Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота
бакалавра
з теми: «**РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПОЛІГРАФІЧНОГО ВІДДІЛУ ПІДПРИЄМСТВА**»

Виконав:
студент 3 курсу, групи KNms1-B19
спеціальності 122 Комп’ютерні науки
Нарольський Денис В’ячеславович

Керівник:
Федорчук В. А.,
доктор технічних наук, професор,
професор кафедри комп’ютерних наук

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП

З появою веб-технологій комп'ютер починають використовувати абсолютно нові верстви населення Землі. Можна виділити дві найбільш характерні групи, що знаходяться на різних соціальних полюсах, які були стрімко залучені в нову технологію, можливо, навіть крім їх власного бажання. З одного боку, це були представники елітарних груп суспільства - керівники крупних організацій, президенти банків, топ-менеджери, впливові державні чиновники і так далі. З іншого боку, це були представники найширших верств населення - домогосподарки, пенсіонери, діти.

При появі технології веб, комп'ютери обернулися лицем до цих двох абсолютно протилежних категорій потенційних користувачів. Еліту об'єднувала одна межа – через високу відповідальність і практично стовідсоткову зайнятість “великі люди” ніколи не користувалися комп'ютером; типовою була ситуація, коли з комп'ютером працював секретар. У якийсь момент часу вони зрозуміли, що комп'ютер їм може бути корисний, що вони можуть результативно використовувати той невеликий час, який можна виділити на роботу за комп'ютером. Вони раптом зрозуміли, що комп'ютер - це не просто модна і дорога іграшка, але інструмент отримання актуальної інформації для бізнесу. При цьому їм не потрібно було витрачати багато часу, щоб освоїти технологію роботи з комп'ютером (в порівнянні з тим, як це було раніше).

З кожним новим днем технології розвиваються, і старим технологіям які раніше відповідали всім стандартам та якістю, вже не під силу впоратись з задачами теперішнього часу [2, 5], і тому люди почали придумувати нові можливості для розробки веб-сайтів, і таким чином появилися чи мало нових технологій.

Одна із технологій якраз таких є фреймворки, з появою яких стало набагато зручніше там простіше розробляти веб-сайти, фреймворки вирішують проблеми, з якими не міг впоратися звичайний JavaScript.

Є багато проблем які вирішуються фреймворками, але я приведу саме основні проблеми які вони вирішують:

- фреймоврк дозволяє досягти простоти супроводжуванності проекту;
- можливість реалізації будь-яких бізнес-процесів, а не лише тих що були спочатку закладені в систему;
- можливості легко реалізувати SPA (single page application);
- використання фреймворку зобов'язує мати структуру додатку;
- фреймворк має менший код програми, і він стає «чистішим», що позитивно відображається на швидкості розробки, а також підтримці і усуненні помилок у коді;
- наявність структури, має на увазі модульність програми, а це дає можливість простіше працювати над додатком кільком розробникам одночасно.

Об'єктом дослідження є процеси створення інформаційної системи поліграфічного відділу підприємства на основі застосування сучасних технологій розробки веб-сайтів.

Предметом дослідження є застосування фреймворку React для створення інформаційної системи поліграфічного відділу підприємства.

Метою дослідження є створення повнофункціональної інформаційної системи для поліграфічного відділу підприємства на основі застосування сучасних технологій розробки веб-сайтів.

Для досягнення поставленої мети необхідно виконати такі **завдання дослідження:**

- з'ясувати основні технічні характеристики інформаційної системи та провести опис задачі вибору інструментальних засобів;
- розробити структуру інформаційної системи та її алгоритм функціонування;
- провести програмний код та тестування інформаційної системи;
- обґрунтувати економічну складову інформаційної системи.

РОЗДІЛ 1. ТЕХНІЧНІ ХАРАКТЕРИСТИКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ОПИС ЗАДАЧІ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

1.1. Постановка задачі

Перед автором було поставлено перше завдання: розробити структурну схему проекту веб-сайту для використання в середовищі Internet. На мою думку, веб-сайт, що розробляється, повинен володіти наступними особливостями:

- гнучкістю, зручністю використання та швидкістю роботи веб-сайту;
- веб-сайт повинен надавати можливість реєстрації користувачів та їх авторизації, та форму для замовлення.
- веб-сайт повинен надавати користувачам можливість введення інформації через форми для замовлення різної поліграфічної продукції

Вимоги до інформаційної системи можна сформулювати такі:

Були поставленні наступні вимоги:

- надання повної інформація про діяльність поліграфічного підприємства;
- опис продукції підприємства;
- наявність контактної інформації;
- зручність елементів управління і меню, переходів, відкривання лінків;
- читабельність інформації, дотримання належного стилю її викладу;
- використання зрозумілих заголовків та ключових слів;
- легкість пошуку інформації;
- простота і швидкість реєстрації;
- оригінальність та можливість застосування творчого підходу до розвитку веб-сайту, реалізація інноваційності і креативності;
- простота, вишуканість дизайну;
- наочність і зрозумілість викладеної інформації;
- створення багатомовного сайту.

1.2. Найкращі JavaScript-фреймворки та тенденції веб-розробки

JavaScript залишається на вершині. Багато веб-розробників визнають, що JavaScript має недоліки і складнощі. Однак це, як і раніше, найбільш популярна мова програмування [16, 19]. В результаті проведеного на Stack-Overflow в 2020 опитування 69,7% з 47 184 опитаних професійних розробників віддали перевагу JavaScript. (див. рис. 1.1) [9].

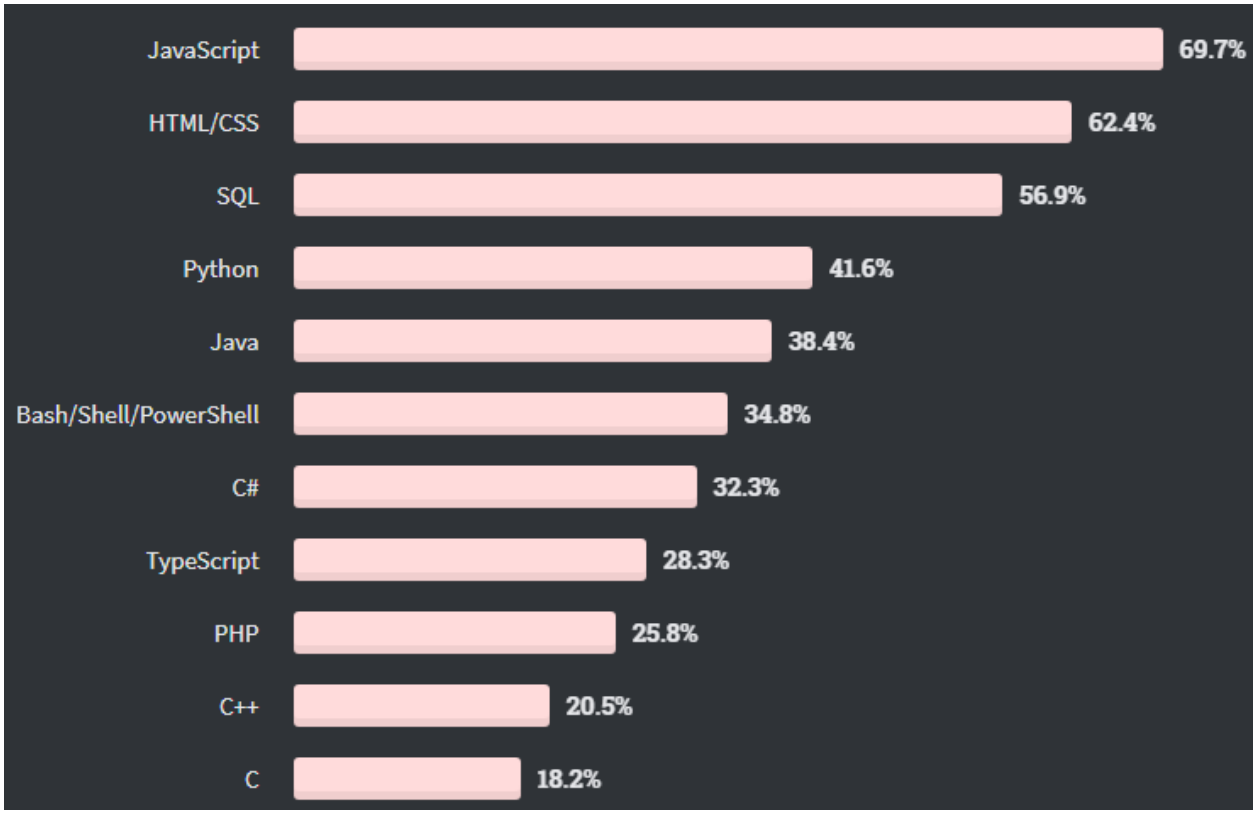

Рисунок 1.1 – Тенденції веб-розробки

Зрозуміло, що JavaScript не є ідеальним. І все ж таки не можна не відзначити його багату екосистему з великою кількістю фреймворків, бібліотек та інших корисних інструментів, а також величезне співтовариство розробників JS. До того ж існує ще й спеціальний технічний комітет, який працює над способами поліпшення JavaScript. Виходить, що в найближчому майбутньому витіснити JavaScript з лідируючих позицій у веб-розробці практично неможливо. Хоча TypeScript, ймовірно, наблизиться і стане ще привабливішою альтернативою.

TypeScript. Його часто називають покращеною версією JavaScript, і на те є всі підстави. TypeScript використовує всі сильні сторони JavaScript (адже він компілюється в JS) і поєднує їх із власними потужними функціональними можливостями, такими як статична типізація, підтримка модулів та інтерфейсів тощо. Учасники опитування на Stack Overflow назвали Typescript (67,1%) найулюбленішою мовою програмування одразу після Rust (86,1%). Згідно з останнім звітом від GitHub, JavaScript зараз теж лідирує за популярністю. Але при цьому звіт свідчить про швидке зростання затребуваності TypeScript в останні роки. (див. рис. 1.2) [9].

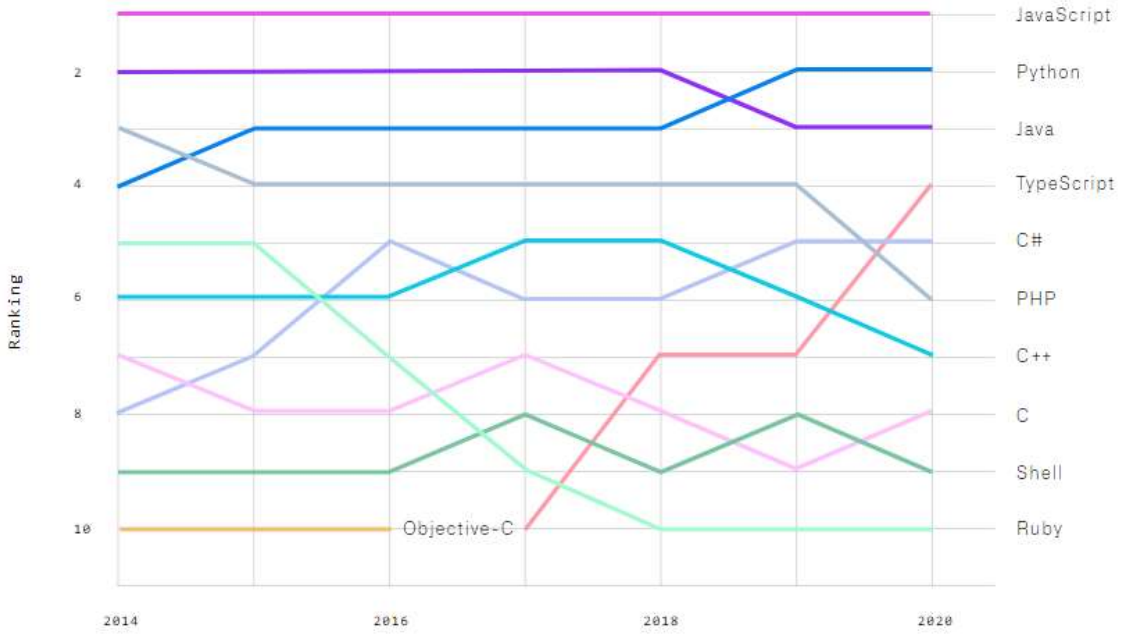

Рисунок 1.2 – Демонстрація росту популярності TypeScript

JavaScript-фреймворки. Ми вже звикли бачити серед провідних JavaScript-фреймворків React, Angular та Vue.js. Звіт про стан фронтенду State of Frontend 2020 не відкрив нічого нового у перевагах більш ніж 4500 професійних фронтенд-розробників, які брали участь в опитуванні.(див. рис. 1.3) (див. рис. 1.4) [9].

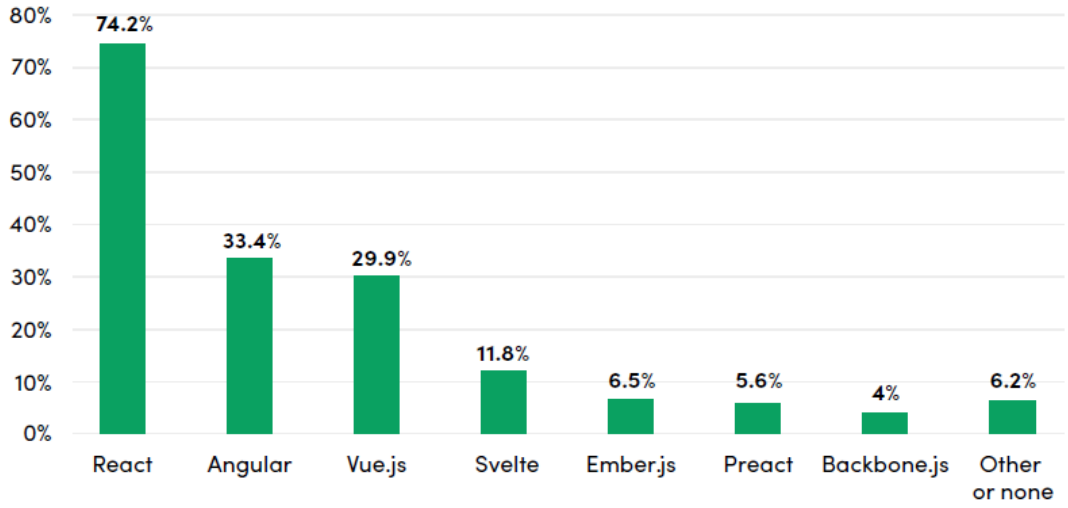

Рисунок 1.3 – Рейтинг використання фреймворків за останній рік

Але якщо подивитися на те, які фреймворки вони хочуть використовувати або вивчати для своїх майбутніх проектів, то в трійці лідерів виявиться нове ім'я [9].

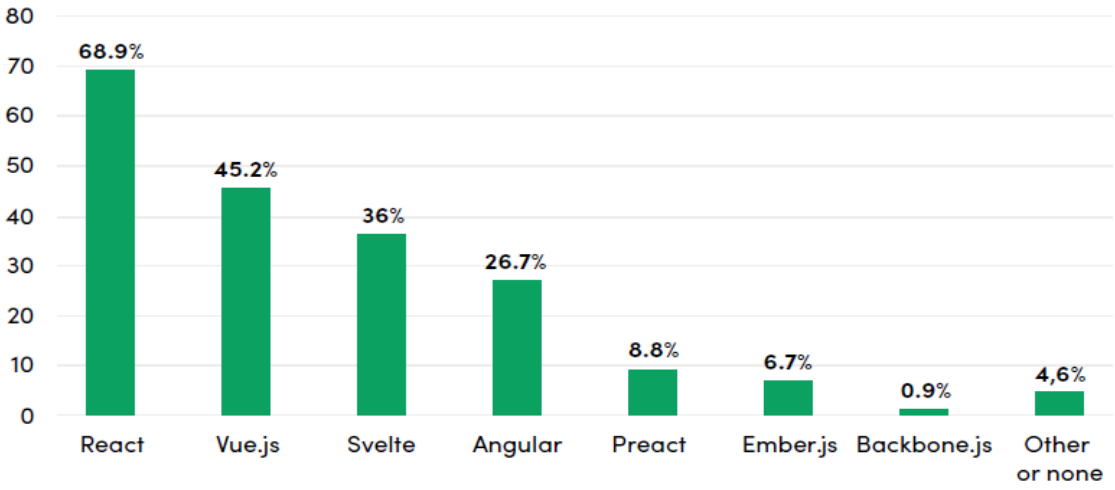

Рисунок 1.4 – Рейтинг фреймворків яким віддають перевагу

На основі вище сказаного можна зробити висновок, що фреймворки набувають все більше популярності серед розробників, хоча кожен із фреймворків має свої переваги та недоліки, але з опитувань користувачів зрозуміло що на першому місці стоїть **React**.

1.3. Технологія створення сайту

Розробка дизайну. Веб-дизайнери розробляють макети веб-шаблонів сторінок. Дизайнер визначає, яким чином кінцевий споживач отримуватиме доступ до інформації та послуг сайту — тобто, займається безпосередньо розробкою інтерфейсу користувача. У більшості випадків сторінки включають графічні елементи. Їхньою підготовкою займаються художники, ілюстратори, фотографи, технічні дизайнери, шрифтовики тощо. Готові шаблони показуються замовнику. У цей момент сторінки ще не можуть містити кінцевого наповнення (це обов'язки дизайнера не входить). Щоб макети виглядали більш наочно, у них міститься довільний вміст. На сленгу дизайнерів такий вміст називається рибою. Якщо замовник задоволений зовнішнім виглядом шаблонів, настає наступна фаза розробки — верстка сторінок сайту.

Верстка. Верстальник отримує макети шаблонів у вигляді простих зображень (наприклад, у форматі JPEG або PNG) або розбитих по шарах (наприклад, PSD або AI). Його завдання — отримати із цих графічних макетів гіпертекстові веб-сторінки з підготовленими для інтернету зображеннями. Одним із складних моментів у роботі верстальника є забезпечення сумісності з безліччю браузерів - програмами для перегляду веб-сторінок (так звана кросбраузерність). Браузери можуть одні й самі елементи розмітки чи правила CSS інтерпретувати по-своєму, у результаті деякі користувачі можуть побачити вміст негаразд, як задумував дизайнер і очікує побачити замовник. Коли верстальник переконається, що більшість браузерів однаково відображають готові шаблони, настає наступна фаза розробки веб-програмування.

Програмування. До програмістів надходять готові шаблони сторінок та вказівки дизайнерів щодо роботи та організації елементів сайту. Програміст створює програмну основу сайту, роблячи її з нуля, використовуючи

фреймворк чи CMS. Вибір мови програмування в даному випадку – питання непринципове. Після того, як сайт готовий до експлуатації, настає наступна фаза – наповнення сайту інформацією.

Використання HTML та програмних систем для розробки. При появі стандарту HTML, цей метод був найпоширенішим. Основною програмою для розробки був Notepad. Але у цього методу є істотні недоліки. Цей спосіб досить трудомісткий. І до того ж зробити нормальний Web-сайт без CSS, JavaScript та інших мов програмування досить важко. Цей підхід Ви вивчали раніше. Хотілось би звернути Вашу увагу на 2 наступних питання HTML5 та Canvas.

HTML5. це нова специфікація мови розмітки, що використовується в створенні веб-сторінок. На відміну від попередніх версій це не просто специфікація мови для гіпертекстової розмітки, а набір різнопланових модулів - від HTML-елементів до відео-, аудіо-, векторної графіки SVG, растрової JavaScript-графіки Canvas, локальних баз даних і навіть різних API браузера. Весь цей список модулів дозволяє HTML5 успішно конкурувати з технологіями Flash і Silverlight. Причому успішна конкуренція з Flash можлива ще й тому, що HTML5 потенційно набагато менше навантажує процесор комп'ютера, ніж Flash, не вимагає установки плагінів і оновлень, а значить, менш вразливий для хакерських атак. Фактично саме поява в HTML5 нових тегів <video> та <audio> робить його потенційним конкурентом існуючих технологій від Adobe і Microsoft.

Нова розмітка. HTML 5 вводить кілька нових елементів і атрибутів. Деякі з них технічно є еквівалентами і , але мають своє семантичне значення, наприклад <nav> (навігаційна панель) і <footer>. Ці теги будуть полегшувати роботу користувачам при пошуку, а також обробку сайту. Інші елементи надають нову функціональність, такі як <audio> і <video>. Деякі застарілі елементи HTML 4, такі як <font> і , були видалені з HTML 5.

Приклад веб-сторінки HTML 5

```
<! DOCTYPE html>
 <Html>
  <Head>
   <Title> Example HTML 5 document </ title>
  </ Head>
  <Body>
   <Header> ... </ header>
   <Nav> ... </ nav>
   <Section>
      <Article>
         ...
      </ Article>
   </ Section>
   <Aside> ... </ aside>
   <Footer> ... </ footer>
 </ Body>
 </ Html>
```

Відмінності від HTML 4

Основні відмінності HTML 5 від HTML 4 складають:

- Нові правила лексичного розбору;
- Нові елементи - header, footer, section, article, video, audio, progress, nav, meter, time, aside, canvas;
- Нові типи input-елементів;
- Нові атрибути;
- Глобальні атрибути - id, tabindex, repeat;
- Застарілі елементи прибрані - center, font, strike.

Нові API. Крім визначення розмітки, в HTML 5 визначені application programming interface, API. Існуючі інтерфейси DOM (Document Object Model - «об'єктна модель документа») розширені, також були додані нові API:

- малювання 2D-картиноквреальномучасі;
- контроль над відтворенням медіа файлів, який може використовуватися, наприклад, для синхронізації субтитрів з відео;
- зберігання даних у браузері;

- редагування;
- drag-and-drop;
- робота з мережею.

Обробка помилок. HTML 5-сумісні браузери дуже гнучкі при обробці помилок, на відміну від XHTML. HTML 5 розроблений так, що не підтримують його браузери можуть спокійно ігнорувати елементи HTML 5. На відміну від четвертої, п'ята версія чітко прописує правила лексичного розбору, щоб різні браузери відображали один і той же результат в разі некоректного синтаксису.

Canvas (англ. Canvas - «полотно») – елемент HTML5, призначений для створення растрового двомірного зображення за допомогою скриптів, зазвичай на мові JavaScript. Початок відліку блоку знаходиться зліва зверху. Від нього і будується кожен елемент блоку. Використовується, як правило, для відтворення графіків для статей та ігрового поля в деяких браузерних іграх.

1.4. Інструментальні засоби для створення сайтів

Створення за допомогою конструктора сайтів. Даний спосіб вважається менш витратним. Більшість конструкторів розповсюджуються на безкоштовній основі і часто надають послуги хостингу, онлайн-каси та ін.

Найвищими у списку популярності конструкторами є:

- **Tilda** — найпопулярніший умовно-безкоштовний конструктор на російському ринку. Створення кредиту на Тільда передбачає можливість широких настроїв та інтеграції.
- **Wix** — конструктор із простою функціональністю та інтуїтивно зрозумілим інтерфейсом. Часто використовується для розробки бюджетних рішень, однак має досить багато обмежень.
- **Bitrix24** — система, легко інтегруюча веб-сайт із CRM Битрикс24. Конструктор Bitrix24 SEO-оптимізований, добре індексується і зручний в налаштуванні. Це ідеальний варіант для початку комерційних проектів і початку продажу. Основні переваги використання конструкторів — зручний інтерфейс, простота налаштування. Маса готових шаблонів і безкоштовний хостинг дозволяють швидко запустити проект в мережа — функціональність конструкторів, які дозволяють швидко створити сайт навіть новачкам. Більше того, для невеликої доплати більшість конструкторів пропонують додаткові опції та функції для зростання та розширення проекту. З недоліків конструкторів слід виділити обмеженість настроїв у безкоштовних версіях. Більшість безкоштовних рішень пропонують досить типові параметри, не дозволяють перенести розроблений сайт на власний домен. Окремим мінусом варто відзначити відсутність доступу до коду. Це вимагає притягнення програмістів після перенесення ресурсу з конструктора на свій домен і хостинг.

Створення сайту на CMS-системі. Система управління – універсальне рішення, що дозволяє запустити проект швидко, недорого та з необхідною функціональністю. За фактом CMS є програмним забезпеченням, що включає двигун і базові функції, що дозволяє працювати зі структурою та контентом. Оптимальними системами управління для комерційних проектів є 1С-Бітрікс, WordPress, Drupal [12].

Переваги використання CMS:

- Ергономічність і простота управління - готовий двигун має інтуїтивно зрозумілий інтерфейс і зручну навігацію.
- Функціональність та багато налаштувань — за допомогою базових функцій CMS можна реалізувати будь-яку структуру веб-сайту. Крім того, ядро CMS передбачає просту інтеграцію з більшістю сторонніх IT-продуктів.
- Оптимізованість - ресурс на базі CMS, як правило, вже має технічну та пошукову оптимізацію, добре ранжується в пошукових системах. Недолік сайту на CMS - або дорога ліцензія, або слабко опрацьований двигун, який може бути вразливим.

Оптимальним співвідношенням ціна-якість вважається CMS 1С-Бітрікс. Створення сайту на Бітрікс вимагає придбання платної ліцензії, проте рішення пропонує велику функціональність, швидку технічну-підтримку та простоту управління проектом після релізу.

1.5. Особливості використання фреймворків у веб-розробці

Фреймворки – це програмні продукти, які спрощують створення та підтримку технічно складних чи навантажених проектів. Фреймворк, зазвичай, містить лише базові програмні модулі, проте специфічні для проекту компоненти реалізуються розробником з їхньої основі. Тим самим досягається не лише висока швидкість розробки, а й велика продуктивність та надійність рішень.

Веб-фреймворк – це платформа для створення сайтів та веб-застосунків, що полегшує розробку та об'єднання різних компонентів великого програмного проекту. За рахунок широких можливостей у реалізації бізнес-логіки та високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків та веб-сервісів.

Основні переваги фреймворків полягають у економічній ефективності та доцільності використання фреймворків. З точки зору бізнесу розробка на фреймворку майже завжди економічно ефективніша і якісніша за результатом, ніж написання проекту чистою мовою програмування без використання будь-яких платформ.

Розробка без використання платформи може бути правильним рішенням лише у двох випадках - або проект дуже простий і не потребує подальшого розвитку, або дуже навантажений і вимагає дуже низькорівневої оптимізації (наприклад, веб-сервіси з десятками тисяч звернень за секунду). У всіх інших випадках розробка на програмній платформі швидше та якісніше.

Якщо порівнювати фреймворки з іншими класами платформ — SaaS, CMS або CMF — то фреймворки значно ефективніше використовувати у проектах зі складною бізнес-логікою та високими вимогами до швидкості роботи, надійності та безпеки. Але в найпростіших і типових проектах без значних вимог швидкість і вартість розробки на фреймворку буде вищою, ніж на SaaS або CMS.

Технічні переваги фреймворків. Однією з головних переваг використання фреймворків і те, що фреймворк визначає уніфіковану

структуру для побудованих з його основі додатків. Тому програми на фреймворках значно простіше супроводжувати і доопрацьовувати, оскільки стандартизована структура організації компонентів зрозуміла всім розробникам на цій платформі і не потрібно довго розбиратися в архітектурі, щоб зрозуміти принцип роботи програми або знайти місце реалізації того чи іншого функціоналу.

Більшість фреймворків для розробки веб-застосунків використовує парадигму MVC (модель-представлення-контролер) — тобто дуже у багатьох фреймворках ідентичний підхід до організації компонентів програми і це ще більше спрощує розуміння архітектури програми навіть на незнайомому розробнику фреймворку. Проектування архітектури ПЗ при розробці на фреймворку теж дуже спрощується — у методологіях фреймворків зазвичай закладено найкращі практики програмної інженерії і просто дотримуючись цих правил можна уникнути багатьох проблем та помилок у проектуванні.

По суті, фреймворк — це безліч конкретних та абстрактних класів, пов'язаних між собою та впорядкованих згідно з методологією фреймворку. Конкретні класи зазвичай реалізують взаємні відносини між класами, а абстрактні класи є точки розширення, в яких закладений у фреймворк базовий функціонал може бути використаний «як є» або адаптований під завдання конкретного додатка. Для забезпечення розширення можливостей більшості фреймворків використовуються техніки об'єктно-орієнтованого програмування: наприклад, частини програми можуть успадковуватися від базових класів фреймворку або окремі модулі можуть бути підключені як домішки.

Системи веб-фреймворків також багаті на рішення, що готові до реалізації багатьох функціональних можливостей. Розробникам під час роботи над типовими завданнями не треба «винаходити велосипеди», оскільки вони можуть скористатися вже створеною спільнотою реалізацією. А це не тільки скорочує витрати часу і грошей, а й дозволяє досягти вищої стабільності рішення — компонент, який використовується і допрацьовується тисячами

інших розробників, зазвичай, більш якісно реалізований і краще протестований на всіляких сценаріях, ніж рішення, яке може в адекватні терміни розробити один розробник чи навіть невелика команда.

Фреймворки – це не бібліотеки. Бібліотека – це найпростіший компонент архітектури програмного забезпечення. Програмна бібліотека може бути використана просто як набір підсистем близької функціональності, не впливаючи на архітектуру основного програмного продукту та не накладаючи на неї жодних обмежень. А Фреймворк не просто дає розробнику потрібний функціонал, але ще й диктує правила побудови архітектури додатка, задаючи на початковому етапі розробки поведінку за умовчанням, формуючи каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог. Фреймворк також може включати допоміжні програми, бібліотеки коду, мову сценаріїв та інше програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекту.

Порівняння з альтернативами. Перед веб-розробниками часто стоїть вибір між коробковими CMS та фреймворками для реалізації проекту. У кожного з підходів є свої переваги та недоліки, нижче ми розглянемо переваги та недоліки розробки на фреймворках.

Переваги фреймворків:

- розробка на фреймворку (на відміну від самописних рішень) дозволяє досягти простоти супроводжуваності проекту;
- можлива (і відносно проста) реалізація будь-яких бізнес-процесів, а не лише тих, що спочатку закладені в систему. Також проекти на базі фреймворків легко масштабуються та модернізуються;
- рішення на фреймворках, як правило, працюють значно швидше і витримують велике навантаження, ніж CMS та самописні системи. Саме тому багато популярних інтернет-магазинів працюють не на коробкових CMS, а на фреймворках. За рівнем безпеки рішення на фреймворках значно перевершують самописні системи і можна порівняти з CMS (зазвичай, сайти на фреймворках навіть безпечніше).

Недоліки фреймворків:

- терміни розробки типового функціоналу на фреймворках більші, ніж при використанні CMS. Фреймворки містять лише базові компоненти бізнес-логіки рівня програми, тому багато функцій реалізуються індивідуально;
- для розробки на фреймворку потрібне розуміння бізнес-процесів, які потрібно реалізувати. Наприклад, якщо в CMS вже є якийсь процес обробки замовлень, то фреймворки такого не надають.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМУ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Інтерфейс програми

Титульна сторінка (головна) будь-якого сайту повинна максимально інформативно і в стислому об'ємі відображати необхідну користувачеві інформацію про сайт. На головній сторінці необхідно помістити логотип веб-сайту, основне меню сайту (для навігації по його структурі).

Веб-сайт призначена не тільки для досвідчених користувачів, але й для тих людей, які недавно почали працювати на персональних комп'ютерах.

Розглянемо детальніше і продемонструємо наочно інтерфейс веб-сайту.

Для перегляду веб-сайту потрібно відкрити його за відповідним посиланням, на момент розробки веб-сайту він знаходився на локальному сервері. Для відкриття веб-сайту на локальному сервері потрібно в браузері перейти за посиланням localhost:3000, після чого з’являється головна сторінка веб-сайту даного проекту, на якому знаходиться посилання на інформацію про підприємство, а саме: на сторінки реклами підприємства, сторінку продукції, сторінку інформації про підприємство та сторінку контактів для звернення до підприємства (див. рис. 2.1).

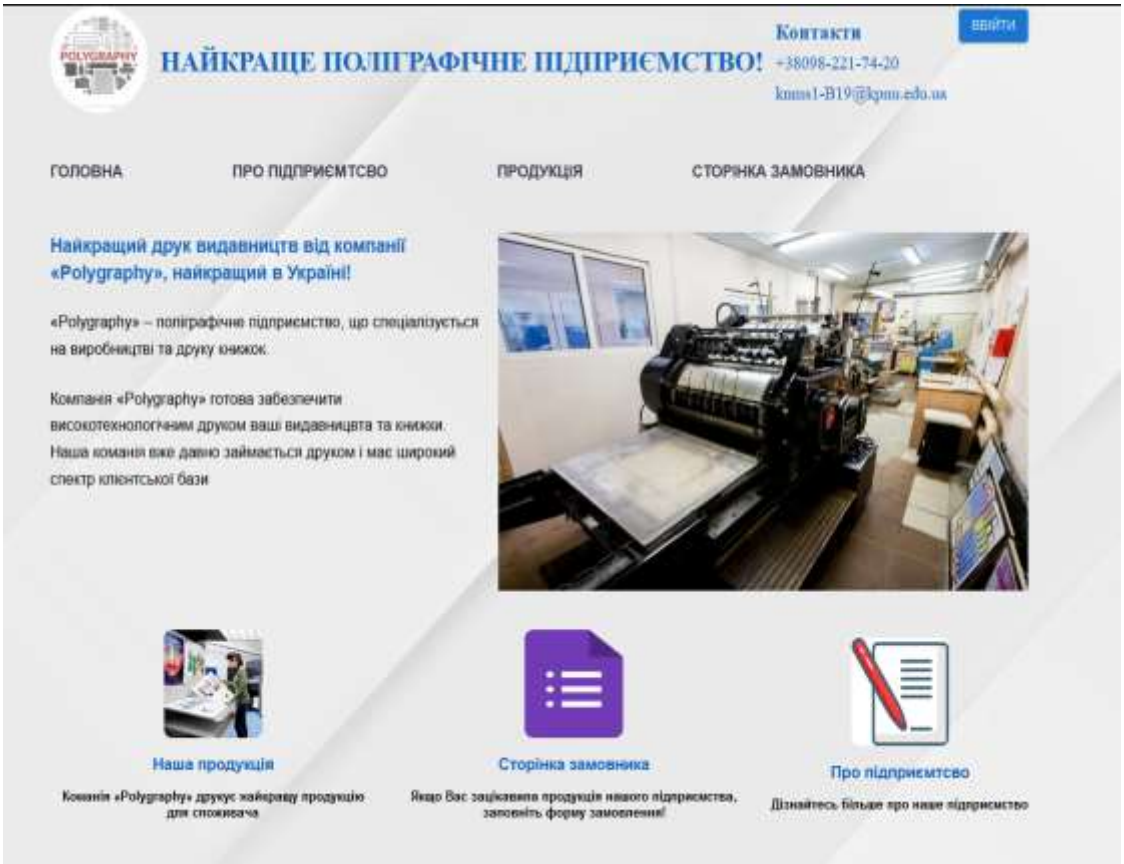

Рисунок 2.1 – Головна сторінка сайту

Для перегляду продукції, яку випускає підприємство, потрібно перейти на відповідну сторінку «Продукція» (див. рис. 2.2).

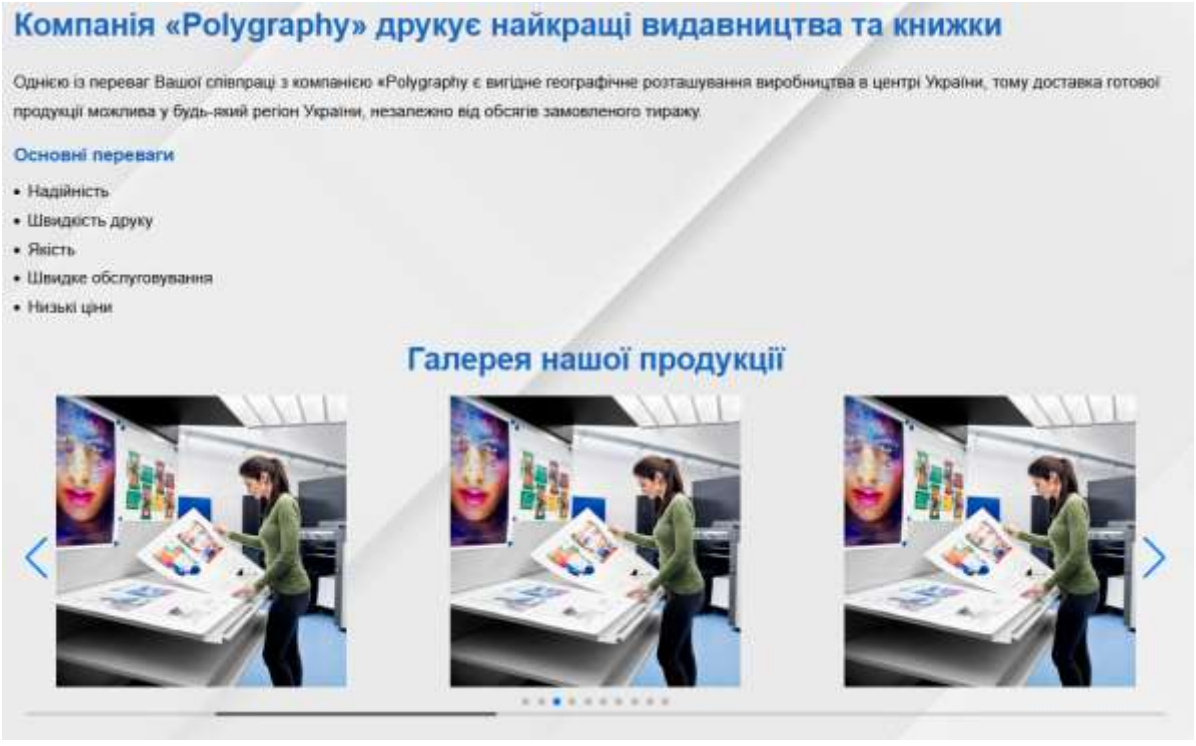

Рисунок 2.2 – Сторінка з продукцією підприємства

Для перегляду інформації про підприємство потрібно перейти на відповідну сторінку «Про підприємства» (див. рис. 2.3).

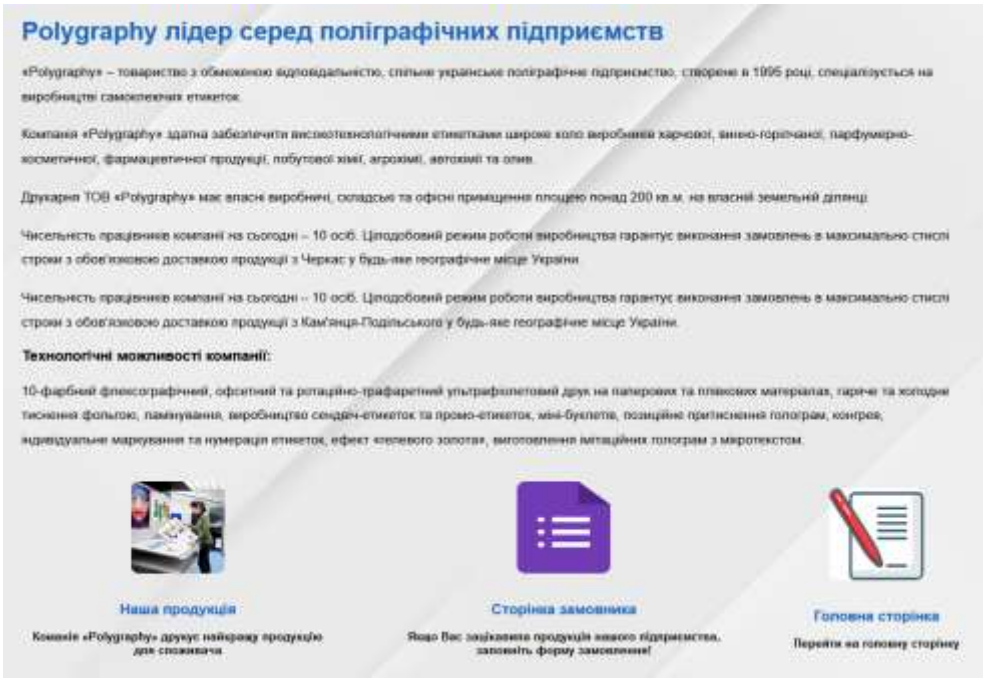

Рисунок 2.3 – Сторінка з інформацією про підприємство

Якщо користувачу сподобалось підприємство та його продукція, він може перейти на сторінку замовника та відправити замовлення. Для перегляду сторінки замовника потрібно перейти на відповідну сторінку «Сторінка замовника» (див. рис 2.4).

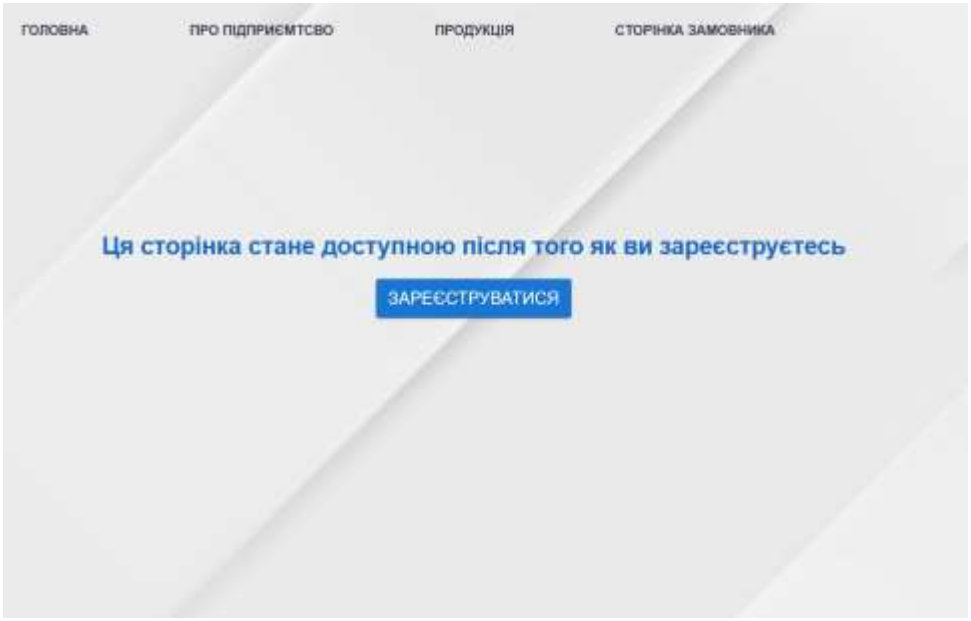

Рисунок 2.4 – Сторінка замовника

Але щоб замовити щось на сайті потрібно для цього зареєструватися це можна зробити двома способами:

- натиснути в верхній частині сайту на кнопку «Увійти»;
- перейти на сторінку замовника та натиснути на кнопку «Зареєструватися».

Після того як користувач натиснув на кнопку йому відкривається форма для реєстрування (див. рис. 2.5)

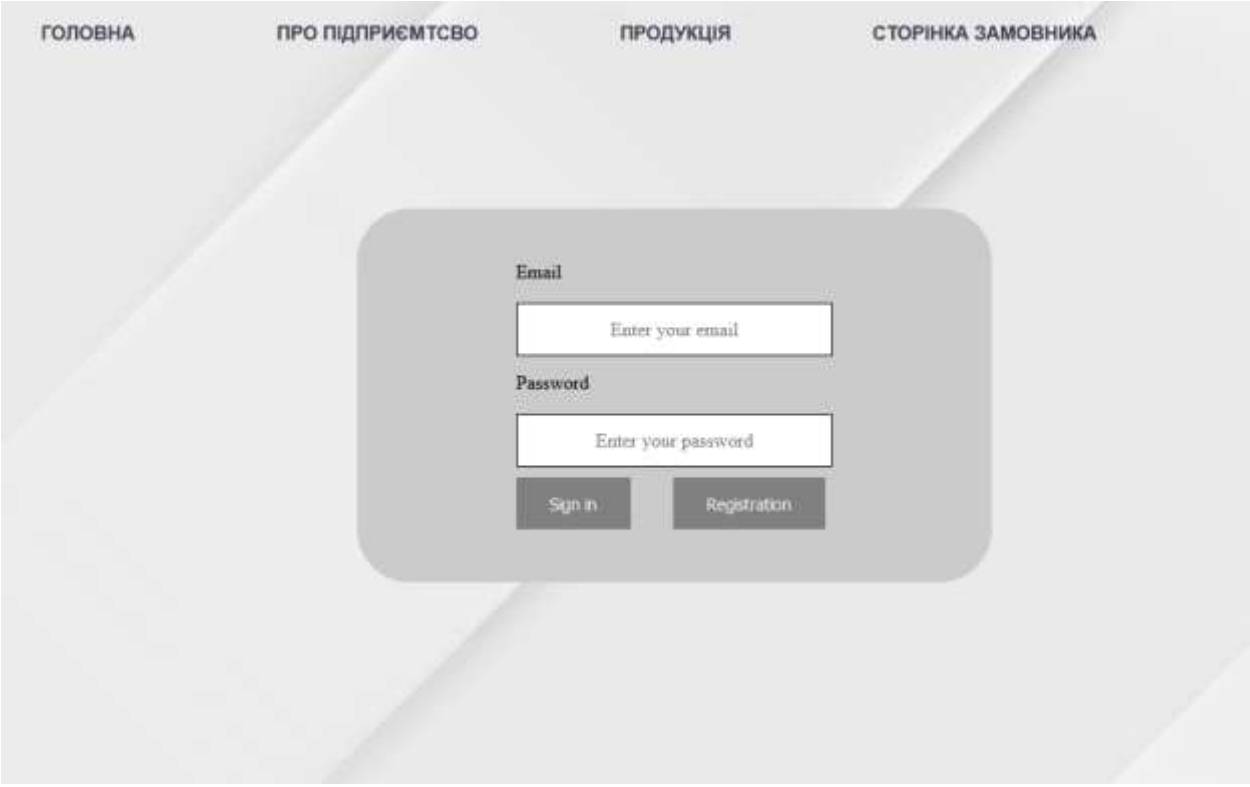

Рисунок 2.5 – Форма для реєстрування

Після того як користувач зареєструвався та увійшов в систему, йому стануть доступні усі можливості сайту. Отже коли користувач увійшов в систему його перекидає на сторінку замовника (див. рис. 2.6)

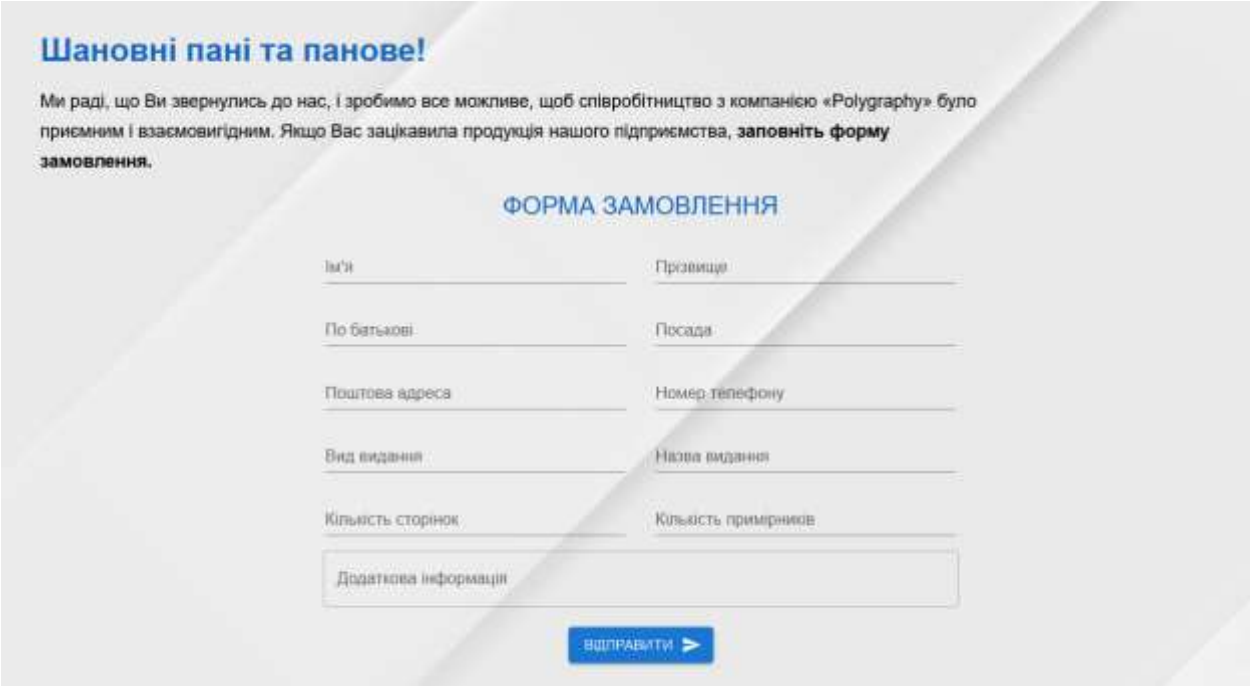

Рисунок 2.6 – Сторінка замовника

На сторінці замовника користувач може заповнити форму замовлення, і відправити дані про себе, та дані про те що йому потрібно надрукувати.

У складовій частині сайту, є клієнтська частина, і є серверна частина, після того як користувач ввів дані, чи то є дані при реєстрації чи то є дані про замовлення, вони відправляються на сервер, тобто на back-end, там вони оброблюються і відправляються у базу даних. Для розробки проекту, я вибрав саме базу даних MongoDB [8].

Для того щоб взаємодіяти з базою даних, її потрібно спочатку створити (див. рис 2.7)

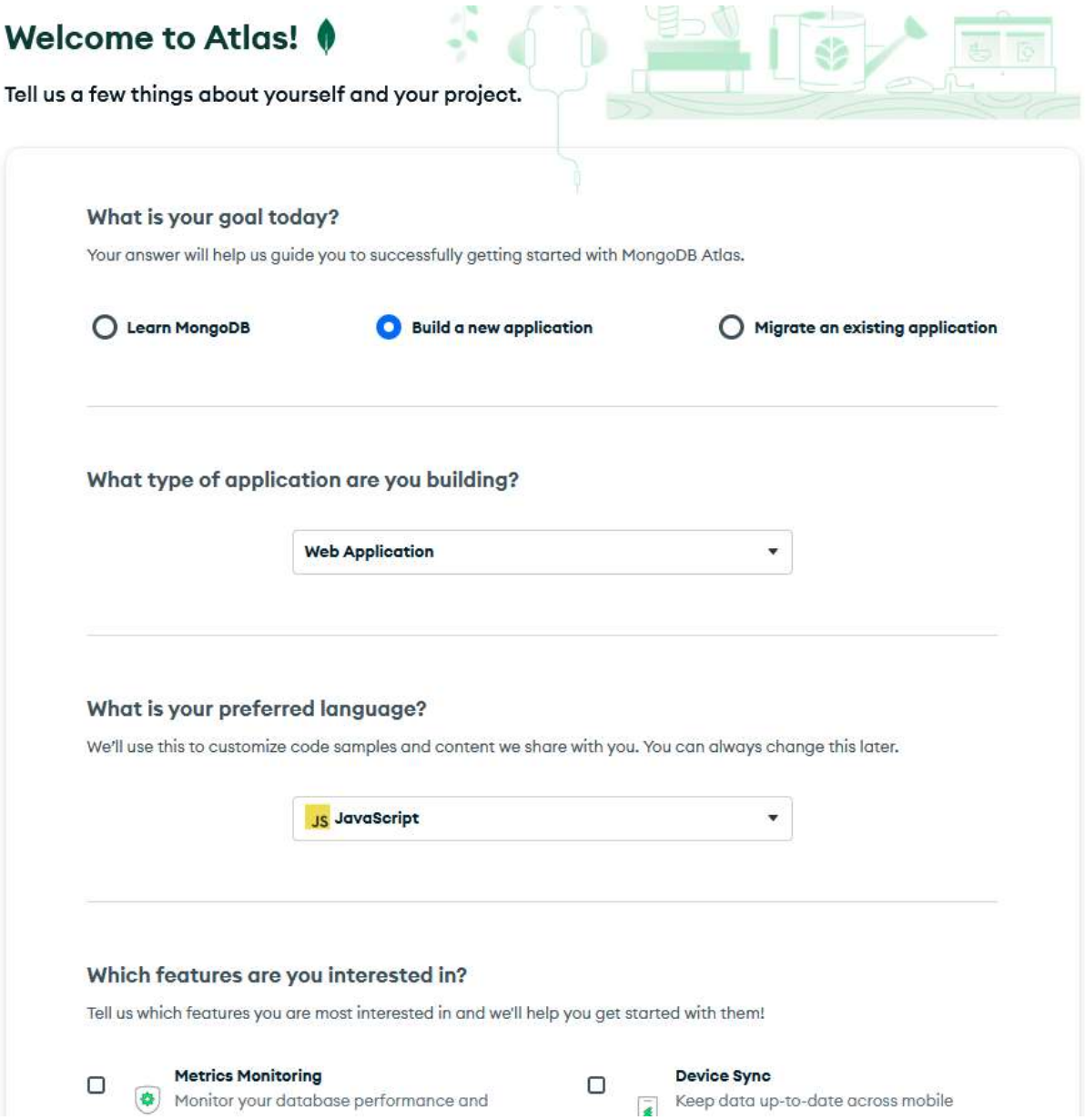

Рисунок 2.7 – Створення бази даних

Після того як ми увійшли в нашу базу даних потрібно вказати саме для чого ви будете використовувати її, вказуєте що маєте розробити новий додаток (**Build a new application**), вказуєте якого саме типу має бути цей додаток (**Web Application**), і вибираєте якою мовою програмування ви будете користуватися, у моєму випадку це мова **JavaScript**, після чого нажимаєте завершити.

Після того як ми натиснули завершити нас перекидає на сторінку створення бази даних (див. рис. 2.8)

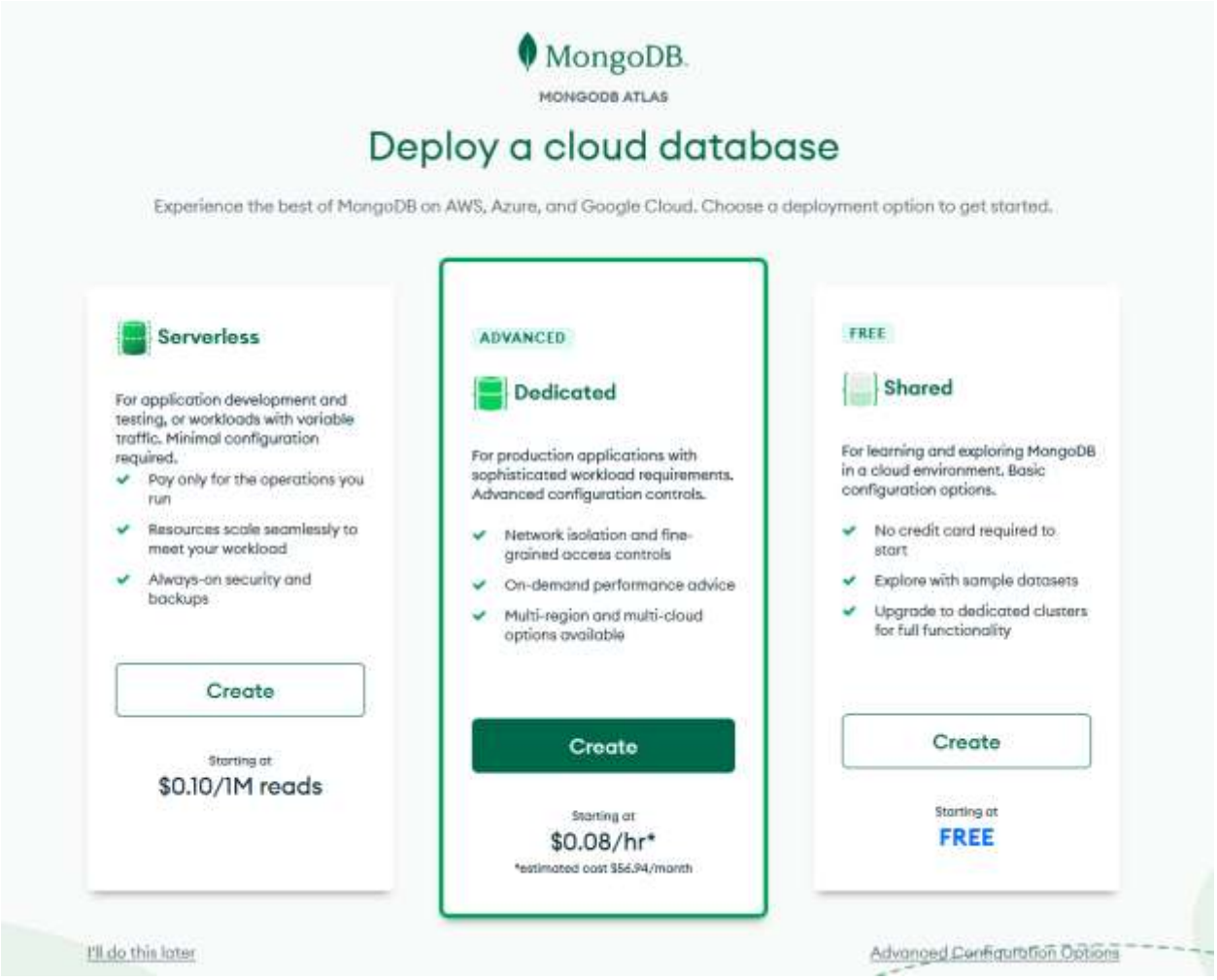

Рисунок 2.8 – Створення хмарної бази даних

Вибираємо безкоштовну версію, і натискаємо (**Create**), після чого нас перекидає на сторінку де потрібно вибрати який саме хмарний сервіс, та регіон вам підходить (див. рис. 2.9)

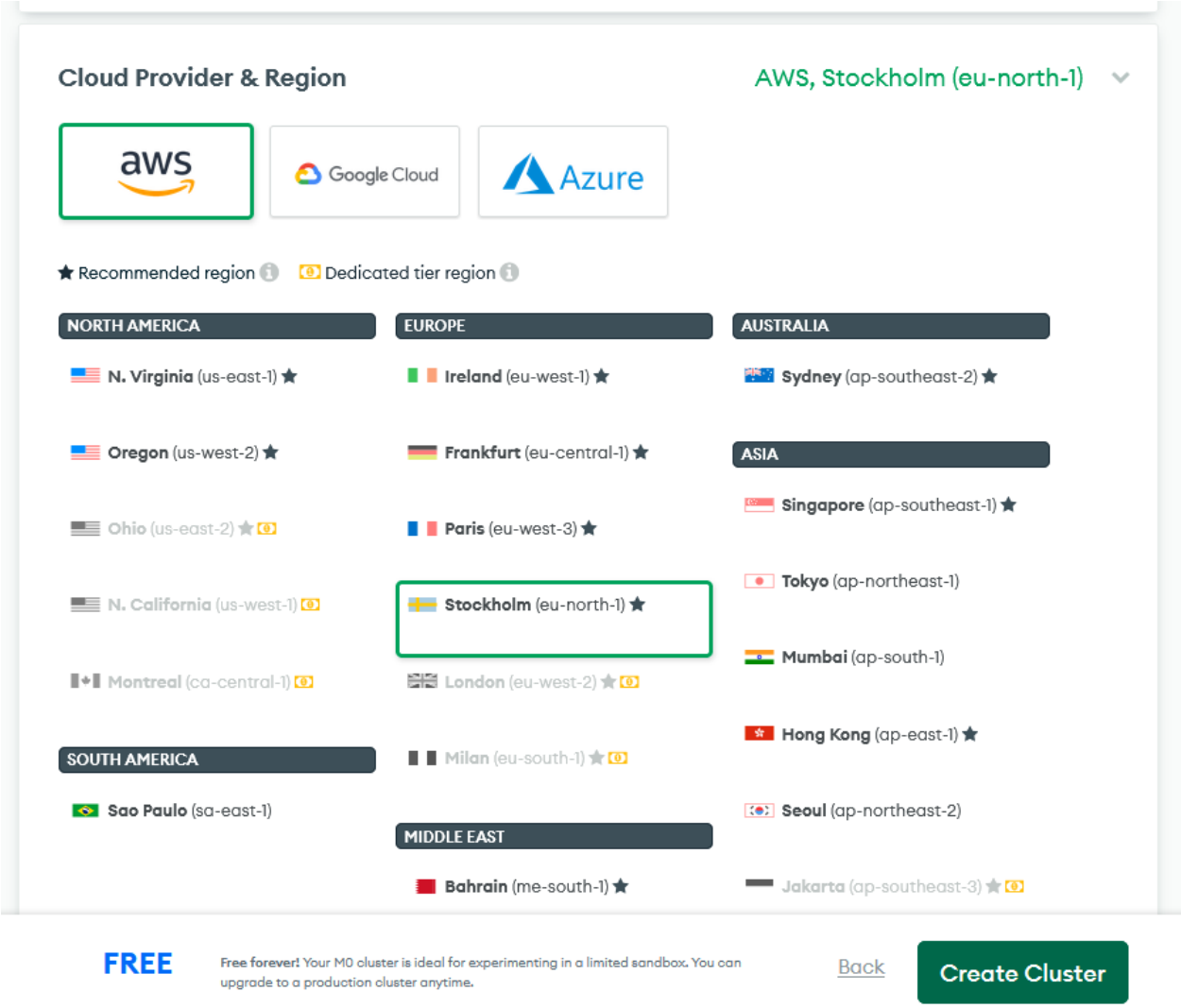

Рисунок 2.9 – Вибір хмарного сервісу та регіону

У моєму випадку я вибрав **AWS** та вибрав регіон **Stockholm** і натиснув на кнопку створити кластер (**Create Cluster**). Після чого нас перекидає на сторінку де вже потрібно налаштувати нашу базу даних для подальшого використання, для початку нам потрібно вибрати яким саме способом ми будемо приєднуватися до нашої бази даних, у моєму випадку я вибрав з-за допомогою логіна та пароля (див. рис. 2.10).

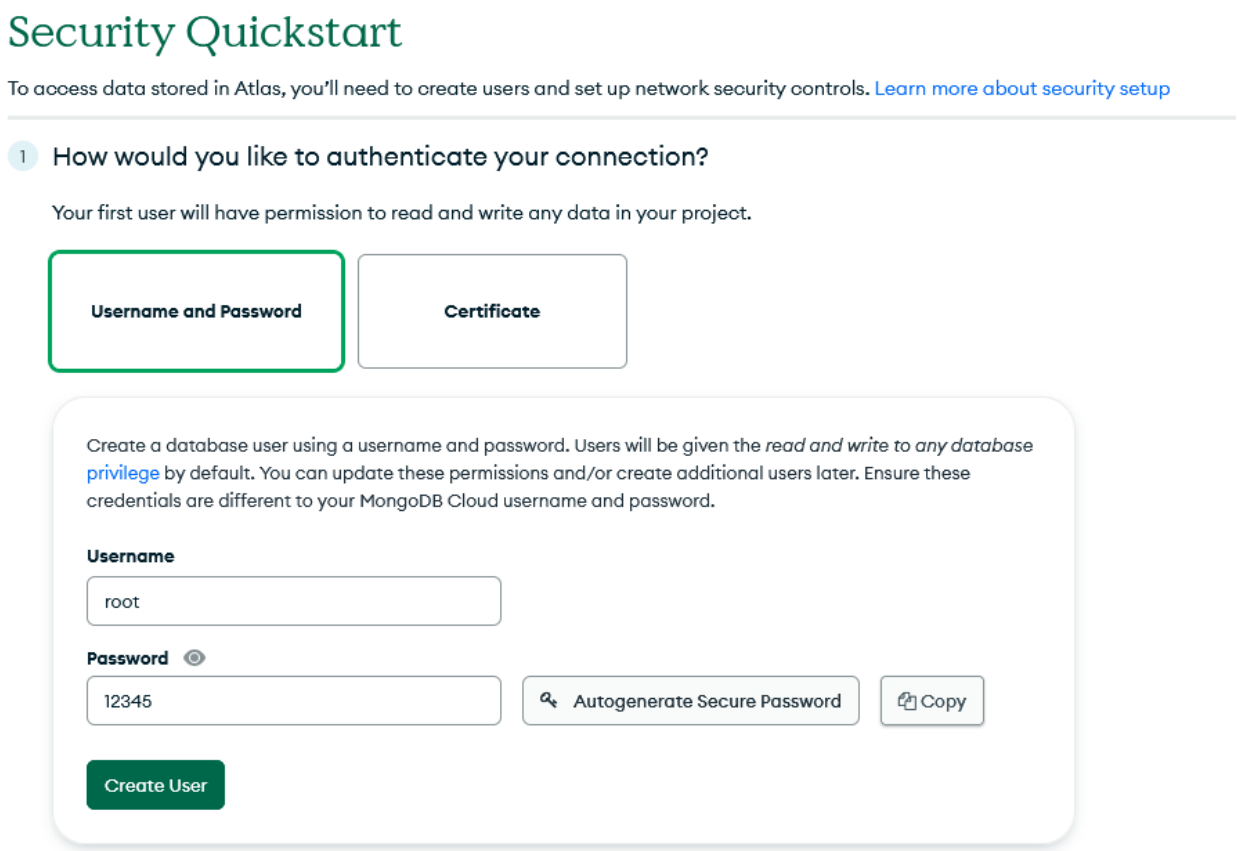

Рисунок 2.10 – Вибір способу підключення до бази даних

Після чого в нас запитують звідки саме ми хочемо законектитись, або з локальної мережі, або з хмарної мережі, у моєму випадку я підключусь з локальної мережі, та виберу свій IP address (**Add My Current IP Address**) (див. рис. 2.11).

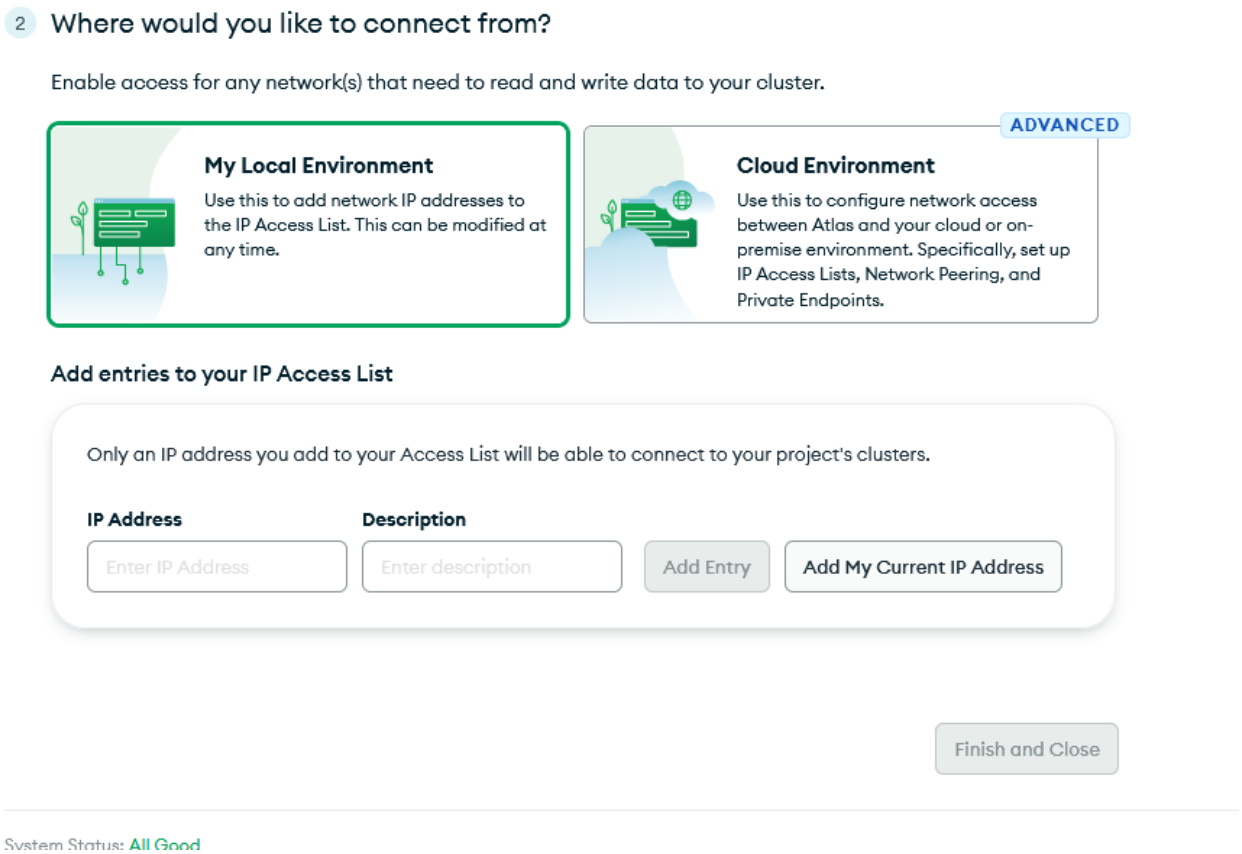

Рисунок 2.11 – Підключення до бази даних

Далі ми натискаємо на кнопку (**Finish and Close)** і в нас створюється база даних з якою ми можемо взаємодіяти.

Тепер нам потрібно приєднати наш back-end до бази даних, для цього заходимо в нашу базу даних, натискаємо на кнопку (**Connect**) (див. рис. 2.12)

Рисунок 2.12 – Приєднання back-end до бази даних

Тепер ми вибираємо яким методом ми будемо підключати у моєму випадку це метод підключення до мого додатка (**Connect your application**) (див. рис. 2.13)

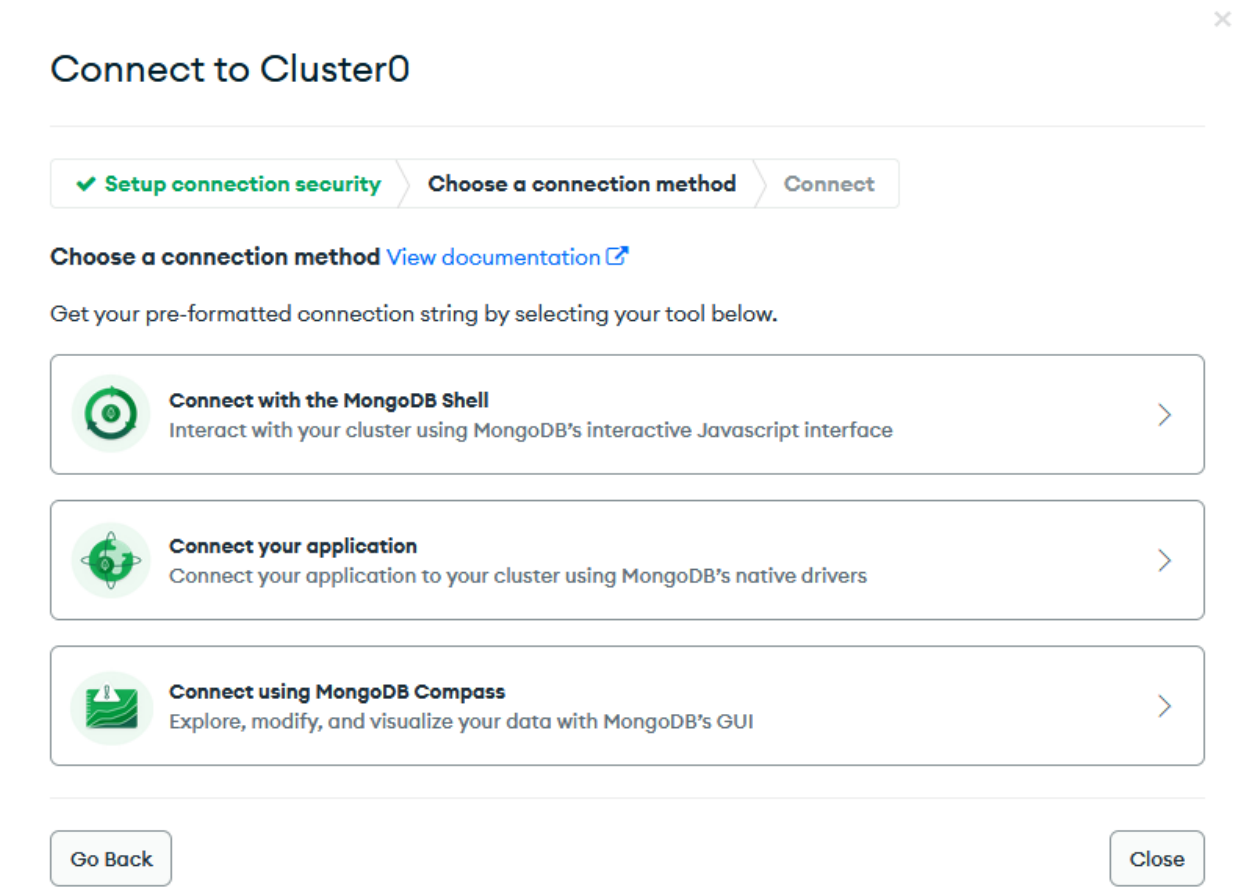

Рисунок 2.12 – Методи підключення бази даних

Після чого нас перекидає на сторінку де показано як потрібно приєднатися з бази даних (див. рис. 2.13)

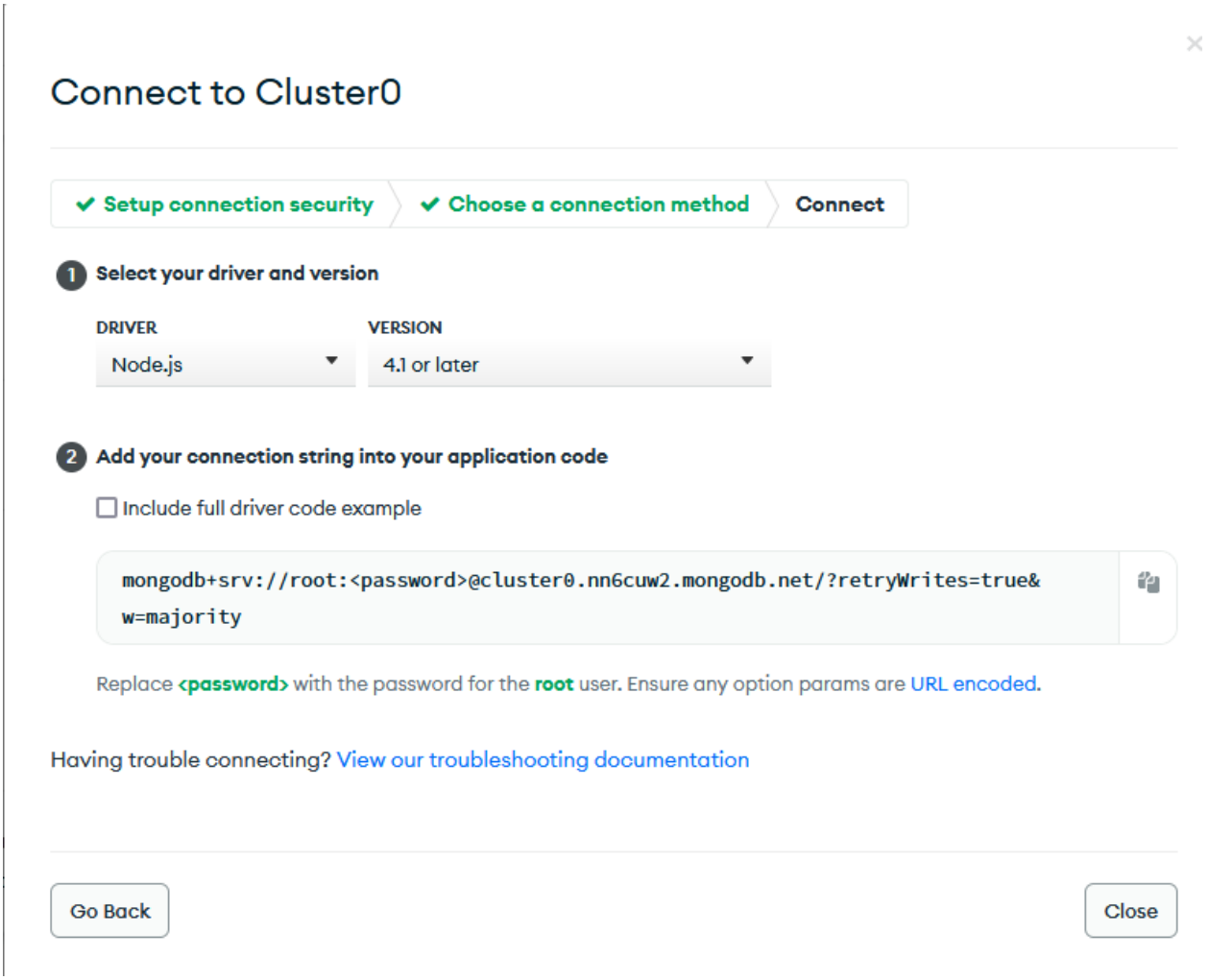

Рисунок 2.13 – Приєднання бази даних

Тепер ми можемо підключити нашу базу даних до нашого back-end і взаємодіяти з ним за допомогою цієї строки підключення.

2.2 Алгоритм функціонування сайту

На рис. 2.7 представлена структурна схема організації сайту, яка враховує всю специфіку веб-сайту, що розробляється.

Рисунок 2.7 – Організація навігації сайту

На головній сторінці відображається основна інформація про підприємство.

Сторінка «Продукція» містить слайдер з типами товарів, які випускає підприємство та основні переваги використання саме цього підприємства.

Сторінка «Про підприємство» містить інформацію про підприємство та основні напрямки його діяльності і технологічні можливості.

Сторінка «Сторінка замовника» містить форму для замовлення товарів які можуть містити такі поля:

- Ім’я;
- Прізвище;
- По батькові;
- Посада;
- Поштова адреса;
- Номер телефону;

- Вид видання;
- Назва видання;
- Кількість сторінок;
- Кількість примірників;
- Додаткова інформація.

РОЗДІЛ 3. ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Розробка програмної реалізації сайту

При створенні сайту використовувався фреймворк React, який є найкращий серед фреймворків(на думку автора). Та для візуалізації сайту і надання йому стилів використовувався CSS Modules.

Розглянемо створення головної сторінки сайту (рис. 3.1). Вона є основоположною для всіх інших.

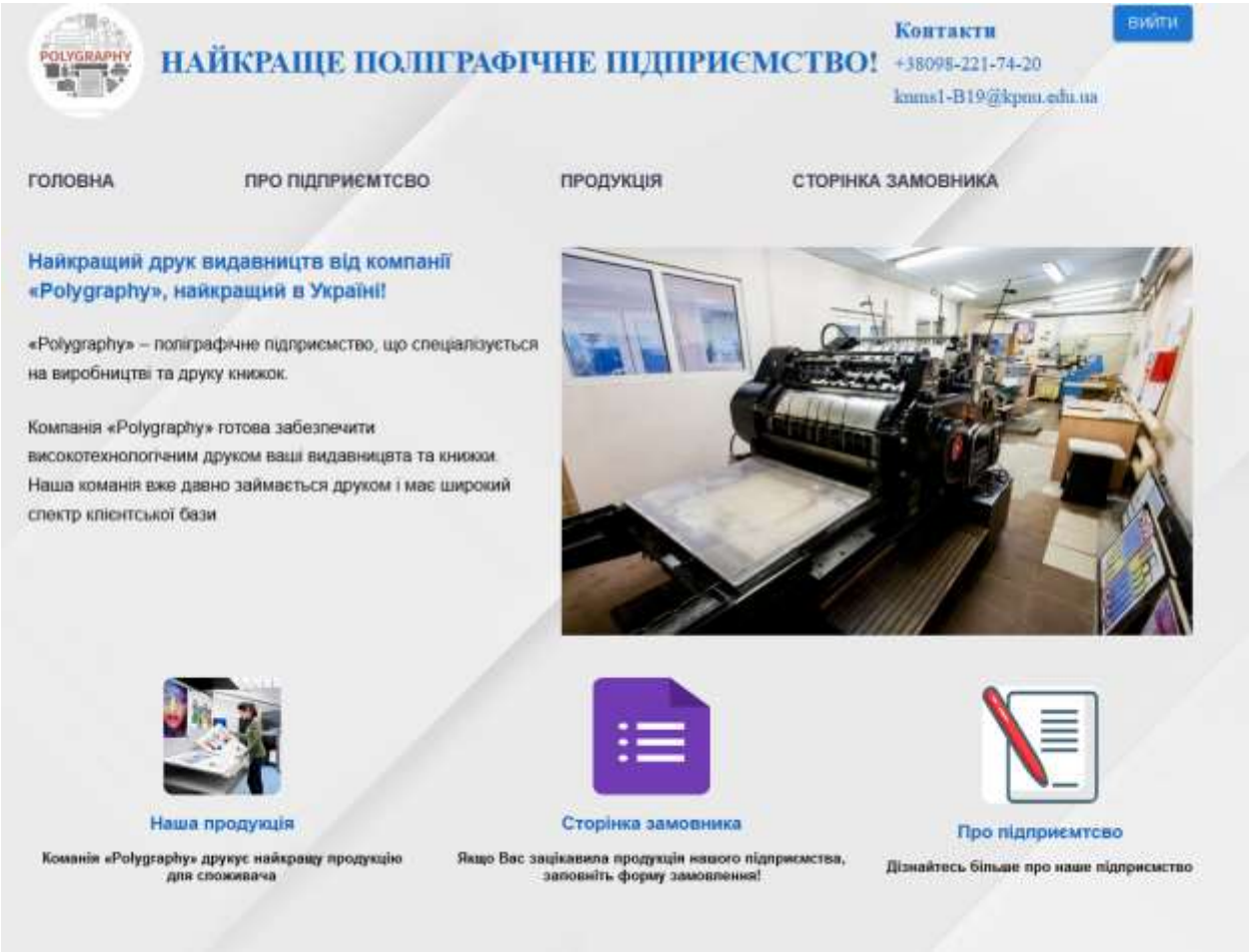

Рисунок 3.1 – Розробка головної сторінки сайту

Як видно з (рис. 3.1), в основі сторінки лежить головний контейнер, куди були поміщені ще 3 контейнери а саме:

- Контейнер з інформацією
- Контейнер з зображенням
- Контейнер з навігацією по сайту

Будь-яка інформація (текстова або графічна) заноситься безпосередньо в елементі контейнера. Таким чином, досягається структурованість сторінки, що дозволяє найзручніше редагувати будь-який її елемент. Так само для зручності правки і логічного відділення одного сегменту від іншого кожному з них привласнено назву класу (**div className={s.wrapp}**)

Відповідно до розробленої структури був спроектований сайт, він містить частину Header(верхня сторінка сайту) її користувач бачить завжди, та посилання за допомогою **BrowserRouter** які при натисканні на гіперпосилання(**Link/NavLink**) завантажую частину сайту і відрисовує її, але сторінка не оновлюється а все залишається на місці, завдяки **Роутінгу.** Це працює таким чином є **Route** який слідкує за url сторінка, і якщо url підходить то **Route** завантажує елемент сторінки.

Як і планувалося на етапі постановки завдання, сайт містить всі необхідні структурні і навігаційні елементи: контактну інформацію у верхній частині сайту, форму замовлення, реєстрацію на сайт, та слайдер з продукцією підприємства. При натисненні на посиланні «Продукція» виводиться інформація про матеріали, які випускаються підприємством. Це набагато спрощує пошук матеріалів для клієнтів (рис. 3.2).

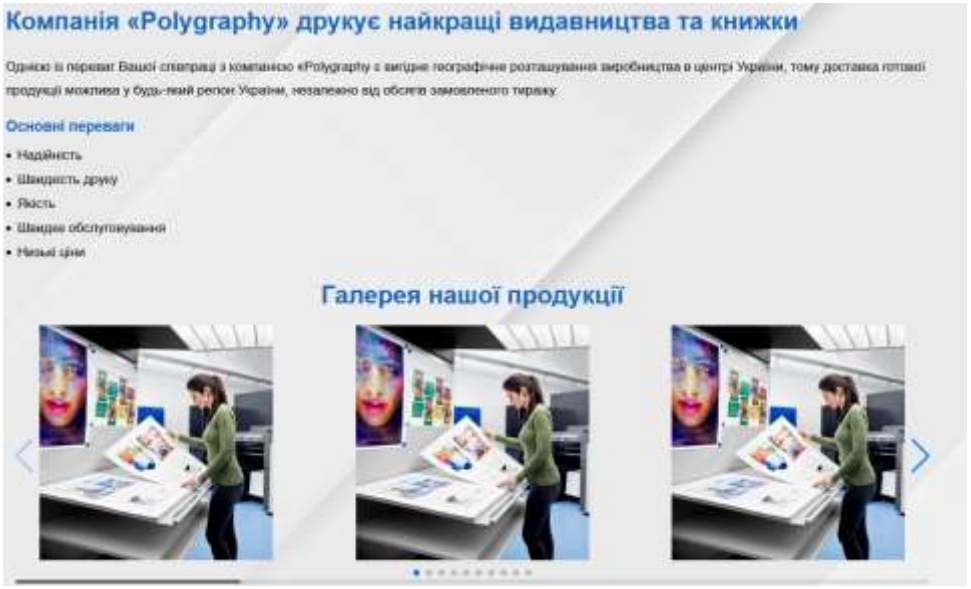

Рисунок 3.2 – Вікно з продукцією підприємства

На цій сторінці написано основні переваги використання саме цього підприємства, та на цій сторінці також реалізована галерея з свайпером, тобто

користувач може прокручувати галерею з товарами за допомогою стрілочок, повзунка або за допомогою натискання на зображення та перетягування в сторони.

Також користувач має можливість вказати свої дані та замовити друк того видавництва який йому потрібно (Рис. 3.3). Після того як користувач заповнить всю анкету, та натиснить кнопку відправити, уся інформація яку вказав користувач, відправиться у базу даних MongoDB, і ця інформація буде там зберігатися.

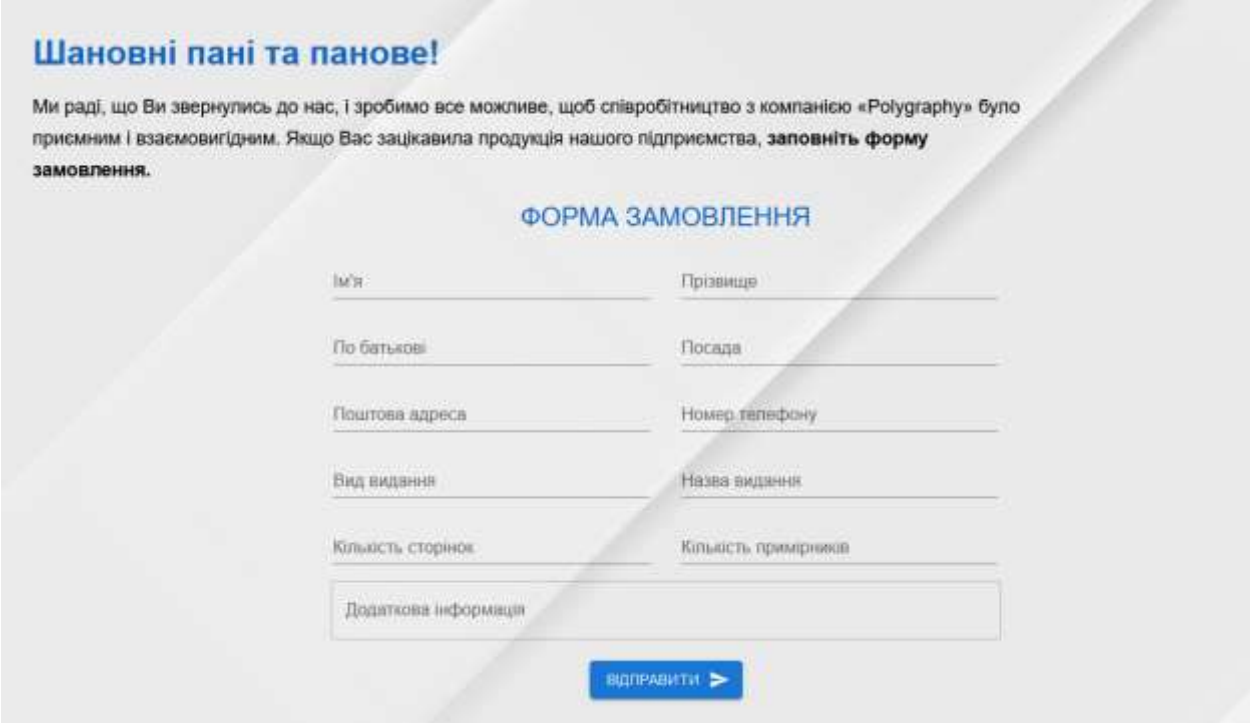

Рисунок 3.3 – Сторінка замовника

3.2. Тестування програмної частини

У процесі створення сайту, на етапі розробки, дуже важливо перевірити працездатність сайту. Для перевірки працездатності сайту проводиться спеціальне тестування.

Під час тестування сайт перевіряється на відповідність технічному завданню, перевіряються його технічні характеристики. Залежно від вихідного технічного завдання в процесі тестування можуть здійснюватися такі перевірки: перегляд сайту на різних моніторах. Перегляд сайту на моніторах різних розмірів і при різному дозволі дозволяє оцінити як буде виглядати

дизайн сайту на комп'ютерах потенційних відвідувачів: чи з'являється горизонтальна смуга прокручування на маленьких екранах, чи не зміщаються елементи при різному дозволі й т.д. перегляд сайту в різних бразузерах. Перегляд сайту в різних інтернет-браузерах і їхніх версіях, дозволяє перевірити кросбраузерність сайту; перевірка часу завантаження сайту на різній швидкості інтернет-підключення. Як швидко завантажуються сторінки сайту на мінімальній і на максимальній швидкості підключення; перевірка працездатності гіперпосилань на сайті. Чи є на сайті «биті» посилання або посилання, що ведуть не на ті сторінки.

Відображення кольорів на сайті при різних настроюваннях колірної палітри монітора. Дозволяє довідатися, як будуть відображені кольори сайту на моніторах потенційних відвідувачів; перевірка відображення шрифтів, анімації й графічних зображень. Дозволяє знайти й усунути можливі проблеми із правильним відображенням зображень, анімаційних роликів і шрифтів на сайті; перевірка властивостей кожної сторінки сайту: заголовків, ключових слів, описи або інших мета-тегів; перевірка змісту кожної сторінки на відповідність вихідним матеріалам замовника й перевірка правопису на кожній сторінці.

Для цього відкриваємо один з популярний браузерів (напр. Google Chrome) і в строку адреси вводимо. Зачекавши кілька секунд можна побачити головну сторінку сайту, отже швидкість завантаження в нормі. Тепер потрібно протестувати його по різним критеріям описаним вище. Перевіримо сайт на кросбраузерність. Відкриваємо сайт у браузерах Opera, Mozilla, Chrome і аналогічно відкриваємо сайт. Отже можемо зробити висновок, що сайт працює корректно.

Налагодження програми. Налагодження програм – це спеціальний етап у розробці програми, що полягає у виявлені та усунення програмних помилок, факт існування яких уже встановлено. Але щоб усунути помилку – її потрібно спочатку знайти. Під час розробки сайту не виникло жодних помилок, або багів.

РОЗДІЛ 4. ЕКОНОМІЧНА СКЛАДОВА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Розрахунок вартості продукту

У дипломному проекті виконана розробка інформаційної технології для поліграфічного підприємства.

Окупність Інтернет проекту (Сайту), не дивлячись на всі його очевидні переваги, є актуальним питанням, як і для будь-якого іншого бізнес проекту.

Переваги які дає сайт:

- зниження накладних витрат;
- можливість знаходження нових партнерів і замовників;
- розширення території ведення бізнесу;
- рекламу товарів і послуг;
- вирішення кадрових питань;
- підняття престижу і довіри до фірми;
- міжнародну популярність.

Сайт окупить себе в той момент, коли прибуток, одержаний від усіх угод, здійснених за допомогою сайту, зрівняється з сумою витрат на його створення.

Тобто в спрощеному вигляді має бути забезпечено умову:

Окупність = Витрати,

де: Витрати - загальна сума всіх витрат на створення сайту і підтримку сайту, грн;

Окупність - загальний прибуток від усіх угод, що відбулися за допомогою сайту, грн.

Окупність = Прибуток * Угоди,

де: Прибуток - середній прибуток від однієї угоди, грн;

Угоди - число угод, які були складені.

Отже, можна ускладнити розрахунок окупності сайту, враховуючи цілий ряд моментів. Наприклад, можна враховувати, що між моментом вкладення грошей в розробку сайту і моментом окупності сайту проходить кілька місяців, тобто брати до уваги, що «гроші завтра не є гроші сьогодні».

Якщо ж фірмою узятий кредит на створення сайту, то необхідно врахувати і банківський відсоток, що виплачується за використання кредиту.

Середній прибуток від угоди кожна фірма може визначити самостійно. Кількість угод залежить від числа відвідувачів сайту. За статистикою 4-5% загального числа відвідувачів сайту купують товари або замовляють послугу, пропоновані на сайті. Число відвідувача в день, за умови проведення заходів щодо просування сайту можна оцінити по таблиці:

Таблиця 4.1

Таблиця відвідувачів

Число базових HTML сторінок	10	50	100	150	200
Число відвідувачів в робочий день, не менше	10-15	50-70	100-120	140-170	180-220
Число відвідувачів у вихідні та святкові дні, не менше	3-5	20-25	40-60	70-90	90-110
Число відвідувачів в рік, не менше тисячі	3-3,5	14-16	28-32	40-50	50-65

Математично точно розрахувати рентабельність сайту складно з багатьох причин. На рентабельність сайту впливають найрізноманітніші чинники, багато з яких швидко змінюються з плином часу. Серед них:

1. Географічне розташування фірми.

• найбільш високий коефіцієнт рентабельності роботи сайту характерний для Києва, Дніпра та інших великих міст. Чим менше населений пункт, тим, як правило, менше для сайту і коефіцієнт рентабельності. Мова

при цьому йде про порівняння сайтів однакових за обсягом, датою створення, професійно-технічному рівню виконання, однієї тематики тощо.

2. Конкурентоспроможність товарів і послуг в даний момент часу.

• необхідно враховувати сезонність продажів, ступінь насиченості ринку конкретним товаром, рівень попиту на конкретний товар в конкретній місцевості в певний період часу, співвідношення показника "ціна / якість" у порівнянні з аналогічними товарами і послугами, представленими на сайтах-конкурентах.

3. Число користувачів Інтернет в даному населеному пункті та регіоні і багато інших параметрів.

Таблиця 4.2

Перелік етапів та робіт по розробці проекту

Найменування		Вид роботи		Виконавець, посада
стадії	етапу	шифр	зміст роботи	
1	2	3	4	5
1 Підготовча стадія	1.1 Вивчення стану питання	1.1.1	Дослідження проблеми	
		1.1.2	Вивчення та аналіз аналогічних розробок	
		1.1.3	Економічне обгрунтування доцільності виконання проекту	
	1.2 Розробка технічного завдання (ТЗ)	1.2.1	Складання ТЗ	

2 Технічна пропозиція	2.1 Аналіз ТЗ та техніко-економічне обгрунтування проекту	2.1.1	Доведення техніко-економічного обгрунтування	
3 Теоретична розробка	3.1 Теоретичне вивчення задачі	3.1.1	Визначення переліку технологій, які використовуватимуться при розробці та мови програмування	
		3.1.2	Розробка алгоритмів роботи програми на високому рівні	
		3.1.3	Розробка структури програмного забезпечення та схеми взаємодії її компонент	
Практична реалізація	4.1 Розробка дизайну	4.1.1	Розробка шаблону дизайна	
		4.1.2	Блочна верстка	
	4.2 Розробка структури сайту	4.2.1	Розробка ядра сайту	
		4.2.2	Система управління сайтом	

Зведені результати тривалості та трудомісткості розробки та реалізації проекту.

Таблиця 4.3

Таблиця витрат на заробітну плату

Шифр роботи	Найменування роботи	Виконавець, посада, спеціальність	Кількість виконавців	Тривалість виконання роботи, дні	Денна тарифна ставка, грн	Оплата праці, грн
1	2	3	4	5	6	7
1.1.3 - 1.2.1	Економічне обґрунтування доцільності виконання проекту, складання ТЗ		1	2	100	60
3.1.1 - 3.1.3	Теоретична розробка		1	6	100	250
4.1.1 - 4.2.2	Практична реалізація		1	3	100	2060
5.1.1 - 5.2.3	Доробка всього комплексу програмного забезпечення		1	10	100	20
6.1.1 - 6.1.3	Заключна стадія		1	1	100	20
	Всього заробітна плата					2500

Розрахунок інвестиційних витрат

Витрати на науково-дослідницьку роботу по розробці програмних засобів та апаратури, щодо даного дипломного проекту, включають наступні елементи:

- витрати на основну заробітну плату виконавців. Це сума часткових заробітних плат за кожну роботу і вона приведена в таблиці 5.5.

$$ЗП_{основна}= 2500 \text{ грн.}$$

- додаткова заробітна плата. Її можна обчислити за формулою:

$$ЗП_{додаткова}=0,12*ЗП_{основна}$$

$$ЗП_{додаткова}= 300 \text{ грн.}$$

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$$ФЗП=ЗП_{основна}+ЗП_{додаткова}$$

$$ФЗП=2800 \text{ грн.}$$

- обов'язкові відрахування на заробітну плату (єдиний соціальний внесок – 22%). Таким чином обов'язкові відрахування складають:

$$Відр = ФЗП * 0,22= 616 \text{ грн.}$$

- накладні витрати розраховуємо за формулою:

$$НВ=0,4*ЗП_{основна}.$$

$$НВ = 1000 \text{ грн.}$$

Для створення сайту необхідний обсяг фінансування в розмірі 25000 грн. З них:

1. Шаблони дизайн = 3500 грн.
2. Верстка сайту : Блокова нарізка HTML - шаблону =8500 грн .
3. Система управління сайтом = 4000 грн .
4. Розробка ядра сайту = 9000 грн.

Сайт підтримує одну мову- українську.

4.2. Розрахунок економічної ефективності від впровадження

Економічна ефективність - результат економічної діяльності, економічних програм і заходів, що характеризується відношенням отриманого економічного ефекту, до витрат факторів, ресурсів, який зумовив отримання цього результату, досягнення найбільшого обсягу виробництва із застосуванням ресурсів певної вартості Система показників ефективності виробництва повинна давати всебічну оцінку використання всіх ресурсів підприємства і містити всі загальноекономічні показники. Дуже важливо, щоб розрахунки ефективності виробництва велися безупинно: на стадіях проекту плану, затвердження плану, по мірі його виконання.

Розрахунок показників економічної ефективності. Так як за основу беруться безкоштовні версії програмних продуктів HTML, CSS та JavaScript, в витратну частину створення веб-сайту належать такі витрати як: витрати по електроенергії, витрати по розміщенню в мережі інтернет (хостинг) та інші всілякі витрати на канцелярські товари та витратні матеріали для комп'ютера.

Таблиця 4.4

Розрахунок щомісячних витрат електроенергії компанії

Назва	**кількість**	**кВт/год**	**кВт за добу (приблизно)**	**кВт в міс.**
Комп’ютер	1	0,17	1,53	45,9
Освітлення	3	0,36	9,72	392,85
Тому:		0,53	11,25	438,75

Розрахунок електроенергії для восьмигодинного робочого дня.

Для підприємства 1 кВт / год= 1,42

В місяць 1,42*438,75 = 623,02 грн.

Таблиця 4.5

Розрахунок щомісячних витрат на утримання Веб-сайту.

Назва	**Сума**
Електроенергія	623,02
Хостинг	30
Інтернет	100
Інші витрати	50
Тому:	803,02

$R_{Пост}$ = 803,02 грн - постійні витрати.

Річна сума амортизаційних відрахувань розраховується за формулою:

$$A = \frac{Ф * N_A}{100\%},$$

де Ф - первісна вартість основних фондів за видами;

NA - норма амортизації за видами основних фондів, у%.

Річну суму амортизаційних відрахувань відобразимо в таблиці 7

Таблиця 4.6

Розрахунок річної суми амортизаційних відрахувань

Елементи основних фондів	**Кількість**	**Вартість**	**Сума**	**Норма амортизації**	**Амортизаційні відчислення**
Комп'ютер	1	10000	10000	20%	2000
Приміщення	13,6 м^2	70	952	3%	28,56
РАЗОМ:					2028,56

Таким чином, річна сума амортизаційних відрахувань становить 2028,56 грн.

Виходячи з того, що трудомісткість створення системи становить 30 днів, розраховуємо амортизацію обладнання за цей період за формулою:

$$A_{\text{факт}} = \frac{\text{А}_{\text{год}} * \text{Т}_{\text{факт}}}{365},$$

Розрахуємо суму амортизаційних відрахувань для перерахованої групи обладнання з урахуванням числа календарних днів на розробку програмного забезпечення (веб-сайту) за формулою:

А = 2028,56 *30/365= 166,73 грн.

Отже, витрати на період розробки програмного продукту розрахуємо за формулою:

$$\text{З}_{\text{пр}} = \frac{\text{З}_{\text{м}} * \text{Т}_{\text{факт}}}{\text{Д}},$$

где $\text{З}_{\text{м}}$ – місячні витрати;

$\text{Т}_{\text{факт}}$ - число календарних днів на розробку інтернет - магазину;

Д - число днів в періоді (місяць).

$\text{З}_{\text{пр}}$ =803,02*30/22=1095,03 грн.

Розрахуємо собівартість програмного продукту за формулою:

Сст - собівартість розробки програми

Сст = Зпр + А + ЄСП

Сст = 517,5 + 166,73 + 113,85= 798,03 грн.

Дана собівартість є приблизною, оскільки в ній не враховані деякі деталі, які істотно не вплинуть на підсумок.

ССТ ≈ 798,03 грн.

Виходячи з нормального рівня рентабельності 20% ми можемо визначити ціну розробленої нами програми:

$$\text{Ц} = \text{С}_{\text{ст}} + \frac{\text{С}_{\text{ст}} * R}{100\%},$$

де Сст - собівартість розробки програми;

R - планований рівень рентабельності.

Ц = 798,03 + 798,03 * 20/100 = 957,63 грн.

Витрати на впровадження програмного продукту становлять 957,63 грн.

В економічній частині дипломного проекту було розраховано його повну собівартість, усі витрати на розробку продукту, зведений кошторис капітальних затрат на розробку та створено таблиці з обліками погодинних оплат за виконану роботу, таблиця з етапами роботи та витраченим на них часом.

Даний економічний план не вміщує в собі розрахунку коефіцієнту економічної ефективності чи періоду окупності, оскільки був розроблений як практична частина дипломного проекту і для впровадження на планується.

ВИСНОВОК

Веб-сайт це обличчя тієї фірми, тієї установи, людини, яка розмістила її в WWW. Саме тому сьогодні веб-дизайну приділяється така величезна увага, бо від нього на пряму залежить популярність того або іншого інформаційного ресурсу мережі. Недаремно зараз професія веб-дизайнера є одною з самих високооплачуваних.

Людина, що створює веб-сторінку, сполучає свої знання і навики зі своїм творчим потенціалом. Уміння творити, ось що виділяє справжнього веб-програміста. Для того, щоб створити веб-сторінку, яка б радувала око, потрібно поєднувати в собі якості художника і програміста.

Підводячи підсумок всьому вище сказаному, хочеться відзначити, що фреймворки стають все більш популярними і затребувані, вони вирішують низку проблем і тому їх все частіше починають використовувати у повсякденній розробці.

В ході виконання дипломного проектування був отриманий повнофункціональний веб-сайт, повністю готовий до застосування. Даний сайт орієнтований для широкого спектру застосування online інформації. З його допомогою користувачі зможуть отримувати необхідну інформацію про всю продукцію підприємства і замовляти товари та послуги. При розміщенні його в глобальній мережі географія розповсюдження зростає до масштабів всієї країни.

При розробці веб-сайту були проаналізовані сучасні веб-технології, що дозволяють створювати інтерактивні веб-сторінки. Розроблений сайт задовольняє всім вимогам, поставленим на етапі постановки завдання.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Ставровский А. Б., Карнаух Т. О. Перші кроки програмування. Київ: Диалектика. 2005. 389 с.

2. Кріс Міннік, Єва Холланд: JavaScript для чайників. 2019. 320 с.

3. Етан Браун : Вивчаємо JavaScript. Посібник із створення сучасних веб-сайтів. 2016. 368 с.

4. David Flanagan – JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition (June 9, 2020) 706 pages.

5. David Flanagan – JavaScript: Definitive Guide: Activate Your Web Pages (Definitive Guides) 6th Edition (May 13, 2011). 1096 pages.

6. Електронний ресурс [Режим доступу]: https://reactjs.org/

7. Електронний ресурс [Режим доступу]: https://javascript.info/

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

9. Jennifer Robbins HTML5 Pocket Reference – O'Reilly Media; Fifth edition (September 10, 2013) 182 pages.

10. Jon Duckett HTML and CSS: Design and Build Websites – John Wiley & Sons; 1st edition (November 8, 2011) 490 pages.

11. Основні фреймворки JavaScript та тенденції веб-розробки у 2021 році Електронний ресурс [Режим доступу]: https://itchef.ru/articles/85473/

12. CMSList. Огляд CMS. Сайт про системи управління сайтом. http://www.cmslist.ru

13. Luis Atencio – The Joy of JavaScript 1st Edition (March 2, 2021). 360 pages.

14. Електронний ресурс [Режим доступу]: https://www.w3schools.com/nodejs/

15. Електронний ресурс [Режим доступу]: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

16. Nicholas C. Zakas – Professional JavaScript for Web Developers 3rd Edition (January 18, 2012). 960 pages.

17. Adam D. Scott – JavaScript Cookbook: Programming the Web 3rd Edition (August 10, 2021). 538 pages.

18. Robin Nixon – Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites 3rd Edition (June 16, 2014). 786 pages.

19. Robin Nixon – Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites 6th Edition (August 17, 2021). 826 pages.

ДОДАТКИ

Додаток А

В кореневій папці проекту містяться дві папки: client та server, в яких розташовані усі налаштування та програмний код самого продукту (Рисунок А.1).

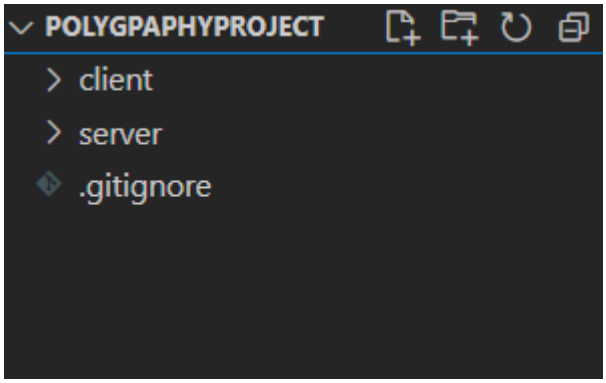

Рисунок А.1 – Структура папок проекту

Спочатку подивимось на серверну частину сайту, яка є основою. Основним файлом серверної частини є app.js де ми підключаємо усі наші плагіни та фреймворки.(Рисунок А.2)

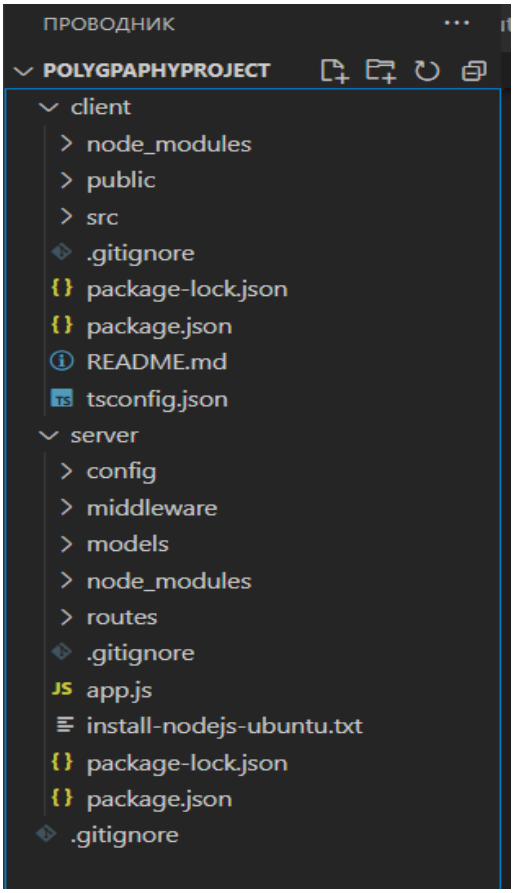

Рисунок А.2 – Серверна частина сайту

Файл app.js де відбувається усі підключення та налаштування.

```
const express = require("express");
const config = require("config");
const path = require("path");
const mongoose = require("mongoose");
const cors = require("cors");

const app = express();

app.use(express.json());
app.use(cors());

app.use(express.json({ extended: true }));

app.use("/api/auth", require("./routes/auth.routes"));
app.use("/api/link", require("./routes/link.routes"));
app.use("/t", require("./routes/redirect.routes"));
app.use("/api/orders", require("./routes/orders.routes"));

if (process.env.NODE_ENV === "production") {
  app.use("/", express.static(path.join(__dirname, "client", "build")));

  app.get("*", (req, res) => {
    res.sendFile(path.resolve(__dirname, "client", "build", "index.html"));
  });
}

const PORT = config.get("port") || 5000;

async function start() {
  try {
    await mongoose.connect(config.get("mongoUri"), {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      // useCreateIndex: true,
    });
    app.listen(PORT, () =>
      console.log(`App has been started on port ${PORT}...`)
    );
  } catch (e) {
    console.log("Server Error", e.message);
    process.exit(1);
  }
}

start();
```

В папці конфіг знаходиться папка default.json де ми і вказуємо інформацію про базу даних, порт, та URL

```
{
  "port": 5000,
  "jwtSecret": // секретний ключ
  "mongoUri":
"mongodb+srv://root:<password>@cluster0.yzho9ys.mongodb.net/?retryWrites=true&w=majority",
  "baseUrl": "http://localhost:5000"
}
```

В папці models розташовані моделі, та схеми за якими відбуваються POST запроси на сервер(базу даних)

```
const { Schema, model, Types } = require("mongoose");

const schema = new Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  links: [{ type: Types.ObjectId, ref: "Link" }],
});

module.exports = model("User", schema);
```

Це є схема користувачів які реєструються на сайті

```
const { Schema, model, Types } = require("mongoose");

const schema = new Schema({
  name: { type: String },
  surname: { type: String },
  father: { type: String },
  position: { type: String },
  mailing: { type: Number },
  pnone: { type: Number },
  title: { type: String },
  type: { type: String },
  copies: { type: Number },
  numberOfPages: { type: Number },
  addInformation: { type: String },
});

module.exports = model("Order", schema);
```

А це схема замовлень які здійснюють користувачі.

В папці routes зберігаються роути, користувачів, там замовлень, простішою мовою логіка реєстрації та логіка замовлень.

Переходимо до клієнтської частини сайту в папці client знаходиться 3 папки, з налаштуванням, папка з html, та папка з програмною складовою (Рисунок А.3)

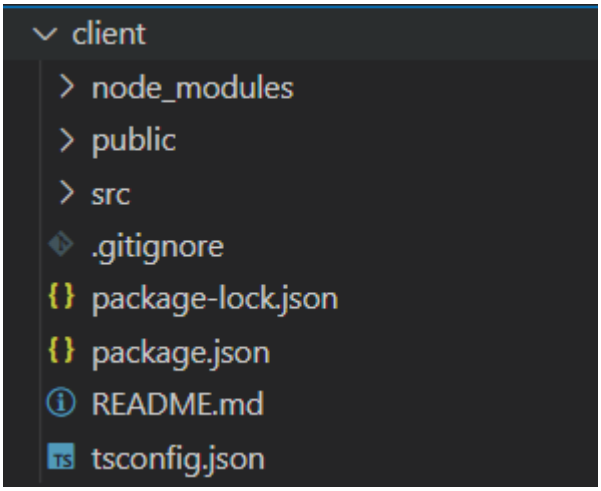

Рисунок А.3 – Клієнтська частина сайту

Сама головна папка, це папка src де і відбувається програмування та створення сайту та його налаштування (Рисунок А.4)

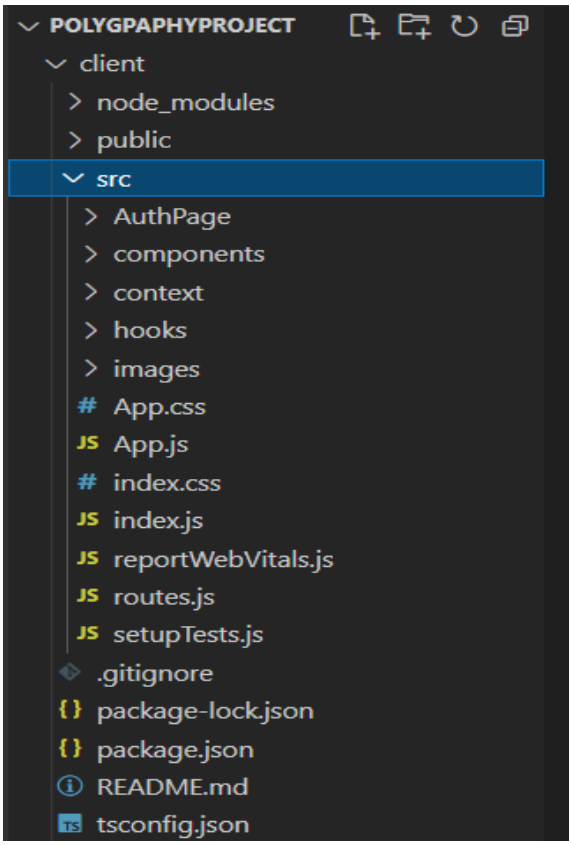

Рисунок А.4 – Структура клієнтської частини

Головним файлом є App.js який і відображається на сайті

```
import React from "react";
import { BrowserRouter } from "react-router-dom";
import { useRoutes } from "./routes";
import { useAuth } from "./hooks/auth.hook";
import { AuthContext } from "./context/Auth.Context";

function App() {
  const { token, login, logout, userId } = useAuth();
  const isAuthenticated = !!token;
  const routes = useRoutes(isAuthenticated);
  return (
    <AuthContext.Provider
      value={{ token, login, logout, userId, isAuthenticated }}
    >
      <BrowserRouter>
        <div> {routes} </div>
      </BrowserRouter>
    </AuthContext.Provider>
  );
}

export default App;
```

Потім іде файл з назвою routes.js де вписані усі компоненти програми які відображаються на сторінці.

```
import React, { useState } from "react";
import Header from "./components/Header/Header";
import AuthPage from "./AuthPage/AuthPage";
import { Route, Routes } from "react-router-dom";
import "./App.css";
import Main from "./components/Main/Main";
import About from "./components/About/About";
import Products from "./components/Products/Products";
import OrderPage from "./components/OrderPage/OrderPage";
import OrderPage2 from "./components/OrderPage2/OrderPage2";
import Footer, { StartingPage } from "./components/Footer/Footer";
import Header2 from "./components/Header2/Header2";
import Main2 from "./components/Main2/Main2";
import About2 from "./components/About2/About2";

export const useRoutes = (isAuthenticated) => {
  if (isAuthenticated) {
    return (
```

```
    <div className="wrapper">
      <div className="header">
        <div className="container">
          <div className="content">
            <Header />
          </div>
        </div>
      </div>
      <div className="wrapp">
        <Routes>
          <Route path="/Main" element={<Main />}></Route>
          <Route path="/About" element={<About />}></Route>
          <Route path="/Products" element={<Products />}></Route>
          <Route path="/OrderPage" element={<OrderPage />}></Route>
        </Routes>
      </div>
      {/* <div className="footer_footer">
        <div>
          <div className="footer">
            <Footer />
          </div>
        </div>
      </div> */}
    </div>
  );
}
return (
  <div className="wrapper">
    <div className="header">
      <div className="container">
        <div className="content">
          <Header2 />
        </div>
      </div>
    </div>
    <div className="wrapp">
      <Routes>
        <Route path="/" element={<Main2 />}></Route>
        <Route path="/SecondAbout" element={<About2 />}></Route>
        <Route path="/Products" element={<Products />}></Route>
        <Route path="/SecondOrderPage" element={<OrderPage2 />}></Route>
        <Route path="/Registration" element={<AuthPage />}></Route>
      </Routes>
    </div>
    {/* <div className="footer_footer">
        <div>
          <div className="footer">
            <Footer />
          </div>
        </div>
      </div> */}
```

```
    </div>
  );
};
```

Спочатку йде перевірка користувач авторизований або ні, якщо так показується перша частина коду, якщо ні показується друга частина коду.(Рисунок А.5)

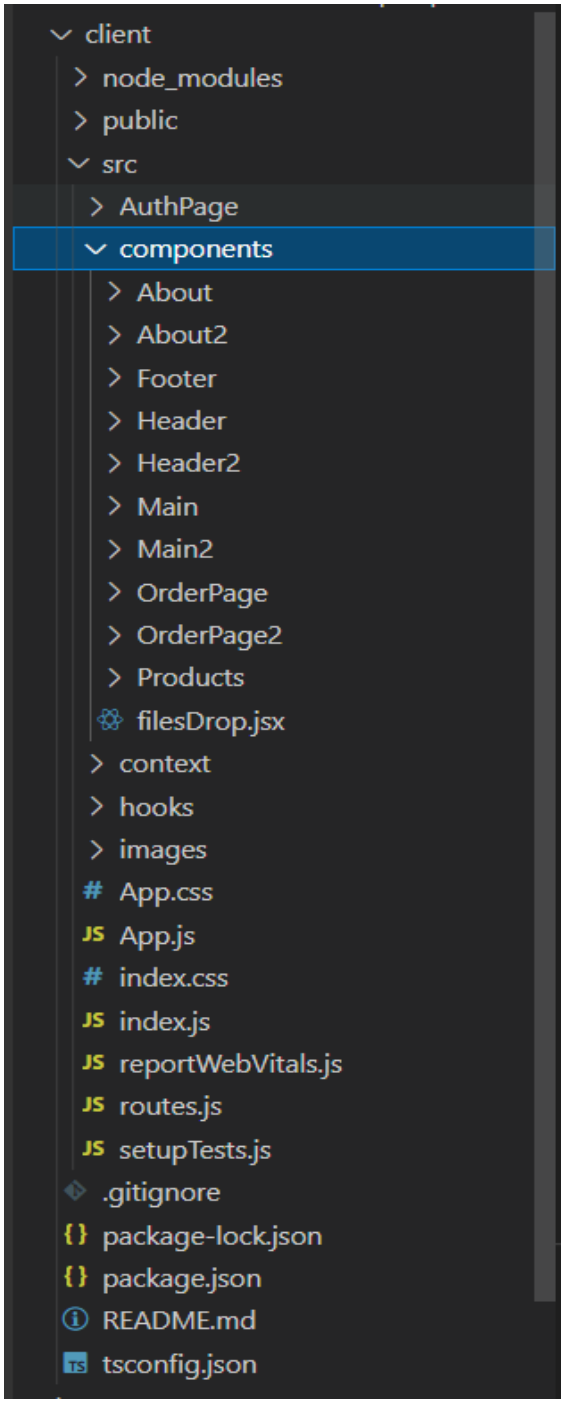

Рисунок А.5 – Папка components де знаходяться усі компоненти сайту