МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
підпис

«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Система автоматизації збору інформації до туристичної карти "Фортеці України"

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям підготовки 6.050102 – "комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	Кривуля Г.Ф. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Новаченко В.В. (ініціали, прізвище)
Група:		КІ-14з

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 – "комп'ютерна інженерія"
(шифр і назва)
Спеціальність
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Новаченко Валерію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система автоматизації збору інформації до туристичної карти "Фортеці України"

керівник проекту (роботи) Кривуля Г.Ф., д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "14" 05 2018р. № 118/48
2. Термін подання студентом роботи 17.06.2018
3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області та постановка задачі, Вибір засобів для розробки, розроблення системи автоматизації збору інформації для туристичної карти Фортеці України, Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	ст. викл. Критська Я.О.		

7. Дата видачі завдання 14.05.2018

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з предметною галуззю	03.05 – 15.05	
2	Аналіз існуючих аналогів	16.05 – 18.05	
3	Вибір засобів для розробки парсера	19.05 – 25.05	
4	Розробка інтерактивної карти і парсера	26.05 – 03.06	
5	Розробка розділу «Охорона праці та	04.06 – 08.06	
	безпека в надзвичайних ситуаціях»		
6	Оформлення пояснювальної записки	09.06 – 15.06	

Студент _____ Новаченко В.В.
(підпис) (прізвище та ініціали)

Керівник _____ Кривуля Г.Ф.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота бакалавра складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

Робота містить 81 сторінку, 24 рисунків, 4 додатки, 25 джерел.

Об'єкт дослідження: додаток, що надає інформацію про музеї та фортеці України.

Мета: розробка додатку, що надає актуальну інформацію про музеї та фортеці України. Отже, щоб отримати потрібну інформацію не треба її шукати на просторах інтернету, а можна буде отримати на одному сайті, який буде збирати дані з різних місць, тому перейшовши по посиланню можна отримати більш детальну і корисну інформацію.

В процесі розробки розглядаються різні засоби і бібліотеки парсингу. Проведена їх порівняльна характеристика.

Ключові слова: PHP, MySQL, парсинг (parsing), HTML, SVG, DOM.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Парсер - засіб синтаксичного аналізу

Парсинг - процес синтаксичного аналізу інтернет-ресурсу з метою вилучення деяких даних.

CSV - Comma-Separated Values

DOM - Document Object Model

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

JSON - JavaScript Object Notation

MySQL – My Structured query language

PHP – Hypertext Preprocessor

SVG - Scalable Vector Graphics

Web – система доступу до пов'язаних між собою документів на різних комп'ютерах, підключених до Інтернету

XML – eXtensible Markup Language (розширювана мова розмітки)

ВСТУП

У сучасному світі все більшого значення набувають електронні видання, які прийшли на заміну паперовим. Серед чинників, які обумовлюють зростаючий попит на інформацію в електронному вигляді можна відзначити широке охоплення аудиторії, відсутність обмежень на обсяг, простоту пошуку, наочність й інтерактивність, можливість взаємодії з користувачем [1]. Інтерактивність при роботі з електронними документами дозволяє як знайомитися з інформацією, так і активно працювати з нею. Особливо актуальні серед інтерактивних засобів способи візуального представлення інформації, такі як відео, анімація, 3D-графіка та ін. Веб-документи, створені за допомогою нових стандартів, таких як HTML5, дозволяють реалізувати всі перераховані вище вимоги до інтерактивності і наочності подання інформації.

Розробка сучасних WEB-додатків вимагає зусиль безлічі розробників вузької спеціалізації. Незважаючи на вузькопрофільність, кожен з них повинен глибоко розуміти тенденції сучасних технологій з метою їх найбільш ефективного використання при створенні інтерактивних WEB-додатків. Серед засобів, які допомагають в розробці інтерактивних WEB-додатків, можна відзначити: Тег canvas (в перекладі з англ. Означає полотно), введений в HTML5, що дозволяє швидко і ефективно працювати з растровою і векторною графікою [2]. Даний елемент застосовується для створення різноманітних додатків, включаючи елементи навігації, графічні інструменти, повноцінні програми, аркадні та онлайн ігри і симулятори. Тег canvas створює область на веб-сторінці, яка може бути використана для відтворення графіки в режимі реального часу за допомогою сценаріїв (наприклад, за допомогою мови JavaScript), а також може бути використаний для створення годин, анімації і т. Д.

Тег svg дозволяє малювати векторні зображення у форматі svg (Scalable Vector Graphics) [3], заснованому на форматі опису даних xml. Такі зображення вигідно відрізняються тим, що не змінюють свого якості при масштабуванні. Так як даний формат є стандартним, можливо імпортування зображень з таких програмних продуктів, як Inkscape і Adobe Illustrator. Також зображення доступні для зміни з javascript у вигляді dom (document object model).

CSS (Cascading Style Sheets), а саме третя версія даного інструменту, дозволяє, крім безпосереднього управління стилями і блоками WEB-документа, також і створення анімації засобами браузера. З появою CSS3 розробники стали віддавати перевагу цей інструмент canvas і svg для малювання ілюстрацій [4]. Підтримка анімації елементів за допомогою браузера значно прискорює промальовування і робить її більш плавною, ніж

реалізація її більш старими методами (наприклад, за допомогою jQuery). CSS анімацію доцільно використовувати для реалізації популярної техніки так званого parallax scrolling [5], коли при прокручуванні документа фон і вміст переміщаються з різною швидкістю, створюючи ефект тривимірного простору в WEB-додатку.

Наведені вище інструменти сприяють створенню красивих, зручних і насичених можливостями WEB-додатків, спрощують розробку більш простими методами. Прихід у світ WEB-технологій HTML5 привів до майже повної відмови від старих інструментів, таких як Adobe Flash, Java Applet і Microsoft Silverlight [6], які мали проблеми з безпекою, сумісністю з різними платформами і не були вільними. В даний час інтерактивні WEB-сторінки, створені з використанням сучасних методів, можуть працювати на будь-яких домашніх, мобільних і стандартних платформах, таких як Smart TV, iPhone, Android, Windows Phone, Windows, Linux і MacOS, в будь-яких доступних для них браузерах, дозволивши проблеми з сумісністю.

Метою роботи є розробка додатку, що надає актуальну інформацію про музеї та фортеці України.

Результати роботи: розроблено додаток, що надає актуальну інформацію про музеї та фортеці України з можливістю розширення.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Для того щоб система автоматизації збору інформації, що розробляється була ефективно реалізована і мала успіх, необхідно з самого початку розуміти що необхідно отримати в результаті. Тому спочатку проаналізуємо існуючі аналогічні рішення і сформулюємо основні вимоги до розроблюваного додатку, потім приступимо до проектування. Адже проектування - це дуже важливий етап при розробці проекту. Від того, як добре ми спроектуємо базу даних і архітектуру додатку залежить дуже багато чого. Не вірно прийняте рішення при проектуванні в подальшому може спричинити серйозні проблеми. Тому етапу проектування необхідно приділити належну увагу.

1.1 Туристичні карти фортець і музеїв України

Практично в кожній області України можна знайти стародавні споруди. Про багато з них ми знаємо давно, в деяких навіть встигли побувати. Величні фортеці, потужні стіни яких вистояли не одне століття. Є й замок, на який жителі України дивляться набагато частіше, ніж на всі інші, бо зображений він на купюрі 200 гривень - це замок Любарта у Луцьку. Є і ті, які менш популярні серед українців і гостей нашої країни.

Але є в Україні і такі замки і фортеці, якими можна захоплюватися і пишатися. Величні фортеці, потужні стіни яких вистояли не одне століття і ще століттями будуть дивувати наших нащадків, заслуговують того, щоб бути побаченими. Так ось саме вони і викликають найбільший інтерес.

Звичайно більшість замків, які є в Україні, розташовані в Західній Україні. Красиві замки і фортеці Західної України не вимагають додаткових рекомендацій. В основному це відомі перлини архітектури, більшість з яких збереглись до наших днів в хорошому стані. Хоча, звичайно, не всім колишнім замкам-красеням так пощастило ... Але і вони можуть стати досить мальовничим фоном для Ваших карткових фотографій. Це замки Волині, Тернопільщини, Львівщини, Івано-Франківщини, Буковини і Закарпаття. Сьогодні ми побуваємо в замках, фортецях і палацах України, в яких обов'язково варто побувати.

Щодня різні пошуковий системи обробляють мільярди пошукових запитів користувачів, але далеко не завжди їх відповіді містять вичерпну інформацію на поставлене запитання. Пошук туристичних маршрутів, екскурсій, музеїв та перлин стародавньої архітектури не є винятком. Так, наприклад, щоб знайти фортеці та музеї конкретного регіону з деякими додатковими критеріями, необхідно відвідати далеко не один сайт. Само собою, щоб не витрачати час на пошуки, виникає бажання мати під

рукою потрібну інформацію про конкретне місто, особливо в наші дні, коли різні мобільні пристрої вже зайняли міцне місце в житті людей і продовжують активно розвиватися. Тому інформаційні додатки для швидкого доступу як до послуг, так і до інформації, є досить затребуваними на ринку, причому це стосується всіх сфер життя суспільства.

У даній області вже існують рішення зі своїми перевагами і недоліками: деякі додатки не обмежуються інформацією про музеї і фортеці, надаючи інформацію також про багато інших подій, деякі надають розширену інформацію про одну конкретну фортецю (як правило, дуже відомому). Проаналізувавши існуючі рішення в даній області, можна скласти уявлення як має виглядати подібний додаток для максимального комфорту користувача.

У рамках даної дипломної роботи було вирішено розробити додаток, що надає актуальну інформацію про музеї та фортеці України. Основними завданнями даної роботи є:

1) Отримання інформації про музеї та фортеці за допомогою парсингу відповідних джерел інформації.
2) Реалізація списку фортець та музеїв для виведення інформації в зручному вигляді.
3) Реалізація інтерактивної карти України з зазначеними на ній фортецями.
4) Розробка додатку з можливістю розширення.

1.1.1 IGOtoWorld.com

Веб-сайт IGOtoWORLD.com не обмежується інформацією про музеї, фортеці, готелі, а також пропонує різноманітні тури (рис. 1.1 – 1.2).

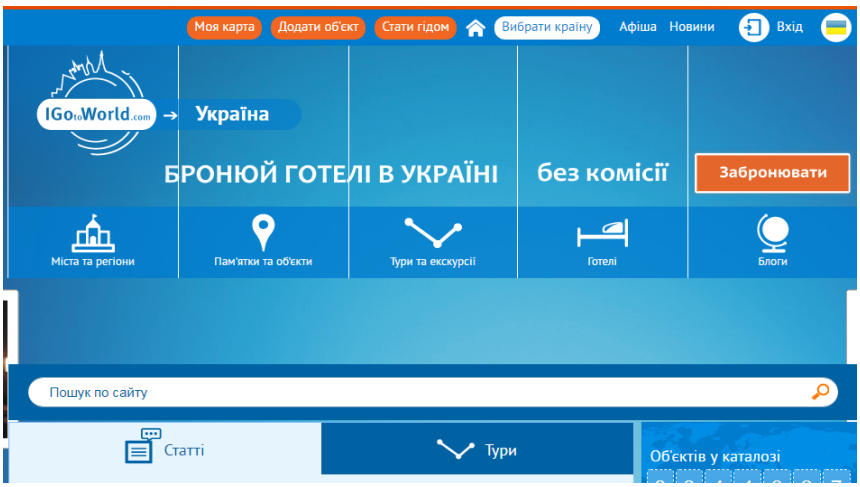


Рисунок 1.1 – Головна сторінка

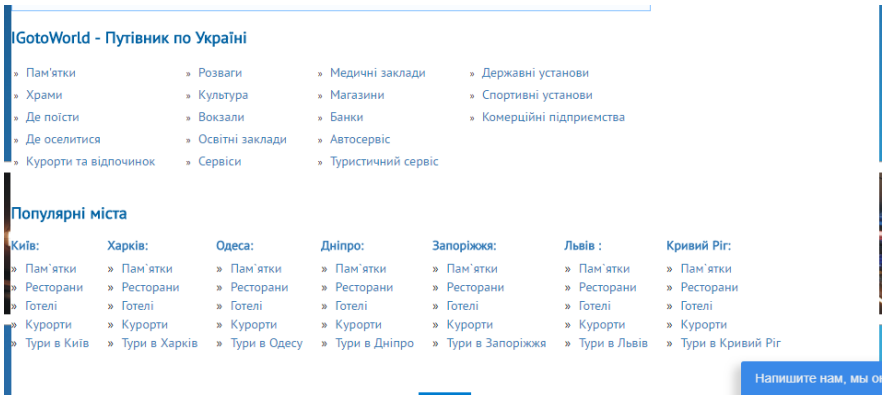


Рисунок 1.2 – Вибір міста та події

Переваги:

– великий список підтримуваних міст;

– можливість залишити відгук про будь-який із заходів і / або закладу;

– онлайн-підтримка;

– багатий вибір об’єктів;

– підтримка декількох мов(українська, російська, англійська).

Недоліки:

– серед найпопулярніших великі і відомі міста, а менші міста не отримали належної уваги;

– не завжди актуальна інформація щодо часу роботи і вартості квитків, а також мало інформації щодо міста розташування та як доїхати(рис. 1.3).

Рисунок 1.3 – Схема проїзду та години роботи

1.1.2 LVIVGALLERY.org.ua

Цей сайт також пропонує багато корисної інформації про музеї та виставки. На відміну від раніше розглянутого рішення веб-сайт орієнтований тільки на місто Львів та область і надає інформацію по музеям Львова та Львівської області.

Переваги:

- зручний фільтр пошуку музеїв та подій(рис.1.4);
- багатий вибір об’єктів;
- актуальна та достатня інформація щодо робочого часу, вартості квитків, місця знаходження , а також проїзду(рис. 1.5);

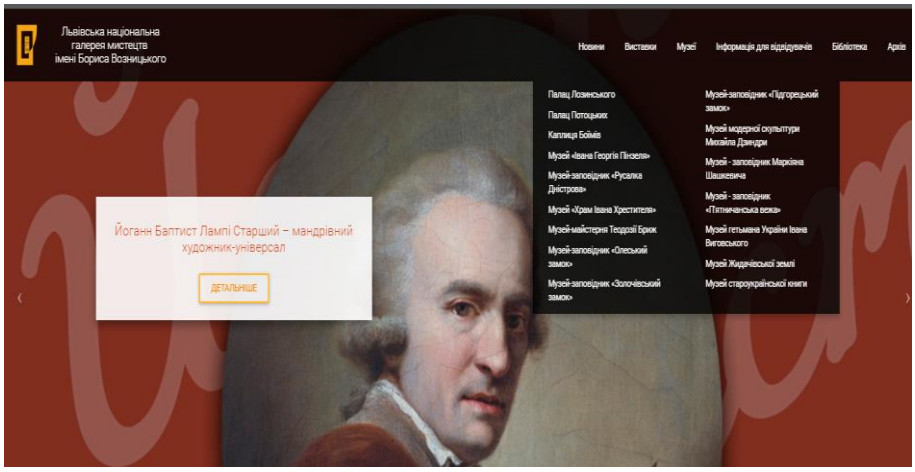


Рисунок 1.4 – Меню фортець та музеїв

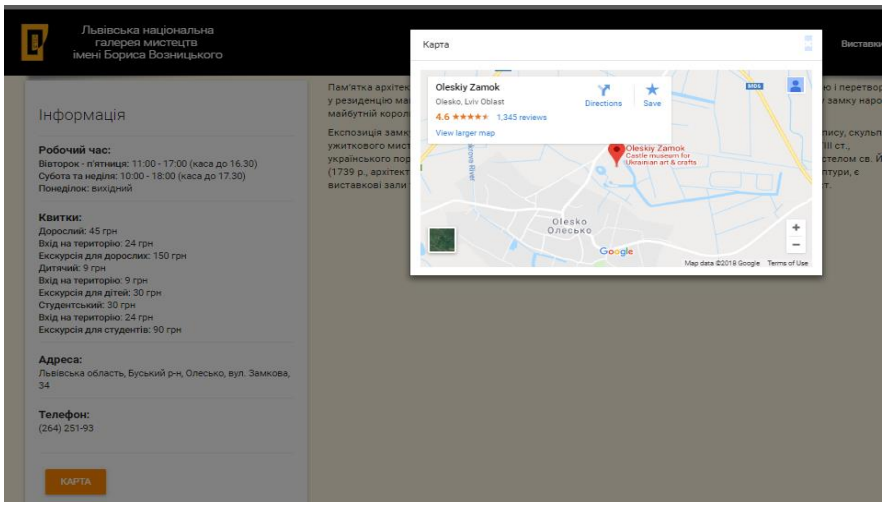


Рисунок 1.5 – Інформація про робочий час,вартість квитків,контактні дані та схему проїзду

Недоліки:

- замало загальної інформації при опису історії фортеці(рис. 1.5);

– інформація надається тільки для Львову та області;

– наявність однієї мови;

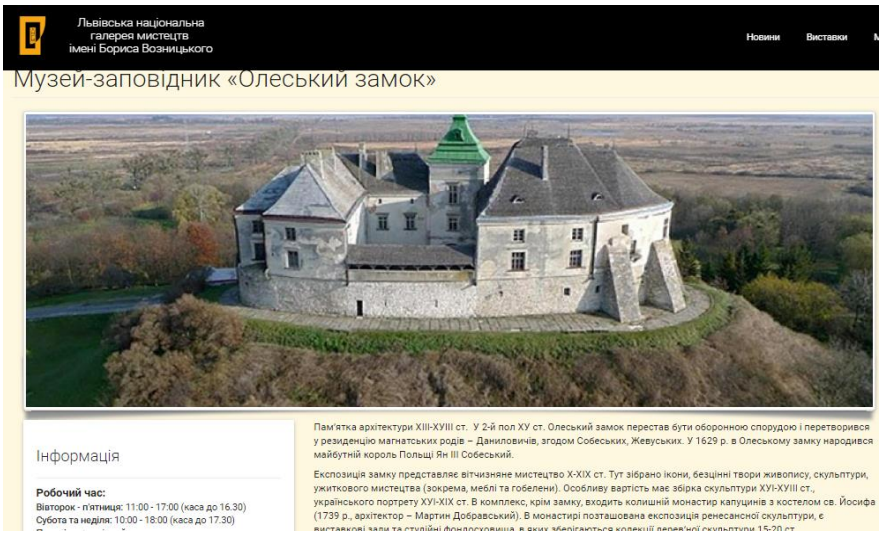


Рисунок 1.6 – Інформаційна сторінка Олеського замку

1.1.3 GLOSS.ua

Отже, до вашої уваги велика добірка найцікавіших замків і фортець України – з Півдня на Схід, з Півночі і Центру на Захід.

Даний веб-сайт несе більш ознайомчу інформацію щодо фортець різних куточків країни. На головній сторінці присутня різноманітна інформація щодо культурного життя та подій Києва(рис. 1.9).

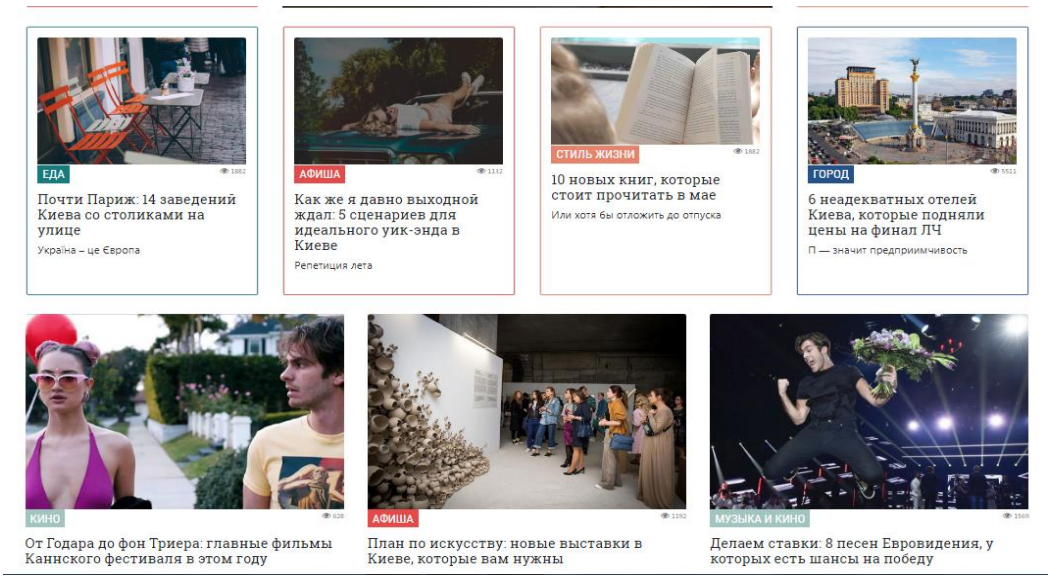


Рисунок 1.7 – Актуальні події

До переваг можна віднести достатньо великий перелік замків і фортець України та різноманітних подій, які постійно оновлюються.

Недоліки теж доволі видимі: відсутня інформація щодо робочого часу, вартості квитків, місця знаходження , а також проїзду(рис. 1.7 – 1.8).

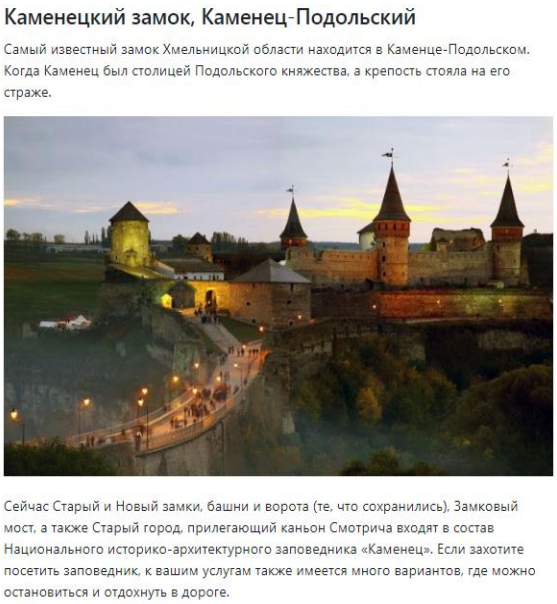


Рисунок 1.8 – Кам’янецький замок

Рисунок 1.9 – Луцький замок

Таблиця 1.1 – Порівняльний аналіз аналогічних рішень

Вимоги	IGOtoWORLD.com	LVIVGALLERY.org.ua	GLOSS.ua
Загальна інформація о фортецях	так	так	так
Інтерактивна карта	ні	ні	ні
Багатомовність	так	ні	ні
Час роботи, ціна квитків	так /ні	так	ні
Схема проїзду	так /ні	так	ні
Динамічне оновлення інформації	так	так	так

Аналіз існуючих рішень показав, що сьогодні існує безліч сайтів подібної тематики. Однак, або їх функціональність вельми обмежена, або дані надані цими ресурсами охоплюють лише частину фортець. Відсутність зручного інтерфейсу для

пошуку, неповні відомості та подання результатів пошуку робить їх незручними у використанні. Як наслідок-такі сайти несуть мало корисної інформації і лише деякі з них представляють актуальні дані. Тому було прийнято рішення про розробку власного програмного продукту, в якому необхідно врахувати недоліки існуючих систем.

1.2 Технічне завдання на розробку

Одною із основних потреб людського життя є інформація. Ми з кожним днем пізнаємо щось нове, читаємо, дивимось ТБ, слухаємо радіо, спілкуємося з людьми. Інколи людина, користуючись Інтернетом, втрачає багато свого дорогоцінного часу, щоб найти конкретну інформацію, адже потрібно переглянути не один сайт, наткнутися на рекламу, інтерфейс деяких сайтів є доволі таки складним і найти там щось потрібне займає багато часу. Інколи людина заходить у глухий кут, потрібна інформація не знайдена, залишається обмаль часу на пошук. Виникає потреба в пришвидшені процесу збору інформації з інших джерел.

Отже в даному проекті буде розроблятися система автоматизації збору інформації, що надає актуальну інформацію про музеї та фортеці України з можливістю розширення.

1.2.1 Функціональні характеристики

Інформаційна система повинна надавати наступні можливості:

- пошук інформації на інтерактивній карті, для конкретної області ;
- ознайомлення з замками та фортецями обраної області;
- перегляд детальної інформації щодо обраного замку;
- отримання інформації про часи роботи та вартості вхідних квитків;
- отримання інформації про розташування та схему проїзду;
- інформація повинна бути достовірна і актуальна на даний період часу;
- система повинна бути ефективною та надійною;
- не містити сторонніх або шкідливих програм, що можуть привести до непрацездатності системи;
- система має бути розроблена таким чином, щоб мати можливість розширення.

1.2.2 Вимоги до інтерфейсу

Так як цільова аудиторія, яка буде користуватися системою - це туристи, мандрівники і прості громадяни, тому процес інтерактивної взаємодії користувача із системою повинен відповідати таким вимогам, як: швидкість, зручність, низький рівень втоми.

Інтерфейс повинен мати такі якості:

- інтуїтивно зрозумілий;
- забезпечувати простоту переходу від виконання однієї функції до іншої;
- максимальна простота його використання і готовність в повній мірі задовольнити запити користувача при розв’язанні визначених задач;
- максимальна простота його використання, він не повинен містити зайвих декоративних деталей, які заважають;
- повинен бути консистентним, тобто, ґрунтуватись на використанні відомих, загальноприйнятих методів і засобів представлення інформації.
- в ідеалі процес взаємодії користувача з системою не повинен представляти ніяких труднощів;
- підтримувати сучасні версії браузерів.

1.2.3 Вимоги до програмного забезпечення серверної частини

Для функціонування системи необхідно наступне програмне забезпечення:

- Веб-сервер – Apache версії не ниже 1.3.26;
- СУБД – MySQL версії не нижче 3.23;
- PHP. Для роботи продукту потрібна наявність PHP версії не нижче 5.3.x.

Мінімальні системні вимоги до комп’ютера, на який встановлюється серверна частина:

- операційна система: Windows, Mac OS та Linux.
- частота процесора: 1,5 Ггц або вище;
- оперативна пам'ять: 1 Гб або більше;
- місце на жорсткому диску: 120 Мб або більше.

1.2.4 Вимоги до клієнтського програмного забезпечення

Система, що розробляється повинна бути доступна для перегляду за допомогою найбільш популярних браузерів:

- Opera 6.0 та вище;
- Google Chrome 10.0 та вище;
- Mozilla 1.7 та вище.

Для коректного функціонування клієнтської частини радимо обирати ПК, які відповідають наступним мінімальним системним вимогам:

- Операційна система: Windows, Mac OS та Linux;
- Частота процесора: 1 ГГц або вище;
- Оперативна пам'ять: 512 Мб або більше;
- Місце на жорсткому диску: 120 Мб.

Висновок по розділу 1

Створення системи автоматизації збору інформації з кожним роком набуває все більшої популярності, адже спрямоване на полегшення роботи користувача та надає постійний доступ до потрібних та актуальних даних, та дуже економить час на пошук, підготовку та видачу даних. Тому використання системи автоматизації для отримання необхідної інформації про музеї та фортеці України і не тільки, а також в інших сферах людської діяльності, на даний момент являється актуальної темою.

2 АНАЛІЗ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ПАРСИНГУ

2.1 Середовище розробки та інструменти для отримання і зберігання даних

При розробці системи автоматизації збору інформації будуть використовуватися такі технології як:

- Inkscape - вільно розповсюджуваний векторний графічний редактор для всіх операційних систем;
- Sublime Text3 - універсальний текстовий редактор. Він працює на OS X, Windows і Linux. Підтримує багато різних плагинів, а також синтаксис багатьох мов програмування;
- Apache HTTP-сервер;
- база даних mySQL та панель адміністрування MySQL – phpMyAdmin.

2.1.1 Поняття веб-додатку, його переваги

Всім користувачам персональних комп'ютерів відомо, що таке додаток Windows. Це одна з програм, яка встановлюється на комп'ютер і працює в операційному середовищі ОС Windows. Текстові та графічні редактори, медіаплейери й поштові клієнти тощо. Розглянемо, що ж з себе представляє веб-додаток, ось декілька визначень з різних джерел.

Веб-додаток – розподілений додаток, в якому клієнтом виступає браузер, а сервером – веб-сервер[10].

Веб-додатки – це програми, написані скриптовою мовою (Perl, PHP та ін.) або написані мовою високого рівня та відкомпільовані під відповідну ОС (C, C++ та ін.), які працюють на стороні веб-сервера та призначені для створення інтерфейсу між користувачем та веб-сайтом.

Отже, веб-додаток – це комп’ютерна програма, яка працює в браузері, як Microsoft Word працює в ОС Windows. Тому, для доступу до програми потрібні браузер та Інтернет. Зберігання та обробка інформації при такій організації обчислень відбувається на віддаленому сервері, а веб-переглядач служить програмою-клієнтом і призначеним для користувача інтерфейсом (рис. 2.1) [11].

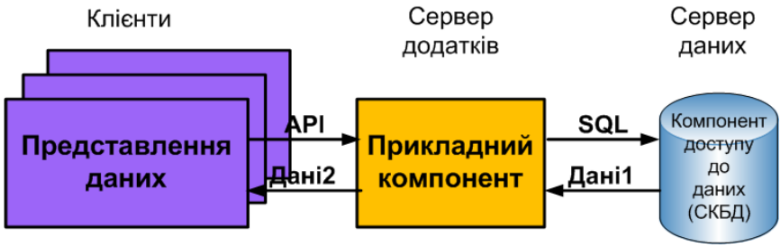


Рисунок 2.1 - Схема роботи веб-додатку

Текстовий пакет Microsoft Office буде працювати тільки під Windows. А ось для веб-додатка операційна система встановлена на комп'ютері не має значення. Тому, що і операційною системою, і користувацьким інтерфейсом веб-додатка є браузер.

В результаті такої універсальності, постійний користувач веб-додатка 6 може абсолютно без проблем працювати зі своєю улюбленою програмою на будь-якому зі своїх девайсів, починаючи з офісного стаціонарного комп'ютера, закінчуючи планшетом і смартфоном.

У прикладному відношенні – веб-додатки мають ту істотну перевагу: вже розроблено багато програм і сервісів, за допомогою яких будь-яка людина, не будучи програмістом і навіть просунутим користувачем, може створювати різні корисні програми для своєї зручності та розваги. Причому абсолютно безкоштовно.

Коротко опишемо, що представляють собою технології, які зазвичай використовуються для створення веб-додатків.

- Веб-сервер. Веб-сервером називається програма, яка аналізує приходять запити і формує готові документи, відправляються користувачеві. Як веб-сервер часто виступає Apache, як найбільш популярний в Інтернеті. За різними оцінками його частка становить майже 50% від загального числа використовуваних веб-серверів.

- Мова програмування. PHP - популярний універсальна мова, який особливо підходить для веб-розробки.

- СУБД. MySQL

- Веб-інтерфейс для СУБД. phpMyAdmin - веб-інтерфейс для створення і управління базами даних MySQL. Дозволяє переглядати таблиці, змінювати їх зміст, модифікувати структуру, робити вибірку даних, сортувати інформацію. Усі дії здійснюються прямо у браузері, в спеціально розробленому під нього дружньому інтерфейсі.

2.1.2 Парсери

В даний час всі процеси, де застосовується синтаксичний аналіз, використовують парсери - програми для проведення візуального або програмно-автоматизованого синтаксичного і лексичного аналізу або розбору будь-якого документа з метою вилучення з нього необхідних даних. Це і різні автоматизовані перекладачі з однієї мови на іншу, і транслятори мов програмування, які формують програмний код на машинно-орієнтовану мову, це і мова SQL-запитів і тому подібні застосування.

Перш, ніж розробляти парсер, необхідно визначитися з його визначенням і призначенням.

Отже, парсер контенту - це не що інше, як скрипт, здатний сортувати інформацію, виділяючи найважливішу і обробляючи її згідно з алгоритмом, створеному для вирішення того чи іншого завдання. Для чого, наприклад, можна використовувати парсер контенту.

По-перше, як відомо, найкращі сайти - це ті Інтернет-ресурси, на яких є цікава актуальна інформація. Нікому не потрібні вчорашні новини і сенсації, наприклад. Також, коли мова йде про сайти-обмінники валют, де необхідно змінювати інформацію про курс валют часом по кілька разів на день, створення парсера вкрай необхідно. Скрипт в таких випадках буде виконувати всю роботу сам, цілодобово відстежуючи зміни курсу валют в Нацбанку.

По-друге, парсер необхідний для автоматичного оновлення вашого Інтернет-ресурсу. Користувачі, які зайшли на вашу сторінку один раз і виявили там застарілу інформацію, більше ніколи не повернуться на неї. Саме тому для збереження постійних користувачів, а також для залучення нових необхідне регулярне оновлення інформації на вашій інтернет-сторінці.

По-третє, парсер контенту - ідеальний інструмент для миттєвого наповнення сайту корисними даними. Так, коли в мережі величезна кількість сайтів різної спрямованості, лише мала їх частина виявляється в кінцевому підсумку корисними користувачам. Саме тому, важливо, щоб інформація на вашій веб-сторінці була не тільки актуальною, але ще і корисною.

І, по-четверте, - централізація даних. Відомо, що інформації в мережі дуже багато, при цьому самої різної. Вся проблема в тому, що вся ця інформація розкидана по безлічі Інтернет-рессурсах і зібрати її не так просто. Використовуючи парсер контенту, можна об'єднати вичерпну кількість корисної інформації у себе на сайті, тим самим залучаючи все більше відвідувачів. Такий хід - відмінний варіант для далекоглядних розробників, які піклуються про те, щоб навіть випадкові відвідувачі, ставали постійними. Саме для вирішення такої задачі присвячена ця дипломна робота.

Ми розробимо програму-парсер, яка буде збирати дані з інформаційних сайтів про фортеці та музеї України. Завдяки розробленому додатку всі отримані дані будуть розміщені в одному місці, що дасть змогу відвідувачам знайти необхідну інформацію за короткий період часу. Потрібно всього лиш зайти на сторінку додатка, вибрати потрібний розділ, і перейшовши по посиланню, користувач в результаті отримає потрібну йому інформації. Не потрібно більше заморочуватись з пошуком інформації, переходити по необмеженій кількості посилань, задавати фільтрацію даних, натикатись на кучу реклами, все це за вас зробить програма- парсер.

Отже переваги парсеру контенту очевидні. Тому даний скрипт можна використовувати тільки на користь при грамотному підході і бажанні розробити web-додаток з високим ступенем відвідуваності.

2.1.3 Семантика HTML- коду

Отже для збору інформації необхідно розібратися в досліджуваній області. Типовий веб-сайт зазвичай складається з набору веб-сторінок, а веб-сторінки зазвичай являють собою текстові файли у форматі * .html. Веб-сторінки можуть містити посилання на інші файли в інших форматах, а також гіперпосилання. Гіперпосиланням називають частину гіпертекстового документа, що посилається на інший елемент в документі, на інший об'єкт або на елементи цього об'єкта.

Для створення HTML-сторінки використовують HTML-теги. Кожен HTML-тег має своє призначення і дає інформацію браузеру про тому, як його слід відображати. Крім HTML-тегів використовуються ідентифікатори і класи, що дозволяють змінити відображення при використанні каскадних таблиць стилів (CSS). Зазвичай при верстці (написанні HTML коду) структура веб-сторінки розділяється на 3-й блоку: шапка - header, контент - content і підвал - footer. У кожному блоці знаходиться специфічна інформація щодо веб-сайту, така як назва сайту, основний текст і реквізити. Наприклад, зазвичай в якості нижньої частини (підвал) веб-сторінки встановлюють клас або ідентифікатор «footer», де може знаходиться назва компанії, авторські права, дублікат основного меню веб-сайту, контактна або коротка інформація про компанію.

Перехід на новий стандарт HTML 5 привів до використання нових тегів, як наприклад: header, footer, main. За новим стандартом теги тепер мають смислове навантаження і тепер повинні використовуватися в залежності від їх призначення. Таким чином вводяться теги являють собою збагачення семантичного змісту веб-сторінки, такі зміни призводять до уточнення типу мультимедіа об'єктів (відео, зображення, звук і т.д.), розширення універсальних блокових і текстових елементів для визначення

інформаційного посилу інформації. Дизайнери веб-сайтів зазвичай усвідомлено використовують стандарт 5-ої версії HTML, так як це спрощує читаність коду і його аналіз пошуковими роботами.

Для отримання даних з сайтів необхідно написати модуль. Інформація повинна бути розділена на безліч різних категорій, що дозволить спростити роботу з нею. Орієнтовна категоризація даних:

- посилання (текст в тезі «a» і посилання «href»)
- текст (весь текст сторінки)
- заголовки
- спеціалізовані теги (meta, head, title)
- теги для конкретизації вибірки (strong, div, теги навігації)
- зображення (шлях «src», текст опису «alt»)
- і ін.

Завдяки переходу майже всього інтернету на Web2.0 (HTML5) з'явилася можливість розкладання сторінки на семантичні одиниці, що дозволить спростити категоризацію. Тому в якості додаткової вибірки буде ще кілька категорій:

1. контент (вміст «div.content, div # content, article, section»)
2. шапка (вміст «#header, .header, header»)
3. земельна ділянка (вміст «#footer, .footer, footer»)
4. навігація (вміст «.nav a, #nav a, #menu a, .menu a, ul a»)

Вибірка виконана з урахуванням використання стандартів як HTML 4, так і HTML5 - при наявності необхідного класу або ідентифікатора тег ставиться за стандартом 4-й версії інакше за стандартом 5-ої версії.

Тепер, коли нам відомі основні можливі групи контенту можна розглянути варіант збереження отриманих даних. Так як модель буде модифікуватися, значить необхідно зберегти дані, щоб вони були доступні для дослідження впливу тих чи інших параметрів. У нашому випадку зберігання в файлі буде не таким зручним, так як необхідно буде використовувати відносини, а в базах даних (БД) це дуже просто реалізовано. При зборі бажано зберігати «сирі» дані, в нашому випадку це вихідний код веб-сторінок. Так само спрощується пошук, сортування та організація даних. Отже простим запитом можна отримати необхідну вибірку.

2.2 Вибір інструментів для реалізації туристичної карти фортець України

Визначившись зі структурою веб-додатку, проаналізувавши веб-сайти фортець знайдених в Інтернеті, для створення додатку був використаний наступний програмний комплекс:

- мова гіпертекстової розмітки HTML;
- мова стилів CSS;
- база даних MySQL;
- локальний сервер Apache;
- мови програмування PHP та JavaScript.

Всі ці програми і програмні комплекси важливі для розробки і роботи веб-додатку.

2.2.1 HTML

HTML (HyperText Markup Language) - мова розмітки гіпертексту.

З назви стає зрозумілим його призначення - вказувати браузеру, як розмістити елементи на сторінці при її відображенні на екрані монітора. За допомогою HTML створюються Web-сторінки, які знаходяться в глобальній комп'ютерній мережі Інтернет. Те, що ви бачите при перегляді сторінки в Internet, це інтерпретація браузером HTML-тексту. HTML - це не мова програмування в традиційному сенсі, вона є мовою розмітки. HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.[12]

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

2.2.2 CSS

CSS (англ. Cascading Style Sheets) - каскадні таблиці стилів. Це спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

CSS - це мова для опису презентації веб-сторінок, включаючи кольори, макет і шрифти. Це дозволяє адаптувати презентацію до різних типів пристроїв, таких як великі екрани, маленькі екрани або принтери. CSS не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML. Відокремлення HTML від CSS полегшує підтримку сайтів, обмін таблицями стилів на сторінках та індивідуальність сторінок у різних середовищах.[12]

Переваги:

- інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
- різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;
- прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

2.2.3 JavaScript

Мова програмування JavaScript розроблений фірмою Netscape для створення інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки вбудованих додатків, що виконуються як на стороні клієнта, так і на стороні сервера. Синтаксис мови дуже схожий на синтаксис Java - тому його називають - Java-подібним.

Основні області застосування JavaScript поділяються на такі категорії:

- Динамічне створення документа за допомогою сценарію;

– Оперативна перевірка достовірності заповнених користувачем полів форм HTML до передачі їх на сервер;

– Створення динамічних HTML-сторінок спільно з каскадними таблицями стилів і об'єктної моделлю документа;

– Взаємодія з користувачем при вирішенні "локальних" завдань, що вирішуються додатком JavaScript, вбудованому в HTML-сторінку [13].

2.2.4 PHP

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення.[14]

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

Популярність мови PHP у побудові сайтів роз'яснюється роз'яснюється величезним набором вбудованих засобів, що допомагають у розробці.

Ось де-які найбільш значні з них:

- POST і GET параметри, а також інші змінні оточення автоматично вилучаються і заносяться в масиви;

- PHP здатний взаємодіяти з величезним числом систем для управління базами даних, такими як MySQL, PosgreSQL, Oracle, ODBC, Інтерфейс PDO та інші;

- HTTP-заголовки відправляються автоматично;

- можливість роботи з PHP-авторизацією;

- можливість роботи з сесіями та cookies;

- можливість роботи з сокетами та локальними файлами. Так само є можливість працювати з видаленими файлами;

- обробка файлів, завантажуваних на сервер;

- можливість роботи з XForms.

2.2.5 Локальний сервер Apache

Apache HTTP-сервер - відкритий веб-сервер. Apache є кросплатформним програмним забезпеченням, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS та інші операційні системи.[15]

Apache розробляється і підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apache Software Foundation.

В 1996 році Apache обійшов NCSA HTTPd з того часу є найпопулярнішим веб-сервером у світі. На початок червня 2013 року Apache встановлений на 53,34% (358 974 045 серверів) для порівняння на другому місці Microsoft IIS їхня частка 17,22% (115 920 681 серверів).

Тому, будучи безкоштовної відкритої програмою, Apache по функціональними можливостями і надійності не поступається комерційним серверам, а широкі можливості конфігурації дозволяють налаштувати його для роботи практично з будь-якої конкретної системою.

Отже, відразу після установки, доступний повністю працюючий веб-сервер Apache, що працює на локальному комп'ютері, на якому може працювати необмежена кількість сайтів, що дуже ефективно для розробки і налагодження сценаріїв PHP без завантаження його файлів на віддалений сервер.

2.2.6 База даних MySQL

MySQL - вільна система керування реляційними базами даних. Зараз MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL вважається гарним рішенням для малих та середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніші можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.[16]

Можливості сервера MySQL:

- простота установки та використання;
- підтримується необмежено кількість користувачів, які одночасно працюють з БД;
- кількість рядків у таблицях може досягати 50 мільйонів;

- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Для адміністрування MySQL будемо використовувати панель phpMyAdmin.

phpMyAdmin - це безкоштовний програмний інструмент, написаний на PHP, призначений для адміністрування MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій на MySQL та MariaDB. Часто використовувані операції (керування базами даних, таблицями, стовпцями, відносинами, індексами, користувачами, дозволами тощо) можна виконувати за допомогою інтерфейсу користувача, тоді як ви все ще маєте можливість безпосередньо виконувати будь-які SQL-інструкції.[17]

2.3 Технології парсингу

2.3.1 Загальна інформація щодо технології парсингу

Парсинг - це лінійне зіставлення послідовності слів з правилами мови. Поняття «мова» розглядається в найширшому контексті. Це може бути звичайна людська мова (наприклад, українською), яка використовується для комунікації людей. А може і формалізована мова, зокрема, будь-яка мова програмування.

Процес витягнення структурованої корисної інформації з сайту називається парсинг (parsing), а інструменти для реалізації даного процесу - парсерами (аналізатори).

Парсинг сайтів - послідовний синтаксичний аналіз інформації, яка розміщена на Інтернет-сторінках. Текст Інтернет-сторінок – це ієрархічний набір даних, структурований за допомогою людських і комп'ютерних мов. Людською мовою надана інформація, знання, заради яких, власне, люди і користуються Інтернетом, взагальному контент. Комп'ютерні мови (HTML, JavaScript, CSS) визначають як інформація виглядає на моніторі, розмітку сторінки, її стиль [7].

Парсери - це програми для автоматизованого збирання та структурування інформації з сайтів. Як правило, розробляються для кожного сайту окремо, з урахуванням його структурних та технічних особливостей [8].

Так само існують і готові рішення, що дозволяють витягати інформацію з сайту після попередньої конфігурації, не написавши ні єдиного рядка коду. Ці рішення часто стоять дорого і не володіють тією гнучкістю, яку можуть давати рішення, розроблені за конкретним сайтом.

Застосування парсинга може бути актуальним в наступних випадках:

- Якщо необхідно автоматично оновлювати сторінки вашого сайту, тобто

автоматично додавати статті, новини або інший контент;

— Для підтримки актуальності використовуваної на сайті інформації;

— Якщо необхідно об'єднати інформацію, так як вона часто перебуває на різних сайтах.

— Великі обсяги. В епоху бурхливого зростання мережі і жорстокої конкуренції вже всім ясно, що успішний веб-проект немислимий без розміщення великої кількості інформації на сайті. Сучасні темпи життя призводять до того, що контенту має бути не просто багато, а дуже багато, в кількостях, що набагато перевищують межі, можливі при ручному заповненні.

— Часте оновлення. Обслуговування величезного потоку динамічно мінливої інформації не в силах забезпечити одна людина або навіть злагоджена команда операторів. Часом інформація змінюється щохвилини і в ручному режимі оновлювати її навряд чи доцільно.

Також отримані дані можна використовувати самими різними способами, наприклад:

— маркетингові дослідження;

— моніторинг СМИ в реальному часі;

— аналіз громадського думки;

— автоматичне ціноутворення (при розумному застосуванні) на базі аналізу конкурентів цін;

— побудова списку потенційних користувачів на базі інформації про користувачів ресурсів конкурентів;

— створення API для сайтів без API.

Як правило, сайти розробляються з урахуванням того, що читання інформації з їх сторінок буде людина. Але формат представлених даних, зрозумілий для людини, частіше не такий зрозумілий програмним засобами. Крім того, структура представлених даних варіюється від сайту до сайту, тому не існує універсального засобу для вилучення інформації з них.

В процесі парсинга застосовуються скриптові мови програмування: PHP, Perl, Ruby, Python, JavaScript і багато інших. Він здійснюється шляхом написання так званого парсеру. Парсер - це програма, за допомогою якої відбувається автоматична обробка сторінок сайту для отримання необхідних даних. Парсер сайтів створюється для збору

великої бази контенту на сайт і для здійснення пошуку необхідної для користувачів інформації.

Останнім часом Інтернет став все більше поповнюватися новими інтернет-магазинами. Тому що на сьогоднішній день наявність власного інтернет-магазину є зручним і перспективним напрямком в бізнесі. Ви можете створити його і самостійно, але якщо ви не дуже розбираєтеся в програмуванні, то можна створити інтернет-магазин за допомогою парсинга. А наповнити ваш сайт потрібною інформацією допоможе парсер інтернет-магазинів. Він здатний парсити товари з необхідної вам категорії. Використовуючи парсер, можна самостійно створювати будь-які магазини.

Парсинг сайтів є ефективним рішенням для автоматизації збору і зміни інформації.

У порівнянні з людиною, комп'ютерна програма-парсер:

1. швидко обійде тисячі веб-сторінок;
2. акуратно відокремить технічну інформацію від «людської»;
3. безпомилково відбере потрібне і відкине зайве;
4. ефективно упакує кінцеві дані в необхідному вигляді.

Результат (будь то база даних чи електронна таблиця), звичайно ж, потребує подальшої обробки[7].

2.3.2 Методи вилучення інформації

Можна виділити наступні методи вилучення інформації з сайтів:

1. Ручний - в даному випадку роль парсера виконує людина. Він виробляє весь ланцюжок дій, необхідний для отримання потрібної інформації. Інформація збирається звичайним копіпастом.

2. Гибридный - користувач як і раніше виконує основні дії для отримання інформації, але може використовувати допоміжні програмні засоби для автоматизації збирання, наприклад, браузерний плагін, який на основі вилучення конфігурації вилучає і структурує інформацію з певних сторінок сторінки при активації.

3. Автоматичний - отримання та структурування інформації виконується автоматично[8].

Для того, щоб одразу отримати значення декількох полів з 10-15 сторінок, писати окремий парсер може бути і недоцільно, однак у випадку великого числа сторінок, або високої періодичності збору інформації, варто задуматися про автоматизацію даного процесу.

Процес витяг інформації з окремої веб-сторінки можна розбити на наступні етапи:

1. Побудова запиту для отримання інформації.
2. Виконання запиту та отримання відповіді.
3. Обробка відповіді, вилучення та структурування необхідної інформації.
4. Передача отриманої інформації для подальшої обробки.

Процес вилучення інформації може бути простим: завантажити URL-адресу, дізнатись інформацію та віддати одержувача; а може бути і складним: авторизватися в системі, сконструювати запит за інформацією з заголовків і значень JavaScript-змінених на сторінці, ім'я яких може змінюватися з кожним запитом.

2.3.3 Інструменти парсингу

Написання парсерів не вимагає монументальних знань про використовувану мову програмування будь то PHP, Ruby або Python. Також необов'язково мати академічні відомості про супутні технології. Однак, дещо доведеться вивчити. Перерахуємо веб-технології, які доведеться знати кожному, кого цікавить професійне створення синтаксичних аналізаторів:

— Щоб створити первинний алгоритм роботи майбутнього парсеру, доведеться проаналізувати вихідний код сторінок сайту-донора. Само собою, без знань (хоча б на середньому рівні) HTML, CSS і JavaScript ніяк не обійтися.

— Для більш глибокого занурення в тему рекомендується до вивчення технологія DOM, що дозволяє з максимальним ефектом працювати з ієрархічним деревом веб-документа.

— На найважливішому і складному етапі - написанні аналізатора - потрібно знання будь-якого з інструментів текстової обробки. Один з варіантів для пошуку потрібних шматків тексту скористатися регулярними виразами. Безумовно, «регулярки» користуються заслуженою славою як могутнього засобу для вирішення багатьох нетривіальних завдань. Однак цей спосіб не є найкращим, а часто і небажаним. По-перше, все-таки, регулярні вирази досить складно освоїти для подальшої роботи. Для цього потрібно володіти досить специфічним мисленням. По-друге, html-код на більшості сайтів не валідний, а часто і некоректний. Навіть найзапекліші фахівці можуть заплутатися в метасимволах і квантифікаторах, намагаючись передбачити всі випадки даної задачі. Тому, оптимальним виходом буде не винахід колеса, а використання готових бібліотек для парсинга, трохи нижче вказані найпопулярніші рішення для кожного з мов. Втім, це зовсім не означає що регулярні вирази можна і зовсім невивчати. Найчастіше саме за допомогою них зручно вирішувати багато проблем, які не взмозі вирішити конкретна

бібліотека для парсинга.

— Для ефективної роботи з ієрархічними структурами даних - бажано володіти парадигмою об'єктно-орієнтованого програмування. Семантичне дерево можна будувати і за допомогою багатовимірних асоціативних масивів, але, цей спосіб не логічний у випадку, коли дерево містить велику кількість вузлів, ООП відмінно підтримується всіма мовами розглянутими на сторінках цього сайту.

— Фінальна обробка результатів передбачає збереження даних в структурованому вигляді. Зазвичай на виході потрібна база даних. Так що знадобляться міцні знання SQL, і, як мінімум, MySql і PostgreSQL.

— Не обійтися без впевненого володіння функціями для роботи з файлами. Цілком можливо, дані доведеться дописувати в CSV-файли або конвертувати в електронні таблиці.

— Відомості про XML і XPath можуть знадобитися як на етапі синтаксичного аналізу (іноді доводиться парсити не сайти, а RSS-потоки), так і на стадії кінцевого збереження результатів (тобто, на основі сторонніх сайтів іноді необхідно створити все ту ж стрічку RSS).

— Іноді спарсені дані заливаються в нову базу даних за допомогою JSON. Дану javascript-технологію теж необхідно освоїти, нічого складного вона з себе не представляє[7].

2.4 Парсинг і обробка веб-сторінки на PHP: вибір бібліотек

Завдання спарсити і обробити необхідну інформацію зі стороннього сайту встає перед веб-розробником досить часто і з найрізноманітніших причин: таким чином можна заповнювати свій проект контентом, динамічно довантажувати якусь інформацію і так далі [9].

У таких випадках перед програмістом постає питання: яку з десятків бібліотек вибрати? Тож ми розглянемо найпопулярніші варіанти і виберемо з них найкращий.

2.4.1 Регулярні вирази

Навіть не дивлячись на те, що регулярні вирази - це перше, що спадає на думку, використовувати їх для справжніх проектів не варто[9].

Так, з простими завданнями регулярні вирази справляються краще за всіх, але його

використання значно ускладнюється, коли потрібно спарсити велику і складну частку HTML-коду, яка, до того ж, не завжди відповідає якомусь певному шаблону і взагалі може містити синтаксичні помилки.

Замість «допилювання» свого регулярного виразу при кожному найменшому зміні коду рекомендуємо використовувати інші інструменти - це і простіше, і зручніше, і надійніше.

2.4.2 XPath і DOM

DOM і XPath не є бібліотеками в звичному сенсі цього слова, це стандартні модулі, які вбудовані в PHP починаючи з п'ятої версії. Саме відсутність необхідності використовувати сторонні рішення робить їх одними з кращих інструментів для парсинга HTML сторінок[9].

На перший погляд може здатися, що низький поріг входу - це не про них, деякі місця і справді є дуже складними. Але це тільки на перший погляд: варто тільки трохи розібратися з синтаксисом і базовими принципами, як XPath тут же стане для вас інструментом для парсинга номер один.

Ось, наприклад, код з використанням DOM і XPath, який шукає в розмітці все теги і модифікує їх атрибути src:

```
$dom = new DOMDocument;
$dom->loadHTML($html);
$images = $dom->getElementsByTagName('img');
foreach ($images as $image) {
        $image->setAttribute('src', 'http://example.com/' .
$image->getAttribute('src'));
}
$html = $dom->saveHTML();
```

Проте, даний варіант не позбавлений мінусів - для парсинга використовується движок, в першу чергу призначений для роботи з XML, а XML і HTML хоч і є дуже схожими мовами, але все-таки різняться. З цього випливають специфічні вимоги до розмітки: наприклад, всі HTML теги повинні бути закриті.

2.4.3 Simple HTML DOM

Simple HTML DOM - PHP-бібліотека, що дозволяє довільні HTML-код за допомогою зручних jQuery-подібних селектор[9].

Вона позбавлена головного недоліку XPath - бібліотека вміє працювати навіть з невалідним HTML-кодом, що значно спрощує роботу. Ви також забудете про проблеми з кодуванням: всі перетворення виконуються автоматично.

Як і JQuery, Simple HTML DOM вміє шукати і фільтрувати вкладені елементи, звертатися до їх атрибутам і навіть вибирати окремі логічні елементи коду, наприклад, коментарі.

У цьому прикладі спочатку завантажується, а потім модифікується заздалегідь заготовлений HTML-код: у другому рядку відбувається додавання атрибута class зі значенням bar першому ліпшому елементу div, а в наступному рядку ми замінюємо текст елемента з id = "world" на foo.

```
$html = str_get_html('<div id="hello">Hello</div><div
id="world">World</div>');
$html->find('div', 1)->class = 'bar';
$html->find('div[id=world]', 0)->innertext = 'foo';
echo $html;
```

Незважаючи на не найвищу продуктивність, в порівнянні з іншими варіантами, Simple HTML DOM має найбільшу поширеність в інтернеті - для новачків це робить написання коду з її використанням значно простіше.

2.4.4 phpQuery

Як і Simple HTML DOM, phpQuery є PHP варіантом JQuery, але на цей раз більш схожим на свого «старшого javascript-брата»[9].

Перенесено майже все, що є в JS-фреймворку: підтримка селектор, атрибутів, маніпуляцій, обходу, плагінів, подій (в тому числі імітації кліків і т.д.) і навіть AJAX. Використовувати можна як через PHP, так і через командний рядок у вигляді окремого додатка.

Більш того, згідно з нашим бенчмарк, phpQuery виявився в 8 (!) Разів швидше Simple HTML DOM.

Ось невеликий приклад на phpQuery, в якому відбувається обробка заздалегідь вибраних елементів списку (li):

```
foreach(pq('li') as $li) {
        // Можно вывести различные данные обычным текстом
        $tagName = $li->tagName;
        $childNodes = $li->childNodes;
        // А можно добавить обертку phpQuery (аналог $() в
```

```
JQuery) и, например, добавить к элементу какой-то класс
            pq($li)->addClass('my-second-new-class');
    }
```

2.4.5 cURL

cURL (скор. від client URL) - розроблена в 1998 році Деніелом Стейнбергом ↑ кроссплатформенная утиліта для автоматизації передачі віддалених файлів з використанням різних протоколів (http, https, ftp, gopher, telnet, dict, file і ldap)[7].

Для вбудовування cURL в різні мови програмування розроблена бібліотека libcurl. Починаючи з версії 4.0.2 бібліотека за замовчуванням включена в конфігурацію PHP.

Крім підтримки основних мережевих протоколів та сертифікатів, cURL забезпечує туннелирование через http-проксі, працює з http-куками, успішно взаємодіє з різноманітними системами аутентифікації, відновлює докачку в разі обриву.

Функції cURL:

resource curl_init ([string url])

Ініціалізація cURL-сесії. При цьому функція поверне cURL-дескриптор, який потім використовується при викликах функції curl_setopt (), curl_exec () і curl_close (). Необов'язковий параметр url встановлює опцію CURLOPT_URL, що позначає адресу, з яким будуть проводитися подальші операції. Якщо це параметр не буде встановлено, то його в подальшому можна визначити за допомогою функції curl_setopt ().

bool curl_setopt (resource ch, string option, mixed value)

Попередньо встановлено опції для cURL-сесії. Зазвичай функція викликається кілька разів, встановлюючи різні параметри для сеансу. Параметр ch є ідентифікатором сесії (cURL-дескриптор), виставлений функцією curl_init (). Параметр option є назвою опції яку потрібно встановити, а value - її значенням.

bool curl_exec (resource ch)

Запуск cURL-сесії на виконання. При цьому будуть використані параметри, встановлені за допомогою викликів функції curl_setopt (). ch - cURL-дескриптор поточної сесії, отриманий в результаті виклику функції curl_init (). У разі невдачі, функція повертає false.

void curl_close (resource ch)

Закриття cURL-сесії. Дескриптор ch при цьому знищується, ресурси виділені на сесію звільняються.

Треба відзначити, функції тут не все. Перераховано тільки ті, які обов'язково беруть участь в будь-який cURL-сесії.

Приклад:

```
<?php
//Инициализируем cURL-сессию
$ch = curl_init ("http://www.yandex.ru/");
//открываем для записи файл yandex.txt
//сюда внесём исходный html-код
$fp = fopen ("yandex.txt", "w");
 //указываем куда копировать содержимое
curl_setopt ($ch, CURLOPT_FILE, $fp);
 //Заголовок - не выводим
curl_setopt ($ch, CURLOPT_HEADER, 0);
 //Поехали!
curl_exec ($ch);
 //Закрываем cURL-сессию
curl_close ($ch);
 //Закрываем дексриптор файла
fclose ($fp);
 //И вставляем на страницу полученный код
include 'yandex.txt';
?>
```

Висновок: У своєму міні-дослідженні ми дійшли висновку, що в більшості випадків для парсинга краще використовувати бібліотеку phpQuery: вона швидка, функціональна і сучасна.

З іншого боку, для зовсім простих завдань можливо використовувати стандартні модулі PHP, такі як XPath, Simple HTML DOM або регулярні вирази.

2.5 Етапи парсинга

Парсинг html-сторінки вдає із себе процес, який можна розбити на три етапи[7]:

1. Отримання початкового коду веб-сторінки - скачати програмний код тієї сторінки сайту, з якої необхідно витягти інформацію. У різних мовах для цього передбачені різні способи, наприклад, в PHP найчастіше використовують бібліотеку cURL або ж вбудовану функцію file_get_contents.

2. Витяг з html-коду необхідних даних. Отримавши сторінку, необхідно обробити її - відокремити звичайний текст від гіпертекстової розмітки, вистроїти ієрархічне дерево

елементів документа, коректно зреагувати на невалідний код, виокремити зі сторінки саме ту інформацію, заради якої і відбувається весь цей процес. Можна, звичайно ж, використовувати для цього регулярні вирази, проте є більш зручний шлях - спеціалізовані бібліотеки.

3. Фіксація результату. Благополучно обробивши дані на сторінці, потрібно їх зберегти в необхідному вигляді для подальшої обробки. Спарсені дані зазвичай заносяться в базу даних, однак є й інші варіанти. Іноді потрібно записати в CSV-файл або будувати ієрархічні JSON-структури, іноді конвертувати в excel-таблицю, а може навіть згенерувати динамічний rss-потік.

Як правило, потрібно спарсити не одну сторінку сайту-донора, а декілька, а може навіть і всі. У цьому випадку після проходження кроків 1-3 у алгоритм парсера повинен бути закладений перехід на наступну сторінку сайту, щоб і з неї вилучити весь необхідний матеріал. Обхід всіх потрібних сторінок сайту забезпечується різними способами.

– По-перше, обробляючи чергову сторінку, парсер можна навчити не тільки отримувати необхідні дані, але і заносити в свою базу даних всі внутрішні посилання, що зустрічаються на шляху. Звертаючись до свого сховища лінків, програма послідовно відвідує сторінки сайту, до тих пір поки не обійде їх всіх.

– По-друге, при первинному аналізі сайту найчастіше можна простежити логіку формування url для сторінок. І потім, генерувати адреси відповідно до виявлених закономірностей.

– По-третє, деякі парсери розраховані, як не дивно, на «ручний» обхід веб-ресурсу. Користувач, клікаючи по посиланнях, сам вирішує які сторінки відвідувати, які ні. А программа у фоновому режимі запам'ятовує необхідні дані.

Зрозуміло, поєднувати різні методи ніщо не заважає.

2.6 Синтаксичний аналіз

Отже, отримавши вихідний код сторінки, можна починати її аналіз, відокремлюючи ту інформацію, заради якої парсився сайт[7].

Цілком резонне питання - який інструмент вибрати для обробки? У минулі часи у програмістів не було особливого вибору крім як вдаватися до аналізу сторінки за допомогою регулярних виразів. Але, враховуючи їхню складність і опираючись на те, що на даний момент є набагато простіші шляхи вирішення даного питання, детально розглядати і реалізовувати цей метод не будемо.

Неприємною реальністю в використанні регулярних виразів є напівкоректний html-

код більшості сайтів. Хоча веб-браузери в більшості випадків відображають все вірно, небагато веб-ресурсів притримуються 100% відповідності стандартам W3C.

В принципі, можна самостійно побудувати дерево документа і потім працювати з ним за допомогою технології Document Object Model. Після цього буде простіше аналізувати некоректний html-кодом. Однак, часу і сил на написання власного інтерпретатора сторінок піде чимало.

Згодом з'явилися чудові безкоштовні рішення, покликані полегшити життя кодерам. Написання парсерів спростилося завдяки спеціалізованим бібліотекам для парсинга. Більше немає потреби у різноманітних текстових масках - все вже давно зроблено за нас.

2.7 Збереження даних

Матеріал, отриманий з розпарсенного сайту, необхідно упакувати у вигляді, придатному для подальшого використання. Конкретний формат залежить від того як в подальшому буде оброблятися зібрана інформація. Найчастіше це бази даних MySQL або PostgreSQL. Заливати в БД можна не тільки за допомогою запитів SQL, але і за допомогою JSON через Ajax. У багатьох випадках із спарсенного контенту за допомогою XML формується RSS-потік, що вельми зручно при використанні даних, без процедури рерайтинга. Іноді результат парсинга поміщають в CSV-файл - оскільки цей текстовий формат дуже простий у подальшій обробці, легко конвертується в SQL-запити і без проблем відкривається в Excel. У спеціальних випадках потрібно, щоб кінцеві дані були представлені у вигляді електронних таблиць XLS[7].

2.7.1 SQL

При парсингу кожної сторінки сайту-донора, витягнуті дані, як правило, тут же заносяться в базу даних. Для повної реалізації даної задачі нам не обійтися без впевнених знань про MySQL найбільш використованої СУБД для веб-розробок[7].

Зазвичай у вигляді БД спарсенний контент і потрібний для перенесення на інший веб-ресурс. Однак часто наповненням бази даних справа не закінчується. Інформацію з SQL доводиться конвертувати в інші формати: JSON, текстовий CSV (який, в свою чергу, є зручним буфером для подальшої обробки), таблично-процесорний XLS (в PHP є спеціальні додаткові бібліотеки для перетворення SQL-даних в електронні таблиці Excel), XML (веб-серверний RSS-агрегатор).

2.7.2 CSV

CSV (англ. Comma-Separated Values - значення через кому) - формат текстових файлів для зберігання табличної інформації. Один рядок - один запис в таблиці. Роздільник значення полів у записі - кома. Втім, замість коми можна використовувати і інші символи[7].

Формат CSV популярний оскільки дуже простий і універсальний. Його можна легко: правити в будь-якому текстовому редакторі; відкривати в Microsoft Excel і OpenOffice.org Calc; змінювати функціями PHP для роботи з файлами; конвертувати в SQL і назад.

При парсингу досить-таки часто у Вас можуть запитати спарсені дані в цьому форматі. Що не викликає ні найменших труднощів і дуже зручно для подальшої обробки інформації. Приклад CSV-файлу(рис. 2.2).

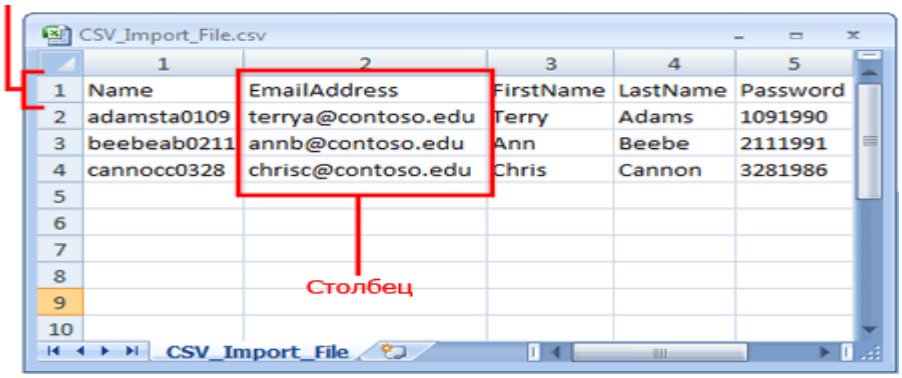


Рисунок 2.2 - Структура CSV-файлу

2.7.3 XLS

Іноді спарсенний матеріал потрібно представити у вигляді електронної таблиці Excel. Безумовно, електронні таблиці неможливо редагувати як звичайний текстовий файл. Досить відкрити документ XLS в текстовому редакторі (Notepad ++, Sublime Text, наприклад) і виявити там набір незрозумілих символів, щоб прийти до висновку про марність стандартних функцій для роботи з файлами. Просто дописати в кінець файлу чергові дані (як у випадку з CSV), витягнуті з сайту-донора, не вийде[7].

Що ж, якщо стандартні функції не підходять, то на зміну їм використовують нестандартні. Існує чимало бібліотек, покликаних полегшити роботу з книгами Excel. Підключивши їх, можна як зчитувати інформацію з готових електронних таблиць, так і засобами мов веб-програмування формувати XLS-документи. Додавати нові записи в таблицю можна безпосередньо відразу після вилучення контенту з сайту-донора. Або ж спарсити матеріал в базу даних SQL, а потім з БД сформувати електронну таблицю.

2.7.4 JSON

У міру обходу сторінок сайту-донора часто зібрана інформація тут же за допомогою запитів SQL заноситься в базу даних. Альтернативою є JSON - текстовий формат обміну даними оснований на JavaScript[7].

У даного методу чимало безперечних переваг перед тим же SQL. JSON відрізняють мовонезалежність, крайня простота і лаконічність синтаксису, наявність готових рішень для обробки даних. Але головна перевага - це можливість прямої взаємодії браузера і сервера, що дозволяє реалізовувати алгоритми пов'язані з фоновим занесенням інформації в базу даних, роботою з файловим кешем, серіалізацією, десеріалізацією динамічних структур та ін.

2.7.5 Document Object Model

DOM (Document Object Model) - багатоплатформовий багатомовний програмний інтерфейс, що надає додаткам і скриптам доступ (в тому числі і на зміну) до структури та змісту HTML, XHTML, XML-документів[7].

Document Object Model дозволяє ефективно вибудувати ієрархічне дерево веб-документа (рис. 2.3) для подальшої роботи з ним.

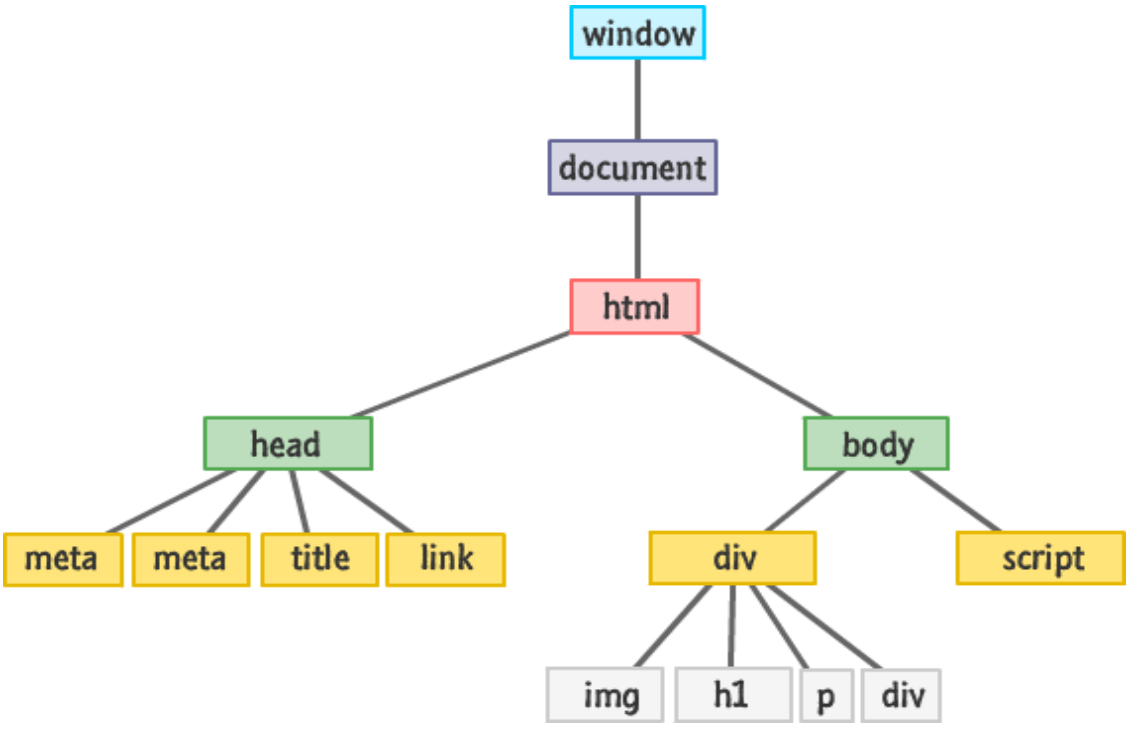


Рисунок 2.3 – Ієрархічне дерево веб-документа

Велика частина бібліотек для парсинга основані на DOM.

DOM є нащадком Dynamic HTML Object Model - об'єктною моделлю веб-документів, розробленої для Netscape Navigator і Internet Explore 4 в 1997 році. Новинка дозволяла за допомогою JavaScript на стороні клієнта змінювати HTML-структуру документа і маніпулювати каскадних таблицями стилів.

Обидва браузера з часом перестали самостійно розвивати Dynamic HTML Object Model, яка зараз вважається морально застарілою технологією. Netscape згодом взагалі припинив існування, IE підтримує дану технологію з метою забезпечення сумісності.

Поточною версією є DOM2, її взяли на озброєння всі браузерні движки. DOM3 поки що має експериментальний (рекомендований) статус і навіть основні браузери підтримують його не в повній мірі. В PHP5, Ruby1.9.x і Python3 також DOM3 реалізований лише частково.

2.8 Захист від парсингу та методи його обходу

Для власників сайтів парсери - небажані гості. Вони даремно витрачають обчислювальні ресурси сервера, трафік, за який можливо потрібно платити, а погано спроектовані парсери можуть обрушити величезне число запитів на сервер. Витягнуті дані можуть використовуватися в корисливих цілях, на зразок крадіжки матеріалів з одного ресурсу і публікації на іншому від чужого імені, порушуючи авторство[8].

У зв'язку з цим розробники можуть застосовувати різні заходи боротьби з автоматичними парсером, а саме:

- динамічна зміна структури коду сторінки з метою ускладнення добування інформації з блоків;
- показ капчи при авторизації / реєстрації / перевищенні числа запитів за одиницю часу;
- блокування клієнтів за середнім обсягом трафіку в одиницю часу;
- аналіз поведінки клієнтів і блокування / вимога пройти капчу для продовження роботи при підозрілу поведінку.

Ці проблеми можна вирішити:

- якщо інформація дуже важлива, то можна вдатися до screen scraping'у - вилучення інформації не з тексту сторінки, а з її зображення;
- розпізнавання капчи можна передати стороннім сервісам;
- розтягнути процес парсинга в часі;

– імітувати поведінку реального користувача: кліки, руху миші, гортання сторінки.

В теорії все досить просто, але на ділі можуть потрапляти комбіновані прийоми, що роблять розробку парсеру економічно невигідною. Тому дуже важливо вчасно виявляти подібні випадки, щоб не витратити багато часу даремно.

Висновки по розділу 2

У розділі було розглянуто основні особливості парсингу веб-сайтів та проблеми які виникають. Визначено актуальність задачі, роль програми-парсера. Досліджено інструментарій за допомогою якого можна реалізувати дане завдання, а також проаналізовано варіанти синтаксичного розбору веб- сторінок, імпорту та експорту даних. Коротко ознайомились зі способами захисту від парсингу та методами їх обходу.

3 РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 База даних

База даних є надзвичайно важливим елементом будь-якої інформаційній системи. В якості сервера бази даних будемо використовувати систему управління базами даних-MySQL та веб-інтерфейс(панель управління) для адміністрування СУБД MySQL – phpMyAdmin(рис. 3.1).

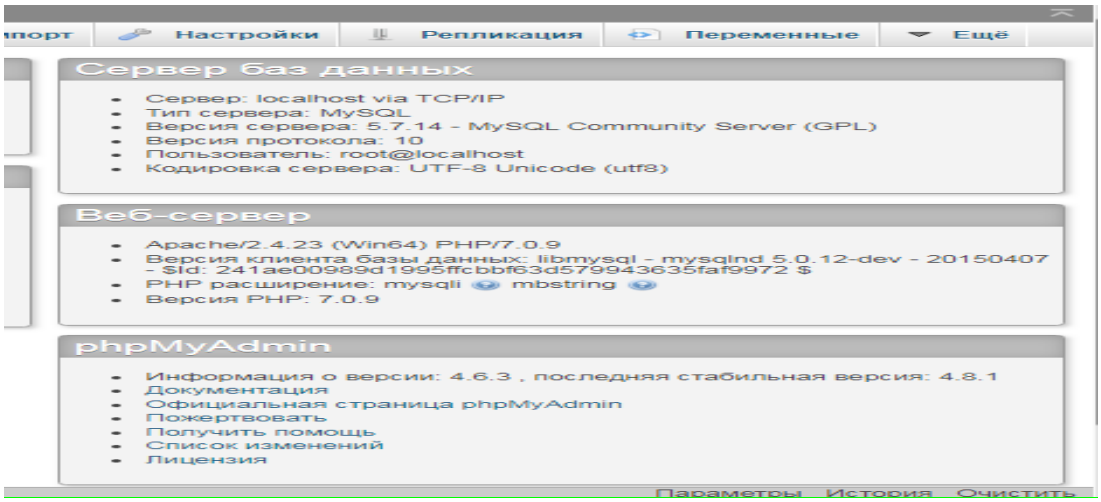


Рисунок 3.1 – Інформація про сервер баз даних, веб-сервер та phpMyAdmin

Так як система буде здійснювати пошук і видачу інформації про музеї та фортеці залежно від області, тому будуть створені бази даних із фортецями по областях(рис. 3.2).

При вивченні предметної області були виділені основні сутності та зв’язки між ними(рис 3.3).

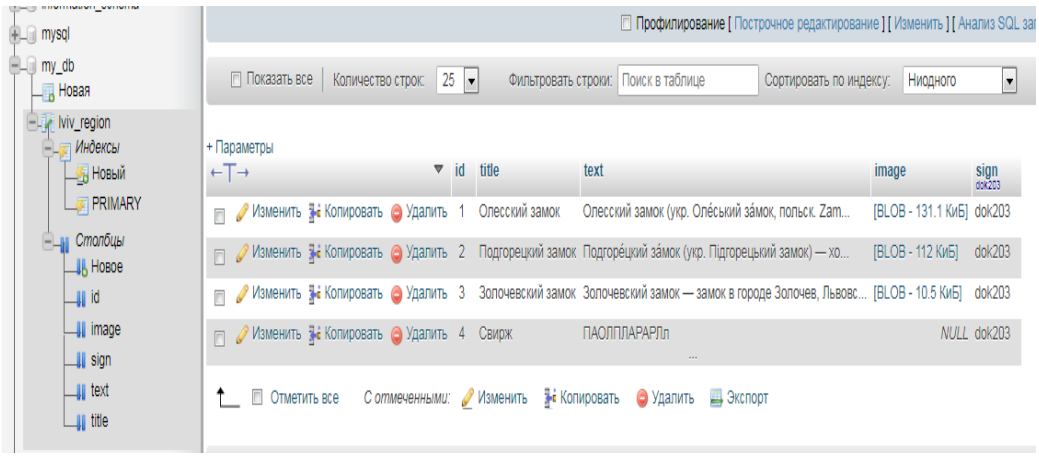


Рисунок 3.2 – Пример бази даних Львівскої області

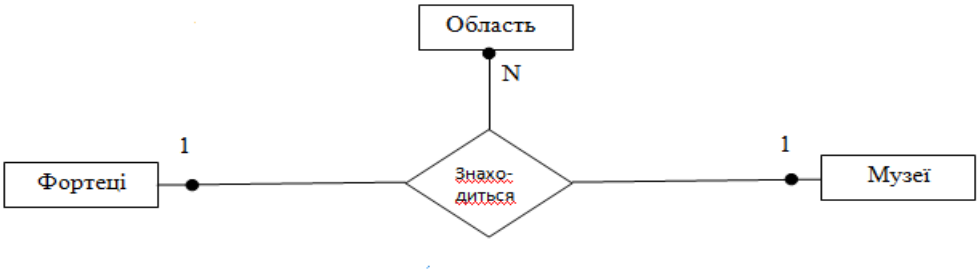


Рисунок 3.3 – Логічна модель даних(ER - діаграма)

Зв’язок Область – Фортеці: конкретна фортеця одна, а в області може бути багато фортець. Отже зв’язок між цими сутностями буде один до багатьох(1:N).

Зв’язок Область – Музеї: конкретний музей один, а в області може бути багато музеїв. Отже зв’язок між цими сутностями буде один до багатьох(1:N).

3.1.1 Доступ до бази даних

Щоб з'єднатися з базою даних застосовуємо функцію mysqli_connect(). Вона приймає всі конфігураційні налаштування (адрес сервера, ім’я бази даних, ім’я користувача) і підключається до сервера. У разі помилки підключення спрацьовує оператор die(), який виводить повідомлення про помилку і завершує роботу скрипта. А в разі успішного підключення функція mysqli_connect() повертає об'єкт підключення у вигляді змінної $connection.

Після закінчення роботи підключення потрібно закрити. Для цього застосовується функція mysqli_close(), яка в якості параметра приймає об'єкт підключення.

Щоб здійснити запит до бази даних, треба використати функцію mysqli_query(), яка приймає два параметри: об'єкт підключення($connection) і рядок запиту на мові SQL.

Функція mysqli_query() повертає об'єкт $result, який містить результат запиту. У разі невдачі даний об'єкт містить значення false[18] (рис 3.4).

Gallery_lviv.php | style1.css | parser_pidgirzi.php | parser_olesko.php | query_db.php

```php
<?php
$connection = @mysqli_connect('localhost','root','','my_db')or die("error" .mysqli_error($connection));
mysqli_set_charset($connection,'utf8' );

$result = mysqli_query($connection,"SELECT * FROM lviv_region") or die("error" .mysqli_error($connection));

mysqli_close($connection);
 $array = mysqli_fetch_array($result)
?>
```

Рисунок 3.4 – Скрипт підключення до бази даних

3.2 Інтерфейс додатку

Згідно вимог до інтерфейсу, які були описані вище (розділ 1.2.2) було вирішено спроектувати головну сторінку додатку с краткою загальною інформацією та інтерактивною мапою.

Також система повинна володіти типовими сторінками, які будуть мати спільні: «шапку»(header), навігацію(navigation), «підвал»(footer) і будуть відрізнятися тільки контентом. Тому будуть розроблені окремі файли .php, які за допомогою скрипта зможуть підключатися до будь-якої сторінки(рис. 3.5).

```
<head>
 <meta charset="utf-8">
<title>Замки Украины</title>
<link rel="stylesheet" type="text/css" href="css/style1.css">
<link rel="stylesheet" type="text/css" href="css/style_gallery.css">
</head>
<body>
 <!-- MAIN -->
<?php
 require "header.php"
?>
   <!-- NAVIGATION -->
<?php
 require "navigation.php"
?>
 <!-- /NAVIGATION -->

 <div class="intmap"><p><h1>ГАЛЛЕРЕЯ ЗАМКОВ ЛЬВОВЩИНЫ </h1> </p>
```

Рисунок 3.5 – Фрагмент коду з підключенням файлів header.php та navigation.php

CSS3 дозволяє створювати деяку анімацію, але скрипти JavaScript дозволяють зробити дещо складніше. JavaScript допомагає придати сторінці інтерактивності та динамічності. Тому для створення інтерактивної мапи на головній сторінці будемо використовувати CSS3, JavaScript та векторний графічний редактор Inkscape.

В графічний редакторі зробимо контур області, відредагуємо та через редактор XML отримаємо координати та вставимо до коду(рис. 3.6 – 3.7).

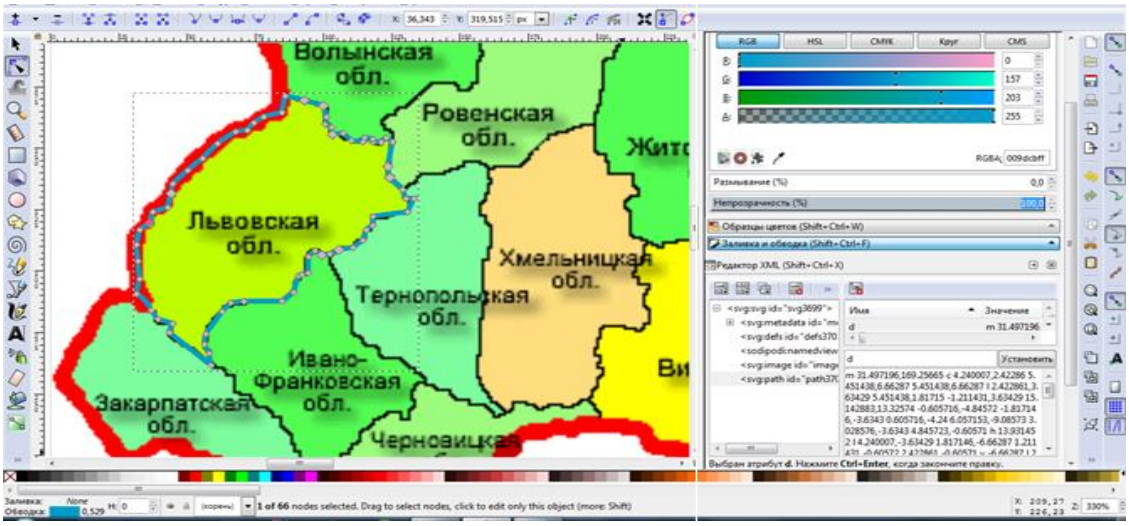


Рисунок 3.6 – Робота в графічному редакторі

```
<svg viewBox="20 20 685 460" xmlns="http://www.w3.org/2000/svg" version="1.1" xmlns:xlink="
http://www.w3.org/1999/xlink">
<a href="Gallery_lviv.php">
   <path  data-id="lv" class="region" id="lviv"  d="m 84.300848,88.199152 c
   7.771041,1.573747 12.930464,3.476997 16.567792,4.872882 l 0.11274,6.334745
   2.3237,4.872881 4.3856,1.50005 4.98561,1.49823 4.72197,-0.86185 3.86194,1.94915
   4.31105,-2.43644 c -1.18127,3.2204 -0.69504,7.72455 -1.80006,10.53306 0.007,2.44171
   -0.0613,4.95968 1.50005,5.84746 2.02294,2.31981 6.03178,3.44807 4.42195,7.94753
   1.39832,0.20886 2.54581,1.04478 2.36189,5.20925 l -5.922,1.42368 c -2.14023,2.23516
   -4.78304,2.46004 -7.72206,1.50005 l -1.94916,5.36017 c -0.62697,1.69413 -3.95927,3.24089
   -4.38559,6.33475 -2.37264,0.42349 -4.27426,0.37595 -5.84746,0 l -2.21097,2.99827
   -4.423783,0.48729 -2.623718,-2.811 c -3.184988,0.27373 -3.889432,4.88841
   -3.898305,10.72034 l -3.636483,3.41102 c 1.083337,3.02373 -1.500315,5.74745
   -5.434714,7.27113 -5.097736,-0.23433 -10.195473,-1.28678 -15.29321,2.28735
   -4.533599,-2.64591 -7.877337,4.22706 -11.694915,7.30932 0.36285,2.73019 0.590507,5.50158
   -1.987341,7.4966 2.499912,2.43644 3.134595,4.87289 2.43644,7.30933 l -6.296557,-5.06016
   -7.309322,-4.87288 1.461865,-5.36017 -6.334746,-0.48729 -0.749111,-4.27286 c
   -8.260859,-5.41957 -5.704072,-13.24306 -7.534787,-20.09155 -0.291488,-8.22828
   1.323124,-13.27973 3.411017,-17.54237 6.067914,-2.51762 8.354108,-9.25909
   13.231323,-15.55504 3.458822,-6.96942 8.557989,-11.47833 13.569524,-16.11869 l
   14.618644,-7.796611 c 7.386756,-2.924416 10.184372,-6.683222 8.771187,-11.207627 z"
   description="<img src='image/blazons/lv.png'>Львовская область и её замки: Злочевский,
   Олеско, Подгорцы, Свирж, Жолква...">
</path>
</a>
```

Рисунок 3.7 – Фрагмент коду з координатами з графічного редактора

Далі за допомогою CSS3 та JavaScript підсвітимо область та виведемо інформаційне вікно(рис. 3.8 – 3.9).

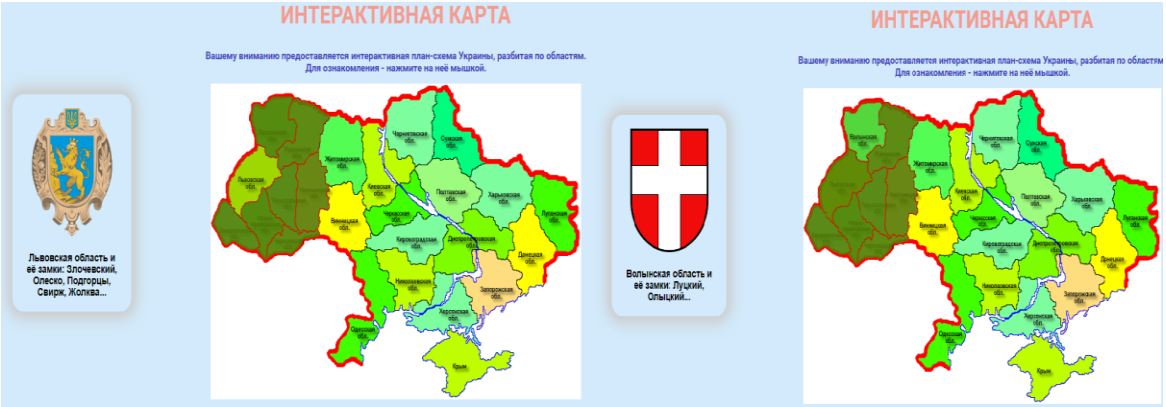


Рисунок 3.8 – Реалізація інтерактивної карти

```
$('.region').hover (
    function() {
        $('.description').html($(this).attr('description'));
        $('.description').fadeIn();
    },
    function() {
        $('.description').fadeOut(50);
    }
)
```

Рисунок 3.9 – Скрипт до інтерактивної карти

3.3 Реалізація переліку фортець

Так як сторінки з переліком фортець відносяться до типових то їхній інтерфейс буде однаковим. Кожна сторінка матиме «шапку»(header), навігацію(navigation), «підвал»(footer) і буде відрізнятися лише наповненням.

Перелік фортець матиме форму таблиці з назвою і фото фортеці, а також мінімальною інформацією про фортецю. Для більш детального ознайомлення необхідно перейти на сторінку фортеці за посиланням(Ознайомитися детальніше…)(рис. 3.10).

Рисунок 3.10 – Перелік фортець Львівщини та Волині(фрагмент)

3.4 Збір та акуталізація інформації з інших сайтів за допомогою парсинга

3.4.1 Мова програмування PHP для створення парсера

Більшість парсерів пишуться на PHP [7]. Причиною є деякі особливості мови:

- низький поріг входження;
- «заточена» під роботу з Web;
- наявність добре опрацьованих тематичних бібліотек.

Саме на цій мові написано переважна кількість сайтів, а значить парсери на PHP найпростіше вбудовувати в їх архітектуру.

Також дя роботи парсера потрібен локальний сервер – спеціальна програма, яка дозволяє протестувати сайт чи веб-додаток на домашньому комп'ютері без виходу в Інтернет. Це дуже важливо для динамічних сайтів з php-скриптами. Для звичайних html+css сайтів сервер не потрібен.

3.4.2 Завантаження веб-сторінок

Щоб проаналізувати веб-сторінки, нам спочатку потрібно завантажити їх, необхідно отримати її html-код. Будемо використовувати бібліотеку SimpleHTMLDOM, для цього до папки проекту потрібно завантажити файл simple_html_dom.php, це і є вся бібліотека.

Для того, щоб бібліотека працювала у скрипті потрібно її підключити(рис. 3.11).

```
require_once 'simple_html_dom.php'//підключення бібліотеки
```

Після підключення бібліотеки необхідно завантажити дані з віддаленого URL або з локального файлу і для цього скористаємося функцією file_get_html в результаті чого отримаємо об'єкт типу simple_html_dom.

Після того, як html текст упакований в об'єкт, можна приступати безпосередньо до пошуку потрібних елементів.

При завантаженні великого обсягу даних відбувається витік пам'яті в бібліотеці, тому потрібно після закінчення одного циклу її чистити. Робить це метод clear(рис. 3.11).

Gallery_lviv.php | style1.css | parser_pidgirzi.php | parser_olesko.php

```
1  <?php
2  require_once 'header.php';
3  require_once 'navigation.php';
4  require_once 'simple_html_dom.php';//підключення
   бібліотеки
5
6  $html = file_get_html('https://ua.igotoworld.com/ua/
       poi_object/2392_pidhoretskyi-castle.htm');
7  if ($html->innertext!='' and count($html->find('p')))
       {
8  $html->clear();
9  unset($html);
10 ?>
```

Рисунок 3.11 – Скрипт отримання даних з url-адреси

3.4.3 Аналіз та обробка даних

Після завантаження і отримання даних з url-адреси треба проаналізувати структуру веб-сторінки, а також знайти потрібні елементи. Отже необхідно розібрати сторінку на складові і вилучити зі сторінок необхідну нам інформацію. Для цього розглянемо її вихідний код за допомогою браузера (рис. 3.12).

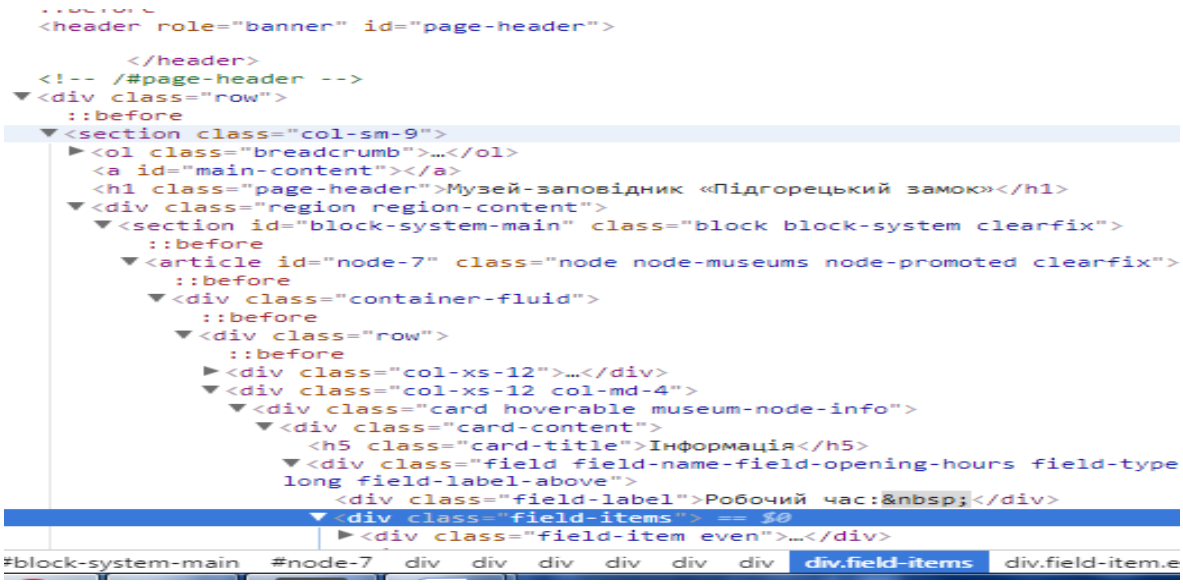


Рисунок 3.12 – Структура HTML-документа.

Поглянувши на структуру HTML-документа, стає зрозумілим, що розмітка в HTML складається з чотирьох основних компонентів: елементів – «тегів» (та їхніх атрибутів), базових типів даних(контенту), символьних мнемонік та декларації типу документа.

Проаналізувавши HTML сторінки були виділені основні блоки, що містять корисну інформацію. Тепер необхідно витягти зі сторінок необхідну інформацію(рис. 3.13).

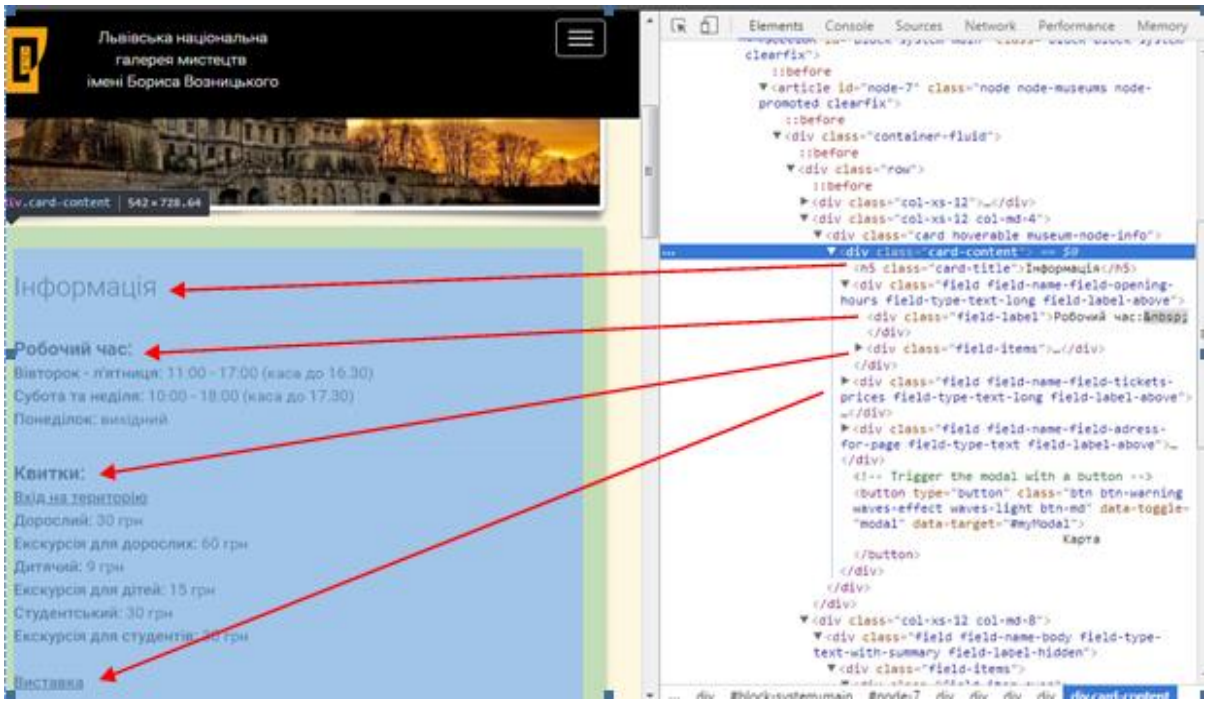


Рисунок 3.13 – Структура HTML-документа(з частиною web-сайту).

Більшість пошукових функцій виконує метод find (selector, [index]). Якщо другий аргумент не заданий, метод повертає масив елементів, а якщо ж заданий, то елемент цього масиву з індексом index.

Таким чином, дані які треба парсити це загальна інформації про фортецю,час роботи, ціна квитків, місце розташування тощо.

Розібравши html-код ми зрозуміли, що потрібна нам інформація міститься у тегу де class=’field-items’(рис. 3.14).

```
Gallery_lviv.php | style1.css | parser_pidgirzi.php | parser_olesko.php

18
19  $html = file_get_html('http://lvivgallery.org.ua/museums/
        muzey-zapovidnyk-oleskyy-zamok');
20  if ($html->innertext!='' and count($html->find('p'))){
21
22  ?>
23  <p  style="text-align: left;padding: 20px 50px 0 50px;margin: 0
        20px;"><h5 style="text-align: left;padding: 20px 50px 0 50
        px;margin: 0 20px;">Інформація належить сайту: http://
    lvivgallery.org.ua</h5><h3 style="text-align: justify;padding:
        20px 50px 0 50px;margin: 0 20px;">Додаткова інформація:</h3
    > </p>
24  <?php }
25  foreach ($html->find('div.field-items') as $div) {
26   ?>
27            <div style="width:80%;text-align: justify;padding: 0 50
                px 0px 10px;margin: 0 20px;/*list-style: none;*/">
28              <?php echo  $div->innertext ?></div>
29  <?php }
30
31
32  $html->clear();
33  unset($html);
34
35  require_once 'footer.php';
36  ?>
```

Рисунок 3.14 – частина PHP-скрипту для парсингу сайту.

Витягнуті з веб-сайту дані після збереження та перетворення за допомогою таблиць стилів - були виведені на потрібну нам сторінку нашої системи зі збору інформації(рис. 3.15).

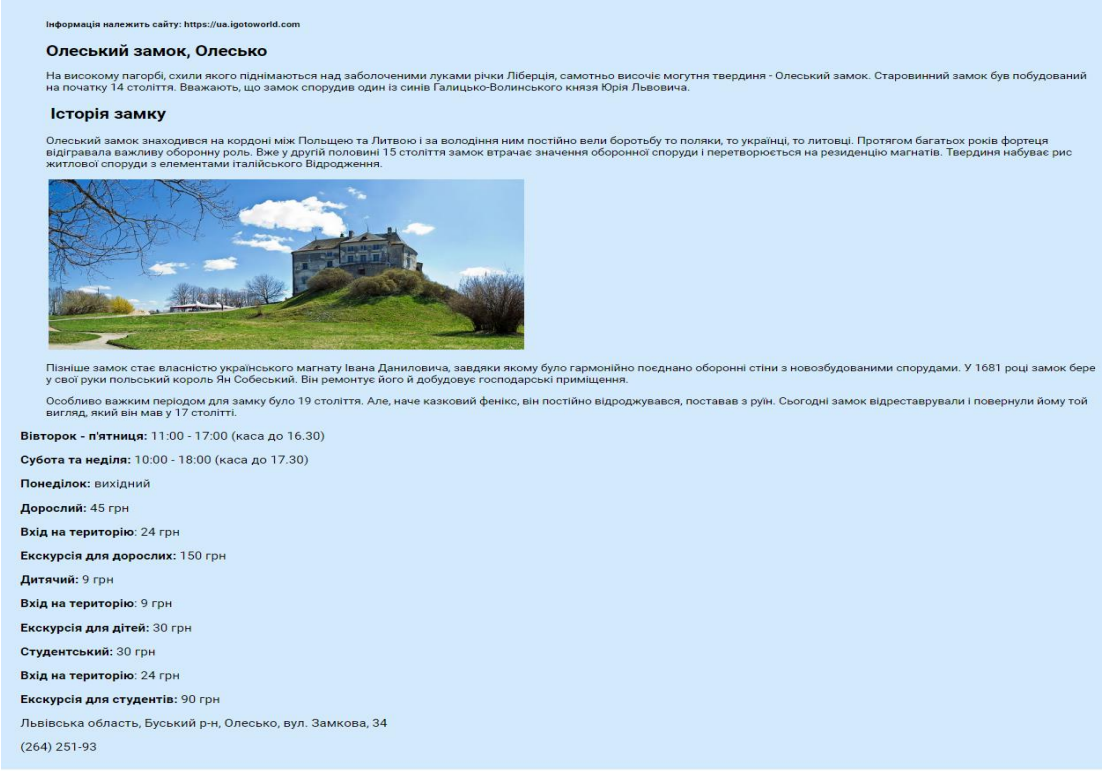


Рисунок 3.15 – Результат роботи скрипта- парсера

Висновки по розділу 3

В даному розділі були розглянуті можливості PHP для створення парсерів і подальшого використання спарсеної інформації. Розглянули структуру html документа. Також були розглянуті різні етапи розробки нашої системи та спроектувані головна та типові сторінки системи.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних і шкідливих виробничих факторів, причин пожеж. Розглянуто заходи, які дозволяють забезпечити гігієну праці та виробничу санітарію. На підставі аналізу розроблено заходи з техніки безпеки і рекомендації з пожежної профілактики. І оскільки завданням на дипломне проектування є програмне забезпечення, то аналіз потенційно небезпечних і шкідливих виробничих факторів виконується для персонального комп'ютера, на якому передбачається реалізація та розробка системи автоматизації збору інформації.

4.1 Аналіз стану умов праці

Для створення системи автоматизації збору інформації достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

Оформлення дипломного проекту з розробки системи автоматизації збору інформації за фізичним навантаженням відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав'язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м'язово-скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Тобто наявні психофізіологічні небезпечні та шкідливі фактори:

а) фізичного перевантаження:

- статичного;

- динамічного;

б) нервово-психічного перевантаження:

- розумового перенапруження;

- монотонності праці; - перенапруження аналізаторів;

- емоційних перевантажень.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені ДСанПіН 3.3.2.007-98[23]. Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви: - для розробників програм тривалістю 15 хвилин через кожну годину роботи.

4.2 Аналіз потенційних небезпечних і шкідливих виробничих факторів при роботі з персональним комп'ютером

Основними характеристиками персонального комп'ютера є наступні:

1) робоча напруга U = + 220 + -5%;
2) робочий струм I = 2A;
3) споживана потужність P =-350 Вт.

Роботу користувача розробленої підсистеми слід віднести до категорії Іа (легкі фізичні роботи). До даної категорії відносяться всі види діяльності, які виконуються сидячи, з періодичним ходінням, і не потребують фізичного напруження НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин» [24].

При експлуатації даного програмного продукту існують такі небезпечні і шкідливі виробничі фактори:

1) фізичні:
    1) підвищений рівень напруги електричної мережі, замикання якої може статися через тіло людини;
    2) підвищена або знижена вологість повітря;
    3) підвищена або знижена рухомість повітря;
    4) підвищений рівень статичної електрики;
    5) підвищена напруженість електричного поля;
    6) відсутність або нестача природного світла;
    7) знижена освітленість робочої зони;
    8) підвищений рівень шуму на робочому місці;
    9) підвищений рівень електромагнітного випромінювання;
    10) знижена контрастність;
2) психофізіологічні;

3) фізичні перевантаження:
   11) статичні;
   12) динамічні;
4) нервово-психічні перевантаження:
   13) розумове перенапруження;
   14) монотонність праці;
   15) перенапруження аналізаторів;
   16) емоційні перевантаження.

4.3 Заходи з охорони праці

4.3.1 Загальні заходи безпеки

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання приклад деяких заходів безпеки:

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 м^3, то потрібно подати не менш як 30 м^3/год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 м^3, то потрібно подати не менш як 20 м^3/год повітря;

в) якщо об'єм приміщення становить понад 40 м^3, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

- зниження рівня шуму та вібрації:

а) у джерелі виникнення, шляхом застосування раціональних конструкцій, нових матеріалів і технологічних процесів;

б) звукоізолювання устаткування за допомогою глушників, резонаторів, кожухів, захисних конструкцій, оздоблення стін, стелі, підлоги тощо;

в) використання засобів індивідуального захисту).

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;
- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;
- не тягнути за мережевий кабель, щоб витягти вилку з розетки;
- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;
- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;
- не залишати включені електроприлади без нагляду;
- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;
- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

4.3.2 Електробезпека

Основним небезпечним фактором при роботі з ЕОМ є небезпека ураження людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані виявити наявність електричної напруги на обладнанні.

Проходячи через тіло людини, електричний струм чинить на нього складний вплив, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон та інших органів тканин організму) дій.

Тяжкість ураження людини електричним струмом залежить від цілого ряду чинників:

1) значення сили струму;
2) електричного опору тіла людини і тривалості протікання через нього струму;
3) типу і частоти струму;
4) індивідуальних властивостей людини і навколишнього середовища.

Приміщення для ЕОМ відноситься до приміщень без підвищеної небезпеки, тобто в приміщення, в яких відсутні умови, що створюють підвищену або особливу небезпеку. Небезпека ураження електричним струмом існує всюди, де використовуються електроустановки, тому приміщення без підвищеної небезпеки не можна назвати безпечними.

Електробезпека забезпечується:

1) відповідною конструкцією електроустановок;
2) застосуванням технічних способів і засобів захисту;
3) організаційними і технічними заходами.

Конструкція електроустановок відповідає умовам їх експлуатації та забезпечує захист персоналу від дотику до струмоведучих частин.

Основними технічними способами і засобами захисту від ураження електричним струмом, що використовуються окремо або в поєднанні один з одним, є:

1) захисне заземлення;
2) занулення;
3) вирівнювання потенціалів;
4) мале напруга;
5) електричне поділ мереж;
6) захисне відключення;
7) ізоляція струмоведучих частин;
8) компенсація струмів замикання на землю;
9) захисні пристрої;
10) попереджувальна сигналізація, блокування, знаки безпеки;
11) ізолюючі захисні та запобіжні пристосування.

4.3.3 Розрахунок захисного заземлення

Основними технічними способами і засобами захисту від ураження електричним струмом, що передбачаються в даному дипломному проекті, є:

1) захисне заземлення,
2) занулення,
3) захисне відключення,
4) ізоляція струмоведучих частин.

Завдання захисного заземлення - усунення небезпеки ураження струмом у випадку дотику до корпусу та інших струмоведучих металевих частин електроустановок, які опинилися під напругою.

Розрахунок заземлюючого контуру виконується виходячи з умови:

$$R_{зу} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п} \leq 4 \quad Ом \tag{4.1}$$

де $R_З$ - опір заземлювача (стержня, труби, куточка і т.д.), Ом;

$R_П$ - Опір лінії, що з'єднує заземлювачі, Ом;

n - кількість заземлювачів;

$\eta_З$ і $\eta_П$ - Коефіцієнти екранування відповідно заземлювача і з'єднує смуги ($\eta_з$= 0,2 ÷ 0,9; $\eta_П$ = 0,1 ÷ 0,7).

Опір заземлювача розраховується за формулою 4.2

$$R_з = \frac{\rho}{2 \cdot \pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + \frac{1}{2} \cdot \ln\frac{4 \cdot t + l}{4 \cdot t - l}\right), \tag{4.2}$$

де ρ - питомий опір ґрунту (взяти з довідкової літератури);

l - довжина заземлювача (для труб 2-3 м, для стрижнів до 10 м), м;

d - діаметр заземлювача (для стрижнів 0,01 - 0,03 м, для труб 0,03 - 0,05 м);

t - відстань від середини забитого в грунт заземлювача до рівня землі (необхідно враховувати, що відстань від верхнього кінця заземлювача до поверхні землі має бути не менше 0,5), м.

Розрахуємо опір заземлювача:

$$R_з = \frac{60}{2 \cdot \pi \cdot 3}\left(\ln\frac{2 \cdot 3}{0.03} + \frac{1}{2} \cdot \ln\frac{4 \cdot 1 + 3}{4 \cdot 1 - 3}\right) = 19.96 \tag{4.3}$$

Опір лінії, що з'єднує заземлювачі розраховується за формулою 4.4

$$R_п = \frac{\rho}{2 \cdot \pi \cdot l} \cdot \ln\frac{2 \cdot L^2}{b \cdot t}, \tag{4.4}$$

де L - довжина лінії, що з'єднує заземлювачі (при контурному заземленні вона приблизно дорівнює периметру виробничої будівлі), м;

b - ширина смуги (0,03 - при прокладанні всередині будівлі і 0,05 - при прокладанні поза будівлею), м;

t - глибина заземлення від рівня землі (0,5 м.).

Розрахуємо опір лінії, що з'єднує заземлювачі

$$R_{П} = \frac{60}{2 \cdot \pi \cdot 3} \cdot \ln \frac{2 \cdot 50^2}{0.03 \cdot 0.5} = 14.37 \quad (4.5)$$

Необхідна кількість заземлювачів, розраховується за формулою 4.6

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \quad (4.6)$$

де 4 - допустимий загальний опір;

2 - коефіцієнт сезонності.

Розрахуємо необхідну кількість заземлювачів,

$$n = \frac{2 \cdot 19,9}{4 \cdot 0.5} = 19,9 \approx 20 \quad (4.7)$$

Округлимо результат в більшу сторону і отримуємо необхідну кількість заземлювачів - 20. Маючи всі необхідні дані розрахуємо опір заземлюючого контуру.

$$R_{зу} = \frac{19,96 \cdot 14,37}{14,37 \cdot 20 \cdot 0,5 + 19,96 \cdot 0,4} = 1,89 \leq 4 \quad Ом \quad (4.8)$$

Опір заземлюючого контуру 1,89 Ом, що відповідає умові Rзу < 4Ом.

4.4 Заходи, що забезпечують виробничу санітарію та гігієну праці

4.4.1 Мікроклімат

Трудова діяльність людини завжди протікає в певних метеорологічних умовах, які визначаються поєднанням температури повітря, швидкості його руху і відносної вологості, тиском і тепловим випромінюванням від нагрітих поверхонь. Оскільки експлуатація проектованого програмного засобу відбувається в приміщенні, то ці показники в сукупності (за винятком тиску) називаються мікрокліматом виробничого приміщення. В даний час основним нормативним документом нормалізації мікроклімату є ГОСТ 12.1.005-88 ССБТ Загальні санітарно-гігієнічні вимоги до повітря робочої зони [20].

Важкість праці характеризує сукупну дію всіх елементів, складових умови праці, на працездатність людини, його здоров'я, життєдіяльність і відновлення робочої сили. У

такому представлені поняття тяжкості праці однаково застосовні як до розумової, так і до фізичної праці. Згідно [20] тяжкість роботи персоналу, який обслуговує ЕОМ, відноситься до легкої категорії 1б (роботи, виконувані сидячи, не вимагаючи систематичного фізичного напруження і перенесення важких предметів) [20]. Загальні санітарно-гігієнічні вимоги до повітря робочої зони. Оптимальні норми мікроклімату в робочій зоні, що забезпечуються для робіт легкої категорії 1б приведені в таблиці 4.3.

Таблиця 4.3 - Оптимальні норми мікроклімату

Період року	Температура, ºС	Відносна вологість,%	Швидкість вітру, м / с, не більше
Холодний	21 - 23	60 - 40	0,1
Теплий	22 - 24	60 - 40	0,2

4.4.2 Освітлення

Світло є природною умовою існування людини . Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Гарне освітлення діє тонізуюче, створює гарний настрій, поліпшує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла у світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IV в розряд точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) дорівнює 200 лк. Джерелом природного освітлення є сонячне світло. У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2015 Природне і штучне освітлення[22].

Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє [22] і для даного приміщення в світлий час доби достатньо природного освітлення. Світильники загального освітлення розташовуються над робочими поверхнями у рівномірно-прямокутному порядку.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n \qquad (4.9)$$

де Sb – площа віконних прорізів, м^2 ;

Sn – площа підлоги, м^2 .

Sn = a·b = 4·4 = 16 м^2 ,

S = 1/8·16 = 2 м^2 .

Приймаємо 1 вікно площею S = 2 м^2. Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 4 м, ширина 4 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна. Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n визначається по формулі (4.10):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \qquad (4.10)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2 ; S = 25 м^2 ;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.10), отримуємо:

$$n = \frac{300 \cdot 20 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 1.6$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою–220 В.

4.5 Рекомендації щодо пожежної безпеки

Виникнення пожежі можливо, якщо на об'єкті є горючі речовини, окислювач і джерела запалювання. Для оцінки пожежної небезпеки слід проаналізувати ймовірність взаємодії цих трьох чинників.

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

1) поліамід - матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420 ° С;
2) полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 ° С, температура самозаймання 530 ° С;
3) склотекстоліт ДЦ - матеріал друкованих плат, трудногорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;
4) пластикат кабельний №.489 - матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1;
5) деревина - будівельний і оздоблювальний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255 ° С, температура самозаймання 399 ° С.

Згідно НАПБ Б. 03.002-2007 таке приміщення належить до категорії "В" (пожежонебезпечної). [25]

Простору всередині приміщень в межах яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали у відповідності з ПУЕ відносяться до пожежонебезпечної зони класу П-IIа.

Потенційними джерелами запалювання можуть бути:

1) іскри і дуги короткого замикання;
2) електрична іскра при замиканні і розмиканні ланцюгів;
3) перегріви від тривалого перевантаження;
4) відкритий вогонь і продукти горіння;
5) наявність речовин, нагрітих вище температури самозаймання;
6) розрядна статичну електрику.

Причинами можливого загоряння і пожежі можуть бути:

1) несправність електроустановки;
2) конструктивні недоліки обладнання;

3) коротке замикання в електричних мережах;
4) запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згоряння, що виділяються під час пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор та ін.

При горінні пластмас, крім звичних продуктів згоряння, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти; формальдегіди; хлористий водень; фосген; синильна кислота; аміак; фенол; ацетон; стирол. Пожежо-вибухонебезпечність речовин і матеріалів. Номенклатура показників і методи їх визначення ГОСТ 12. 1. 044 - 89 ЕСБТ[21].

Для захисту персоналу від впливу небезпечних і шкідливих факторів пожежі проектом передбачається застосування промислового протигаза фільтруючого з коробкою марки В (жовтий).

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем вентиляції та кондиціонування, розвиненою системою електроживлення ЕОМ. Небезпека загорання в ЕОМ пов'язана з великою кількістю щільно розташованих на платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80 ... 100 ° С), що може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання.

Пожежна безпека при застосуванні ЕОМ забезпечується:

1) системою запобігання пожежі:
2) системою протипожежного захисту:
3) організаційно-технічними заходами.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі та важкогорючі) не надається технічно можливим. Тому проектом передбачаються способи і засоби запобігання утворенню (або внесення) в горюче середовище джерел запалювання, таких як:

1) застосування електроустаткування, відповідної пожежонебезпечної і вибухонебезпечної зонами відповідно до ПУЕ;
2) застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалювання;

3) виключення можливості появи іскрового розряду в займистою середовищі з енергією, яка дорівнює і вище мінімальної енергії запалювання.

Висновки до розділу 4

У розділі "Охорона праці" виконаний аналіз потенційних небезпек при роботі із засобами обчислювальної техніки, на підставі якого розроблено заходи з техніки безпеки, заходи, що забезпечують виробничу санітарію та гігієну праці, розрахунки природного та штучного освітлень, рекомендації з пожежної профілактики, які підтверджені відповідними розрахунками.

ВИСНОВКИ

Метою даної дипломної роботи було дослідження методів вилучення вмісту веб-сторінок, а також розробка системи автоматизації збору інформації до туристичної карти України.

У першому розділі проекту були поставлені цілі і завдання, необхідні для реалізації проекту, ми з'ясували наскільки актуальна тема проекту. Були розглянуті існуючі аналоги, аналіз яких допоміг нам у визначенні напрямку розробки прототипу і дизайну.

У другому розділі на етапі підготовки ми розглянули і підібрали апаратні і програмні засоби для подальшої роботи. Розібрали етапи парсингу, розглянули методи експорту та імпорту даних. Коротко ознайомилися зі способами захисту від парсингу.

У третьому розділі при розробці розглянули можливості PHP для створення парсерів. Були розглянуті різні етапи проектування нашої системи.

Таким чином завдяки аналізу, який проводився під час проектування була розроблена система автоматизації збору інформації до туристичної карти України.

Нажаль на даному етапі система охоплює декілька областей і має обмежений функціонал, але надалі планується додавання нових фортець та музеїв зінших областей України.

СПИСОК ЛІТЕРАТУРИ

1. Абрамян Г.В. Нормативно-правовая, инновационно-исследовательская и ресурсно-технологическая модели совместной деятельности вуза с учреждениями региона в информационной среде / Г.В. Абрамян // Информатика: проблемы, методология, технологии: Материалы XVI Международной научно-методической конференции / Под редакцией А.А. Крыловецкого. – 2016. – С. 20–25.
2. Графика в HTML5 [Електронний ресурс]. – Режим доступа: https://www.w3.org/standards/webdesign/graphics (дата обращения: 11.01.2017).
3. Дэйли Д. Разработка веб-приложений с помощью SVG [Текст] / Д. Дэйли, Д. Фрост. – Нью-Йорк: Майкрософт, 2012. – 294 с.
4. Фрайн Б. HTML5 и CSS3 Разработка сайтов для любых браузеров и устройств [Текст] / Б. Фрайн. – СПб.: Питер, 2014. – 304 с.
5. How to Parallax Scrolling [Електронний ресурс]. – Точка доступу: URL: http://www.w3schools.com/howto/howto_css_parallax.asp
6. Будущее за HTML или прощай Flash! [Електронний ресурс]. – Режим доступа: https://www.searchengines.ru/buduschee_za_h.html
7. Парсинг html-сайтов с помощью PHP, Ruby, Python [Електронний ресурс]. – Точка доступу: URL: http://parsing.valemak.com/ru/what-why-how/.
8. Web parsing: задачи, проблемы, инструменты [Електронний ресурс]. – Точка доступу: URL: https://inostudio.com/ru/article/web-parsing.html.
9. Парсинг и обработка веб-страницы на PHP: выбираем лучшую библиотеку [Електронний ресурс]. – Точка доступу: URL: https://tproger.ru/digest/parse-html-via-php/
10. [Веб застосунок [Електронний ресурс]. – Точка доступу: URL: http://uk.wikipedia.org/wiki/
11. Реферат на тему: «Засоби створення Web-додатків» [Електронний ресурс]. – Точка доступу: URL: URL: http://ifreestore.net/2267/ – Реферат на тему: «Засоби створення Web-додатків».
12. HTML & CSS [Електронний ресурс]. – Точка доступу: URL: https://www.w3.org/standards/webdesign/htmlcss
13. Зольников Д.С. PHP5. Как самостоятельно создать сайт любой сложности. – 2-е изд. стер. – М.: НТ Пресс, 2007. – 272с.
14. PHP [Електронний ресурс]. – Точка доступу: URL: https://uk.wikipedia.org/wiki/PHP

15. Apache HTTP Server [Електронний ресурс]. – Точка доступу: URL: https://uk.wikipedia.org/wiki/Apache_HTTP_Server

16. MySQL [Електронний ресурс]. – Точка доступу: URL: https://uk.wikipedia.org/wiki/MySQL

17. Bringing MySQL to the web [Електронний ресурс]. – Точка доступу: URL: https://www.phpmyadmin.net

18. Подключение к MySQL и выполнение запросов [Електронний ресурс]. – Точка доступу: URL: https://metanit.com/web/php/7.2.php

19. PHP Simple HTML DOM Parser Manual [Електронний ресурс]. – Точка доступу: URL: http://simplehtmldom.sourceforge.net/manual.htm

20. ГОСТ 12.1.005-88 "ССБТ Загальні санітарно-гігієнічні вимоги до повітря робочої зони".

21. ГОСТ 12.1.044-89 ССБТ. ГОСТ 12.1.044-89 ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения.

22. ДБН В.2.5-28:2015 Природне і штучне освітлення.

23. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

24. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно- обчислювальних машин.

25. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою.

Додаток А

Лістинг файлу home.php(головна сторінка)

```
<!DOCTYPE html>
<html>
     <head>
          <title>Замки Украины</title>
          <meta charset="utf-8">
          <link rel="stylesheet" type="text/css"
href="css/style1.css">
     </head>
     <body >
<?php
  include_once "header.php"
?>
     <!-- /HEADER -->
     <!-- NAVIGATION -->
<?php
  require "navigation.php"
?>
          <!-- /NAVIGATION -->
     <!-- MAIN -->
     <main>
          <!-- <div class="container"> -->
          <h1 class="title" onmouseover="changeText(this)"
onmouseout="returnChangeText(this)" >ЗАМКИ УКРАИНЫ</h1>
               <div class="block" >
                    <img class="descr"
src="image/header.jpg" ><p>Практически в каждой области Украины
можно найти древние сооружения. О многих мы знаем давно, в
некоторых даже успели побывать. Величественные крепости, мощные
стены которых выстояли не одно столетие. Есть и замок, на
который жители Украины смотрят гораздо чаще, чем на все
остальные, ибо изображён он на купюре 200 гривен – это замок
Любарта в Луцке.Есть и те, которые менее популярны среди
украинцев и гостей нашей страны.</p><img class="descr1"
src="image/text.jpg" ><p>Но есть в Украине и такие замки и
крепости, которыми можно восхищаться и гордиться. Величественные
крепости, мощные стены которых выстояли не одно столетие и еще
веками будут удивлять наших потомков, заслуживают того, чтобы
быть увиденными. Так вот  именно они и вызывают наибольший
интерес. </p> <p>Конечно большинство замков, которые есть в
Украине, расположены в Западной Украине. Красивые замки и
крепости Западной Украины не требуют дополнительных
рекомендаций. В основном это известные жемчужины архитектуры,
большинство из которых сохранилось до наших дней в хорошем
состоянии. Хотя, конечно, не всем бывшим замкам-красавцам так
повезло... Но и они могут стать довольно живописным фоном для
Ваших открыточных фотографий. Это замки Волыни, Тернопольщины,
Львовщины, Ивано-Франковщины, Буковины и Закарпатья. Сегодня мы
побываем замках, крепостях и дворцах Украины, в которых
обязательно стоит побывать. </p><br>
```

```
                    </div>
                    <div class="intmap"><p><h1>ИНТЕРАКТИВНАЯ
КАРТА</h1> </p></div>
                    <p style="text-align: center;color:
#4141c3"><b>Вашему вниманию предоставляется интерактивная план-
схема Украины, разбитая по областям.<br>
                Для ознакомления - нажмите на неё мышкой.</p>
               <div class="map">
                    <svg viewBox="20 20 685 460"
xmlns="http://www.w3.org/2000/svg" version="1.1"
xmlns:xlink="http://www.w3.org/1999/xlink">
                    <a href="Gallery_lviv.php">
                         <path  data-id="lv" class="region"
id="lviv"  d="m 84.300848,88.199152 c 7.771041,1.573747
12.930464,3.476997 16.567792,4.872882 l 0.11274,6.334745
2.3237,4.872881 4.3856,1.50005 4.98561,1.49823 4.72197,-0.86185
3.86194,1.94915 4.31105,-2.43644 c -1.18127,3.2204 -
0.69504,7.72455 -1.80006,10.53306 0.007,2.44171 -0.0613,4.95968
1.50005,5.84746 2.02294,2.31981 6.03178,3.44807 4.42195,7.94753
1.39832,0.20886 2.54581,1.04478 2.36189,5.20925 l -5.922,1.42368
c -2.14023,2.23516 -4.78304,2.46004 -7.72206,1.50005 l -
1.94916,5.36017 c -0.62697,1.69413 -3.95927,3.24089 -
4.38559,6.33475 -2.37264,0.42349 -4.27426,0.37595 -5.84746,0 l -
2.21097,2.99827 -4.423783,0.48729 -2.623718,-2.811 c -
3.184988,0.27373 -3.889432,4.88841 -3.898305,10.72034 l -
3.636483,3.41102 c 1.083337,3.02373 -1.500315,5.74745 -
5.434714,7.27113 -5.097736,-0.23433 -10.195473,-1.28678 -
15.29321,2.28735 -4.533599,-2.64591 -7.877337,4.22706 -
11.694915,7.30932 0.36285,2.73019 0.590507,5.50158 -
1.987341,7.4966 2.499912,2.43644 3.134595,4.87289
2.43644,7.30933 l -6.296557,-5.06016 -7.309322,-4.87288
1.461865,-5.36017 -6.334746,-0.48729 -0.749111,-4.27286 c -
8.260859,-5.41957 -5.704072,-13.24306 -7.534787,-20.09155 -
0.291488,-8.22828 1.323124,-13.27973 3.411017,-17.54237
6.067914,-2.51762 8.354108,-9.25909 13.231323,-15.55504
3.458822,-6.96942 8.557989,-11.47833 13.569524,-16.11869 l
14.618644,-7.796611 c 7.386756,-2.924416 10.184372,-6.683222
8.771187,-11.207627 z" description="<img
src='image/blazons/lv.png'>Львовская область и её замки:
Злочевский, Олеско, Подгорцы, Свирж, Жолква...">
                         </path>
                    </a>
                    <a href="Gallery_volyn.php">
                         <path  data-id="vl" class="region"
id="volyn" d="m 84.300848,88.199152 8.9376,2.626582
7.630192,2.2463 0.11274,6.334745 2.3237,4.872881 v 0 l
9.37121,2.99828 4.72197,-0.86185 3.86194,1.94915 4.31105,-
2.43644 c 1.0681,1.69707 -1.20278,-1.91106 -2.53032,-4.02034 l -
1.99713,-2.721412 4.66707,-2.969954 4.47001,-3.145695 8.98266,-
8.558377 8.0613,-0.848558 4.66707,3.818512 1.16989,-3.569975
3.79719,-0.900032 -0.30001,-7.834086 1.64564,-4.118523 -0.0728,-
5.339885 2.66995,-3.269964 -5.09135,-7.637024 -1.69712,-2.969953
-8.83706,1.74859 -0.92136,-6.839939 c -0.11823,-3.024259
2.29779,-5.085111 1.69712,-11.879814 1.80328,-2.107579 3.9582,-
```

```
5.621686 6.56125,-10.928308 1.03722,-4.174892 -1.73893,-5.299084
-4.6156,-6.342866 -7.35135,-2.017329 -14.65495,-1.095821 -
21.21395,0.848558 -8.30339,-0.792921 -10.21778,1.714275 -
14.92256,2.421407 -0.74119,3.580389 -8.31665,6.199748 -
13.401191,12.604104 -4.106761,0.07852 -7.03605,-4.868879 -
16.474088,-3.269965 -4.69737,1.368813 -4.516763,10.054592 -
6.364186,15.698326 -2.587761,3.868181 0.09732,10.372783
2.121395,16.546884 3.527451,6.620592 6.35019,15.198285
6.657778,25.748641 z" description="<img
src='image/blazons/vl.png'>Волынская область и её замки: Луцкий,
Ольцкий..."></path>
                    </a>
                    <a href="Gallery_rivne.php">
                         <path class="region" id="rovno" d="m
129.60458,126.98822 -4.33334,-4.6789 -1.50005,-5.84746 1.80006,-
10.53306 v 0 c 1.33181,-2.24725 -2.95922,-4.4945 -4.52745,-
6.741752 l 3.16059,-2.799905 2.10008,0.900032 c 0.50381,-
2.306202 0.67579,-4.991646 3.87641,-4.215776 v 0 l 8.98266,-
8.558377 c -1.43479,0.151031 5.37009,-0.565273 8.0613,-0.848558
2.07316,-0.352811 1.8857,3.464932 4.66707,3.818512 l 2.51994,-
2.219927 2.44714,-2.25008 -0.30001,-7.834086 1.64564,-4.118523 -
0.0728,-5.339885 v 0 l 2.66995,-3.269964 v 0 c 1.05684,1.394799
-5.252,-6.829199 -6.78847,-10.606977 l -8.83706,1.74859 -
0.92136,-6.839939 1.69712,-11.879814 c 3.2294,-3.642769
5.33117,-7.285539 6.56125,-10.928308 3.84609,-1.671631 8.34141,-
2.57445 10.99253,-1.059519 1.97109,1.925785 4.71525,3.811233
8.25029,3.000106 3.00408,0.692075 6.97862,1.093288
9.60034,2.55009 3.87401,3.084326 7.10997,1.013403
10.35037,2.100075 2.25687,5.563322 7.15378,3.28145
9.60034,6.450227 1.90277,3.62625 5.3326,2.630325 8.10028,4.50016
2.70597,2.069493 5.25835,4.879198 9.90035,3.000106 l
0.60003,3.000106 v 4.500159 h -2.40009 l -0.60002,2.100074 -
3.90014,0.600021 -0.60002,3.600128 -3.30012,0.30001 c -
1.07595,2.632821 -0.0971,6.035473 -0.90003,9.450334 l -
1.20004,1.650059 -0.30001,3.900138 -3.75013,0.150005 -
0.15001,12.45044 -1.80006,0.300011 -0.15001,5.700201 c -
0.9617,0.634466 -0.3353,1.886343 -1.20004,3.300117 l -
0.45002,3.900138 -5.10018,2.400081 c -4.33851,0.88725 -
8.02467,1.99194 -11.10039,3.30012 v 1.20004 c -3.46361,1.55918 -
4.26015,3.78513 -6.30022,5.7002 l -3.60013,4.80017 -
12.00042,0.90004 -3.00011,-2.40009 h -6.90024 l -0.30001,1.80006
-3.30012,1.50006 -3.00011,1.50005 h -8.7003 l -1.50006,3.60013 -
2.40008,0.90003 z" description="<img
src='image/blazons/rv.png'>Ровенская область и её замки: Дубно,
Острог, Губкив..."></path>
                    </a>
                    <a href="lviv.php">
                         <path class="region" id="zakarpat'e"
d="m 33.314342,173.1938 3.634292,2.72572 1.817146,0.30286
0.908573,4.08857 4.542864,0.45429 -0.302857,3.6343
0.908573,2.72571 c 1.413335,1.58926 2.675242,1.23436
3.937149,2.12001 l 4.951407,3.54445 2.014319,3.11842
3.331434,0.90857 2.120004,3.02857 2.725719,-0.60571
0.302857,3.33143 4.391436,-0.15142 -1.060001,6.51144 c
```

```
2.241832,1.49684 2.703889,0.87173 5.602867,0.90857 l
2.574289,5.75429 4.240008,-0.15143 c 1.271489,1.17118
1.526313,2.06652 1.817145,4.08858 h 3.028577 l -0.302858,3.93715
4.542865,3.63429 c -0.402399,1.94122 -0.251117,3.05192
0.151429,3.78573 l 0.908573,2.72571 1.968575,-0.75714 -
0.302858,4.24001 -3.634292,1.51429 c -2.109464,3.10671 -
7.027202,3.80828 -11.811449,4.24001 l -7.268583,-0.60572 c -
2.422861,-2.97521 -5.300009,-2.81178 -7.72287,-2.5743 -
0.719285,-1.48904 -3.032813,-2.3404 -5.754295,-3.02857 -
1.259398,-1.83492 -3.393742,-2.61487 -6.208583,-2.57429 -
0.364147,-2.906 -2.481581,-2.30542 -3.937149,-3.02858 -
2.321909,-0.5325 -4.643817,-0.85107 -6.965726,0.60572 -
1.419691,2.31979 -4.821602,3.12074 -9.691445,4.69429 -3.628322,-
2.42136 -4.831067,-5.0692 -7.268584,-6.36001 -2.548221,-4.2918 -
4.976604,-10.91182 -8.177156,-13.17431 -2.781251,-7.62183 -
6.2466357,-8.40233 -9.3885875,-12.41716 2.4773912,-3.26134
2.6632444,-7.46949 3.6743957,-10.58243 1.208619,-2.5884
2.3077543,-5.61474 3.8970458,-6.68046 3.597109,-2.13774
4.042296,-10.12824 8.480014,-14.23431 l 3.937149,-3.63428 c
3.74633,0.48998 7.953662,0.36529 9.388588,3.93714 z"
description="<img src='image/blazons/zak.png'>Закарпатская
область и её замки: Ужгород, Мукачево, Невицкий,
Середне..."></path>
                    </a>
                    <a href="">
                         <path class="region" id="iv-frank" d="m
83.463265,241.72834 c 8.263539,3.02328 9.37559,9.74897
20.291455,16.35431 0.76951,-2.04336 1.30323,-4.14567 -0.1774,-
6.75157 v -3.02858 l 1.81714,-1.21143 0.30286,-1.81715 -
1.21143,-0.60571 0.0887,-3.25795 1.67647,-4.35024 0.052,-2.99183
2.72572,0.30286 1.51428,-4.24001 3.6343,-0.60571 -0.60572,-
4.24001 3.33143,-0.90857 1.81715,-2.12001 v -2.12 l 3.63429,-
0.60572 1.51429,-2.12 4.54286,-0.60572 0.30286,3.93715
3.33144,0.30286 -0.30286,-4.24001 1.81715,-1.21143 0.60571,-
3.02857 -1.51429,-0.60572 0.60572,-4.84572 1.21143,-1.86911
3.33143,0.052 -2.12,-4.24 -1.81715,0.60571 v -2.72572 l -
2.42286,0.30286 -3.02858,-3.93715 h -6.66286 l -1.21143,-1.21143
-0.60572,-3.63429 -4.24001,-0.30286 -1.51428,-3.93715 -3.02858,-
0.60571 -0.30286,-7.8743 -1.81714,-1.81715 -0.60572,-5.45144 -
1.21143,-2.12 -0.60572,-5.45144 -3.02857,-4.54286 -1.51429,-
1.81715 0.90857,-3.33143 -4.240005,-0.30286 -0.605715,-1.51429 -
2.725719,-0.30286 -1.639736,3.45689 c -0.559324,2.29974
2.149115,8.33407 -1.994556,6.53742 v 0 l -2.120004,2.72572 v
4.24 l -4.240007,3.93715 H 71.171549 v 1.81715 l -4.240008,-
0.60572 -3.028576,1.21143 -0.605715,2.12001 c -2.344069,1.10309
-3.606362,2.56677 -4.240008,4.24 l -1.817146,1.81715 v 5.45144 l
-1.21143,1.81714 2.422861,3.02858 0.605715,3.93715
2.120004,3.02857 3.331434,0.30286 -0.302858,2.42286 3.634292,-
0.30285 -0.302857,6.66287 5.602867,0.90857 2.574289,5.75429
4.240008,-0.15143 1.817145,4.08858 h 3.028577 l -
0.302858,3.93715 4.542865,3.63429 0.151429,3.78573
0.908573,2.72571 1.968575,-0.75714 -0.302858,4.24001 c -
3.634292,1.51429 -5.671879,2.81853 -8.302604,4.63157 z"
```

```
description="<img src='image/blazons/iv-fr.png'>Ивано-
Франковская область и её замки: Пнів..."></path>
                    </a>
                    <a href="">
                         <path class="region" id="chernovzy" d="m
103.75472,258.08265 c 0.74242,-2.25052 0.16829,-4.50105 -
0.1774,-6.75157 v -3.02858 l 1.81714,-1.21143 0.30286,-1.81715 -
1.21143,-0.60571 0.0887,-3.25795 v 0 l 1.67647,-4.35024 v 0 l
0.052,-2.99183 2.72572,0.30286 1.51428,-4.24001 v 0 0 l 3.6343,-
0.60571 -0.60572,-4.24001 v 0 l 3.33143,-0.90857 1.81715,-
2.12001 v -2.12 l 3.63429,-0.60572 v 0 l 1.51429,-2.12 4.54286,-
0.60572 0.30286,3.93715 3.33144,0.30286 -0.30286,-4.24001
1.81715,-1.21143 0.60571,-3.02857 -1.51429,-0.60572 -0.0367,-
2.70419 1.85389,-4.01064 3.33143,0.052 v 0 l 2.46977,1.14645
1.71322,2.35568 4.49721,-0.64246 0.21415,2.35568 2.35568,1.92738
4.49721,-0.42831 v 2.56984 l 5.78212,-0.21416 3.21229,3.6406 -
0.42831,1.28491 1.92738,0.42831 0.21415,-1.49907 2.99814,-
0.64246 2.35568,-1.07076 -0.21415,-3.42644 5.13966,-0.21416 -
0.21415,1.71323 h 3.42644 l 1.49907,2.56983 1.49907,-0.64246 v -
1.71322 l 1.92737,0.4283 -0.4283,-1.71322 5.13966,0.85661
0.42831,1.07077 3.85475,0.21415 1.28491,-2.14153 1.71322,-
1.28491 h 2.78399 l 2.56983,2.78398 -0.4283,1.49907
2.35568,1.28492 -1.07077,5.56797 -7.92365,-1e-5 c -
0.97747,2.38306 -4.06174,2.13262 -7.28119,1.71322 -
2.07909,1.24921 -5.95417,1.86986 -11.99255,0.85662 -
1.96575,0.67009 -3.45868,2.75867 -4.28306,6.85289 l -
6.21042,6.63872 -10.06518,7.06705 -2.78398,2.14152 -4.28306,-
0.4283 c -2.28429,0.351 -4.56859,-1.04792 -6.85288,-1.71322 -
6.56735,1.30341 -13.1347,1.25199 -19.70205,0.85661 -
3.04928,3.11391 -9.84287,2.98943 -15.74249,10.73628 z"
description="<img src='image/blazons/Chernivtsi.png'>Черновицкая
область и её замки: Хотинская крепость..."></path>
                    </a>
                    <a href="">
                         <path class="region" id="ternopil" d="m
167.17742,217.41102 -2.12,0.30286 -0.60571,1.51429 -
1.81715,0.30285 -0.90857,-1.81714 -2.42286,-3.02858 -5.45144,-
0.30286 -0.60572,-1.81714 -4.54286,-0.30286 -2.12001,-1.51429 v
-2.72572 l -5.45143,0.60572 -0.60572,-2.42286 h -2.42286 l -
0.30286,-2.12001 -2.42286,-2.72572 -1.81714,0.30286 v -1.81714 l
-1.81715,-0.30286 -3.33144,-4.24001 -6.36001,0.30286 -1.51428,-
1.21143 -0.30286,-2.72572 -0.60572,-1.21143 h -3.33143 l -
0.90857,-1.21143 -0.90858,-1.81715 -0.60571,-1.51429 -
2.12001,0.60572 -0.90857,-1.51429 -0.30286,-7.57144 -2.12,-
1.51429 v -4.84572 l -0.60572,-0.30286 v -1.51429 l -1.21143,-
0.30285 v -5.14858 h -0.60571 v -2.12001 l -0.90857,-0.30286 -
0.60572,-2.12 -0.90857,-1.51429 h -0.90858 l -0.30285,-3.93715
1.51429,-1.81714 2.12,0.60571 -0.15143,-2.12 5.90572,0.30286 -
0.60571,-2.12001 2.42286,-1.81714 1.81715,-0.60572 v -2.72572 l
1.81714,-0.30286 1.21143,-1.81714 v -3.02858 h 5.14858 v -
1.21143 h 1.81715 l 0.30286,-0.90857 h 3.93715 l 2.12,-0.90857 v
-5.14858 h -1.81715 l -0.60571,-3.33144 2.72572,-2.72572 h 2.12
c 1.84178,-1.51921 -0.45084,-2.78698 0.90857,-3.63429 l
9.38859,-0.15142 c -0.49302,-1.50138 0.8043,-0.97478 2.27143,-
```

```
1.81716 l 1.96858,0.15143 0.30286,-1.51428 h 2.72571 l -
0.30285,-2.12001 6.05715,0.30286 0.60572,1.21143 h 1.66571 l
1.66572,1.81715 -1.21143,3.63429 1.81714,0.30286 v 1.51428 l -
1.51428,0.30286 v 3.63429 l -0.90858,0.30286 v 3.02858 l -
2.12,1.21143 -0.30286,4.84572 2.12001,-0.30286 0.30285,3.93715
2.12001,0.90858 v 8.48001 l -1.21143,0.30286 -0.30286,5.14858 h
-1.81715 v 13.02288 h 1.51429 l 0.30286,3.02857 -1.21143,1.21143
-0.30286,5.45144 -1.21143,1.21143 v 14.53717 l -1.21143,0.60572
v 9.38858 h 1.81715 l -0.60572,1.81715 1.21143,1.81714 c
0.73342,1.08373 2.10681,1.52748 2.12,3.33144 l 0.90858,1.81714
2.12,0.60572 z" description="<img
src='image/blazons/tern.png'>Тернопольская область и её замки:
Кременец, Збаражский замок, ..."></path>
                    </a>
                    <a href="">
                         <path class="region" id="hmelnizk" d="m
167.17742,217.41102 -3.33143,-3.93715 -2.12,-0.60572 -0.90858,-
1.81714 v 0 l -3.33143,-5.14858 0.60572,-1.81715 h -1.81715 v -
9.38858 l 1.21143,-0.60572 v -14.53717 l 1.21143,-1.21143
0.30286,-5.45144 1.21143,-1.21143 -0.30286,-3.02857 h -1.51429 v
-13.02288 h 1.81715 l 0.30286,-5.14858 1.21143,-0.30286 v -
8.48001 l -2.12001,-0.90858 -0.30285,-3.93715 -2.12001,0.30286
0.30286,-4.84572 2.12,-1.21143 v -3.02858 l 0.90858,-0.30286 v -
3.63429 l 1.51428,-0.30286 v -1.51428 l -1.81714,-0.30286
1.21143,-3.63429 v 0 0 0 0 l 2.83199,-0.37843 8.56611,-0.42831
3.21229,-3.64059 -0.42831,-1.71323 1.71323,-1.49906 4.4972,-
4.49721 1.07077,0.4283 -0.64246,-2.35568 3.21229,0.21416 -
0.21415,-1.28492 4.28305,0.42831 -0.42831,-1.28492 h 2.78399 l -
0.21415,-1.28492 2.35568,0.42831 0.21415,-1.28492 h 5.78212 l -
0.64246,10.92179 1.92738,1.71322 2.35568,-0.4283 -
0.21415,2.56983 1.28491,0.42831 v 5.35381 l 8.99442,-0.21415 -
0.42831,4.92551 2.56983,1.28492 0.21415,2.35568 -2.78398,0.21415
-2.56983,2.56983 0.21415,2.78399 1.92737,1.49907 -
0.21415,9.42272 1.28492,1.07076 2.99814,-0.21415 -
0.21416,1.49907 2.14153,-0.64246 v 1.92737 l -1.07076,0.21415
0.21415,3.85475 -0.85661,1.07077 -0.21415,2.56983 -
2.14153,0.85661 v 2.35568 h -1.07076 l -0.21416,1.92737
1.71322,0.21416 0.21416,2.99813 h 4.28305 v 1.71323 l -
1.07076,0.21415 v 1.71322 l -1.49907,0.21415 v 5.13967 l
2.14152,2.14153 1e-5,4.92551 h -1.71323 l 0.21416,1.92737 -
1.71323,0.85661 -1.07076,0.42831 -0.4283,-1.49907 -
2.78399,0.85661 -1.07076,-1.92738 -7.7095,0.42831 -
1.92737,1.28491 0.64245,3.42645 -1.07076,0.21415 v 3.85475 l -
1.07076,-0.21415 -0.21416,3.42644 2.14153,0.21415
0.21415,1.92738 -1.28491,1.07076 -0.21416,1.92738 h 2.35568 l
0.21416,2.56983 -1.92738,1.71322 -0.21415,3.42644 -
1.92737,1.28492 v 2.99814 l -1.28492,0.85661 -2.78399,-0.21415 -
1.28491,2.35568 h -1.71322 l -0.21416,1.28491 -3.64059,-0.4283 v
-1.28492 l -5.13967,-0.21415 -0.4283,1.28491 -1.28492,0.21416 v
0.85661 l -1.92737,0.21415 -1.07077,-1.28492 -0.21415,-1.07076 h
-2.99814 v -1.28492 l -5.13966,0.21416 0.21415,2.99813 z"
description="<img src='image/blazons/hmel.png'>Хмельницкая
область и её замки: Каменец-Подольский, Межибож,
Староконстянтинов, Сатанов..."></path>
```

```
                    </a>
                    </svg>
               <div class="description">
               </div>
                    <img src="image/original.gif"/>
          </div>
          </main >
          <!-- /MAIN -->
          <!-- /FOOTER -->
     <?php
          require "footer.php"
     ?>
          <!-- /FOOTER -->
               <script src="script/jquery-3.3.1.min.js"></script>
               <script type="text/javascript"
src="script/js.js.js"></script>
          </body>
     </html>
```

Додаток Б

Лістинг файлу header.php(«шапка»)

```
               <div class="hjpg" style="background-image:
          url('image/header.jpg'); width: 100%; height: 200px;
          position: relative;">
                    <h2 style="text-align: center; color:
#e455dd; padding: 100px 50px;margin: 0 0 19px 0; ">УДИВИТЕЛЬНЫЕ
ПУТЕШЕСТВИЯ ПО ЗАМКАМ И КРЕПОСТЯМ УКРАИНЫ</h2>
               </div>
```

Лістинг файлу navigation.php(меню навігації)

```
     <div><nav class="nav"><link rel="stylesheet" type="text/css"
href="css/style1.css">
               <div class="container">
                    <ul class="menu">
                        <li><a href="home.php">Home</a></li>
                                   <li><a
href="Gallery1.html">Gallery</a>
                         <ul class="submenu">
                              <li><a
href="Gallery_lviv.php">Львовская </a></li>
                                   <li><a
href="Gallery_volyn.php">Волынская</a></li>
                                   <li><a
href="Gallery_rivne.php">Ровенская </a></li>
                                   <li><a href=#>Закарпатская
</a></li>
                                   <li><a
href="Gallery_lviv.php">Львовская</a></li>
```

```
                                    <li><a
href="Gallery_volyn.php">Волынская</a></li>
                                    <li><a
href="Gallery_rivne.php">Ровенская</a></li>
                                    <li><a
href=#>Закарпатская</a></li>
                                  </ul>
                    </li>
                        <li><a href="lviv.php">Contacts</a></li>
                    </ul>
               </div>
          </nav></div>
```

Лістинг файлу footer.php(«підвал»)

```
          <footer>
               <div class="footer">
                    <a href="lviv.php"> Team d'Ok &#169 <?php
echo gmdate("F d.Y."); ?> All rights reserved.</a>
               </div>
          </footer>
```

Додаток В

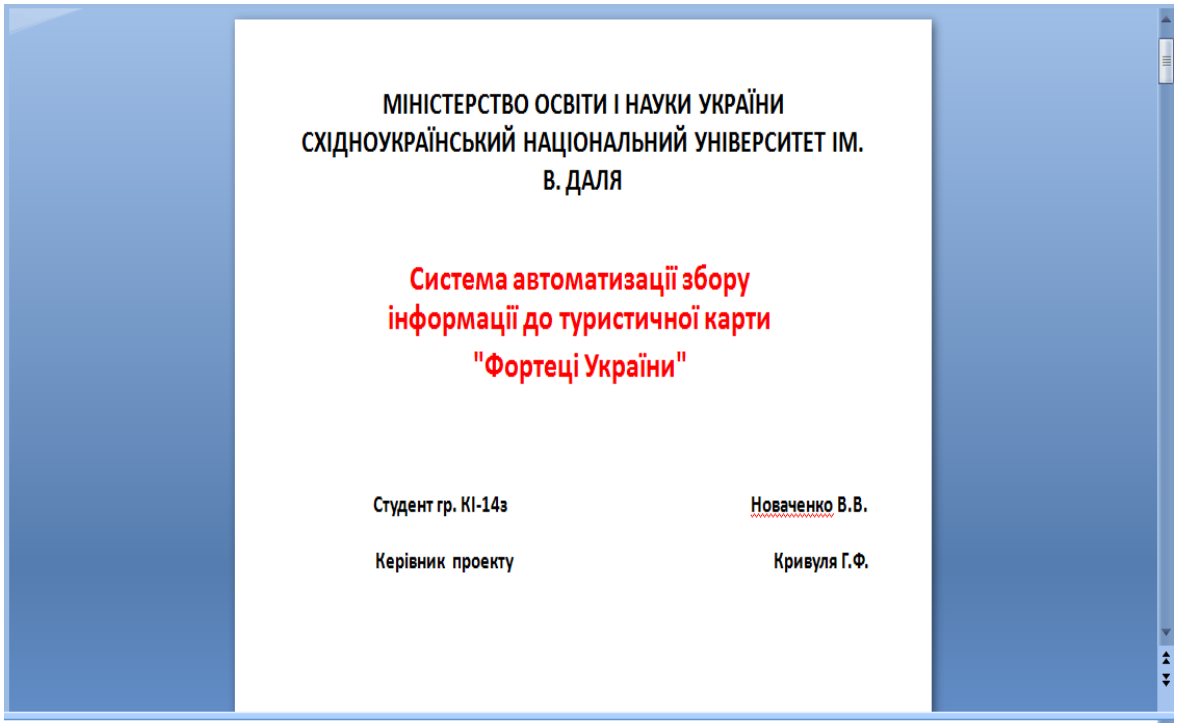
Лістинг файлу parser_olesko.php (сбір даних про Олеський замок)

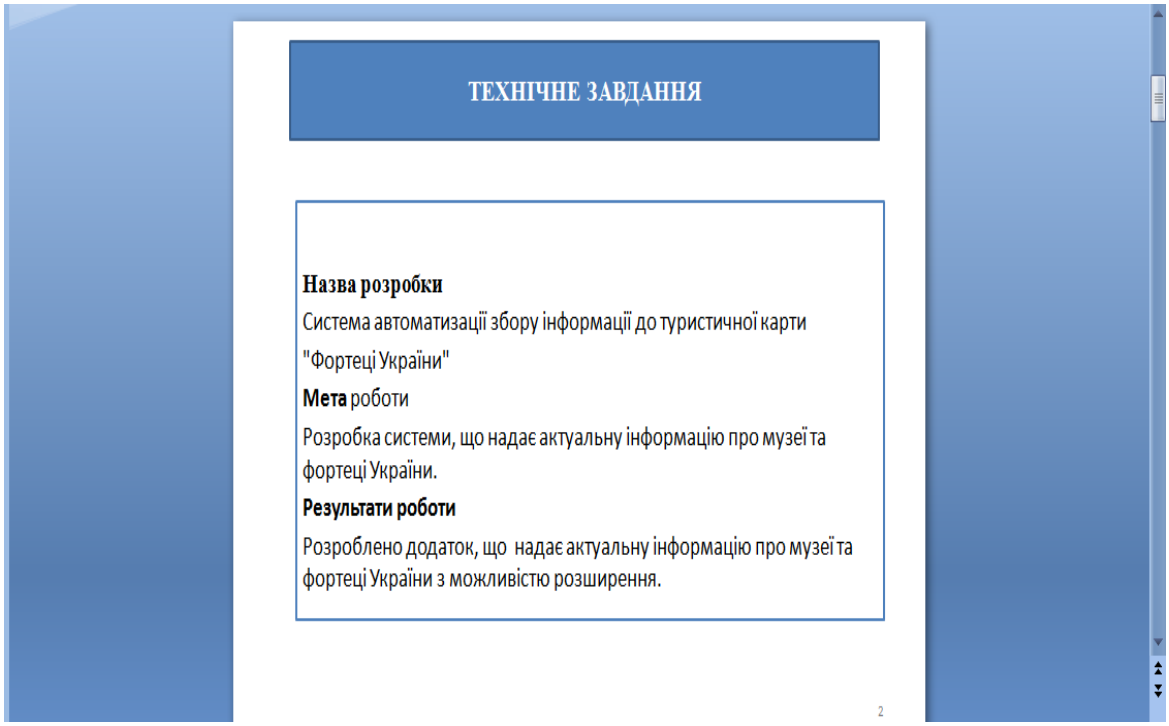
```
     <?php
     require_once 'header.php';
     require_once 'navigation.php';
     require_once 'simple_html_dom.php';
     $html =
file_get_html('https://ua.igotoworld.com/ua/poi_object/2400_oles
ko-castle.htm');
     if ($html->innertext!='' and count($html->find('p'))){
     ?>
     <p ><h5 style="text-align: left;padding: 20px 50px 0
50px;margin: 0 20px;">Інформація належить сайту:
https://ua.igotoworld.com</h5><h2 style="text-align:
justify;padding: 20px 50px 0 50px;margin: 0 20px;">Олеський
замок, Олесько</h2></p>
     <?php }
     foreach ($html->find('div.tour-description-text') as $div) {
          ?>
          <div style="/*width:90%;*/text-align: left;padding: 0px
100px 0 50px; margin: 0px 0 0 20px;background-color:
#d2eafb;"><?php echo $div->innertext ?></div>
     <?php }
```

```
     $html =
file_get_html('http://lvivgallery.org.ua/museums/muzey-
zapovidnyk-oleskyy-zamok');
     if ($html->innertext!='' and count($html->find('p'))){
     ?>
     <p  style="text-align: left;padding: 20px 50px 0
50px;margin: 0 20px;"><h5 style="text-align: left;padding: 20px
50px 0 50px;margin: 0 20px;">Інформація належить сайту:
http://lvivgallery.org.ua</h5><h3 style="text-align:
justify;padding: 20px 50px 0 50px;margin: 0 20px;">Додаткова
інформація:</h3> </p>
     <?php }
     foreach ($html->find('div.field-items') as $div) {
      ?>
               <div style="width:80%;text-align: justify;padding:
0 50px 0px 10px;margin: 0 20px;/*list-style: none;*/">
                 <?php echo  $div->innertext ?></div>
     <?php }
       $html->clear();unset($html);
     require_once 'footer.php';
     ?>
```

Додаток Г

Презентація:





ЗАВДАННЯ РОБОТИ

1) Отримання інформації про музеї та фортеці за допомогою парсингу.
2) Реалізація списку фортець та музеїв для виведення інформації в зручному вигляді.
3) Реалізація інтерактивної карти України з зазначеними на ній фортецями.
4) Розробка додатку з можливістю розширення.

3

ПАРСИНГ

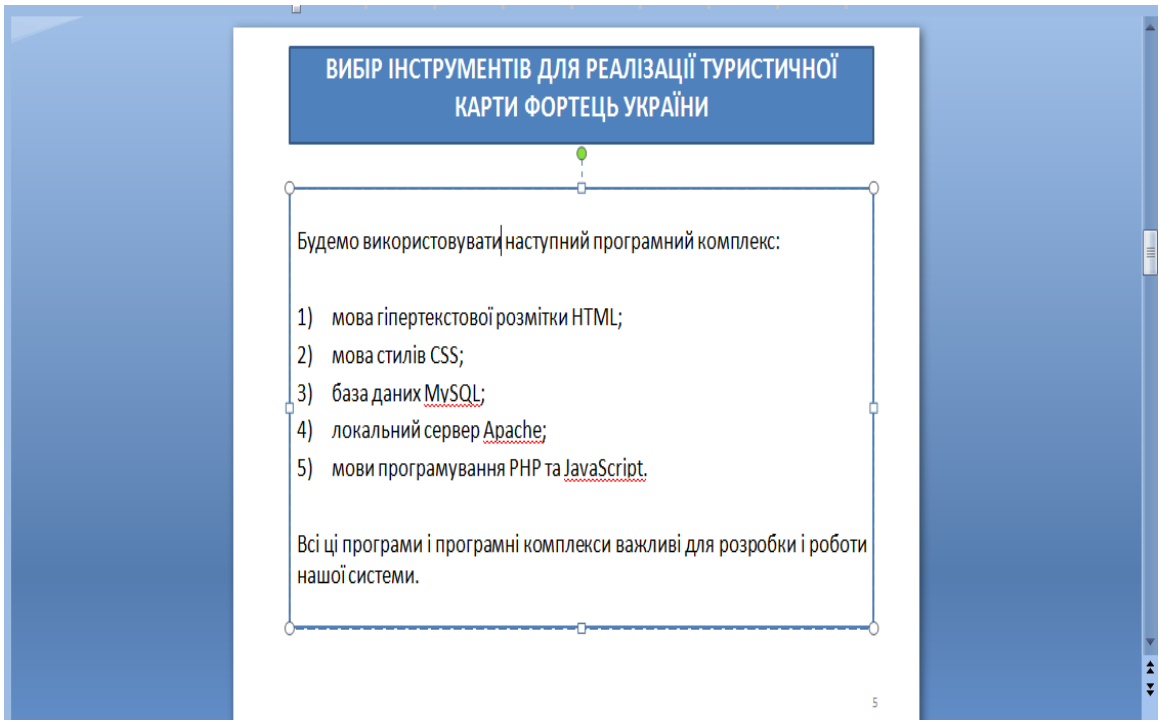
Процес витягнення структурованої корисної інформації з сайту називається парсинг (parsing), а інструменти для реалізації даного процесу - парсерами (аналізатори).

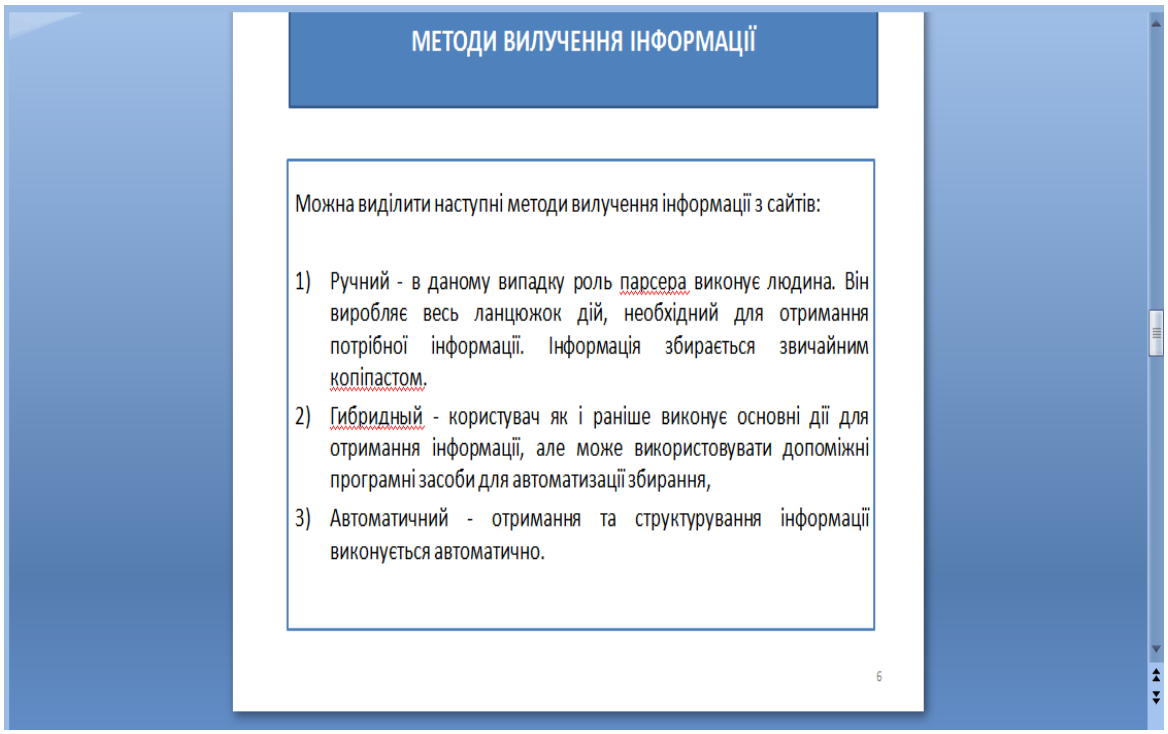
Парсинг сайтів - послідовний синтаксичний аналіз інформації, яка розміщена на Інтернет-сторінках.

У порівнянні з людиною, комп'ютерна програма-парсер:

1) швидко обійде тисячі веб-сторінок;
2) акуратно відокремить технічну інформацію від «людської»;
3) безпомилково відбере потрібне і відкине зайве;
4) ефективно упакує кінцеві дані в необхідному вигляді.

4


ВИБІР ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ ТУРИСТИЧНОЇ КАРТИ ФОРТЕЦЬ УКРАЇНИ
Будемо використовувати наступний програмний комплекс:
1) мова гіпертекстової розмітки HTML;
2) мова стилів CSS;
3) база даних MySQL;
4) локальний сервер Apache;
5) мови програмування PHP та JavaScript.
Всі ці програми і програмні комплекси важливі для розробки і роботи нашої системи.
5


МЕТОДИ ВИЛУЧЕННЯ ІНФОРМАЦІЇ
Можна виділити наступні методи вилучення інформації з сайтів:
1) Ручний - в даному випадку роль парсера виконує людина. Він виробляє весь ланцюжок дій, необхідний для отримання потрібної інформації. Інформація збирається звичайним копіпастом.
2) Гибридный - користувач як і раніше виконує основні дії для отримання інформації, але може використовувати допоміжні програмні засоби для автоматизації збирання,
3) Автоматичний - отримання та структурування інформації виконується автоматично.
6

ВИБІР БІБЛІОТЕК

1) Регулярні вирази
2) XPath і DOM
3) Simple HTML DOM
4) phpQuery
5) cURL

Для парсинга краще використовувати бібліотеку phpQuery: вона швидка, функціональна і сучасна.

З іншого боку, для зовсім простих завдань можливо використовувати стандартні модулі PHP, такі як XPath, Simple HTML DOM або регулярні вирази.

7

ЕТАПИ ПАРСИНГА

Парсинг html-сторінки вдає із себе процес, який можна розбити на три етапи:

1) Отримання початкового коду веб-сторінки - скачати програмний код тієї сторінки сайту, з якої необхідно витягти інформацію.
2) Витяг з html-коду необхідних даних. Отримавши сторінку, необхідно обробити її - відокремити звичайний текст від гіпертекстової розмітки, вистроїти ієрархічне дерево елементів документа, коректно зреагувати на невалідний код, виокремити зі сторінки саме ту інформацію, заради якої і відбувається весь цей процес.
3) Фіксація результату. Благополучно обробивши дані на сторінці, потрібно їх зберегти в необхідному вигляді для подальшої обробки.

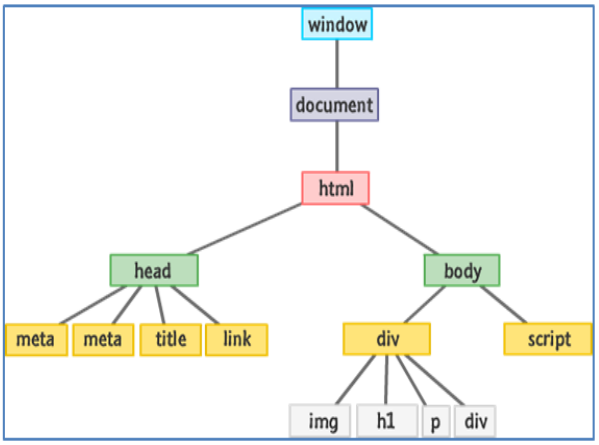
8

ЗБЕРЕЖЕННЯ ДАНИХ

1) MySQL - найбільш використованої СУБД для веб-розробок;
2) CSV (англ. Comma-Separated Values - значення через кому) - формат текстових файлів для зберігання табличної інформації;
3) XLS - спарсенний матеріал представляють у вигляді електронної таблиці Excel;
4) JSON - часто зібрана інформація тут же за допомогою запитів SQL заноситься в базу даних. Альтернативою є JSON - текстовий формат обміну даними оснований на JavaScript;
5) Document Object Model дозволяє ефективно вибудувати ієрархічне дерево веб-документа для подальшої роботи з ним.

9

ІЄРАРХІЧНЕ ДЕРЕВО ВЕБ-ДОКУМЕНТА



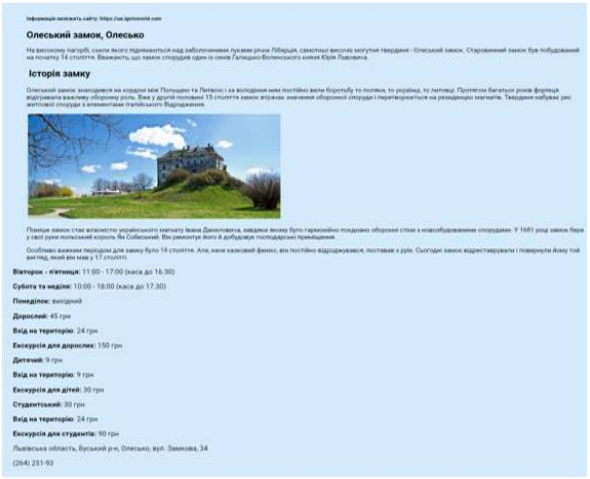
10

HTML-ДОКУМЕНТ

```
    $html =
file_get_html('http://lvivgallery
.org.ua/museums/muzey-zapovidnyk-
oleskyy-zamok');
    if ($html->innertext!='' and
count($html->find('p'))){
    ?>
    foreach ($html-
>find('div.field-items') as $div)
{
     ?>
                ///
                  <?php echo
$div->innertext ?></div>
    <?php }
      $html-
>clear();unset($html);
    require_once 'footer.php';
    ?>
```

11

РЕЗУЛЬТАТ РОБОТИ СКРИПТА- ПАРСЕРА



Інформація написана сайту: https://ua.igotoworld.com

Олеський замок, Олесько

На високому пагорбі, схили якого підмиваються над заболоченими луками річки Лівчиця, самотньо високі могутні твердині - Олеський замок. Старовинний замок був побудований на початку 14 століття. Вважають, що замок спорудив один із синів Галицько-Волинського князя Юрія Львовича.

Історія замку

Олеський замок знаходився на кордоні між Польщею та Литвою і за володіння ним постійно вели боротьбу то поляки, то українці, то литовці. Протягом багатьох років фортеця відігравала важливу оборонну роль. Вже у другій половині 15 століття замок втратив значення оборонної споруди і перетворюється на резиденцію магнатів. Тогодашні набувачі рис житлової споруди з елементами італійського Відродження.

Поляки замок стає власністю українського магнату Івана Даниловича, завдяки якому було гармонійно поєднано оборонні стіни з новозбудованими спорудами. У 1681 році замок бере у свої руки польський король Ян Собеський. Він ремонтує його й добудовує господарські приміщення.

Особливо важким періодом для замку було 19 століття. Але, нами казковий фенікс, він постійно відроджувався, поставав з руїн. Сьогодні замок відреставрували і повернули йому той вигляд, який він мав у 17 столітті.

Вівторок - п'ятниця: 11:00 - 17:00 (каса до 16.30)
Субота та неділя: 10:00 - 18:00 (каса до 17.30)
Понеділок: вихідний
Дорослий: 45 грн
Вхід на територію: 24 грн
Екскурсія для дорослих: 150 грн
Дитячий: 9 грн
Вхід на територію: 9 грн
Екскурсія для дітей: 30 грн
Студентський: 30 грн
Вхід на територію: 24 грн
Екскурсія для студентів: 90 грн
Львівська область, Буський р-н, Олесько, вул. Замкова, 34
(264) 251-93

12