Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Стрижак Анастасії Олексіївни*
(ПІБ)

академічної групи *122-17-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка автоматизованої інформаційної систему для обліку та організації вантажних перевезень логістичного підприємства "1776 Logistics"*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*Удовік І.М.*			
розділів:				
спеціальний	*Удовік І.М.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-2* (група) *Стрижак Анастасії Олексіївни* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка автоматизованої інформаційної системи для обліку та організації вантажних перевезень логістичного підприємства "1776 Logistics"*

затверджена наказом ректора НТУ «ДП» від *21.05.2021 р.* № *770-л*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ _____
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Стрижак А.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 90 с., 14 рис., 0 табл., 3 додатка, 31 джерело.

Об’єкт розробки: розробка веб сайту з дашбордом для логістичних компаній.

Мета роботи: розробити веб-сайт для підприємства, яке займається логістикою. Основними цілями створення даного продукту є:

- полегшити отримання водіями, продавцями, діловими партнерами і співробітниками компанії актуальної інформації про поїздки, вантажі і машини;

- скоротити витрати на служби технічної або інформаційної підтримки, збільшити кількість дзвінків до підприємства;

- надати можливості для збільшення обсягу перевезень вантажів по Україні;

- створити можливість перегляду, постінгу і букінгу поїздок з комп’ютеру.

У вступі розглянуто проблему розробки веб-додатку для підприємства за допомогою HTML5, JavaScript та PHP мов програмування з використання інструментарію Bootstrap, а також необхідність і актуальність створення даного програмного продукту.

В першому розділі наведені основні причини необхідності створення веб-сайту, розглянуті та порівняні сучасні популярні інструменти для створення веб-сайтів та додатків.

В другому розділі описана методика створення веб-сайту. Методика містить перелік усіх етапів, які необхідні для конструювання сайту підприємства. Детально розглянуто інтерфейс системи та її можливості.

В економічному розділі визначено трудомісткість розробленого інформаційного ресурсу, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Список ключових слів: LANDING PAGE, BOOTSTRAP, RESPONSIVE WEB DESIGN, JAVASRIPT, URL, МЕТА-ТЕГ, ТЕГ, PHP, HTML, CSS, WEB-ФОРМА.

ABSTRACT

Explanatory note: 90 p., 14 pics., 0 tab., 3 add, 31 sources.

Object of development: development of a website with a dashboard for logistics companies.

Purpose: to develop a website for a company engaged in logistics. The main purposes of creating this product are:

- facilitate the receipt of drivers, sellers, business partners and employees of the company up-to-date information about travel, cargo and vehicles;

- reduce the cost of technical or information support services, increase the number of calls to the company;

- provide opportunities to increase the volume of cargo transportation in Ukraine;

- Create the ability to view, post and book trips from your computer.

The introduction discusses the problem of developing a web application for the enterprise using HTML5, JavaScript and PHP programming languages using Bootstrap tools, as well as the need and relevance of creating this software product.

The first section presents the main reasons for the need to create a website, discusses and compares modern popular tools for creating websites and applications.

The second section describes how to create a website. The methodology contains a list of all stages that are necessary for designing the site of the enterprise. The interface of the system and its capabilities are considered in detail.

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

Keyword list: LANDING PAGE, BOOTSTRAP, RESPONSIVE WEB DESIGN, JAVASRIPT, URL, META TAG, TAG, PHP, HTML, CSS, WEB FORM.

ЗМІСТ

ВСТУП

Кваліфікаційна робота повинна відображати знання, вміння і компетенції бакалавра, які він повинен був отримати на курсі університету на спеціальності 122 "Комп'ютерні науки". Робота полягає в створенні інформаційної системи, а саме веб-сайту і дашборду для логістичної компанії, що відповідає вимогам до неї.

Дашборд (англ. Dashboard; альт. Назви: "Панель управління", "Інформаційна панель") — це тип графічного призначеного для користувача інтерфейсу, який часто надає короткі відомості про ключові показники ефективності (англ. Key performance indicators, KPI), що відносяться до конкретної мети або бізнес-процесу. В інших випадках інформаційна панель — це інша назва звіту про хід роботи або звіту і вважається формою візуалізації даних. Панель управління часто доступна через веб-браузер і зазвичай пов'язана з регулярно поновлюваними джерелами даних.

Логістична галузь не сильно розвинена в Україні і, крім очевидних проблем, які цьому перешкоджають, таких як корупція і монополізація, водіям, диспетчерам і господарям вантажів також потрібна система, яка б спрощувала їх роботу. Дашборд активно використовується в логістичних компаніях за межами України, такими як США і Канада. Найбільший приклад з них — це Amazon Relay. Дана робота являє собою альтернативу цьому додатку для нашої країни.

Дашборд має такі переваги:

- Надає інформацію водіям про вантаж, місце і час навантаження, місце і час вивантаження, вимоги до транспортного засобу;
- Дає можливість вантажовласникам знайти перевізника онлайн;
- Спрощує роботу диспетчерів при побудові маршруту;
- Спрощує роботу бухгалтерів при підрахунку зарплат водіям;
- Дає можливість знайти і проїхати найбільш зручний маршрут в будь-якій точці країни;

- Економить ресурси, так як з обслуговуючого персоналу потрібно три зміни по одній людині в диспетч відділі і близько двадцяти людей у відділі підтримки, повний штат 24/7 стає непотрібним.

Актуальність і сфера використання. Дашборд є зручним і необхідним інструментом для ведення бізнесу в сфері логістики. Він корисний всім сторонам, які беруть участь в даній галузі. Водій може дивитися свою поїздку онлайн через мобільний пристрій, диспетчер менше користуватися телефонією, вибудовуючи маршрут, власник вантажу при належному рейтингу буде завжди забезпечений перевізниками. Також, як і водій (компанія-перевізник) при належному рейтингу завжди буде забезпечений роботою. Компанія-власник даного веб-сайту може мати і своїх водіїв, вантажі, склади і вони будуть в пріоритеті, так як є перевіреними.

Метою даного проекту є покращення умов роботи в логістичній галузі в Україні, а також її безпосередній розвиток.

Для досягнення поставленої мети в роботі ставились та вирішувались такі основні завдання:

- аналіз предметної області;
- уточнення вимог до дашбордів;
- розробка алгоритму роботи веб-сайту.

У відповідності до проведеного аналізу в роботі поставлені наступні функціональні задачі та вимоги до розроблюваного веб-сайту:

- зручний інтуїтивно зрозумілий інтерфейс;
- робота на ПК, ноутбуках, мобільних пристроях різних конфігурацій та під управлінням різних ОС;
- не бути вимогливим до складу програмних та апаратних засобів;
- витримувати велику кількість одночасних користувачів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Різноманітні дані всюди навколо нас. Згідно з дослідженням EMC Digital Universe, до 2020 року в нашому цифровому ландшафті існує близько 40 трильйонів мегабайт або 40 зеттабайт. Це незбагненний обсяг інформації.

Дані багато в чому змінили життя людства, допомагаючи поліпшити процеси, ініціативи та інновації в організаціях у всіх секторах за рахунок здатності проникнення в суть. Але сьогодні є так багато статистичних даних, фактів і цифр, що без додаткових інструментів більшість даних просто знаходяться в великому безпорядку чи ще гірше — загублюються.

Використовування інформаційних панелей — одне з кращих досягнень в області бізнес-аналітики.

Панель даних — це інструмент, який надає централізовані інтерактивні засоби моніторингу, вимірювання, аналізу та вилучення релевантної бізнес-інформації з різних наборів даних в ключових областях, відображаючи інформацію в інтерактивному, інтуїтивно зрозумілий і візуального огляду.

По суті, що означає дашборд? Він пропонує користувачам вичерпний огляд різних внутрішніх відділів, цілей, ініціатив, процесів або проектів своєї компанії. Вони вимірюються за допомогою ключових показників ефективності (KPI), які забезпечують розуміння, яке допомагає стимулювати зростання і поліпшення.

Онлайн-інформаційні панелі забезпечують негайний доступ з можливістю навігації до аналітики, яка може підвищити прибуток підприємств за рахунок безперервної комерційної еволюції.

Щоб правильно визначити інформаційні панелі, треба враховувати той факт, що без існування інформаційних панелей і методів звітності на інформаційних панелях компаніям довелося б просіяти колосальні стопки неструктурованих даних, що неефективно і потребує багато часу. В якості

альтернативи бізнесу доведеться діяти, маючи надію тільки на випадковість у найбільш важливих процесах, проектах і внутрішніх ідеях, що далеко не є ліпшим варіантом для розвитку бізнесу і отримання прибутків.

Тепер розглянемо одну з основних функцій панелей даних: відповіді на важливі бізнес-питання. Отже, як і головна функція приладової панелі?

Як згадувалося раніше, панель управління даними може відповідати на безліч питань, пов'язаних з бізнесом, в залежності від конкретних цілей, завдань і стратегій.

Взявши необроблені дані з різних джерел і консолідуючи їх, перш ніж представляти їх на настоюваної панелі моніторингу, панелі моніторингу даних можуть допомогти зрозуміти найбільш цінні дані компанії і дати можливість знайти дієві відповіді на найгостріші питання бізнесу.

За рахунок зв'язування з конкретними ключовими показниками ефективності, які відповідають бізнес-цілям компанії, вона може заглиблюватися в певні галузі інформації, створювати контрольні показники і на постійній основі вимірювати свій успіх.

При цьому бізнес компанії буде управлятися даними і, як прямий результат, стане більш успішним.

Теперь розглянемо пример дашборду логістичної сфери, найбільш розповсюджений в Північний Америці. Relay — це спроба Amazon автоматизувати процес Load Board, підключившись до мільйонів водіїв вантажівок (рис. 1.1.). За даними Convoy, провідного ради по завантаженню для власників-операторів, ринок вантажоперевезень оцінюється в 800 мільярдів доларів, і не дивно, що Amazon зайнявся цією галуззю.

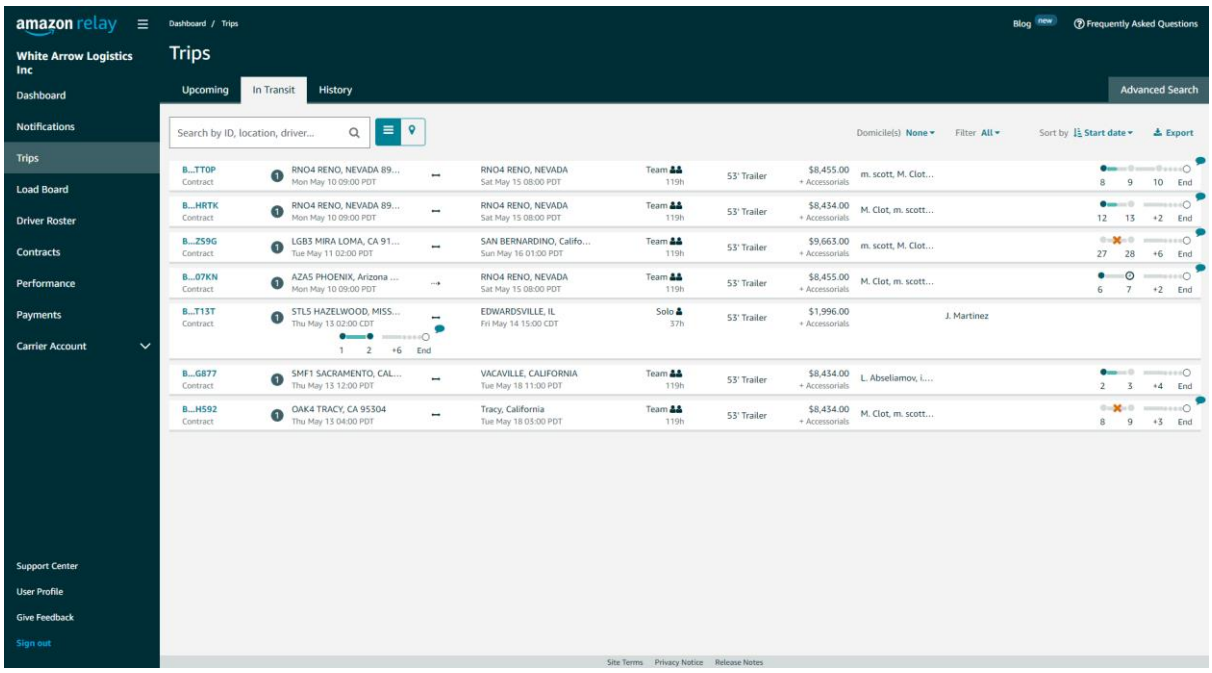

Рис. 1.1. Amazon Relay з точки зору перевізника

Amazon Relay дуже популярний серед нових операторів-власників, бо цей інформаційний ресурс зручний і зрозумілий.

При цьому Load Boards, такі як Amazon Relay Load Board (рис. 1.2.), служать важливою частиною для багатьох власників-операторів, особливо для тих, кому потрібна швидке завантаження на більш високооплачуваний ринок і повернення додому, який приносить прибуток.

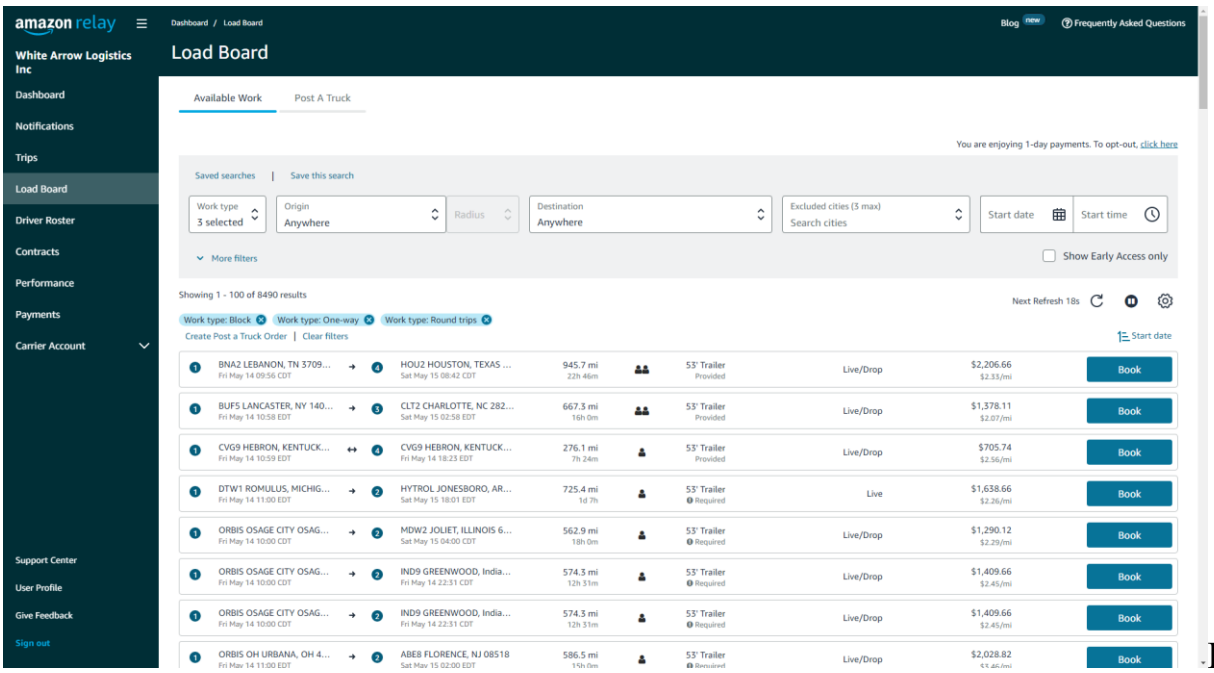

Рис. 1.2. Amazon Load Board з точки зору перевізника

Схожий веб-сайт має назву Power DAT (рис. 1.3.), але не такий зручний, як амазон. Він має меньший функціонал, але їм усе одно користуються що перевізники, що власники вантажу.

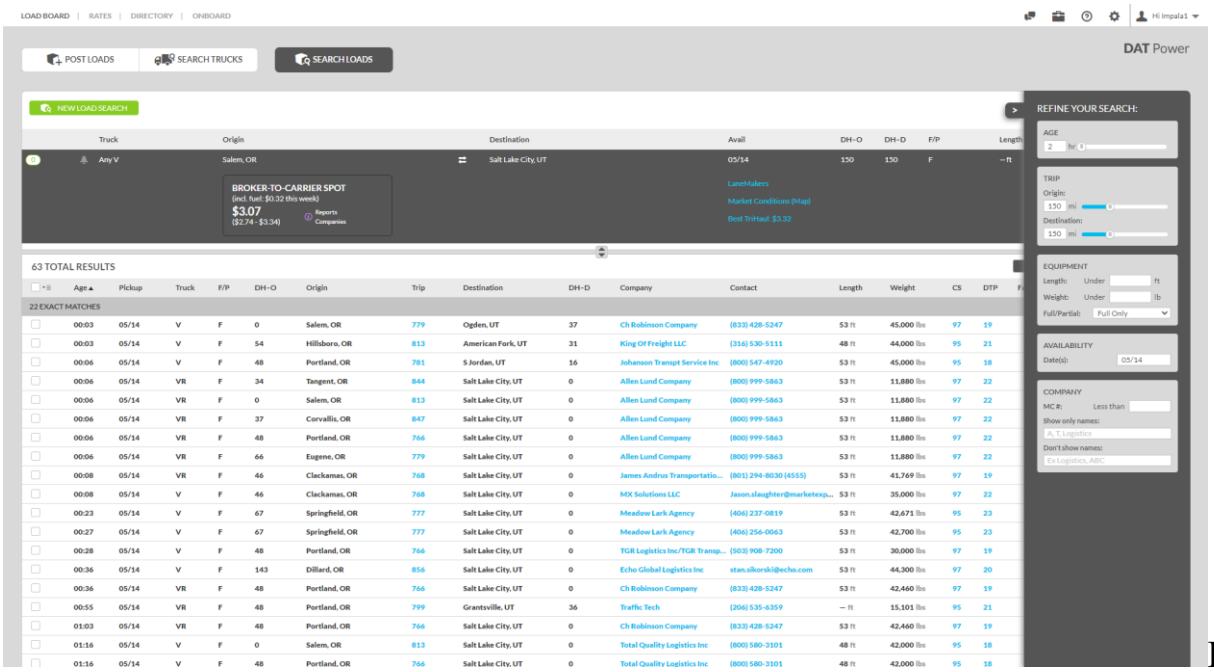

Рис. 1.3. Power DAT з точки зору перевізника

Підводячи підсумки, можна визначити основні вигоди від створення дашборду для території України. Сайт с дашбордом дає можливість потенційним водіям і власникам вантажів знаходить як компанію-власника, так і друг друга, інформація на сайті є дуже ефективною і може надати обширну інформацію про діяльність компанії, про її вантажі і послуги. Інформаційна панель працює цілодобово і вимагає мінімальних витрат на свій зміст, розвинений дашборд дає можливість працівникам мати ще один спосіб знайти працю, а власникам перевезти свій товар, створення сайту поліпшить імідж компанії-власника. Мінімальні витрати на розробку, створення і підтримку сайту не можуть бути істотним мінусом.

Основними вимогами, які ставлять звичайні користувачі до онлайн-дашборду як до сайту є:

- зрозумілий інтерфейс та зручна система навігації по лоад борду;
- зручна система посилань, що дозволяє оптимальним способом одержати необхідну користувачу інформацію;
- мінімальна кількість дій користувача для здійснення покупки.

Також переваги використання онлайн-дашбордів включають:

- візуальна презентація показників ефективності;
- можливість виявлення та корекції негативних тенденцій;
- вимірювання ефективності/неефективності;
- можливість створення детальних звітів, що відображають нові тенденції;
- можливість приймати більш обґрунтовані рішення на основі зібраної ділової інформації;
- вирівняти стратегії та організаційні цілі;
- заощаджує час порівняно із створенням декількох звітів;
- миттєве отримання загальної відомості про всі системи;

- швидке визначення викидів даних та кореляцій;
- зведення звітності в одному місці;
- доступність на мобільних пристроях для швидкого доступу до показників.

Інформаційні панелі можуть бути розбиті залежно від ролі і можуть бути стратегічними, аналітичними, оперативними чи інформаційними. Інформаційні панелі — це третій крок на інформаційній сходах, який демонструє перетворення даних у все більш цінні ідеї.

Стратегічні інформаційні панелі підтримують менеджерів на будь-якому рівні організації та забезпечують швидкий огляд того, що особи, що приймають рішення, повинні контролювати стан здоров'я та можливості бізнесу. Інформаційні панелі цього типу орієнтовані на високі показники ефективності та прогнози. Стратегічні інформаційні панелі отримують користь від статичних знімків даних (щодня, щотижня, щомісяця та щокварталу), які не змінюються постійно з одного моменту в інший.

Інформаційні панелі для аналітичних цілей часто включають більше контексту, порівнянь та історії, а також тонкі оцінювачі ефективності. Аналітичні інформаційні панелі зазвичай підтримують взаємодію з даними, наприклад, деталізацію деталей, що лежать в основі. Інформаційні панелі для моніторингових операцій часто розробляються інакше, ніж ті, що підтримують прийняття стратегічних рішень або аналіз даних, і часто вимагають моніторингу діяльності та подій, які постійно змінюються і можуть вимагати уваги та реакції в будь-який момент.

1.2. Призначення розробки та галузь застосування

Веб-платформа інтернет-борда призначена для використання у сфері електронної комерції (торгівлі) та логістики, за допомогою мережі Інтернет. Основним завданням даної розробки являється можливість онлайн продажу вантажів перевізникам.

Розроблений проект повинен відповідати сучасним стандартам, та задовольняти вимоги продиктовані замовником, а також бути зручним для кінцевого користувача, мати інтуїтивно зрозумілий інтерфейс для користувача, працювати на різних пристроях різних конфігурацій, та під управлінням різних ОС.

1.3. Постановка завдання

Оскільки все більша кількість людей користується мережею Інтернет і віддають перевагу зручності, метою даної кваліфікаційної роботи є створення інтернет-дашборда и лоад борда, що надає можливість замовникові зручно керувати бізнесом, а користувачу купувати необхідні йому вантаж не залишаючи свого офісу чи транспорту.

Розробка даного програмного рішення включає в себе:

- Створення дизайну веб-сайту;
- Створення системи управління – движка;
- Створення бази даних.

Загалом, цей веб-сервіс складається з двох частин:

- Front-end – видима частина сайту, інтерфейс користувача та його дизайн;
- Back-end – система управління сайтом, частина сайту що являється закритою для користувача і відповідає за управління функціоналом інформаційної панелі.

Сама ж система також поділена на три частини:

1. Частина користувача-перевізника – дана частина знаходиться в доступності будь-якому користувачеві.
2. Частина користувача-власника вантажу – даний розділ системи доступний лише для авторизованих користувачів.

3. Частина адміністратора – цей розділ системи доступний лише по паролю адміністратору сайту.

1.4. Вимоги до функціональних характеристик

Функціонал сайту – це найважливіша частина в розробці даного проекту, на яку слід звернути особливу увагу. Чим оптимальніше набір функцій лоад борду – тим комфортніше почувають себе клієнти.

В залежності від ролі користувача можуть змінюватись вимоги до набору функцій. Проте цей набір для більшості лоад бордів буде схожим.

Весь функціонал можна розділити на декілька груп за їх призначенням:

- пошуковий функціонал;
- демонстраційний функціонал – для представлення вантажів перевізникам та вантажівок власникам вантажу;
- інформаційний функціонал – відповідальний за інформування клієнтів;
- SEO-функціонал, який спрощує просування інтернет-магазину в пошукових системах;
- комунікаційний функціонал – відповідальний за спілкування з клієнтами.

Пошукові функції, такі як, пошук товарів, сортування, фільтрація і підбір товарів являються одними із самих важливих функцій лоад борду, чим зручнішим буде даний функціонал.

Пошук вантажів чи вантажівок – зазвичай передбачає наявність функцій для пошуку вантажу і вантажівок в списку на лоад борді. Частіше за все для цього використовується пошуковий рядок, але в цій роботі за це буде відповідати окрема сторінка, де можна буде знайти і букнути (від англ. book) потрібний вантаж чи запостить вантажівку.

Сортування вантажу – передбачає функцію по упорядкуванню поїздок в списку, виходячи з таких параметрів як:

- ціна – користувач буде обирати мінімальну ціну у фільтрі;
- ціна за кілометр — щоб клієнт бачив, підходить йому загальна ціна чи ні;
- локація пикапа;
- локація делівері;
- тип поїздки — кругова поїздка, поїздка в одну сторону, блок;
- тип водія — команда чи соло;
- тип вантажівки — 26’, 28’, 53’;
- статус трейлера — потребується чи надається;
- Дата і час поїздки.

Функція фільтрації поїздок також відноситься до однієї із необхідних, оскільки більшість клієнтів шукають поїздку по необхідним їм категоріям та характеристикам.

Фільтрація і підбір поїздок – передбачає вибір зі списку за необхідними параметрами.

Демонстраційний функціонал – для більшості випадків для демонстрації поїздки достатньо лише в текстовому вигляді перерахувати всі її деталі та дату пікапа, а також показати те, скільки отримує перевізник.

Також досить корисна функція – це схожі поїздки. Так як перевізник не завжди може знайти потрібний йому поїздку, лоад борд має надавати йому інші варіанти, які є близькими по локаціям та ціні до тих параметрів, які ввів користувач.

Інформаційний функціонал. Розсилка новин юзерам. Простий і стандартний модуль для любого сайту, який додатково стимулює клієнтів до використання ресурсом. Необхідно пам'ятати, що великі інтернет сайти, будь то магазини чи лоад борди, 80% свого прибутку отримують за рахунок постійних клієнтів.

Комунікаційний функціонал. Комунікація з клієнтами (чати, месенджери, консультанти). «З клієнтом треба працювати» — це девіз будь-якого продавця. Для цього потрібно надавати покупцям якісну технічну підтримку і в штаті співробітників повинен бути як мінімум один менеджер-консультант. Однак, це організаційне питання, а ось питання технічне — це забезпечити дашборд необхідним комунікаційним функціоналом. У ТЗ на створення дашборду потрібно чітко прописати наявність комунікаційних інструментів, інакше можна втратити клієнтів.

Зараз є безліч варіантів подібного функціоналу, і використовувати всі варіанти на одному сайті вкрай нерозсудливо, так як при занадто широкому виборі клієнт може загубитися. Однак, наявність швидких і безкоштовних засобів зв'язку обов'язково. Отже, що може бути корисно:

- онлайн-чати на сайті інтернет-магазину (зазвичай у вигляді чату з консультантом);
- зворотній зв'язок;
- ICQ-менеджер;
- Skype;
- безкоштовний телефон (зазвичай для дорогих проектів);
- дзвінок з сайту;
- замовлення зворотнього дзвінка.

SEO-функціонал. Наявність SEO функцій обов'язково, так як будь-який серйозний проект повинен просуватися в пошукових системах, а без пошукової оптимізації гарних позицій в пошуковій видачі не бачити. До SEO функцій можна віднести можливість прописаний унікальних мета тегів опису, ключових слів, тайтлів і ЧПУ. Також варто відзначити валідність коду, тобто його чистоту і правильність написання. Необхідно, щоб скрипт сайту не генерував дублів сторінок сайту.

Мобільна версія лоад борду. Не можна назвати даний пункт функцією, так як мобільна версія створюється або окремо, або скрипти інтернет-магазину

прописуються так, щоб коректно працювати в браузерах мобільних телефонів. Величина мобільних користувачів інтернету з кожним днем зростає і необхідно пам'ятати про це, щоб не втрачати покупців.

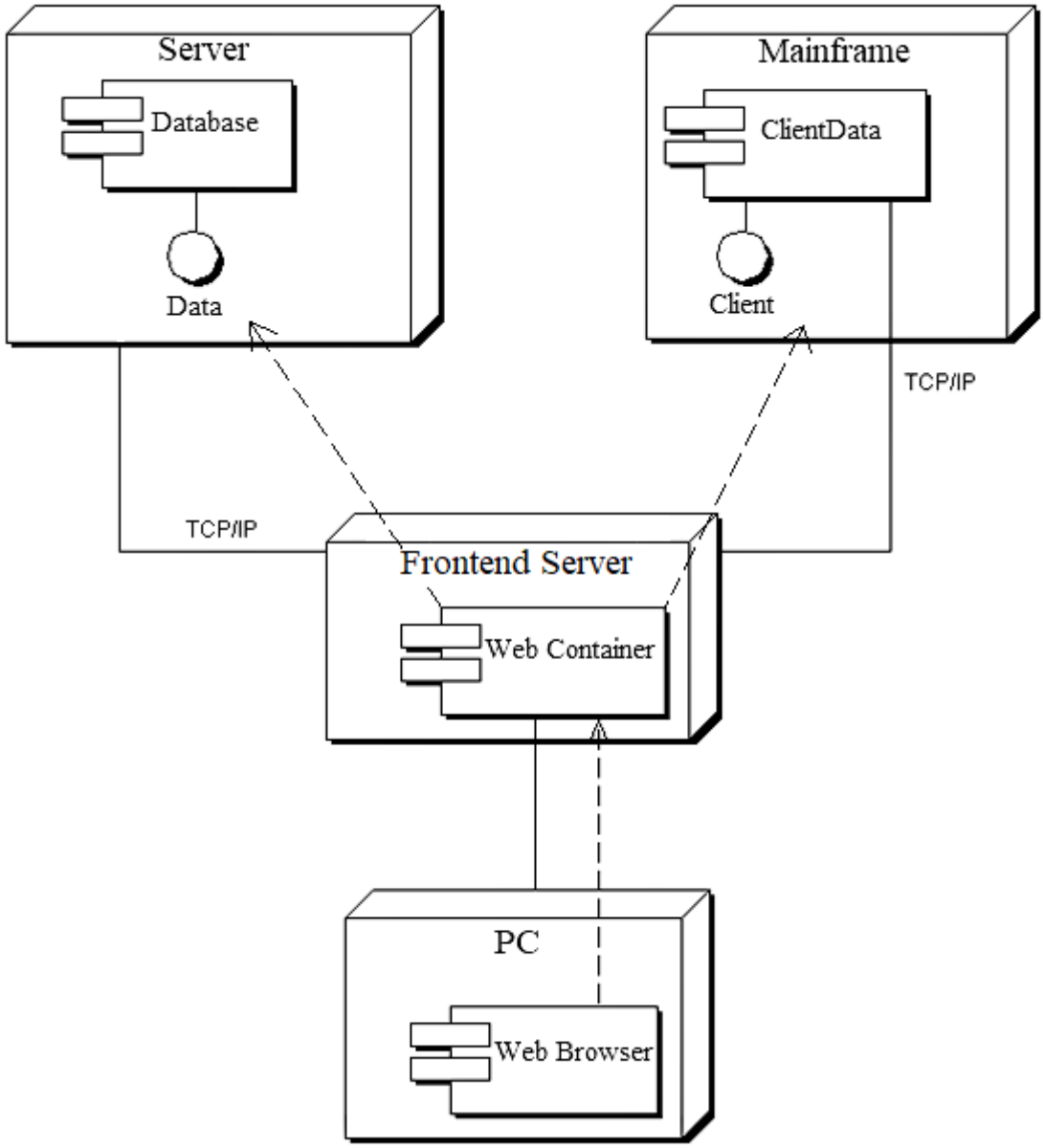

Рис. 1.4. Діаграма розгортання проекту

1.4.1. Вимоги до функціональних характеристик частини користувача-перевізника

Користувач-перевізник — одна з трьох груп користувачів, для опису вимог якої потребується чітке розуміння, що ця група хоче і що цій групі потрібно. Перевізник — це компанія або особа, яка на законних підставах має право перевезення вантажів наземним, водним та повітряним транспортом. Зазвичай

перевізник працює з вантажовідправниками, щоб перевозити вантажі з одного місця в інше.

Існує два основних типи перевізників або способів доставки товарів:

- Звичайний перевізник: відноситься до постачальника транспорту, який пропонує свої послуги будь-якій особі чи компанії, оскільки він має право це робити за ліцензією, наданою регулюючим органом. Звичайний перевізник може працювати з більшою кількістю вантажовідправників протягом одного дня, оскільки він не зобов'язаний жодним контрактом.
- Контрактний перевізник: стосується компанії або особи, яка надає транспортні послуги для зазначеного вантажовідправника на довгостроковій основі. Це означає, що перевізник досягає спільної згоди з вантажовідправником та погоджується працювати за певних умов протягом строку дії контракту.

Для цих видів перевізників потрібно дві різні сторінки. Для звичайних перевізників — лоад борд (вантажна дошка), а для контрактних — контракти. Функціонал у них схожий, але присутня різниця у самій основі цих двох типів перевізників. Хоч контракт виглядає, як звичайний блок, чи довга кругова поїздка, він таким не є. По-перше, контракт повинен починатись і закінчуватись на одній и тій самій локації. По-друге, контракт береться на кілька тижнів, або навіть місяців. І компанія-власник цього ресурсу буде віддавати перевагу саме контрактам, тому що їх буде простіше побудувати і перевізник не буде мати права відмовитись від усього контракта через не сподобавшуюся ногу чи вантаж, бо в такому випадку по контракту перевізник має заплатити штраф.

Всі поїздки, котрі не були включені в контракти, стають блоками, круговими поїздками та поїздками в одну сторону. Їх можна букати з лоад борду і вони є одноразовими.

Зверху сторінок контрактів та лоад борду має бути фільтр, котрий вже був описаний вище. Знизу має бути список доступних поїздок, відповідних до параметрів пошуку. В самому низу — поїздки, які близькі до параметрів пошуку, але не зовсім їм відповідають. Така функція потрібна для того, щоб не загубити клієнтів, тобто якщо користувач не може знайти потрібні йому поїдки, він не виходив з сайту, а міг побачити альтернативні варіанти. Також У перевізника буде така функція, як “запостити вантажівку”. Щоб додати вантажівку треба заповнити форму, де будуть такі поля:

- Номер;
- Модель;
- Тип;
- Локація;
- Тип водія.

1.4.2. Вимоги до функціональних характеристик частини користувача-власника вантажу

Користувач-власник вантажу хоче, щоб його вантаж як можна скоріше був перевезений в те місто чи будь-який інший населений пункт. Йому важливо, щоб перевезення було швидким та безпечним. Тому на його борді буде трохи інший фільтр. Для цього користувача буде одна унікальна вкладка “трак борд” чи “доступні вантажівки”. Вони будуть сортуватися по бажанню користувача, але є важлива відміна від того же лоад борду: в списку вантажівки перевізника обов’язково з боку мають рейтинг перевізника, який складається з кількох факторів, таких як:

- Пунктуальність;
- Прийняття;
- Безперебійність.

Від 0% до 89.9% — рейтинг за оцінкою C, “рискований перевізник”, відображається в самому низу.

Від 90% до 97.9% — рейтинг за оцінкою B, “могло бути краще”, відображається у середині.

Від 98% до 100% — рейтинг за оцінкою A, “надійний перевізник”, відображається у самому верху за списком.

Подібний рейтинг також реалізований у власників вантажів. Якщо це вантаж від компанії-власника ресурсу, то він буде відображатися зверху. А далі йдуть компанії, які сортуються по рейтингу, якій складається з кількох факторів:

- Безпека;
- Пунктуальність;
- Надійність.

Безпека — якість вантажу, відсутність кримінальних подій на території складу, чистота і якість доріг на складах.

Пунктуальність — готовність вантажа к тому часу, який вказаний у розкладі.

Надійність — відсутність афер в історії компанії.

Рейтинг будується за такою ж схемою, як і у перевізників:

Від 0% до 89.9% — рейтинг за оцінкою C, “рискований власник вантажу”, відображається в самому низу.

Від 90% до 97.9% — рейтинг за оцінкою B, “могло бути краще”, відображається у середині.

Від 98% до 100% — рейтинг за оцінкою A, “надійний власник вантажу”, відображається у самому верху за списком.

1.4.3. Вимоги до функціональних характеристик частини адміністратора

Адміністратор повинен мати доступ к усім веб-сторінкам, які існують на сайті, також в його роль входить можливість коригувати будь-яку інформацію на будь-якій сторінці. Також у адміністраторів повинна бути своя додаткова сторінка, де вони зможуть побачити заявки от інших користувачів та їх контактні дані.

1.5. Вимоги до інформаційної безпеки

При роботі в веб-сайтом досить встановлених в системі (ПК, ноутбука, мобільних пристроїв) вимог до інформаційної безпеки. Особливих вимог до інформаційної безпеки при роботі з сервісом не встановлюється. Веб-сервіс працює через браузер користувача і не потребує особливих прав для його використання.

В свою чергу при розробці сайту слід звернути увагу на деякі аспекти.

Хостинг — в першу чергу необхідно вибрати надійного хостинг-провайдера. Важливо щоб ваш хостинг-провайдер підтримував регулярне резервне копіювання; вів всебічні журнали дій; виконував моніторинг мережевої активності. Також одним з важливих чинників є система повідомлень про аномальні діях на акаунті, можливе зараження сайту і так далі. Технічна підтримка повинна повідомити про порушення та забезпечити хоча б мінімальними інструкціями про методи вирішення виниклої проблеми і сприяти в її вирішенні.

Використовувати захищене з'єднання — шифрування каналу зв'язку між сайтом і браузером клієнта для передачі інформації. В наш час актуальним є використання TLS (Transport Layer Security - безпека транспортного рівня), який за звичкою багато хто до цих пір називають SSL (Secure Sockets Layer - рівень захищених сокетів).

Важливо використовувати останні (актуальні) версії криптографічних протоколів для належного захисту даних.

Відмінною практикою буде використання HSTS (HTTP Strict-Transport-Security) — механізм, який активує форсоване захищене з'єднання по HTTPS. Дана політика безпеки дозволяє відразу ж встановлювати безпечне з'єднання, замість використання HTTP. Механізм використовує особливий заголовок HTTP Strict-Transport-Security, для перемикання користувача, який зайшов по HTTP, на HTTPS-сервер.

1.6. Вимоги до складу та параметрів технічних засобів

Дашборд являється веб-сайтом, який є невибагливим до технічних характеристик пристрою і повинен працювати на пристроях з будь якою ОС, та конфігурацією. Основними умовами являється наявність встановленого веб-браузеру, та можливість виходу до мережі Інтернет. Також пристрій повинен мати дисплей с розширенням 320x240px, або більше.

1.7. Вимоги до інформаційної та програмної сумісності

Для роботи серверної частини веб-сайту дашборда, сервер має підтримувати роботу с наступними модулями:

- Apache HTTP Server – версії 2.4 та вище;
- PHP – версії 7.1;
- MySQL – версії 8.0 та вище.

1.8. Порівняння інструментів для створення сайту

В наш час існує багато різноманітних способів створення сайту. Далі описано декілька з них :

- конструктори сайтів;
- frameworks (CSS frameworks);
- системи управління контентом.

Розглянемо переваги та недоліки цих методів:

1. Конструктори сайтів

Переваги:

- достатньо пройти просту реєстрацію, а хостинги і домени залишаться в минулому;
- це нічого не буде коштувати, тому не потрібно турбуватися про початковий капітал та великі вкладення;
- створення сайту за допомогою сервісу не займе багато часу і вже через годину-дві ви можете отримати першу версію свого сайту;
- Якщо у користувача виникнуть питання в процесі роботи з конструктором — він може звернутися за допомогою до технічної підтримки;
- так, як над створенням шаблонів працювали дизайнери, то користувач може отримати стильний цікавий сайт з гармонійної структурою і зовнішнім виглядом;
- в шаблонах вже закладено механізм для оформлення замовлень, а також прийому платежів.

Це і є основні плюси використання конструкторів для створення сайтів, але не дивлячись на цей перелік існує безліч мінусів в даному підході, про які піде мова нижче.

Недоліки:

- безкоштовно можна отримати тільки доменні імена третього рівня, це означає, що на великі доходи від цього сайту можна не розраховувати;
- сервіс для конструктора може показувати відвідувачам рекламу на правах безкоштовного надання шаблонів (якщо компанія хоче позбутися від подібних банерів, потрібно буде окремо платити);
- перенесення сайту на інший хостинг також зажадав додаткових грошових вкладень;

- в конструкторі можна створити сайт з обмеженим функціоналом. тому потрібно буде докуповувати до базового набору додаткові плагіни, скрипти і т.д .;
- обмежені можливості для SEO просування (не унікальність, поведінковий фактор, безкоштовний домен і хостинг все це буде тим, в чому компанія буде програвати перед іншими учасниками, які створювали сайт з нуля);
- престижність — у користувачів може формуватися упереджене ставлення до компанії, яка не вкладає кошти у свій розвиток і вдосконалення, а значить, формується негативне сприйняття ресурсу, який не приносить прибуток.

2. Frameworks (CSS frameworks)

Фреймворк — заготовки або шаблони для програмної платформи, що визначають архітектуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних модулів програмного проекту. Доречно використання терміну «каркас».

Говорити про каркасний підхід можна, як підхід до побудови програм, де будь-яка конфігурація програми будується з двох частин:

- постійна частина — це каркас, незмінний від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина;
- змінні модулі (або точки розширення).

CSS-фреймворк — це бібліотека, що має на увазі більш просте, сумісний зі стандартами оформлення сторінок, з використанням каскадних таблиць стилів. Так само як і в різних мовах програмування, такий фреймворк — це набір готових прийомів, який, в даному випадку, використовується для верстки веб-сторінок.

Однією з переваг при виборі даного інструменту є те, що існують вже готові рішення для створення веб-сторінок, для різного виду дизайну та структури.

Мінусами є те, що в них буває дуже багато зайвого коду, яких фактично ніде не використовується.

3. Системи управління контентом

«Система управління контентом», кажучи простою мовою, CMS (або як ще часто її називають "движок сайту") — це конструктор, за допомогою якого сайт змайструвати зможе будь-хто, якщо не пошкодує на освоєння цього конструктора кілька годин. Для дизайну сайту використовуються готові шаблони, які, звичайно, можна редагувати під свої потреби, але це робиться дуже рідко.

Основними плюсами створення веб-сайту даним методом є те, що:

- це безкоштовно або дуже дешево. Тобто для створення сайту, наприклад, на Joomla, не потрібно платити величезні гроші розробнику, так як система досить проста, щоб все зробити самостійно;
- це швидко, для створення сайту на CMS буде потрібно кілька годин або днів (в залежності від складності сайту) і не потрібно витрачати величезні ресурси на складання плану архітектури сайту, планувати базу даних для сайту, розробляти дизайн і так далі;
- велика кількість розширень, плагінів. Наприклад, щоб зробити до сайту галерею фотографій, досить просто знайти в інтернеті відповідний плагін для вашої CMS, не більше того;
- CMS популярні, а це в свою чергу означає, що якщо у користувача виникнуть труднощі, то він завжди зможете знайти людей, які також використовують вашу CMS, і які вже стикалися з такою проблемою. Як правило таких людей дуже багато, і рішення можуть бути універсальними для всіх.

Незважаючи на всі вище перераховані плюси, у таких систем є і досить вагомі мінуси, а саме:

- користувач не знаєте, як це працює. Незважаючи на те, що асортимент плагінів і розширень для популярних CMS дуже великий, завжди можна знайти те, що нам запропонувати не зможуть. І ось коли виникає якесь нетипове завдання, що вимагає спеціалізованої роботи (наприклад, створити менеджер завдань для мети), то компанія просто потрапить в глухий кут. Розбиратися у внутрішніх кодах Joomla — це невдячна робота. І виходить, що користувач дуже жорстко обмежен в своїх можливостях. Як наслідок — зробити можна багато, але не все;
- безпека сайту дуже низька, оскільки CMS системи дуже популярні, їх внутрішню архітектуру знають всі. А це означає, що будь-яка людина знає і бачить наскрізь, як зроблений сайт. Щодня в інтернеті з'являється купа статей, як зламати Joomla або Wordpress, причому для того, щоб зламати сайт, не обов'язково бути хакером;
- внутрішня структура сайту неоптимізована. Наприклад, на Joomla на кожну сторінку створюється кілька копій такої сторінки. Будь-який фахівець з просування сайтів пояснить, чому пошукові системи не просувають сайт на верхні сходинки пошукової видачі, і з цим нічого неможна поробити;
- будь-який зайвий крок може усе зламати, оскільки у кожної CMS постійно виходять нові версії, то і всі плагіни, і розширення стають дуже залежні від цих версій. Це означає, що встановивши плагін не тієї версії на свою CMS користувач зможе отримати на додачу купу багів або зовсім зробити сайт неробочим. Потім докопатися до причини може стати дуже важко;
- як уже було згадано, сайти на відомих CMS дуже не люблять пошукові.
- оскільки пошукові роботи бачать, що сайт створений на популярній CMS, то він не сприймає його серйозно.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Опис основних етапів розробки сайту

Для початку нам необхідно визначитися з типом сайту. Кілька різновидів залежать від намічених цілей і функцій, які планується покласти на нього. За характером цілей сайти поділяються на комерційні та некомерційні.

Ресурси, в основному пов'язані з організацією продажу товарів і послуг, рекламним PR компанії, носять комерційний характер. Найтиповіші їх представників:

- лендінги – сайти спрощеного змісту, візитки з дуже конкретною інформацією про компанії (рис. 2.1.1.). Як правило, лендінг складається з однієї сторінки, на якій розміщена вся необхідна користувачеві інформація про товар або послугу. Основна мета такої сторінки – привертати увагу клієнта, мотивувати його зробити замовлення. Вона може пропонувати

користувачеві товар або послугу, а може агітувати за підписку, наприклад за розсилку. Все залежить від того, на яке цільове дію він орієнтується;

Рис. 2.1.1. Приклад лендингу

- інтернет-магазини – сайти, які орієнтовані виключно на продаж товару або послуги (рис. 2.1.2.). Це форма електронної торгівлі, яка дозволяє споживачам купувати товари або послуги за допомогою веб-браузера. Споживачі знаходять цікавить товар, відвідуючи сайт роздрібного продавця безпосередньо або шляхом пошуку серед альтернативних постачальників. Торгівля може відбуватися через невеликий локальний магазин, великого роздрібного продавця, магазин електронної комерції або приватна особа, яка продає товари через сторонній сервіс. Завдяки цьому клієнти інтернет-магазину мають можливість переглянути описи і вибрати товари, покласти їх в корзину, сформувати замовлення в особистому кабінеті, вибрати засіб оплати і доставки, оплатити замовлення. Інтернет-каталог повинен мати просту навігацію, щоб користувач не заплутався і міг швидко знаходити потрібні

йому товари. Тому структура будь-якого інтернет-магазину формується за принципом «від простого до складного» і включає велику кількість веб-сторінок, які виконують інформаційну або практичну функцію;

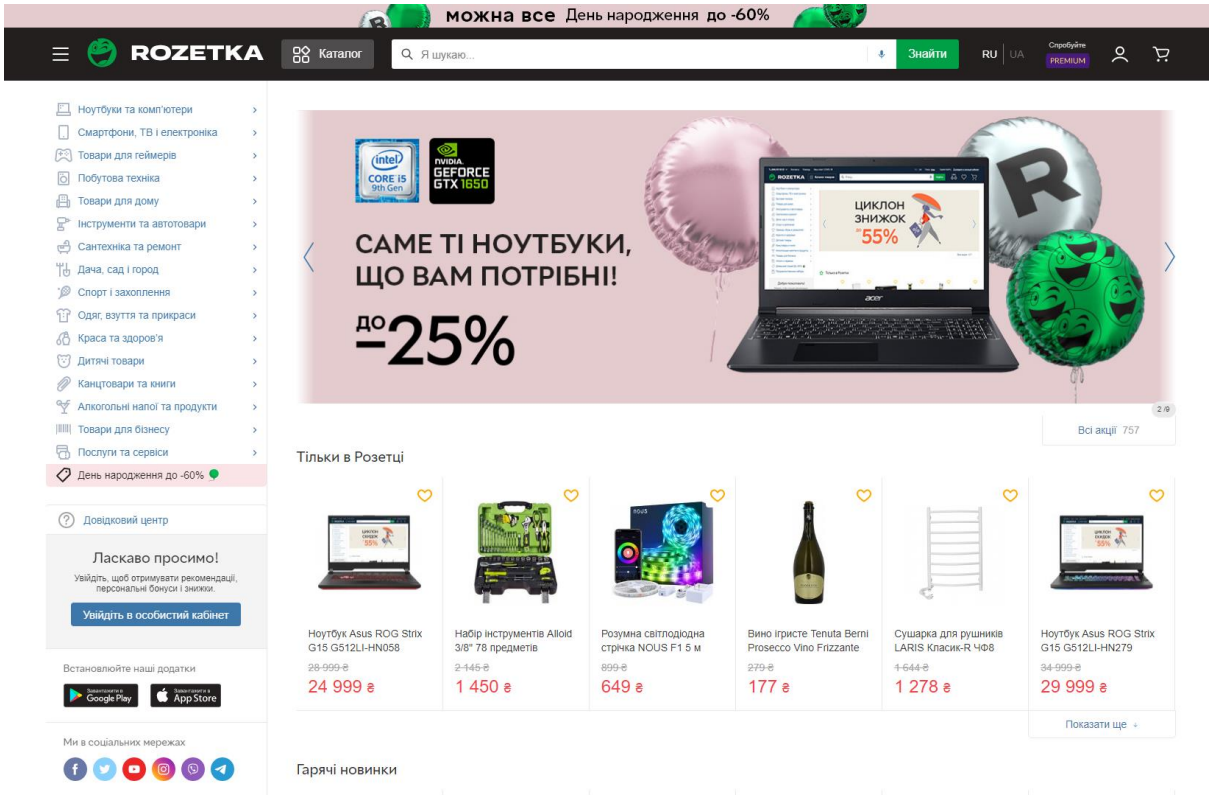

Рис. 2.1.2. Приклад інтернет-магазину

- промо-сайти – сайти, які передбачають рекламні функції для створення привабливості якогось продукту (рис. 2.1.3.). Промо-сайт - це сторінка, створена в інтернеті спеціально для просування товарів, брендів або послуг, і того, щоб підштовхнути покупця до вибору вашої фірми. Така промо-сторінка в більшій мірі являє собою буклет зі списком пропонованих фірмою товарів. Розробка промо сайту найчастіше ведеться на замовлення великих компаній, які мають широкий асортимент продукції або послуг. На таких сторінках також можуть висвітлюватися події, акції і заходи, що плануються компанією. Промо-сайт створюється, коли вже є готовий корпоративний сайт, це допомагає привернути увагу до бренду і збільшити чисельність продажів товарів. Промо-сторінку на вміщує в собі демо-ролики, точні описи товарів або послуг, численні

мультимедійні засоби, розроблені веб-дизайнерами для залучення уваги покупців до товару. Промосайт є більш привабливим для відвідувачів, ніж корпоративний сайт - він відрізняється менш офіційним дизайном, також на промо сторінках допускається якась гра слів при презентації послуги для того, щоб більш цікавим способом представити товар або послугу;

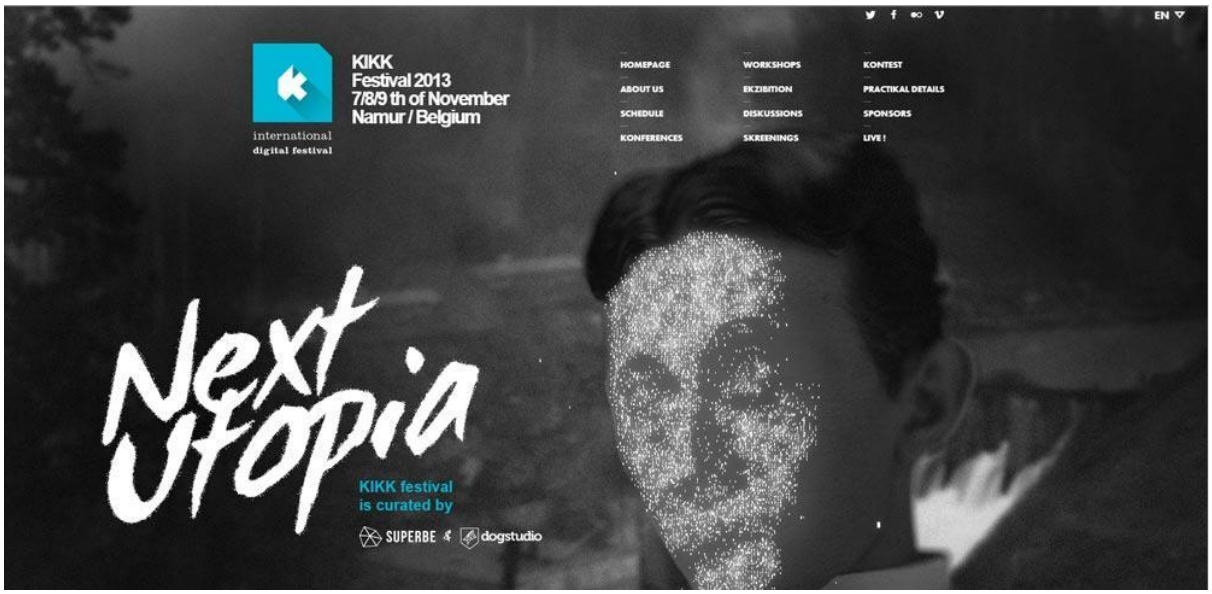

Рис. 2.1.3. Приклад промо-сайту

- сайти з платним контентом (рис. 2.1.4.). Платний контент зазвичай відноситься до одного з двох понять, що стосуються онлайн або цифрових медіа. Термін часто відноситься до контенту на веб-сайті, який повинен бути оплачений для доступу або використання відвідувачами. Це може вказувати на матеріали, до яких інакше не можна отримати доступ без оплати, або на контент, який можна переглянути, але не завантажити без оплати. Платний контент також може бути контентом, створеним для веб-сайту кимось, кому платять за ці матеріали. Точне значення «платного контенту» часто залежить від контексту, в якому він використовується, хоча в цілому воно зазвичай відноситься до контенту, знайденого на веб-сайті або інших цифрових носіях. Одним з найбільш поширених варіантів використання цього терміна є посилання на будь-який тип контенту, який можна знайти на веб-сайті та за який хтось повинен платити плату за

доступ. Зазвичай це трапляється на веб-сайтах, на яких стягується плата за підписку, хоча можуть використовуватися і інші способи оплати. Наприклад, хтось може мати вільний доступ до веб-сайту, на якому розміщені відеофайли, але може знадобитися оплатити окремі файли, які він або вона бажає завантажити або ліцензувати. Цей тип платного контенту можна знайти на декількох різних веб-сайтах, які часто дозволяють безкоштовно переглядати сайти до справляння плати за платний контент. Веб-сайт може дозволяти гостям переглядати зразки матеріалів, розміщених на цьому сайті, але стягувати плату за повні версії цих матеріалів. Аналогічним чином, деякі сайти можуть дозволяти гостям доступ до контенту сайту, але ті, хто платить, можуть переглядати ці матеріали без перегляду реклами. Деякі веб-сайти, на яких розміщені файли, можуть дозволяти гостям завантажувати ці файли, але вони доступні для завантаження без обмежень і обтяжень пропускної здатності тільки в якості платного контенту. Платний контент може також посилатися на контент, створений для веб-сайту як частина платного угоди. На деяких веб-сайтах розміщується контент, який створюється безкоштовно, як правило, адміністратором цього сайту або його учасниками. Для інших сайтів потрібно контент, створений іншими користувачами, який може бути створений для оплати власником або адміністратором сайту. Зазвичай це також називається платним контентом, і контекст, в якому використовується ця фраза, може розрізняти два значення терміна.

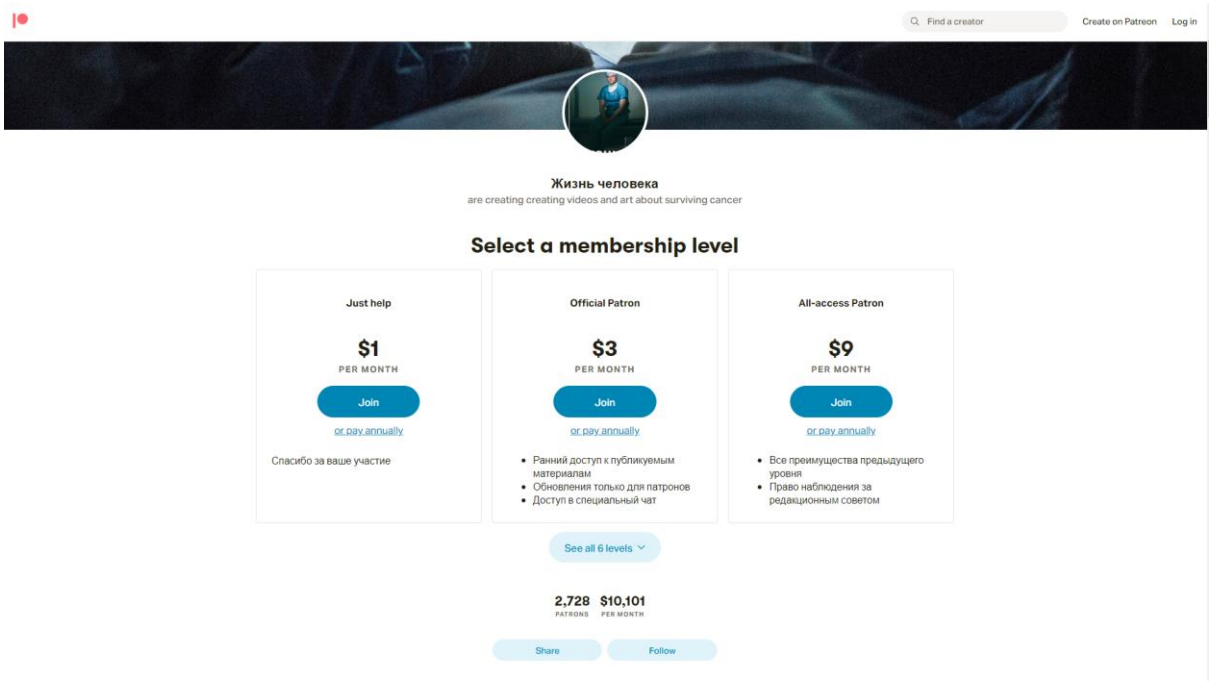

Рис. 2.1.4. Приклад сайта с платним контентом

Сайти з некомерційним змістом не передбачають продажних функцій. Вони бувають:

- новинними;
- інформаційними;
- розважальними.

Перш ніж приступити до процесу створення сайту, треба визначити проблему, яку він допоможе вирішити. Зокрема, для клієнта, що займається бізнесом, веб-ресурс – обов'язковий елемент у системі організації продажів. Тому необхідність розробки сайту обумовлена неефективністю діючого мережевого майданчика, або відсутністю віртуального представництва.

Також одним із найважніжих кроків створення сайту є визначення цілей та задач, які необхідні для бізнесу.

До цілей відноситься все, чого хоче досягти підприємство в майбутньому. Це мрії, очікування, надії, яким судилося або не судилося збутися через деяких

скоєних дій. Для того щоб аналізувати ступінь реалізації поставлених цілей, вони повинні бути вимірними, досяжними, прив'язаними до певного строку.

До завдань можна віднести конкретні кроки, які наближають реалізацію цілей. Завдання можна більш виразно спланувати. Крім того, вони спираються на об'єктивні критерії і наявні умови для їх втілення. Це задані розміри бюджету, особливості технічних вимог і т.д. Якщо завдання поставлені адекватно, то досягнення цілей можливо.

Після визначення цілей та постановки завдань можна переходити безпосередньо до створення сайту (web-додатку), цей процесс умовно можно розділити на 3 етапи :

- планування;
- дизайн;
- розробка.

Планування включає в себе :

- створення ідеї;
- розробку структури проекту;
- опрацювання макету проекта.

Створення ідеї. На цьому етапі необхідно визначитися з тематикою проекту (сайту, сервісу). Далі, відповідно за обраною темою, необхідно зібрати відповідні матеріали, текстові та графічні.

Розробка структури проекту. Коли визначилися з темою проекту, підібрали необхідний матеріал, наступним етапом є розробка структури проекту. Структура проекту має визначає розділи сайту, відповідно до яких буде формуватися навігаційне меню і будуватися дизайн проекту. На цьому етапі класифікувати матеріал за темами і розділами.

Опрацювання макету проекту. Після того, як визначилися зі структурою проекту, можна скласти макет проекту (схематично).

Для моделювання начерку часто використовують папір і ручку, Photoshop, будь-який інший редактор графіки, або програми для прототипування макетів сайтів та додатків. Важливо відзначити, що даний етап – не є відмальовуванням готового дизайн-макету, а всього лише схематичне уявлення, виконание для розуміння того, як на сайті будуть розташовуватися основні інформаційні блоки, графіка та інші елементи дизайну.

Основні елементи сторінки.

Найчастіше основними елементами сторінки є: блок вмісту (wrapper, container), логотип, навігація, контент, футер (нижній колонтитул), вільний простір (вільний простір - це не елемент дизайну, а поняття, пам'ятаючи про яке при складанні макета сторінки, наш проект не буде виглядати як нагромадження блоків). Далі треба вирішити, який тип дизайну необхідно обрати. Гумовий та фіксований макети. В фіксованих макетах використовується значення розмірів елементів в пікселях, в гумових же – в процентах.

Фіксований макет, має на увазі під собою, що в незалежності від дозволу екрану користувача ваш сайт завжди буде займати однакову ширину.

Гумовий макет має на увазі, що сторінка сайту буде намагатися зайняти весь доступний їй простір на екрані користувача, підлаштовуючись під дозвіл.

В даному контексті варто усвідомити такі поняття, як чуйний веб-дизайн (Responsive Web Design aka. RWD) і адаптивний веб-дизайн (Adaptive Web Design aka. AWD). Перше поняття вкладається в концепцію «гумового» і означає, що при зміні розміру екрану ваш сайт підлаштовується під нього, друге поняття має на увазі, що при розробці ви визначаєте основні дозволу (розміри екрану), під які буде підлаштовуватися (адаптуватися) ваш контент. В обох випадках слід розробляти не один, а кілька макетів, які будуть відповідати різним дозволам

екрану. Часто створюється 3 макета під дозволу Android Phone (iPhone), Android Tablet (iPad) і Desktop.

Модульна сітка.

Перед складанням схеми проекту так само необхідно усвідомити поняття модульної сітки. Модульна сітка має на увазі під собою поділ сторінки на окремі колонки по вертикалі і вибудовування контенту, при розробці дизайн макета, саме по цій сітці.

Найбільш популярною системою є модульна сітка 960 Grid System, яка максимально ділить сторінку на 12, 16 і 24 колонки. Максимум в ширині сітка має 960 пікселів. Дане рішення засноване на тому, що більшість сучасних моніторів, на момент створення сітки, мали дозвіл не менше 1024 на 768 пікселів. Створення макета на основі даної сітки, в подальшому, допоможе значне прискорити процес розробки (верстки).

Так само варто відзначити, що при розробці «гумового» макету сторінки існує поняття максимальної ширини. Дане твердження ґрунтується на зручність сприйняття інформації. Якщо припустити, що наш сайт не має максимального значення по ширині, то на великих моніторах інформація буде сильно розтягуватися і її незручно буде читати. Найчастіше обмежуються шириною в 1000-1280 пікселів.

Завдяки модульній сітці блоки контенту і елементи будуть розташовуватися на певній відстані один від одного та будуть мати деяку ширину, що в подальшому буде візуально приємно користувачеві та не буде викликати у нього будь-які незручності в сприйнятті сайту.

Отже, модульна сітка – це візуальна абстракція, візуальний розподіл сторінки на однакові по ширині стовпці з однаковими відступами між ними.

Макет веб-сторінки.

В нашому випадку використовується наступний макет (рис. 2.1.5.).

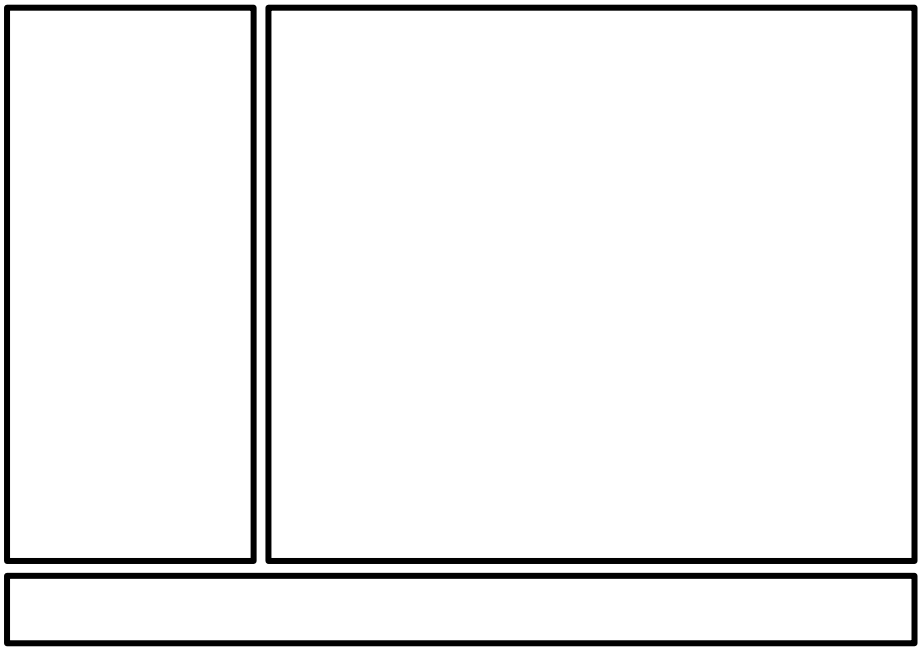

Рис. 2.1.5. Схематичний макет веб-сторінки

Після визначення типу та макету сайту переходимо до створення дизайну.

Дизайн.

На даному етапі розпочати варто з визначення колірної гами проекту. Вона була підібрана на основі ключових слів, за якими здійснюють пошук користувачі в мережі за тематикою роботи підприємства, а також візуального сприйняття.

Основними кольорами на сторінці виступають: білий, сіро-білий та темно-зелений (рис. 2.1.6.).

Рис. 2.1.6. Основні кольори сайту

Кольорова палітра підбиралась згідно принципу AIDA (Attention Interest Desire Action).

AIDA (УІБД) – дане поняття застосовується при дизайні головних сторінок, верхньої частини сайту, сторінок акцій, тощо. Де необхідно підштовхнути користувача до тієї або іншої дії: підписка, покупка та ін. Якщо перевести цей акронім на українську, то ми отримаємо такі поняття:

- **У**вага;
- **І**нтерес;
- **Б**ажання;
- **Д**ія.

Фреймворки.

Варто так само відзначити, що іноді, при розробці дизайну сторінки, використовують фреймворки Bootstrap, Foundation, Material Design Lite, які, крім готових елементів дизайну (кнопки, форми введення та ін.,). Пропонують свою модульну сітку, CSS сніппети (частина коду, розмітки, яка може неодноразово використовуватися) для вставки елементів в сторінку (тих же кнопок, елементів форм та ін.) і класи розмітки, JS скрипти для відповідних інтерактивних елементів.

Розробка.

Отже, процес дизайну макета сторінки плавно перетікає в процес «оживлення» зробленого на попередніх етапах. Перш ніж відразу починати писати HTML, CSS, JS та PHP варто трохи поговорити про редактори коду і структуру проекту.

Редактори коду.

З найбільш популярних редакторів коду на сьогодні можна виділити три:

- Sublime Text (http://www.sublimetext.com/3);

- Atom (https://atom.io/);
- Brackets (http://brackets.io/).

Від частини, всі ці редактори схожі за принципом роботи, коли при установці ми отримуємо редактор, в який потім можемо «доставити» необхідні модулі та плагіни. Різниця лише в технологіях, які були використані при написанні редакторів, якщо Sublime Text написаний за допомогою C ++ і Python, то 2 інших використовують JavaScript, HTML, CSS (Less). За рахунок цієї різниці Sublime Text може працювати трохи швидше своїх колег, а тому саме на нього ліг вибір для написання коду.

Структура проекту.

Під структурою проекту розуміється зберігання файлів проекту в його директорії. Часто доводиться бачити, коли всі файли «звалені» разом, назви файлів дані «КАПС», цифрами або українськими літерами тощо. По-перше, це неповага до того, хто буде працювати з проектом далі, а по-друге, чим більше буде проект, тим більше буде ставати файлів і врешті-решт, ви просто заплутаєтеся, що до чого відноситься і що потрібно, а що ні.

Найкраще окремі категорії файлів поміщати в свої папки: картинки в папку images або img, css файли в папку css, а javascript в папку js. У корені буде лежати тільки index.html і сторінки сайту, або тільки index.html, сторінки в окремій папці pages. Дотримуючись цих правил ніколи не заплутаєтесь в проекті.

Отже, визначившись з редактором коду, структурою ми можемо приступати до розробки. Перш за все варто відзначити, що верстка сторінки робиться поетапно: спочатку пишеться HTML-структура (HTML-код), потім додаються стилі, а після, якщо необхідно, пишуться скрипти (JS), PHP код та додаються необхідні плагіни і бібліотеки.

З огляду на вищесказане, ми можемо умовно розділити роботу над проектом на наступні етапи:

- написання HTML;
- написання CSS;
- написання JS;
- написання PHP.

2.2. Мова гіпертекстової розмітки HTML5

HTML можна назвати основною мовою Всесвітньої павутини. Більшість веб-сторінок, розміщених в Інтернеті, написані в будь-якій з варіацій HTML. За допомогою нього розробники визначають те, як мультимедіа, текст або гіперпосилання відображатимуться серед іншого контенту в браузері.

Починаючи від елементів, які встановлюють зв'язку з вашим документом (гіпертекстом), до елементів які роблять ці документи інтерактивними (наприклад форми) - все це є складовими частинами HTML.

Стандарт HTML був розроблений W3C або Консорціумом Всесвітньої Павутини в 1997 році. У HTML для визначення структури тексту використовуються теги; теги й елементи виділяються з використанням символів < і >. Лише деякі з прикладів для раніше згаданих тегів – це заголовки, таблиці, абзаци і т.д.

У свою чергу, браузери відповідають за візуалізацію вмісту сторінки за допомогою цих тегів. HTML не був єдиним стандартом веб-розробки. У перші дні розвитку Інтернету всі теги контенту і стилі присутні на одній гігантській, громіздкий (і досить складний) мові. Згодом W3C прийшла до рішення про необхідність поділу контента і стилю сторінки. Це призвело до створення таблиць стилів або CSS. В даний час теги, які використовуються для визначення стилю тексту (наприклад font), небажані і майже не використовуються, на їх місце прийшли таблиці стилів і тільки теги визначення вмісту (наприклад h1), як і раніше становлять ядро HTML.

З плином часу в HTML було багато оновлень, і в даний час його найновішою версією є HTML5. Він звичайно перш за все є мовою розмітки, але він отримав безліч функцій на відміну від HTML і усунув деякі з суворих обмежень, які були присутні в XHTML. Хоча HTML5 оновлюється практично щодня, однак нових випущених пронумерованих випусків немає. Основною відмінністю між HTML і HTML5 є те, що ні аудіо, ні відео не є складовою частиною HTML, тоді як вони обидва можуть розглядатися як невід'ємні частини HTML5.

Основні відмінності між HTML і HTML5.

Єдиною постійною річчю в області інформаційних технологій є те, що періодичні оновлення та зміни неминучі. Жодна мова не може уникнути оновлень або нових випусків. HTML не є винятком. HTML5 був випущений з метою поліпшення роботи Всесвітньої павутини, як для розробників, так і для звичайних користувачів. Як уже згадувалося, найбільшою перевагою, яке має HTML5 над своїм ненумерований попередником, є те, що у нього є підтримка аудіо і відео високого рівня, яка не була частиною специфікації в попередніх HTML. Інші відмінності між HTML і HTML5:

- SVG, canvas і інша віртуальна векторна графіка підтримуються в HTML5, тоді як в HTML використання векторної графіки стало можливим тільки при використанні його в сполученні з різними технологіями, такими як Flash, VML, Silver-light і т.д.
- HTML5 використовує бази даних SQL і кеш додатків для тимчасового зберігання даних, тоді як в HTML для цього використовується тільки кеш браузера.
- Ще одна відмінність між HTML і HTML5, про який варто згадати: перший не дозволяє запуск JavaScript в коді (замість цього він працює в потоці інтерфейсу браузера), тоді як останній забезпечує повну підтримку JavaScript для запуску у фоновому режимі.
- HTML5 не грунтується на SGML, і це дозволяє йому мати покращені правила синтаксичного аналізу, які забезпечують поліпшену сумісність.

- У HTML5, в тексті можуть використовуватися вбудовані MathML і SVG, тоді як це неможливо в HTML.
- Деякі з застарілих елементів були повністю видалені: isindex, noframes, acronym, applet, basefont, dir, font, frame, frameset, big, center, strike, tt.
- HTML5 підтримує нові види елементів управління, наприклад, dates and times, email, number, range, tel, url, search і т.д.
- В HTML5 з'явилося багато нових елементів. Ось деякі з найбільш важливих: summary, time, aside, audio, command, data, datalist, details, embed, wbr, figcaption, figure, footer, header, article, hgroup, bdi, canvas, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, track, video.

2.3. Мова сценаріїв JavaScript

Синтаксис мови JavaScript багато в чому нагадує синтаксис Сі та 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):
- 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 - 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, на відміну від DOM, що вже підтримується всіма браузерами,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 браузери,
- 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 ними різні маніпуляції:

- отримання вузлів;
- зміна вузлів;
- зміна зв'язків між вузлами;
- видалення вузлів.

Розташування всередині сторінки. Щоб додати JavaScript-код на сторінку, використовують HTML теги <script></ script>.

Скрипт, що виводить модальне вікно з класичним написом «Тест» у середині браузера:

<script type="text/javascript">

alert('Тест');

</script>

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

<a href="delete.php" onclick="return confirm('Ви впевнені?');"> Видалити </ a>

Відділення від розмітки. У наведеному прикладі при натисканні 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.

Використання коду JavaScript в контексті розмітки сторінки в рамках ненав'язливого JavaScript розцінюється як погана практика. Аналогом (за умови постачання посилання ідентифікатором alertLink)

Видалити

Повний код наведеного прикладу:

window.onload = function() {

var linkWithAlert = document.getElementById("alertLink");

linkWithAlert.onclick = function() {

return confirm('Ви впевнені?');

};

};

Можливість винесення коду 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 ()).

Атрибут language (language = "JavaScript"), незважаючи на його активне використання, відноситься до не рекомендованих (deprecated), відсутній в DTD, тому вважається некоректним.

2.4. Каскадні таблиці стилів CSS

Технологія опису зовнішнього вигляду документа, написаного мовою розмітки.

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 у налаштуваннях браузера, перевизначають авторські стилі, і вживається до всіх документах.

Стилі складаються зі списку правил. Кожне правило, у свою чергу, складається з одного або більше селекторів, розділених комами, і блоку визначень. Блок визначення оточеного фігурними дужками, і складається з набору властивостей і їх значень. Схематично це можна показати так:

селектор, селектор {

властивість: значення;

властивість: значення;

властивість: значення; }

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 !important;
2. Стиль прописаний прямо в тегу;
3. Кількість ідентифікаторів (#id) в селекторі (чим більше, тим більше пріоритет);
4. Кількість класів (.class) в селекторі;
5. Кількість імен тегів в селекторі.

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

Приклад таблиці стилів:

p {

font-family: "Garamond", serif;

}

h2 {

font-size: 110 %;

color: red;

background-color: white;

}

.note {

color: red;

background: yellow;

font-weight: bold;

}

p#paragraph1 {

margin: 0;

}

a:hover {

text-decoration: none;

}

#news p {

color: blue;

}

Тут наведено шість правил, селектори *p, h2,. Note, p#paragraph1, a: hover і # news p.*

У перших двох правилах визначаються властивості HTML-елементів p (абзацу) і h2 (Заголовок другого рівня). Абзаци буде відображено шрифтом Garamond, або, якщо такий шрифт недоступний, яким-небудь іншим шрифтом sans («serif»). Заголовок другого рівня буде відображено червоним кольором на білому фоні з збільшеним кеглем.

Третє правило буде застосовано до всіх елементів, атрибут class визначений як 'note'. Наприклад:

* Цей абзац буде відображено червоним жирним шрифтом на жовтому фоні. *

Четверте правило буде застосовуватися тільки до елементів p, атрибут id яких дорівнює paragraph1. Такі елементи не будуть мати зовнішніх відступів (margin).

Шосте правило визначає стиль hover для елементів a. За умовчанням в більшості браузерів текст елементів a підкреслюється. Це правило прибере підкреслювання, коли курсор миші знаходиться над цими елементами.

Останнє правило застосовується до тих p, які знаходяться всередині елемента з атрибутом id, рівним «news».

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 різних 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 смартфоні 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 закешовані;
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:

1. Різного відображення верстки в різних браузерах (особливо застарілих), які по різному інтерпретують одні й ті ж дані CSS;
2. Часто зустрічається необхідність на практиці виправляти не тільки один CSS-файл, але і теги HTML і код PHP, які складним і не наглядним 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-фреймворку, 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 .

Історія. CSS – одна з широкого спектру технологій, схвалених консорціумом W3C, і які одержали загальну назву «стандарти Web». У 1990-х роках стала ясна необхідність стандартизувати Web, створити якісь єдині правила, за якими програмісти і веб-дизайнери проектували б сайти. Так з'явилися мови HTML 4.01 і XHTML і стандарт CSS.

На початку 1990-х різні браузери мали свої стилі для відображення веб-сторінки. HTML розвивався дуже швидко і був здатний задовольнити всі існуючі

на той момент потреби з оформлення інформації, тому CSS не отримав тоді широкого визнання.

На відміну від багатьох існуючих на той момент мов стилю, CSS використовує успадкування «від батька до нащадку», тому розробник може визначити різні стилі, ґрунтуючись на вже визначених раніше стилях.

У середині 1990-х Консорціум Всесвітньої павутини (W3C) став виявляти цікавість до CSS, і в грудні 1996 року була видана рекомендація CSS1

Рівень 1 (CSS1). Рекомендація W3C, прийнята 17 грудня 1996 року, відкоректована 11 січня 1999. Серед можливостей, що надаються цією рекомендацією:

- параметри шрифтів. Можливості за завданням гарнітури і розміру шрифту, а також його стилю - звичайного, курсивного або напівжирного;
- кольори. Специфікація дозволяє визначати кольори тексту, фону, рамок та інших елементів сторінки;
- атрибути тексту. Можливість задавати міжсимвольний інтервал, відстань між словами і висоту рядка (тобто міжрядкові відступи);
- вирівнювання для тексту, зображень, таблиць та інших елементів;
- властивості блоків, такі як висота, ширина, внутрішні (padding) і зовнішні (margin) відступи і рамки. Так само в специфікацію входили обмежені кошти з позиціонування елементів, такі як float і clear.

Рівень 2 (CSS2). Рекомендація W3C, прийнята 12 травня 1998. Побудована на CSS1 зі збереженням зворотної сумісності. Додавання до функціональності:

- блочне верстання. З'явилися відносне, абсолютне та фіксоване позиціонування. Дозволяє управляти розміщенням елементів по сторінці без табличної верстки;
- типи носіїв. Дозволяє встановлювати різні стилі для різних носіїв (наприклад монітор, принтер, КПК);

- звукові таблиці стилів. Визначає голос, гучність і іншим чином для досягнення більшої гнучкості (див. для сліпих відвідувачів сайту);
- сторінкові носії. Дозволяє, наприклад, встановити різні стилі для елементів на парних і непарних сторінках під час друку;
- розширений механізм селектору;
- покажчики;
- генерується зміст. Дозволяє встановити текст або картинку, який буде відображатися до або після потрібного елемента.

Рівень 2.1 (CSS2.1). Робоча версія W3C від 8 вересня 2009 року. Побудована на CSS2, містить виправлення помилок.

Рівень 3 (CSS3). Сильно розширена в порівнянні з попередніми версіями. Нововведення, починаючи з малих, на кшталт заокруглених кутів блоків, закінчуючи трансформацією (анімацією) і, можливо, введенням змінних. На даний час поки що не стандартизований і підтримується невеликою кількістю браузерів і виключно новими їх версіями.

Включення до HTML. CSS можна включати в HTML чотирма способами:

1. Посилання в HTML, а самі CSS в окремому файлі:

   <link rel="stylesheet" type="text/css" href="style.css" />;

2. Ще один спосіб підключити CSS, що знаходяться в окремому файлі:

   <style type="text/css" media="all"> @ import "style.css"; </ style>;

3. Безпосередньо в HTML-документі:

   <style type="text/css">
   body (
   color: red;
)

</style>;

4. Безпосередньо в елемент:

<p style="font-size: 21px; color: green;">Розповідь про те, як шкідливо довго дивитись телевізор </ p> XML. Для включення CSS в XML існує ще один спосіб:

<?xml-stylesheet href = "style.css" type = "text / css"?.

2.5. Мова програмування PHP

Мова PHP, завдяки своїм особливостям широко використовується при розробці динамічних веб-сайтів. До основних властивостей мови можна віднести:

- автоматичне вилучення POST і GET-параметрів, а також змінних оточення веб-серверу в зумовлені масиви;
- файлові функції успішно обробляють як локальні, так і віддалені файли;
- автоматична відправка HTTP-заголовків;
- робота з cookies і сесіями;
- обробка файлів, що завантажуються на сервер;
- робота з HTTP заголовками і HTTP авторизацією;
- робота з віддаленими файлами.

Принцип роботи PHP полягає в наступному:програмний код інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта. На відміну від таких скриптових мов програмування, як JavaScript, користувач не має доступу до PHP-коду, що є перевагою з точки зору безпеки але значно погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів,які виконуються вже на стороні клієнта.

PHP – мова, яка може бути вбудована безпосередньо в html–код сторінок, які, в свою чергу коректно будуть оброблені PHP–інтерпретатором. Механізм PHP просто починає виконувати код після першої екрануючої

послідовності (<?) і продовжує виконання до того моменту, коли він зустріне парну екрануючу послідовність (?>).

Синтаксис PHP подібний синтаксису мови Сі. Для роботи програми не потрібно описувати будь-які змінні, які використовуються модулі і т. п. Будь-яка програма може починатися безпосередньо з оператора PHP.

Типова програма на PHP містить наступні складові частини:

```
<?php
echo 'Тест!';
?>
```

PHP виконує код, що знаходиться всередині обмежувачів, таких як <? php?>. Все, що знаходиться поза обмежувачів, виводиться без змін. В основному це використовується для вставки PHP-коду в HTML-документ, наприклад, так:

```
<html>
<head>
<title>Тест PHP</title>
</head>
<body>
<?php echo 'Hello, world!'; ?>
</body>
</html>
```

Імена змінних починаються з символу $, тип змінної оголошувати не потрібно. На відміну від імен функцій і класів, імена змінних чутливі до регістру. Змінні обробляються в рядках, укладених в апострофи або подвійні лапки, і heredoc-рядках (рядках, створених за допомогою оператора <<<).

Імена змінних відповідають тим же правилам, що й інші найменування в PHP. Правильне ім'я змінної має починатися з букви або символу підкреслення з подальшими в будь-якій кількості літерами, цифрами або символами підкреслення. Це можна відобразити регулярним виразом: *'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'*

Можливо використання символів: a-z, A-Z, и ASCII-символів з 127 по 255 (0x7f-0xff).

```
<?php
$var = "Інтернет_";
$Var = "Сайт";
echo "$var, $Var"; // виводить " Інтернет, Сайт "?>
```

PHP розглядає перехід на новий рядок як пробіл, так само як HTML та інші мови з вільним форматом. Інструкції розділяються за допомогою крапки з комою (;), за винятком деяких випадків.

PHP підтримує три типи коментарів: у стилі мови Сі (обмежені / * * /), C (що починаються з / / і що йдуть до кінця рядки) та оболонки UNIX (з # до кінця рядка).

PHP є мовою програмування з динамічною типізацією, що не вимагає вказівки типу при оголошенні змінних, так само як і самого оголошення змінних. Перетворення між скалярними типами часто здійснюються неявно без додаткових зусиль (втім, PHP надає широкі можливості і для явного перетворення типів). Скалярні типи даних:

- цілий тип (integer);
- речовинний тип даних (float, double);
- логічний тип (boolean);
- строковий тип (string);
- спеціальний тип NULL.

І нескалярні:

- «ресурс» (resource);
- масив (array);
- об'єкт (object);
- анонімна функція (closure) або псевдотип callback.

Діапазон цілих чисел (integer) в PHP Залежить від платформи (зазвичай це діапазон 32-бітних знакових цілих чисел, тобто від -2147483648 до 2147483647). Числа можна задавати у десятковій, вісімковій і шістнадцятковій системах числення. Діапазон дійсних чисел (double) також залежить від платформи (для 32-бітної архітектури діапазон дозволяє оперувати числами від ± 1.7 × 10-308 до ± 1.7 × 10 308).

PHP надає розробнику і логічний тип (boolean), що здатен приймати тільки два значення TRUE («Істина») і FALSE («Брехня»). При перетворенні в логічний тип число 0, порожній рядок, нуль в рядку «0», NULL і порожній масив вважаються FALSE. Всі інші значення автоматично перетворюються у TRUE.

Посилання на зовнішні ресурси мають тип «ресурс» (resource). Змінні даного типу, як правило, представляють собою дескриптор, що дозволяє управляти зовнішніми об'єктами, такими як файли, динамічні зображення, результуючі таблиці бази даних тощо.

Масиви (array) підтримують числові і рядкові ключі і є гетерогенними. Масиви можуть містити значення будь-яких типів, включаючи інші масиви. Порядок елементів і їх ключів зберігається. Не зовсім коректно називати php-масиви масивами, насправді це швидше упорядкований хеш.

Покажчик на функцію в PHP може бути представлений замиканням або типом callback. Замикання доступне з версії 5.3 та в коді виглядає як просте визначення функції, в яку явно можна тягнути значення з контексту. Приклад: *function ($ args .. $ argsN) use ($ ctxVar, $ ctxVar1) (definition;).* callback тип може бути представлений: рядком (інтерпретується як назва функції); масивом де нульовий і перший елемент рядка (інтерпретується як назва статичної функції в

класі); масивом де нульовий елемент об'єкт, а перший рядок (інтерпретується як метод у об'єкту). Для перевірки чи є значення викликуваним слід використовувати is_callable ($ var)

Суперглобальними масивами (англ. Superglobal arrays) в PHP називаються зумовлені масиви, що мають глобальну область видимості без використання директиви global. Велика частина цих масивів містить вхідні дані запиту користувача (параметри GET-запиту, поля форм при посилці методом POST, куки і т. п.).$GLOBALS.

За допомогою мови PHP на сайті була реалізована функція відправки веб-форми користувача на електронну пошту підприємства для подальшої обробки.

2.6. Робота з Bootstrap

Використання даного інструментарію підходить для веб-розробників, які володіють основами HTML, CSS та jQuery.

Завантаження і установка Bootstrap

Завантажити і використовувати Bootstrap можна декількома способами. Для початку, можна скористатися npm. Тут знадобиться така команда:

npm install bootstrap

Bootstrap можна підключити до сторінки з використанням мережі доставки контенту. Для цього треба додати наступне посилання в тег <head>:

<link_rel="stylesheet"_href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"_integrity="sha384n5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm">

Далі потрібно завантажити Bootstrap з його офіційного сайту - https://getbootstrap.com/.

Система сіток Bootstrap

Система (Bootstrap Grid System) призначена для створення макетів сторінок. Вона спрощує розробку чуйних веб-сайтів. Сітка розділена на 12

колонок, ця структура, налаштована так, як потрібно розробнику, і вона є основою макета сторінки.

Для того щоб використовувати Bootstrap-сітку, потрібно додати клас .row до головного елементу сторінки. Під час налаштування розмірів вкладених елементів використовують такі класи (замість зірочки в кінці імені класу вказується число стовпців базової 12-колоночной сітки, яке повинен займати конкретний елемент) :

1. col-lg- * - клас, який використовується для сторінок, призначених для пристроїв з великим екраном на зразок ноутбуків;

2. col-md- * - клас для сторінок, розрахованих на пристрої з екраном середнього розміру, таких, як планшети;

3. col-sm- * - клас для сторінок, які розраховані маленькі екрани, наприклад, такі, як у смартфонів.

Мультимедійні запити

Використання таких медіа-запитів в наших файлах Sass для створення ключових точок розриву в нашій сітці :

*/ * Дуже маленькі пристрої (телефони, менш ніж 768px) * /*

*/ * Невеликий пристроях (планшети, 768px і вище) * /*

@media (min-width: @ screen-sm-min) {...}

*/ * Середні пристроїв (настільних комп'ютерів, 992px і вище) * /*

@media (min-width: @ screen-md-min) {...}

*/ * Великі пристроїв (великі комп'ютери, 1200px і вище) * /*

@media (min-width: @ screen-lg-min) {...}

Іноді треба розширювати ці медіа-запити, що включають max-width для обмеження CSS до більш вузького набору пристроїв.

@media (max-width: @ screen-xs-max) {...}

@media (min-width: @ screen-sm-min) and (max-width: @ screen-sm-max) {...}

@media (min-width: @ screen-md-min) and (max-width: @ screen-md-max) {...}

@media (min-width: @ screen-lg-min) {...}

Для будування розмітки для мобільних пристроїв та настільних комп'ютерів використовуються класи розмітки :

.col-xs- .col-md-**

Приклад :

<! - Стовпці (col) для мобільного ->

* *

* . col-xs-12 .col-md-8 </ div>*

* . col-xs-6 .col-md-4 </ div>*

</ div>

<! - Стовпці (col) починаються з 50% шириною на мобільних пристроях і збільшуються до 33,3% на настільних пристроях ->

<! - Стовпці (col) завжди 50% в ширину, на мобільних і настільних комп'ютерах ->

* *

* . col-xs-6 .col-md-4 </ div>*

* . col-xs-6 .col-md-4 </ div>*

* . col-xs-6 .col-md-4 </ div>*

</ div>

<! - Колонки завжди 50% в ширину, на мобільний і робочий стіл ->

* *

* . col-xs-6 </ div>*

* . col-xs-6 </ div>*

</ div>

Еластичні блоки

Перетворення будь-якої схеми сітки фіксованої ширини в макет повної ширини, можна змінивши зовнішній вигляд .container на .container-fluid.

* *

* ... *

* *

Виходячи с вище вказаних параметрів фреймворку Bootstrap була виконана «Гумова» верстка сайту.

2.7. Опис інтерфейсу користувача

Інтерфейс сайту – це його «лице», перший етап взаємодії з користувачем. Цей етап дуже важливий, тому що саме на ньому користувач сайту (потенційний клієнт або партнер) вирішує, буде він користуватись товарами чи послугами, чи піде назавжди. Розробляючи інтерфейс необхідно враховувати, що їм можуть користуватися люди, які погано розбираються у технологіях і пошуку інформації, тому розробка інтерфейсу повинна вестись з дотриманням наступних правил:

1. Інтерфейс повинен бути зручним. Завдяки логічній побудові структури сайту визначається його зручність;

2. Інтерфейс повинен бути виконаний в спокійних, ненав'язливих кольорах, але водночас зазивати користувача до певних дії;

3. Інтерфейс повинен бути зрозумілим. Необхідно подавати інформацію на сайті таким чином щоб її міг зрозуміти будь-хто. У цьому допоможуть правильно

розроблена система акцентів (кольорові блоки з інформацією, великі кнопки тощо);

4. Інтерфейс має бути цікавим.Для цього необхідно наповнити сайт цікавою унікальною інформацією, придумати чим утримати увагу користувача. Інтерфейс сайту повинен бути розрахований:

A. На різні розділювачі здатності монітору, як мінімум 1024x768 пікселів;
B. На різні версії браузерів, в тому числі дуже застарілих, наприклад InternetExplorer 6.

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

Інтерфейс сайту формується з трьох частин:

- шаблонів файлів (templates);
- налаштувань каскадних таблиць стилів CSS, які зберігаються у файлі style.css;
- файлів зображень, які прописані у style.css і напряму у файлах шаблонів чи модулів.

При такій організації інтерфейс сайту зберігається в одному місці і не дублюється, що робить код простіше. Параметри таблиць та шарів описанні у файлах шаблонів та у style.css.

Звернення до файлів стилів ми будемо використовувати на всіх сторінках сайту, шаблонах та модулях. CSS параметри вбудовуються в HTML код:

<link rel="stylesheet" href="css/style.css">

Враховуючи вимоги до інтерфейсу, була розроблена зручна структура сайту.

Зазвичай, перше, що бачить користувач заходячи на веб-сторінку, це головну сторінку, або верхню частину лендінгу. Враховуючи ці дані, користувачеві (потенційному клієнту або парнеру) важливо в перші секунди перебування на сторінці дізнатись, чим займається підприємство чи компанія. Але ми маємо інший випадок і перше, що бачить користувач, це форма для реєстрації (2.7.1.).

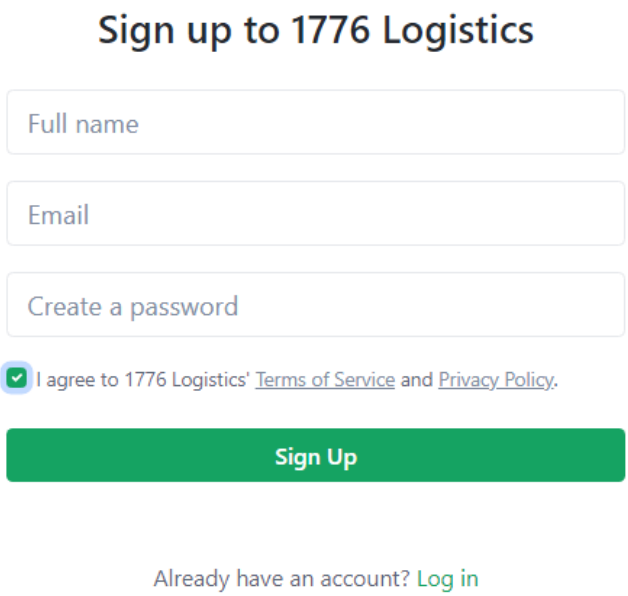

Рис. 2.7.1. Реєстрація

Якщо користувач вже має аккаунт, то він переходить на сторінку верифікації (2.7.2.).

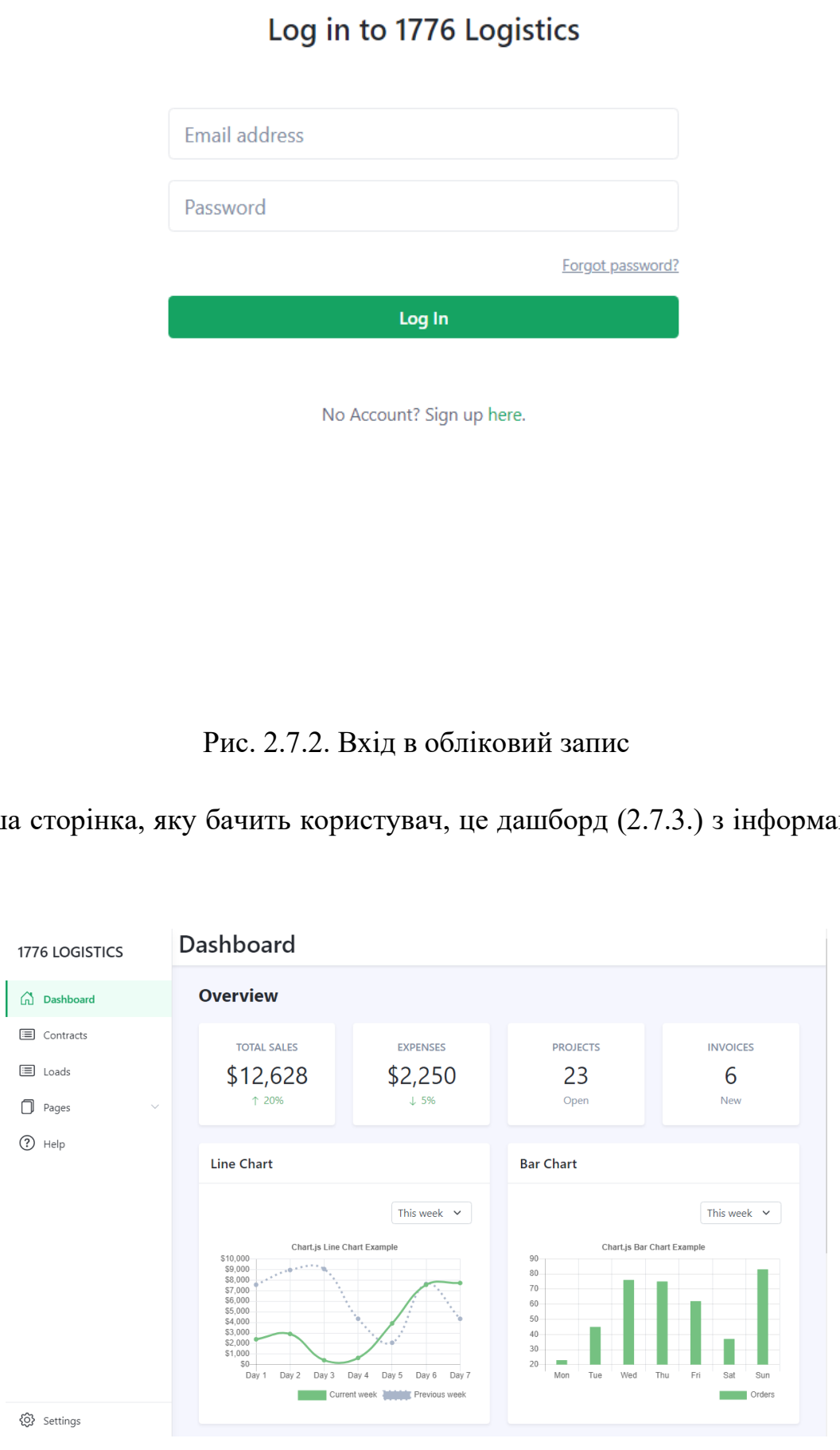

Рис. 2.7.2. Вхід в обліковий запис

Перша сторінка, яку бачить користувач, це дашборд (2.7.3.) з інформацією компанії.

Рис. 2.7.3. Дашборд

Звісно, користувач хоче побачити свої погрузки. Це він зможе зробити на цій сторінці (2.7.4.).

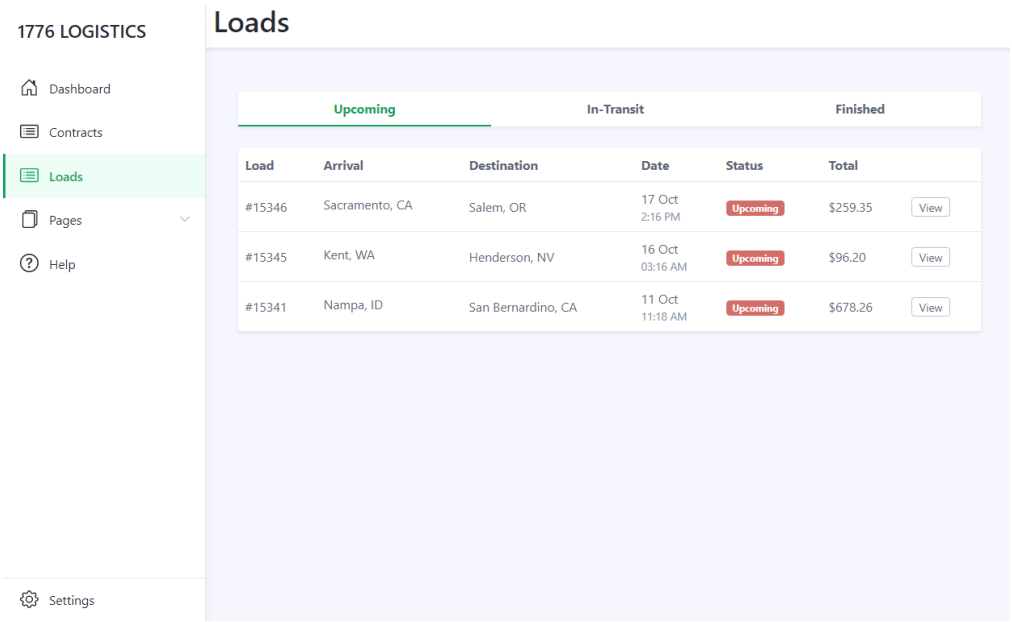

Рис. 2.7.4. Погрузки

Сайт не був розміщений на хостингу, так як ще не був перевірений главою компанії.

Умови надання послуг хостингу бувають безкоштовні (або умовно-безкоштовні) і платні. Першим варіантом користуються особи, що недавно почали свою діяльність в Інтернеті, причому переважно, як хобі. Цей варіант істотно підрізає права користувача, а також нав'язує демонстрацію реклами самого власника хостингу. У зв'язку з цим проекти, орієнтовані на довгострокові ділові відносини, повинні базуватися на платній платформі.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

1. Передбачуване число операторів – 850;
2. Коефіцієнт складності програми – 1,7;
3. Коефіцієнт корекції програми в ході її розробки – 0,08;
4. Часова заробітна плата програміста, грн/г – 80,0;
5. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1.5;
6. Коефіцієнт кваліфікації програміста, який визначається залежно від стажу роботи за даною спеціальністю – 1.0;
7. Вартість машино-години ЕОМ – 1.0.

У процесі створення ПЗ нормування праці ускладнено в силу творчого характеру праці програміста. Тому, трудомісткість розробки ПЗ розраховується на основі системи моделей з різною точністю оцінки.

$$t=tu+ta+tn+t\text{отл}+t\text{д}, \text{ чол.-г.} \quad (3.1)$$

де tu – затрати праці на дослідження алгоритму розв'язання задачі, чол.-г;

ta – затрати праці на розробку блок-схеми алгоритму, чол.-г;

tn – затрати праці на програмування по готовій блок–схемі, чол.-г;

tотл – затрати праці на налагодження програми на ЕОМ, чол.-г;

tд – затрати праці на підготовку документації по завданню, чол.-г.

Ці затрати праці визначаються через умовне число операторів при розробці ПЗ, в число яких входять ті оператори, які необхідно написати в процесі роботи

над програмою з урахуванням можливих уточнень у постановці завдання і вдосконалення алгоритму.

Умовне число операторів в програмі обчислюється за формулою:

$$Q = qC(1 + p), \qquad (3.2)$$

де q – передбачуване число операторів q = 850;

с – коефіцієнт складності програми с = 1,7;

р – коефіцієнт кореляції програми в ході її розробки р = 0,08.

$$Q = 850 * 1,7 * (1 + 0,08) = 1\,560,6$$

Затрати праці на вивчення опису завдання ti визначається з урахуванням уточнення опису та кваліфікації програміста.

$$tu = QB \div (75..85)K = (1560.6 * 1.4)/(78 * 1.0) = 28.01 \text{чол.-г.}, \quad (3.3)$$

де В – коефіцієнт збільшення затрат праці внаслідок недостатнього опису завдання:

$$B = 1,2...1,5;$$

К – коефіцієнт кваліфікації програміста, який визначається залежно від стажу роботи за даною спеціальністю. Він становить при стажі роботи, роки:

- до 2 – 0,8;
- від 2 до 3 – 1,0;
- від 3 до 5 – 1,1...1,2;
- від 5 до 7 – 1,3...1,4;
- вище 7 – 1,5...1,6.

Затрати праці на розробку алгоритма для рішення задачі:

$$ta = Q(20..25)K = 1560.6/(23 * 1.4) = 67.85 \text{чол.-г.}, \qquad (3.4)$$

Витрати на складання програми по готовій блок-схемі:

$$tn = Q(20..25)K = 1560.6/(23 * 1.4) = 67.85\text{чол.-г.}, \quad (3.5)$$

Затрати праці на налагодження програми на ЕОМ:

$$t\text{отл} = Q/(4..5)K = 1560.6/(4 * 1.0) = 390.15\text{чол.-г.}, \quad (3.6)$$

$$t\text{отл}^K = 1.5t\text{отл} = 1.5 * 390.15 = 585.22 \text{ чол.-г.}, \quad (3.7)$$

Витрати на підготовку документації:

$$t\text{д} = t\text{др} + t\text{до}\text{чол.-г.}, \quad (3.8)$$

де tдр – трудомісткість підготовки матеріалів:

$$t\text{др} = Q/(15..20)K = 1560.6/(17 * 1.0) = 91.8\text{чол.-г.} \quad (3.9)$$

tдо – трудомісткість редагування, друку та оформлення документації:

$$t\text{до} = 0.75t\text{др} = 0.75 * 91.8 = 68.85\text{чол.-г.} \quad (3.10)$$

$$t\text{д} = t\text{др} + t\text{до} = 91.8 + 68.85\text{чол.-г.} \quad (3.11)$$

У підсумку отримуємо, що трудомісткість розробки ПЗ становить:

$$t = 28.01 + 67.85 + 67.85 + 390.15 + 160.65 = 714.51\text{чол.-г.} \quad (3.12)$$

3.2. Рахунок витрат на створення програмного забезпечення

Витрати на створення ПО (Кпо) включають витрати на заробітну плату виконавців програми (з/п), визначену множенням сумарної трудомісткості розробки ПО (t) на середню зарплату з нарахуваннями програміста і вартості машинного часу, необхідного для відладки програми на ЕОМ (Змв), визначеною виходячи з вартості 1-го машинного години конкретного типу ЕОМ, і витрат машинного часу на налагодження.

$$K_{\text{ПО}} = З_{\text{ЗП}} + З_{\text{МВ}}, \text{ грн.} \quad (3.13)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * С_{ПР} = 714.51 * 80.0 = 57160.8 грн., \quad (3.14)$$

де t – загальна трудомісткість, чол.-г.;

Спр – середня годинна заробітна плата програміста, грн./год;

Спр =40,0 грн./год.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МЧ} = t_{отл} * С_{МЧ} = 390.15 * 10 = 3901.5 грн., \quad (3.15)$$

де tотл – трудомісткість налагодження програми на ЕОМ, год.;

Смч – вартість машинного часу ЕОМ, грн./год.

$$К_{ПО} = 57160.8 + 3901.5 = 61062.3 \text{ грн.} \quad (3.16)$$

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУТП. Очікуваний період розробки ПЗ:

$$T = t/B_k * F_p міс., \quad (3.17)$$

де Bk – кількість виконавців;

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

$$T = 714.51/1 * 176 = 4.06$$

Таким чином, період розробки програми складе приблизно 4 місяці.

Висновок: Визначено трудомісткість розробленої інформаційної системи (714,51 людино-годин), проведений підрахунок вартості роботи по створенню програми (61062,3 грн) та розраховано час на його створення (4 місяці). Цей

термін пов’язаний з значним числом операторів і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування на готовому алгоритмі, налагодження додатку і підготовка документації.

ВИСНОВОК

В дипломному проекті було розглянуто проблему розробки web-додатку для підприємства за допомогою HTML5, JavaScript та PHP мов програмування, а також необхідність і актуальність створення даного програмного продукту.

В роботі зроблено огляд сучасних технології створення сайтів та розглянуті різні системи керування контентом.

Результатом дипломного проекту є працюючий web-сайт, який включає у себе HTML5 та CSS, а також працює з POST запитами користувачів (PHP) та використанням технологій JavaScript.

В процесі розробки сайту було визначено основні вимоги, з якими повинен бути узгоджений сайт та його вміст. Були визначені особливості та алгоритм обирання теми для створення дизайну сайту згідно із типом підприємства та його спеціалізацією. Був сконструйований сайт, в якому було проведене налаштування зовнішнього вигляду, функціональних можливостей та вмісту.

Дипломна проект має практичне застосування, оскільки сайт був розроблений на замовлення підприємства «1776 Logistics». На даний момент сайт проходить перевірку власником компанії.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (714.51 люд.-г.), проведений підрахунок вартості роботи по створенню програми (61062.3 грн) та розраховано час на його створення (4 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Необхідні технології створення сайтів, - https://cetera.ru/about/articles/creating-a-website-html-css-js/

2. Переваги та недоліки конструкторів сайтів, - https://seosreda.com.ua/plyusyi-i-minusyi-konstruktorov-saytov/ 3. CSS-framework, - https://en.wikipedia.org/wiki/CSS_framework

4. Плюси та мінуси систем керування контентом Joomla, WordPress, Drupal, DLE 5. https://iq-project.ru/info/pros-and-cons-of-cms 6. Кращі фреймворки для використання, - https://jonathanmh.com/best-css-frameworks-sass/

7. Сайти створені за допомгою фреймворку Bootstrap, - https://www.zina.design/bootstrap/sites/

8. Основні відмінності HTML та HTML5, - https://www.hostinger.com.ua/rukovodstva/chto-takoe-html-i-ih-razlichiya 9. http://htmlbook.ru 10. http://ru.wikipedia.org/wiki/JavaScript 11. http://www.wordpress.com/about 12. http://javascript.ru/tutorial/dom

13. Марк Декстер, Луис Лэндри. Joomla! Programming: Издательство: Вильямс, 2013 г.

14. 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.

15. Вагнер Билл. C# Эффективноепрограммирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.

16. Зиборов В.В. Visual C# 2012 на примерах / В.В. Зиборов. - М.: БХВ-Петербург, 2013. - 480 с.

17. Ишкова Э.А. Самоучитель C#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 с.

18. Касаткин А.И. Профессиональноепрограммирование на языкеси. Управление ресурсами / А.И. Касаткин. - М.: Высшая школа, 2012. - 432 с.

19. Лотка Рокфорд. C# и CSLA .NET Framework. Разработкабизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.

20. Мак-Дональд Мэтью. Silverlight 5 с примерами на C# для профессионалов / Мэтью Мак-Дональд. - М.: Вильямс, 2013. - 848 с.

21. Марченко А.Л. Основыпрограммирования на C# 2.0 / А.Л. Марченко. - М.: Интернет-университетинформационныхтехнологий, Бином. Лаборатория знаний, 2011. - 552 с.

22. Подбельский В.В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.

23. ПрайсДжейсон. Visual C# 2.0. Полноеруководство / ДжейсонПрайс , Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, 2010. - 736 с.

24. РихтерДжеффри. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / ДжеффриРихтер. - М.: Питер, 2013. - 928 с.

25. ТроелсенЭндрю. Язык программирования C# 5.0 и платформа .NET 4.5 / ЭндрюТроелсен. - М.: Вильямс, 2015. - 486 с.

26. ТроелсенЭндрю. Языкпрограммирования C# 2008 и платформа .NET 3.5 / ЭндрюТроелсен. - М.: Вильямс, 2010. - 370 с.

27. Фримен Адам. ASP.NET MVC 3 Framework с примерами на C# для профессионалов / Адам Фримен , СтивенСандерсон. - М.: Вильямс, 2011. - 672 с.

28. Англо-русскийсловарь по вычислительнойтехнике. Компьютеры, мультимедиа, сети, Интернет, телекоммуникации, Windows / ред. М.С. Гуткин. - М.: ЭТС, 2013. - 496 с.

29. ВаракинАлександр. Windows XP. Обновлениямультимедиа. Windows MediaPlayer& Windows MovieMaker / АлександрВаракин. - М.: Майор, 2014. - 192 с.

30. Иванов В. Б. Компьютер, мультимеда, IP - телефония. Программы и программирование: моногр. / В.Б. Иванов. - М.: Майор, 2012. - 240 с.

31. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 6.050101 «Комп'ютерні науки / І.М. Удовик, Л.М. Коротенко, О.С. Шевцова; Нац. гірн. ун-т. – Д : НТУ «Дніпровська політехніка», 2018. – 65 с.

Додаток А

ЛІСТИНГ ПРОГРАМИ

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<title>1776 Logistics</title>
<!-- Meta -->
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="1776">
<link rel="shortcut icon" href="favicon.ico">
<!-- FontAwesome JS-->
<script defer src="assets/plugins/fontawesome/js/all.min.js"></script>
<!-- App CSS -->
 <link id="theme-style" rel="stylesheet" href="assets/css/portal.css">
</head>
<body class="app">
<header class="app-header fixed-top">
<div class="app-header-inner">
<div class="app-header-content">
<div><a><h2> Dashboard</h2></a></div>
</div>
</div>
<div id="app-sidepanel" class="app-sidepanel sidepanel-hidden">
<div id="sidepanel-drop" class="sidepanel-drop"></div>
<div class="sidepanel-inner d-flex flex-column">
<a href="#" id="sidepanel-close" class="sidepanel-close d-xl-none">&times;</a>
<div class="app-branding">
<a class="app-logo" href="index.html"><span class="logo-text">1776 LOGISTICS</span></a>
</div><!--//app-branding-->
<nav id="app-nav-main" class="app-nav app-nav-main flex-grow-1">
<ul class="app-menu list-unstyled accordion" id="menu-accordion">
```

```
<li class="nav-item">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

<a class="nav-link active" href="index.html">

<span class="nav-icon">

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-house-door" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M7.646 1.146a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 .146.354v7a.5.5 0 0 1-.5.5H9.5a.5.5 0 0 1-.5-.5v-4H7v4a.5.5 0 0 1-.5.5H2a.5.5 0 0 1-.5-.5v-7a.5.5 0 0 1 .146-.354l6-6zM2.5 7.707V14H6v-4a.5.5 0 0 1 .5-.5h3a.5.5 0 0 1 .5.5v4h3.5V7.707L8 2.207l-5.5 5.5z"/>

<path fill-rule="evenodd" d="M13 2.5V6l-2-2V2.5a.5.5 0 0 1 .5-.5h1a.5.5 0 0 1 .5.5z"/>

</svg>

</span>

<span class="nav-link-text">Dashboard</span>

</a><!--//nav-link-->

</li><!--//nav-item-->

<li class="nav-item">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

<a class="nav-link" href="docs.html">

<span class="nav-icon">

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-card-list" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M14.5 3h-13a.5.5 0 0 0-.5.5v9a.5.5 0 0 0 .5.5h13a.5.5 0 0 0 .5-.5v-9a.5.5 0 0 0-.5-.5zm-13-1A1.5 1.5 0 0 0 0 3.5v9A1.5 1.5 0 0 0 1.5 14h13a1.5 1.5 0 0 0 1.5-1.5v-9A1.5 1.5 0 0 0 14.5 2h-13z"/>

<path fill-rule="evenodd" d="M5 8a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7A.5.5 0 0 1 5 8zm0-2.5a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7a.5.5 0 0 1-.5-.5zm0 5a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7a.5.5 0 0 1-.5-.5z"/>

<circle cx="3.5" cy="5.5" r=".5"/>

<circle cx="3.5" cy="8" r=".5"/>

<circle cx="3.5" cy="10.5" r=".5"/>

</svg>

</span>
```

Contracts

<!--//nav-link-->

<!--//nav-item-->

<li class="nav-item">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-card-list" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M14.5 3h-13a.5.5 0 0 0-.5.5v9a.5.5 0 0 0 .5.5h13a.5.5 0 0 0 .5-.5v-9a.5.5 0 0 0-.5-.5zm-13-1A1.5 1.5 0 0 0 0 3.5v9A1.5 1.5 0 0 0 1.5 14h13a1.5 1.5 0 0 0 1.5-1.5v-9A1.5 1.5 0 0 0 14.5 2h-13z"/>

<path fill-rule="evenodd" d="M5 8a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7A.5.5 0 0 1 5 8zm0-2.5a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7a.5.5 0 0 1-.5-.5zm0 5a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 0 1h-7a.5.5 0 0 1-.5-.5z"/>

<circle cx="3.5" cy="5.5" r=".5"/>

<circle cx="3.5" cy="8" r=".5"/>

<circle cx="3.5" cy="10.5" r=".5"/>

</svg>

Loads

<!--//nav-link-->

<!--//nav-item-->

<li class="nav-item has-submenu">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-files" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M4 2h7a2 2 0 0 1 2 2v10a2 2 0 0 1-2 2H4a2 2 0 0 1-2-2V4a2 2 0 0 1 2-2zm0 1a1 1 0 0 0-1 1v10a1 1 0 0 0 1 1h7a1 1 0 0 0 1-1V4a1 1 0 0 0-1-1H4z"/>

<path d="M6 0h7a2 2 0 0 1 2 2v10a2 2 0 0 1-2 2v-1a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H6a1 1 0 0 0-1 1H4a2 2 0 0 1 2-2z"/>

</svg>

**

Pages

**

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-chevron-down" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z"/>

</svg>

<!--//submenu-arrow-->

<!--//nav-link-->

* *

<ul class="submenu-list list-unstyled">

*<li class="submenu-item">Notifications
*

*<li class="submenu-item">Account
*

*<li class="submenu-item">Settings
*

*
*

* *

*
<!--//nav-item-->*

<li class="nav-item">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

**

**

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-question-circle" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8 15A7 7 0 1 0 8 1a7 7 0 0 0 0 14zm0 1A8 8 0 1 0 8 0a8 8 0 0 0 0 16z"/>

<path d="M5.255 5.786a.237.237 0 0 0 .241.247h.825c.138 0 .248-.113.266-.25.09-.656.54-1.134 1.342-1.134.686 0 1.314.343 1.314 1.168 0 .635-.374.927-.965 1.371-.673.489-1.206 1.06-1.168 1.987l.003.217a.25.25 0 0 0 .25.246h.811a.25.25 0 0 0 .25-.25v-.105c0-.718.273-.927 1.01-1.486.609-.463 1.244-.977 1.244-2.056 0-1.511-1.276-2.241-2.673-2.241-1.267 0-2.655.59-2.752.286zm1.557 5.763c0 .533.425.927 1.01.927.609 0 1.028-.394 1.028-.927 0-.552-.42-.94-1.029-.94-.584 0-1.009.388-1.009.94z"/>

</svg>

**

Help

<!--//nav-link-->

*
<!--//nav-item-->*

*
<!--//app-menu-->*

</nav><!--//app-nav-->

* *

<nav class="app-nav app-nav-footer">

<ul class="app-menu footer-menu list-unstyled">

<li class="nav-item">

<!--//Bootstrap Icons: https://icons.getbootstrap.com/ -->

**

**

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-gear" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8.837 1.626c-.246-.835-1.428-.835-1.674 0l-.094.319A1.873 1.873 0 0 1 4.377 3.06l-.292-.16c-.764-.415-1.6.42-1.184 1.185l.159.292a1.873 1.873 0 0 1-1.115 2.692l-.319.094c-.835.246-.835 1.428 0 1.674l.319.094a1.873 1.873 0 0 1 1.115 2.693l.16.291c.415.764.42 1.6 1.185 1.184l.292-.159a1.873 1.873 0 0 1 2.692 1.116l.094.318c.246.835 1.428.835 1.674 0l.094-.319a1.873 1.873 0 0 1 2.693-1.115l.291.16c.764.415 1.6-.42 1.184-1.185l-.159-.291a1.873 1.873 0 0 1 1.116-2.693l.318-.094c.835-.246.835-1.428 0-1.674l-.319-.094a1.873 1.873 0 0 1-1.115-2.692l.16-.292c.415-.764-.42-1.6-1.185-1.184l-.291.159A1.873 1.873 0 0 1 8.93 1.945l-.094-.319zm-2.633-.283c.527-1.79 3.065-1.79 3.592 0l.094.319a.873.873 0 0 0 1.255.52l.292-.16c1.64-.892 3.434.901 2.54 2.541l-.159.292a.873.873 0 0 0 .52 1.255l.319.094c1.79.527 1.79 3.065 0 3.592l-.319.094a.873.873 0 0 0-.52 1.255l.16.292c.893 1.64-.902 3.434-2.541 2.54l-.292-.159a.873.873 0 0 0-1.255.52l-.094.319c-.527 1.79-3.065 1.79-3.592 0l-.094-.319a.873.873 0 0 0-1.255-.52l-.292.16c-1.64.893-3.433-.902-2.54-2.541l.159-.292a.873.873 0 0 0-.52-1.255l-.319-.094c-1.79-.527-1.79-3.065 0-3.592l.319-.094a.873.873 0 0 0 .52-1.255l-.16-.292c-.892-1.64.902-3.433 2.541-2.54l.292.159a.873.873 0 0 0 1.255-.52l.094-.319z"/>

<path fill-rule="evenodd" d="M8 5.754a2.246 2.246 0 1 0 0 4.492 2.246 2.246 0 0 0 0-4.492zM4.754 8a3.246 3.246 0 1 1 6.492 0 3.246 3.246 0 0 1-6.492 0z"/>

</svg>

Settings

<!--//nav-link-->

<!--//nav-item-->

<!--//nav-link-->

<!--//nav-item-->

<!--//footer-menu-->

</nav>

<!--//app-sidepanel-footer-->

<!--//sidepanel-inner-->

<!--//app-sidepanel-->

</header><!--//app-header-->

<h1 class="app-page-title">Overview</h1>

<h4 class="stats-type mb-1">Total Sales</h4>

$12,628

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-arrow-up" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8 15a.5.5 0 0 0 .5-.5V2.707l3.146 3.147a.5.5 0 0 0 .708-.708l-4-4a.5.5 0 0 0-.708 0l-4 4a.5.5 0 1 0 .708.708L7.5 2.707V14.5a.5.5 0 0 0 .5.5z"/>

```
</svg> 20%</div>

</div><!--//app-card-body→

<a class="app-card-link-mask" href="#"></a>

</div><!--//app-card→

</div><!--//col→

<div class="col-6 col-lg-3">

<div class="app-card app-card-stat shadow-sm h-100">

<div class="app-card-body p-3 p-lg-4">

<h4 class="stats-type mb-1">Expenses</h4>

<div class="stats-figure">$2,250</div>

<div class="stats-meta text-success">

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-arrow-down" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8 1a.5.5 0 0 1 .5.5v11.793l3.146-3.147a.5.5 0 0 1 .708.708l-4 4a.5.5 0 0 1-.708 0l-4-4a.5.5 0 0 1 .708-.708L7.5 13.293V1.5A.5.5 0 0 1 8 1z"/>

</svg> 5% </div>

</div><!--//app-card-body→

<a class="app-card-link-mask" href="#"></a>

</div><!--//app-card→

</div><!--//col→

<div class="col-6 col-lg-3">

<div class="app-card app-card-stat shadow-sm h-100">

<div class="app-card-body p-3 p-lg-4">

<h4 class="stats-type mb-1">Projects</h4>

<div class="stats-figure">23</div>

<div class="stats-meta"> Open</div>

</div><!--//app-card-body→

<a class="app-card-link-mask" href="#"></a>
```

```
</div><!--//app-card→
</div><!--//col→
<div class="col-6 col-lg-3">
<div class="app-card app-card-stat shadow-sm h-100">
<div class="app-card-body p-3 p-lg-4">
<h4 class="stats-type mb-1">Invoices</h4>
<div class="stats-figure">6</div>
<div class="stats-meta">New</div>
</div><!--//app-card-body→
<a class="app-card-link-mask" href="#"></a>
 </div><!--//app-card→
</div><!--//col→
</div><!--//row→
<div class="row g-4 mb-4">
<div class="col-12 col-lg-6">
<div class="app-card app-card-chart h-100 shadow-sm">
<div class="app-card-header p-3">
<div class="row justify-content-between align-items-center">
<div class="col-auto">
<h4 class="app-card-title">Line Chart</h4>
</div><!--//col→
<div class="col-auto">
<div class="card-header-action">
</div><!--//card-header-actions→
</div><!--//col→
</div><!--//row→
</div><!--//app-card-header→
```

```
<div class="app-card-body p-3 p-lg-4">
<div class="mb-3 d-flex">
<select class="form-select form-select-sm ml-auto d-inline-flex w-auto">
<option value="1" selected>This week</option>
<option value="2">Today</option>
<option value="3">This Month</option>
<option value="3">This Year</option>
</select>
</div>
<div class="chart-container">
<canvas id="canvas-linechart" ></canvas>
</div>
</div><!--//app-card-body→
</div><!--//app-card→
</div><!--//col→
<div class="col-12 col-lg-6">
<div class="app-card app-card-chart h-100 shadow-sm">
<div class="app-card-header p-3">
<div class="row justify-content-between align-items-center">
<div class="col-auto">
<h4 class="app-card-title">Bar Chart</h4>
</div><!--//col→
<div class="col-auto">
<div class="card-header-action">
</div><!--//card-header-actions→
</div><!--//col→
</div><!--//row→
```

```
</div><!--//app-card-header→
<div class="app-card-body p-3 p-lg-4">
<div class="mb-3 d-flex">
<select class="form-select form-select-sm ml-auto d-inline-flex w-auto">
<option value="1" selected>This week</option>
<option value="2">Today</option>
<option value="3">This Month</option>
<option value="3">This Year</option>
</select>
</div>
<div class="chart-container">
<canvas id="canvas-barchart" ></canvas>
</div>
</div><!--//app-card-body→
</div><!--//app-card→
</div><!--//col→
</div><!--//row→
<div class="col-12 col-lg-6">
<div class="app-card app-card-stats-table h-100 shadow-sm">
<div class="app-card-header p-3">
<div class="row justify-content-between align-items-center">
<div class="col-auto">
<h4 class="app-card-title">Stats List</h4>
</div><!--//col→
<div class="col-auto">
</div><!--//col→
</div><!--//row→
```

```
</div><!--//app-card-header→

<div class="app-card-body p-3 p-lg-4">

<div class="table-responsive">

<table class="table table-borderless mb-0">

<thead>

<tr>

<th class="meta">Area</th>

<th class="meta stat-cell">Performance</th>

<th class="meta stat-cell">Today</th>

</tr>

</thead>

<tbody>

<tr>

<td><a href="#">SMF</a></td>

<td class="stat-cell">100</td>

<td class="stat-cell">

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-arrow-up text-success" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8 15a.5.5 0 0 0 .5-.5V2.707l3.146 3.147a.5.5 0 0 0 .708-.708l-4-4a.5.5 0 0 0-.708 0l-4 4a.5.5 0 1 0 .708.708L7.5 2.707V14.5a.5.5 0 0 0 .5.5z"/>

</svg>  8%

</td>

</tr>

<tr>

<td><a href="#">RNO</a></td>

<td class="stat-cell">67</td>

<td class="stat-cell">0%</td>

</tr>

<tr>
```

<td>MEM</td>

<td class="stat-cell">56</td>

<td class="stat-cell">

<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-arrow-down text-danger" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8 1a.5.5 0 0 1 .5.5v11.793l3.146-3.147a.5.5 0 0 1 .708.708l-4 4a.5.5 0 0 1-.708 0l-4-4a.5.5 0 0 1 .708-.708L7.5 13.293V1.5A.5.5 0 0 1 8 1z"/>

</svg> 20% </td>

</tr>

<tr>

<td>DEN</td>

<td class="stat-cell">24</td>

<td class="stat-cell">-</td>

</tr>

<tr>

<td>LAS</td>

<td class="stat-cell">17</td>

<td class="stat-cell">15%</td>

</tr>

</tbody>

</table>

<!--//table-responsive→

<!--//app-card-body→

<!--//app-card→

<!--//col→

<!--//row→

<footer class="app-footer">

```
</footer><!--//app-footer→

</div><!--//app-wrapper→

<!-- Javascript →

<script src="assets/plugins/popper.min.js"></script>

<script src="assets/plugins/bootstrap/js/bootstrap.min.js"></script>

<!-- Charts JS →

<script src="assets/plugins/chart.js/chart.min.js"></script>

<script src="assets/js/index-charts.js"></script>

<!-- Page Specific JS →

<script src="assets/js/app.js"></script>

</body>

</html>
```

Додаток Б

ВІДГУК

керівника економічного розділу

Додаток В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	

Диплом Стрижак.docx	Пояснювальна записка
Диплом Стрижак.pdf	Пояснювальна записка
Програма	
1776.rar	Архів. Містить коди програми ш скомпильовану програму
Презентація	
Презентація Стрижак.ppt	Презентація дипломного проекту