НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____20__ р.

Дипломна робота
на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп'ютерні науки
(код і назва)

на тему: Сайт розрахунку цифрових фільтрів. Інтерфейс користувача ___
_____
_____

Виконав (-ла): студент __4__ курсу, групи ____ДА-32_____
(шифр групи)

_____Войтенко Павло Олександрович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник_____старший викладач Бритов О.А ____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний Рощина Н. В._____доцент к. е. н.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____старший викладач Бритов О.А._____ ____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Войтенку Павлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Сайт розрахунку цифрових фільтрів. Інтерфейс користувача, керівник роботи_____старший викладач Британ О.А_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10»___05____ 2017 р. № 1477-с

2. Термін подання студентом роботи _____09.06.2017 _____

3. Вихідні дані до роботи
Розробити інтерфейс користувача для додатку, що виконує розрахунок рекурсивних цифрових фільтрів низьких частот, високих частот, смугових та режекторних фільтрів з використанням апроксимуючих функцій Батерворта, Чебишова та Кауера. Вхідні дані: межі та відхилення смуг пропускання та затримання, частота дискретизації, тип фільтру та апроксимації, порядок фільтру. Вихідні дані: мапа нулів та полюсів фільтру, передаточна функція у каскадній формі, графіки АЧХ, ЛАЧХ, ФЧХ, імпульсної та перехідної характеристик.

4. Зміст роботи
1) Проаналізувати сучасні особливості та технологій розробки сайтів;
2) Обрати програмні засоби для розробки та алгоритм взаємодії;
3) Розробити інтерфейс користувача для сайту розрахунків цифрових фільтрів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н. В., доцент.		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1			
2			
3			
4			
5			
6			

Студент _____ П.О. Войтенко

(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ О.А. Бритов

(підпис) (ініціали, прізвище)

_____

АНОТАЦІЯ

бакалаврської дипломної роботи Войтенка Павла Олександровича
на тему: «Сайт розрахунку цифрових фільтрів. Інтерфейс користувача»

Дипломна робота присвячена розробці інтерфейсу користувача для сайту розрахунку цифрових фільтрів.

В роботі досліджено сучасні особливості розробки веб-сайтів, було визначено головні вимоги до ергономічності веб-сайтів.

Технологію для розробки було обрано ASP.Net Core, а зв’язок веб-сайту із веб-сервісом розрахунків здійснено на основі протоколу SOAP.

Створений в ході роботи сайт побудований за паттерном MVC і складається із: 1 моделі, 5 представлень (з них 4 - часткових) та 1 контроллеру.

Програмна частина сайту складається з наступних модулів:

- Допоміжний модуль для побудови HTML-розмітки ;
- Модуль-будівник математичних формул;
- Модуль валідації;
- Модуль запитів до сервісу.

Загальний обсяг роботи – 74 сторінки, 29 рисунків, 6 таблиць, 10 посилань.

Ключові слова: цифровий фільтр, веб-сайт, інтерфейс користувача, ергономічність, ASP.Net Core.

АНОТАЦИЯ

бакалаврской дипломной работы Войтенко Павла Александровича

На тему: «Сайт расчета цифровых фильтров. Пользовательский интерфейс»

Дипломная работа посвящена разработке пользовательского интерфейса для сайта расчетов цифровых фильтров.

В работе исследуются современные особенности разработки веб-сайтов, были определены главные требования к эргономичности и юзабилити веб-сайтов.

Технологией для разработки была выбрана ASP.Net Core, а связь веб-сайта с веб -сервисом расчетов будет осуществляться на основе протокола SOAP.

Созданный в процессе работы сайт построен по паттерну MVC и состоит из: 1 модели, 5 представлений (из них 4 – частичніе) и одного контроллера.

Програмная часть сайта состоит из следующих модулей:

- Вспомогательній модуль построения HTML-разметки;
- Модуль-строитель математических формул;
- Модель валидации;
- Модель запросов к сервису.

Общий объем работы – 74 страницы, 29 рисунков, 6 таблиц, 10 ссылок.

Ключевые слова: цифровой фильтр, веб-сайт, пользовательский интерфейс, эргономичность, ASP.Net Core/

ANNOTATION

On Pavlo Voitenko bachelor's degree

thesis: "Digital filter calculation website. User interface"

This thesis is devoted to development of user interface for digital filter calculation website.

In this thesis modern features of web development are researched, the main requirements for ergonomics and usability of websites.

ASP.Net Core was chosen as a base technology for development, the communication with calculation web service is carried out via SOAP protocol.

The website created in scope of this thesis is based on the MVC pattern and consists of: 1 model, 5 views (4 of them – partial), and 1 controller.

Software part of the site consists of the following modules:

- HTML-builder module;
- Mathematic formula builder module;
- Validation module;
- Service request module.

The total volume of the work – 73 pages, 29 images, 6 tables, 10 references.

Keywords: digital filter, website, user interface, ergonomics, ASP.Net Core.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASP.Net Core - технологія від компанії Microsoft, призначена для створення різного роду веб-додатків

NuGet – клієнтський інструмент для створення та використання програмних пакетів.

HTTP – HyperText Transfer Protocol

IIS - Internet Information Services - набір серверів для декількох служб Інтернету від компанії Майкрософт.

MVC – паттерн Model-View-Controller

SOAP - протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML.

RPC – Remote procedure call

XML - eXtensible Markup Language

WSDL - Web Services Description Language

AJAX - Asynchronous Javascript and XML

POST-запит – HTTP-запит, при якому веб-сервер приймає данні, що містяться в тілі сообщенияповідомлення, для збереження.

ВСТУП

На сучасному етапі розвитку електроніки цифрові фільтри – це обов'язковий елемент будь-якої цифрової схеми. Цифрові фільтри використовуються практично всюди, де необхідна обробка електричних сигналів: у електричних схемах різноманітних пристроїв, мікропроцесорів, у спектральному аналізі тощо.

Задача розрахунку коефіцієнтів цифрового фільтру не нова. Її рішення доступні на багатьох платформах та у безлічі видів. Щоправда, кожне рішення має свої недоліки та переваги.

Актуальність.

У теперішній час розробка програмних та апаратних продуктів націлена на зменшення необхідного часу розробки. Саме тому при розробці пристроїв, що використовують цифрові фільтри, гостро постає проблема швидкого розрахунку їх параметрів.

Існуючі рішення умовно поділяються на дві групи: комплексні програмні засоби та веб-сайти. Комплексні програмні засоби, як наприклад Matlab, гарно пристосовані до вирішення такої задачі, але в більшості випадків їх використання недоцільне через велику надмірність невикористаних функції та, зазвичай, високу вартість таких продуктів.

Веб-сайти, які надають послуги розрахунків цифрових фільтрів зазвичай мають не зручний для користувача інтерфейс, або мають доступними лише обмежену кількість функцій. Також часто недоліком є використання іноземної мови для сайту.

Саме тому актуальною виглядає розробка веб-сайту, який надавав користувачам можливість розраховувати цифрові фільтри та мав би зручний інтерфейс.

Мета.

Метою даної роботи є розробка користувацького інтерфейсу сайту розрахунку цифрових фільтрів із врахуванням сучасних тенденцій веб-розробки та дизайну. Розробка дизайну включає розробку макету сторінки з урахуванням сучасних вимог до ергономічності веб-сторінок.

Основні завдання.

До основних завдань роботи можна віднести:

- Дослідження особливостей розробки дизайну веб-сторінок;
- Визначення вимог ергономіки до веб-сайтів;
- Обґрунтувати вибір інструментальних засобів;
- Розробити макет сайту;
- Розробити сайт розрахунку цифрових фільтрів.

Об’єкт дослідження – сайт розрахунку цифрових фільтрів.

Предмет дослідження – інтерфейс користувача сайту розрахунку цифрових фільтрів.

1 АНАЛІЗ ПРИНЦИПІВ І ТЕХНОЛОГІЙ СТВОРЕННЯ САЙТІВ

1.1 Основні особливості розробки веб-сторінок

У сучасному соціокультурному середовищі, веб-інфраструктура є одним з найбільш важливих джерел інформації, а також засобів спілкування, за допомогою ресурсів, які називаються веб-сайтами. При цьому розвиток самої мережі інтернет, який відбувся за невеликий навіть за сучасними мірками проміжок часу, призвів до бурхливої еволюції інформаційних ресурсів - вузлів мережі, веб-сторінок.

Якщо на початку свого розвитку, інформація, що розміщувалася на сторінках сайтів вже сама по собі, представляла ключову цінність, то на сьогодні, однієї тільки інформації для успіху ресурсу було недостатньо.

Причини цього лежали в розвитку самої мережі, а також у збільшенні як користувачів мережі, так і творців інформаційного наповнення - контенту. [1, с 17] На початковому етапі, вкрай обмежена кількість розробників та споживачів контенту, цілком була задоволена, примітивним, текстовим інтерфейсом, практично позбавленим інтерактивних елементів, оскільки вже ці можливості, дозволяли їм розміщувати і споживати інформацію в режимі реального часу, без трудомісткого і витратного за часом традиційного розміщення інформації в паперових виданнях. Однак, з розвитком мережі, зі збільшенням користувачів, змінилася і сама парадигма інформаційного наповнення мережі.

Мережа інтернет знаходиться в постійній трансформації, як через розвиток самої інфраструктури мережі, яка забезпечує зростання швидкості і обсяги обміну інформацією, так і розповсюдження мережі у більш віддалені місця планети; так і через появу інноваційних пристроїв створення і споживання контенту. Так якщо, всього 5-6 років тому, основна маса пристроїв, підключених до мережі інтернет представляла собою стаціонарні комп'ютери, то вже сьогодні це місце зайняли різноманітні мобільні пристрої.

Самі по собі веб-технології також набули значного розвитку. В першу чергу це тісно пов'язано з частою появою нових версій браузерів і підтримкою ними нових технологій. Кожна така технологія, чи фреймворк спрощують ті, чи інші аспекти веб-розробки та надають нові можливості безпосередньо розробнику. На сьогодні існує безліч технологій, що дозволяють легко створювати функціональні веб сторінки у короткі проміжки часу, поєднувати їх зі системами управління базами даних та легко масштабувати, підтримувати мобільні версії сторінок.

Як результат маємо, головний акцент у розробці веб додатків зміщується з безпосередньої розробки внутрішньої, технічної частина на розробку «правильного» дизайну, з урахуванням майбутнього контенту та цільової аудиторії. [2]. Це пояснюється тим, що більшість практичних проблем розробки вже були вирішені попереднім поколінням розробників, чи легко усуваються використанням тієї, чи іншої платформи розробки.

Таким чином, основну увагу при розробці користувацького інтерфейсу слід приділити саме розробці «правильного» дизайну.

1.1.1 Ергономіка як основний показник якості дизайну сайту

Призначення інтерфейсу - бути для користувача зручним і безпечним інструментом отримання, аналізу та передачі інформації, необхідної користувачеві для вирішення різноманітних життєвих і професійних завдань. Щоб відповідати своєму призначенню інтерфейс повинен бути ергономічним, тобто доступним в засвоєнні, зручним в експлуатації, відповідати запитам і можливостям користувачів, зберігати їх здоров'я і працездатність, мати естетичну привабливість.

Ергономіка - наукова дисципліна, що комплексно вивчає діяльність людини (групу людей), що використовує (використовують) технічні засоби і технології.

В контексті розробки веб-сторінок ергономіка може бути визначена як здатність ефективно реагувати на потреби користувачів і забезпечувати їм комфорт при перегляді сторінки.

У загальному сенсі слова термін ергономіка - це використання наукових знань про людину (психології, фізіології, медицини) з метою поліпшення умов роботи на робочому місці. Ергономіка в цілому характеризується двома принципами:

- Комфорт в процесі використання, який виражається у зменшенні фізичної та психологічної втоми;
- Безпека, яка передбачає вибір відповідних рішень для захисту користувача.

Із критеріїв ергономічності сайтів можна виділити наступні:

- Очікування користувачів: шукана інформація та вимоги до графічного оформлення варіюються для кожного користувача
- Звички, тобто придбана поведінка;
- Вік, в цілому характеризує здатність адаптуватися до нового та швидкість реакції користувача при перегляді;
- Обладнання - одна з головних перешкод. Відображення сайту може відрізнятися від одного пристрою до іншого, зокрема в залежності від браузера і дозволу екрану;
- Рівень знань: не всі відвідувачі є фахівцями в інформаційних технологіях. Ергономіка веб-сайту повинна бути розрахована на найменш досвідченого користувача.

Психологічні дослідження, проведені на людях показали деякі здібності, а також деякі обмеження. Мета ергономіки - скористатися цими психологічними елементами при створенні сайту для того, щоб реалізувати інтерфейс, який є ефективним і зручним для користувача.

Однією з найголовніших характеристик ергономічності є комфортність та практичність у використанні, тобто юзабіліті.

Важливість юзабіліті полягає в тому, що користувач виділяє лише обмежений проміжок часу на ознайомлення з сайтом, зрозуміння його структури та отримання бажаної інформації, виконання поставлених задач. Оскільки не існує очевидної інструкцій користування інтернет-сайтами, то дизайн повинен розроблятися так, щоб споживач міг на інтуїтивному рівні сприйняти спісоб використання. Якщо відвідувач не може швидко знайти інформацію, то в більшості випадків він прийме рішення його покинути і перейти на інший, комфортніший ресурс. Таким чином, юзабіліті, призначене, для оцінки зручність сайту, в результаті впливає і на такі показники, як конверсія і відвідуваність сайту.

Найчастіше користувачі просто переглядають, а не читають уважно те, що запропонував їм дизайнер. На думку Стіва Круга головна причина в тому, що користувачам байдуже. "Якщо ми знайдемо щось, що працює, ми будемо цим користуватися. Нам не цікаво, як це все працює, головне щоб працювало правильно. Якщо ваша аудиторія сприймає зроблене вами як дошку оголошень, тоді створюйте якісний дизайн дошки оголошень. "[3]

Також Круг виділив наступні принципи, або закони юзібіліті:

1. «Не змушуйте мене думати!» - Відповідно до першого закону юзабіліті Круга, веб-сторінка повинна бути очевидною. Головне завдання при створенні дизайну - позбутися питань - всі рішення користувачі повинні приймати свідомо, враховуючи позитивні та негативні ефекти, а токож можливі альтернативи.

Якщо навігація і структура сайту не зрозумілі на рівні інтуїції, кількість виникаючих питань зростає і зрозумілість сайту для користувачів значно зменшується. Зрозуміла структура, правильні візуальні підказки і посилання, що легко розпізнаються, значно допомагають користувачам знаходити шлях до шуканої інформації чи сервісу.

2. «Не має значення, скільки разів я повинен натиснути, поки кожне натиснення не потребує роздумів та обирається однозначно». Другий закон юзабіліті Круга каже, що кількість натиснень кнопок та дій на сторінці може бути

скільки завгодно великим, якщо кожна дія очевидна для користувача. Щоправда, це протирічить «Правилу трьох натисків» введеному та описаному Джеффрі Зельдманом. Згідно з його працями «правило трьох кліків засноване на тому, яким чином люди використовують Інтернет і це правило допоможе створювати сайти з інтуїтивної структурою з логічною ієрархією».[4]

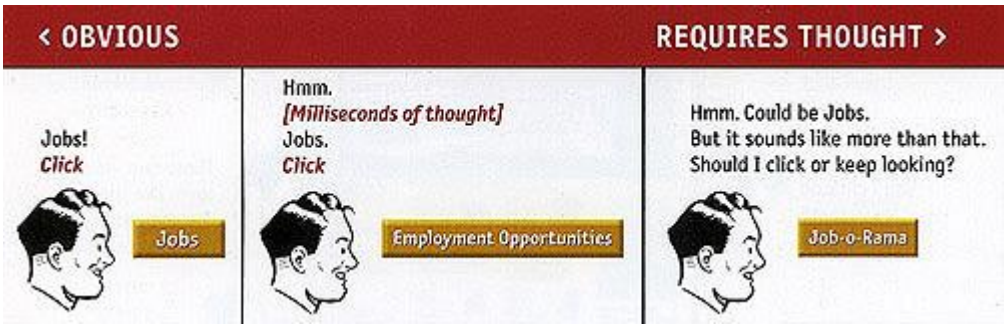

Рисунок 1 [5]

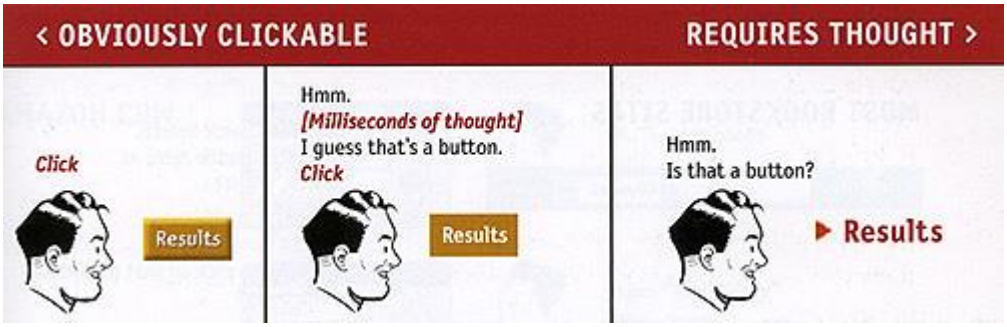

Рисунок 2 [5]

Приклади кнопок на Рис. 1-2 чітко ілюструють даний принцип. На Рис.1 ліва кнопка має очевидну і зрозумілу назву, яка не займає у користувача час на роздуми. Середня кнопка несе ту саму функцію, але визначення цієї функції потребує мінімального часу на обдумування. Третя кнопка лише віддалено описує необхідну функцію, що значно погіршує досвід взаємодії з користувачем. Так само, ліва частина Рис.2 містить кнопку, яка не викликає у користувача сумнівів, щодо її функціональності. Середня ж і права кнопки не очевидні для звичайного користувача. Середня і права кнопки змушують переконатися у своїй функціональності, а їх використання скоріш за все призведе до зменшення аудиторії та втрати довіри.

3. «Позбавтесь половини слів на кожній сторінці, потім приберіть ще половину». – Третій закон юзібіліті Круга тісно пов’язаний з першими двома за своєю метою – позбавити користувача зайвих роздумів. Читання великих об’ємів тексту займає великий проміжок часу та, як наслідок, зменшує зацікавленість у сторінці.

Також варто додати, що даний принцип прямо пов’язаний з тим, як людина сприймає інформацію на веб сторінці.

Рисунок 3 - Результати аналізу Якоба Нільсена

У 2006 році дослідник Якоб Нільсен виявив, що люди переглядають веб-сторінки за певною схемою. Відвідувачі сайту сканують інформацію на сторінці в лічені секунди. Якщо позначити лініями послідовність перегляду станиці, то отримаємо F-подібну форму, тому що погляд користувача зазвичай ковзає зліва направо в напрямку до нижньої частини сторінки. Схема, отримана Нільсеном , зображена на Рис. 3.[5] Варто зазначити, що користувачі звертають додаткову увагу на так звані «якорі», тобто помітні елементи сторінки, що може бути використано для привернення уваги.

Серед інших засобів покращення ергономічності сторінки можна виділити:

- Зменшення часу очікування для користувачів. Чим менше дій потрібно виконати користувачу для того, щоб спробувати наданий сервіс, тим більша ймовірність що випадковий відвідувач насправді спробує його в дії.

Відвідувачі, які потрапили на сайт вперше, бажають спробувати сервіс, а не заповнювати довгі форми для створення облікового запису, який вони може і зовсім не будуть використовувати. Необхідно надати користувачам можливість переглядати сторінки сайту і спробувати використовувати сервісом без необхідності отримувати від них особистих даних.

- Концентрація уваги користувачів. Якщо веб-сайт містить як статичний, так і динамічний контент, деякі аспекти призначеного для користувача інтерфейсу привертають більше уваги, ніж інші. Цілком очевидно, що зображення помітніші, ніж текст - так само, як і пропозиції, виділені жирним шрифтом більш помітні, ніж звичайні.

Людське око - складний нелінійний пристрій і веб-користувачі можуть миттєво визначати межі, шаблони і можливості руху. Тому реклама заснована на відео або така, що містить рухомі об'єкти, - неймовірно дратує і відволікає, але з боку маркетингу вона виконує свою функцію - звертає на себе увагу користувача.

- Підтримка у полі зору важливих елементів структури та дизайну. Дати користувачам зрозуміти, які функції їм доступні - фундаментальний принцип успішного дизайну. Не має значення, яким саме способом це досягається. Важливим є те, наскільки легко сприймається контент і наскільки користувачам легко працювати з системою.

- Підтримка зрозумілості текстів. Веб відрізняється від друкованих видань тим; що існує необхідність пристосовувати стиль написання текстів до вимог та очікувань користувачів і способам перегляду через браузери. Відвідувачі не читатимуть рекламні тексти. Великі абзаци без зображень та ілюстрацій, без виділених жирним або курсивом елементів, частіше за все будуть пропускатися, а перебільшення ігноруватимуться.

Необхідно чітко доносити сутність та ідею тексту. Уникайте дотепних, специфічних назв, які використовуються всередині компанії, незнайомих технічних термінів. Наприклад, якщо необхідно описати сервіс і

необхідно, щоб користувачі зареєструвалися, то "зареєструватися" краще, ніж "почніть зараз!", що, в свою чергу краще, ніж "випробуйте наші сервіси".[6]

1.1.2 Особливості використання кольорів на веб-сторінці

Колір - один з найважливіших інструментів маніпуляції людьми в Інтернеті, важливість якого часто недооцінюється. Причина цьому – користувачі зазвичай не помічають кольори на сторінці, вони залишаються на тлі. Користувачі оцінюють сайт в цілому. Але саме за допомогою кольору передаються емоції, почуття, створюється атмосфера сайту. Правильно володіючи цим інструментом, можна як схвилювати користувача і стимулювати його до дії, такої, як придбання товару, так і створити спокійну і довірливу атмосферу на сайті.

Колір для сайту - це не просто прикраса, колір змінює сприйняття людини і є далеко не останнім чинником в результативності сервісу. Це пояснюється зі сторони чистої психології, це робота зі свідомістю і підсвідомістю.

На сьогодні вважається, що несумісних кольорів та відтінків не існує. Вибір кольорової гами повністю залежить від мети, поставленої дизайнеру, та головної ідеї сайту.

Задача підбору кольорової гами значно спрощується з використання кольорового кола, які зображено на Рис. 4-6.

На колірному колі є сім основних кольорів: червоний, оранжевий, жовтий, зелений, блакитний, синій і фіолетовий (кольори веселки). Решта ж кольорів отримується шляхом змішування, такі кольори називаються додатковими або другорядними, найкраще поєднувати 2-3 основні кольори і другорядні відтінки.

В теорії кольору виділяють наступні основні поняття поєднання кольорів:

Контрастність - це протилежність кольорів в колірному колі. У кожного кольору є протилежний колір, який з ним найбільш контрастує. Для того, щоб його знайти, необхідно вибрати колір, а контраст йому буде створювати той, який знаходиться на протилежному боці кола.

Такі кольори можна вважати додатковими, тому як на сайті вони матимуть рівні значення. У більшості випадків їх застосовують для того щоб акцентувати увагу на будь-якому елементі.

Аналогічні кольори - це кольори, які добре виглядають разом, вони доповнюють і підкреслюють один одного, створюючи відчуття затишку і комфорту. Знайти їх просто: необхідно обрати колір і взяти колір поруч із ним в колірному колі, саме він і буде аналогічним.

Теплі і холодні кольори - це розмежування кольорів за настроєм і відчуттям.

Підбір кольору – це перший етап створення кольорової гами сайту. Також необхідно, щоб він гармоніював з іншими, створюючи цілісну картину, в якій, кожен колір має своє значення і перебуває на своєму місці.

Від правильного вибору колірної схеми залежить багато характеристик сайту: наприклад, рівень відвідуваності, продажу на сайті. Це пояснюється тим, що людині повинно бути комфортно, затишно і зручно на сторінці.

В сучасному дизайні існує багато способів поєднувати кольори, але надалі буде розглянуто лише ті, що використовуються найчастіше і придбали найбільшу популярність у колі дизайнерів.

Тріада (триколірна) - це колірна схема, на якій всі кольори всередині колірного кола утворюють трикутник. Вона є найбільш збалансованою і надійною схемою. Друга за кількістю кольорів у палітрі, вона викликає почуття інтересу, кольори переливаються подібно веселці.

Для того щоб отримати тріаду, потрібно вибрати 3 різних кольори розташованих в 120 градусах один від одного. Приклад тріадної колірної схеми рображено на Рис. 4.

Рисунок 4 - Тріадна колірна схема

В основі подвійної комплементарної схеми (Рис. 5) лежать 4 кольори, які контрастують хрест на хрест один з одним, утворюючи прямокутник. Палітра насичена кольорами і за рахунок цього може бути виграшною: створює відчуття радості і щастя. Чотири обрані кольори створюють на колірному колі прямокутник, саме тому таку схему часто називають прямокутною.

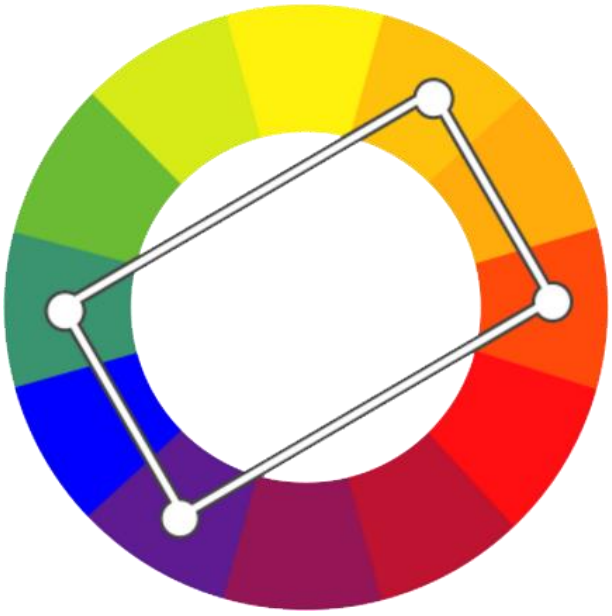

Рисунок 5 - Прямокутна колірна схема

Рисунок 6 - Послідовна колірна схема

Послідовна схема поєднує в собі кольори, які можна назвати аналогічними. Послідовне розташування кольорів дає відчуття комфорту і заспокоєння.

Для того щоб підібрати кольори для цієї схеми, необхідно обрати кольори, які в колірному колі розташовуються один за одним.

1.2 Обгрунтування вибору інструментальних засобів

В ході вибору інструментальних засобів використовувались наступні критерії: швидкість генерації веб-сторінок фреймворком, наявність стандартних засобів з поєднання веб-сторінки з веб сервісом розрахунків, інтеграція із середовищем розробки, необхідність додаткових засобів розробки.

Був визначений наступний список можливих інструментальних засобів:

- ASP.Net Core
- Django
- Ruby on rails
- Spring MVC
- Yii

Серед наведених варіантів був обраний ASP.Net Core додаток. Його переваги та недолыки наведены нижче.

Фреймворк Django не має очевидних недоліків – він має високу продуктивність та широко використовується для розробки сайтів. Його головним недоліком є обмеженість можливостей впровадження додатку на не Unix-подібні системи. Це може призвести до непередбачуваної поведінки та поломки веб додатку. ASP.Net, з іншого боку, не поступається за функціональними характеристиками та може працювати під будь-якою системою.

Ruby on rails, або просто Rails, - досисить популярний вибір для задач такого типу. Фреймворк кросплатформенний та простий у використання, бо використовує мову надвисокого рівня, легко поєднується з базами даних. Щоправда, високий рівень мови накладає деякі обмеження – Rails значно поступається іншим фреймворкам у продуктивності. Також, зазвичай, розробка під Rails займає набагато більше часових та людських ресурсів.

Spring MVC базується на мові Java, з чого можна зробити висновок про його високу пробуктивність роботи. З його допомогою можна розробляти декілька додатків за єдиним шаблоном: як веб-сервіс, так і окремий додаток. Це значно зменшує необхідний час розробки та підвищує якість продукту. Єдиним та головним недоліком Spring MVC є громіздкість у використанні. Для його коректної роботи необхідний менеджер залежностей, як приклад Maven. У випадку ASP. Net Core, такої необхідності не виникає, бо використовується єдине середовище розробки – Visual studio, чого немає у Django та Rails.

Yii – це фреймворк PHP, що використовує бази даних для генерації готового коду. Сразу ж варто додути, що це значно ускладнює процес розробки, тому що згенерований код також необхідно перевірити та відредагувати. Також, Yii наслідує йсу існуючі недоліки PHP, що робить його не бажаним у використанні.

Отже, як інструментальний засіб для розробки сайту було обрано ASP.Net Core MVC.

1.2.1 ASP.Net Core як засіб розробки веб-додатків

ASP.NET Core - це істотне переопрацювання ASP.NET. ASP.Net Core має декілька нових концепцій, які будуть описані нижче.

ASP.NET Core являє собою новий фреймворк з відкритим вихідним кодом для побудови сучасних крос-платформенних хмарних додатків, наприклад, веб-додатків, додатків Інтернет Речей і мобільних серверів. ASP.NET Core програми можуть працювати на .NET Core або на повній версії .NET Framework. Він був спроектований, щоб забезпечити оптимальне середовище розробки для додатків, розгорнутих в хмарі або працюючих на спеціальних виділених серверах (on-premise). Він складається з модульних компонентів з мінімальними витратами, тому зберігає гнучкість при побудові необхідних рішень. Ви можете розробити і запустити ASP.NET Core програми крос-платформенно: на Windows, Mac і Linux. ASP.NET Core має відкритий вихідним код на GitHub.

Перша поява ASP.Net відбулася майже 15 років тому як частина .NET Framework. З тих пір мільйони розробників використовували його для створення і впровадження великих веб-додатків, і з того часу було дадано та покращено безліч можливостей.

ASP.NET Core має цілий ряд архітектурних змін, які зробили фреймворк набагато компактнішим, зробили модульну структуру меншою та компактнішою.

Головна відмінність ASP.NET Core (ASP.Net 5.0) від попередньої версії - фреймворк більше не заснований на System.Web.dll. Він заснований на безлічі гранульованих і добре поділений пакетів NuGet. Це дозволяє оптимізувати додаток, щоб включити лише ті пакети NuGet, які необхідні для функціонування додатку. Переваги меншої кількості залежностей програми включають в себе жорсткіший контроль системи безпеки, зменшення потреби у обслуговуванні, підвищення продуктивності і зниження витрату моделі плата-за-те-що-використовується.

Серед ключових характеристик ASP.Net Core варто виділити:

- Єдина платформа для створення веб-інтерфейсів та веб-сервісів;
- Інтеграція сучасних фреймворків для розробки клієнтськох частини програм із сучасними методологіями розробки;
- Система конфігурації середовища, зорієнтована на підтримку хмарних технологій;
- Вбудовані ін'єкції залежностей;
- Новий легкий і модульний провідник запитів HTTP
- Можливість розміщення на IIS або на окремому сервері чи процесі;
- Підтримує справжнього «сусіднього» управління версіями;
- Повністю надається та підтримується у вигляді набору NuGet пакетів;
- Нові інструменти розробки, які значно спрощують сучасну веб-розробку;
- Можливість побудови та запуску крос-платформних додатків ASP.NET на Windows, Mac і Linux
- Наявність відкритого вихідного коду та орієнтація на підтримку спільноти розробників.[7]

ASP.Net Core надає можливість створювати HTTP-сервіси, які мають широке коло клієнтів, в тому числі браузері і мобільні пристрої. Також надається вбудована підтримка декількох форматів даних та домовленостей щодо вмісту.

Також, варто додати, що додатки ASP.Net Core побудовані на MVC паттерні (Model-View-Controller). Це значно змінює хід розробки та тестування, надає можливість використання модульних тестів.

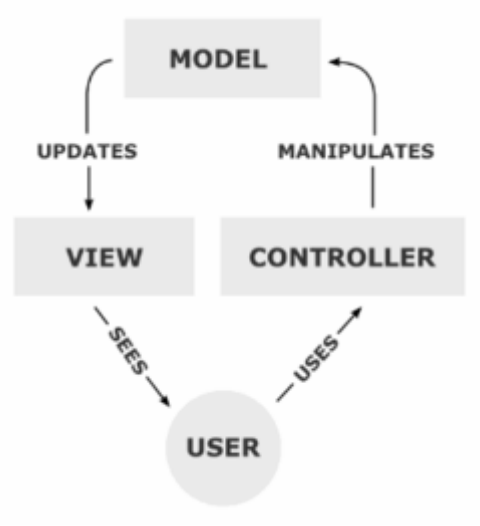

Рисунок 7 - Структура паттерну MVC

Одним з характерних моментів платформи ASP.NET Core є застосування патерну MVC. Причому версія MVC-фреймворка, який застосовується в ASP.NET Core, має номер 1.0.

Сам патерн MVC не є новою ідеєю в архітектурі додатків, він з'явився ще в кінці 1970-х років в компанії Xerox як спосіб організації компонентів в графічному додаток на мові Smalltalk.

Концепція паттерна MVC передбачає поділ додатка на три компоненти:

Модель (model): описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як правило, об'єкти моделей зберігаються в базі даних.

У MVC моделі представлені двома основними типами: моделі представлень, які використовуються представлення для відображення і передачі даних, і моделі домену, які описують логіку управління даними.

Модель може містити дані, зберігати логіку управління цими даними. У той же час модель не повинна містити логіку взаємодії з користувачем і не має визначати механізм обробки запиту. Крім того, модель не повинна містити логіку відображення даних в поданні.

Представлення (view): відповідають за візуальну частину або призначений для користувача інтерфейс, нерідко - це html-сторінка, через яку користувач

взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час уявлення не повинно містити логіку обробки запиту користувача або управління даними.

Контролер (controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує введені користувачем дані і обробляє їх. В залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей.

Рисунок 8 - Відношення між елементами паттерну MVC

У такій схемі модель є незалежним компонентом - будь-які зміни контролера або подання ніяк не впливають на модель. Контролер і представлення - відносно незалежні компоненти. Так, з представлення можна звертатися до певного контролеру, а з контролера генерувати представлення, але при цьому нерідко їх можна змінювати незалежно один від одного.

Таке розмежування компонентів додатка дозволяє реалізувати концепцію розділення відповідальності, при якій кожен компонент відповідає за свою строго окреслену область. Це значно полегшує побудову структури роботи окремих компонентів. І завдяки цьому додаток легше розробляти, підтримувати і тестувати окремі компоненти. Припустимо, якщо для розробки важлива візуальна частина або фронтенд, то ми можемо тестувати уявлення незалежно від контролера. Або ми можемо зосередитися на бекенді і тестувати контролер.

Однак невірно повністю асоціювати всю платформу ASP.NET Core з MVC. MVC - це лише патерн, який реалізується в рамках платформи. Існує можливість створити проект за шаблоном Empty, де не буде ніяких контролерів, моделей, уявлень, де буде наявний лише один клас Startup. І через цей клас побудувати всю обробку запиту. Але природно застосування MVC полегшує розробку додатків.

Модель у таких додатках будується на базі Entity framework. Таким чином, зникає потреба у додатковій розробці бази даних для веб-додатку.

Модель прив'язки автоматично передає дані з HTTP-запитів до безпосередніх дій сайту.

Модель валідації підтримує автоматичну перевірку даних на стороні як клієнта, так і сервера.

ASP.NET Core призначений для інтеграції з популярними фреймворками для розробки інтерфейсів користувача, в тому числі AngularJS, KnockoutJS і Bootstrap. [8]

1.3 Побудова алгоритму взаємодії сервісу та сайту

1.3.1 Протокол SOAP як засіб створення взаємодії

Взаємодія веб-сторінки із веб-сервісом розрахунків здійснюватиметься за допомогою протоколу SOAP.

SOAP (від англ. Simple Object Access Protocol) - простий протокол доступу до об'єктів) - це протокол обміну структурованими повідомленнями в розподіленому обчислювальному середовищі. Спочатку SOAP був призначений в основному для реалізації віддаленого виклику процедур (RPC – Remote procedure call). Зараз протокол використовується для обміну довільними повідомленнями в форматі XML, а не тільки для виклику процедур. Офіційна специфікація останньої версії 1.2 протоколу Ніяк не розшифровує назву SOAP. SOAP розширює протокол XML-RPC.

Веб-сервіс за такої взаємодії, надає можливість спілкування з сервером чітко структурованими повідомленнями. Це пояснюється тим, що, що веб-сервіс створює певні вимоги до даних, які він може обробляти.. На будь-яке повідомлення, яке не відповідає правилам, веб-сервіс відповість помилкою. Варто зазначити, що помилка також буде передана у вигляді xml з чіткою структурою.[9]

Правила, за якими складаються повідомлення для веб-сервісу описуються так само за допомогою xml і також мають чітку структуру. Для такого опису використовується WSDL – Web service description language – мова опису веб сервісів. Тобто якщо веб-сервіс надає можливість виклику якогось методу, він повинен дати можливість клієнтам дізнатися які параметри для даного методу використовуються. Якщо веб-сервіс очікує отримати рядок для методу Method1 як параметр і рядок повинен мати ім'я Param1, то в описі веб-сервісу ці правила будуть вказані.[10]

Як параметри можуть передаватися не тільки прості типи, а й об'єкти, колекції об'єктів. Опис об'єкта зводиться до опису кожної складової об'єкта. Якщо об'єкт складається з декількох полів, це означає, що описуватися має кожне поле, його тип і назва. Поля також можуть бути складними об'єктами і потребуватимуть додаткового опису. Цей процес на закінчується поки об'єкти не будуть розкладені на примітивні типи даних: рядок, булеве значення, число, дата тощо. Втім простими можуть бути й інші, специфічні типи даних. Важливо, щоб клієнти-користувачі сервісу могли зрозуміти які значення можуть в них міститися.

Для ініціації взаємодії з сервісом, клієнту необхідно лише знати адресу (URL). За цією адресою міститиметься wsdl-файл з описом доступних методів та їх параметрів.

Серед переваг такого підходу можна виділити наступні:

- У більшості систем опис методів і типів даних відбувається в автоматичному режимі. Тобто розробнику сервісу необхідно лише вказати, що

даний метод можна викликати через веб-сервіс, і wsdl-опис буде згенеровано автоматично;

- Опис, має чітку структуру, обробляється будь-яким soap-клієнтом. Тобто який би не був веб-сервіс, клієнт зрозуміє які дані веб-сервіс приймає. З цього опису клієнт може побудувати свою внутрішню структуру класів об'єктів;
- Вміст запитів проходит автоматичну валідацію.
  - Xml-валідація – xml-файл повинен бути well-formed. Невалідні XML-документи не будуть оброблятися та призведуть до відправлення помилки у відповідь;
  - Schema-валідація - xml повинен мати певну структуру. Знову, невалідні XML-документи призведуть до генерації помилки;
- Перевірка відповідності типів даних, що передаються, до типів даних, що очікуються, виконується Soap-сервером;.
- Веб-сервіси можуть працювати як використовуючи soap-протокол, так і використовуючи http, тобто через get-запити. Тобто, якщо в якості параметрів використовуютсья прості дані (без структури), то можна викликати звичайны get-запити до веб-сервісу та передавати параметри, як атрибути запиту.

Серед недоліків можні виділити:

- Невиправдано великий розмір повідомлень. Це зумовлено збітковою структурою формату XML – зі збільшенням кількості тегів, зростає кількість надлишкової інформації.
- Автоматична зміна опису веб-сервісу може призвести до виходу з ладу усіх клієнтів.

1.3.2 Опис алгоритму взаємодії веб-сторінки із веб-сервісом

Веб-додаток розробляється за клієнт-серверною архітектурою, тобто частини додатку, що відповідають за користувацький інтерфейс та безпосереднь сервіс розрахунків функціональна та архітектурно незалежні один від одного.

В такому випадку необхідно додатково розглянути алгоритм взаємодії сервісу із веб сторінкою. Діаграма послідовності наведена на рис. 9.

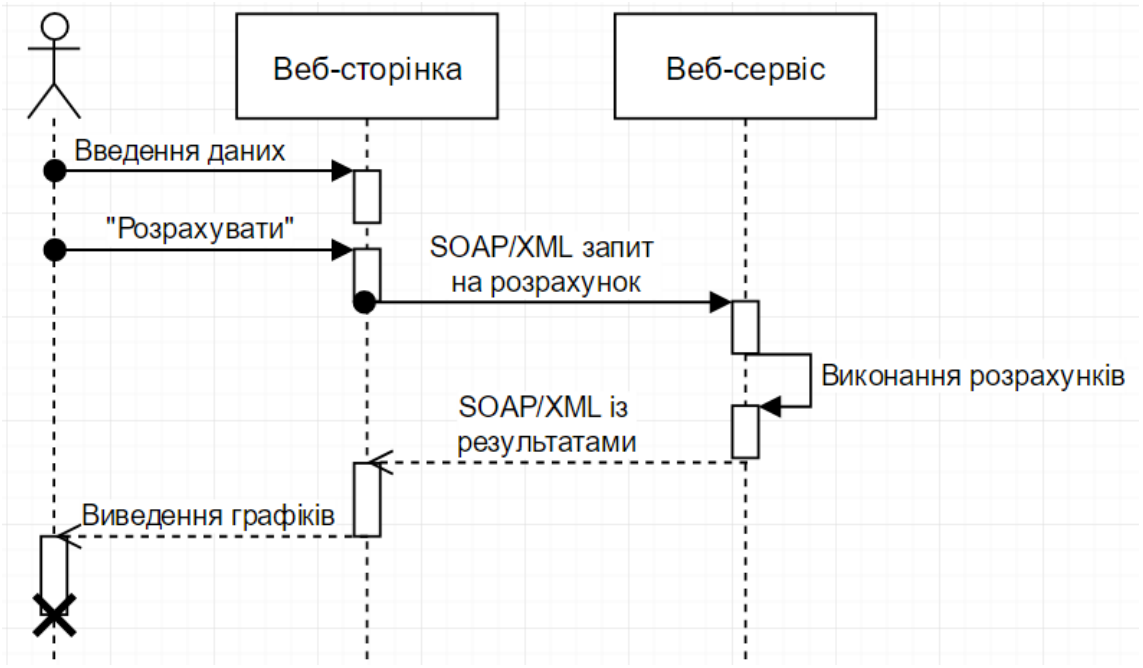

Рисунок 9 - Діаграма послідовності алгоритму взаємодії сервісу та веб-сторінки

Розглянемо алгоритм взаємодії більш детально. Користувач, відкривши сторінку, заповнює необхідні поля. Після введення, щоб отримати результати, необхідно натиснути відповідну кнопку – «Розрахувати». Це призведе до створення SOAP/XML запиту до веб-сервісу розрахунків. Створений SOAP-документ міститиме усі параметри, введені користувачем, а також ідентифікатори метода сервісів, які потрібно викликати. Переданий SOAP-документ буде оброблений на стороні сервісу. Якщо файл відповідає вимогам сервісу та не був пошкоджений у процесі передачі, буде викликано відповідний метод сервісу.

Роботу самого сервісу сприймаємо як «чорний ящик» та не розлядаємо.

Отримані результати веб-сервіс розрахунків серіалізує у SOAP-відповідь, яка й повернеться до веб-сторінки. Далі, з отраманих результатів буде побудовано відповідну частину html-документу, яка й буде представлена користувачу.

1.4 Висновок до розділу

У даному розділі було розглянуто особливості створення веб-сайтів у сучасних веб технологіях, було визначено головні проблеми, які виникають при розробці веб-сторінок, було визначено головні вимоги до побудови ергономічної веб сторінки, вимоги до ергономіки.

Також було обрано технології, які будуть використовуватися для побудови веб сторінки. Вибір був здійснений порівнянням існуючих технологій, які використовуються для задач такого типу.

На основі цієї інформації в наступному розділі були здійснені конкретні рішення при побудові макету сторінки та практичної реалізації.

2 ПРАКТИЧНА РЕАЛІЗАЦІЯ САЙТУ РОЗРАХУНКУ ЦИФРОВИХ ФІЛЬТРІВ

2.1 Розробка макету веб-сторінки

Обов’язковим кроком для створення будь-якої веб-сторінки є створення макету сайту.

Макет сайту - це варіант дизайну майбутнього сайту. Дизайн визначає зовнішній вигляд - те, на що в першу чергу звертає увагу користувач. Від дизайну багато в чому залежить загальне сприйняття інформації, яку містить сайт.

В процесі створення макету сайту обирається колірна гамма, розташування меню, шрифт, навігація і багато інших елементів сторінки. Дизайн визначає не тільки форму подачі інформації, а й інтерфейс користувача. Тому дизайн сайту повинен нести не тільки естетичну функцію, але практичну.

Створений макет задовольняє вимогам ергономіки та юзабіліті. Колірна гамма підібрана з урахуванням психологічних особливостей сприйняття кольорів людиною.

Макет сайту, розроблений в ході роботи, наведено на рис. 10.

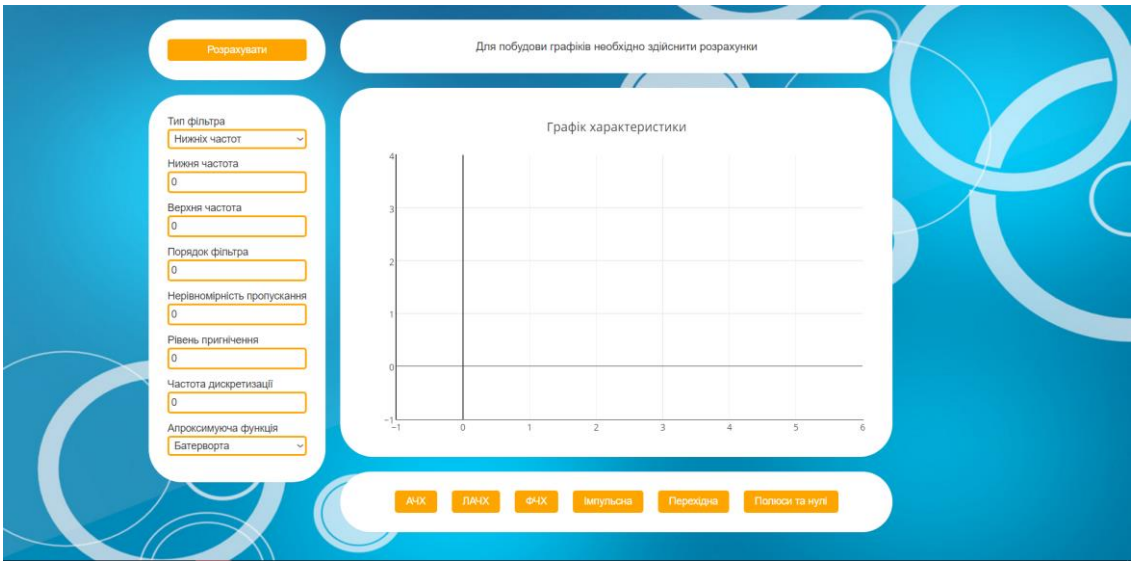

Рисунок 100 - Розроблений макет сайту

Вміст сторінки поділений на п’ять частин за функціональним призначенням. Кожна така частина відділена від інших білим тлом, яке надалі називатиметься панеллю. Розглянемо кожну з них окремо:

1. Ліва верхня панель містить єдину кнопку «Розрахувати». Її головна задача – викликати перерахунок передавальної характеристики і графіків характеристик фільтру. При успішному розрахунку, користувач побачить амплітудно-частотну характеристику фільтру та передавальну характеристику. Якщо при розрахунках виникли помилки, то користувач побачить замість передавальної характеристики, повідомлення з помилкою.

2. Ліва нижня панель, містить список полів, які відображають можливі вимоги до цифрових фільтрів: величини нижньої та верхньої частоти, міри допустимих нерівномірностей пропускання, рівень пригнічення поза межею полоси пропускання, порядок фільтру, частоту дискретизації, тип апроксимуючої функції.

3. Права верхня панель міститиме інформативні повідомлення для користувача. У разі успішного розрахунку характеристик фільтру, ця панель міститиме передавальну характеристику. Якщо ж при розрахунках виникли помилки, або введені дані не валідні для розрахунків, панель міститиме повідомлення для користувача.

4. Права середня панель містить графік бажаної характеристики. Побудований графік представлений у вигляді графіку функції, отриманої у результаті роботи сервісу розрахунків.

5. Права нижня панель містить кнопки, що відповідають за переключення характеристик, що зображаються на графіках. На графіку може бути зображена одна з п’яти отриманих характеристик: амплітудно-частотна, фазо-частотна, логарифмована амплітудно-частотна характеристика, імпульсна та перехідна характеристики, а також мапа полюсів та нулів передаточної функції.

Такий поділ елементів дозволяє зосередити увагу користувачів на необхідних елементах. Якщо взяти до уваги результати досліджень Якоба

Нільсена [5], то можна легко переконатися в тому, що увагу користувачів приверне кнопка «Розрахувати» та панель з полями для вводу, що й необхідно для такого сайту. Поле з повідомленням, яке знаходиться вгорі сторінки, очевидне і зрозуміле, а, як наслідок, дасть користувачу зрозуміти, що необхідно зробити, щоб отримати бажану характеристику.

Кнопки перемикання графіків не привертають зайву увагу, так як функціонально необхідні лише після виконання розрахунків.

Обрана кольорова гамма побудована на контрасті двох кольорів – синього та помаранчевого. При чому синій використовується як основний колір, а помаранчевий – як колір акцентів. Білий колір в сучасному веб-дизайні не вважається кольором гамми, тому можна зробити висновок, що використано діадну колірну схему.

Такий вибір колірної схеми не перевантажує сторінку кольорами, що попереджає втому користувачів, та дозволяє підкреслити акцентовані елементи.

З побудованого макету можна визначити, що сайт, отриманий в результаті, буде складатися лише з однієї, головної сторінки. Усі оновлення елементів відбуватимуться без переходів між сторінками та перезавантаженнями.

Для побудови макету сайту було використано Adobe Photoshop CC 2017. Такий засіб створення макетів не є типовим, чи звичайним. Його використання можна пояснити тим, що створення базових елементів дизайну в ньому здійснюється набагато швидше, ніж у більш професійних аналогах – InDesign, QuarkXPress, або Illustrator.

2.2 Розробка веб-сайту

Розробка прототипу веб-сторінки проводилася у інтегрованому середовищі розробки – Visual studio 2017 Community.

Веб-сторінка побудована за паттерном MVC, а тому поділена на три частини: візуальне представлення, контроллери поведінки та модель.

Власне, створений сайт можна поділити на наступні модулі:

- Головний контроллер – FilterModelsController.cs. Даний клас відповідає за загальне функціонування сайту – саме в ньому виконується обробка запитів користувача та викликається перебудова представлень;
- Допоміжний клас для побудови HTML-розмітки – HtmlBuildHelper.cs. Головна функція цього класу – побудова розмітки для кнопок на головній сторінці;
- Клас-будівник математичних формул – MathMLBuilder.cs. Цей клас виконує функцію відмалювання математичних формул на сторінці, що є досить не тривіально задачею у вебі;
- Модуль валідації – ValidationController.cs. Даний клас валідує створену користувачем модель – перевіряє введені значення на коректність;
- Модуль запитів до сервісу. Це центральний модуль проекту, адже саме він пов’язує створений користувацький інтерфейс із веб-сервісом розрахунків.

Розглянемо кожен із модулів більш детально.

2.2.1 Загальна архітектура сайту розрахунку цифрових фільтрів

За побудову HTML-розмітки сторінки, у сайті відповідає єдиний контроллер: FilterModelsController.cs. Він містить в собі 12 методів, що відповідають усім можливим діям користувача. Ці 12 методів, або Дій (Actions у контексті ASP.Net Core) так чи інакше викликають побудову HTML-розмітки усієї сторінки, чи лише її частини.

Для вирішення задачі створення сайту розрахунку цифрових фільтрів, в процесі розробки було створено п’ять представлень: 1 повне і 4 часткових.

Зв’язок Дій контроллеру із представленнями зображено на Рис. 11.

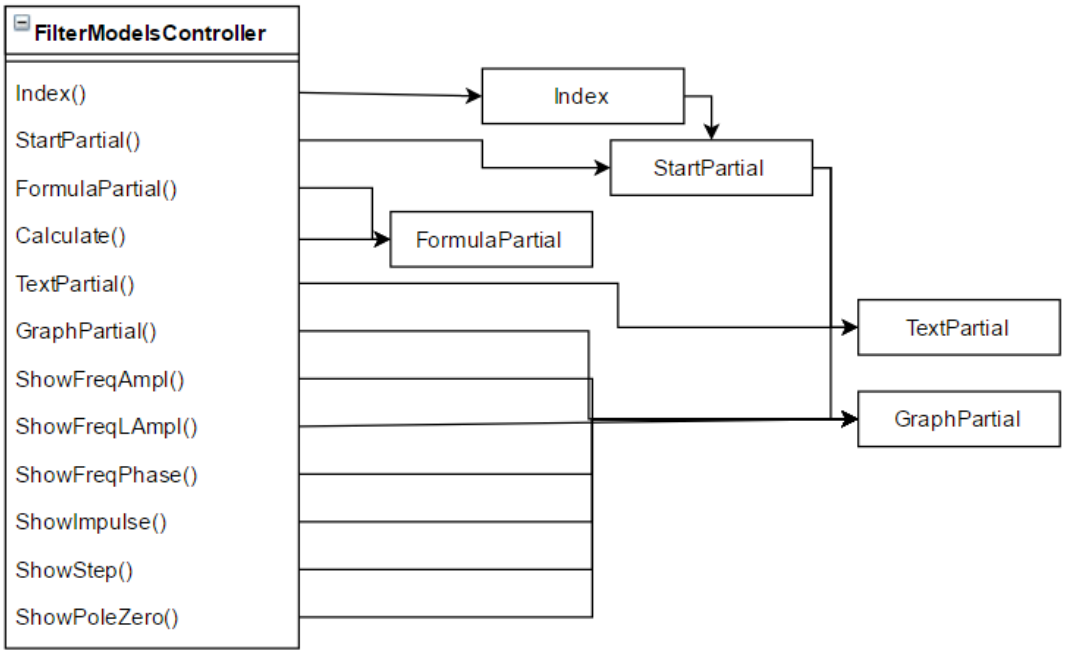

Рисунок 11 - Зв'язок Дій контроллеру із представленнями

Часткове представлення – це представлення, яке вимальовується всередині іншого представлення. Результуючий HTML-файл генерується вставленням часткового представлення у викликаюче представлення. Як і звичайні представлення, часткові файли з розширенням .cshtml.

Часткові представлення - ефективний спосіб розбити великі представлення на дрібніші компоненти. Вони можуть зменшити дублювання контенту представлення і дозволяють використовувати у декількох представленнях схожих елементів.

В нашому випадку, до часткових представлень було віднесено елементи, що потребували постійного перемалювання: панель із графіком (GraphPartial), панель із формулою (FormulaPartial), панель із текстом (TextPartial) та панель, що відмальовує одночасно і графік, і текст (StartPartial).

Усі ці представлення використовують чітко визначену модель – FilterModel. Ця модель не прив’язана до бази даних та використовується лише для передачі інформації про параметри цифрового фільтру, що розраховується, між контроллером та представленнями.

Представлення Index містить у собі усі статичні елементи сторінки, тобто: кнопку «Розрахувати», поля для введення, кнопки перемикання графіків, а також виклик для часткового зображення StartPartial. Таким чином Index – це тіло сайту.

Окрім головних структурних елементів це представлення містить ще й елементи коду, що відповідають за прив’язання полів для вводу до використаної моделі. Таке приєднання здійснюється з використанням javascript-фреймворку AngularJS. При першій побудові сторінки на стороні клієнта, модель FilterModel конвертується у формат json, який легко і зручно поєднується із моделлю AngularJS.

Кожне поле для вводу приєднується до моделі з двох боків: значення моделі FilterModel переноситься до полів введення при перебудові сторінки, а будь-яка ручна зміна значень поля викликає оновлення значень у моделі AngularJS.

Для виклику Дій контроллеру використовуються AJAX запити. Практично, натиснення кнопки «Розрахувати», або кнопок зміни графіка викликає POST запит до контроллеру. Разом із цим запитом надсилається і поточний стан моделі, тобто модель, що зберігається у AngularJS конвертується назад, у формат зрозумілий серверу ASP.Net Core. Результатом такого запиту є HTML-розмітка результуючого часткового відображення. У випадку перемикання графіка – це лише панель із графіком. Якщо ж натиснено кнопку «Розрахувати», то буде виконано дві послідовні дії – виклик Дії Calculate контроллера, яка повертає панель формули у разі успіху, чи панель тексту, у випадку невдачі. Потім викликається натиснення на кнопку «АЧХ», що викликає побудову графіку амплітудно-частотної характеристики.

Варто додати, що для полів введення існують обмеження по заповненню. Вони викликані тим, що для розрахунку характеристик цифрових фільтрів нема необхідності передавити усі можливі параметри.

Так, для фільтрів нижніх та верхніх частот нема необхідності передавати частоти другої перехідної полоси; якщо зазначено порядок фільтру, то для

розрахунків не потрібне пригнічення у полосі пригнічення, і навпаки. Також, якщо заповнено порядок фільтру, то нема необхідності заповнювати частоти поруч із полосами пригнічення.

У всіх цих випадках поля для введення не будуть відображатися, а їх значення будуть стерті.

Поле «Порядок фільтру» повинне бути видиме лише у випадку, якщо поле «Рівень пригнічення» не заповнене:

$$showOrder = !rippleStopFilled,$$

де showOrder – поле «Порядок фільтру» видиме,

rippleStopFilled – поле «Рівень пригнічення заповнене»

Поле «Нижня частота» повинне бути видиме лише у випадках, коли порядок фільтру не заповнений, або розраховується фільтр нижніх частот або режекторний фільтр:

$$showLowFreq = !orderFilled || (orderFilled \;\&\&\; (lowFreqFilter \;||\; bandStopFilter)),$$

де orderFiller – поле «Порядок фільтру» заповнене,

lowFreqFilter – обрано тип фільтра «Нижніх частот»

bandStopFilter – обрано тип фільтра «Режекторний»

Поле «Верхня частота» повинно бути видимим у схожих випадках, що й поле «Нижня частота». Необхідно лише змінити очікувані типи фільтрів:

$$showHighFreq = !orderFilled || (orderFilled \;\&\&\; (highFreqFilter \;||\; bandPassFilter)),$$

де orderFiller – поле «Порядок фільтру» заповнене,

highfFreqFilter – обрано тип фільтра «Верхніх частот»

bandPassFilter – обрано тип фільтра «Полосовий»

Поля для другої перехідної полоси частот використовуються лише у випадку режекторного та полосового фільтрів, але вони також мають обмеження за використанням порядку:

$$showBandLowFreq = bandPassFilter||(bandStopFilter \;\&\&\; !\,orderFilled);$$
$$showBandHighFreq = bandStopFilter||(bandPassFilter \;\&\&\; !\,orderFilled)$$

де showBandLowFreq – поле «Нижня частота» (для другої полоси) видиме,
showBandHighFreq – поле «Верхня частота» (для другої полоси) видиме,
bandPassFilter – обрано тип фільтру «Полосовий»,
bandStopFilter – обрано тип фільтру «Режекторний»,
orderFilled – поле «Порядок фільтру» заповнене.

У представленні Index використовується створений допоміжний клас для побудови html-розмітки для кнопок. Цей модуль представлений класом HtmlBuildHelper. Його використання зумовлене необхідністю зменшити кількість створеної власноруч однотипної розмітки на сторінці.

Сама побудова HTML-елементів заснована на застосуванні стандартних засобів бібліотеки System.Xml.Linq, яка надає зручний інтерфейс для побудови XML-структури елементів за допомогою об’єктів XElement.

Наразі, клас містить лише один відкритий, публічний метод – побудову кнопки, але наявні й допоміжні методи, що дозволять легко розширити кількість доступних для побудови елементів.

Представлення FormulaPartial містить блок тегів <math> в середині яких міститься виклик модулю побудови математичних формул, а також містить посилання на часткове представлення GraphPartial. Це зроблено з метою спрощення процесу оновлення сторінки при перерахунку цифрового фільтру.

За такої реалізації, після отримання відповіді від сервісу обидві панелі будуть перемальовані в одному запиті до контроллера, що значно зменшує час очікування користувача.

Значення коефіцієнтів передавальної функції представленням отримуються зі статичного контейнера.

Представлення GraphPartial містить результуючі характеристики цифрового фільтру, отримані в результаті розрахунків.

Зазначимо, що Дій контроллера, які починаються з “Show*” лише змінюють значення поля моделі «Який графік виводити» на необхідне значення і перенаправлять подальші дії до GraphPartial

Для побудови графіка функції використовуються можливості javascript-фреймворку Plotly – сучасної платформи для побудови різноманітних схем та графіків.

Для побудови графіку на панелі, до її HTML-розмітки було додано порожній тег , який буде заповнений на стороні клієнта, та блок <script> який містить логіку заповнення.

```
var xaxis = [@Html.Raw(GraphsMap.GetAxixX(Model.GraphToShow, Model.SampleRate))];
var yaxis = [@Html.Raw(GraphsMap.GetAxixY(Model.GraphToShow))];

var plot = [{
       x: xaxis,
       y: yaxis,
       mode: viewMode
}];
data = plot;

Plotly.newPlot('graph', data, layout);
```

Рисунок 12 – Приклад побудови графіку використовуючи засобами Plotly

На Рис. 12 зображено приклад виклику побудови графіку засобами Plotly. Розглянемо його більше детально. Метод Plotly.newPlot перебудовує графік у вказаному блоці html за вказаними даними.

Множини значення для осей абсцис та ординат повинні бути представлені у вигляді масивів. Для їх заповнення використовується статичний контейнер, де вони і зберігаються.

В результаті, отримані значення для осей X та Y поєднуються у єдиний об’єкт графіку та передаються до Plotly.

Побудова мапи полюсів та нулів містить деякі особливості, адже отримані значення необхідно виводити у вигляді окремих точок. З цією метою використовується спеціальна властивість графіку у Plotly – viewmode. Завдяки її значенню ‘markers’ точки виводяться по одній, що й було необхідно. Також для мапи полюсів і нулів генерується одиничне коло, яке генерується програмно.

Варто зазначити, що імпульсна та перехідна характеристики також виводяться у вигляді окремих точок.

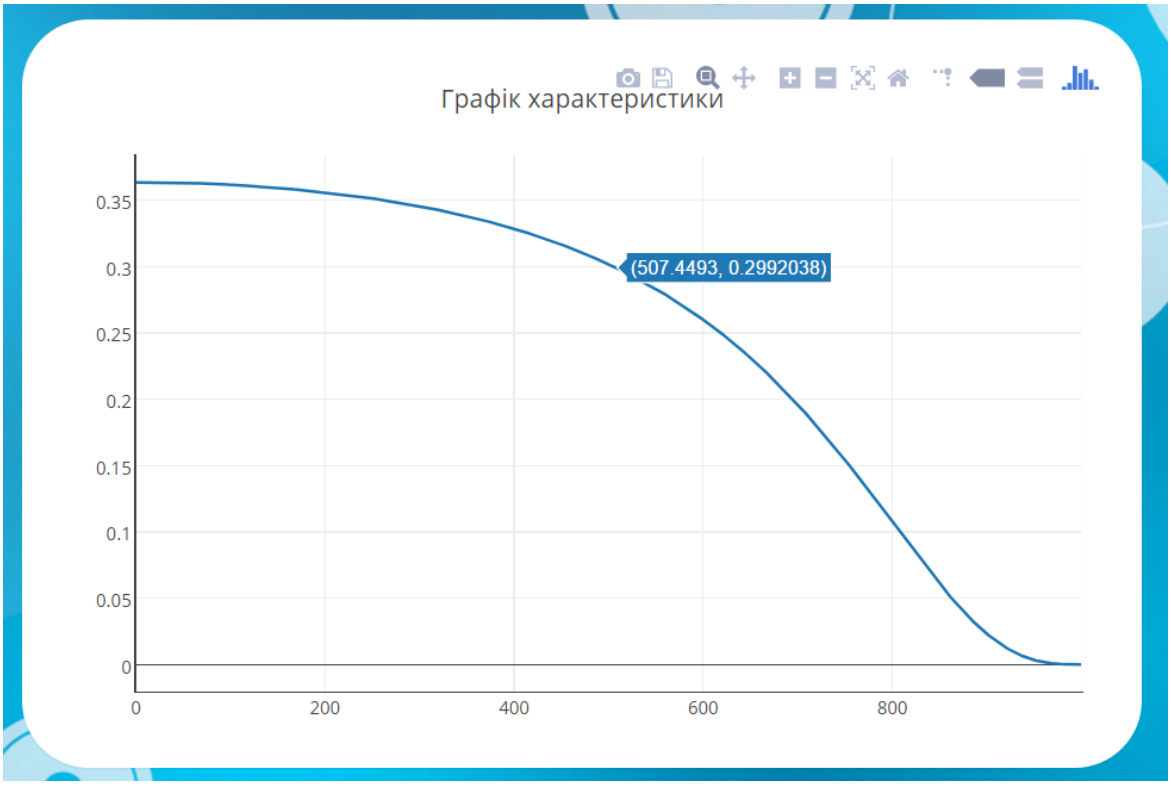

Рисунок 13 – приклад отриманого графіку характеристики

Використання Plotly дає багато додаткових можливостей (приклад графіку Рис. 13):

- Виведення найближчого значення при наведенні, або порівняння значень двох графіків;
- Наближення на віддалення відображеної області;
- Перетаскування відображеної області, виділення бажаної для переглядання області;
- Редагування графіку у хмарному сервісі Plotly;
- Збереження графіка як png зображення.

Представлення TextPartial містить панель на яку виводиться текстове значення поля Text моделі. Це представлення використовується для виведення повідомлень користувачу.

Представлення StartPartial містить одночасно два часткові представлення – TextPartial та GraphPartial. Це представлення використовується у випадках, коли необхідно вивести повідомлення користувачу. В такому разі поточно відображений графік буде затертим, і буде відображене повідомлення для користувача.

Варто зазначити, що Дій контроллера, які починаються з “Show*” лише змінюють значення поля моделі «Який графік виводити» на необхідне значення і перенаправлять подальші дії до GraphPartial

2.2.2 Особливості введення математичних формул у вебі

Побудова математичних формул на веб сторінці – це нетривіальна задача, адже на теперішній час не існує єдиного способу їх генерації.

Існує три способи створення формул на веб сторінках:

- Створення зображень з формулою да додавання її безпосередньо до сторінки. Цей спосіб, очевидно, найпростіший у реалізаціїї, але несе за собою багато недоліків: використані зображення растрові, відсутність динамічної зміни формул;
- Використання TeX розмітки. TeX-розмітка - система типографічного набору, призначена для створення книг, особливо тих, де багато математичних

формул. Підготовка рукопису в форматі системи TeX означає, що комп'ютеру точно вказують, як перетворити текст сторінки. Із недоліків цього способу можна виділити те, що з’являється необхіжність вивчення нового синтаксису.

- Використання MathML розмітки. MathML - це мова розмітки, створена на основі XML для представлення математичних символів і формул в документах мережі Інтернет. MathML рекомендовано до використання математичної групою W3C. Загальний принцип використання MathML полягає в тому, що математичні конструкції вбудовуються в звичайний HTML-документ і (якщо браузер або спеціальна програма підтримує цю специфікацію) адекватно відтворюються при завантаженні документа з мережі.

Для використання на сайті розрахунку цифрових фільтрів було обрано саме останній варіант.

Для коректної побудови розмітки для формул було створено окремий клас, що відповідає за побудову MathML-розмітки із списків коефіцієнтів передавальної функції.

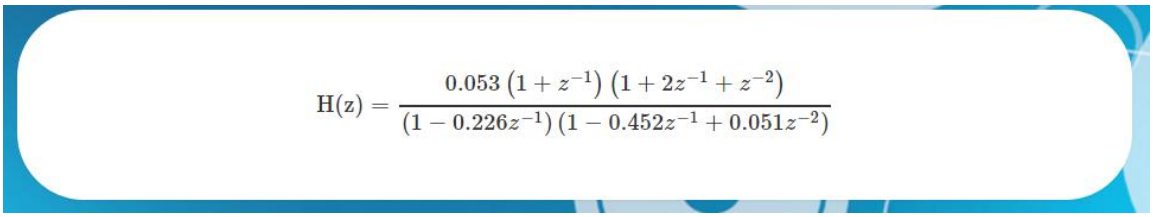

Рисунок 14 - Приклад побудованої формули

```
<math xmlns="http://www.w3.org/1998/Math/MathML" display="block" id="mathBlock">
  <mrow>
    <mtext>H(z)</mtext>
    <mo>=</mo>
    <mfrac>
      <mrow>
        <mn>0.053</mn>
        <mfenced>
          <mrow>
            <mn>1</mn>
            <mo>+</mo>
            <msup>
              <mi>z</mi>
              <mn>-1</mn>
            </msup>
          </mrow>
        </mfenced>
        <mfenced>
          <mrow>
            <mn>1</mn>
            <mo>+</mo>
```

```
            <mn>2</mn>
            <msup>
              <mi>z</mi>
              <mn>-1</mn>
            </msup>
            <mo>+</mo>
            <msup>
              <mi>z</mi>
              <mn>-2</mn>
            </msup>
          </mrow>
        </mfenced>
      </mrow>
      <mrow>
        <mfenced>
          <mrow>
            <mn>1</mn>
            <mo>-</mo>
            <mn>0.226</mn>
            <msup>
              <mi>z</mi>
              <mn>-1</mn>
            </msup>
          </mrow>
        </mfenced>
        <mfenced>
          <mrow>
            <mn>1</mn>
            <mo>-</mo>
            <mn>0.452</mn>
            <msup>
              <mi>z</mi>
              <mn>-1</mn>
            </msup>
            <mo>+</mo>
            <mn>0.051</mn>
            <msup>
              <mi>z</mi>
              <mn>-2</mn>
            </msup>
          </mrow>
        </mfenced>
      </mrow>
    </mfrac>
  </mrow>
</math>
```

Рисунок 15 – Створена MathML-розмітка

Розглянемо структуру побудованої розмітки.

Уся формула зберігається всередині тегу <math>, який визначає межі математичних елементів на HTML- сторінці. Для структурного поділу елементів у MathML використовується тег <mrow>. Текстові елементи містяться в тезі <mtext>. Для створення правильного відображення операції ділення до тегу <mfrac> додаються два елементи <mrow>. Перший елемент відповідає чисельнику, другий – знаменнику. Варто зазначити, що залежність результату від порядку елементів протирічить головним принципам XML, але дозволяє MathML функціонувати мінімізуючи кількість атрибутів елементів.

Для створення дужок використовується тег <mfenced>, який може містити в собі лише один елемент <mrow>.

В кінці кінців, тег <mn> використовують для відображення цифр, <mo> - для відображення математичних операцій, а <mi> - для відображення ідентифікаторів та символів.

Для кожного із перерахованих тегів у MathMLHandler міститься відповідний метод, що дозволяє ефективно використовувати можливості розмітки.

Одним із недоліків такого підходу є обмежена підтримка MathML у сучасних браузерах. Повністю підтримує MathML лише баузер Mozila Firefox.

Для усунення цієї проблеми було використано javascript-фреймворк MathJax, який за допомогою коду javascript дозволяє підтримувати MathML у браузерах, які не мають нативної підтримки MathML.

Оновлення математичної розмітки на сторінці виконується наступним шматком коду:

```
MathJax.Hub.Queue(["Typeset", MathJax.Hub]);
```

Рисунок 16

2.2.3 Реалізація модулю валідації даних моделі

Очевидно, що дані, введені користувачем необхідно перевірити на коректність. Причиною цьому є наявні обмеження на вимоги до характеристик цифрових фільтрів.

Валідаційний модуль представлений класом ValidationController, який перевірає усі наявні вимоги. Результат валідації записується у відповідне поле моделі – Text. Якщо після валідації це поле заповнене, то процес розрахунку буде призупинено та Дію буде перенаправлено на TextPartial для виведення повідомлення користувачу.

Розглянемо наявні у системі валідації.

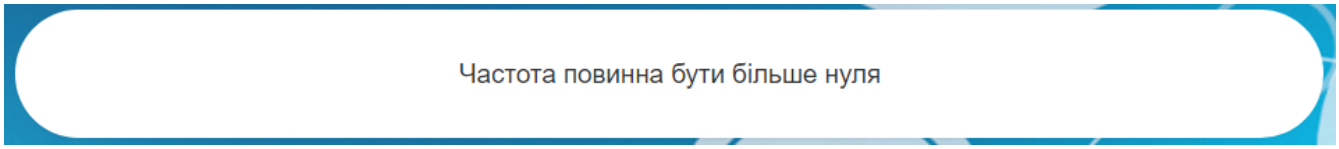

Рисунок 17

Валідаційне повідомлення з Рис. 17 виникатиме у випадку, коли хоча б одна з введених значень частот (Нижня, верхня, нижня для полоси, верхня для полоси, або частота дискретизації) менше, або рівна нулю.

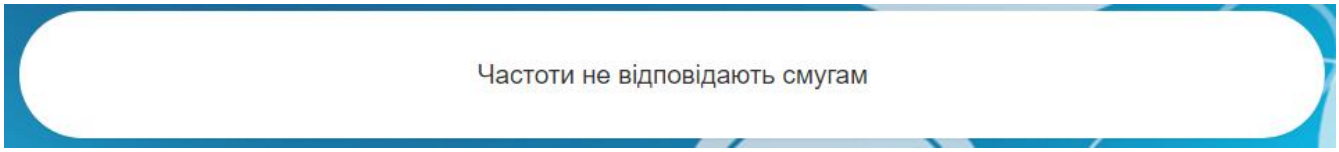

Рисунок 18

Валідаційне повідомлення з Рис. 18 виникатиме у випадку, якщо введені користувачем частоти не відповідають реальним смугам. Наприклад, для фільтру нижніх частот необхідно, щоб верхня частота була більша за нижню тощо.

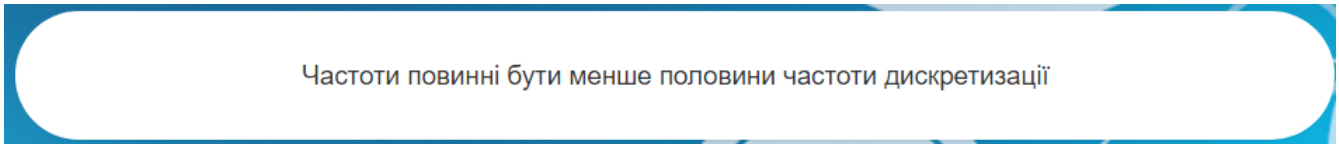

Рисунок 19

Валідаційне повідомлення з Рис. 19 виникатиме у випадку, якщо хоча б одна із введених частот, що використовують у розрахунках буде більше половини частоти дискретизації.

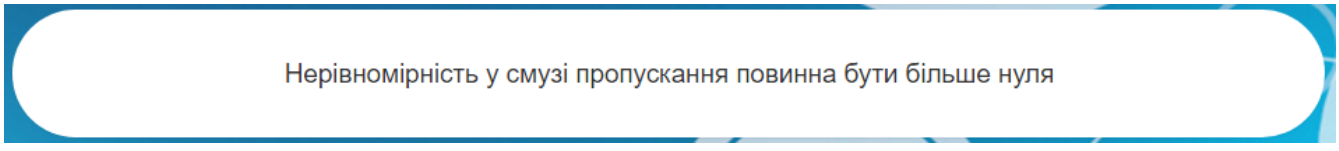

Рисунок 20

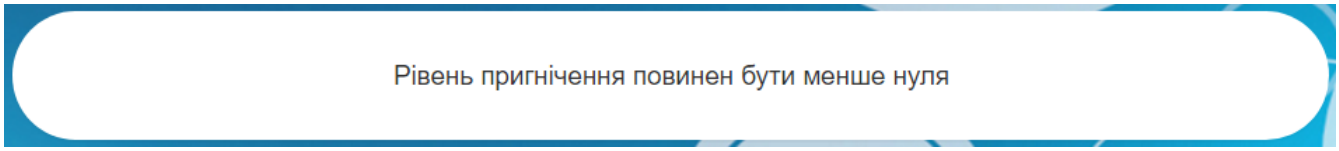

Рисунок 21

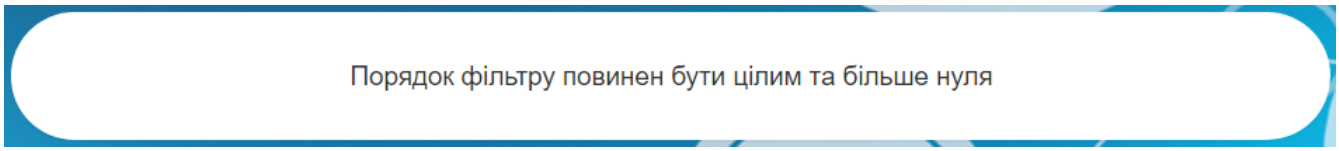

Рисунок 22

Валідаційні повідомлення з Рис. 20 - 22 перевіряють відповідно, що значення поля «Нерівномірність пропускання» повинно бути більше нуля; що значення поля «Рівень пригнічення» - менше нуля, а значення поля «Порядок фільтру» повинно бути цілим додатнім числом.

Варто зазначити, що дві останні перевірки взаємовиключаючі, тобто, з них двох повинна виконатись лише одна, адже доступним до запонення буде лише одне з двох полів.

У випадку ж, якщо ні поле «Рівень пригнічення», ані поле «Порядок фільтру» не заповнені, буде створено валідаційне повідомлення з Рис. 23.

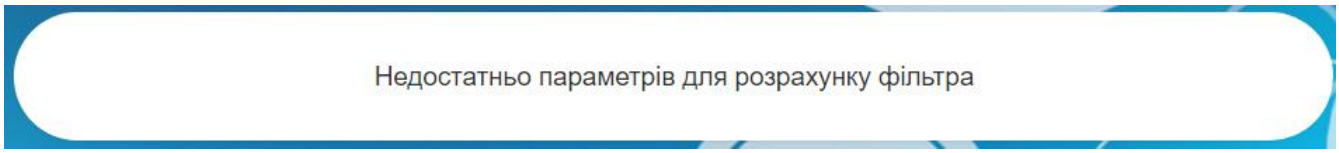

Рисунок 23

Особливістю реалізації даних валідацій можна вважати те, що повідомлення про помилки користувач бачитиме лише по черзі, тобто кожне натиснення кнопки «Розрахувати» може створити лише одне валідаційне повідомлення, не зважаючи на те, що їх може існувати декілька.

2.2.4 Реалізація взаємозв’язку між сайтом і веб-сервісом

Як вже зазначалося вище, для створення зв’язку між сайтом та веб-сервісом було використано протокол SOAP.

У середовищі Visual Studio 2017 для проектів ASP.Net Core вбудованої підтримки SOAP-сервісів немає. Саме тому, щоб підключитись необхідно спершу встановити пакет NuGet: Microsoft WCF Web Service Reference Provider – Preview.

З його допомогою легко можна створити посилання на веб-сервіс, лише вказавши його endpoint – адресу, за якою знаходиться wsdl файл сервісу.

Якщо за вказаною адресою дійсно є wsdl-опис сервісу, то у проекті сайту, у теці «Connected services» буде створено файл Reference.cs. У цьому файлі міститиметься опис доступних методів сервісу. У такому випадку запит до сервісу виглядатиме як зображено на Рис. 24.

```
private buildIirResponse SendRequest(FiltersPortClient client, buildIirRequest request)
{
	var taskResponse = client.buildIirAsync(request);
	var result = taskResponse.Result;
	return result.buildIirResponse;
}
```

Рисунок 24

Відповідь, яка отримується від клієнта сервісу у такому випадку – це асинхронне завдання. Для того щоб дістати дійсний результат розрахунків, необхідно очікувати відповіді від сервісу. Така відповідь записується до статичного контейнера і зберігається до використання користувачем.

2.3 Висновок до розділу

У цьому розділі, на основі попередніх досліджень, було здійснено конкретні практичні рішення для побудови сайту - було описано рішення, які були прийняті на етапі створення макету веб сторінки.

Також у цьому розділі міститься опис практичної реалізації сайту розрахунків цифрових фільтрів.

3 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, який виконує роль інтерфейсу користувача у сайті розрахунку цифрових фільтрів. Продукт був розроблений за допомогою фреймворку ASP.Net Core (ASP.Net 5.0) у середовищі розробки Microsoft Visual Studio 2017 Community.

Програмний продукт призначено для використання на персональних комп'ютерах, чи мобільних засобах, з використанням веб-браузерів під управлінням будь-якої операційної системи.

Нижче наведено аналіз різних варіантів вибору базового фреймворку з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі функції, які потім розподіляються по двом

групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

3.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки інтерфейсу користувача для сайту розрахунку цифрових фільтрів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із будь-яким встановленим веб-браузером та незалежно від оперційної системи, що використовується;

– забезпечувати високу швидкість виконання розрахунків, виведення їх результатів на екран;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку впровадження та вдосконалення;

– передбачати мінімальні витрати на впровадження програмного продукту.

3.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

- F_1 – вибір фреймворку розробки;
- F_2 – розпізнавання вхідних даних;
- F_3 – побудова графіків характеристик.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

a) Використання фреймворку ASP.Net Core;
b) Використання фреймворку Ruby on Rails.
c) Написання строрінки власноруч

Функція F_2:

a) Введення даних вручну у відповідні поля на формі;
b) Використання «слайдерів» для вибору значень.

Функція F_3:

a) Використання можливостей Javascript-фреймворку Plotly для побудови графіків;
b) Використання модулю Charts з веб-фреймоворку DevExtreme;
c) Написання власного модулю для створення графіків.

3.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 26). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 1).

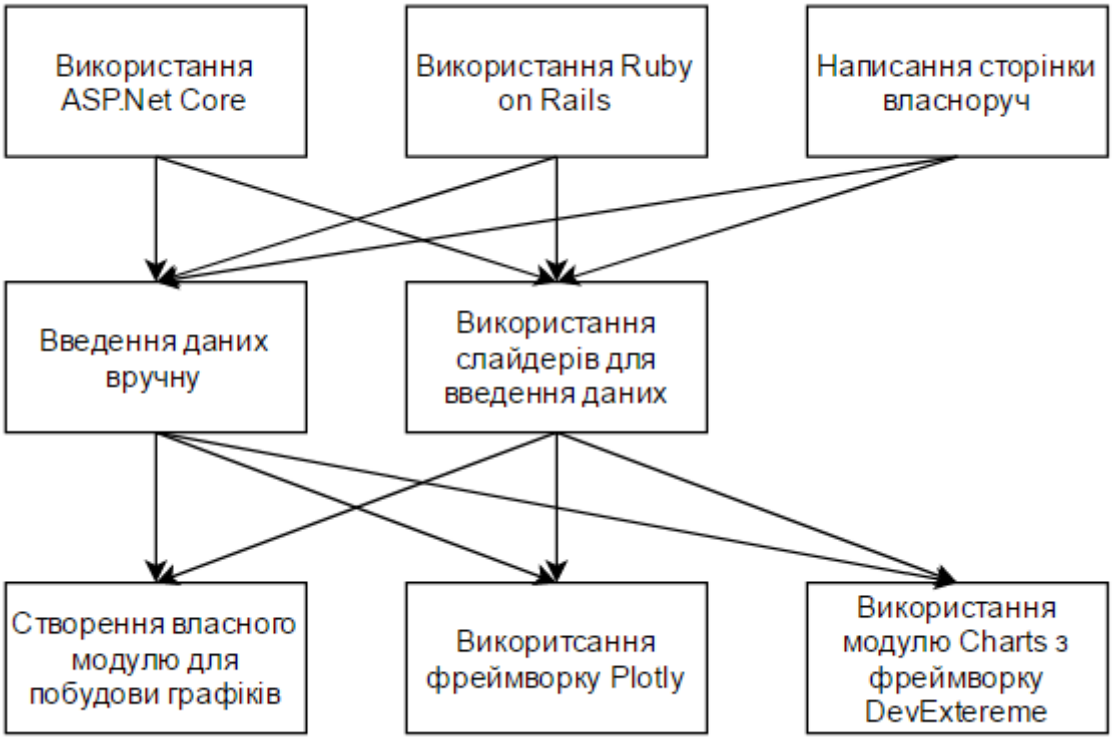

Рисунок 25 – Морфологічна карта

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F_1	*А*	Написання коду займає небагато часу	Залежність від платформи .Net Core або .Net Framework
	Б	Кросплатформенність. Легкість роботи з базами даних	Втрачає швидкодію зі зростанням важкості розрахунків та логіки
	В	Максимальна гнучкість та відповідність вимогам	Зростання часу розробки, через необхідність розробки «з нуля»
F_2	А	Легкий у реалізації	Менша зручність у використання
	Б	Зручність у використанні	Великі витрати часу на розробку. Низька наглядність інформації

Основні функції	Варіанти реалізації	Переваги	Недоліки
F_3	*А*	Створений для побудови графіків. Наявність додаткових функції для перегляду інформації	Створений код виконується на стороні клієнта (веб-браузера)
	Б	Легкість і зручність розробки. Побудова графіків на сервері.	Використовується лише один модуль.
	В	Повна відповідність вимогам до графіку. Гнучкість та зручність у використання та розробці.	Надзвичайно великі витрати часу на розробку.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці веб-сторінки деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Розробка програмного продукту має очевидні часові обмеження. Також, у сучасній веб-розробці недоцільно розробляти продукти «з нуля». Для зменшення часу розробки та для уникнення вже вирішених у сучасних фреймворках проблем, доцільніше використовувати певну програмну базу.

Саме тому, розглядатимемо лише дві опції: А, та Б.

Функція *F2:*

Для коректної роботи веб-додатку, користувач повинен заповнити 9 полів зі значеннями. Для цього, щоб підвищити точність вводу, доцільніше використовувати звичайні поля.

Саме тому, опцію Б відкинуто.

Функція *F3:*

Побудова графіку у продукті – це один з основних показників точності результату, тому, для підвищення якості, опція Б буде відкинута. Використання

опції В не доцільне, через надзвичайно високі вимоги для побудови, задля використання єдиного модуля.

Саме тому, розглядатимемо лише опцію А.

В результаті, маємо два можливі варіанти, які й будуть досліджені:

1. F1a – F2a – F3a
2. F1a – F2a – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

3.2 Обґрунтування системи параметрів ПП

3.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – час побудови сторінки (час відповіді сторінки)
- *X2* – час перебудови частини сторінки (перебудова графіку)
- *X2* – можливості зі збереження даних;
- *X4* – потенційний об’єм необхідного коду.

X1: Відображає швидкодію процесу генерації html розмітки для веб-сторінки залежно від обраного фреймворку розробки.

X2: Відображає швидкодію процесу генерації частини сторінки, яка відображає необхідний графік характеристики

X3: Відображає об’єм оперативної пам’яті персонального комп’ютера-серверу, необхідний для збереження та обробки даних під час виконання запитів від коритсувачів.

*X4:*Показує розмір програмного коду який необхідно створити безпосередньо розробнику для виконання усіх функціональних вимог.

3.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 2.

Таблиця 2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Час побудови сторінки	X1	с	10	5	1
Час перебудови частини сторінки	X2	с	5	1	Миттєво
Можливості зі збереження даних	X3	Мб	32	16	8
Потенційний об’єм необхідного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 2 будуються графічні характеристики параметрів – рис. 26 – рис. 29.

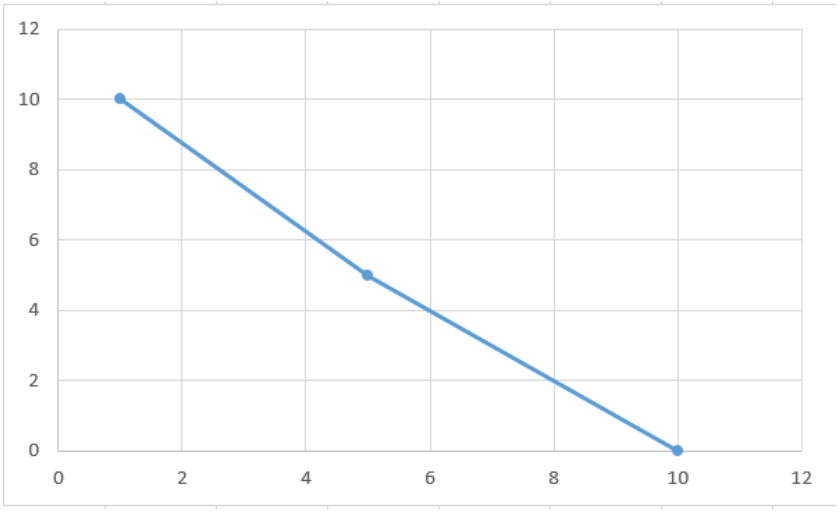

Рисунок 26 – X1, Час побудови сторінки

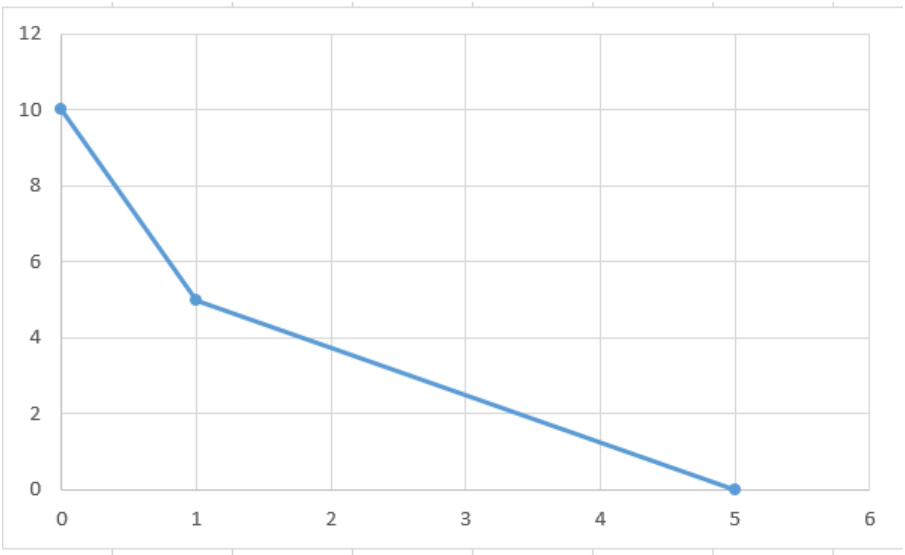

Рисунок 27 – X2, Час перебудови частини сторінки

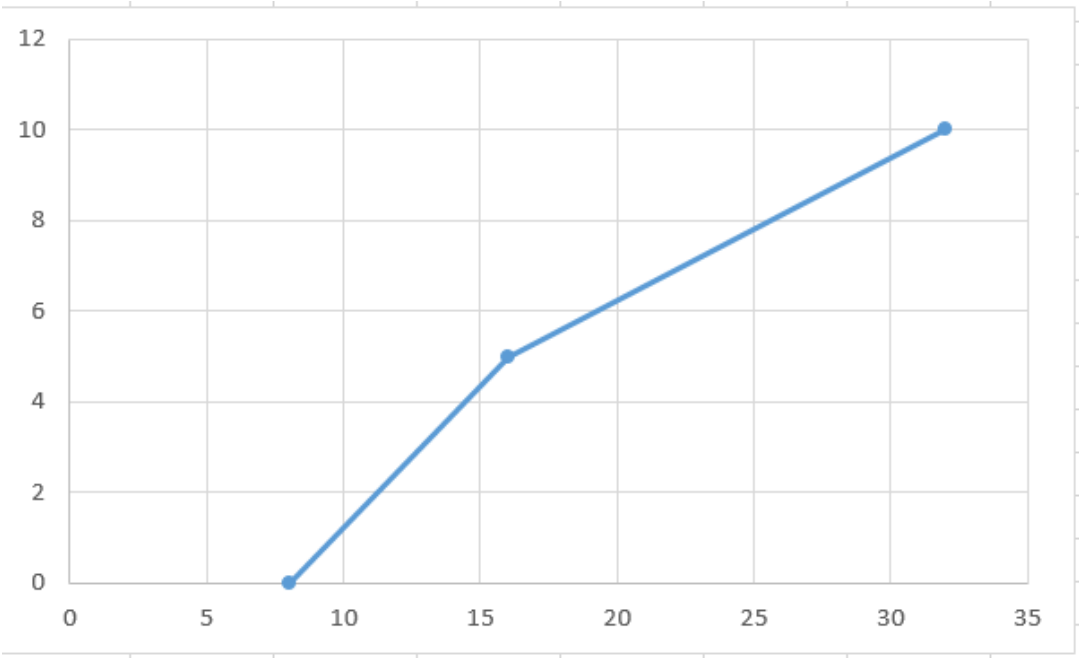

Рисунок 28 – X3, Можливості зі збереження даних

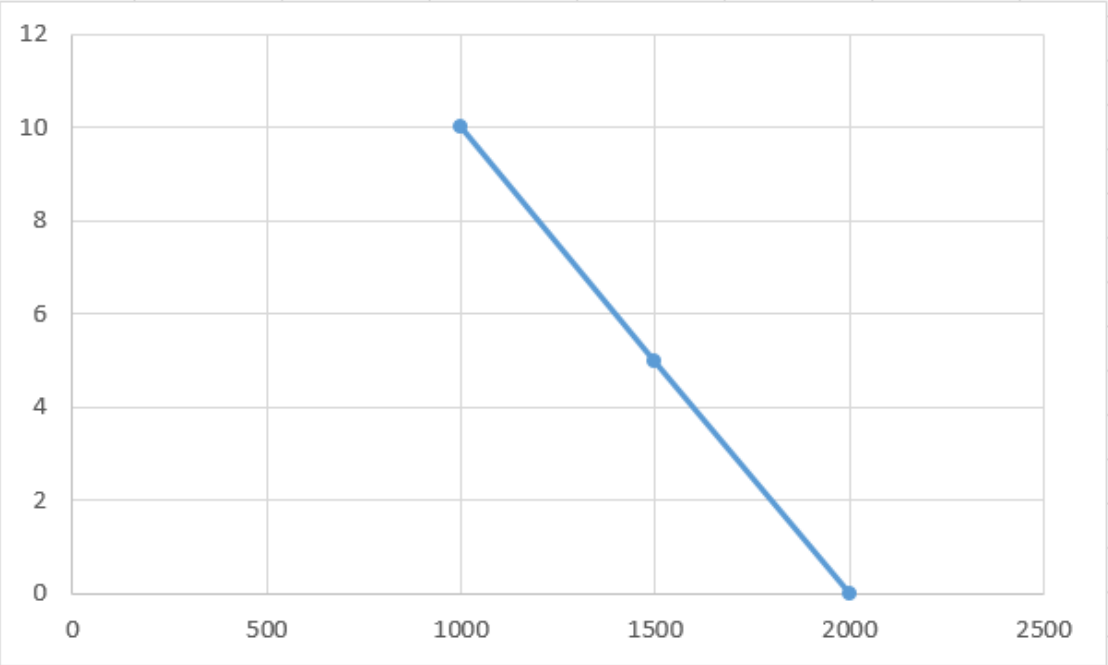

Рисунок 29 – X4, Потенційний об’єм необхідного коду

3.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає зручніший у використанні інтерфейс для розрахунку параметрів цифрових фільтрів.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 3.

Таблиця 3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта					Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5			
X1	Час побудови сторінки	с	1	1	2	1	2	5	-5	25
X2	Час перебудови частини сторінки	с	2	3	1	2	1	9	-1	2
X3	Можливості зі збереження даних	Мб	3	2	3	3	3	14	4	16
X4	Потенційний об’єм необхідного коду	кількість строк коду	4	4	4	4	4	20	10	100
	Разом		10	10	10	10	10	50	0	143

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 50,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 10.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 143.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 143}{5^2(4^3 - 4)} = \frac{3984}{1500} = 1.144 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.

Таблиця 4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	>	=	=	<	=	1
X1 і X3	>	>	=	>	<	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	=	=	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	<	>	<	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5, \text{при } X_i > X_j \\ 1{,}0, \text{при } X_i = X_j \\ 0{,}5, \text{при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \square\, aij \,\square$.

Для кожного параметра зробимо розрахунок вагомості $K_{\text{ві}}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,0	1,5	1,5	5,0	0,312	19,0	0,322	70,25	0,325
X2	1,0	1,0	1,5	1,5	5,0	0,312	19,0	0,322	70,25	0,325
X3	0,5	0,5	1,0	1,5	3,5	0,218	12,25	0,205	40.75	0,188
X4	0,5	0,5	0,5	1,0	2,5	0,167	9,25	0,155	34.37	0,159
Всього:					16	1	59,5	1	215.62	1

3.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	X_1	2	9	0,325	2.925
		X_2	0.5	7.5	0,325	2.437
	Б	X_1	3	8	0,325	2.6
		X_2	1	5	0,325	1.625
F_2	А	X_3	12	7.5	0,188	1.41
F_3	А	X_4	1250	7.5	0,159	1.192

За даними з таблиці 6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

Визначаємо рівень якості кожного з варіантів:

$$К_{К1} = 2.925 + 2.6 + 1.41 + 1.192 = 8.127$$

$$К_{К2} = 2.437 + 1.625 + 1.41 + 1.192 = 6.664$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

3.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$Т_О = Т_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М},$$

де $Т_Р$ – трудомісткість розробки ПП;$К_П$ – поправочний коефіцієнт;$К_{СК}$ – коефіцієнт на складність вхідної інформації; $К_М$ – коефіцієнт рівня мови програмування;$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість

дорівнює: $Т_Р = 15$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П = 1.7$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний: $К_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ} = 0.8$. Тоді, за формулою, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 70 \cdot 1.7 \cdot 0.8 = 20.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто

$Т_Р = 30$ людино-днів, $К_П = 0.9, К_{СК} = 1, К_{СТ} = 0.8$:

$Т_2 = 30 \cdot 0.9 \cdot 0.8 = 21.6$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_1 = 20.4 \cdot 8 = 163.2$ людино-годин;

$Т_2 = 21.6 \cdot 8 = 172.8$ людино-годин;

Найбільш високу трудомісткість має завдання 2.

В розробці беруть участь один програміст з окладом 10000 грн., один дизайнер з окладом 8000грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.},$$

де М – місячний оклад працівників;T_m – кількість робочих днів на місяць;t – кількість робочих годин в день.

$$С_Ч = \frac{10000 + 8000}{2 \cdot 21 \cdot 8} = 53{,}57 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д,$$

де $С_Ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за завданнями становить:

I. $С_{ЗП} = 53{,}57 \cdot 163.2 \cdot 1.2 = 10491.15$ грн.

II. $С_{ЗП} = 53.57 \cdot 172.8 \cdot 1.2 = 11108.28$ грн.

Відрахування на єдиний соціальний внесок становлять 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 10491.15 \cdot 0.22 = 2308.06$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 11108.28 \cdot 0.22 = 2443.83$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot M \cdot K_З = 12 \cdot 10000 \cdot 0{,}2 = 24000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + K_З) = 24000 \cdot (1 + 0.2) = 28800 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 28800 \cdot 0{,}22 = 6336 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 12000 грн.

$$С_А = К_{ТМ} \cdot K_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 12000 = 3450 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 12000 \cdot 0.05 = 690 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = (365 - 118 - 8 - 16) \cdot 8 \cdot 0.9 = 1605.6 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot K_З \cdot Ц_{ЕН} = 1605.6 \cdot 0{,}156 \cdot 1.93 = 483.42 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 12000 \cdot 0{,}67 = 8040 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$$С_{ЕКС} = 28800 + 6336 + 3450 + 690 + 483.42 + 8040 = 47799.42 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{С_{ЕКС}}{Т_{ЕФ}} = \frac{47799.42}{1605.6} = 29.78 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} T$$

I. $С_М = 29.78 \cdot 163.2 = 4860.10$ грн.;

II. $С_М = 29.78 \cdot 172.8 = 5145.99$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0{,}67$$

I. $С_Н = 10491.15 \cdot 0{,}67 = 7029.08$ грн.;

II. $С_Н = 11108.28 \cdot 0{,}67 = 7442.55$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 10491.15 + 2308.06 + 4860.10 + 7029.08 = 24688.39$ грн.;

II. $С_{ПП} = 11108.28 + 2443.83 + 5145.99 + 7442.55 = 26140.65$ грн.;

3.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР_j} = {К_{К_j}}/{С_{Ф_j}},$$

$$К_{ТЕР_1} = \frac{8.127}{24688.39} = 3.29e-4;$$

$$К_{ТЕР_2} = \frac{6.664}{26140.65} = 2.54e-4;$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР_1} = 3.29e-4$.

3.6 Висновки до розділу

В даному розділі проведено повний функціонально-вартісний аналіз програмного продукту – інтерфейсу користувача для сайту розрахунків цифрових фільтрів, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження продукту з технічної точки зору: було визначено його основні функції та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій .

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного продукту що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу

оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР_1} = 2.79e-4$

Цей варіант реалізації програмного продукту має такі параметри:

- Використаний фреймворк – ASP.Net Core;
- Введення даних власноруч, у відповідні поля
- Побудова графіків за допомогою фреймворку Plotly.

Даний варіант виконання програмного комплексу задовольняє поставленим вимогам.

ВИСНОВОК

В ході даної дипломної роботи було досліджено особливості сучасної розробки веб-сайтів та розроблено інтерфейс користувача для сайту розрахунку цифрових фільтрів.

Було визначено, що на сучасному етапі розвитку мережі Інтернет, веб-сайт не може приваблювати велику кількість користувачів, маючи лише базовий, текстовий користувацький інтерфейс. Сучасні веб-сторінки повинні бути зручними у використанні та комфортними для користувача. Також важливо зазначити, що у наш час більшість пристроїв, що використовують мережу Інтернет – мобільні.

Як наслідок, головними критеріями якості веб-сайтів стала їх ергономічність та юзабіліті. Саме принципи юзабіліті використовувалися при розробці макету веб-сторінки.

Іншим важливим аспектом розробки веб-сайтів було визначено вибір кольорової гамми. Для цього було досліджено теорію кольору – колірні кола та схеми вибору гамми по ньому.

Було здійснено порівняльний аналіз популярних існуючих технологій. Порівняння здійснювалося за наступними характеристиками: швидкість генерації веб-сторінок, наявність стандартних засобів зв’язку веб-сторінки із веб-сервісом, наявність інтегрованого середовища розробки, необхідність додаткових засобів для розробки та роботи сайту. За наведеними параметрами було обрано технологію ASP.Net Core.

Як засіб зв’язку веб-сайту із веб-сервісом було обрано протокол SOAP. В поєднанні із засобами Microsoft WCF Web Service Reference Provider – Preview, доступними через NuGet у середовищі Visual Studio 2017, це дозволило мінімізувати кількість коду, що необхідна для створення простого способу використання веб-сервісу розрахунків із веб-сайту.

Для початку розробки було створено макет веб-сторінки, яка враховувала усі вимоги до ергономічності та юзабіліті, перераховані у першому розділі. Контент сторінки було розділено на функціональні частини та виділено п’ятьма панелями. Колірна гамма були обрана з метою зменшення втоми користувача від веб-сторінки та концентрації уваги на безпосередньо результатах розрахунків (передавальній функції та графіках характеристик). Обрані кольори – помаранчевий та синій, а схема на колірному колі – діадна, або контраст.

Безпосередньо розробка веб-сайте розрахунку цифрових фільтрів проводилася в середовищі Visual Studio 2017. Продукт був розроблений за паптерном MVC, тобто може бути поділений на три частини: модель, представлення та контроллер.

Візуальна частина сайту створена за допомогою п’яти представлень формату cshtml – суміші типової html-розмітки із елементами серверного C# коду. З них одне представлення повне, а чотири – часткові.

Програмну частину сайту можна поділити на наступні модулі:

- Допоміжний модуль для побудови HTML-розмітки ;
- Модуль-будівник математичних формул;
- Модуль валідації;
- Модуль запитів до сервісу.

Також було проведено функціонально-вартісний аналіз продукту, з метою визначення його економічної ефективності. Розроблений продукт отримав показник техніко-економічного рівня якості $К_{ТЕР_1} = 3.29e-4$, який виявився кращим за іншу альтернативу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Макдональд М. HTML5. Недостающее руководство / Макдональд М - БХВ-Петербург – 2012 —480 с.

2. Горностаева Е. А. Современные проблемы дизайна сайтов и вызовы нового времени /Горностаева Е. А. // Молодой ученый. — 2015. — №1. — С. 38-40.

3. Krug S. Rocket Surgery Made Easy : The Do-it-yourself Guide to Finding and Fixing Usability Problems / Krug S - Pearson Education —2010 —168 с.

4. Zeldman Jeffery. Taking Your Talent to the Web: Making the Transition from Graphic Design to Web Design. / Zeldman Jeffery - New Riders. — 2001 —448 с.

5. Якоб Нильсен. Веб-дизайн: анализ удобства использования веб-сайтов по движению глаз / Якоб Нильсен, Кара Перниче - Вильямс — 2010 — 480 с

6. Jacob Nielsen. Designing Web Usability /Jacob Nielsen - New Riders —2000 —432 с.

7. Офіційна документація Microsoft з ASP.Net Core: Introduction to ASP.NET Core — Режим доступа: https://docs.microsoft.com/en-us/aspnet/core/. Дата доступа: 14.10.2016

8. Адам Фрімен. ASP.NET Core MVC с примерами на C# для профессионалов // Вильямс — 2017 — 992 с.

9. Переклад специфікації SOAP Version 1.2 Part 0. Primer W3C Recommendation — https://www.w3.org/2002/07/soap-translation/russian/part0.html Дата доступу: 24.06.2003

10. Опис специфікації Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language W3C Recommendation — Режим доступа: https://www.w3.org/TR/wsdl20/. Дата доступа: 26.06.2007