              >
              Справочник изделий
        </router-link>
      </li>
      <li>
        <a href="/profile/archive">
          Архив изделий
        </a>
      </li>
      <li>
        <router-link
                class=""
                :to="'/profile/companies'"
              >
              Мои компании
        </router-link>

      </li>
      <li>
        <a href="/profile/edit-profile">
          Личные данные
        </a>
      </li>
    </ul>
```

```
  </nav>
</template>

<script>

export default {
  name: "Navigation-profile",

  data() {
    return {
      menuShow: false
    };
  },
  computed: {
    nav(){
      return this.$store.getters.nav
    },
    navAuth(){
      return this.$store.getters.navAuth
    }
  }
};
</script>

<style scoped>
</style>
```

checkout.vue

```
<template>
  <section class="section">
     <div class="container">
       <div class="row justify-content-center">
         <div class="col-xl-12 col-md-8 col-12">
           <div class="form-block">
             <div class="title title_form">Оформить заказ</div>
             <div class="form__item">
               <select v-model="newOrder.order.company_uuid" class="select">
                 <option value="">Выбрать компанию</option>
                 <option
                       v-for="(item) in allCompaniesInfo"
                       :key="item.uuid"
                       :value="item.uuid"
                 >
                 {{item.short_name}}
                 </option>
               </select>
             </div>

             <div class="boxInfo__block">
               <div class="boxInfo__item" v-for="(item, index) in selected" v-bind:key="item.uuid">
                 <div class="row align-items-center">
                   <div class="col-sm-2">
```

```
                <img src="@/assets/images/box.png" alt="">
              </div>
              <div class="col-sm-4">
                <div class="boxInfo__name">
                 {{item.name}}
                </div>
              </div>
              <div class="col-sm-2">
                <div class="boxInfo__size">
                 {{ item.width }}x{{ item.length}}x{{ item.height}}
                </div>
              </div>
              <div class="col-sm-3">
                <input type="number" v-model="item.product_quantity"> <span> шт</span>
              </div>
              <div class="col-sm-1">
                <div class="del" v-on:click="delSelected(index)">
                  <img src="@/assets/images/del.png" alt="">
                </div>
              </div>
            </div>
          </div>
          <div class="boxInfo__add-button-block" @click="show = !show">
            <div class="button button_add" >
              Добавить коробку
            </div>
            <ul v-bind:class="{open: show}" class="boxInfo__add-list">
              <li v-for="item in allBoxes" v-bind:key="item.uuid" v-on:click="checkBox(item)">
                <input type="hidden" v-bind:value="item.uuid">
                <input type="hidden" v-bind:value="item.name" >
                <input type="hidden" v-bind:value="item.width" >
                <input type="hidden" v-bind:value="item.length" >
                <input type="hidden" v-bind:value="item.height" >
                {{item.name}}
              </li>
            </ul>
          </div>
          <button v-on:click="addNewOrder" value="" class="button button_order">Оформить
заказ</button>
        </div>
      </div>
    </div>
  </div>
 </section>

</template>

<script>
export default {

  name: "checkout",
```

```
data() {
  return {
    quantity:'',
    selected: [],
    show: false,
    product_quantity: '',
    newOrder:{
      person: localStorage.person_uuid,
      order:{
        company_uuid: null,
        order_details: []
      }
    }
  }
},
methods: {
  checkBox: function(item) {
    this.selected.push({
      s_product_uuid: item.uuid,
      name: item.name,
      width: item.width,
      length: item.length,
      height: item.height
    })
  },
  async addNewOrder(){
    for(var item of this.selected){
      this.newOrder.order.order_details.push({
        product_uuid: item.s_product_uuid,
        quantity: item.product_quantity
      })
    }
    await this.$store.dispatch('addOrder', { newOrder: this.newOrder });
    this.selected = [];
    this.newOrder.order.order_details = []

  },
  delSelected: function(index){
    this.selected.splice(index, 1);
    this.newOrder.order.order_details.splice(index, 1)
  },
},
mounted() {
  this.$store.dispatch('getAllBoxes');
  this.$store.dispatch('getAllCompanies', {
    person_uuid: this.uuidPerson,
  });
},
computed: {
  allBoxes() {
    return this.$store.getters.allBoxes;
```

```
    },
    allCompaniesInfo() {
      return this.$store.getters.allCompaniesInfo;
    },
    uuidPerson() {
      return this.$store.getters.uuidPerson;
    },
  }
}
</script>

<style scoped>
</style>
```

order-history.vue

```
<template>
  <section class="section">
    <div class="container">
      <div class="row justify-content-center">
        <div class="col-xl-12 col-md-8 col-12">
          <div class="form-block">
            <div class="title title_form">История заказов</div>
            <div class="order-history">
              <div class="order-history__item" v-for="item in orderHistory" v-bind:key="item.uuid">
                <div class="order-history__block">
                  <img class="order-history__icon" src="@/assets/images/order-icon.png" alt="">
                  <div class="order-history__number">
                    Заказ № 132345
                  </div>
                  <div class="order-history__date">
                    {{item.order_date}}
                  </div>
                </div>
                <div class="order-history__estimated-date">
                  <span>Предполагаемая дата исполнения:</span> {{item.estimated_date}}
                </div>
                <div class="order-history__quantity">
                  <span>Товар в заказе:</span> 5 шт
                </div>
                <div class="order-history__price">
                  <span>Цена:</span> {{item.amount_total}}
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
</template>

<script>
```

```
export default {
  name: "orderHistory",

  data() {
    return {};
  },
  mounted() {
    this.$store.dispatch('getOrders');
  },
  computed: {
    orderHistory() {
      return this.$store.getters.orderHistory;
    },
  }
};
</script>

<style scoped>
</style>
```

catalog.vue

```
<template>
  <section class="section">
    <div class="container">
      <div class="row justify-content-center">
        <div class="col-xl-12 col-md-8 col-12">
          <div class="form-block">
            <div class="title title_form">Справочик изделий</div>
            <div class="boxInfo__block">
              <div class="boxInfo__item" v-for="item in boxesForPerson" v-bind:key="item.uuid">
                <div class="row align-items-center">
                  <div class="col-sm-2">
                    <img src="@/assets/images/box.png" alt />
                  </div>
                  <div class="col-sm-3">
                    <input type="hidden" v-bind:value="item.uuid">
                    <div class="boxInfo__name">{{ item.name }}</div>
                    <div class="boxInfo__submit">
                      Техкарта
                      <span class="text-success" v-if="item.submited">согласована</span>
                      <span class="text-danger" v-else>не согласована</span>
                    </div>
                  </div>
                  <div class="col-sm-2">
                    <div class="boxInfo__size">{{ item.width }}x{{ item.length}}x{{
item.height}}</div>
                  </div>
                  <div class="col-sm-5">
                    <button class="button button_brown" v-if="item.submited">В архив</button>
                  </div>
                </div>
              </div>
```

```
          </div>
          <div class="row">
            <div class="col-6">
              <button class="button button_registration">
                <a href="/profile/card">Оставить заявку на новую техкарту</a>
              </button>
            </div>
            <div class="col-6">
              <button class="button button_registration">
                <a href="/profile/checkout">Перейти к оформлению заказа</a>
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
</template>

<script>
export default {
  name: "catalog",

  data() {
    return {};
  },
  computed: {
    boxesForPerson() {
      return this.$store.getters.boxesForPerson;
    }
  },
  mounted() {
    this.$store.dispatch("getPersonBoxes");
  }
};
</script>

<style scoped>
</style>

card.vue
<template>
  <section class="section">
    <div class="container">
      <div class="row justify-content-center">
        <div class="col-xl-12 col-md-8 col-12">
          <div class="form-block">
            <div class="title title_form">Заявка на техкарту</div>

            <form class="form" action="#" v-if="boxInfo">
              <div class="row">
```

```
            <div class="col-md-3">
              <div class="form__item">
                <input
                  class="input form-control"
                  type="text"
                  placeholder="Название"
                  v-model="newCard.product.name"
                >
              </div>
              <div class="form__item">
                <the-mask class="input form-control" :mask="['################']" v-
model="newCard.product.length" placeholder="Длина, мм"/>

              </div>
              <div class="form__item">
                <the-mask class="input form-control" :mask="['################']" v-
model="newCard.product.width" placeholder="Ширина, мм"/>

              </div>
              <div class="form__item">
                <the-mask class="input form-control" :mask="['################']" v-
model="newCard.product.height" placeholder="Высота, мм"/>

              </div>

            </div>
            <div class="col-md-3">
              <div class="form__item">
                <select v-model="newCard.product.cardboard_uuid" placeholder="" class="select">
                  <option value="">Марка картона</option>
                  <option
                    v-for="item in boxInfo.cardboard"
                    v-bind:key="item.uuid"
                    v-bind:value="item.uuid"
                    >
                    {{item.name}}
                  </option>
                </select>
              </div>

              <div class="form__item">
                <select v-model="newCard.product.bumps_uuid" placeholder="" class="select">
                  <option value="">Марка гофры</option>
                  <option
                    v-for="item in boxInfo.bumps"
                    v-bind:key="item.uuid"
                    v-bind:value="item.uuid"
                    >
                    {{item.name}}
                  </option>
                </select>
              </div>
```

```
                <div class="form__item">
                  <select v-model="newCard.product.cardboard_color_uuid" placeholder=""
class="select">
                    <option value="">Цвет картона</option>
                    <option
                      v-for="item in boxInfo.cardboard_colors"
                      v-bind:key="item.uuid"
                      v-bind:value="item.uuid"
                      >
                      {{item.name}}
                    </option>
                  </select>
                </div>
              </div>
              <div class="col-md-4">
                <div class="form__item">
                  <select v-model="newCard.product.standart_uuid" class="select">
                    <option value="">Номер шаблона по ГОСТу</option>
                    <option
                      v-for="item in boxInfo.boxes_standarts"
                      v-bind:key="item.uuid"
                      v-bind:value="item.uuid"
                    >
                      {{item.name}}
                    </option>
                  </select>
                  <button type="button" class="btn btn-primary" @click="showModal = true">
                    Посмотреть шаблон
                  </button>
                </div>
                <form id="uploadForm" name="uploadForm" method='post' enctype="multipart/form-
data">
                  <input type="hidden" v-bind:value="uuidProduct" name="product">

                  <label class="label-file" for="logo" id="logo-text">Прикрепить логотип</label>
                  <input class="input-file" type="file" id="logo" name="logo" v-
on:change="alertFilenameLogo">

                  <label class="label-file" for="template" id="template-text">Прикрепить
чертеж</label>
                  <input class="input-file" type="file" id="template" name="template"  v-
on:change="alertFilenameTemplate">
                </form>
                <div class="form__item">

                </div>
              </div>
            </div>

          </form>
          <div class="buttons-list">
```

```
            <input type=button class="button button_registration" value=Сохранить
@click="uploadCardInfo">
          </div>
        </div>
      </div>
    </div>
    <div class="modal" :class="{ show: showModal}" v-show="showModal" @close="showModal
= false" v-if="boxInfo">
      <div class="modal-background" @click="showModal = false"></div>
      <div
        class="content"
        v-for="item in boxInfo.boxes_standarts"
        v-bind:key="item.uuid"
        v-if="newCard.product.standart_uuid == item.uuid"
      >
        <img v-bind:src="item.template_image">
      </div>
  </div>
 </section>
</template>
<script>
export default {
 name: "card",
 data() {
  return {
   newCard:{
    person: localStorage.person_uuid,
    product:{
     length: '',
     width: '',
     height: '',
     box_color: '',
     standart_uuid: '',
     cardboard_uuid: '',
     cardboard_color_uuid: '',
     bumps_uuid: '',
     name: '',
    },
   },
   showModal: false
  }
 },
 methods: {
  async uploadCardInfo(){
   await this.$store.dispatch('addNewCard', { newCard: this.newCard });
  if(this.uuidProduct != 0){
    let data = new FormData(document.getElementById('uploadForm'))
    var imageLogo = document.querySelector('#logo')
    var imageTemplate = document.querySelector('#template')
    console.log(imageLogo.files[0])
    console.log(imageTemplate.files[0])
```

```
        data.append('logo', imageLogo.files[0])
        data.append('template', imageTemplate.files[0])
        this.$store.dispatch('addImages', { data: data });
        this.newCard.product.name = "";
        this.newCard.product.length = "";
        this.newCard.product.height = "";
        this.newCard.product.width = "";
        this.newCard.product.cardboard_color_uuid = "";
        this.newCard.product.box_color = "";
        this.newCard.product.standart_uuid = "";
        this.newCard.product.cardboard_uuid = "";
        this.newCard.product.bumps_uuid = "";
      } else{
        console.log("uuid_product is absent")
      }
    },
    alertFilenameLogo() {
      var thefile = document.getElementById('logo');
      document.getElementById('logo-text').innerHTML =  thefile.value;
    },
    alertFilenameTemplate() {
      var thefile = document.getElementById('template');
      document.getElementById('template-text').innerHTML =  thefile.value;
    }
  },
  mounted() {
    this.$store.dispatch('getBoxInfo');
  },
  computed: {
    options(){
      var options= [];
      for(var item of this.boxInfo.boxes_standarts){
        options.push(item.name)
      }
      return options
    },
    boxInfo() {
      return this.$store.getters.boxInfo;
    },
    uuidProduct() {
      return this.$store.getters.uuidProduct;
    },
  }
};
</script>

<style scoped>
.modal.show{
  display: block
}
  .modal-background{
    position: fixed;
```

```
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.7)
   }
   .modal .content{
    position: absolute;
    width: 70%;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%)
   }
   .buttons-list{
    width: 245px;
   }
</style>

archive.vue
<template>
  <section class="section">
    <div class="container">
      <div class="row justify-content-center">
        <div class="col-xl-12 col-md-8 col-12">
          <div class="form-block">
            <div class="title title_form">Архив изделий</div>
            <div class="boxInfo__block">
               <div class="boxInfo__item" v-for="item in archiveBoxesForPerson" v-
bind:key="item.uuid">
                 <div class="row align-items-center">
                   <div class="col-sm-2">
                     <img src="@/assets/images/box.png" alt="">
                   </div>
                   <div class="col-sm-4">
                     <div class="boxInfo__name">
                      {{ item.name }}
                     </div>
                   </div>
                   <div class="col-sm-2">
                     <div class="boxInfo__size">
                       {{ item.width }}x{{ item.length}}x{{ item.height}}
                     </div>
                   </div>

                 </div>
               </div>
             </div>
          </div>
        </div>
      </div>
    </div>
  </section>
```

```
</template>

<script>
export default {
  name: "archive",
  data() {
    return {};
  },
  computed: {
    archiveBoxesForPerson() {
      return this.$store.getters.archiveBoxesForPerson;
    },
  },
  mounted() {
     this.$store.dispatch('getArchiveBoxes');
    },
};
</script>

<style scoped>
</style>

my-companies.vue
<template>
  <section class="section companies">
    <div class="profile__content">
      <div class="title title_profile">Мои компании</div>
      <div class="company__wrapper">
        <div
          class="company__item"
          v-for="(item) in allCompaniesInfo"
          :key="item.uuid"
        >
          <div class="company__name">
            {{item.short_name}}
          </div>
          <div class="company__block-btn">
            <router-link v-bind:to="'/profile/edit-company/' + item.uuid" class="company__link
company__link_edit"></router-link>
            <!-- <a href="/profile/edit-company" class="company__link company__link_edit" ></a> --
>
            <a href="#" class="company__link company__link_delete" v-
on:click="delCompany(item.uuid)"></a>
          </div>
        </div>
        <a href="/profile/add-companies" class="button button_add">Добавить</a>
      </div>
    </div>
  </section>
</template>
<script>
export default {
```

```
 name: "myCompanies",
 data() {
  return {
    arrayComanyInfos: null
  };
 },
 methods: {
  delCompany(company_uuid){
    this.$store.dispatch('deleteCompany', {
      person_uuid: this.uuidPerson,
      company_uuid,
    });
  }
 },
 mounted() {
  this.$store.dispatch('getAllCompanies', {
    person_uuid: this.uuidPerson,
  });
 },
 computed: {
  allCompaniesInfo() {
    return this.$store.getters.allCompaniesInfo;
  },
  uuidPerson() {
    return this.$store.getters.uuidPerson;
  },
 }
};
</script>

<style scoped>
</style>
```

add-companies.vue

```
<template>
<section class="section">
   <div class="container">
     <div class="row justify-content-center">
       <div class="col-xl-12 col-md-8 col-12">
         <div class="form-block">
           <div class="title title_form">Добавить компанию</div>
           <div class="subtitle">Заполните все обязательные поля</div>

           <form class="form" action="#" @submit.prevent="addCompany">
             <div class="row">
               <div class="col-md-6">
                 <div class="form__item">
                   <input
                     class="input form-control"
                     type="text"
                     placeholder="Короткое название"
                     v-model="newCompany.company.short_name"
```

```
  >
</div>
<div class="form__item">
  <input
    class="input form-control"
    type="text"
    placeholder="Полное название"
    v-model="newCompany.company.full_name"
  >
</div>
<div class="form__item">
  <input
    class="input form-control"
    type="number"
    placeholder="Код ЕРДПОУ"
    v-model="newCompany.company.edrpou"
  >
</div>
<div class="form__item">
  <input
    class
    type="checkbox"
    value="Плательщик НДС"
    v-model="newCompany.company.vat"
  >
  <label>Плательщик НДС</label>
</div>
<div class="form__item">
  <input
    :disabled="isDisabledEdrpou"
    class="input input_nds form-control"
    type="number"
    placeholder="номер свидетельства плательщика НДС"
    v-model="newCompany.company.vat_number"
  >
</div>
<div class="form__item">
  <input
    class="input form-control"
    type="number"
    placeholder="ИНН"
    v-model="newCompany.company.inn"
  >
</div>
<div class="form__item">
  <input
    class="input form-control"
    type="text"
    placeholder="Юридический адрес"
    v-model="newCompany.company.jur_address"
  >
</div>
```

```
              <div class="form__item">
                <input
                  class="input form-control"
                  type="text"
                  placeholder="Физический адрес"
                  v-model="newCompany.company.real_address"
                >
              </div>
            </div>
            <div class="col-md-6">
              <div class="form__item">
                <input
                  class="input form-control"
                  type="text"
                  placeholder="Адрес доставки по умолчанию"
                  v-model="newCompany.company.delivery_address"
                >
              </div>
              <div class="form__item">
                <input class type="checkbox"
                       v-model="useRealAdress"
                       v-on:change="useAddress">
                <label> Использовать физический адрес</label>
              </div>
              <div class="form__item">
                <masked-input class="input form-control" v-
model="newCompany.company.bank_account" mask="AA 11 111111 \00000 11111111111111"
placeholder="Номер банковского счета (IBAN)" />
              </div>
              <div class="form__item">
                <input
                  class="input form-control"
                  type="text"
                  placeholder="ФИО руководителя"
                  v-model="newCompany.company.director"
                >
              </div>
            </div>
          </div>
          <div class="title title_form">Контактные лица</div>
          <div class="row">
            <div class="col-md-6"
                v-for="(item, index) in newCompany.company.contacts"
                :key="item.id">
              <div class="persone-contact">
                <button type="button" class="btn btn-danger " v-on:click
="delContactPerson(index)">удалить контакт</button>

              <div class="form__item">
                <input
                  class="input form-control"
                  type="text"
```

```
            placeholder="ФИО"
            v-model="item.fio"
           >
          </div>
          <div class="form__item">
           <masked-input class="input form-control" v-model="item.phone1" mask="\+\38 (011)
111 -11-11" placeholder="Телефон 1" />

          </div>
          <div class="form__item">
           <masked-input class="input form-control" v-model="item.phone2" mask="\+\38 (011)
111 -11-11" placeholder="Телефон 2" />
          </div>
          <div class="form__item">
           <span>
            <input class type="checkbox"
                id="viber"
                v-model="newCompany.viber"
                >
            <label for="viber"> Viber</label>
           </span>
           <span>
            <input class type="checkbox"
                id="telegram"
                v-model="newCompany.telegram"
                >
            <label for="telegram"> Telegram</label>
           </span>
           <span>
            <input class type="checkbox"
                id="whatsapp"
                v-model="newCompany.whatsapp"
                >
            <label for="whatsapp"> Whatsapp</label>
           </span>
           <span>
            <input class type="checkbox"
                id="skype"
                v-model="newCompany.skype"
                >
            <label for="skype"> skype</label>
           </span>
          </div>
          <div class="form__item">
           <input
            class="input form-control"
            type="text"
            placeholder="viber"
            v-model="item.viber"
            :disabled="isDisabledViber"
           >
          </div>
```

```
            <div class="form__item">
              <input
                class="input form-control"
                type="text"
                placeholder="telegram"
                v-model="item.telegram"
                :disabled="isDisabledTelegram"
              >
            </div>
            <div class="form__item">
              <input
                class="input form-control"
                type="text"
                placeholder="whatsapp"
                v-model="item.whatsapp"
                :disabled="isDisabledWhatsapp"
              >
            </div>
            <div class="form__item">
              <input
                class="input form-control"
                type="text"
                placeholder="skype"
                v-model="item.skype"
                 :disabled="isDisabledSkype"
              >
            </div>
            </div>

          </div>
          <div class="col-md-6">
            <div class="button button_add" v-on:click="addContactPerson">
              Добавить контакт
            </div>
          </div>
        </div>
        <div class="row">
          <div class="col-12">
            <div class="buttons-list">
              <button type="submit" class="button button_registration">Сохранить</button>
            </div>
          </div>
        </div>
      </form>
    </div>
  </div>
 </div>
</section>
</template>

<script>
```

```
import MaskedInput from 'vue-masked-input'
export default {
  name: "addCompanies",
  components: {
    MaskedInput
  },
  data() {
    return {
      mfoError: false,
      newCompany: {
        person_uuid: localStorage.person_uuid,
        company: {
          short_name: null,
          full_name: null,
          edrpou: null,
          jur_address: null,
          real_address: null,
          delivery_address: null,
          director: null,
          bank_account: null,
          bank_uuid: null,
          inn: null,
          vat: null,
          vat_number: null,
          contacts: [ ],
        },
        viber: null,
        telegram: null,
        whatsapp: null,
        skype: null,
      },
      mfo: null,
      useRealAdress: false,
      arrayMfo: [],
      fio: null,
      nextContact: 0
    };
  },
  methods: {
    addContactPerson(){
      this.newCompany.company.contacts.push({
        id: this.nextContact++,
        fio: '',
        phone1: '',
        viber: '',
        telegram: '',
        whatsapp: '',
        skype: ''
      })
    },
    delContactPerson: function(x){
      this.newCompany.company.contacts.splice(x, 1);
```

```
    },
    getMfo(){
      if(this.mfo.length > "5" && this.mfo.length<"7"){
        this.$store.dispatch('getMfo', {mfo: this.mfo});
        this.newCompany.company.bank_uuid = this.mfoInfo.uuid;
        this.mfoError = false
      } else{
        this.mfoError = true
      }
    },
   addCompany() {
      this.$store.dispatch('addNewCompany', { newCompany: this.newCompany, $router:
this.$router });
    },

    useAddress(){
      if(this.useRealAdress === true){
        this.newCompany.company.delivery_address = this.newCompany.company.real_address
      }
    }
  },
  computed: {
    isDisabledEdrpou(){
      if (this.newCompany.company.vat === true){
        return false
      } else {
        return true
      }
    },
    isDisabledViber(){
      if (this.newCompany.viber){
        return false
      } else {
        return true
      }
    },
    isDisabledTelegram(){
      if (this.newCompany.telegram){
        return false
      } else {
        return true
      }
    },
    isDisabledWhatsapp(){
      if (this.newCompany.whatsapp){
        return false
      } else {
        return true
      }
    },
    isDisabledSkype(){
      if (this.newCompany.skype){
```

```
        return false
      } else {
        return true
      }
    },
    mfoInfo() {
      return this.$store.getters.mfoInfo;
    },
  },
};
</script>

<style scoped>
.input:disabled{
  display: none
}
.input_nds:disabled{
  display: block !important
}
.mfoError{
  position: absolute;
  color: red;
  top: -25px
}
.mfoerror{
  border: 1px solid red
}

.persone-contact{
  position: relative;
}
.persone-contact .btn-danger{
  position: absolute;
  top: -45px;
  right: 0;
}
</style>

edit-company.vue
<template>
  <section class="section section_registration">
    <div class="container">
      <div class="row justify-content-center">
        <div class="col-xl-12 col-md-8 col-12">
          <div class="form-block">
            <div class="title title_form">Изменить компанию</div>

            <form class="form" action="#" @submit.prevent="" v-if="companyInfo">
              <div class="row">
                <div class="col-md-6">
                  <div class="form__item">
                    <input
```

```
    class="input form-control"
    type="text"
    placeholder="Короткое название"
    v-model="companyInfo.short_name"
   >
  </div>
  <div class="form__item">
   <input
    class="input form-control"
    type="text"
    placeholder="Полное название"
    v-model="companyInfo.full_name"
   >
  </div>
  <div class="form__item">
   <input

    class="input form-control"
    type="number"
    placeholder="Код ЕРДПОУ"
    v-model="companyInfo.edrpou"
   >
  </div>
  <div class="form__item">
   <input
    class
    type="checkbox"
    value="Плательщик НДС"
    v-model="companyInfo.vat"
   >
   <label>Плательщик НДС</label>
  </div>
  <div class="form__item">
   <input

    class="input input_nds form-control"
    type="number"
    placeholder="номер свидетельства плательщика НДС"
    v-model="companyInfo.vat_number"
   >
  </div>
  <div class="form__item">
   <input
    class="input form-control"
    type="number"
    placeholder="ИНН"
    v-model="companyInfo.inn"
   >
  </div>
  <div class="form__item">
   <input
    class="input form-control"
```

```
      type="text"
      placeholder="Юридический адрес"
      v-model="companyInfo.jur_address"
    >
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="Физический адрес"
      v-model="companyInfo.real_address"
    >
  </div>
</div>
<div class="col-md-6">
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="Адрес доставки по умолчанию"
      v-model="companyInfo.delivery_address"
    >
  </div>


  <div class="form__item">
    <input
      class="input form-control"
      type="number"
      placeholder="Номер банковского счета"
      v-model="companyInfo.bank_account"
    >
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="number"
      placeholder="Код МФО банка"
      v-model="companyInfo.mfo"

    >
  </div>
  <!-- <div class="form__item">
    <div class="input form-control">{{ arrayMfo.bank_name }}</div>
  </div> -->
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="ФИО руководителя"
      v-model="companyInfo.director"
    >
```

```
            </div>
          </div>
        </div>
        <div class="title title_form">Контактные лица</div>
        <div class="row" >
          <div class="col-md-6" v-for="(item, index) in companyInfo.contacts" :key="item.id">
           <div class="persone-contact">
            <button type="button" class="btn btn-danger " v-on:click
="delContactPerson(index)">удалить контакт</button>
             <div class="form__item">
             <input
               class="input form-control"
               type="text"
               placeholder="ФИО"
               v-model="item.fio"
             >
            </div>
            <div class="form__item">
             <input
               class="input form-control"
               type="tel"
               placeholder="Телефон 1"
               v-model="item.phone1"
             >
            </div>
            <div class="form__item">
             <input
               class="input form-control"
               type="tel"
               placeholder="Телефон 2"
               v-model="item.phone2"
             >
            </div>

            <div class="form__item">
             <input
               class="input form-control"
               type="text"
               placeholder="viber"
               v-model="item.viber"
             >
            </div>
            <div class="form__item">
             <input
               class="input form-control"
               type="text"
               placeholder="telegram"
               v-model="item.telegram"
             >
            </div>
            <div class="form__item">
             <input
```

```
                  class="input form-control"
                  type="text"
                  placeholder="whatsapp"
                  v-model="item.whatsapp"
                 >
                </div>
                <div class="form__item">
                 <input
                  class="input form-control"
                  type="text"
                  placeholder="skype"
                  v-model="item.skype"
                 >
                </div>
               </div>
              </div>
              <div class="col-md-6">
               <div class="button button_add" v-on:click="addContactPerson">
                Добавить контакт
               </div>
              </div>
             </div>
             <div class="row">
              <div class="buttons-list">
               <button type="submit" class="form__btn form__btn_registration" v-
on:click="saveCompany">Сохранить</button>
              </div>
             </div>
           </form>
          </div>
        </div>
      </div>
    </div>
  </section>
</template>

<script>
export default {
  name: "editCompany",
  data() {
    return {
      short_name: null,
      id: this.$route.params.id,
      person_uuid: localStorage.person_uuid,
      nextContact: 0
    };
  },
  created(){
    this.$store.dispatch('getCompany', {id: this.id, person_uuid: this.person_uuid})
  },
  methods: {
    addContactPerson() {
```

```
      this.companyInfo.contacts.push({
        id: this.nextContact++,
        fio: "",
        phone1: "",
        viber: "",
        telegram: "",
        whatsapp: "",
        skype: ""
      });
    },
    delContactPerson: function(x){
      this.companyInfo.contacts.splice(x, 1);
    },
    saveCompany() {
      this.$store.dispatch('updateCompany', { company: this.companyInfo, $router: this.$router });
      //this.$store.dispatch('saveChangesCompany', { companyInfo: this.companyInfo, $router:
this.$router });
    },
  },
  computed: {
    companyInfo() {
      return this.$store.getters.companyInfo;
    },
    isDisabledEdrpou() {
      if (this.companyInfo.vat === true) {
        return false;
      } else {
        return true;
      }
    }
  }
};
</script>

<style scoped>
.input:disabled {
  display: none;
}
.input_nds:disabled {
  display: block !important;
}
.persone-contact{
  position: relative;
}
.persone-contact .btn-danger{
  position: absolute;
  top: -45px;
  right: 0;
}
</style>
```

edit-profile.vue

```
<template>
<section class="section">
   <div class="container">
    <div class="row justify-content-center">
     <div class="col-xl-7 col-md-8 col-12">
      <div class="form-block">
       <div class="title title_form">Личные данные</div>

       <form class="form" action="#" @submit.prevent="">

        <div class="row">
         <div class="col-12"
          >
         <div class="form__item">
          <input
           class="input form-control"
           type="text"
           placeholder="ФИО"
           v-model="fio"
          >
         </div>
         <div class="form__item">
          <input
           class="input form-control"
           type="tel"
           placeholder="Телефон 1"
           v-model="phone1"
          >
         </div>
         <div class="form__item">
          <input
           class="input form-control"
           type="tel"
           placeholder="Телефон 2"
           v-model="phone2"
          >
         </div>
         <div class="form__item">
          <span>
           <input class type="checkbox"
                id="viber"
                v-model="checkViber"
                >
           <label for="viber"> Viber</label>
          </span>
          <span>
           <input class type="checkbox"
                id="telegram"
                v-model="checkTelegram"
                >
           <label for="telegram"> Telegram</label>
          </span>
```

```
    <span>
      <input class type="checkbox"
          id="whatsapp"
          v-model="checkWhatsapp"
          >
      <label for="whatsapp"> Whatsapp</label>
    </span>
    <span>
      <input class type="checkbox"
          id="skype"
          v-model="checkSkype"
          >
      <label for="skype"> skype</label>
    </span>
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="viber"
      v-model="viber"
      :disabled="isDisabledViber"
    >
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="telegram"
      v-model="telegram"
      :disabled="isDisabledTelegram"
    >
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="whatsapp"
      v-model="whatsapp"
      :disabled="isDisabledWhatsapp"
    >
  </div>
  <div class="form__item">
    <input
      class="input form-control"
      type="text"
      placeholder="skype"
      v-model="skype"
       :disabled="isDisabledSkype"
    >
  </div>
</div>
```

```
            </div>
            <div class="row">
              <div class="buttons-list">
                <button type="submit" class="form__btn
form__btn_registration">Сохранить</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
 </section>
</template>

<script>
export default {
 name: "editProfile",
 data() {
  return {
   fio: null,
   phone1: null,
   phone2: null,
   checkViber: null,
   checkTelegram: null,
   checkWhatsapp: null,
   checkSkype: null,
   viber: null,
   telegram: null,
   whatsapp: null,
   skype: null,
   nextContact: 0
  };
 },
 methods: {
  addContactPerson(){
   this.newCompany.company.contacts.push({
    id: this.nextContact++,
    fio: '',
    phone1: '',
    viber: '',
    telegram: '',
    whatsapp: '',
    skype: ''
   })
  },
  getMfo(){
   const axios = require("axios");
   if(this.mfo.length > "5" && this.mfo.length<"7"){
    axios
     .get("http://test1.iti.dp.ua/api/bank/?mfo=" + this.mfo)
     .then(response => {
```

```
          (this.arrayMfo = response.data);
          this.newCompany.company.bank_uuid = this.arrayMfo.uuid
        })
      }
    },
    getCompany(){
      const axios = require("axios");
       axios
         .get("http://test1.iti.dp.ua/api/bank/?mfo=" + this.mfo)
         .then(response => {
          (this.arrayMfo = response.data);
          this.newCompany.company.bank_uuid = this.arrayMfo.uuid
        })
    },
    addCompany() {
      this.$store.dispatch('addNewCompany', { newCompany: this.newCompany, $router:
this.$router });
    },
    useAddress(){
      if(this.useRealAdress === true){
        this.newCompany.company.delivery_address = this.newCompany.company.real_address
      }
    }
  },
  computed: {
    isDisabledEdrpou(){
      if (this.newCompany.company.vat === true){
        return false
      } else {
        return true
      }
    },
    isDisabledViber(){
      if (this.checkViber){
        return false
      } else {
        return true
      }
    },
    isDisabledTelegram(){
      if (this.checkTelegram){
        return false
      } else {
        return true
      }
    },
    isDisabledWhatsapp(){
      if (this.checkWhatsapp){
        return false
      } else {
        return true
      }
```

```
    },
    isDisabledSkype(){
      if (this.checkSkype){
        return false
      } else {
        return true
      }
    }
  }
};
</script>

<style scoped>
.input:disabled{
  display: none
}
.input_nds:disabled{
  display: block !important
}
</style>
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи