РЕФЕРАТ

Кваліфікаційна робота на здобуття освітнього ступеню «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПс-43, 2023 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: 51 с., 17 рис., 4 табл., 3 додатків.

Предметна область – створення веб-сайту документообігу міської ради

Мета роботи – проектування бази даних та створення веб-сайту документообігу міської ради

Метод дослідження – опис предметної області шляхом опрацювання та аналізу уже існуючих веб-сайтів міських рад; проектування бази даних, використовуючи методи нормалізації; написання і відлагодження програми. Створення архітектури додатку та побудова UML-діаграм за допомогою програми «Rational Rose». Для написання коду та його відлагодження використовуватиметься середовище програмування Microsoft Visual Studio

Отримані результати – розроблено веб-сайт документообігу Чортківської міської ради та база даних для зберігання інформації про документи та інше.

Ключові слова: інтернет провайдер, база даних, СУБД, інформаційна система, Er-діаграма, зв'язок, представлення, нормалізація, html5, css3, веб-сайт, інформаційна система, asp.net.

ANNOTATION

Qualification work for obtaining a bachelor's degree in specialty 121 - Software engineering. Ternopil National Technical University named after Ivan Pulyuya, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SPS-43, 2023. The explanatory note to the qualification work for obtaining the bachelor's degree contains: 51 pages, 17 figures, 4 tables, 3 appendices.The object of research is the creation of a document circulation website of the City Council

The purpose of the work is to design a database and create a document management website of the City Council

Research method – description of the subject area by processing and analyzing already existing websites of city councils; database design using normalization methods; writing and debugging the program. Creation of application architecture and construction of UML diagrams using the program "Rational Rose". The Microsoft Visual Studio programming environment will be used to write the code and debug it

The results obtained are the development of the document management website of the Chortkiv City Council and a database for storing information about documents and other things.

Key words: internet provider, database, dbms, information system, er diagram, communication, representation, normalization, html5, css3, website, information system, asp.net.

ЗМІСТ

ВСТУП

В наш час, присутність дієвої системи автоматизації управління діловодством та обігом документів свідчить про успішність установи та її керівництва. Це свідчить про те, що всі працівники керівництва апарату належним чином керуються, володіють необхідними знаннями, діють єдиною командою, дисципліновані та зацікавлені у максимально ефективному виконанні покладених на них обов'язків.

Сучасні міста потребують автоматизованих систем управління інформацією через потребу обробки великого обсягу даних, що надходять з усіх куточків міста, таких як відеозаписи з камер спостереження, місцезнаходження громадського транспорту, метеорологічні дані та інші.

Одна з таких систем - це системи документообігу, які широко використовуються урядовими установами, оскільки вони дозволяють систематизувати акти для зручного майбутнього доступу до них. Крім того, такі системи надають публічний доступ для громадян. Таким чином, кожен мешканець міста матиме можливість переглянути цікавий йому документ, наприклад, під час поїздки на роботу у громадському транспорті.

Усі дані про документи будуть зберігатися в створеній базі даних, де будуть застосовані тригери, представлення та транзакції для забезпечення безпеки та недоторканості даних.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області та опис суб’єкту автоматизації

Сучасні інформаційні технології базуються на базах даних і системах їх керування, які постійно зростають у своїй ролі основного засобу зберігання, обробки та доступу до великих обсягів інформації. Це особливо важливо, оскільки обсяги інформації, збереженої в базах даних, постійно збільшуються, що вимагає збільшення продуктивності таких систем [1].

Кожне сучасне місто потребує автоматизованої системи управління інформацією, оскільки існує необхідність обробки великого обсягу даних, що надходять з різних організацій та установ міста. Ця інформація включає, наприклад, відеозаписи з камер спостереження, місцезнаходження громадського транспорту, метеорологічні дані та інші.

Системи обліку документів широко використовуються урядовими установами, оскільки вони дозволяють упорядковувати акти для зручного майбутнього доступу до них. Однією з важливих функцій таких систем є публічний доступ для громадян. Ідеальною ситуацією було б, якщо кожен мешканець міста мав можливість переглядати цікаві йому документи, наприклад, під час поїздки на роботу у громадському транспорті.

Об'єктом дослідження у дипломному проекті є документація міської ради, яка характеризується датою затвердження та типом. Всі акти поділяються на офіційні документи міської ради, протоколи, акти регуляторної політики, проекти рішень, програми та списки податків. Офіційні документи поділяються на рішення міської ради, рішення виконавчого комітету та розпорядження міського голови. Проекти рішень поділяються на "проекти рішень сесії" та "проекти рішень виконавчого комітету". Документація регуляторної політики включає нормативно-правові акти, акти громадських слухань, плани діяльності з підготовки регуляторних актів, проекти регуляторних актів та перелік діючих регуляторних актів,

прийнятих міською радою [2]. Документи типів "Протоколи", "Програми" та "Список податків" не мають поділу на підгрупи, але основною характеристикою для них є дата прийняття.

У дипломному проекті розроблений веб-сайт, який призначений для надання відкритого доступу до документів міської чи селищної ради. Функціонал сайту обмежується можливістю пошуку актів за критеріями та сортуванням за датою прийняття для знаходження найбільш актуальних документів.

1.2 Огляд подібних проектних рішень

Сьогодні немає кращого способу збереження даних, ніж їх оцифрування та автоматизація опрацювання за допомогою комп'ютерів. Чимало сучасних міст переходять на електронне цифрування історично важливих документів з метою збереження. Проте зберігання даних не має сенсу, якщо вони не доступні для використання. Тому багато міст створюють веб-сайти, на яких документи, призначені для перегляду, стають доступними широкому загалу користувачів. Такі сайти також спрощують процес отримання інформації про акти, які можуть бути потрібні громадянам для вирішення їхніх справ.

Аналізуючи кілька подібних сайтів, було помічено, що вони призначені переважно для спеціалізованої аудиторії, зокрема для інвесторів. Розглянемо кілька прикладів таких рішень [3].

Сайт «https://doc.drohobych-rada.gov.ua» представляє собою сайт документообігу Дрогобицької міської ради (див. рис. 1.1).

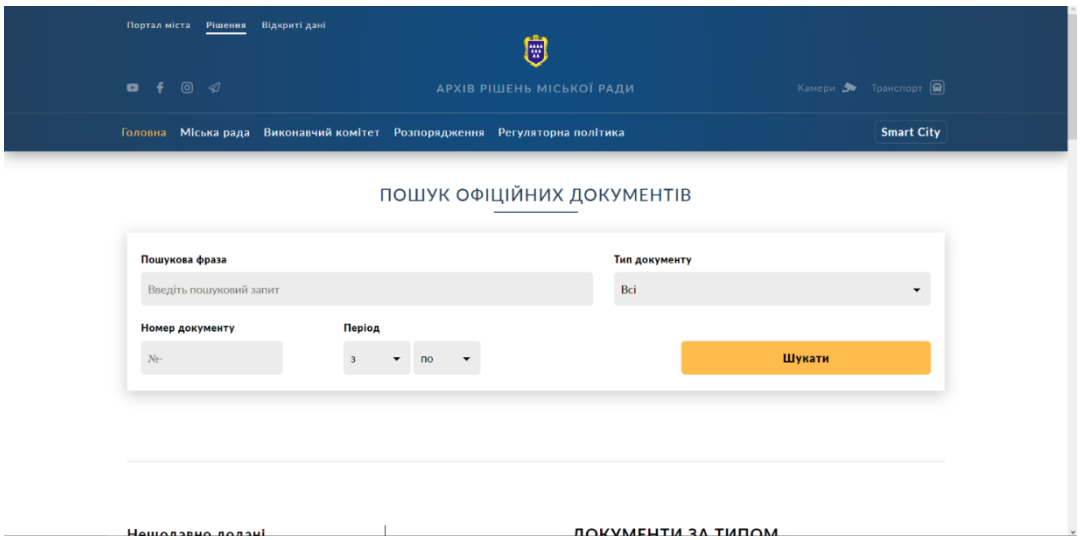


Рисунок 1.1 – Головна сторінка Дрогобицької міської ради

Переваги цього сайту включають:

– Можливість пошуку документів.

– Інтуїтивно зрозуміла навігація по сайту.

– Наявність діаграми, що відображає кількість та типи доданих документів за роками.

– Присутність адаптивного дизайну.

– Швидке завантаження сторінок.

– Можливість підписки на онлайн розсилку.

Також варто зазначити кілька недоліків:

– Відсутність пошукових меню на сторінці переліку документів певного типу.

– Відсутність явного відокремлення документів один від одного на сторінці.

Сторінка «https://www.chortkivmr.gov.ua/dokymentu/» містить інформацію про документацію Чортківської міської ради. Розглянемо одну з вкладок на цьому сайті (див. рис. 1.2).

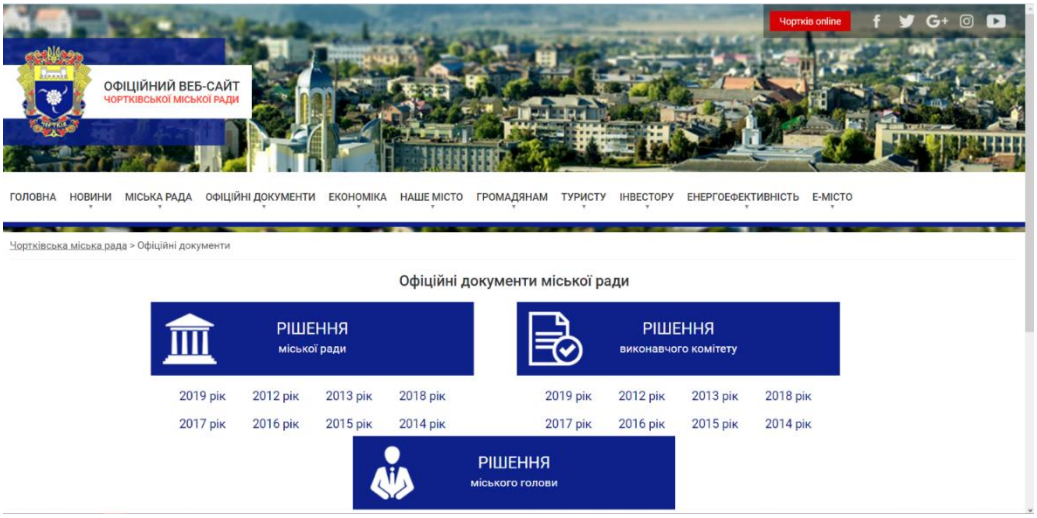


Рисунок 1.2 – Сторінка документації Чортківської міської ради

Переваги цього сайту включають зручні умови для пошуку документів. До недоліків можна віднести:

– Відсутність адаптивного дизайну.

– Відсутність явного відокремлення документів один від одного на сторінці.

– Повільне завантаження сторінок.

Сторінка «https://www.vmr.gov.ua/Docs/default.aspx» представляє собою сайт документації Вінницької міської ради (див. рис. 1.3).

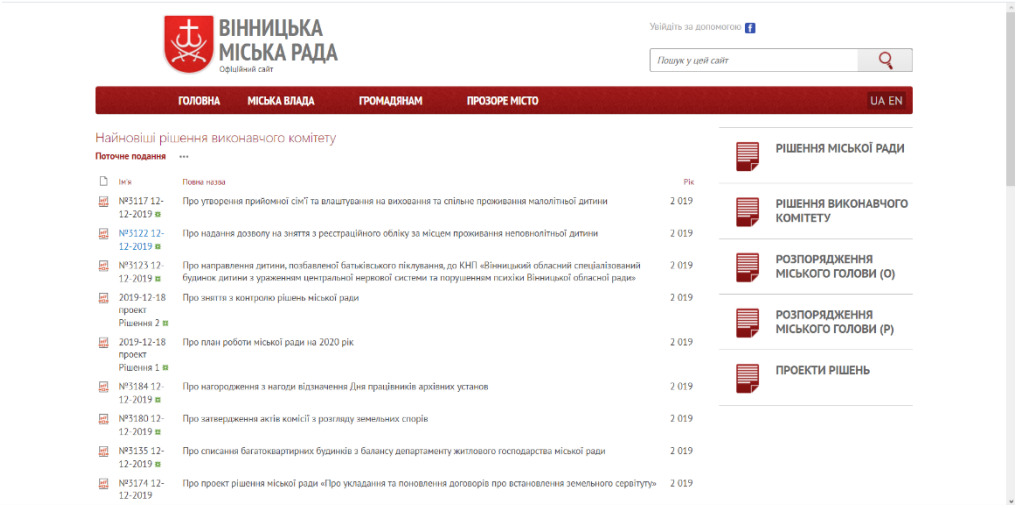


Рисунок 1.3 – Сайт документації Вінницької міської ради

Перевагою цього сайту є можливість переключення мови. До недоліків можна віднести:

– Незрозумілу навігацію по сайту.
– Незручні умови для пошуку за критеріями.
– Тривале завантаження сторінок.
– Хаотичне розміщення компонентів сайту.

Оскільки головною темою цих сайтів є документація, різні ефекти на сторінках можуть бути непотрібними. Враховуючи всі перелічені переваги та недоліки, можна утворити загальне уявлення про те, який сайт відповідатиме критеріям зручного веб-додатку [4].

2 ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдання

Більшість автоматизованих систем обліку інформації не доступні для широкого загалу. Проте іноді це все-таки має сенс. Перед створенням веб-додатку, виникла потреба надати жителям міста доступ до значної кількості документів різних типів, які містять інформацію про історичний розвиток міста. Крім того, у міській раді існує багато актів, які можуть зацікавити інвесторів та підприємців [5]. Ця інформація також має велике значення для звичайних мешканців міста, оскільки вони безпосередньо пов'язані з життям міста, і будь-який документ може мати відношення до них.

Ось чому важливо, щоб сайт мав зручне меню пошуку документів за такими критеріями:

– Ключові слова.

– Дата.

2.2 Проєктування бази даних

Для забезпечення можливості адміністратору сайту вручну керувати контентом, потрібно створити базу даних, визначити ключові та неключові поля, встановити зв'язки між ними та створити нормалізований набір відношень [6]. Оскільки проект має стосуватись документації міської ради, можна визначити інформаційні об'єкти, що відносяться до цієї сфери.

Опис призначення сутностей наведено в таблиці 2.1.

Таблиця 2.1 – Опис сутностей бази даних «CityGov»

Ім’я сутності	Опис	Псевдонім	Особливості використання
Офіційні документи	Збереження даних про офіційні документи	OfDoksDs	Містить інформацію про документи, видані міським головою
Програми	Збереження даних про програми міської ради	ProgramsDs	Містить план дій у певній галузі
Проекти рішень	Збереження даних про проети рішень	ProjectRDs	Проекти, які ще не затвердженні і знаходяться на розгляді
Протоколи	Збереження даних про протоколи	ProtokolsDs	Містить інформацію про протоколи
Регуляторна політика	Збереження даних про регуляторну політику	RegPolDs	Містить інформацію про документи регуляторної політики міської ради
Типи офіційних документів	Містить типи офіційних документів	DoksTypes	Містить перелік типів офіційних документів
Типи регуляторної політики	Містить типи документів регуляторної політики	RegPolTypes	Містить перелік документів регуляторної політики
Типи проектів рішень	Містить типи проектів	ProjectRTypes	Містить типи проектних рішень
Останні додані документи	Мітить останні додані документи	LastOneDs	Містить перелік декількох останніх доданих документів

Структура сутностей «OfDoksDs», «ProjectRDs» та «RegPolDs» має ідентичну будову [7]. У свою чергу сутності «PodDs» », «ProgramsDs» та «ProtokolsDs», мають свою визначену структуру (див. табл. 1.2).

Таблиця 2.2 – Структура сутностей

Назва сутності	Атрибут	Опис	Домен	Обмеження	Псевдонім	Значення NULL
1	2	3	4	5	6	7
OfDoksDs	Код документу	Унікальний ідентифікатор документу	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Тип	Підтип документу	Стрічка	Max	Type	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	Max	Date	Так
ProjectRDs	Код документу	Унікальний ідентифікатор документу	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Тип	Підтип документу	Стрічка	Max	Type	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	Max	Date	Так
RegPolDs	Код документу	Унікальний ідентифікатор документу	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так

Продовження таблиці 2.2

1	2	3	4	5	6	7
	Тип	Підтип документу	Стрічка	Max	Type	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	Max	Date	Так
LastOneDs	Код документу	Унікальний ідентифікатор документу	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Тип	Підтип документу	Стрічка	Max	Type	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	Max	Date	Так
LastOneDs	Код документу	Унікальний ідентифікатор документу	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Тип	Підтип документу	Стрічка	Max	Type	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	Max	Date	Так
PodDs	Код документу	Унікальний ідентифікатор	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так

Продовження таблиці 2.2

1	2	3	4	5	6	7
	Дата	Дата затвердження документу	Стрічка	max	Date	Так
ProgramsDs	Код документу	Унікальний ідентифікатор	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	max	Date	Так
ProtokolsDs	Код документу	Унікальний ідентифікатор	Ціле число	Первинний ключ	Id	Ні
	Назва	Назва документа	Стрічка	Max	Name	Так
	Ссилка	Посилання н скачування	Стрічка	Max	Link	Так
	Дата	Дата затвердження документу	Стрічка	max	Date	Так

На основі первинних ключів, на етапі проектування бази даних, будуються зв'язки між сутностями, які визначені в таблиці 1.3. Для опису концептуальної моделі предметної області була використана ER-діаграма [8], яка наведена у додатку А.

Таблиця 2.3 – Опис зв’язків між сутностями бази даних «CityGov»

Ім’я сутності	Назва зв’язку	Ім’я сутності	Кардинальність
OfDoksDb	має	OfDoksTypes	1 : 1
ProjectRDb	має	ProjectRTypes	1 : 1
RegPolDb	має	RegPolTypes	1 : 1

Нормалізація бази даних до третьої нормальної форми включає кроки для забезпечення вимог реляційної моделі даних [9].

Перша нормальна форма (1НФ) вимагає, щоб кожне відношення містило первинний ключ, який ідентифікує записи. Відповідно до цього вимоги, у всіх відношеннях бази даних вже присутні стовпці "Id", які виступають як унікальний ідентифікатор. Крім того, кожен атрибут містить лише одне значення. З огляду на це, можна стверджувати, що всі відношення бази даних відповідають вимогам 1НФ.

Друга нормальна форма (2НФ) вимагає, щоб кожна сутність містила функціональні залежності між описовими атрибутами та ключовим. Після аналізування всіх відношень бази даних, можна стверджувати, що вони відповідають вимогам 2НФ [10].

Нормалізація до третьої нормальної форми (3НФ) передбачає усунення транзитивних залежностей між описовими атрибутами та ключовим. Після аналізу кожної сутності в проектованій базі даних можна стверджувати, що всі відношення бази даних "CityGov" відповідають вимогам 3НФ.

Для створення бази даних використовується мова Transact SQL, яка дозволяє визначити структуру таблиць та логічні зв'язки між ними за допомогою первинних та зовнішніх ключів [11].

Нижче наведено запит, який створює таблиці "OfDoksTypes", "ProjectRTypes" та "RegPolTypes" для зберігання підтипів документів.

Лістинг 2.1 – SQL-запит для створення таблиць «OfDoksTypes» «ProjectRTypes» «RegPolTypes» бази даних

```
use [SityGov]
CREATE TABLE [OfDoksTypes](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL)
GO
CREATE TABLE [ProjectRTypes](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,)
Go
CREATE TABLE [RegPolTypes](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL)
```

Результатом виконання наступного запиту є створення трьох таблиць «OfDoksDb», «ProjectRDb» та «RegPolDb».

Лістинг 2.2 – SQL-запит для створення таблиць «OfDoksDb», «ProjectRDb» та «RegPolDb» бази даних

```
use [SityGov]
 [Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
[Link] NVARCHAR (50) NULL,
[Date] NVARCHAR (50) NULL,
FOREIGN KEY (Type) REFERENCES OfDoksTypes (Id))
GO
CREATE TABLE [ProjectRDb](
```

Продовження лістингу 2.2

```
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
[Link] NVARCHAR (50) NULL,
[Date] NVARCHAR (50) NULL,
FOREIGN KEY (Type) REFERENCES ProjectRTypes (Id))
Go
CREATE TABLE [RegPolDb](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
[Link] NVARCHAR (50) NULL,
[Date] NVARCHAR (50) NULL,
FOREIGN KEY (Type) REFERENCES RegPolTypes (Id))
```

Наступний запит створює наступні три таблиці: «ProtokolsDb», «ProgramsDb» та «PodDb»

Лістинг 2.3 – SQL запит для створення таблиць «ProtokolsDb», «ProgramsDb» та «PodDb»

```
use [SityGov]
CREATE TABLE [ProtokolsDb](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
GO
CREATE TABLE [ProgramsDb](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
 [Link] NVARCHAR (50) NULL,
[Date] NVARCHAR (50) NULL)
Go
```

Продовження лістингу 2.3

```
CREATE TABLE [PodDb](
[Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
[Name] NVARCHAR (50) NULL,
[Type] INT Null,
[Link] NVARCHAR (50) NULL,
[Date] NVARCHAR (50) NULL)
```

2.3 Проєктування інформаційної системи

З метою зручного логічного представлення роботи системи використовується опис функціональності системи, який включає в себе візуальне відображення послідовності дій, що можуть бути виконані системою відповідно до зовнішніх впливів користувачів або інших програмних систем [12]. При проектуванні інформаційної системи були створені наступні типи діаграм:

– Діаграма використання (Use case diagram).
– Діаграма класів (Class diagram).
– Діаграма послідовності (Sequence diagram).

Діаграма використання відображає використання акторів (користувачів) та інтерфейсів системи, а також відношення між цими елементами. В деяких випадках елементи діаграми можуть бути поміщені в прямокутник, який представляє систему в цілому [13]. На діаграмі використання для сайту документообігу міської ради (рисунок 2.1) кожному актору відповідають тільки ті дії, які він може здійснювати.

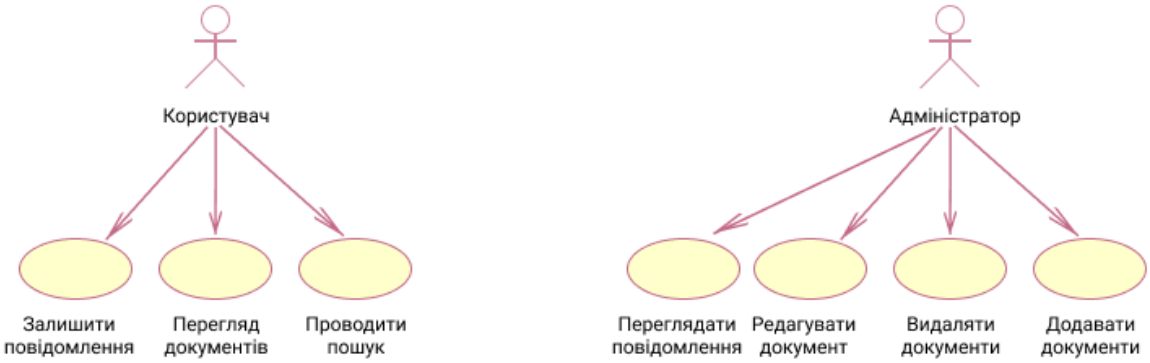


Рисунок 2.1 – Діаграма використання для сайту документообігу міської ради

Діаграма класів використовується для відображення статичних елементів, таких як класи, типи даних, їх структуру та взаємозв'язки [14]. Ця діаграма служить для представлення статичної структури системи в термінології об'єктно-орієнтованого програмування. На діаграмі класів показуються класи, інтерфейси, об'єкти та їх взаємозв'язки [15]. Верхня частина прямокутника обов'язкова і містить назву класу. Друга і третя частини прямокутника можуть бути вказані або пропущені і містять список атрибутів класу і список методів класу відповідно. На діаграмі класів для сайту документообігу міської ради (рисунок 2.2) представлені відповідні класи та їх характеристики [16].

Рисунок 2.2 – Діаграма класів для сайту документообігу міської ради

Діаграма класів отримала свою назву через схожість її структури зі структурою класів у будь-якій об'єктно-орієнтованій мові програмування. Щоб зрозуміти цю аналогію, можна порівняти верхній блок діаграми з назвою класу в ООП, середній блок змінними, а останню частину з методами класу [17].

Діаграма послідовностей відображає взаємодію об'єктів впорядкованих за часом. Вона показує задіяні об'єкти та послідовність відправлених повідомлень. Ця діаграма може бути використана для уточнення діаграм прецедентів та більш детального опису логіки сценаріїв використання. Вона є чудовим інструментом документування проекту з точки зору сценаріїв використання [18]. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і результати, пов'язані з цими повідомленнями. На рисунку 2.3 зображена діаграма послідовностей для сайту документообігу міської ради.

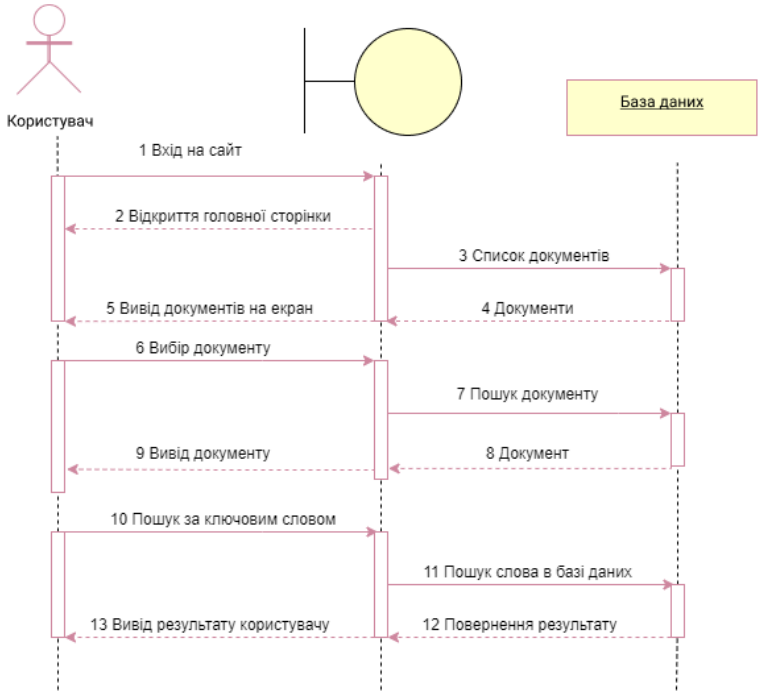


Рисунок 2.3 – Діаграма послідовності для сайту документообігу міської ради

На цій діаграмі показана послідовність дій при вході на сайт документообігу міської ради. Варто зазначити, що ця послідовність не є єдиною правильною, оскільки користувач може почати користування сторінкою з іншого меню, наприклад, меню пошуку. Проте ця послідовність є найбільш ймовірною, якщо ми хочемо протестувати роботу сайту [19].

UML-діаграми можуть бути використані на початковому етапі розробки додатку для полегшення розуміння його функціоналу, а також під час написання коду для представлення змін в функціоналі. В будь-якому випадку вони служать зручним інструментом для усвідомлення логічної частини програми. Тому діаграми є популярними і мають багато варіацій у своєму викладенні.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Проектування інтерфейсу користувача

При старті веб-додатка, користувач відкриває головну сторінку сайту, де відображаються інформація про офіційні документи міської ради, протоколи і останні додані документи. На цій сторінці також розташоване меню навігації, шапка сайту та посилання на соціальні мережі. Зовнішній вигляд цього вікна можна побачити на рисунку 3.1.

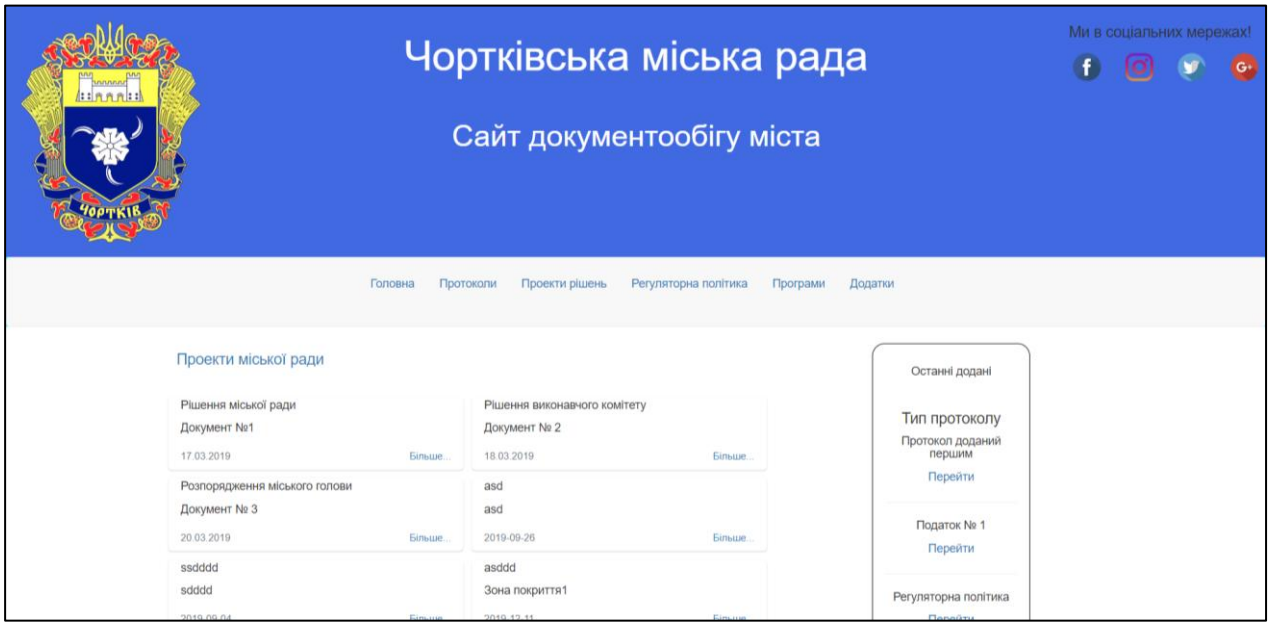


Рисунок 3.1 – Головне меню сайту

При перемиканні між вкладками в навігаційному меню відкриваються подібні вікна, що відрізняються типами документів та параметрами пошуку. На рисунку 3.2 наведено приклад такої сторінки.

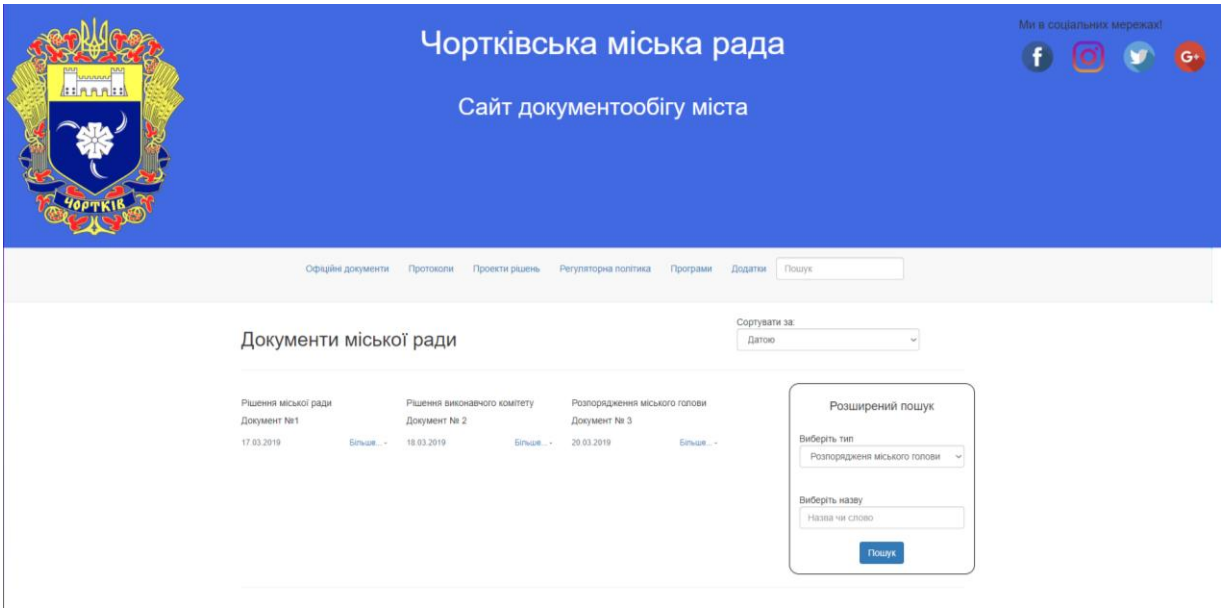


Рисунок 3.2 – Вікно переліку протоколів

У разі потреби отримати більш детальну інформацію про певний акт, користувач може натиснути кнопку " Більше...", що відкриє вікно з додатковими даними. У цьому вікні користувачу буде надана можливість завантажити сам документ, пов'язаний з актом. Приклад такого вікна показано на рисунку 3.3.

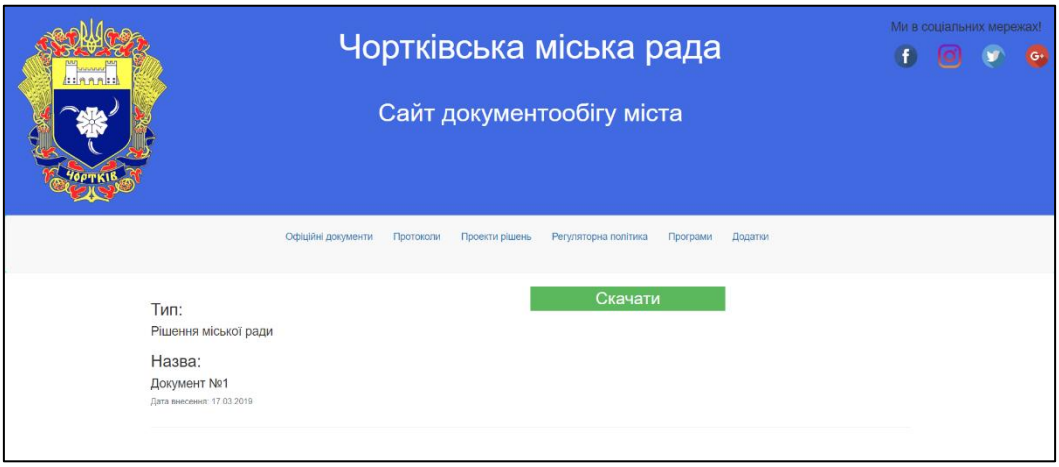


Рисунок 3.3 – Вікно скачування файлу

На створеному веб-сайті також доступна можливість переходу на офіційний веб-сайт Чортківської міської ради. Для цього користувачу

потрібно натиснути на зображення герба, що призведе до переадресації на головну сторінку офіційного сайту (див. рисунок 3.4).

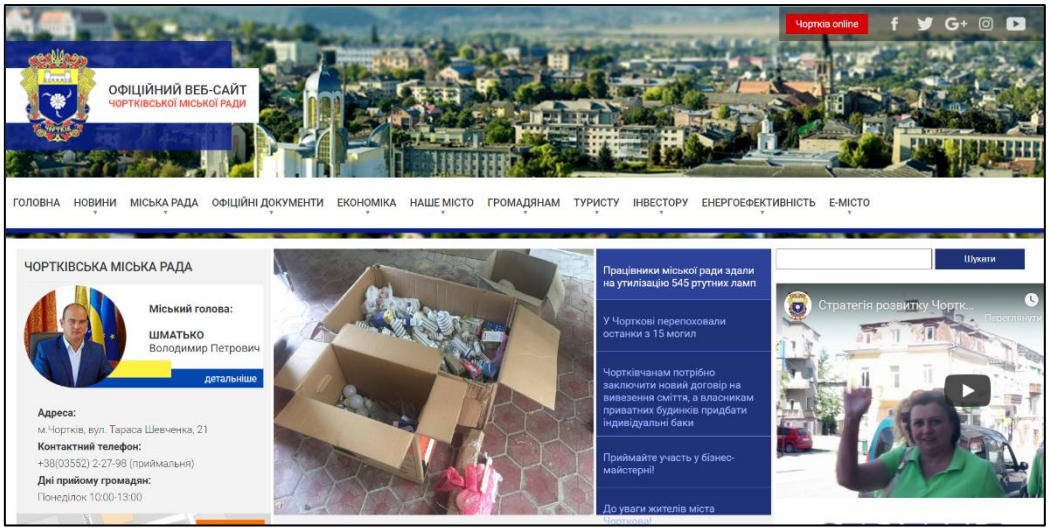


Рисунок 3.4 – Офіційний сайт Чортківської міської ради

З метою зручного внесення інформації в базу даних була розроблена сторінка адміністратора. На цій сторінці адміністратор має можливість додавати, видаляти та редагувати дані в таблицях, а також завантажувати документи для подальшого перегляду користувачами (див. Рисунок 3.5). Це дозволяє забезпечити зручний і ефективний процес керування даними у системі.

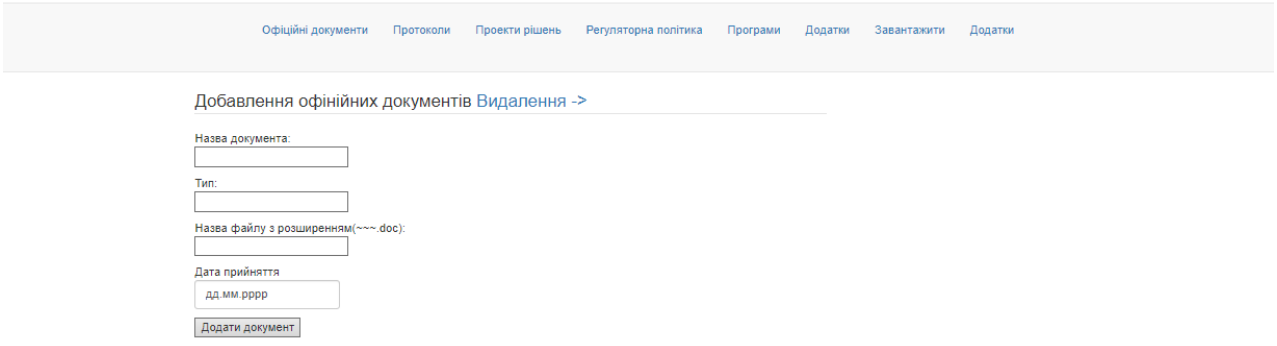


Рисунок 3.5 – загальний вигляд сторінки для адміністратора

При натисканні на пункт «видалення->» на веб-сторінці, відображається вікно, в якому реалізована можливість видалення непотрібного запису (див. рисунок 3.6). Це дозволяє адміністратору з легкістю видаляти обрані дані з бази даних, забезпечуючи актуальність та належну оновленість інформації у системі.

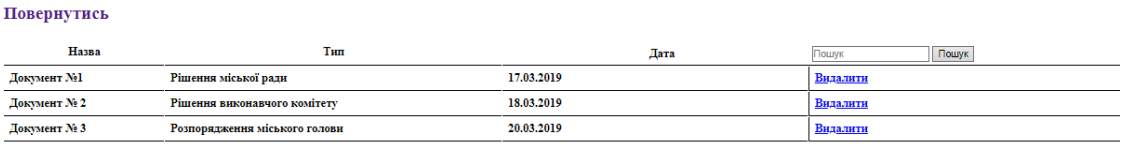


Рисунок 3.6 – Сторінка для видалення даних

3.2 Опис програмних модулів

Для розробки веб-додатку використовувалась технологія ASP.Net з використанням паттерна MVC (Model-View-Controller). Це призвело до поділу всіх файлів проекту на 3 типи:

1. Модель - це файл "DateContext.mdf", який використовується як база даних для зберігання інформації.
2. Контролер - представлений файлом "HomeController.cs". Він відповідає за обробку всіх логічних процесів всередині проекту. У цьому контролері реалізовано алгоритм додавання інформації в базу даних, який наведений у лістингу 3.1.
3. Представлення - це 21 файл, серед яких найважливіші:
   - "Index.cshtml" відображає інформацію про офіційні документи міської ради (лістинг коду наведений у Додатку Б).
   - "ClassForAll.cshtml" є сторінкою, яка використовується як шаблон для виводу інформації.

Загалом, використання технології ASP.Net з паттерном MVC дозволило розподілити компоненти проекту на модель, контролер та представлення, що полегшує розробку та підтримку веб-додатку.

Лістинг 3.1 – Код для додавання інформції в базу даних

```
public ActionResult Info(int? ID, string Name1, string Link1, string Date1){
                SecondClassForAll dok1 = new SecondClassForAll();
                dok1.Id = Convert.ToInt32(ID); dok1.Name = Name1;;
                dok1.Link = Link1;
                dok1.Date = Date1;
                return View(dok1);
            }
```

На верхній частині сайту розташовано статичне меню для навігації, яке реалізовано за допомогою окремого файлу з назвою "_Layout.cshtml". Цей файл містить код, який відповідає за відображення меню та його функціональність. Лістинг коду для цього файлу наведений у лістингу 3.2. Цей шаблон використовується для створення спільного дизайну та навігаційної структури для всіх сторінок сайту.

Лістинг 3.2 – Код меню навігації

```
<!DOCTYPE html>
<html>
<head>
    <link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta charset="utf-8" />
    <link href="~/Content/StyleSheet1.css" rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<style>
    html, body,footer {
        padding: 0;
        margin: 0;

    }
    .headDiv{
        padding: 15px;
    }
```

Продовження лістингу 3.2

```
<body >
    <header>
        <nav class="navbar navbar-default">
            <div class="container-fluid">
                <div class="navbar-header">
                    <ul class="nav navbar">
                                <li>@Html.ActionLink("Офіційні         документи",
"Index", "Home")</li>
                                <li>@Html.ActionLink("Протоколи",     "Protokols",
"Home")</li>
                                <li>@Html.ActionLink("Проекти рішень", "ProjectR",
"Home")</li>
                                <li>@Html.ActionLink("Регуляторна       політика",
"RegPol", "Home")</li>
                                <li>@Html.ActionLink("Програми",       "Programs",
"Home")</li>
                                <li>@Html.ActionLink("Додатки",            "Adds",
"Home")</li><li><li><ul><div><div><nav>
          </header>
          <div class="container body-content">
              @RenderBody()
              <hr />
              <footer>
              </footer>
          </div>
          @Scripts.Render("~/bundles/jquery")
          @Scripts.Render("~/bundles/bootstrap")
          @RenderSection("scripts", required: false)
      </body>
      </html>
```

При розробці веб-орієнтованого додатку за допомогою мови C#, важливо згадати про менеджер пакетів "NuGet". Це безкоштовна система керування пакунками, яка дозволяє зручно встановлювати, видаляти, оновлювати та керувати різними компонентами, такими як бібліотеки класів, фреймворки, сервіси та багато іншого. На рисунку 3.7 показано вигляд інтерфейсу менеджера. Оскільки він інтегрований в Microsoft Visual Studio, вікно керування пакетами знаходиться в навігаційному меню самого середовища розробки.

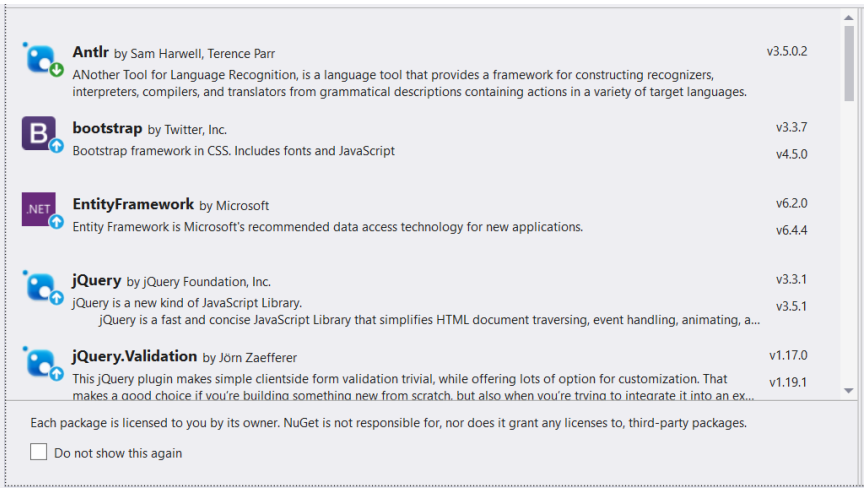


Рисунок 3.7 – Вигляд менеджера пакетів

Серед усіх елементів можна виділити кілька найважливіших:

– «Bootstrap» Framework: це бібліотека стилів та інструментів для верстки сайтів.

– «jQuery»: це бібліотека анімацій та інтерактивних подій, написаних на мові програмування «JavaScript».

– «EntityFramework»: це засіб для роботи з базою даних.

– «WebGrease»: це засіб для оптимізації JavaScript, CSS коду та завантаження зображень на сайт.

Аналізуючи програмні модулі, можна сказати, що в наш час з'явилось багато інструментів, призначених для полегшення роботи розробників. Від бібліотек стилів, які значно економлять час на написанні коду, до пакетів, які автоматично оптимізують роботу додатків.

3.3 Опис результатів тестування

Тестування програмного забезпечення є процесом, який спрямований на перевірку відповідності заявленим вимогам і реально реалізованій функціональності продукту. Воно включає спостереження за роботою програмного забезпечення в умовах, що були створені штучно, а також на обмеженому наборі тестів, обраних певним чином.

Під час тестування можна оцінювати наступні аспекти:

– - Правильна реакція на всі можливі вхідні дані.

– - Виконання функцій за прийнятний час.

– - Практичність використання програмного забезпечення.

– - Сумісність з іншим програмним забезпеченням і операційними системами.

– - Відповідність задачам, які були поставлені замовником.

Існує багато автоматизованих методів для проведення тестування програмного забезпечення. Для перевірки сайту документообігу ми проведемо кілька тестів, які наведені в таблиці 3.1.

Таблиця 3.1 – Створення набору test-case

ID	Тип	Опис	Очікувано	Реально	Pass/ Fail
1	2	3	4	5	6
01	negative	Перевірка сайту на крос браузерність	Сайт повинен відображатись одинаково у браузерах	В різних браузерах сайт має різний вигляд	Fail
02	positive	Перевірка валідності посилань на панелі навігації	Усі гіперпосилання відкривають очікувану	Усі гіперпосилання відкривають очікувану	Pass
03	Positive	Правильність внесення даних	При внесенні інформації за допомогою адмін – панелі дані вносяться правильно	При внесенні інформації за допомогою адмін – панелі дані вносяться правильно	Pass

Продовження таблиці 3.1

1	2	3	4	5	6
04	Positive	Скачування відповідного документу	При натисканні кнопки «Скачати» скачує відповідний документ	При натисканні кнопки «Скачати» скачує відповідний документ	Pass

Також, одним з популярних тестів для веб сторінок є перевірка адаптивності на мобільних пристроях (див. рис 3.8).

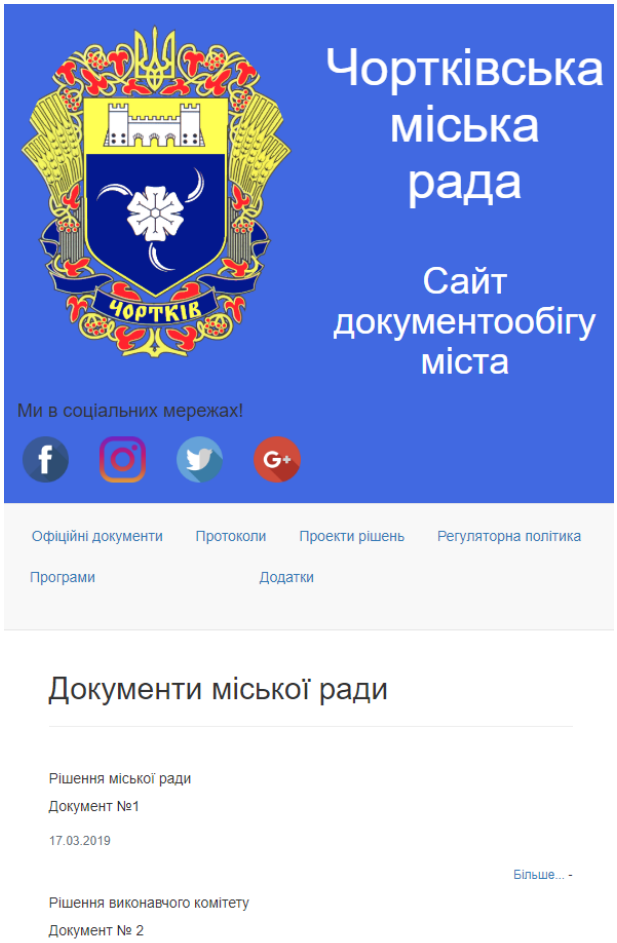


Рисунок 3.8 – Адаптивність під мобільні розширення

Оскільки сайт був розроблений з використанням бібліотеки Bootstrap, яка забезпечує можливості для адаптивного дизайну, можна стверджувати, що цей веб-додаток успішно пройшов цей тест.

З кожним роком вимоги до розробки веб-сайтів стають все більшими. Одним з критеріїв стає "валідність коду". Валідність коду означає відповідність вихідного коду сайту нормам і правилам, встановленим Консорціумом Всесвітньої Павутини (W3C). Для проведення цього тесту ми скористаємося інтернет-ресурсом під назвою "validator.w3", де перевіримо першу сторінку нашого сайту (див. рис. 3.9).

Рисунок 3.9 – Тестування на валідність коду

Існує багато додатків, які дозволяють перевірити веб-сайти за різними критеріями, але на мою думку, найзручнішими є браузерні розширення. З їх допомогою можна легко перевірити потрібну інформацію без потреби закривати веб-сторінку. Тому для аналізу нашого веб-додатку ми скористаємося одним з таких розширень, яке називається "Web Developer Checklist" (див. рис. 3.10). Після огляду звіту про виконану роботу можна сказати, що наш сайт є адаптивним для різних розширень екрану, він має швидку продуктивність, що забезпечує швидке завантаження сторінок, і він є зручним та легким у використанні. Варто відмітити, що вкладка "Валідність HTML коду" показує негативний результат тестування. Це пов'язано з тим, що через використання технології ASP.NET, код вставляється безпосередньо в розмітку сторінки, що автоматичне тестування вважає неправильним за певними правилами.

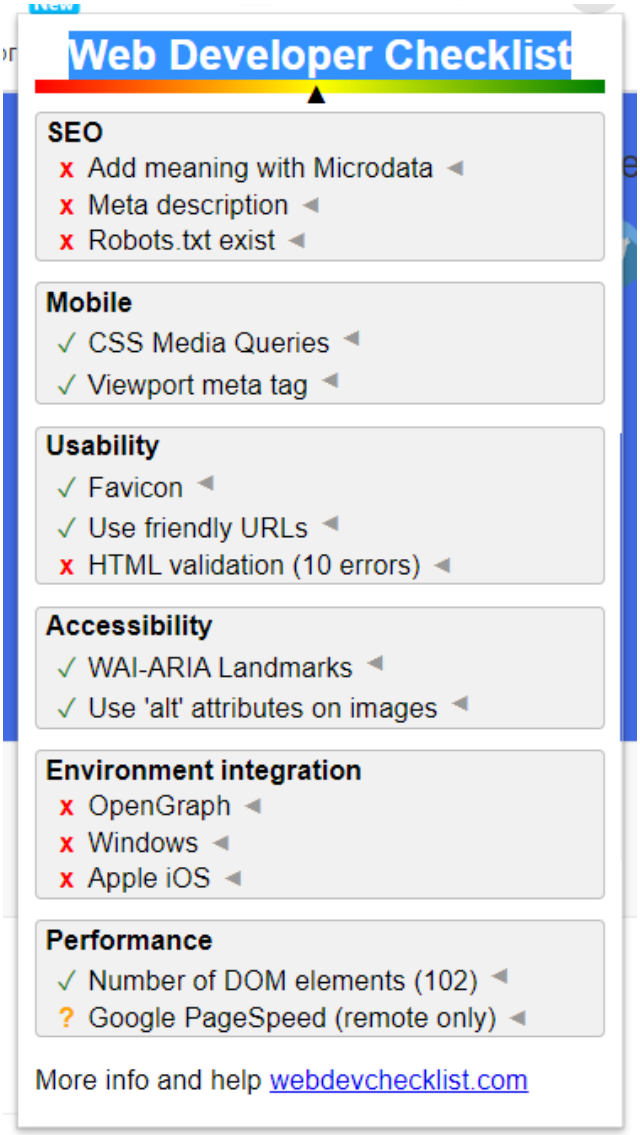


Рисунок 3.10 – Перевірка сайту за допомогою браузерного розширення

Отже, після проведення декількох тестів, варто відзначити, що створення ідеального додатку, який б не мав жодних помилок, є практично неможливим завданням. Тому проведення таких перевірок є необхідною складовою частиною життєвого циклу будь-якого продукту, незалежно від того, чи є це програма чи проста річ. На щастя, сьогодні існує велика кількість тестів, які призначені для різних завдань, починаючи з перевірки валідності коду і закінчуючи моніторингом вихідного трафіку додатку.

4 Безпека життєдіяльності, охорони праці

4.1 Актуальність безпеки життєдіяльності людини

В сучасному світі забезпечення безпеки життєдіяльності людини стає все більш актуальною та важливою проблемою. Розуміння і свідоме ставлення до ризиків та загроз, які можуть впливати на безпеку людей у різних сферах їхнього життя, стає необхідністю для забезпечення якісного та тривалого існування. Отже, детальний аналіз актуальності безпеки життєдіяльності людини в сучасному світі вкрай важливий [20].

Актуальність безпеки життєдіяльності людини виходить з різних аспектів і має широкий зв'язок зі всіма сферами людського життя. Перш за все, фізична безпека є невід'ємною частиною життя кожної людини. Загрози такі, як природні катаклізми, аварії, злочинність та терористичні акти, можуть серйозно підірвати безпеку і навіть життя людей. Тому розуміння актуальності безпеки життєдіяльності з точки зору фізичної безпеки стає першочерговим завданням.

Окрім фізичної безпеки, актуальність безпеки життєдіяльності також пов'язана з охороною здоров'я та безпекою у сфері оточуючого середовища. Забруднення повітря, води та ґрунту, незбалансоване харчування та несанкціоноване використання хімічних речовин можуть негативно впливати на здоров'я людини та призводити до різних захворювань. Тому особлива увага до актуальності безпеки життєдіяльності повинна бути зосереджена на збереженні здоров'я та забезпеченні безпечного середовища для людей.

Також, актуальність безпеки життєдіяльності проявляється в сфері соціальної безпеки. Соціальні загрози, такі як безробіття, бідність, насильство, дискримінація та інші форми соціальної небезпеки, можуть суттєво впливати на якість життя людей та порушувати гармонію та стабільність суспільства. Отже, розуміння актуальності безпеки

життєдіяльності з соціального погляду є ключовим для створення справедливого та безпечного соціуму [21].

У світі, який швидко змінюється, актуальність безпеки життєдіяльності людини стає все більш складною задачею. Глобалізація, технологічний прогрес та інші фактори створюють нові загрози та виклики для безпеки людей. Важливо адаптуватися до цих змін та розробляти ефективні стратегії для забезпечення безпеки в усіх сферах життя.

У підсумку, актуальність безпеки життєдіяльності людини є невід'ємною складовою сучасного світу. Розуміння основних аспектів безпеки, таких як фізична безпека, охорона здоров'я, соціальна безпека та врахування сучасних викликів та тенденцій, допомагає створювати безпечне та стійке середовище для життя та розвитку людей. Тільки шляхом постійного аналізу та удосконалення стратегій безпеки можна забезпечити належний рівень безпеки для всіх людей та покращити їхнє життя.

4.2. Інженерно-психологічні принципи професійного добору

Професійний добір є невід'ємною складовою успішної кар'єри та ефективного функціонування організацій. Його мета полягає в відборі кандидатів, які мають відповідні навички, здібності та характеристики для виконання конкретних професійних завдань. У цьому контексті, інженерно-психологічні принципи професійного добору набувають особливої актуальності та значимості.

Один з інженерно-психологічних принципів професійного добору полягає в аналізі вимог до конкретної посади або ролі. Це передбачає ретельне вивчення функцій, обов'язків та компетенцій, які необхідні для успішного виконання роботи. Інженерно-психологічний підхід допомагає

визначити ключові критерії та вимоги, які повинні бути задоволені кандидатом [22] .

Другий принцип полягає в розробці та використанні ефективних методів та інструментів для оцінки потенційних кандидатів. Це включає психологічні тести, інтерв'ю, оцінку здатностей та здібностей. Інженерно-психологічні методи дозволяють об'єктивно оцінити особистісні риси, мотивацію, комунікативні навички та інші фактори, які можуть впливати на успішність кандидата у конкретній посаді.

Третій принцип пов'язаний з адаптацією професійного добору до конкретного контексту та організаційної культури. Кожна організація має свої особливості, цінності та вимоги, які необхідно враховувати під час процесу добору. Інженерно-психологічні принципи дозволяють адаптувати методи та критерії добору до конкретного контексту, що забезпечує кращу злагоду між кандидатами та організацією.

Четвертий принцип включає систематичний моніторинг та оцінку ефективності професійного добору. Це передбачає збір та аналіз даних про успішність нових працівників, їхню продуктивність та задоволеність роботою. Інженерно-психологічний підхід дозволяє виявити потенційні проблеми та вдосконалити процес добору для майбутніх вакансій.

Висновок, інженерно-психологічні принципи професійного добору є необхідними для ефективного відбору та підбору кандидатів на різні посади. Вони допомагають забезпечити відповідність між вимогами посади та характеристиками кандидатів, що веде до покращення продуктивності та ефективності організації. Інженерно-психологічний підхід дозволяє раціоналізувати процес добору та забезпечити належний рівень об'єктивності та надійності в процесі прийняття рішень щодо відбору кандидатів.

4.3 Проведення інструктажів з охорони праці.

Проведення інструктажів з охорони праці є важливим етапом забезпечення безпеки та запобігання нещасним випадкам на робочому місці. В дипломній роботі студента з інформатики можна розглянути питання проведення інструктажів з охорони праці та дослідити їх ефективність в інформаційній галузі [23].

Інструктажі з охорони праці мають на меті ознайомити працівників з потенційними небезпеками на робочому місці, правилами безпеки та процедурами дій у випадку надзвичайних ситуацій. Вони повинні бути проведені перед початком роботи, при зміні умов праці, а також при введенні нового обладнання або технологій.

У дипломній роботі можна дослідити ефективність проведення інструктажів з охорони праці в інформатичній галузі та їх вплив на забезпечення безпеки працівників. Розглянутий може бути процес підготовки та проведення інструктажів, а також оцінка рівня розуміння та дотримання працівниками правил безпеки.

Дослідження можуть включати аналіз поточних практик проведення інструктажів, виявлення можливих проблем та пропозиції щодо їх вдосконалення. Також можна провести опитування серед працівників для з'ясування їхнього ставлення до інструктажів та ефективності отриманої інформації.

Висновки з дослідження можуть допомогти розробити рекомендації та практичні рішення для поліпшення процесу проведення інструктажів з охорони праці в інформатичній галузі, що сприятимуть забезпеченню безпеки працівників та запобіганню нещасним випадкам.

ВИСНОВКИ

Документаційні та інформаційні ресурси міської ради формуються на основі документування управлінської діяльності відділів та управлінь. Ці документи фіксують різні форми виконання обов'язків, покладених на них міською радою. Інформація також розміщується на вебсайтах, де доступні документи з адміністративно-управлінської діяльності міської ради, завдання, нормативно-правові акти, акти регуляторної діяльності, правила реєстрації, ліцензування та зразки документів для звернень громадян. Також на сайті розміщуються документи, що стосуються виконання бюджету, цільових програм, комунальних послуг та інше.

У результаті дипломного проєкту було розроблено вебдодаток, який надає відкритий доступ до офіційних документів міської ради. Для цього було проведено аналіз документації міської ради, створено базу даних "cityGov" та розроблено вебсайт.

Користувачі вебдодатку мають можливість завантажити цікавлячий їх документ для подальшого ознайомлення.

У розробці вебсайту використовувалась технологія ASP .NET з використанням паттерна MVC. Для створення бази даних використовувалися SQL-запити.

ПЕРЕЛІК ПОСИЛАНЬ

1. UI дизайн URL: https://habr.com/ru/post/321312 (дата звернення: 25.04.2019).

2. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань : навч. посіб. Львів : Магнолія-2006, 2012. 584 с.

3. Гарнаев А. Самонавчання Visual Studio .NET 2013 СПб. : БХВ-Пітербург, 2013, 336 с.

4. Гороховатський В. О. Визначення трудомісткості при розробленні програмних комплексів. Системи обробки інформації. 2014. Вип. 2. С. 92-98. URL: http://nbuv.gov.ua/UJRN/soi_2014_2_22 (дата звернення 03.05.2020).

5. Грофф Джеймс Р. SQL:повне керівництво : пер. с англ. М. : Вильямс, 2015. 960 с.

6. Конноллі Томас. Бази даних: проектування, реалізація і впровадження. Теорія и практика : нав. рук., пер. с англ. СПб. : BHV, 2011. 1096 с.

7. Методичні вказівки до виконання дипломного проекту для студентів спеціальності 121 «Інженерія програмного забезпечення» / упоряд. К. І. Барціховська, Н. В. Оляніна, Я. І. Баумкетнер : Вид-во ГК ТНТУ, 2014. – 52 с.

8. Методичні вказівки до виконання економічного розділу дипломної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / упоряд. К. І. Барціховська. Гусятин : Вид-во ГК ТНТУ, 2020. 39 с.

9. Методичні вказівки до виконання курсової роботи з дисципліни «Бази даних» для спеціальності 5.05010301 «Розробка програмного забезпечення» / упоряд. Н. В. Оляніна. Гусятин : Вид-во ГК ТНТУ, 2015. 49 с.

10. Пасічник В. В., Резніченко В. А. Організація баз даних та знань : підруч. для вузів. К. : Видавнича група BHV, 2006. 384 с.

11. Положення про автоматизовану систему документообігу суду. URL: http://court.gov.ua/sudova-vlada/969076/polozhenniapasds/ (дата звернення 08. 03. 2017).

12. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин : наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65. URL: http://zakon2.rada.gov.ua/laws/show/z0293-10 (дата звернення: 10.05.2020).

13. Про затвердження Примірної інструкції з охорони праці під час експлуатації електронно-обчислювальних машин : наказ Міністерства доходів і зборів України від 5.09.2013 № 443. URL: http://minrd.gov.ua/ Prim_rna_nstrukts_ya.doc (дата звернення: 02.05.17).

14. Робота с документами. URL: http://www.softportal.com/software-25233-rabota-s-dokumentami.html (дата звернення: 30.03.2019).

15. Ріхтер Джеффрі. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.0 на мові C#. СПб. : 2012. 459 с.

16. Розрахунок собівартості ПЗ. URL: https://studfiles.net/preview/ 5466695/page:3 (дата звернення: 6.05.2019).

17. Ручне тестування URL: https://uk.wikipedia.org/wiki/Ручне_тестування (дата звернення: 6.05.2019).

18. Облік документів. URL: http://www.softportal.com/software-42131-uchet-dokumentov-elektronnij-arhiv-dokumentov.html (дата звернення: 30.05.2019).

19. Шарп Дж. Microsoft Visual C#. Повний посібник. СПб. : Пітер, 2017, 848 с.

20. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. – Львів: Манголія 2006, 2007. 499 с.

21. Зеркалов Д.В. Безпека життєдіяльності. Навчальний посібник. - К.: Основа, 2011.

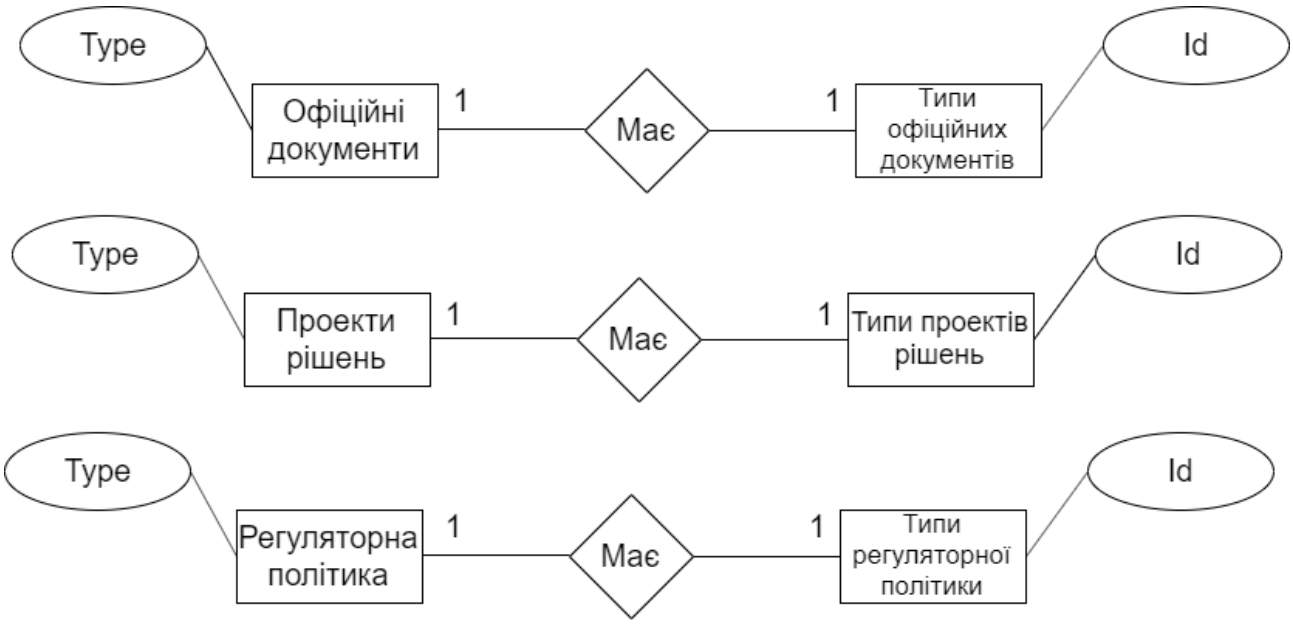
22. Купчик М.П., Гандзюк М.П., Степанець І.Ф. та ін. Основи охорони праці. – К.: Основа, 2000. 416 с.

23. Основи охорони праці. / Під ред. Ткачука К.Н., Халімовського Н.О. – К.: Основа, 2006. 448 с.

ДОДАТКИ

Додаток А

ER–діаграма бази даних «SityGov»



Додаток Б

Код файлу «HomeController.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;
using WebApplication1.Models;
using System.IO;
using System.Net;
using System.Web;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
namespace WebApplication1.Controllers
{
    public class HomeController : Controller
    {
        DateContext db = new DateContext();
        public ActionResult Upload(HttpPostedFile file)
        {
            string fname = Path.GetFileName(file.FileName);
            if (file != null && file.ContentLength > 0)
            {

                file.SaveAs(Server.MapPath(Path.Combine("~/Files/", fname)));
            }
            return RedirectToAction("AdminForm");
        }
        public void LastAdd(string s1, string s2, string s3, string s4, string s5)
        {
            int idd;
            foreach (var item in db.LastOneDs)
            {
                idd = item.Id;
                LastOneD mas = db.LastOneDs.Find(idd);
                db.LastOneDs.Remove(mas);
                LastOneD mas1 = new LastOneD();
                mas1.Id = Convert.ToInt32(s1);
                mas1.Name = s2;
                mas1.Type = s3;
                mas1.Link = s4;
                mas1.Date = s5;
                db.LastOneDs.Add(mas1);
                db.SaveChanges();
                break;
            }
        }
        public ActionResult OfdoksDel1()
        {
            var mas = db.OfDoksDs;
            return View(mas.ToList());
        }
        [HttpGet]
        public ActionResult MyAction(string searchString)
        {
            OfDoksD aaa = new OfDoksD();
            ClassForAll[] al1 = new ClassForAll[50];
            int n = 0;
            foreach(var a in db.OfDoksDs){
```

Продовження додатку Б

```
if (a.Name == searchString)
            {
                aaa = a;
                break;
            }
        }
        return View(aaa);
        foreach (var item in db.OfDoksDs.ToList())
        {
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = item.Type;
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        foreach (var item in db.ProjectRDs)
        {
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = item.Type;
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        foreach (var item in db.RegPolDs)
        {
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = item.Type;
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        foreach (var item in db.PodDs)
        {
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = "none";
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        foreach (var item in db.ProjectRDs)
        {
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = "none";
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        foreach (var item in db.RegPolDs)
```

Продовження додатку Б

```
            if (searchString == item.Name)
            {
                al1[n].Name = item.Name;
                al1[n].Type = "none";
                al1[n].Date = item.Date;
                al1[n].Link = item.Link;
                n++;
            }
        }
        return View(al1.ToList());
    }
    protected void btnUploadClick(HttpPostedFile file)
    {
        //HttpPostedFileBase file = Request.Files["MyFile"];
        if (file != null && file.ContentLength > 0)
        {
            string fname = Path.GetFileName(file.FileName);
            file.SaveAs(Server.MapPath(Path.Combine("~/Files/", fname)));
        }
    }
    public ActionResult OfdoksDel11(int? id)
    {
        OfDoksD mas = db.OfDoksDs.Find(id);
        db.OfDoksDs.Remove(mas);
        db.SaveChanges();
        return RedirectToAction("OfdoksDel1");
    }
    public ActionResult DownloadFiles()
    {
        return View();
    }
    public ActionResult Index()
    {
        var doks1 = db.OfDoksDs;
        return View(doks1.ToList());
    }
    public ActionResult OfDocs()
    {
        var doks1 = db.OfDoksDs;
        return View(doks1.ToList());
    }
    public ActionResult Info(int? ID, string Name1, string Link1, string Date1)
    {
        SecondClassForAll dok1 = new SecondClassForAll();
        dok1.Id = Convert.ToInt32(ID);
        dok1.Name = Name1;;
        dok1.Link = Link1;
        dok1.Date = Date1;
        return View(dok1);
    }
    public ActionResult Programs()
    {
        var doks1 = db.ProgramsDs;
        return View(doks1.ToList());
    }
    public ActionResult Adds()
    {
        var doks1 = db.PodDs;
        return View(doks1.ToList());
    }
    public ActionResult RegPol()
    {
        var doks1 = db.RegPolDs;
```

Продовження додатку Б

```
        return View(doks1.ToList());
         }
         public ActionResult Protokols()
         {
             var doks1 = db.ProtokolsDs;
             return View(doks1.ToList());
         }
         public ActionResult ProjectR()
         {

             var doks1 = db.ProjectRDs;
             return View(doks1.ToList());
         }
         public ActionResult About()
         {
             ViewBag.Message = "Your application description page.";

             return View();
         }
         public ActionResult Contact(int? ID, string Name1, string Type1, string Link1,
string Date1)
         {
             ClassForAll dok1 = new ClassForAll();
             dok1.Id = Convert.ToInt32(ID);
             dok1.Name = Name1;
             dok1.Type = Type1;
             dok1.Link = Link1;
             dok1.Date = Date1;
             return View(dok1);
         }
        [HttpGet]
         public FileResult GetFile(string FileName)
         {
             string file_path = Server.MapPath("~/Files/"+ FileName);
             string file_type = "application/doc";
             string file_name = FileName;
             return File(file_path, file_type, file_name);
         }
         public ActionResult AdminForm()
         {
            return View();
         }
         [HttpPost]
         public ActionResult AdminForm(OfDoksD ofDoks1)
         {
             db.OfDoksDs.Add(ofDoks1);
             db.SaveChanges();
             LastAdd(ofDoks1.Id.ToString(), ofDoks1.Name, ofDoks1.Type, ofDoks1.Link,
ofDoks1.Date );
             return Redirect("AdminForm");
         }

    }
}
```

Додаток В
Код файлу «StyleSheet1»

```
header{
    top: 0;
    background-color:aqua;

}
aside{
    border: solid 1px black;
    padding:15px;
    border-radius:15px;
}
section {
}
.text_main{
    padding-top:15px

}
.imgmn{
    padding-left:10%;
    padding-right:10%;
}
li{
    text-decoration:none;
}
h4{
    text-decoration-color:black;
}
.pageBlock{
    box-decoration-break:clone;
    text-decoration:none;
    width:auto;
    height:100%;
    margin-bottom: 0px;
    bottom:0;
}
a{
    text-decoration:none;
    height:auto;
    width:auto;
}
.pageBlock:hover {
    text-decoration: none;
    display: block;
    height: 100%;
    width: 100%;
    a

{
    text-decoration: none;
    color: #1dfe94;
}
a:hover {
    text-decoration: none;
    color: #1dfe94;
}
.btn {
    background-color: #1E90FF;
    color: #FFE4B5;
}
.asideButt {
    height: auto;
    width: auto;
```

Продовження додатку В

```
background: #40E0D0;
    border: solid 2px black;
    border-radius: 15px;
}
.OfDocsD{
    width:auto;
    padding: 10px;
    margin-right:5px;
    margin-bottom:5px;
    border: solid 2px black;
    font-size:20px;
}
.spanOf {
    margin:5px;
    padding:5px;
}
.OfDivD{
    border:solid 2px black;
}
```