Зміст

[Анотація 5](#_Toc11779646)

[Annotation 6](#_Toc11779647)

[Вступ 7](#_Toc11779648)

[1 Техніко-економічне обґрунтування доцільності проектування 8](#_Toc11779649)

[1.1 Суть проблеми, що виникла на сучасному етапі розвитку науки, техніки, технологій 8](#_Toc11779650)

[1.2 Існуючі способи вирішення технічної проблеми 8](#_Toc11779651)

[1.3 Прогнозування собівартості нової розробки 10](#_Toc11779652)

[1.4 Розрахунок капітальних вкладень 11](#_Toc11779653)

[1.5 Прогнозування величини експлуатаційних витрат 11](#_Toc11779654)

[2 Аналіз методів та засобів створення веб-додатків 13](#_Toc11779655)

[2.1 Клієнтська частина 13](#_Toc11779656)

[2.2 Серверна частина 18](#_Toc11779657)

[2.3 Аналіз існуючих баз даних 23](#_Toc11779658)

[2.4 Редактор коду Visual Studio Code 28](#_Toc11779659)

[2.5 Виконання технічних вимог 29](#_Toc11779660)

[3 Розробка сервісу автоматизації роботи бібліотеки коледжу 30](#_Toc11779661)

[3.1 Обґрунтування потреби веб-сайту 30](#_Toc11779662)

[3.2 Розробка структури сайту 30](#_Toc11779663)

[3.3 Розробка головної сторінки сайту 33](#_Toc11779664)

[3.4 Розробка та дизайн основних складових шапки сайту 38](#_Toc11779665)

[3.5 Встановлення локального серверу 45](#_Toc11779666)

[3.6 Підключення бази даних 46](#_Toc11779667)

[3.7 Авторизація та реєстрація користувача 47](#_Toc11779668)

[3.8 Адаптивний веб-дизайн 51](#_Toc11779670)

[4 Економічна частина 55](#_Toc11779671)

[4.1 Розрахунок капітальних витрат 55](#_Toc11779672)

[4.2. Розрахунок експлуатаційних витрат при використані програмного продукту 59](#_Toc11779673)

[4.3 Розрахунок обсягу робіт при використанні програмного продукту 63](#_Toc11779674)

[4.4 Розрахунок економічної ефективності 64](#_Toc11779675)

[5 Охорона праці 66](#_Toc11779676)

[5.1 Характеристика робочого місця 66](#_Toc11779677)

[5.2 Безпека праці 66](#_Toc11779678)

[5.3 Умови праці 70](#_Toc11779679)

[5.4 Ергономіка робочого місця 78](#_Toc11779680)

[5.5 Пожежна безпека 81](#_Toc11779681)

[5.6 Надзвичайні ситуації 83](#_Toc11779682)

[Висновки 84](#_Toc11779683)

[Список літератури 85](#_Toc11779684)

# Анотація

У дипломному проекті було розроблено сервіс автоматизації роботи бібліотеки коледжу «BookLibrary» для перегляду книг, які знаходяться в бібліотеці ВК НУХТ. Для розробки проекту було використано сучасні засоби розробки сайтів: Vue, Node.js.

У першому розділі дипломного проекту описується техніко-економічне обґрунтування та суть технічної проблеми, що виникла на сучасному етапі розвитку техніки та технологій.

У другому розділі описується аналіз організаційно-економічних та технологічних аспектів діяльності сайту, аналіз програмних методів та засобів для розробки веб-сайту.

У третьому розділі описується розробка програмного забезпечення та структури Web-сайту «BookLibrary»

У четвертому розділі обчислюється обсяг робіт при використанні програмного продукту та розрахунок річного економічного ефекту для споживача від запровадження нового програмного продукту.

У п’ятому розділі описана техніка безпеки і електробезпека при експлуатації комп'ютерної техніки.

# Annotation

During this diploma work was developed automated library service, that allows to browse through books, that stored in library of Vinnytsya Collage National University of Food Technology. Project was developed with use of modern tools such as Node,js for servers side and Vue.js for frontend.

In first chapter of diploma, described technical and economical justification and core of technical problem that appears in modern stage of progress of technologies.

Second chapter of work is dedicated to analysis of modern economical and technical aspects of web site activity. Also analyzed modern tools of web development and selected suitable pack of tools for Book Library project.

Third chapter of diploma work is about development of server side and frontend of Book Library project.

The fourth chapter is dedicated to calculation of amount of work to developa web site and the calculation of the annual economic benefit for consumers from the introduction of the new software.

In the fifth chapter of diploma is described of they safely rules and the rules of the electrical safey for the exploitation the computer equipment and the technical devices.

# Вступ

Веб-технологія повністю змінила наші уявлення про роботу з інформацією, та і з комп'ютером взагалі. Виявилось, що традиційні параметри розвитку комп’ютерної техніки – продуктивність, пропускна спроможність, ємність пристроїв, що запам'ятовують, не враховували головного – системи інтерфейсу з людиною. Застарілий механізм взаємодії людини з інформаційною системою стримував впровадження нових технологій і зменшував користь від їх застосування. І лише коли інтерфейс між людиною і комп'ютером був спрощений до природності сприйняття, відбувся вибух інтересу до можливостей комп’ютерної техніки.

З розвитком освіти в Україні все частіше виникає необхідність у наданні необхідної літератури студентам навчальних закладів. Також постає проблема в управлінні та зберіганні навчальної літератури та організації видачі її студентам. Більшість існуючих систем працюють у ручному режимі та не мають організованої структури зберігання даних про літературу, не мають можливості зберігання електронних примірників, завантаження їх, а також не містять можливості організації управління видачою літератури студентам навчальних закладів. Саме тому розробка веб сайту, що зможе ефективно та зручно організувати зберігання електронних примірників літератури, керувати видачою начальних посібників та художньої літератури учням навчальних закладів є актуальною на сьогоднішній день.

# 1 Техніко-економічне обґрунтування доцільності проектування

# 1.1 Суть проблеми, що виникла на сучасному етапі розвитку науки, техніки, технологій

Розробка сайту-бібліотеки – це один з напрямків в області розробки веб-сайтів. Основне призначення сайту-бібліотеки - це організація зберігання літератури, та додавання нової літератури на сервер з можливістю скачування електронних копій.

Одним із завдань, що найчастіше постає перед сайтом бібліотекою, є організація електронного зберігання примірників книг. Надаючи доступ до літератури безпосередньо в онлайн-режимі, такий вид сайту як сайт -бібліотека це завдання може ефективно виконати.

Метою розробки сайту бібліотеки є недосконалість сучасних аналогів програмного забезпечення. Зокрема, сайт із функціями онлайн доступу до літератури дозволяє економити час на відвідання бібліотеки та надає можливість записатись на видачу фізичної версії примірника книги, та прийти отримати її у бібліотеці коледжу.

Процес розробки веб-сайту неможливо уявити без попередньої постановки цілей і завдань а також ретельного аналізу сучасних аналогів майбутнього продукту. Аналіз сайтів конкурентів, аналіз цільової аудиторії майбутнього сайту, виявлення «сильних» і «слабких» сторін проекту – розробка сайту бібліотеки завжди починається з виконання подібних завдань.

## 1.2 Існуючі способи вирішення технічної проблеми

Існує маса різноманітних систем, фреймворків, мов, сервісів, бібліотек та структур, які можуть створювати та генерувати веб-сторінки на веб-сайтах. Але більшість з них об’єднує той факт, що усі веб-сайти будуються з допомогою мови розмітки [HTML](http://uk.wikipedia.org/wiki/HTML) та каскадних таблиць стилів [CSS](http://uk.wikipedia.org/wiki/HTML).

Для якісної підтримки сервісу необхідно забезпечити роботу професійного колективу, який виконуватиме такі функції:

* розроблення та тестування нових функціональних можливостей;
* адміністрування (адміністрування операційної системи, веб-сервера, баз даних, оновлення програмного забезпечення, облік проблем тощо);
* генерування інформаційного наповнення;
* управління інформаційним;
* аналіз виконаної роботи, її оцінювання та визначення перспектив;
* розроблення інструкцій, делегування повноважень всередині колективу і контроль за якістю та своєчасністю виконання завдань.

Для полегшення розробки сервісу використовувались фреймворки.

Програмний фреймворк — це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно — програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту.

Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

У фреймворків є дві основні функції: робота на серверній стороні (бекенд) і робота на клієнтській стороні (фронтенд).

Фронтенд-фреймворки пов'язані з зовнішньою частиною програми. Простими словами, вони відповідають за зовнішній вигляд програми. Бекенд відповідає за внутрішній устрій додатку.

Розглянемо обидва типи детальніше.

* Серверні фреймворки. Правила та архітектура таких фреймворків не дає можливості створити веб-додаток з багатим інтерфейсом. Вони обмежені в своїй функціональності, проте все одно є можливість створеняя простих сторінок і різних форм. Також вони можуть формувати вихідні дані і відповідати за безпеку в разі атак. Все це виразно може спростити процес розробки. Серверні фреймворки в основному відповідають за окремі, але критично важливі частини програми, без яких воно не зможе нормально працювати.
* Клієнтські фреймворки. На відміну від серверних, клієнтські фреймворки ніяк не пов'язані з логікою програми. Цей тип фреймворків працює в браузері. З їх допомогою можна поліпшити і впровадити нові призначені для користувача інтерфейси. Фронтенд-фреймворки дозволяють створювати різні анімації і односторінкові додатки. Всі клієнтські фреймворки відрізняються по функціональності і використання.

## 1.3 Прогнозування собівартості нової розробки

Для визначення доцільності розробки електронного додатку для вивчення методів спрощення логічних функцій необхідно спрогнозувати орієнтовну собівартість розробки та можливі експлуатаційні витрати. Така інформація потрібна і розробникам так ы користувачам.

Для прогнозування собівартості нової розробки скористаємось методом питомої ваги. Даний метод застосовується тоді, коли є можливість розрахувати одну із прямих статей витрат і встановити приблизно питому вагу даної статті витрат в собівартості аналогічної продукції.

Собівартість одиниці нової продукції можна спрогнозувати за формулою:

[грн], (1.1)

де – величина однієї з статей прямих витрат, яка вибрала за основу, грн;

– питома вага однієї з статей прямих витрат в собівартості аналога, %;

– коефіцієнт, який ураховує конструктивні та технологічні особливості нової розробки,

## 1.4 Розрахунок капітальних вкладень

Капітальні вкладення (К) та експлуатаційні витрати (Е) визначаються приблизно спрощеним способом.

Величина капітальних вкладень (К) може бути розрахована за формулою:

[грн], (1.2)

де В – коефіцієнт, який враховує витрати на розробку, придбання, транспортування, вантаж, налагодження нової розробки тощо; В 1,1 1,8

А – коефіцієнт, який враховує прогнозований прибуток, податок, який повинен виплачувати виробник тощо; А 1,5 2

S – собівартість нової розробки, грн, яка оцінюється приблизним способом;

## 1.5 Прогнозування величини експлуатаційних витрат

Експлуатаційними витратами є такі витрати, які забезпечують нормальне функціонування цього вибору в період експлуатації.

В найбільш спрощеному вигляді величина експлуатаційних витрат в розрахунку за 1 рік може бути прогнозована за формулою:

(1.3)

де – коефіцієнт, який враховує витрати на амортизацію, електроенергії, обслуговування, ремонти тощо. Рекомендується приймати значення Для обчислювальної техніки значення рекомендується приймати по максимуму;

А – коефіцієнт, який враховує прогнозований прибуток, податок, який повинен виплачувати виробник тощо; А 1,5 2;

S – собівартість нової розробки, грн, яка оцінюється приблизним способом;

– доля часу, який витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту, в загальному часі своєї роботи .

Виходячи із отриманих прогнозних розрахунків, можна зробити висновок, що створення сервісу автоматизації роботи бібліотеки коледжу є актуальним питанням на сучасному етапі розвитку науки, техніки і технологій та доцільним, що і підтверджують прогнозні розрахунки.

# 2 Аналіз методів та засобів створення веб-додатків

# 2.1 Клієнтська частина

Клієнтська частина – це практичне застосування технологій HTML, CSS та JavaScript для створення сайту або веб-додатку, щоб користувач переглядав їх та взаємодіяв з ними напряму. Вимоги що відносяться до клієнтської розробки пов’язані з інструментами та техніками що використовуються для створення сайту постійно змінюються, і розробник повинен постійно слідкувати за розвитком цих інструментів та технологій.

Задача в розробці сайту – впевнитись, що коли користувач відкриє сайт – він побачить релевантну інформацію у зручному для читання вигляді. Це ускладняється ще й тим, що тепер користувачі використовують велику різноманітність пристроїв з різними розмірами екрану і розширеннями, таким чином змушуючи верстальника враховувати ці аспекти при розробці сайту.

Розробник повинен гарантувати, що сайт коректно відображається в різних браузерах (cross-browser), різних операційні системи (cross-platform) і на різних пристроях (cross-device), що вимагає ретельного планування на стороні розробника.

Є кілька інструментів які можуть бути використані для розробки клієнтської частини веб-сайту, розуміння, які інструменти краще всього підходить для виконання конкретних завдань відзначає різницю між оптимізованим та нераціональним сайтом.

Першою з тріади технологій веб сайтів є мова розмітки HTML, це стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документа та ідеологію структурної розмітки тексту. Попри те, що HTML – штучна комп'ютерна мова, вона не є мовою програмування. HTML разом із каскадними таблицями стилів та вбудованими скриптами – це три основні технології побудови веб-сторінок.

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.

Проте для більш зручної стилізації використовуються каскадні таблиці стилів (CSS) – це спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі - сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки - розділення змісту сторінки (даних) та їхньої візуальної презентації.

Однак якщо є потреба створення динамічної та інтерактивної веб системи – потрібно застосувати JavaScript, що дозволить взаємодіяти користувачу з інтерфейсом та організувати клієнт-серверну комунікацію.

JavaScript – динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови С, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

Vue.js – JavaScript-фреймворк для створення інтерфейсу користувача через реактивне зв’язування даних.

Легко інтегрується в проекти з використанням інших JavaScript-бібліотек.

Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Особливості:

* Шаблони
* Реактивність
* Переходи
* Роутинг

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, які можна проаналізувати за допомогою специфічних браузерів і HTML-парсерів. Vue компілює шаблони у віртуальні функції візуалізації DOM. У поєднанні з системою реактивності Vue може обчислити мінімальну кількість компонентів для повторної візуалізації та застосування мінімальної кількості DOM-маніпуляцій, коли змінюється стан програми.

Об'єктна модель документа (англ. Document Object Model, DOM) – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами. З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленням документів у вигляді дерева. Все це призначено для того, аби надати можливість комп'ютерній програмі доступу та динамічної модифікації структури, змісту та оформлення документа.

Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM. Наприклад:

* автоматичне застосування класів для CSS переходів та анімацій;
* інтегрування сторонніх бібліотек для CSS анімацій, таких як Animate.css;
* використовувати JavaScript для прямих маніпуляцій з DOM під час переходів;
* інтегрування сторонніх JavaScript бібліотек анімацій, таких як Velocity.js.

Vue надає інтерфейс для зміни того, що відображається на сторінці на основі поточного шляху URL - незалежно від того, як його було змінено. Крім того, використання інтерфейсного маршрутизатора дозволяє навмисно переходити шлях веб-переглядача, коли певні події браузера (тобто кліки) відбуваються на кнопках або посиланнях.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Less – це динамічна мова стилів, яка інтерпретується в каскадні таблиці стилів (CSS).

Основні інструменти Bootstrap:

* Сітки (grid) — наперед задані, готові до використання колонки
* Шаблони (template) — фіксовані чи адаптивні шаблони сторінок
* Типографіка (typography) — опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо
* Мультимедіа (media) — засоби управління зображеннями та відео
* Таблиці (table) — засоби оформлення таблиць, які зокрема забезпечують сортування
* Форми (form) — класи для оформлення як форм, так і деяких подій
* Навігація (nav, navbar) — класи для оформлення вкладок, сторінок, меню і панелей навігації
* Сповіщення (alert) — класи для оформлення діалогових вікон, підказок і спливаючих вікон
* Іконочний шрифт (icon font) — набір іконок у вигляді шрифту, складається майже з 500 компонентів.

### 2.2 Серверна частина

Веб-сервер – це [сервер](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80), що приймає [HTTP](https://uk.wikipedia.org/wiki/HTTP)-запити від [клієнтів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), зазвичай [веб-браузерів](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), видає їм [HTTP](https://uk.wikipedia.org/wiki/HTTP)-відповіді, зазвичай разом з [HTML](https://uk.wikipedia.org/wiki/HTML)-сторінкою, зображенням, [файлом](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), медіа-потоком або іншими даними. Веб-сервер - основа [Всесвітньої павутини](https://uk.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D1%81%D0%B2%D1%96%D1%82%D0%BD%D1%8F_%D0%BF%D0%B0%D0%B2%D1%83%D1%82%D0%B8%D0%BD%D0%B0) [13]. Веб-сервером називають як [програмне забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює. Клієнти дістаються веб-сервера за [URL](https://uk.wikipedia.org/wiki/URL)-адресою потрібної їм [веб-сторінки](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) або іншого ресурсу.

Додатковими функціями багатьох веб-серверів є:

* ведення [журналу серверу](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3) про звернення користувачів до ресурсів;
* [автентифікація](https://uk.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F) користувачів;
* підтримка сторінок, що динамічно генеруються;
* підтримка [HTTPS](https://uk.wikipedia.org/wiki/HTTPS) для захищених з'єднань з клієнтами.

Сервер – у комп'ютерній термінології термін може стосуватися окремого комп'ютера чи програми. Головною ознакою в обох випадках є здатність машини чи програми переважну кількість часу працювати автономно, без втручання людини, реагуючи на зовнішні події відповідно до встановленого програмного забезпечення. Втручання людини відбувається під час встановлення серверу і під час його сервісного обслуговування. Часто це роблять окремі [адміністратори](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B8%D0%B9_%D0%B0%D0%B4%D0%BC%D1%96%D0%BD%D1%96%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80) серверів з вищою кваліфікацією.

Сервер як комп'ютер – це [комп'ютер](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) у [локальній](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0) чи глобальній мережі, який надає користувачам свої обчислювальні і дискові ресурси, а також доступ до встановлених сервісів; найчастіше працює цілодобово, чи у час роботи групи його користувачів.

Сервер як програма - [програма](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), що надає деякі послуги іншим програмам ([клієнтам](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82)). Зв'язок між клієнтом і сервером зазвичай здійснюється за допомогою [передачі повідомлень](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), часто через мережу, і використовує певний [протокол](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) для [кодування](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) запитів клієнта і відповідей сервера. Серверні програми можуть бути встановлені як на серверному, так і на персональному комп'ютері, щоразу вони забезпечують виконання певних служб (наприклад, [сервер баз даних](https://uk.wikipedia.org/wiki/%D0%A1%D0%9A%D0%91%D0%94) чи [веб-сервер](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80)).

У більшості загального користування сервер фізичного [комп'ютера](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) (система комп'ютерної техніки) призначений запустити одну або декілька послуг (як приймаюча сторона) для задоволення потреб користувачів інших комп'ютерів в мережі. В залежності від обчислювальних послуг, які вона пропонує, це може бути сервер баз даних, файловий сервер, поштовий сервер, сервер друку, веб-сервер, ігровий сервер, або якийсь інший сервер. У контексті архітектури клієнт-сервер, сервер являє собою комп'ютерну програму, яка обслуговує запити інших програм - «клієнтів».

Таким чином, сервер виконує деякі обчислювальні завдання від імені "клієнтів". Сервери часто надають основні послуги через мережу, або в приватних користувачів - всередині великої організації або громадським користувачам - через [Інтернет](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Мережевий сервер являє собою комп'ютер, призначений для обробки запитів і передачі даних на інші (клієнт) комп'ютери по локальній мережі або через Інтернет. Мережеві сервери зазвичай конфігуруються з додатковою пам'яттю і ємністю для обробки навантаження з обслуговування клієнтів.

Вимоги до обладнання для серверів варіюються залежно від сервера додатків. Абсолютна швидкість [процесора](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80) не настільки важлива для сервера, як для настільного комп'ютера. Обов'язки сервера надавати послуги багатьом користувачам по мережі призводять до різних вимог, таких як швидке підключенням до мережі та висока [пропускна спроможність](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%83%D1%81%D0%BA%D0%BD%D0%B0_%D1%81%D0%BF%D1%80%D0%BE%D0%BC%D0%BE%D0%B6%D0%BD%D1%96%D1%81%D1%82%D1%8C_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB%D1%83). Оскільки сервери, як правило, доступні по мережі, вони можуть працювати без монітору. Процеси, які не потрібні для функції сервера не використовуються. Багато серверів не мають графічного інтерфейсу користувача. Крім того, аудіо-та USB інтерфейси можуть бути опущені. Сервери часто працюють протягом тривалого часу без перерви, тому надійність обладнання і довговічність надзвичайно важлива.

Хоча сервери можуть бути побудовані з частин комп'ютера, критично важливі корпоративні сервери не можливі без використання спеціалізованого устаткування з низьким рівнем збою в цілях максимального часу безперебійної роботи, оскільки навіть короткострокові невдачі можуть коштувати дорожче, ніж покупка і установка системи . Наприклад, це може зайняти всього декілька хвилин часу простою на національній [фондовій біржі](https://uk.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BD%D0%B4%D0%BE%D0%B2%D0%B0_%D0%B1%D1%96%D1%80%D0%B6%D0%B0), щоб виправдати рахунок повністю замінити системи з чимось більш надійним. Сервери можуть включати в себе більшу ємність [жорстких дисків](https://uk.wikipedia.org/wiki/%D0%96%D0%BE%D1%80%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA), більше комп'ютерних вентиляторів або водяного охолодження, щоб допомогти усунути тепло, і джерела безперебійного живлення, які забезпечують роботу сервера в разі збою живлення. Ці компоненти забезпечують більш високу продуктивність і надійність за відповідно більш високою ціною.

Розглянемо наступні мови програмування для створення веб-серверу:

* C# ASP.NET MVC,
* PHP,
* NodeJS.

C# ASP.NET MVC – фреймворк для створення веб-серверів, що реалізує шаблон Model-View-Controller. Платформа ASP.NET MVC базується на взаємодії трьох компонентів: контролера, моделі та представлення. Контролер приймає запити, опрацьовує вхідні дані, взаємодіє з моделлю та представленням і повертає результат опрацьовки запиту. Модель представляє собою шар, що описує логіку організації даних. Представлення отримує дані із контролера та генерує елементи користувацького інтерфейсу для відображення інформації.

Для управління розміткою C# ASP.NET MVC використовує мову розмітки Razor.

Перевагами C# ASP.NET MVC є:

* Розділення відповідальності (між контролером, моделлю та представленням);
* Покращена здатність до тестування;
* Відповідність протоколу НТТР;
* Підвищена гнучкість та конфігурація під різні цілі.

Недоліками C# ASP.NET MVC є:

* Обмеження у мові розмітки;
* Обмеження у використані ОС для запуску та розробки серверу;
* Відсутність безкоштовних можливостей розмістити сервер на хостинг-машині.

PHP – скриптова мова програмування створена для генерації HTML-сторінок на стороні серверу. 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.

Перевагами РНР є:

* Наявність інтерфейсів до багатьох баз даних;
* Традиційність;
* Наявність сирцевого коду та безкоштовність;
* Свобода у виборі мови розмітки;
* Ефективність;
* Наявність безкоштовних хостингів.

Недоліками РНР є:

* Незручність дизайну мови
* Змінні з символом «$»
* Складні назви поширених функцій (html\_entity\_decode, mysql\_select\_db, nl2br тощо)
* Не підтримується Unicode в версіях до 6.0
* Непередбачуваність нових версій PHP.
* Змішування логіки представлення та контролеру з моделлю

NodeJS – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Node.js призначений для відокремленого виконання високопродуктивних мережних застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм.

Перевагами NodeJS є:

* Асинхронна однопотокова модель виконання запитів;
* Неблокуючий ввід/вивід;
* Система модулів CommonJS;
* Рушій JavaScript Google V8;
* Ізоморфність коду;
* Свобода у виборі мови розмітки;
* Наявність безкоштовних хостинг-сервісів.

Недоліками NodeJS є:

* Нетрадиційність.

Отже оцінюючи усі переваги та недоліки серверних мов програмування було вибрано мову NodeJS.

### 2.3 Аналіз існуючих баз даних

База даних – сукупність [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96_(%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%82%D0%B5%D1%85%D0%BD%D1%96%D0%BA%D0%B0)), організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти.

Дані у базі організовують відповідно до моделі організації [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96). Таким

чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в [інформаційних системах](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). На даний час [застосунки](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) для роботи з базами даних є одними з найпоширеніших прикладних програм.

У сучасних мережевих інформаційних системах для роботи із загальною базою даних використовують архітектуру [«клієнт-сервер»](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). При цьому в мережі розміщують сервер баз даних. Ним виступає [комп'ютер](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) (або комп'ютери), який містить бази даних, [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) та пов'язане з ними програмне забезпечення, і налаштований для надання користувачам інформаційної системи доступу до бази даних.

Сучасні [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) ([MySQL](https://uk.wikipedia.org/wiki/MySQL), [PostgreSQL](https://uk.wikipedia.org/wiki/PostgreSQL), [Microsoft SQL Server](https://uk.wikipedia.org/wiki/Microsoft_SQL_Server) та інші) працюють відповідно до цієї архітектури. Сервер баз даних, як правило, є достатньо потужною багатопроцесорною системою, яка використовує масиви дисків [RAID](https://uk.wikipedia.org/wiki/RAID) для підвищення надійності зберігання даних. Використання дискових масивів [RAID](https://uk.wikipedia.org/wiki/RAID) дозволяє відновити дані, навіть якщо один з дисків вийшов з ладу.

Первинним призначенням бази даних є зберігання масивів даних. Але їх широко використовують і для збереження адміністративної інформації та спеціалізованих даних, наприклад, для інженерних даних чи для економічних моделей. Прикладами використання баз даних можуть бути:

* автоматизовані системи обліку;
* реєстри та каталоги;
* геоінформаційні системи.

Розглянемо деревоподібну структуру представлення інформації. Деревоподібна (ієрархічна) структура, або дерево – це зв’язний неорієнтований граф, що не містить циклів, тобто петель з замкнутих шляхів.

Як правило, при роботі з деревом виділяють будь-яку конкретну верхівку (початок) та визначають її як коріння дерева.

Недоліком даної моделі для предметної області база даних є надлишковість даних.

Більш широкі можливості для користувача забезпечує мережева модель бази даних, яка є узагальненням ієрархічної моделі і дозволяє відображати відношення між типами записів «багато-до-багатьох», завдяки чому, можна позбутись надлишковості, тобто дублювання сутностей

Саме тому використання мережевої моделі даних є прийнятним, але велика кількість категорій, а також особливості семантичних зв’язків будуть зумовлювати незручності при поданні даних.

Сучасний рівень розвитку систем управління базами даних, що організовані за мережевою моделлю, є досить низьким в порівнянні з реляційними СУБД.

В деяких випадках зростання ієрархічної та мережевої бази даних може призвести до порушення логічної організації даних. Недоліки ієрархічної та мережевої моделі даних привели до появи нової реляційної моделі.

У реляційній моделі даних об'єкти і взаємозв'язки між ними представляються за допомогою таблиць. Взаємозв'язки також подаються як об'єкти. Кожна таблиця представляє один об'єкт і складається з рядків і стовпців.

Перевагами реляційної моделі є незалежність від фізичного рівня представлення, зручність і розуміння організації даних користувачами, можливість розширення бази приєднанням нових елементів й записів без зміни при цьому існуючих підсхем та прикладних програм.

Реляційна модель даних є максимально прийнятною для предметної області, так як структура моделі повністю відповідає реляційним представленням.

Об'єктно-орієнтована база даних - база даних, яка створюється і використовується в середовищі СУБД, яке засновано на принципах об'єктно-орієнтованого підходу і підтримує об'єктну модель даних. Об'єктна база даних являє собою сукупність взаємозв'язаних об'єктів, які відповідають певній схемі.

Після аналізу основних моделей даних, для розробки веб-системи було обрано реляційну модель даних. Саме ця модель має ряд переваг, таких як:  незалежність від фізичного рівня представлення, зручність і розуміння організації даних користувачами, максимальна гнучкість при обробці непередбачених запитів, можливість розширення бази приєднанням нових елементів, записів без зміни при цьому існуючих підсхем та прикладних програм.

Через такі недоліки ієрархічної моделі, як: надлишковість зберігання інформації, так як ієрархічні структури не підтримують взаємозв'язки Б:Б; строгу ієрархічну впорядкованість, яка ускладнює процедури включення та вилучення записів; вилучення вихідних вузлів призводить до вилучення відповідних їм породжених , що вимагає особливої обережності; ускладняється доступ до даних , які лежать на більш низьких рівнях ієрархії, так як кореневий вузол завжди є головним, а доступ до любого породженого вузла може здійснюватися через вихідний, ієрархічна модель не може бути використана для розробки інформаційно-довідкової веб-системи.

А недоліки використання мережевої моделі даних полягають в складності по відношенню до ієрархічних структур. Прикладному програмісту часто необхідно знати логічну структуру бази даних. При реорганізації бази даних не виключається в ряді випадків втрата незалежності даних, що не припустимо в даному випадку.

У ієрархічній моделі варто розглянути лише один недолік який спонукає відмовитись від використання даної моделі даних, це - дублювання даних, тобто якщо об'єкти предметної області мають зв'язки типу «один-до-одного» або «один-до-багатьох», то ієрархічна структура дає змогу зображувати дані без дублювання; проте, якщо є зв'язки типу «багато-до-багатьох», то дублювання даних під час відображення в ієрархічній моделі неминуче.

Серед переваг мережевої моделі даних можна виділити наступні характеристики: можливість ефективної реалізації за показниками затрат пам’яті та оперативності; порівняно з ієрархічною моделлю, мережева модель надає великі можливості у створенні довільних зв’язків. Проте мережева модель має декілька недоліків: велика складність схеми БД, а також складність обробки інформації для звичайного користувача.

Реляційна модель має декілька вагомих переваг, а саме: простота і доступність для розуміння користувачем. Суворі правила проектування, які базуються на математичному апараті. Повна незалежність даних, зміни в прикладній програмі при зміні реляційної БД мінімальні. Для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

Серед недоліків реляційної моделі можна виділити складність представлення предметна області у вигляді «таблиць», а в результаті логічного проектування з'являється множина «таблиць». Це призводить до труднощів розуміння структури даних.

Отже так як реляційна модель має переваги перед розглянутими моделями даних, та немає недоліків – будемо використовувати реляційну модель даних.

MongoDB – документо-орієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць.

MongoDB написана на C++, тому її легко перенести на найрізноманітніші платформи.

Система зберігання даних в MongoDB представляє набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Всі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. І якщо основний вузол з якихось причин виходить з ладу, то один з вторинних вузлів стає головним

Основні можливості MongoDB:

* Документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних)
* Досить гнучка мова для формування запитів
* Динамічні запити
* Повна підтримка індексів
* Профілювання запитів
* Швидкі оновлення «на місці»
* Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео
* Журналювання операцій, що модифікують дані в БД
* Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг.

### 2.4 Редактор коду Visual Studio Code

Visual Studio Code – редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кросплатформенної розробки веб-і хмарних додатків.

Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для

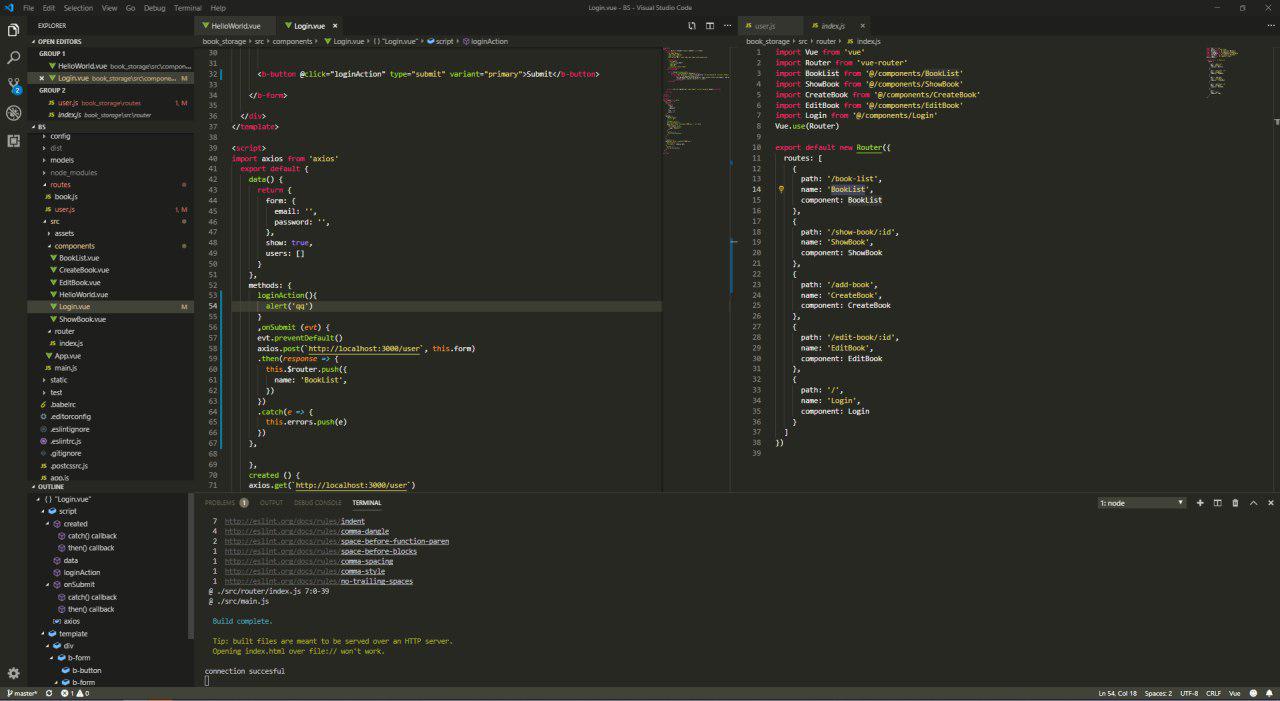
кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації

Git – розподілена система керування версіями файлів та спільної роботи.

IntelliSense – технологія автодоповнення Microsoft, найбільш відома в Microsoft Visual Studio. Дописує назву функції при введенні початкових літер. Крім прямого призначення, IntelliSense використовується для доступу до документації та для усунення неоднозначності в іменах змінних, функцій і методів, використовуючи рефлексію.

Рефакторинг – перетворення програмного коду, зміна внутрішньої структури програмного забезпечення для полегшення розуміння коду і легшого внесення подальших змін без зміни зовнішньої поведінки самої системи.

Нижче наведений зовнішній вигляд вікна програми Visual Studio Code(рисунок 2.1).

Рисунок 2.1 – Зовнішній вигляд вікна програми Visual Studio Code

## 2.5 Виконання технічних вимог

Для виконання проекту рекомендовано наступні параметри комп’ютера:

* операційна система Windows XP SР3, Vista SP2, 7, 8;
* оперативна пам'ять об'ємом 2Гб DDR3;
* процесор AMD або Intel з тактовою частотою не менше 1.8ГГц;
* відеоадаптер з об'ємом пам'яті не менше 256Мб.

# 3 Розробка сервісу автоматизації роботи бібліотеки коледжу

# 3.1 Обґрунтування потреби веб-сайту

Сервіс автоматизації роботи бібліотеки коледжу «BookLibrary» являє собою спеціалізований Web-сайт, який призначений для перегляду книг, оформлення заявки на видачу, завантаження методичних матеріалів та належить коледжу ВК НУХТ.

Даний Web-сайт надає зручний інтерфейс, авторизацію та реєстрацію користувачів, перегляд, редагування, додавання та видалення книг, оформлення заявки на видачу книги, можливість залишити відгук про книгу, та можливість завантаження методичних матеріалів.

«BookLibrary» потребує значно менших витрат на утримання та організацію роботи, оскільки у ньому значно обмеженіша матеріально-технічна база

Основними вимогами, які ставляться користувачами до Web-сайту є:

* зрозумілий інтерфейс та зручна система навігації по сайту;
* зручна система посилань, що дозволяє оптимальним способом одержати необхідну користувачеві інформацію;
* мінімальна кількість дій користувача для завантаження чи додачі методичних матеріалів.
* легкий перегляд книг, що знаходяться в бібліотеці

## 3.2 Розробка структури сайту

Існує декілька підходів до проектування Web-сайтів, які дозволяють поетапно, крок за кроком, реалізувати проект від появи ідеї до її втілення. Вибір підходу залежить від розробників і тих цілей, які вони ставлять перед собою і створюваним проектом.

Більшість розробників використовують базові підходи, які дозволяють чітко визначити цілі сайту та описати його внутрішню структуру.

Процес проектування сайтів передбачає три етапи:

* Концептуальне проектування.
* Логічне проектування.
* Фізичне проектування.

Ці етапи слідують один за одним, хоча у деяких випадках можливий перехід до наступного етапу без закінчення попереднього. Перевагами цього підходу є простота понятійного апарату та висока результативність розробки.

На етапі концептуального проектування сайту розробник збирає необхідну інформацію, визначає цільову аудиторію, здійснює аналіз, збір і документування вимог користувачів, намічає способи їх реалізації.

Основна мета етапу логічного проектування полягає в тому, щоб упорядкувати розділи сайту, привести їх до зручного й зрозумілого вигляду. Це включає організацію інформації на сайті, побудову його структури та навігації по розділах таким чином, щоб користувачі могли легко та швидко знайти необхідну інформацію.

На етапі фізичного проектування відбувається пошук рішень та застосування різних технологій, пов’язаних із технічною реалізацією сайту. На цьому етапі розробник повинен приділяти увагу:

* технологіям, які будуть застосовуватися на сайті;
* програмному забезпеченню, що буде використовуватись;
* можливим проблемам і способам їх усунення;
* як і ким буде обновлятися інформація.

Етап фізичного проектування передбачає візуальне оформлення та тестування, виробництво і контроль якості, запуск та супровід.

Структура будь-якого сайту – це взаємозв’язки між його сторінками.

Графічне подання структури сторінки формує наочне відображення існуючих сторінок та зв’язків між ними, тобто це своєрідна карта сайту.

Створення веб-сайту починається зі створення інформаційної моделі сайту. Будь-яку веб-сторінку можна оцінити за двома параметрами: зміст та зовнішній вигляд. Зовнішній вигляд кожного сайту є унікальним.

Розглянемо загальні функції Web-сайту, які потрібні в більшості випадків. Всі їх можна поділити на кілька груп:

* пошуковий функціонал або просто пошук і підбір книг;
* демонстраційний – для представлення книг;
* інформаційний функціонал, який відповідає за інформування користувачів;
* SEO-функціонал, який полегшує просування Web-сайту в пошукових системах (можливість вказувати META-теги і т.д.);
* соціальний функціонал, наприклад, кнопки соцмереж;
* функціонал, який відповідає за завантаження та додавання методичинх матеріалів;

Структура Web-сайту розроблена таким чином, щоб користувач міг без проблем повертатися на головну чи здійснювати перехід на вкладені сторінки, щоб користувачу легко було орієнтуватися на сайті та приємно перебувати на ньому.

Структура Web-сайту «BookLibrary» показана на рисунку 3.1.

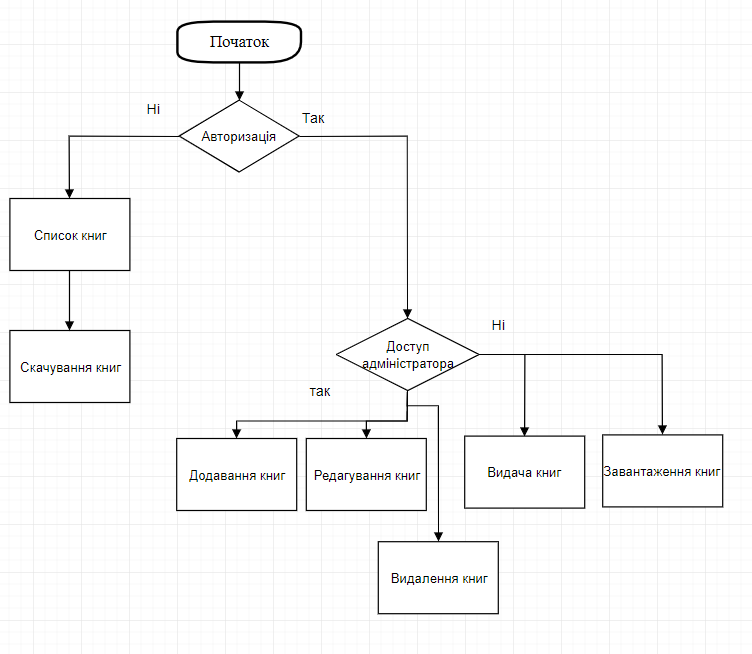


Рисунок 3.1 – Структура Web-сайту «BookLibrary»

## 3.3 Розробка головної сторінки сайту

На будь-якому сайті першою відкривається головна сторінка. У верхній частині головної сторінки зазвичай розташована так звана шапка, яку дублюють на інших сторінках сайту. Це роблять спеціально, адже ця частина відображається у вікні браузера першою і відвідувач насамперед звертає увагу на неї. Щоб забезпечити швидкий перехід до основних тематичних розділів сайту, створюють меню сайту. Вибір структури визначається особливостями завдань, що розв'язуються за допомогою веб-сайту.

Оформлення сторінки сайту починається з розробки макету. Макет сайту створюється відповідно до найкращих практик дизайну взаємодії з користувачем (UI/UX). Для створення Web-сайту «BookLibrary» обрано класичну модель сайту з навігацією у вигляді вкладок, та представлення данних у вигляді списку що розширюється.

Відповідно до обраного фреймворку розробки сайту, було вирішено використати бібліотеку Material Design - Vuetify. Вона дозволяє створювати уніфіковані інтерфейси користувача на всіх пристроях та більшості сучасних браузерів. Також використання даної бібліотеки дозволяє створити веб сайт у стилі Material Design, що є сучасним та стильним вибором у наш час.

Отже проаналізувавши дані методи побудови сторінок та специфіку даного продукту, було обрано метод побудови інтерфейсу користувача у стилі Material Design та використання списку що розширюється для представлення даних для зменшення переобтяження користувача інформацією та згідно з рекомендаціями обраного стилю. Також для створення даного інформаційного сайту використовувалися каскадні таблиці стилів, щоб надати сторінкам певного зовнішнього вигляду і деякі елементи javascript для надання сайту зручності та функціональності. Для побудови сайту було використано уніфікований шаблон, який відображає інформацію на основі запитів користувача до неї. Також було використано адаптивний веб-дизайн для уніфікованого відображення сайту на усіх пристроях та набір методів Vue.js для динамічного оновлення змісту веб сторінки без необхідності її перезавантаження. Розроблено зручний інтерфейс головного вікна, який дозволяє отримати доступ до всіх функцій передбачених сервісом (рисунок 3.2).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Додати книгу | | Редагування обілкового запису | | Реєстрація/  авторизація | Інформація про обліковий запис | Оформлення заявки про видачу книги |
| Емблема | | | | | | |
| Список книг | Вікно інформації | | Додаткова інформація | | Теги | |
| Футер сайту (Додаткове меню) | | | | | | |

Рисунок 3.2 – Структура головного вікна

Після оформлення стилів усіх елементів отримуємо результат оформлення головної сторінки сервісу, який зображено на рисунку 3.3.

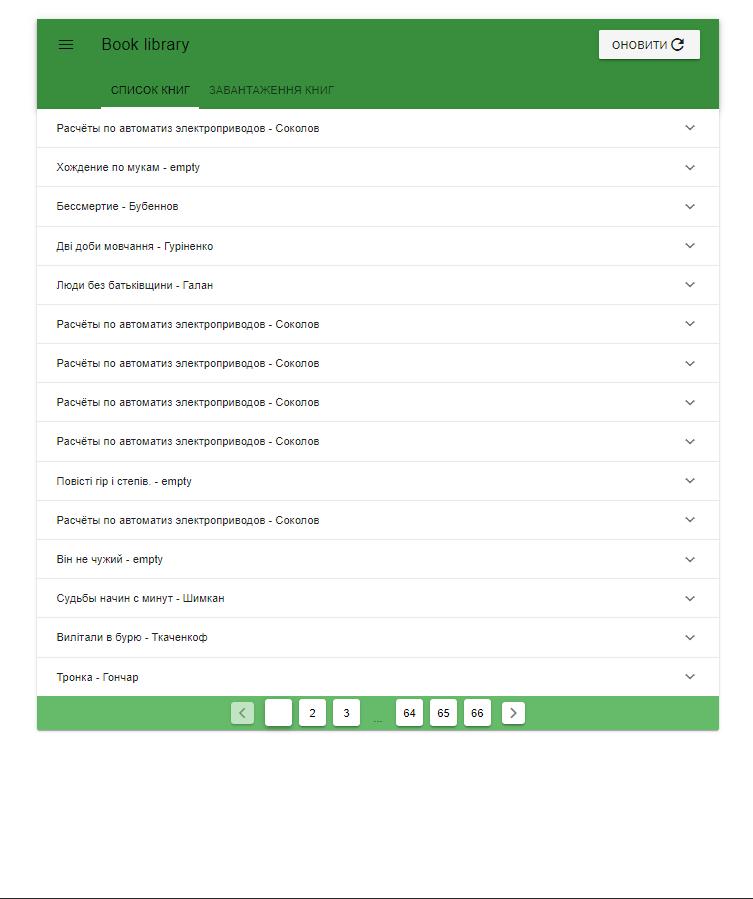


Рисунок 3.3 – Зовнішній вигляд головної сторінки сайту

Сайт складається з функціональної панелі, яка розділена на вкладки за змістом. В лівому верхньому куті панелі наявна кнопка виклику області навігації, що містить у собі кнопку авторизації. Після натиснення кнопки «Ввійти» з’являється діалогове вікно де необхідно ввести логін та пароль (рисунок 3.4).

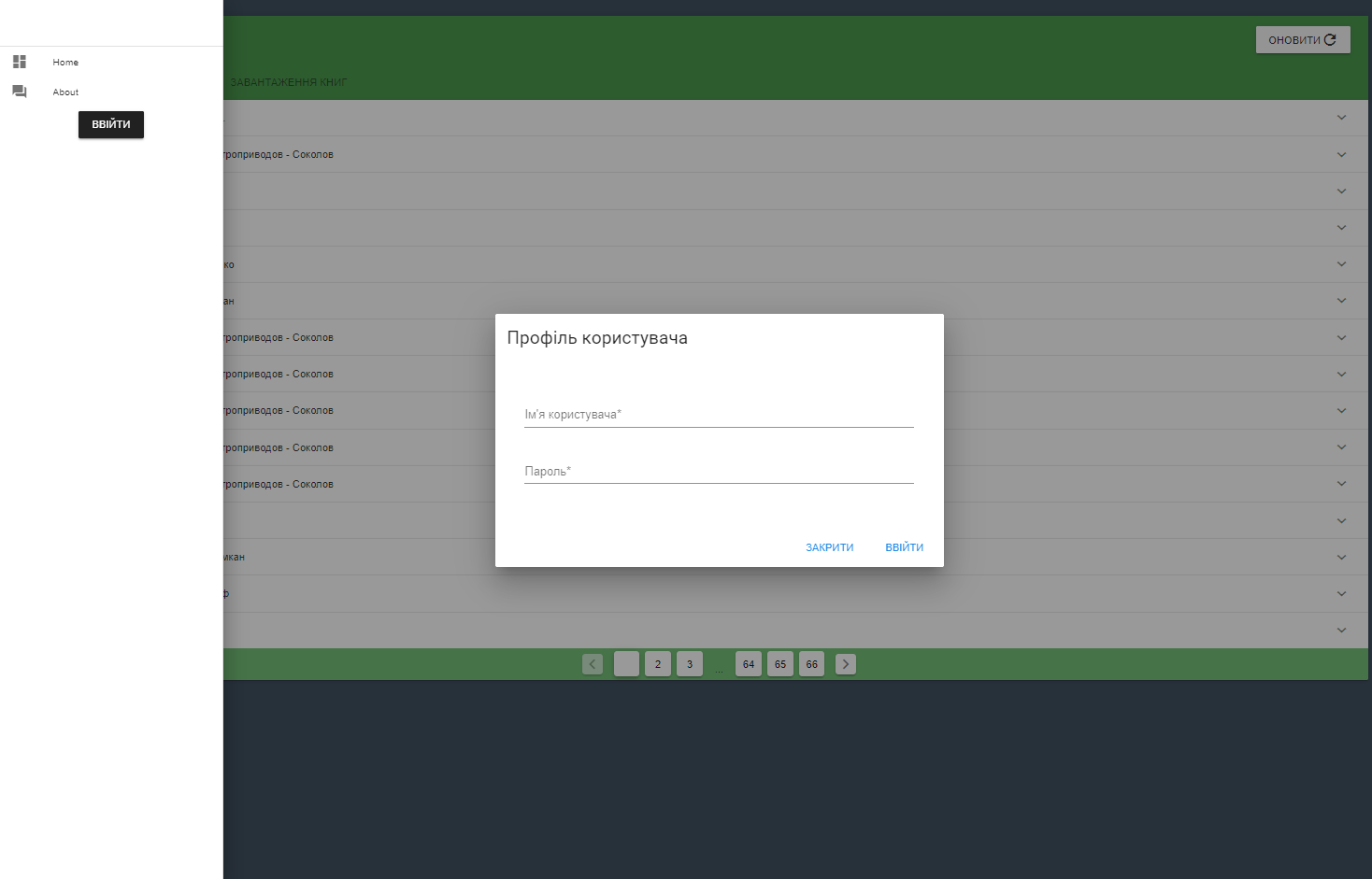


Рисунок 3.4 – Діалогове вікно авторизації

Після успішної авторизації відобразиться вкладка з видачою книг, а також можливість видати та відредагувати книгу (рисунок 3.5).



Рисунок 3.5 – Зображення елементу списку книг після авторизації

Код розмітки головної сторінки сайту наведено в додатку Д.

Реалізовано можливість завантаження власних файлів та книг на сервер з подальшою можливістю їх скачування користувачами веб сайту. Обмеження на завантаження встановлено у розмірі не більше 10 Мб на один файл. Обмеження на скачування відсутнє. Вкладка завантаження зображена на рисунку 3.6

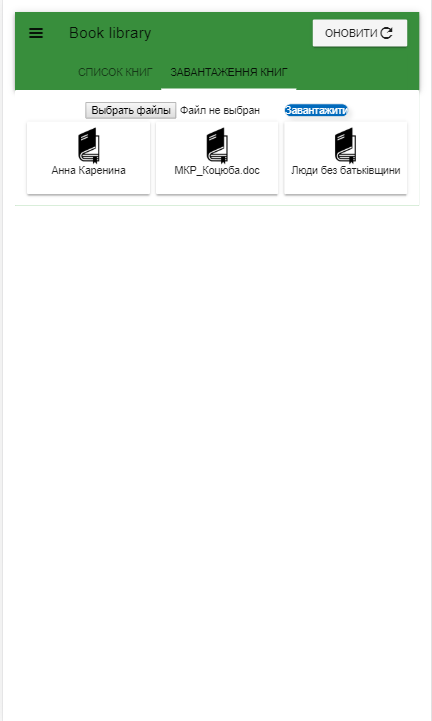


Рисунок 3.6 ­– Вкладка завантаження та скачування книг користувачів

Вкладка з списком виданих книг користувачеві зображений на рисунку 3.7.

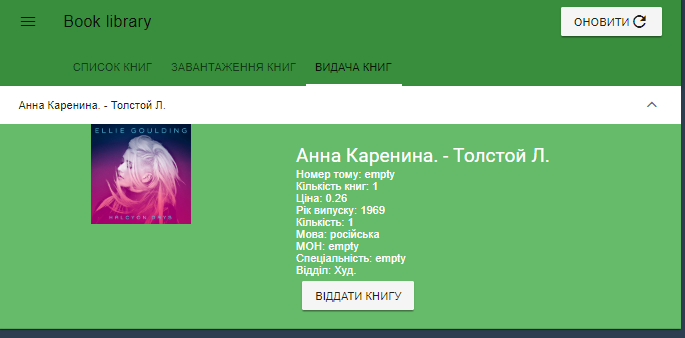


Рисунок 3.7 ­ Вкладка з списком виданих книг

Розробка сайту з використанням Vue.js та Vuetify Material Design є сучасним та ефективним способом розробки реактивних та адаптивних веб-сайтів. При розробці було дотримано загальноприйнятих норм стилю та рекомендацій до створення користувацьких інтерфейсів. Розроблений сайт відповідає вимогам безпеки, швидкості завантаження та ефективності роботи в усіх сучасних веб браузерах.

## 3.4 Розробка та дизайн основних складових шапки сайту

Сайт складається з таких розділів:

* завантаження книг;
* відкрити книгу;
* видача книг;
* кнопка виклику області навігації;
* список книг.

Візуальне оформлення сайту, його сприйняття, графічний інтерфейс – це перше, з чим стикається користувач. Ще до оцінки зручності сайту користувачі бачать, як він виглядає. Розробка візуального оформлення будь-якого сайту – це творчий процес. На цій стадії створюються всі елементи дизайну сайту відповідно до стилю подачі інформації і загальної концепції, сформульованої у креативному брифі проекту.

Кольорова гама сайту витримана у діловому, стриманому і лаконічному, стилі, що підвищує комфорт користувача.

Оскільки дизайн елементів веб-сторінки визначають таблиці стилів CSS. Їх можна розмістити як усередині html документа, так і використати зовнішні CSS-файли, але у даному проекті вирішено використати кольорове оформлення, яке включає у себе бібліотека Vuetify. Вона містить спокійні та лаконічні кольори які не викликають роздратування при довготривалій роботі з сайтом. Написання усього CSS коду проводилось з дотриманням загальних правил та відповідно до міжнародних стандартів.

Враховуючи специфіку сайту, необхідно забезпечити постійний доступ користувача до пошуку та списку книг.

Пошук організовується за допомогою JavaScript. JavaScript – код елементу «Пошук» наведено нижче:

<b-form-group label-cols-sm="3" label="Пошук" class="mb-0">

<b-input-group>

<b-form-input v-model="filter" placeholder="Шукати"></b-form-input>

</b-input-group>

</b-form-group>

</div>

<b-table id = "my-table" :filter="filter" @filtered="onFiltered" striped hover :items="books" :fields="fields" :per-page="perPage"

:current-page="currentPage">

<template slot="actions" scope="row">

<b-btn size="sm" @click.stop="details(row.item)">Details</b-btn>

</template>

</b-table>

<ul v-if="errors && errors.length">

<li v-for="error of errors">

{{error.message}}

</li>

</ul>

</b-col>

onFiltered(filteredItems) {

// Trigger pagination to update the number of buttons/pages due to filtering

this.totalRows = filteredItems.length

this.currentPage = 1

}

Пошук на сайті реалізований наступним чином – у полі пошуку користувач вводить запит, після натискання клавіші «Знайти» формується запит до бази даних і здійснюється пошук по ній; після завершення пошуку за релевантністю виводяться результати, якщо пошук не дав результатів з’являється відповідне попередження і пропозиція по-іншому сформувати запит. Блок-схема роботи пошуку представлена на рисунку 3.8.



Рисунок 3.8 ‑ Блок-схема роботи механізму пошуку

На сайті є можливість завантаження та скачування даних з серверу

Це реалізовано наступним чином:

app.post('/uploadfile', upload.single('picture'), (*req*, *res*) *=>* {

*var* fl = fs.readFileSync(req.file.path);

*var* encode\_file = fl.toString('base64');

*var* finalImg = {

originalFileName: req.file.path,

contentType: req.file.mimetype,

file: new Buffer(encode\_file, 'base64')

};

db.collection('mycollection').insertOne(finalImg, (*err*, *result*) *=>* {

*console*.log(result)

if (err) return *console*.log(err)

*console*.log('saved to database')

res.redirect('/')

})

})

app.get('/files', (*req*, *res*) *=>* {

db.collection('mycollection').find().toArray((*err*, *result*) *=>* {

*const* imgArray= result.map(*element* *=>* element.\_id);

*console*.log(imgArray);

if (err) return *console*.log(err)

res.send(imgArray)

})

});

app.get('/file/:id', (*req*, *res*) *=>* {

*var* filename = req.params.id;

db.collection('mycollection').findOne({'\_id': ObjectId(filename) }, (*err*, *result*) *=>* {

if (err) return *console*.log(err)

res.contentType(result.contentType);

res.send(result.image.buffer);

})

})

Завантаження файлів відбувається шляхом збереження файлу, що обраний для завантаження у локальній файловій системі хостинга, на якому розгорнyта серверна частина сайту.

Після обрання файлу для завантаження, серверне API зберігає даний файл у необхідному форматі в папку «uploads» для подальшого використання. Також інформація про файл зберігається у базі даних сайту. Інформація про файл включає в себе: оригінальну назву файлу, його розмір, назву файлу у локальному сховищі та сам файл закодований у форматі base64 для забезпечення безпеки зберігання даних.

Зразок об’єкта який записаний в базі даних

{

"\_id": {

"$oid": "5cfe48e855d4dd3258193bd1"

},

"contentType":"application/msword",  
 "originalFileName": "uploads\\msword-1560168901988",

"file": "<Binary Data>"

}

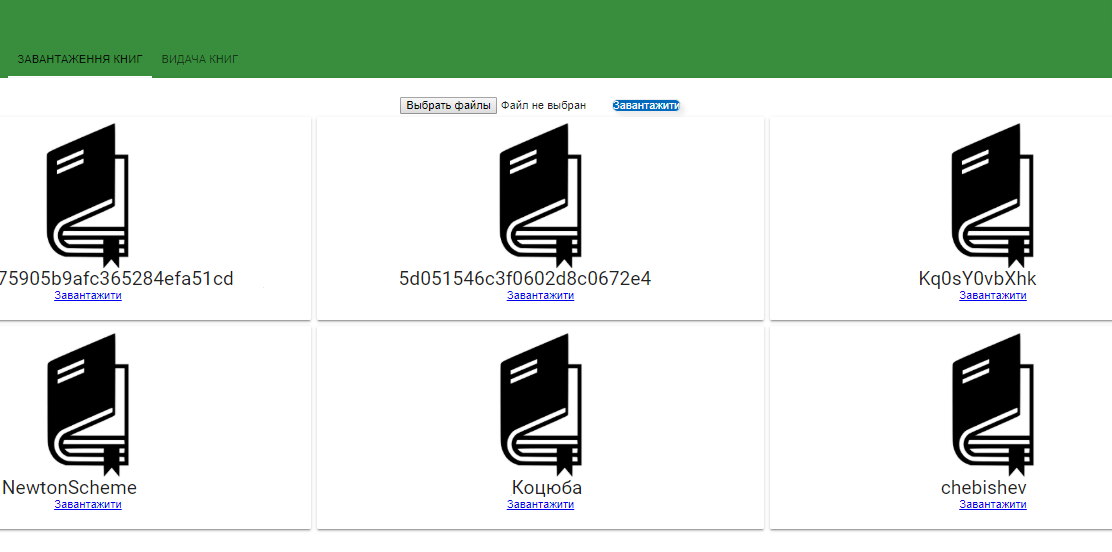


Рисунок 3.9 ­– Інтерфейс завантаження файлів

Після завантаження файли доступні для скачування з вкладки з завантаженням.

Видача існуючих книг відбувається за допомогою серверного АРІ, які видаляють книгу з загального списку книг та додають їх в список книг, які прив’язані до певного користувача. Після повернення книги, відповідний запис додається назад у загальний список книг.

*var* router = require('express').Router();

*var* mongoose = require('mongoose');

*var* Book = mongoose.model('Book');

*var* User = mongoose.model('User');

*var* auth = require('../auth');

router.param('book', *function*(*req*, *res*, *next*, *slug*) {

Book.findOne({ slug: slug})

.populate('owner')

.then(*function* (*book*) {

if (!book) { return res.sendStatus(404); }

req.book = book;

return next();

}).catch(next);

});

Наведений вище фрагмент програмного коду відповідає за встановлення параметрів взаємодії серверної частини з базою даних MongoDB.

*router.get('/', auth.optional, function(req, res, next) {*

*var query = {};*

*var limit = 0;*

*var offset = 0;*

*if(typeof req.query.limit !== 'undefined'){*

*limit = req.query.limit;*

*}*

*if(typeof req.query.offset !== 'undefined'){*

*offset = req.query.offset;*

*}*

*if( typeof req.query.tag !== 'undefined' ){*

*query.tagList = {"$in" : [req.query.tag]};*

*}*

*Promise.all([*

*req.query.owner ? User.findOne({username: req.query.author}) : null,*

*]).then(function(results){*

*var owner = results[0];*

*if(owner){*

*query.owner = owner.\_id;*

*}*

*return Promise.all([*

*Book.find(query)*

*.limit(Number(limit))*

*.skip(Number(offset))*

*.sort({createdAt: 'desc'})*

*.populate('owner')*

*.exec(),*

*Book.count(query).exec(),*

*req.payload ? User.findById(req.payload.id) : null,*

*]).then(function(results){*

*var books = results[0];*

*var booksCount = results[1];*

*var user = results[2];*

*return res.json({*

*books: books.map(function(book){*

*return book.toJSONFor(user);*

*}),*

*booksCount: booksCount*

*});*

*});*

*}).catch(next);});*

Наведений фрагмент програмного коду відповідає за відображення даних які отримані через запит до таблиці books.

*router.post('/', auth.required, function(req, res, next) {*

*User.findById(req.payload.id).then(function(user){*

*if (!user) { return res.sendStatus(401); }*

*var book = new Book(req.body.book);*

*book.owner = user;*

*return book.save().then(function(){*

*console.log(book.owner);*

*return res.json({book: book.toJSONFor(user)});*

*});*

*}).catch(next);*

*});*

*router.get('/:book', auth.optional, function(req, res, next) {*

*Promise.all([*

*req.payload ? User.findById(req.payload.id) : null,*

*req.book.populate('owner').execPopulate()*

*]).then(function(results){*

*var user = results[0];*

*return res.json({book: req.book.toJSONFor(user)});*

*}).catch(next);*

*});*

Дані серверні методи відповідають за додавання нових книг та отримання інформації про кожну окрему книгу.

*router.put('/:book', auth.required, function(req, res, next) {*

*User.findById(req.payload.id).then(function(user){*

*if(req.book.owner.\_id.toString() === req.payload.id.toString()){*

*if(typeof req.body.book.title !== 'undefined'){*

*req.book.title = req.body.book.title;*

*}*

*if(typeof req.body.book.body !== 'undefined'){*

*req.book.body = req.body.book.body;*

*}*

*if(typeof req.body.article.tagList !== 'undefined'){*

*req.book.tagList = req.body.book.tagList*

*}*

*req.book.save().then(function(book){*

*return res.json({book: book.toJSONFor(user)});*

*}).catch(next);*

*} else {*

*return res.sendStatus(403);*

*}*

*});*

*});*

*router.delete('/:book', auth.required, function(req, res, next) {*

*User.findById(req.payload.id).then(function(user){*

*if (!user) { return res.sendStatus(401); }*

*if(req.book.owner.\_id.toString()== req.payload.id.toString()){*

*return req.book.remove().then(function(){*

*return res.sendStatus(204);*

*});*

*} else {*

*return res.sendStatus(403);*

*}*

*}).catch(next);*

*});*

*module.exports = router;*

Наведені серверні методи відповідають за редагування існуючих книг та їх видалення.

## 3.5 Встановлення локального серверу

Прикладний програмний інтерфейс (API) ­ набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено ­ це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Взаємодія клієнтської та серверної частини програмного забезпечення буде здійснюватись за допомогою API. Це дозволить зменшити кількість розроблюваного коду та розмежує клієнтську частину з серверною. Набір методів, через які буде взаємодіяти сервер та клієнтська програма описано у таблиці 3.1.

Таблиця 3.1 – Набір методів сервера

|  |  |  |
| --- | --- | --- |
| Метод | HTTP запит | Опис |
| upload | POST/upload/ | Завантаження файлу на сервер |
| download | get /upload /download/id | Скачування файлу з серверу |
| Books/ | GET /Books / | Отримання списку книг |
| gaveBooks | GET / gaveBooks | Отримання списку виданих книг |
| search | GET/ search /params | Пошук інформації за даними параметрами |
| setUserSettings | POST /setUserSettings/userID | Встановлення налаштувань користувача |

## 3.6 Підключення бази даних

Для зберігання інформації сайту «Book Library» було обрано базу даних MongoDB, через її переваги та швидкість роботи на ряду з зручністю організації зберігання даних. Для підключення серверної частини до бази даних було використано бібліотеку mongoose. Створено моделі даних для усіх необхідних таблиць, таких як: users, books, gavebooks, uploads.

Модель даних books включає в себе:

* slug: {type: String, lowercase: true, unique: true} унікальний ідентифікатор на базі javascript функції та назви книги ;
* cardId: String, - номер книги;
* bookNm: String – назва книги;
* author: String - автор книги;
* bklang: String – мова книги;
* tomeNm: String – том книги;
* quants: String – кількість доступних книг;
* bk\_MON: String, - МОН;
* speclt: String - спеціалізація;
* depart: String - відділ;
* bKyear: String – рік випуску;
* prices: String - ціна;
* img: String – картинка обкладинки.

Модель даних users включає в себе:

* username: ім’я користувача;
* hash: String, - захешований пароль користувача;
* salt: String – ключ для розшифровки пароля.

Модель даних Upload включає в себе:

* slug: унікальний ідентифікатор на базі javascript функції та назви книги;
* bookNm: String, - назва книги;
* path: String – шлях до файлу на сервері.

## 3.7 Авторизація та реєстрація користувача

## Авторизація користувача здійснюється за допомогою JWT токена. JSON Web Token це стандарт токена доступу на основі JSON, стандартизованого в RFC 7519. Використовується для верифікації тверджень.

JWT складається з трьох частин: заголовка, вмісту і підпису. Заголовок

Заголовок (Header) це JSON елемент, який описує до якого типу токену належить даний і які методи шифрування використовувались.

Реалізація jwt аутентифікації в проекті Book Library здійснена стандартним методом: після надсилання запиту на сервер генерується Bearer-Token, який в подальшому буде використовуватись для зберігання сесії користувача. Фрагмент коду що відповідає за генерацію jwt токена наведений нижче:

*class HandlerGenerator {*

*login (req, res) {*

*let username = req.body.username;*

*let password = req.body.password;*

*let mockedUsername = 'admin';*

*let mockedPassword = 'password';*

*if (username && password) {*

*if (username === mockedUsername && password === mockedPassword) {*

*let token = jwt.sign({username: username},*

*config.secret,*

*{ expiresIn: '24h'*

*}*

*);*

*res.json({*

*success: true,*

*message: 'Authentication successful!',*

*token: token*

*});*

*} else {*

*res.send(403).json({*

*success: false,*

*message: 'Incorrect username or password'*

*});*

*}*

*} else {*

*res.send(400).json({*

*success: false,*

*message: 'Authentication failed! Please check the request'*

*});*

*}*

*}*

*index (req, res) {*

*res.json({*

*success: true,*

*message: 'Index page'*

*});}}*

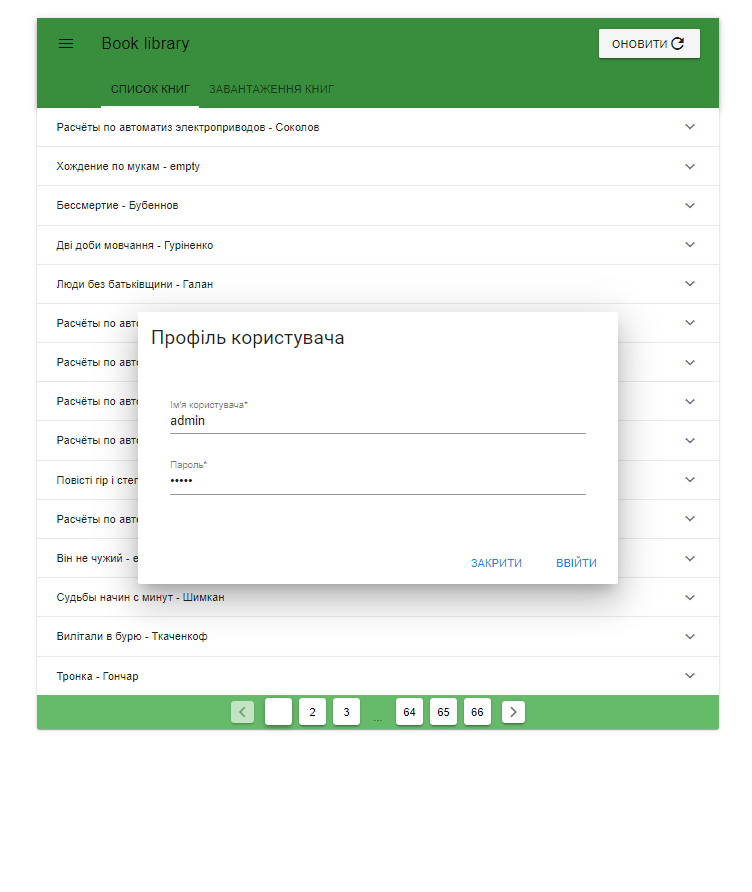
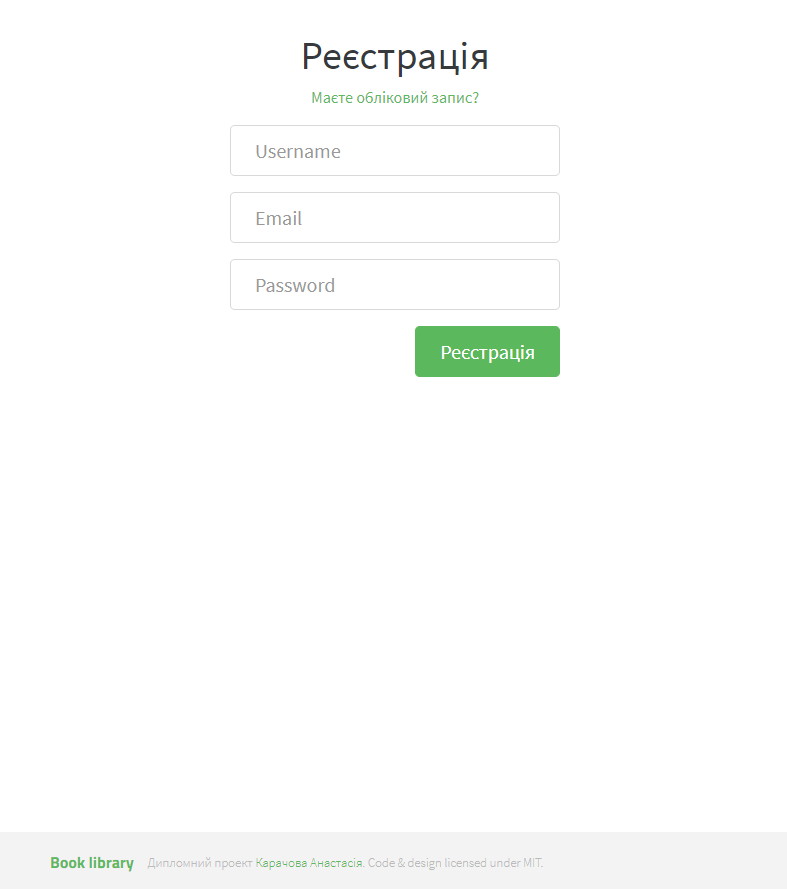
Нижче наведена форма авторизації користувача(рисунок 3.10).

Рисунок 3.10 ­– Форма авторизації користувача

Форма реєстрація має наступний вигляд(рисунок 3.11).

Рисунок 3.11 ­– Форма реєстрації користувача

Для реєстрації використовується passport.js и jwt токен. Логін і пароль записуються в таблицю users. А потім використовується при авторизації.

Passport.js - міддлвер аутентифікації для Node.js. Надзвичайно гнучкий і популярний, Passport можна використовувати у будь-якому застосуванні на базі Express. Набір з різними видами аутентифікації, використовуючи ім'я користувача і пароль, Facebook, Twitter та ін. Ми будемо використовувати Passport для управління сесіями.

Процес Аутентифікації у Node.js:

Мета полягає в тому, щоб реалізувати наступний процес аутентифікації у нашому застосунку:

* Користувач вводить ім'я користувача і пароль.
* Застосунок перевіряє їх на відповідність.
* Якщо вони відповідають один одному, застосунок посилає заголовок jwt, який буде використовуватися для аутентифікації на наступних сторінках.

Коли користувач відвідує сторінки з одного домена, раніше встановлений jwt токен буде додаватися до всіх запитів.

Код, що відповідає за реєстрацію користувача наведений нижче.

*var mongoose = require('mongoose');*

*var router = require('express').Router();*

*var passport = require('passport');*

*var User = mongoose.model('User');*

*var auth = require('../auth');*

*router.put('/user', auth.required, function(req, res, next){*

*User.findById(req.payload.id).then(function(user){*

*if(!user){ return res.sendStatus(401); }*

*if(typeof req.body.user.username !== 'undefined'){*

*user.username = req.body.user.username;*

*}*

*if(typeof req.body.user.email !== 'undefined'){*

*user.email = req.body.user.email;*

*}*

*if(typeof req.body.user.bio !== 'undefined'){*

*user.bio = req.body.user.bio;*

*}*

*if(typeof req.body.user.image !== 'undefined'){*

*user.image = req.body.user.image;*

*}*

*if(typeof req.body.user.password !== 'undefined'){*

*user.setPassword(req.body.user.password);*

*}*

*return user.save().then(function(){*

*return res.json({user: user.toAuthJSON()});*

*});*

*}).catch(next);*

*});*

*module.exports = router;*

## 3.8 Адаптивний веб-дизайн

Адаптивний веб-дизайн – дизайн веб-сторінок, що забезпечує оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки.

Метою адаптивного веб-дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету (від стаціонарних ПК до мобільних телефонів). За технологією адаптивного веб-дизайну не потрібно створювати окремі версії веб-сайту. Один сайт може працювати на всьому спектрі пристроїв.

Переваги адаптивного сайту:

* Адаптивний сайт на мобільному пристрої завантажується швидко, оскільки всі складові елементи спрощені і їх розмір менший.
* Зниження відсотка відмов. Не секрет, що сайт, який завантажується довго, особливо на смартфоні, відвідувач покине швидше. Тому, адаптивний дизайн сайту – це перший крок до залучення цільової аудиторії.
* Економія ресурсів. Вартість розробки мобільного додатку набагато вища, ніж адаптивного дизайну. Як результат, Ви економите фінансові кошти.
* Можливість управління всіма версіями сайту з одного кабінету.

Типи адаптивного дизайну:

* «Гумовий» сайт – програмісти при розробці задають ширину колонок в процентах, а не в пікселях. Така хитрість використовується давно. Але з ростом кількості пристроїв з різною роздільною здатністю і розміром екрана, стає все складніше нею користуватися. Адже досить складно прорахувати потрібні пропорції для колонок, щоб компоненти сайту відображалися правильно;
* адаптивний дизайн сайту – коли розробляється кілька окремих дизайнів одного ресурсу для різних типів «гаджетів» (ПК, планшета, смартфона). Під час заходу на сайт, сервер визначає, з якого типу пристрою був здійснений візит і перенаправляє на відповідну версію сайту;
* “чуйний” дизайн веб-ресурсу – поєднання двох вищеописаних методик, які передбачають «підгонку» сайту під роздільну здатність екрану і в той же час зміну місця розташування і розміру картинок, тексту, колонок, щоб користування веб-ресурсом було максимально комфортним. Це один з найсучасніших методів, який має ряд недоліків і в даний момент на стадії доопрацювання.

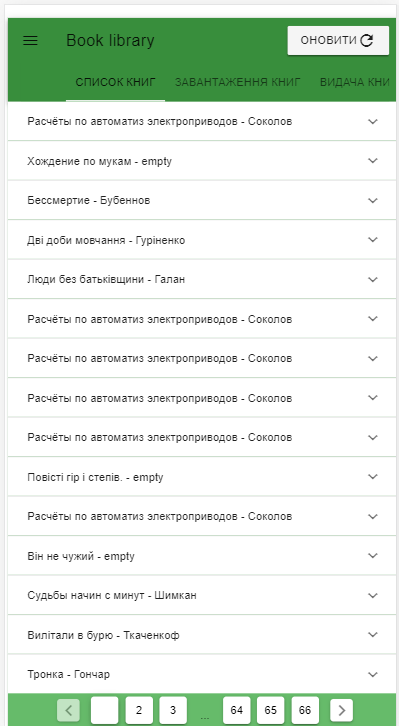
Внаслідок адаптивного веб-дизайну ми можемо легко переглядати інформацію та зручно переміщуватись по сайту(рисунок 3.12).

Рисунок 3.12 – Перегляд списку книг з мобільного пристрою

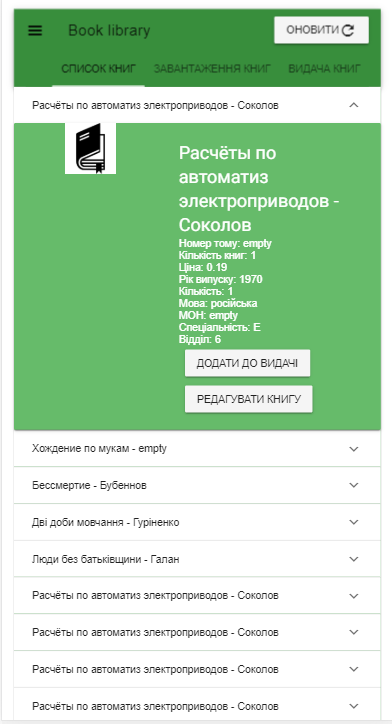
Нижче наведено скріншот перегляду інформації про книгу з мобільного пристрою(рисунок 3.13)

Рисунок 3.13 – Перегляд інформації про книгу з мобільного пристрою

# 4 Економічна частина

## 4.1 Розрахунок капітальних витрат

Кошторис витрат на розробку програмного продукту може включати розрахунок таких основних статей витрат.

4.1.1 Основна заробітна плата розробників, яка розраховується за формулою

[грн], (4.1)

де М – місячний посадовий оклад конкретного розробника, грн;

– число робочих днів у місяць. Приблизно = 21 – 22;

*–* число днів роботи (дослідника).

Зроблені розрахунки записані у таблицю 4.1.

Таблиця 4.1 – Розрахунок основної заробітної плати

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Найменування посади | Місячний посадовий оклад,грн. | Оплата за робочий день | Число днів роботи | Витрати на заробітну плату |
| 1 | Керівник проекту |  | 338,1 | 3 |  |
| 2 | Технік-програміст |  | 250,93 | 20 |  |
| Всього | | | | | 6033,33 |

4.1.2 Додаткова заробітна плата розробників

Розраховується як 10 – 12% від основної заробітної плати розробників (

=

4.1.3 Нарахування на заробітну плату робітників єдиного соціального внеску

4.1.4 Амортизація персонального комп’ютера

Амортизація відрахувань можуть бути розраховані за формулою:

де Ц – балансова вартість персонального комп’ютера, грн;

– норма амортизації, % за рік = 20%;

Т – термін використання комп’ютера, цілі місяці;

4.1.5 Витрати на матеріали

Витрати на матеріали, що були використані на розробку програмного продукту – можуть бути обмежені вартістю декількох дисків, вартість роздруківки програми на принтері, вартості паперу тощо, заносимо до таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на матеріали

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п.п. | Найменування матеріалу, марка, тип, сорт | Ціна, грн. | Витрачено | Вартість витрачених матеріалів, грн. |
| 1 | Папір | 0,2 | 400 | 80 |
| 2 | Ручка | 8 | 2 | 16 |
| 3 | Диски | 10 | 2 | 20 |
| Всього | | | | 116 |

[грн], (4.5)

де – сума витрат на матеріали;

Ктр – коефіцієнт транспортних витрат – 1,1;

4.1.6 Витрати на силову електроенергію

Якщо ця стаття має суттєве значення для проведення випробувань, розраховується за формулою

[грн], (4.6)

де В – вартість 1кВт – години електроенергії В = 1,68грн/кВт;

П – установлена потужність комп’ютера та інших пристроїв, кВт;

Ф – фактична кількість годин роботи комп’ютера та інших пристроїв при створені програмного продукту, годин.

– коефіцієнт використання потужності < 1.

грн).

4.1.7 Інші витрати (накладні витрати – приймаються як 10 – 50% від суми основної заробітної плати розробників)

[грн], (4.7)

де – інші витрати,

4.1.8 Сума всіх попередніх витрат дає загальні витрати на розробку програмного продукту

Таблиця 4.3 – Кошторис витрат на розробку програмного продукту

|  |  |  |  |
| --- | --- | --- | --- |
| Статті витрат | Умовні позначення | Сума, грн | Структура, % |
| 1 | 2 | 3 | 4 |
| Основна заробітна плата |  |  | 61,08 |
| Додаткова заробітна плата |  |  | 6,1 |
| Нарахування на заробітну плату | Нєсв |  | 14,77 |
| Амортизація персонального комп’ютера | А |  | 2,77 |
| Витрати на матеріали |  |  | 1,29 |
| Силова електроенергія |  |  | 1,16 |
| Інші витрати |  |  | 12,83 |
| Разом | В | 9877,32 | 100 |

Наглядно структура капітальних витрат на створення програмного продукту наведена на рисунку 4.1.

Рисунок 4.1 – Структура капітальних витрат

## 4.2. Розрахунок експлуатаційних витрат при використані програмного продукту

4.2.1 Розрахуємо заробітну плату обслуговуючого персоналу

[грн/рік], (4.8)

де 12 – число місяців;

М – місячний посадовий оклад інженерно – технічного працівника, грн;

– частка часу, який витрачає працівник на обслуговування виробу з застосуванням даної програми, в загальному часі своєї роботи - 10-18%;

(грн/рік).

4.2.2 Розрахуємо додаткову заробітну плату -

Додаткова заробітна плата розраховується як 10-12% основної заробітної плати обслуговуючого персоналу

4.2.3 Розраховуємо нарахування на заробітну плату -

Нарахування на заробітну плату обслуговуючого персоналу розраховується як 22% від суми основної та додаткової заробітної плати обслуговуючого персоналу :

[грн], (4.10)

(грн).

4.2.4 Витрати на електроенергію (при живленні із електромережі) розраховуються за формулою:

(4.11)

де В – вартість 1 кВт – години електроенергії, В=1,68 грн/кВт;

П – потужність комп’ютера разом з принтером та іншими приладами, кВт;

Ф – фактична кількість годин роботи комп’ютера разом з іншими приладами за рік, годин, Ф=1800;

– коефіцієнт використання потужності < 1;

- доля часу, який витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту, в загальному часі своєї роботи ;

4.2.5 Розрахуємо амортизаційні відрахування для програмного продукту

[грн], (4.12)

де Ц – вартість робіт зі створення нового програмного продукту;

– річна норма амортизаційних відрахувань – 20%;

4.2.6 Витрати на поточний ремонт комп’ютерної техніки можна розрахувати за формулою:

(4.13)

де Ц – балансова вартість персонального комп’ютера, грн;

Зобс – заробітна плата обслуговуючого персоналу;

Зд – додаткова заробітна плата;

- доля часу, який витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту, в загальному часі своєї роботи .

4.2.7 Розрахуємо інші витрати

Інші витрати розраховуються як 5-10% від загальної суми усіх попередніх витрат:

[грн], (4.14)

(грн).

4.2.8 Розраховуємо експлуатаційні витрати

Сума витрат попередніми статтями дає величину витрат при використанні нового програмного продукту

[грн], (4.15)

Таблиця 4.5 – Кошторис експлуатаційних витрат

|  |  |  |  |
| --- | --- | --- | --- |
| Статті витрат | Умовні позначення | Сума грн. | Структура, % |
| Заробітна плата обслуговуючого персоналу |  |  | 54,84 |
| Додаткова заробітна плата |  |  | 5,48 |
| Нарахування на заробітну плату | Нєсв |  | 13 |
| Амортизаційні відрахування для програмного продукту | А |  | 3,55 |

Продовження таблиці 4.5

|  |  |  |  |
| --- | --- | --- | --- |
| Витрати на поточний ремонт комп’ютерної техніки | Р |  | 14,2 |
| Витрати на електроенергію |  |  | 1,36 |
| Інші витрати |  |  | 7,57 |
| Разом |  | 14016,06 | 100 |

Наглядно структура експлуатаційних витрат наведена на рисунку 4.2.

Рисунок 4.2 – Структура експлуатаційних витрат

## 4.3 Розрахунок обсягу робіт при використанні програмного продукту

4.3.1 Розраховуємо умовний обсяг робіт Q за формулами

де – умовний обсяг робіт при застосування існуючого програмного продукту, умовних одиниць;

– умовний обсяг робіт при застосуванні нового програмного продукту, умовних одиниць;

β – доля часу, який витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту в загальному часі своєї роботи – 15%;

– час виконання конкретної функції або роботи при застосуванні відповідно існуючого та нового програмного продукту, хв.

## 4.4 Розрахунок економічної ефективності

4.4.1 Розраховуємо річний економічний ефект від впровадження нового програмного продукту

[грн/рік] (4.18)

де – експлуатаційні витрати при використанні існуючого програмного продукту, грн/рік;

– експлуатаційні витрати при використанні нового програмного продукту, грн/рік;

*–* умовний обсяг,що виконується за рік при використанні існуючого програмного продукту;

– умовний обсяг, що виконується за рік при використанні нового програмного продукту.

= =9063 (грн/рік).

4.4.2 Розраховуємо термін окупності витрат, які були використані на розробку нового програмного продукту

[років], (4.19)

де В – загальна сума капіталовкладень;

– річний економічний ефект програмного продукту, грн.

= 1,1 (років).

Отже, в результаті отримано термін окупності витрат, які були використані на розробку нового програмного продукту, який складає 1,1 роки.

# 5 Охорона праці

## 5.1 Характеристика робочого місця

Під час роботи на комп'ютерах операторам приходиться постійно знаходиться перед ПЭВМ. Робота на ПЭВМ характеризується впливом на організм людини наступних шкідливих факторів:

* підвищене значення напруги електричного струму;
* рентгенівське випромінювання, що виникає при гальмуванні електронного променя на внутрішній поверхні кінескопа монітора;
* синій люмінофор екрана монітора має часткове випромінювання в ультрафіолетовій області спектра;
* електромагнітні коливання низької частоти, зв'язані з роботою схем розгорнення електронно-променевої трубки монітора;
* електромагнітні поля (ефект відображення);
* підвищений рівень шуму;
* електромагнітне випромінювання (радіочастоти);
* електростатичне поле.

Для забезпечення, найкращих умов для ефективної і безпечної роботи потрібно створити такі умови праці, що будуть комфортними і максимально зменшують вплив даних шкідливих факторів. Необхідно, щоб перераховані шкідливі фактори узгоджувалися з установленими правилами і нормами.

## 5.2 Безпека праці

5.2.1 Захист від шуму

Шум ‑ це сукупність звуків, що несприятливо впливають на організм людини, що заважають його роботі і відпочинкові.

Дослідження показують, що в умовах шуму насамперед страждають слухові функції. Але дія шуму не обмежується впливом тільки на слух. Він викликає помітні зрушення ряду фізіологічних і психічних функцій. Шум шкідливо впливає на нервову систему і знижує швидкість і точність сенсомоторних процесів, зростає число помилок при рішенні інтелектуальних задач. Шум впливає на увагу людини і викликає негативні емоції.

Основним джерелом шуму в приміщеннях, де знаходяться ЕОМ, є устаткування для кондиціонування повітря, друкована і копіювальна техніка, а в самих ЕОМ вентилятори систем охолодження. Відповідно до рівень шуму на робочому місці не повинний перевищувати 59дб. Нормовані рівні шуму забезпечуються шляхом використання звуковбирних матеріалів для облицювання приміщень.

Основними мірами боротьби із шумом є:

* усунення або ослаблення причин шуму в самому його джерелі в процесі проектування й експлуатації устаткування;
* ізоляція джерел шуму від навколишнього середовища засобами звукоізоляції і звукопоглинання;
* раціональне планування виробничого устаткування;
* ізоляція джерел шуму від навколишнього середовища;

5.2.2 Захист від електромагнітних випромінювань

Згідно нормованими параметрами в діапазоні частот 60 кГц ‑ 300Мгц є напруженості Е і Н електромагнітного поля. При частоті від 60 Кгц до 3Мгц напруженість електричного поля може складати до 50В/м, а напруженість магнітного поля до 5А/м.

При роботі монітора генерується електромагнітне випромінювання дуже низької частоти. Поблизу монітора напруженість електромагнітного поля складає від 4 до 7 міллігаусс. Напруженість магнітного поля вище 4 міллігаусс є шкідливою для людини. Таке випромінювання є причиною аномалій при вагітності і викликає зміну на клітинному рівні. Негативно позначається на людському організмі збільшення кількості позитивно заряджених іонів у повітрі, поблизу працюючого монітора. Медичні дослідження, проведені в США, показують, що довгострокове перебування в деіонізованій атмосфері впливають на метаболізм і приводить до зміни біохімічної реакції в крові на клітинному рівні, що нерідко приводить до стресів. При роботі з монітором на ньому накопичується заряд статичної електрики, що приводить до виникнення електростатичного розряду.

Захист від електромагнітного впливу забезпечується екранами з електропропусною поверхнею. У конструкції цих екранів, як правило, передбачена можливість заземлення. Ці екрани забезпечують надійний захист від електромагнітних випромінювань. Існують наступні типи екранів:

* екрани типу "поляроід";
* екрани плівкового типу;
* скляні екрани;
* сітчасті екрани.

Серед представників усіх перерахованих типів є екрани з провідною поверхнею і можливістю заземлення. Ці екрани захищають власника від електростатичного й електромагнітного випромінювання монітора.

5.2.3 Захист від іонізуючих випромінювань

Іонізуюче випромінювання – випромінювання, взаємодія якого з речовиною приводить до утворення в цій речовині іонів різного типу.

Іонізуюче випромінювання складається з заряджених і незаряджених часток, до яких відносяться також фотони.

Енергію часток іонізуючого випромінювання вимірюють в несистемних одиницях електрон ‑ вольтах.

Джерело іонізуючого випромінювання ‑ пристрій, що випускає іонізуюче випромінювання .

З початком застосування дисплейної, техніки, у якій використовується електронно-променеві трубки, виникла проблема захисту оператора від шкідливого рентгенівського випромінювання, джерелом якого вона є. При розробці сучасних дисплеїв враховуються безпечні дози випромінювання і робиться всі, щоб убезпечити людини від шкідливого впливу радіації .

5.2.4 Захист від впливу електростатичного поля

Причиною виникнення електростатичного заряду є те, що електронно-променева трубка в моніторі заряджена негативно, що приводить до нагромадження позитивних аероіонів. При надлишку позитивних аероіонів. їхній позитивних заряд починає відштовхувати мікрочастинки, що завжди є присутнім у повітрі. Вони розганяються і бомбардують особу й очі людини, що знаходиться перед монітором. Це робить дуже негативну дію (роздратування нервової системи і шкіри) .

Напруженість електростатичного поля не повинна перевищувати значень, припустимих.

Зміст легень аероіонів обох знаків у зоні подиху працюючої людини практично може коливатися в межах від 1,5-10 до 5-10 у 1 див повітря.

Для захисту від впливу електростатичного заряду переважніше всього застосовуючи заземлений захисний фільтр.

Так само як захист від статичної електрики в приміщеннях з ЕОМ можна використовувати нейтралізатори й увлажники, а статі повинні мати антистатичне покриття.

Для підтримки нормованих значень концентрації позитивних і негативних, іонів у приміщеннях з ЕОМ рекомендується встановлювати кондиціонери, пристрої іонізації повітря або проводити природне провітрювання тривалістю не менш 10 хвилин після кожних 2 годин роботи.

З метою запобігання шкідливого впливу на організм працюючих людей порошин з аероіонами необхідно щодня проводити вологе збирання приміщень і не рідше 1 рази в зміну видаляти пил з екранів при виключеному моніторі.

## 5.3 Умови праці

5.3.1 Виробничий мікроклімат

Найбільш значним фізичним фактором є виробничий мікроклімат, що характеризується рівнем температури і вологості повітря, а також інтенсивністю рівня радіації.

Використовувані ЕОМ не вимагають створення особливих мікрокліматичних умов для роботи і нормально функціонують у межах припустимих для людини значень температури і вологості.

У виді того, що ЕОМ є джерелами тепловиділення, існує можливість підвищення температури і зниження вологості повітря на робочих місцях, що сприяють роздратуванню шкіри. Мікрокліматичні умови в приміщенні з ЕОМ повинні задовольняти вимогам у відповідності [12]:

* температура навколишнього середовища в холодний період року 20 – 22 C, у теплий період 22 – 25 С;
* відносну вологість повітря 30 - 60%;
* вміст пилу ‑ макс. 0.0001 кг/м при розмірі часток макс. 3 мкм.

Однією з умов здорової і високопродуктивної праці є забезпечення чистоти повітря. Атмосферне повітря у своєму складі містить у процентному відношенні:

азот 78,8%

кисень 20,25%

аргон, неон і інші інертні гази 0,93%

вуглекислий газ 0,03%

Повітря такого складу найбільш сприятливий для подиху людини.

Розглянуте в даній дипломній роботі мережне устаткування і робочі станції в процесі роботи не виробляють ніяких шкідливих речовин. У такий спосіб повітряне середовище в приміщенні, де вони використовується, шкідливих впливів на організм людини не робить і задовольняє вимогам I категорії робіт, згідно.

Оптимальні норми температури, відносній вологості і швидкості руху повітря в робочій зоні виробничих приміщень нормуються і приведені, у таблиці 5.1.

Таблиця 5.1- Норми температури та вологості

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сезон року | Категорія робіт | Температура,  С | Відносна вологість, % | Швидкість руху повітря, м/с |
| Холодний період | I | 22 - 24 | 40 – 60 | 0,1 |
| Теплий період | I | 23 - 25 | 40 – 60 | 0,1 |

5.3.2 Виробниче освітлення

Освітлення служить одним з найважливіших факторів, що впливають на продуктивність праці. Раціонально улаштоване освітлення на робочих місцях операторів є істотним показником високого рівня культури праці, невід'ємною частиною наукової організації праці й естетики виробництва.

Вимоги до раціональної освітленості виробничих приміщень зводяться до наступного:

* правильний вибір джерел світла і систем освітлення;
* створення необхідного рівня освітленості робочих поверхонь;
* обмеження сліпучої дії світла, усунення відблисків;
* забезпечення рівномірного освітлення.

Прийнятний рівень освітлення в приміщенні можна знайти, якщо послідовно вирішити дві задачі:

а) визначити необхідний для оператора рівень освітлення робочого місця зовнішніми джерелами світла.

б) якщо необхідний рівень освітленості виявиться неприйнятним для інших, операторів, що працюють у даному, приміщенні, необхідно знайти спосіб збереження необхідного контрасту зображення іншими засобами. Наприклад, можна розподілити світловий потік з урахуванням розташування робочих місць і засобів відображення, інформації.

При проектуванні й організації робочого місця оператора ЕОМ варто ужити заходів по запобіганню прямих і відбитих відблисків. Прямі відблиски з'являється в результаті наявності джерел світла безпосередньо в поле зору оператора, відбиті відблиски ‑ у результаті наявності усередині ока поверхонь, що відбивають. Прямі відблиски можна зменшити кожним з наступних способів: застосовувати відбите освітлення; користуватися декількома джерелами освітлення меншої потужності замість одного сильного; використовувати засобу екранування прямого світла від очей оператора.

Відбиті відблиски можна зменшити такими способами:

* використовувати розсіяне світло;
* застосовувати матові поверхні;
* розташовувати джерела прямого світла так, щоб кут спостереження оператором робочої площі не збігався з кутом падіння на неї променів світла від ліхтаря.

Важливою задачею є вибір виду освітлення (природне або. штучне) і відповідно до цього ‑ вибір типу виробничого приміщення (з вікнами або без вікон).

Найбільше сприятливо для людини природне освітлення. При природному висвітленні продуктивність праці робітників вище, ніж при штучному. Для достатнього природного освітлення площа вікон повинна складати не менш 1/3 від загальної площі зовнішніх стін.

Однак, варто враховувати, що застосування природного світла має багато недоліків: надходження світла, як правило, тільки з одного боку, нерівномірність освітленості в просторі і т.д.

Для усунення цих недоліків необхідно застосовувати додаткові пристосування. Застосування подвійного світла (сполучення природного і штучного освітлення) фізіологічно мало ефективно і негативно діє на зір, сприяє передчасному стомленню.

Застосування штучного освітлення допомагає уникати багатьох з розглянутих недоліків і створювати оптимальний світловий режим. Однак, застосування приміщень без вікон створює в ряді випадків у людей почуття скрутності і непевності. Особливо сильно це виявляється в приміщеннях малого обсягу. У великих приміщеннях даний недолік практично відсутній, тому тут кращий застосування штучного освітлення можна застосовувати лампи накалювання і люмінесцентні лампи.

Згідно величина освітленості люмінесцентними лампами повинна бути в горизонтальній площині не нижче 300 лк ‑ для системи загального освітлення. З урахуванням зорової роботи високої точності величина освітленості може бути збільшена до 1000 лк.

Крім освітленості великий вплив на діяльність оператора робить колір фарбування приміщення і спектральних характеристик використовуваного світла. Рекомендується, щоб потік відбивав 80-90%, стіни ‑ 50-60%, а стать ‑ 15-30% падаючого на них світла. Приміщення, де розташоване ПЭВМЛ повинно бути світлим, чистим. Стелі і стіни рекомендується офарблювати у світлі тони.

У приміщеннях, де встановлене комп'ютерне устаткування, створені умови, що задовольняють даним вимогам.

5.3.3 Розрахунок освітлення приміщення

Розрахунок освітлення проводиться для наступних його характеристик:

* стеля з білою поверхнею; / таблиця 5.1/
* стіни з зі світлою поверхнею; / таблиця 5.1/
* підлога з темною поверхнею; / таблиця 5.1/
* ширина приміщення 5 м;
* довжина приміщення 10 м;
* висота приміщення 2,5 м;
* встановлюються світильники серії ЛВО 4x18, (4 лампи в світильнику);
* світловий потік лампи Фл = 1150 лм;
* норми освітленості Е = 300 лк на рівні 0,8 м від підлоги.

Для розрахунку освітлення необхідно розрахувати площу приміщення за формулою:

де а − ширина приміщення,

в – довжина приміщення,

(м2).

Далі визначаємо індекс приміщення згідно формули:

де h1 − висота приміщення;

h2 − висота робочої поверхні (0,8 м),

Визначаємо потрібну кількість світильників за формулою:

де Е − необхідна освітленість горизонтальної площини (300 лк),

S − площа приміщення м2,

Кз − коефіцієнт запасу (Кз = 1,4),

U − коефіцієнт використання освітлювальної установки,

Фл − світловий потік однієї лампи (1150 лм).

n − число ламп в одному світильнику.

Визначаємо коефіцієнт використання U, виходячи із значень коефіцієнтів віддзеркалення і індексу приміщення.

Значення коефіцієнтів віддзеркалення приведені в таблиці 5.1.

Таблиця 5.1- Коефіцієнти віддзеркалення

|  |  |
| --- | --- |
| Поверхня | Коефіцієнт віддзеркалення |
| Площина з матеріалів з високою відбиваністю | 80 |
| Площина з білою поверхнею | 70 |
| Площина зі світлою поверхнею | 50 |
| Площина з сірою поверхнею | 30 |
| Площина темно-сірою поверхнею | 20 |
| Площина з темною поверхнею | 10 |

Значення коефіцієнта використання вибирається згідно таблиці 5.2.

Таблиця 5.2 – Значення індексу приміщення

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Коефіцієнти віддзеркалення  поверхонь  приміщення | Стеля | 50 | 50 | 70 | 70 | 50 | 50 | 30 | 0 |
| Стіни | 50 | 30 | 50 | 30 | 50 | 30 | 30 | 0 |
| Підлога | 30 | 10 | 20 | 20 | 10 | 10 | 10 | 0 |
|  | Коефіцієнт використання | | | | | | | | |
| Індекс  приміщення | 0.6 | 36 | 28 | 35 | 29 | 33 | 28 | 28 | 23 |
| 0.8 | 42 | 33 | 40 | 33 | 38 | 32 | 31 | 26 |
| 1.0 | 47 | 37 | 44 | 37 | 41 | 36 | 35 | 29 |
| 1.25 | 52 | 41 | 49 | 42 | 45 | 40 | 39 | 33 |
| 1.5 | 57 | 45 | 53 | 46 | 49 | 44 | 43 | 37 |
| 2.0 | 63 | 51 | 58 | 52 | 54 | 49 | 48 | 42 |
| 3.0 | 76 | 65 | 70 | 65 | 64 | 61 | 60 | 55 |
| 5.0 | 89 | 75 | 81 | 78 | 74 | 72 | 71 | 68 |

Враховуючи коефіцієнти віддзеркалення поверхонь приміщення та його індекс вибираємо, що коефіцієнт використання U дорівнює 70 /таблиця 5.2/

Визначаємо необхідну кількість світильників:

5.3.4 Технічні міри захисту від поразки електричним струмом

Усі технічні міри можна умовно розділити на двох груп.

Технічні захисні міри першої групи забезпечують захист від поразки електричним струмом обслуговуючого персоналу у випадку дотику до струмоведучих частин, до них відносяться:

* контроль стану ізоляції електротехнічних пристроїв і ділянок живильної мережі;
* блокування і захисні огородження;
* оптимальне розташування устаткування, що забезпечує розриви між струмоведучими частинами;
* сигналізація безпеки (світлова, звукова), маркірування і попереджувальні плакати;
* захист від переходу високої напруги на сторону низької напруги;
* застосування низьких напруг 42 і 12 В;
* застосування індивідуальних захисних ізолюючих засобів.

Технічні міри другої групи забезпечують захист від поразки електричним струмом при дотику до корпуса електроустановки у випадку пробою ізоляції струмоведучих частин, до них відносяться:

* захисне заземлення;
* захисне занулення;
* захисне відключення;
* подвійна ізоляція;
* застосування розділових трансформаторів.

Відомо, що надійність і довговічність роботи електротехнічного устаткування багато в чому залежать від стану електричної ізоляції струмоведучих частин. Ушкодження ізоляції досить часто є головною причиною багатьох електричних травм, аварій і пожеж. Фізичний зміст ізоляції, як захисної міри, полягає в обмеженні струму, що протікає по тілу людини, до безпечної величини.

Надійна ізоляція залежить від багатьох факторів і забезпечується застосуванням, визначеного її типу (робочих, посилених і подвійна), що відповідають ізоляційних матеріалів, раціональною конструкцією електроустаткування, нормальними умовами виробничого середовища і, нарешті, правильною організацією профілактики в процесі експлуатації.

Як правило, електротехнічне устаткування має робочу ізоляцію, що повинна витримувати гранично можливі в умовах експлуатації механічні, електричні і теплові навантаження.

Захисне заземлення - це навмисне з'єднання з землею не струмоведучих металевих частин електричного устаткування, апаратури, блискавковідводів і розрядників. Призначення захисного заземлення – знизити до безпечної величини напруга на корпусі щодо землі, що виникає на не струмоведучих частинах електроустановок у випадку замикання на корпус (пробою на корпус) при ушкодженні ізоляції провідників, що несуть робочий струм живлення апаратури.

Таким чином, при наявності заземлення з опором розтіканню струму малої величини, різко зменшується струм, що протікає через тіло людини, що торкнувся ушкодженого заземленого корпуса. До аналогічного висновку можна прийти, розглянувши значення опору заземлення корпусів електроустаткування при харчуванні від мережі перемінного струму.

## 5.4 Ергономіка робочого місця

5.4.1 Робота з дисплеєм

Дисплей повинний відповідати наступним технічним вимогам:

* яскравість світіння не менш 100 кд/м2;
* мінімальний розмір світлової крапки не більш 0,1 мм для кольорового дисплея;
* контрастність зображення знака не менш 0,8;
* частота кадрового розгорнення не менш 7 кГц;
* кількість крапок не менш 640;
* антибликове покриття екрана;
* розмір екрана не менш 31 див по діагоналі;
* висота символів на екрані не менш 3,8 мм; Відстань від очей оператора до екрана повинне бути порядку 40 - 80 див;
* яскравість світіння не менш 100 кд/м2;
* мінімальний розмір світлової крапки не більш 0,1 мм для кольорового дисплея;
* контрастність зображення знака не менш 0,8;
* частота кадрового розгорнення не менш 7 кГц;
* кількість крапок не менш 640;
* антибликове покриття екрана;
* розмір екрана не менш 31 див по діагоналі; висота символів на екрані не менш 3,8 мм;

Відстань від очей оператора до екрана повинне бути порядку 40 ‑ 80 див.

Дисплей повинний бути обладнаний поворотною площадкою, що дозволяє переміщати його в горизонтальній і вертикальній площинах у межах 130-220 мм і змінювати кут нахилу екрана на 10-15 градусів.

Оптимальне співвідношення яскравості між екраном, його ближнім і далеким оточенням (периферійна зорова зона) складає 5:2:1. Співвідношення тло ‑ буква - 5:1. Значно знижений контраст між знаком і тлом екрана сприяє стомленню зорового аналізатора. Суб'єктивна оцінка яскравості сприйманого сигналу залежить від яскравості тла, у зв'язку з чим розрізняють прямий контраст - розраховується для темного об'єкта на світлому тлі зворотний контраст - для світлого об'єкта на темному тлі.

Критична частота мелькання (КЧМ) ‑ це частота появи світлового сигналу, при якій він як подразник сприймається безупинно.

Ця частота залежить від яскравості, розмірів, конфігурації зображення. При нормальних умовах ця КЧМ приблизно дорівнює 15 ‑ 25 Гц, але при зоровому стомленні ця величина знижується.

5.4.2 Робоче місце

Робоче місце оператора для виконання робіт сидячи повинно відповідати вимогам.

Просторова організація робочого місця повинна забезпечувати оптимальну робочу позу:

* голова нахилена вперед на 10 ‑ 20 градусів;
* спина має упор, співвідношення між плечем і передпліччям, а також між стегном і гомілкою ‑ прямий кут.

Основні параметри робочого місця повинні бути регульованими. Цим забезпечується можливість створення сприятливих умов праці окремій людині з обліком геоантропометричних характеристик.

Основні параметри робочого місця, оснащеного персональним комп'ютером:

* висота сидіння 42 ‑ 45 див;
* висота клавіатури від статі 70 ‑ 85 див;
* кут нахилу клавіатури від горизонталі 7 ‑ 15 градусів;
* далекість клавіатури від краю столу 10 ‑ 26 см;
* відстань від центра екрана до статі 90 ‑ 115 см;
* кут нахилу екрана від вертикалі 0 ‑ 30 градусів (оптимальний 15);
* далекість екрана від краю столу 50 ‑ 75 см;
* висота робочої поверхні для записів 74 ‑ 78 см;
* на робочому місці необхідно передбачити підставку для ніг, що рекомендується для усіх видів робіт, зв'язаних із тривалим збереженням положення сидячи.

При проектуванні просторового розміщення робочих місць у загальному залі, необхідно враховувати потребу людини в збереженні персонального простору й оптимальної дистанції, що залежить від характеру взаємин і від зовнішньої ситуації.

Рекомендується розміщати робочі місця на відстані 215 ‑ 400 см. (далека фаза соціальної дистанції).

Присутність іншої особи на цій дистанції не є заважаючим чинником, як на більш близьких дистанціях, тобто людина може продовжувати працювати, незважаючи на присутність іншої людини. Спілкування при цьому не утруднено. При проектуванні здвоєних робочих місць опускається розміщення працівників на відстані 120 ‑ 215 см, що відповідає близькій фазі соціальної дистанції.

Системні блоки встановлюються на робочому місці з урахуванням легкої досяжності до рознімань і органів керування на тильній стороні. Принтер, як правило, розміщають праворуч від користувача. Текст, що друкується, повинний бути видний операторові при перебуванні його в основній робочій позі. Поблизу від принтера в спеціальних відсіках зберігається чистий папір і інші необхідні приналежності.

Сполучні кабелі прокладаються в спеціальних каналах. Пристрій каналів повинен бути таким, щоб сполучні рознімання не перешкоджали витягові кабелів.

Для маніпулятора типу «миша» праворуч від користувача на стільниці повинна бути передбачена вільна площадка, що за формою і розміром повинна бути ідентична поверхні екрана.

## 5.5 Пожежна безпека

Для більшості приміщень, обладнаних ВТ, установлена категорія пожежної небезпеки В – (пальні і непальні рідини, тверді пальні і непальні речовини і матеріали (у тому числі пилу і волокна), речовини і матеріали, здатні при взаємодії з водою, киснем повітря або один з одним тільки горіти за умови, що приміщення, у яких вони маються в наявності або утворяться, не відносяться до категорій А або Б). З огляду на високу вартість електронного устаткування, а також категорію пожежної небезпеки приміщень, у яких воно розміщається, будинку для приміщення, обладнаних ВТ, повинні бути I ступеня вогнестійкості по СНиП.

У виробничому приміщенні необхідно дотримувати наступні правила безпеки:

* проходи, виходи з приміщення, доступи до засобів пожежегасіння повинні бути увесь час вільні;
* устаткування, що знаходиться в експлуатації, повинне бути справно і перевірятися щораз перед початком роботи;
* по закінченні робіт оглянути приміщення, знеструмити електромережу, закрити приміщення.

Число евакуаційних виходів з будинків з кожного поверху і з приміщень повинне бути не менш двох. Ширину евакуаційного виходу (двері) установлюють не менш 0.8 м. Пристрій кручених сходи, розсувних і піднімальних дверей, що обертаються і турнікетів на шляхах евакуації неприпустимо. Також не допускається розміщати на сходових клітках які-небудь приміщення, прокладати технологічні комунікації, улаштовувати виходи підйомників і вантажних ліфтів. На евакуаційних шляхах улаштовують як природне, так і штучне аварійне освітлення.

Пожежні крани встановлюють у коридорах, на площадках сходових кліток, у входів, тобто в доступних, помітних місцях. Ручні вуглекисневі вогнегасники встановлюють у приміщеннях, обладнаних ВТ, з розрахунку один вогнегасник на 40 ‑ 50 кв. м площі, але не менш двох у приміщенні.

Для виявлення початкової стадії загоряння й оповіщення служби пожежної охорони використовують системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, поки пожежа не досягла великих розмірів.

Об'єкти ОЦ крім АПС необхідно обладнати установками стаціонарного автопожаротушіння. Доцільно застосовувати установки газового гасіння пожеж, дія яких засновано на швидкому заповненні приміщення вогнегасячою газовою речовиною, у результаті чого знижується зміст кисню в повітрі.

## 5.6 Надзвичайні ситуації

До надзвичайних ситуацій, що можуть відбутися можна віднести пожежа і коротке замикання.

Відповідно до ДСТ пожежна безпека повинна забезпечуватися системою запобігання пожежі й організаційно ‑ технічними заходами. Протипожежний захист досягається застосуванням засобів пожежогасіння, установок пожежної сигналізації, застосуванням засобів колективного й індивідуального захисту.

У випадку виникнення короткого замикання необхідно якнайшвидше відключити ЕОМ від мережі.

При виникненні пожежі необхідно:

* повідомити в пожежну охорону по телефоні 01 і операторові або секретареві;
* знеструмити електроустаткування;
* приступити до гасіння пожежі первинними засобами пожежогасіння;
* організувати евакуацію персоналу з небезпечної зони;
* організувати зустріч пожежної команди і надати їй повну інформацію про сформовану обстановку.

# Висновки

В результаті виконання дипломного проекту розроблено сервіс «BookLibrary». Розроблений сервіс задовольняє всім вимогам, поставленим на етапі постановки завдання та дозволяє автоматизувати роботу бібліотеки.

Веб-сайт створений з використанням мов HTML, CSS, JavaScript-фреймворків, СКБД MongoDB та Node.js для серверної частини. У проекті реалізовано зручний, інтуїтивно зрозумілий інтерфейс. Дизайн виконаний з використанням оптимальної колірної гами.

В роботі виконано техніко-економічне обґрунтування доцільності розробленого сайту, зроблений порівняльний аналіз можливих варіантів вирішення поставленої задачі. Розглянуто загальні питання розробки та проектування веб-додатків. Приведено план розробки сайту, архітектуру організації компонентів, лістинги програмних модулів та функцій. Детально описана методика роботи сайту. Розроблені організаційно-економічні питання, що стосуються даного проекту.

Проект може бути вдосконаленим та розширеним, що відповідає потребам предметної області діяльності.

# Список літератури

1. Крамер Эрик HTML: Наглядный курс Web-дизайна. М.: Издательский дом, 2008
2. Популярные Web-сервисы: практика использования. Айверсон Уилл. КУДИЦ – ОБРАЗ, 2009.
3. Разработка Web - приложений с помощью PHP и MySQL, 3-е издание. Люк Веллинг, Лора Томсон. Вільямс, 2008.
4. Фролов А.В., Фролов Г.В. Глобальные сети компьютеров. Практическое введение в Internet, E-mail, FTP, WWW, и HTML, программирование для Windiws Sockets. - Диалог - МИФИ, 1996. Ковалев А., Курдюмов И и др. Управление проектом по созданию интернет-сайта - М.: Альпина Паблишер, 2010. - 337 с.
5. Энди Харрис. PHP/MySQL для початківців. Кудиц-образ, 2005.
6. Internet, Сервiси, HTML i web дизайн. Глинський. Деол, 2010.
7. PHP 5.0 для начинающих, или как создать динамичный web-сайт. ‑ 2-е изд., дополненное и исправленное. Леонтьев Б. [Новый издательский дом](http://www.labirint-shop.ru/pubhouse/407/), 2006.
8. WEB-дизайн Руководство пользователя. Под ред. Леонтьева. Познавательная книга, 2008.
9. Титенко С. В. [Моделювання спеціалізованих інформаційних об’єктів в універсальних системах керування Web-контентом](http://www.setlab.net/?view=Tytenko-Eunu-2012) / С. В. Титенко // Вісник Східноукраїнського національного університету імені Володимира Даля – 2012. – №8 (179). Ч.2. — С. 235-239.
10. Питер Л.Д. HTML5 для профессионалов Мощные инструменты для разработки современных веб-приложений / О.К. Брайан, С.Т. Фрэнк. – «Вильямс», 2011. – 272с.
11. Білан С.М. Засоби машинної графіки. Навчальний посібник. – Вінниця: ВДТУ, 2004р.
12. Сериков Я.А. Основи охорони праці. – Харків:ХНАГХ, 2007. – 227с.
13. Навчальний посібник: Інженерні рішення з охорони праці при розробці дипломних проектів інженерно-будівельних спеціальностей: – Київ: Основа, 2011.
14. Методичні вказівки до дипломного проектування для студентів спеціальності 5.05010201 «Обслуговування комп’ютерних систем і мереж». Методичний посібник./Укладачі: Заліська С.С., Петрович С.Д., Тичук Р.Б., Кривобока Г.І. – Вінниця: ВК НУХТ, 2015 – 52с.
15. Етапи створення сайту [Електронний ресурс]: Режим доступу: <http://ua-web.com.ua/ua/stati/etapyi_sozdaniya_sajta.html>