Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів моніторингу систем Front-end

Виконав: студент 2 курсу, групи ІПЗм-18-1

Швець В.С.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньої-наукової програми
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник доц. Валенда Н.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Дудар З.В.
(підпис) (прізвище, ініціали)

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Швецю Володимиру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів моніторингу систем Front-end»
затверджена наказом по університету від ____ _____20__ р.
№ _____
2. Термін подання студентом роботи до екзаменаційної комісії 21 травня 2020р.
3. Вихідні дані до роботи: вимоги до атестаційної роботи, що буде розроблено, технології PHP, JS, HTML5, CSS3 та C#, середовище розробок Visual Studio, Visual Studio Code та Adobe Photoshop CC.

4. Перелік питань, що потрібно опрацювати в роботі: мета роботи, аналіз предметної області, постановка задачі, моделювання та аналіз, опис розробленої системи, висновки.

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц Валенда Н.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка	10.03.2020	виконано
2	Розробка моделі взаємодії даних	22.03.2020	виконано
3	Розробка структури сховища даних	23.03.2020	виконано
4	Створення коду програми	25.03.2020	виконано
5	Тестування і налагодження програми	04.04.2020	виконано
6	Підготовка пояснювальної записки	07.04.2020	виконано
7	Спецчастина	09.04.2020	виконано
8	Підготовка презентації та доповіді	08.05.2020	виконано
9	Попередній захист	18.05.2020	виконано
10	Нормконтроль, рецензування	11.05.2020	виконано
11	Занесення роботи в електронний архів	16.05.2020	виконано
12	Допуск до захисту у зав. кафедри	21.05.2020	виконано

Дата видачі завдання _____ _____20__ р.

Студент _____ Швець В.С.
(підпис)

Керівник роботи ___<u>_____</u> <u>__доц. Валенда Н.А.</u>
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 85 с., 20 рис., 30 джер.
FRONT-END РОЗРОБКА, HTML, JAVASCRIPT, CSS, WEB-ДОДАТОК, RATIONAL ROSE, UML, ACCESS, C#.

Метою роботи є аналіз підходів і розробка методу оцінювання Front-end системи на основі моделі якості. У роботі досліджено поняття Front-end розробки та її місце в створення веб-орієнтованих систем. Визначено основні вимоги, що висуваються до сучасних веб-додатків. Досліджено основні технології, що використовуються в розробці сучасних системи Front-end. Надано характеристики основних моделей якості, які застосовуються до інформаційних систем. Розроблено метод оцінювання системи Front-end на основі моделі якості програмного забезпечення.

Практичне значення одержаних результатів полягає в можливості побудови інтегральної оцінки Front-end системи, яка дозволяє врахувати особливості використання програмного забезпечення такого типу. Це дасть можливість в подальшому визначити найбільш уразливі місця клієнтського додатку, а також досліджувати його поведінку в динаміці.

FRONT-END РОЗРОБКА, HTML, JAVASCRIPT, CSS, WEB-APPLICATION, RATIONAL ROSE, UML, ACCESS, C#.

The aim of the work was to cope with the analysis og the approaches and develop a method for evaluating the Front-end system based on the quality model. The concept of Front-end development and its place in the creation of web-oriented systems is investigated in the work. The basic requirements for modern web applications are defined. The main technologies used in the development of modern Front-end systems are studied. The characteristics of the main quality

models applied to information systems are given. A method for evaluating the Front-end system based on a software quality model has been developed.

The practical significance of the obtained results lies in the possibility of building an integrated assessment of the Front-end system, which allows to take into account the peculiarities of the use of software of this type. This will allow you to further identify the most vulnerable areas of the client application, as well as to explore its behavior in the dynamics.

ЗМІСТ

ВСТУП

В останні роки в розробці та використанні веб-додатків спостерігаються величезні зміни, з’являються нові тенденції в технологіях розробок, які повністю змінюють практику створення таких програмних систем. Якщо раніше веб-додаток представляв собою статичний документ, що формувався на веб-сервері, то зараз він представляє собою об’єкт, зміст якого змінюється динамічно внаслідок дій користувача[1].

Внаслідок цього суттєво зростають обсяги інформації, що зумовлює ускладнення як самого процесу розробки, так і процесу контролю та управління веб-додатками. Звідси зростає важливість застосування принципів розробки якісного програмного забезпечення.

Якість програмного забезпечення – це те, на скільки програмний продукт задовольняє вимогам, що до нього висуваються. Зараз існує велика кількість моделей якості, які засновані на стандартах ISO9000, що регулюють загальні принципи забезпечення якості в усіх галузях. Для програмного забезпечення ці моделі почали застосовуватися ще у 80-х роках минулого століття. З розвитком обчислювальної техніки, появою нових технологій розвивалися нові моделі якості, які враховували зміни, що відбувалися.

Зараз якість програмної розробки є інтегральною характеристикою, яка повинна включати широкий спектр властивостей продукту. Якщо для Back-end розробок показники для оцінювання використовуються давно і відповідають всім сучасним тенденціям в області інженерії програмного забезпечення, то для Front-end ці питання почали розглядатися зовсім недавно, коли веб-сайти перестали бути статичними документо-орієнтованими об’єктами. Перенесення функціональності на бік клієнта, залежність успіхів бізнесу від якості створюваного веб-сайту – все це змусило розглядати питання оцінки Front-end систем.

Традиційні показники, такі, як час завантаження сторінки менш ефективні для сучасних веб-сайтів. Особливістю веб-застосунків в порівнянні з іншими класами програмних засобів є те, що крім коду істотну роль в їхньому успіху відіграє інформаційна архітектура і контент.

Якість програмних засобів оцінюється за допомогою різних моделей та методів, але в більшості випадків вони не завжди враховують особливості саме Front-end розробок, для яких характерна орієнтація на користувача а, отже, необхідність використання в моделях якості так званих "user experience" (користувальницьких практик). Тому розробка нових методів дослідження Front-end систем є актуальною задачею.

Метою дослідження є аналіз підходів і розробка методу оцінювання Front-end системи на основі моделі якості.

Для досягнення мети дослідження необхідно вирішити такі завдання:

– дослідити поняття Front-end розробки та її місце в створення веб-орієнтованих систем;

– визначити основні вимоги, що висуваються до сучасних веб-додатків;

– дослідити основні технології, що використовуються в розробці сучасних системи Front-end;

– дослідити основні моделі якості, які застосовуються до інформаційних систем;

– розробити метод оцінювання системи Front-end на основі моделі якості програмного забезпечення.

Об’єкт дослідження – системи Front-end.

Предметом дослідження є моделі оцінки якості для систем Front-end.

Методи дослідження. Дослідження ґрунтується на методах аналізу наукових праць та джерел зарубіжних та вітчизняних авторів, на методах програмної інженерії для визначення структури веб-орієнтованих додатків, на методах оцінювання якості, на методах рейтингових оцінок.

Елементи наукової новизни роботи: удосконалено метод оцінювання якості системи Front-end на основі використання спеціального виду інтегрального показника.

Практичне значення одержаних результатів полягає в можливості побудови інтегральної оцінки Front-end системи, яка дозволяє врахувати особливості використання програмного забезпечення такого типу. Це дасть можливість в подальшому визначити найбільш уразливі місця клієнтського додатку, а також досліджувати його поведінку в динаміці.

Структура роботи. Дипломне дослідження складається зі вступу, трьох основних розділів, загальних висновків та списку використаних джерел.

1. АНАЛІЗ СТАНУ FRONT-END РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Поняття Front-end розробки

Протягом останніх кількох років переважною платформою для передачі вмісту та послуг була Інтернет. Таким чином, компаніям будь-яких форм і розмірів потрібно бути в Інтернеті та представляти своїх потенційних клієнтів та постійних клієнтів.

Спочатку веб-додатки представляли собою документо-орієнтовані додатки, але з розвитком інформаційно-комунікаційних технологій вони почали наповнюватися динамічним змістом, тобто з’явилась можливість виконувати серверний код в залежності від дії користувача в браузері [21].

Зростання обсягу даних, що використовуються, підвищення вимог до надійності та безпеки, продуктивності веб-додатків викликало необхідність відділення бізнес-логіки, що реалізована в додатку, а також сервісів обробки даних і транзакцій від його інтерфейса. В цьому випадку в самому веб-додатку зазвичай залишається презентаційна частина, а бізнес-логіка, обробка даних, реалізація транзакцій переноситься в сервер додатків у вигляді бізнес-об’єктів. Зараз подібні задачі покладаються на веб-сервіси – незалежні від платформи, об’єктної моделі та клієнта програмні компоненти, які можна викликати з клієнтських веб-додатків через протокол SOAP, який базується на протоколі HTTP та мові XML.

Архітектура веб-додатків – це структура, що складається з взаємозв'язків та взаємодій між компонентами додатків, такими як системи серверного програмного забезпечення, інтерфейси користувача та бази даних. Загальна концепція архітектури веб-додатків відповідає концепції користувача браузера, який запускає додаток, здатний запускатись на кількох веб-сайтах [21; 30].

По суті, архітектуру веб-додатків можна визначити таким чином:

– користувач переглядає певну URL-адресу, яку браузер знаходить та запитує.

– по мережі дані надсилаються з сервера в браузер, потім виконуються браузером, щоб він міг відображати запитувану сторінку.

– користувач переглядає та взаємодіє зі сторінкою.

Архітектура веб-додатків складається з декількох компонентів, які допомагають його побудувати. Ці компоненти можна розділити на дві частини: компоненти програмного інтерфейсу користувача та структурні компоненти.

Компоненти програмного інтерфейсу користувача відносяться до веб-сторінок, на яких відображаються інформаційні панелі, журнали, сповіщення, налаштування конфігурації тощо. Вони не мають значення для структурної розробки програми та орієнтуються більше на користувальницький інтерфейс / досвід.

Структурні компоненти, що ж основою процесу розробки додатків:

– веб-браузер або клієнт – це інтерфейс передачі функціональності веб-додатків, з якими користувач взаємодіє. Цей вміст, доставлений клієнтові, може бути розроблений за допомогою HTML, JavaScript та CSS і не потребує адаптації, пов'язаних з операційною системою. По суті, веб-браузер або клієнт управляє тим, як взаємодіють кінцеві користувачі з додатком.

– сервер веб-додатків – керує бізнес-логікою та збереженням даних і може бути побудований за допомогою PHP, Python, Java, Ruby, .NET, Node.js тощо. Він складається з принаймні централізованого центру або центру управління для підтримки багатошарових програм.

– сервер бази даних надає та зберігає відповідні дані для програми. Крім того, він також може надавати бізнес-логіку та іншу інформацію, якою керує сервер веб-додатків [17; 21].

Таким чином, в загальному випадку архітектура веб-додатку має вигляд наведений на рис. 1.1.

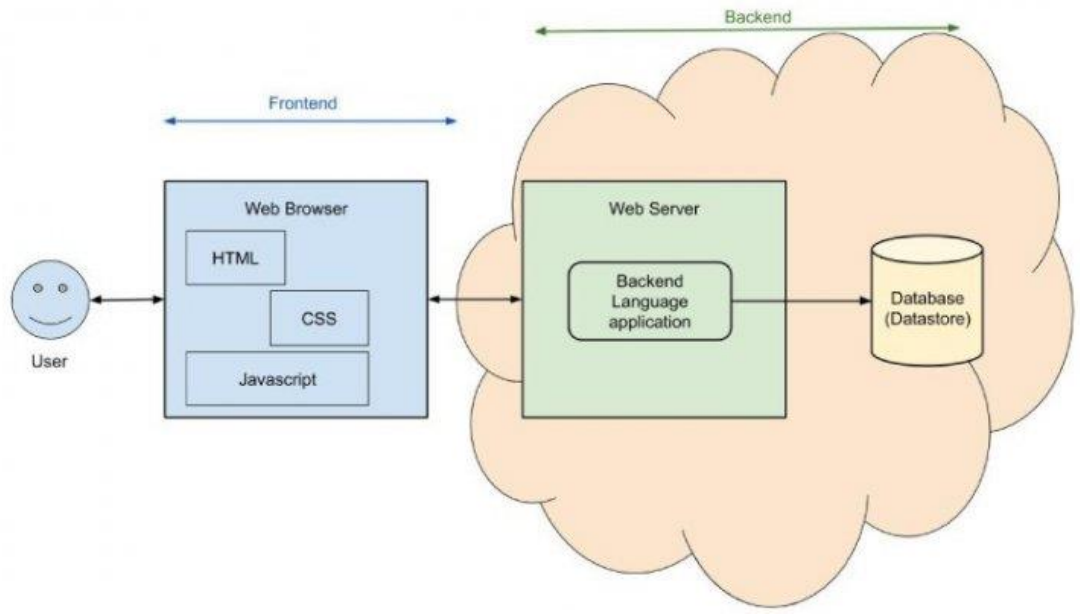

Рисунок 1.1 – Узагальнена структура веб-додатку

Існує три основні підходи до архітектурного проектування веб-застосунків:

– монолітний підхід.

– модульний підхід.

– сервіс-орієнтований підхід.

Монолітна архітектура не передбачає складну структуру веб-застосунку, оскільки вся бізнес-логіка зберігається на сервері, а необхідні дані в базі даних. Як правило подібні програми не відрізняються складністю в розробці і великою вартістю на ранніх етапах. Нова функціональність додається легко і швидко.

Однак підтримувати монолітну систему в довгостроковій перспективі дуже складно і дорого, тому що команди розробників можуть змінюватися і додавати чи виправляти необхідний функціонал стає дедалі важче. Крім того, в рамках веб-додатки однією з проблем «моноліту» є масштабованість. Всі складові системи розташовані в одній точці, а тому потужність самої точки повинна бути відповідною для підтримки сервера і бази даних в робочому стані.

В модульній архітектурі весь функціонал програми розбивається на окремі модулі, кожен з яких відповідає за певну частину функціоналу програми. Кожен модуль програми є функціонально незалежним від іншого, а тому його застосування в різних ділянках коду не викликатиме збоїв в працездатності, по суті це надає простоту рефакторінга і перенесення меж модуля. Варто відзначити, що при зміні одного модуля, інші модулі порушені не будуть, а тому спрощується завдання выдлагодження загальноъ програми. Все, що потрібно, це налаштувати інші модулі, що використовують функціонал зміненого модуля під нові можливості і вимоги останнього, якщо зміни в ньому можуть призвести до збою в роботі програми [16; 19].

Крім того, всі модулі взаємодіють один з одним в синхронному порядку, оскільки знаходяться на одній апаратній частині, а при необхідності для модуля можна виділити окрему базу даних або ж частина даних, з якими цей модуль буде працювати.

Прикладом такої архітектури можуть виступати сучасні PHP фреймворки, які застосовуються для швидкої розробки веб-додатків. В даних фреймворках кожен створений клас-модуль має функціонал конкретної спрямованості і ніяк не взаємодіє безпосередньо з тією частиною програми, для якої не був призначений.

Варто відзначити, що розробка додатків на основі модульної архітектури, починаючи з перших етапів життєвого циклу, вище за вартістю, ніж розробка аналогічного монолітного застосунка, оскільки при внесенні нового функціоналу завжди постає питання щодо того, який модуль буде реалізовувати цей функціонал.

Модульний підхід дозволяє розробляти більш складні програми, що складаються з великої кількості різних модулів.

Подальший розвиток в області розробки програмних за стосунків призвів до винесення окремих модулів на окремі апаратні частини, використання для розробки модулів найбільш зручних засобів, тобто, фактично, до появи сервісно-орієнтованої архітектури.

Сервіси є окремі цілком самодостатні модулі, що володіють власною апаратною базою, а саме розташовуються на окремому сервері. Крім того, вони можуть володіти власною базою даних, а оскільки вони розташовуються на окремих апаратних пристроях, навіть якщо віртуально, то і взаємодія між сервісами здійснюється асинхронно, що може надати як певні переваги, так і деякі недоліки.

Однак, одним з ключових переваг сервіс-орієнтованої архітектури є те, що вона надає можливість незалежного масштабування компонентів, тобто дозволяє збільшувати потужність тільки тих сервісів, які цього потребують, не зачіпаючи інші.

Можливість розробки за стосунків на основі сервіс-орієнтованої архітектури надає можливість розроблювати сервісів на різних мовах програмування. Простий приклад: сервіс доставки повідомлень може бути написаний на NodeJS, де мовою програмування є JavaScript, в той час як основна програма, що використовує цей сервіс, може складатися зі зв'язки React на клієнтській частині і PHP на серверній частині. Для взаємодії цих двох частин необхідно тільки правильно використовувати програмний інтерфейс сервісу в основній частині програми. Розробка сервісів відбувається окремо один від одного, а тому зміни в коді одного сервісу будуть впливати тільки на нього, якщо тільки не змінилася видача даних в програмному інтерфейсі сервісу, що відбувається вкрай рідко, оскільки з форматом виведення визначаються ще на перших етапах розробки сервісу.

З іншого боку, використання сервіс-орієнтованої архітектури вимагає строгого і чіткого окреслення меж функціоналу того чи іншого сервісу. Крім того, оскільки кожен сервіс розробляється на окремій апаратній частині, то перевірити компілятором того чи іншого сервісу передані дані з іншої частини програми буде неможливо, оскільки схеми взаємодії між сервісами застосунку в явному вигляді не існує і сервіс знає тільки про самого себе, інформацію про інші сервіси він отримує тільки ту, що йому надає

розробник. Отже, обробка помилок стає додатковою проблемою, які при використанні даної архітектури слід приділяти особливу увагу.

Таким чином, кожний підхід до проектування архітектури веб-додатку має свої переваги і недоліки, які варто враховувати при розробці

В інженерії програмного забезпечення терміни front-end і back-end позначають розділення відповідальності між шаром презентації (front-end) та рівнем доступу до даних (back-end) частини програмного забезпечення або фізичною інфраструктурою або обладнанням. У моделі клієнт-сервер зазвичай клієнтом вважається front-end, а сервер зазвичай вважається back-end, навіть коли деяка презентаційна робота фактично виконується на самому сервері. Тобто, стороною клієнта (або "front-end") є будь-який компонент, яким маніпулює користувач, а код на стороні сервера (або "back-end") зазвичай знаходиться на сервері, часто фізично віддалений від користувача.

Відмітимо основні особливості архітектури front-end [21; 32]:

Веб-додатки більше не прив'язані до ПК (персональних комп'ютерів). В світі з'являється все більше і більше гаджетів, мобільних телефонів, планшетів, IoT – Internet of Things (Інтернет речей). Варто зазначити, що найбільше Інтернет трафіку проходить через мобільні додатки, а не через ПК.

JavaScript, потужна мова для Веб-розробки перетворилася в повноцінну мову програмування, як широко використовується для front-end розробки;

Для веб-додатків стали доступні такі API (Application Program Interface) як File System, Camera, апаратні датчики та інші;

UX (User Experience) стає вирішальним фактором при виборі продукту користувачами;

Стек JavaScript технологій став використовуватися для кроссплатформеної розробки (один і той же веб-додаток може бути запущено що на Android, що на iOS, що в бразуері).

Визначимо основні вимоги до створюваних веб-додатків.

Перш за все, зазначимо, що в багатьох випадках основний зміст сторінок представляє собою набір інформації. Так, наприклад, крім цін, це

може бути інформація стосовно певних юридичних моментів роботи, інформація щодо курсу валют тощо. Загалом кількість інформації, необхідної для формування сторінки достатньо велика, з іншого боку, така інформація потрібна не кожному користувачеві. Звідси випливає перша вимога до розроблюваних додатків, яка полягає в тому, що отриманий веб-портал повинен бути модульним таким чином, щоб користувачі могли отримати тільки те, що їм потрібно.

Інша вимога до програми веб-порталу, також пов'язана з першою вимогою, полягає в тому, що архітектурна структура повинна передбачати розширення програми в майбутньому.

Додавання нових функцій до існуючих систем часто може спричинити конфлікти з існуючою логікою та новими функціями, які швидше за все, містять помилки, що може зламати систему. Створення різних частин програми веб-додатку максимально незалежним може бути головним фактором, оскільки це може полегшити підтримку додатку згодом.

Додаткова довгострокова вимога щодо додатків розширюваність полягає в тому, що треті сторони повинні мати можливість створювати та додавати вміст, який стане частиною цього додатку.

Оскільки технологічна область веб-розробки та рамки JavaScript швидко розвивається змінюючись та розвиваючись, бажано, щоб архітектура веб-порталу була максимально технологічно незалежною. Технологічна незалежність дала б можливість зміни стеку розвитку в майбутньому при створенні нових функцій веб-порталу. Це також може дозволити команді розробників поступово оновлювати існуючі функції, впроваджувати нові технології.

Сучасна архітектура Front-end додатків змушена реагувати на всі ці зміни новими технологіями, фреймворками і підходами. Варто відзначити, що, якщо раніше будь-які технології дуже часто упиралися в обмежені ресурсні можливості пристроїв, то зараз, з огляду на проривних технологій у

виробництві мікросхем, ця проблема зникає, що також впливає на тенденції архітектури веб-додатків.

1.2. Підходи та вимоги до Front-end розробки

Згідно сучасного підходу в інженерії програмного забезпечення структуру front-end, як і в принципі структуру майже будь-якого програмного забезпечення, умовно можна представити у вигляді ієрархічної структури. На верхньому рівні знаходяться блоки, кожен з яких має якусь свою реалізацію, яка дозволяє зв'язуватися з іншими блоками. Всередині цих блоків можна виділити модулі, які складаються з методи. І кожен вузол цієї ієрархії має якийсь свій спосіб спілкування, так звані вхідні і вихідні параметри.

Сучасна архітектура Front-end базується на трьох принципах [21; 23]:

– потік даних займає центральне місце в архітектурі

З огляду на масштаби і складність середовища, в якій може виконуватися front-end додатки, бізнес логіка коду грає дуже важливу роль. За статистикою велика частина часу розробки зазвичай витрачається на налагодження і читання коду, тому необхідною умовою будь-якої архітектури є швидкість знаходження потрібного нам коду, відповідального за бізнес логіку. Всякий раз, коли потрібно внести якусь нову функціональність в систему, надзвичайно важливо розуміти, як вона вплине на майбутню працездатність і ефективність програмного забезпечення.

Єдиний спосіб здійснення такого роду завдання – це розуміння на кожному етапі механізму переміщення даних по архітектури ПЗ. Тому сьогодні будь-який Front-end фреймворк побудований на цьому принципі, на чіткому розподілі на модуль State (зберігання і маніпуляція з даними) і на модуль View (відображення даних).

Наприклад, такий фреймворк як Angular I використовував принцип двонаправленого потоку даних, що створювало неконтрольовані процеси передачі даних. Після появи патерну проектування Flux, який використовував більш ефективний і надійний принцип односпрямованого потоку даних вийшов Angular II, який використовував як раз принципу односпрямованість, як Flux.

– компонентний підхід:

Компонентний підхід є неминучим наслідком першого пункту про потік даних. Можна виділити три головних аспекти компонентного підходу:

– особлива увага потокам даних. Традиційні архітектури frontend додатків орієнтовані на горизонтальний розподіл функціоналу. Однак врахування першого принципу призводить до того, що компонентою може бути будь-який блок коду, що інкапсулює якусь одну "pure responsibility" логіку (функцію з одиничністю відповідальності)

– view (Представлення), які постійно ускладнюються. В даний час, у зв'язку з появою SPA додатків View стають все складніше і складніше, що ускладнює процес впровадження нових характеристик. MV* подібні архітектури не мають можливості вирішити проблему, в той час як використання компонентного підходу дозволяє спрощувати View класи.

– структура компоненти. Будь-якв компонент має складатися з 4 елементів: Графічна структура (HTML-View), Стилізація (CSS-View), Поведінка (JavaScript-Component), бізнес логіка (JavaScript-Component / Model).

Таким чином, компонентний підхід дозволяє перевикористовувати і тестувати окремі блоки веб-додатку, не змінюючи його архітектуру.

Згідно сучасних підходів компоненти повинні бути:

– незалежні;

– слабкозв'язані;

– перевикористовувані.

Компоненти можуть бути досить складні в середині, але мають бути прості зовні. Тут мається на увазі, що інтерфейс будь компоненти повинен бути настільки простим, щоб процес підключення компоненти до батьківського блоку проходив без побічних ефектів.

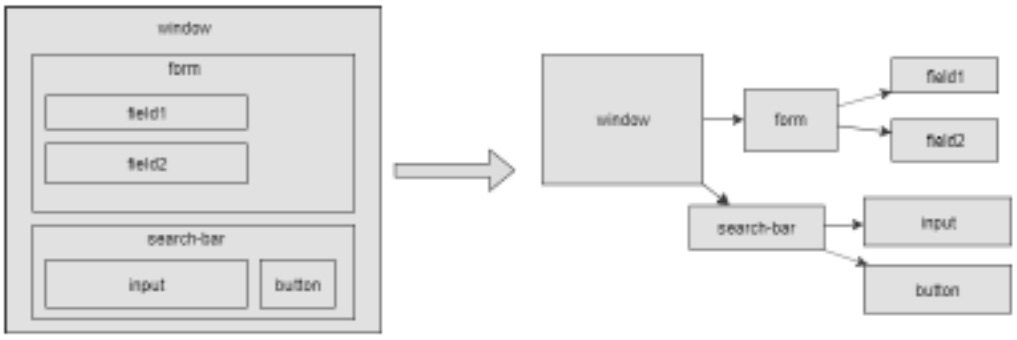

Рисунок 1.2 – Приклад декомпозиції View

– автоматичне вибіркове оновлення DOM

Всі операції з DOM є досить дорогими за витратами ресурсів і часу, тому в якийсь момент Frontend спільнота дійшла до ідеї створити технологію, яка змогла б виконувати з DOM мінімальну можливу кількість дій по перемальовуванні якихось конкретних DOM елементів при зміні даних веб-додатка, так званий State. Результатом таких змін стала поява таких підходів:

– data-binding між Model і View.

– у сукупності з компонентним підходом використання) Data-binding є ідеальним вирішенням проблеми взаємозв'язку даних (Model) і уявлення (View). View можна уявити як функцію моделі, яка знає які поля даних потрібні їй для відтворення тих чи інших компонент. Наприклад, у фреймворку Angular існують шаблони, в React JSX – мова, що є сумішшю HTML і JavaScript мов.

– оповіщення про зміну даних (Change detection).

– однією з вимог також є можливість архітектурного рішень будь-яким чином стежити за тим, які поля State були оновлені, видалені або додані. Дана можливість в Angular I реалізувалася через метод брудної

перевірки (dirty checking), який перевіряв в нескінченному циклі через певний період часу, які поля були змінені. У фреймворку React існують незмінні структури даних і за допомогою подієвої шини відбувається оновлення необхідних полів State. В майбутньому Vue3 планується використання об'єкта Proxy для реалізації data checking (Перевірка даних на зміну):

– оновлення DOM.

– після того як були виявлені зміни State, необхідно виконати перемальовування DOM на основі них. У React для цього використовується технологія VirtualDOM.

– отже, основними вимогами до розробки front-end веб-додатку є [24; 25]:

– розширюваність – здатність системи, яка дозволяє додавати нові функції з мінімальними впливами або зовсім без впливів на його внутрішню структуру та потік даних. Є три найважливіші якості розширюваної програмної системи: модифікованість, ремонтопридатність та масштабованість.

– модифікованість визначається способом поділу функціональності архітектурно таким чином, щоб зміна системи передбачала би найменшу кількість змін в найменшій кількості можливих елементів.

– ремонтопридатність, як визначено Соммервіллом, полягає в розробці системи, яка б враховувала введення нових додаткових вимог без ризику додавання нових помилок

– масштабованість – це здатність системи до розширення у вибраному режимі, без великих змін в її архітектурі. Масштабована система може обробляти більше даних з мінімальним впливом на продуктивність.

– модульність, тобто поділ програмної системи на кілька незалежних модулів, які можуть самостійно виконувати одне чи багато завдань. Ці модулі можуть працювати як основні конструкції для всього програмного забезпечення. Модулі повинні бути розроблені з окремих задач,

щоб їх можна було розробити та / або скласти окремо та незалежно. Переваги модуляції включають покращену ремонтопридатність та функціональність програмного забезпечення модулів, бажаний рівень абстракції, високу згуртованість, багаторазове використання та підвищену безпеку.

– архітектура на основі компонентів. Кожний компонент повинен бути створений для взаємодії з іншими компонентами та їх повторне використання у додатку.

Дані вимоги сформульовані з точки зору розробників і внутрішньої структури веб-додатку. Однак, front-end представляє собою частину програмного забезпечення, яку бачить користувач і через яку він отримує уявлення про роботу додатку, тому до цієї частини ще висуваються окремі так звані представницькі вимоги.

– правильна структуризація веб-сторінки. (X)HTML є основою веб-сайтів, на основі якої здійснюється пошукова оптимізація, тому життєва важливо встановити правильну структуру документа для класів та ідентифікаторів, які забезпечать стиль та взаємодію.

– саме погана семантична структура веб-сторінки досить часто є джерелом багатьох помилок в роботі front-end.

– унікальний візуальний стиль веб-сторінок, який також впливає на структурованість.

– важливим аспектом стилю є перевірка в декількох браузерах та написання стислого, лаконічного коду, який є специфічним, але загальним і одночасно відображає якомога більше рендерів.

– кросплатформеність. Необхідно передбачати можливість того, що користувачі будуть використовувати різні браузери та різні комп’ютери, що може суттєво вплинути на відображення веб-сторінок.

– юзабіліті – характеристика, яка оцінює взаємодію програмного забезпечення з користувачем. Взаємодія додатка з людиною повинна бути зрозумілою, зручною, достатньо простою для сприйняття.

– продуктивність. Для створення веб-додатків, що швидко завантажуються, розмітка, стилі та JavaScript повинні бути масштабованими. Необхідно використовувати скорочувати розміри сторінок там, же це можливо.

Виходячи з цього front-end додатку повинен відповідати таким критеріям:

– у веб-додатку має використовуватися лише попередньо визначена кількість загальних макетів.

– використовувані елементи взаємодії з користувачем повинні бути однаковими для кожного випадку використання.

– розміщення елементів взаємодії з користувачем має бути однаковим, у кожному випадку використання.

– розміри елементів інтерфейсу користувача повинні бути однаковими для кожного випадку використання.

– стан та відгуки про взаємодію користувачів повинні відображатися однаково для кожного випадок використання

– усі використовувані елементи користувача повинні бути реалізовані, використовуючи лише попередньо визначений колір схем.

1.3 Огляд сучасних систем Front-end розробки

Основним набором при розробці front-end є HTML, CSS, JavaScript.

Основою Інтернету є мова розмітки гіпертексту HTML – Hyper Text Markup Language. Вона використовується для логічної (Смисловий) розмітки документа (веб-сторінки). Іноді її неправомірно використовують для управління способом відображення вмісту веб-сторінок на екрані монітора або при виведенні на принтер, але це суттєво порушує вимоги до веб-додатків [19; 20; 30].

Перевагами використання HTML є простота засвоєння та широкі можливості вибору редактора для написання коду.

Для написання досить простої сторінки HTML вистачає, але застосування складного форматування до створення html-документа призводить до значного збільшення розміру підсумкового документа. Тому в таких випадках починають використовувати CSS – Cascading Style Sheets (каскадні таблиці стилів). Це може бути окремий файл, в якому за допомогою спеціальної макромови один раз жорстко проставляється форматування сторінки. Іншими словами, файл CSS виконує роль такого собі шаблону, застосовуваного для форматування тексту, таблиць та інших елементів в документі HTML. Є можливість підключати один і той же фізичний файл CSS до різних web-сторінок сайту. CSS можна використовувати на будь-якому сервері без будь-яких обмежень, оскільки команди CSS виконуються безпосередньо на комп'ютері користувача.

До недоліків даної технології можна тільки віднести відсутність підтримки CSS старими браузерами.

Для додання веб-сторінок динамізму (випадаючі меню, анімація) використовують мови написання скриптів. Такою стандартною мовою в Інтернеті є JavaScript.

JavaScript – це мова програмування, що використовується в складі сторінок HTML для збільшення можливостей. Вона була розроблена фірмою Netscape на базі мови Sun's Java корпорації Sun. JavaScript значно розширює можливості html-документа, створеного з використанням цієї технології. JavaScript інтегрується в файл HTML у вигляді декількох рядків коду (наприклад, це може бути функція, що викликається на виконання спеціальної командою).

Вбудований в браузер інтерпретатор JavaScript сприймає і скрипт, і сам HTML-код як єдиний документ, обробляючи і ті, і інші дані одночасно.

Використання технології JavaScript не вимагає установки і настройки на сервері будь-яких додаткових модулів, оскільки скрипти виконуються безпосередньо на комп'ютері користувача.

HTML, CSS, JavaScript є лінгвістичним забезпечення, в той час як в браузерах документи подаються у вигляді набору об'єктів, множинач типів яких є об'єктною моделлю браузера (BOM). Об'єктна модель браузера унікальна для кожної моделі і таким чином виникають проблеми при створенні міжбраузерних додатків. Тому Веб-консорціум запропонував об'єктну модель документа (DOM), що є стандартним способом представлення веб-сторінок за допомогою набору об'єктів.

На відміну від об'єктної моделі браузера DOM містить набір об'єктів лише для вмісту документа і не має об'єктів, що дозволяють управляти вікнами і рамками вікон.

Модель об'єкта документа (DOM) визначається як інтерфейс програмування для документів HTML та XML. Вона інтерпретує сторінку так, що програми можуть змінювати структуру, стиль та вміст документа. DOM надає документ у вигляді вузлів та об'єктів, що дозволяє мовам програмування підключатися до сторінки (рис. 1.3).

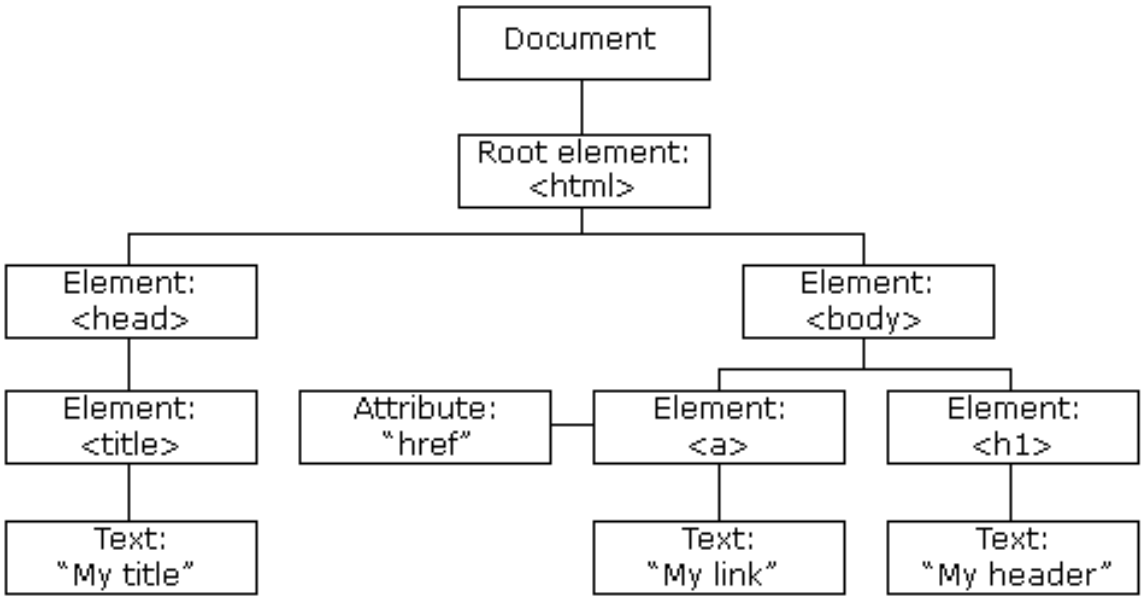

Рисунок 1.3 – Дерево об'єктів HTML DOM

Спочатку різні браузери мали власні моделі DOM, не сумісні з іншими. Для того, щоб забезпечити взаємну і зворотну сумісність, консорціум W3C класифікував цю модель за рівнями, для кожного з яких була створена своя специфікація. Всі ці специфікації об'єднані в загальну групу, що носить назву W3C DOM:

Рівень 0. Включає в себе всі специфічні моделі DOM, які існували до появи Рівня 1, наприклад document.images, document.forms. Ці моделі формально не є специфікаціями DOM, опублікованими консорціумом W3C, а скоріше відображають те, що існувало до початку процесу стандартизації.

Рівень 1. Базові функціональні можливості DOM (HTML і XML) в документах, такі як отримання дерева вузлів документа, можливість змінювати і додавати дані.

Рівень 2. Підтримка простору імен XML, filtered views та подій.

Рівень 3. Складається з шести різних специфікацій:

DOM Level 3 Core;

DOM Level 3 Load and Save;

DOM Level 3 XPath;

DOM Level 3 Views and Formatting;

Level 3 Requirements;

DOM Level 3 Validation.

Поточним рівнем специфікацій DOM є Рівень 2, але, тим не менш, деякі частини специфікацій Рівня 3 є рекомендованими W3C.

DOM ставить кожному елементу або об'єкту, визначеному за допомогою атрибута ID (ідентифікатора об'єкта), функцію JavaScript. А за допомогою таких функцій можна управляти властивостями атрибутів об'єкта, заданих через CSS.

Атрибут ID підтримується всіма видимими HTML-тегуми. Його значення – це унікальне власне ім'я елемента на сторінці, тобто на сторінці не повинно бути декількох елементів з одним ID, на відміну від елемента class,

який є вільним ознакою для декількох елементів. ID – основне поняття динамічного HTML (DHTML). З його допомогою JavaScript визначає унікальний об'єкт, і яким можна керувати за допомогою JavaScript.

DHTML (динамічний HTML) – це комерційний термін, придуманий для опису технологій, які були введені в четвертій версії Web-браузерів і дозволяли обходити обмеження HTML.

DHTML являє собою комбінацію Web-стандартів:

CSS + JavaScript + DOM + xHTML = DHTML

де:

CSS – визначає атрибути об'єктів,

JavaScript – змінює об'єкти,

DOM – знаходить об'єкти,

XHTML – створює об'єкти (виконує розмітку тексту).

Створена на основі DHTML сторінка може змінюватися без звернення до сервера за додатковими даними, тобто представляє собою клієнтський код (client-side-code), який в свою чергу, дозволяє істотно збільшити інтерактивність Web-сторінок.

Переваги DHTML:

– підтримується всіма браузерами.

– використовує стандартні технології.

– можна вносити зміни в Web-сторінку після її завантаження.

– текстові файли DHTML завантажуються швидше, ніж Flash і Java.

– не вимагає додаткових модулів.

– простий у вивченні.

– висока швидкість розробки Web-сторінок.

Недоліки DHTML:

– різний вигляд Web-сторінок через несумісність браузерів і операційних систем.

– javaScript і CSS вельми чутливі до помилок в синтаксисі.

– ненадійна робота в зв'язку з помилками в браузерах.

Ще одним засобом для створення front-end є фреймворки та бібліотеки.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.

Веб-фреймворк – інструмент, який полегшує процес написання і запуску веб-додатків.

Фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

Одна з головних переваг при використанні фреймворків полягає в тому, що такі програми мають стандартну структуру.

Бібліотеки – це набори попередньо написаних фрагментів коду, які використовуються та використовуються для реалізації основних особливостей JavaScript. Фрагмент можна легко інтегрувати в існуючий код проекту за необхідності.

Отже, бібліотеки – це спеціалізований інструмент для конкретних потреб кодування, а не універсальна машина для створення всього існуючого проекту.

Angular – це фреймворк від Google, він сумісний з більшістю поширених редакторів коду. призначений для створення динамічних, односторінкових веб-додатків (SPA) та прогресивних веб-додатків. Після свого первинного випуску даний фрейворк отримав найвищу оцінку за свою здатність перетворювати документи на основі HTML в динамічний контент.

Vue.js – це ще одна структура з відкритим кодом для SPA. Він використовує модель розробки на основі компонентів і дозволяє приєднувати компоненти до проекту. Vue.js – це приклад бібліотеки, яка є скоріше фреймворком. Для використання Vue.js знання HTML та CSS – абсолютна

вимога. Він пропонує велику кількість шаблонів та шаблонів проектування. Vue в першу чергу визнається невеликим розміром документів та синтаксисом на основі HTML.

Ember.js є основою для SPA, мобільних та настільних додатків. Він використовує шаблон model-view-view-model (MVVM). У загальному розумінні це JS-каркас, який працює за MVC-шаблоном розподілу коду. При цьому Ember.js легко інтегрується і може працювати з бібліотеками Handlebars і jQuery.

Bootstrap Framework, який є вільним набором інструментів для створення сайтів і веб-додатків. Він включає в себе HTML і CSS шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу,включаючи JavaScript-розширення.

Bootstrap використовує найсучасніші напрацювання в області CSS і HTML, тому часто необхідно бути уважним при підтримці старих браузерів які не підтримують деякі функції [30].

Основною перевагою Bootstrap є економія часу при розробці, оскільки він дає змогу використовувати шаблони і класи дизайну. Масштабування в динамічних макетах Bootstrap відбувається для різних пристроїв та роздільних здатностей екрану без будь-яких змін в розмітці. Це скорочує час оптимізації проекту.

Для всіх компонентів використовується єдиний стиль і шаблони за допомогою центральної бібліотеки. Дизайн і макети веб-сторінок узгоджуються один з одним.

Крім того, Bootstrap сумісний з усіма основними браузерами і коректно відображається на останніх версіях веб-браузерів: Mozilla Firefox, Google Chrome, Safari, Internet Explorer, Microsoft Edge і Opera.

Особливістю Twitter Bootstrap є відкритий вихідних код і безкоштовне розповсюдження.

React – це бібліотека з відкритим кодом для створення динамічних інтерфейсів користувача, створена та розроблена Facebook. Вона

застосовується для створення веб-додатків з декількома динамічними компонентами. Він в його основі лежить JavaScript та JSX, розширення мови PHP у Facebook. React дозволяє будувати багаторазові HTML-елементи для front-end. До React також входить React Native, спеціалізована міжплатформена система мобільного розвитку.

React характеризується простотою в освоєнні, лаконічністю синтаксису, можливістю створення і використання VirtualDOM, за допомогою якого розвантажуються високонавантажені додатки. За допомогою React розробники створюють окремі компоненти, здатні до переносу з одного проекту в інший.

jQuery, у свою чергу, спрямований на контроль HTML-документів. Він має простий API для управління подіями та дизайном анімації в браузерах. Окрім цього, jQuery застосовується для маніпулювання DOM і також служить інструментом розробки плагінів. Він також оснащений більш легкою бібліотекою крос-браузерів, інтерфейсом jQuery для мобільної основи jQuery Mobile та для побудови графічного інтерфейсу.

Даний фреймворк дозволяє ефективно і зручно працювати з будь-яким з елементів DOM, подіями, використовувати технологію Ajax, створювати складні візуальні ефекти і завжди мати під рукою величезну кількість JS-плагінів для створення користувацьких інтерфейсів дозволяє JavaScript-бібліотека jQuery. За допомогою даного фреймворка веб-розробникам вдається надати сайту динамічність.

D3.js - бібліотека, керована даними для візуалізації даних. Прив’язуючи тимчасові дані до DOM та здійснюючи керовані даними зміни в документі, бібліотека дозволяє керувати даними та робити динамічні візуалізації даних. Він може підтримувати та обробляти великі набори даних та динамічні відповіді на взаємодію та анімацію. Функціональний стиль D3 дозволяє повторно використовувати код і працювати з CSV та HTML.

Як ми бачимо, великі бібліотеки JavaScript мають широку функціональність і є єдиним рішенням для розробників Front-end.

2. ПОБУДОВА ІНТЕГРАЛЬНОЇ ОЦІНКИ СИСТЕМИ FRONT-END РОЗРОБКИ

2.1. Методи оцінювання програмного забезпечення

Проблема оцінювання програмного забезпечення з’явилася досить давно. Вважається, що результатом такого оцінювання повинен стати інтегральний показник якості програмного забезпечення.

Під якістю програмного забезпечення розуміють сукупність властивостей ПЗ, що обумовлюють його придатність задовольняти певні потреби користувачів і фахівців, що беруть участь у створенні і супроводі ПЗ.

Під властивістю (характеристикою) ПЗ розуміють об'єктивну особливість ПЗ (програм і документації), яка виявляється при його розробці, експлуатації та супроводі.

Для об'єктивної оцінки якості ПЗ його властивості необхідно охарактеризувати кількісно. Показник якості ПЗ - кількісна характеристика властивості ПЗ, що входить до складу його якості і розглядається відповідно до певних умов його створення, експлуатації та супроводу. Поряд з показниками якості можуть використовуватися якісні (словесні) оцінки, звані ознаками [22].

Показники якості за кількістю характеризуються властивостей можуть бути одиничними і комплексними (груповими). Одиничний показник відноситься тільки до однієї з властивостей, тоді як комплексний характеризує кілька властивостей ПЗ.

Результатом оцінювання на виході може бути загальна інтегральна оцінка.

Основою для оцінки якості програмного забезпечення виступають міжнародні стандарти:

Якість програмного забезпечення – це ступінь, в якій програмне забезпечення має необхідну комбінацію властивостей (1061-1998 IEEE Standard for Software Quality Metrics Methodology);

Якість програмного забезпечення - це сукупність характеристик програмного забезпечення, що відносяться до його здатності задовольняти встановлені і передбачувані потреби (ISO 8402: 1994 Quality management and quality assurance);

Зараз більшість розробників використовують багаторівневу модель визначення якості програмного забезпечення згідно набору стандартів ISO 9126 [3]. На верхньому його рівні виділено 6 основних характеристик якості програмного забезпечення, кожну з яких можна визначити набором атрибутів, що мають відповідні метрики (рис.2.1).

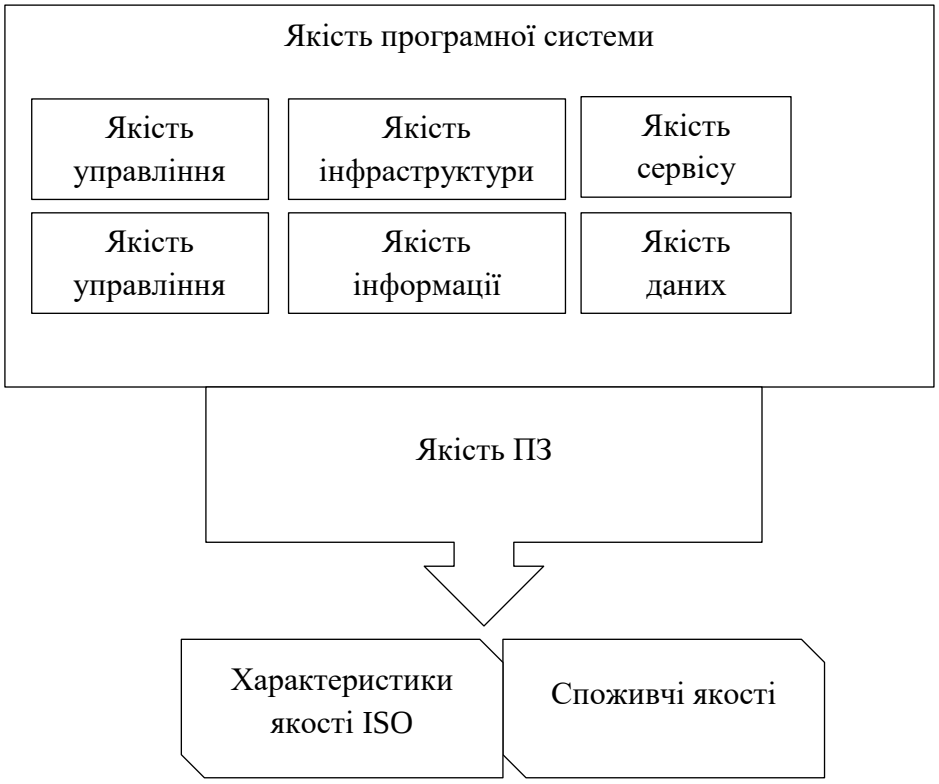

Рисунок 2.1 – Узагальнена модель оцінки якості програмного забезпечення

На цій узагальненій моделі будуються інші моделі оцінки якості ПЗю Ці моделі основуються на критеріях якості та пов’язаних з ними показниках (метриках).

Їх можна розділити на такі види:

– до першого виду можна віднести теоретичні моделі, засновані на гіпотезі відносин між змінними якості;

– до другого виду відносяться моделі "управління даними", засновані на статистичному аналізі;

– до третього виду належить комбіновані моделі, в якій інтуїція дослідника використовується для визначення потрібного виду моделі, а аналіз даних застосовується для визначення констант моделі якості.

Перша модель для оцінки якості програмного забезпечення була запропонована в роботах таких дослідників, як J.A. McCall, P.K. Richards, G.F. Walters [10-12]. Її основне призначення полягало в отриманні повної оцінки якості програмного забезпечення через його різні характеристики. У моделі МакКолла є три основні характеристики для оцінки якості програмного забезпечення:

– використання (метрики – коректність, надійність, ефективність, цілісність і практичність);

– модифікація (метрики – тестованість, гнучкість, а також супроводжуваність, що є важливим для якості нової розроблюваної версії програмного забезпечення);

– переносимість (метрики – мобільність, можливість багаторазового використання, функціональна сумісність, тобто ті чинники якості, які важливі для перенесення програмного забезпечення на інші апаратні і програмні платформи).

Другою моделлю оцінки якості програмного забезпечення є модель Б. Боема [2], яка дозволяє більш точно визначати якість основних характеристик програмного забезпечення, заданих набором показників і метрик. Модель якості Б. Боема представляє характеристики програмного

забезпечення в більшому масштабі, ніж модель МакКолла, оскільки в її основі лежить оцінка якості програмного забезпечення на всіх етапах його життєвого циклу.

Модель Б. Боема як і модель МакКолла відноситься до ієрархічних моделей якості, структурування показників в яких здійснюється спочатку на основі високорівневих, потім проміжних і, нарешті, індивідуальних характеристик.

У цій моделі:

– практичність показує, наскільки легко, надійно і ефективно програмне забезпечення може бути використано для вирішення конкретних завдань;

– супроводжуваність характеризує наявність можливості змінити і повторно протестувати програмне забезпечення;

– мобільність визначає можливості використання програмного забезпечення на різних програмних засобах і апаратних платформах.

До недоліків даної моделі відноситься те, що вона не завжди точно визначає характеристики, оскільки є автоматичною.

За аналогією з моделями МакКолла і Б. Боема для оцінки якості програмного забезпечення R.B. Grady і D.L. Caswell запропонували використовувати FURPS модель [7]. Її відмінність від попередніх моделей полягає в наявності двох шарів показників якості. При цьому перший шар визначає тільки основні якісні характеристики показників, а в другому шарі представлені пов'язані з ними атрибути.

Сучасною модифікацією цієї моделі стала модель FURPS +.

Модель FURPS / FURPS + отримала свою назву за першими літерами основних категорій показників якості програмного забезпечення:

– functionality – функціональність, яка включає в себе в якості додаткових показників особливості, можливості і безпеку використовуваного програмного забезпечення в інноваційних програмних проектах;

– usability – практичність, яка включає в себе в якості додаткових показників можливість обліку людського фактора, ергономічність і наявність повного комплекту користувальницької документації (інструкції, правила інсталяції та деінсталяції і ін.);

– reliability – надійність, яка включає в себе в якості додаткових показників частоту відмов у роботі програмного забезпечення, наявність можливостей для відновлення інформації, а також прогнозованість дій користувача в найближчій перспективі;

– performance – продуктивність, яка включає в себе в якості додаткових показників час відгуку, пропускну здатність обробки певних обсягів інформації, точність одержуваних рішень, простоту і доступність програмного забезпечення для користувачів, використання всіх можливостей інформаційних ресурсів обробки даних;

– supportability – експлуатаційна придатність, яка включає в себе в якості додаткових показників можливості тестування програмного забезпечення та його окремих компонентів (модулів), його здатності до розширення до більш скоєних версій, рівень адаптованості використовуваного програмного забезпечення, необхідність організації супроводу в процесі експлуатації, можливості сумісності використовуваного програмного забезпечення з іншими інформаційними програмами і платформами, можливості зміни конфігурації, простоту обслуговування в процесі експлуатації, стандартні вимоги до установки, а також можливості локального функціонування.

Символ «+» забезпечує розширення FURPS моделі за рахунок додавання до неї:

– обмежень проекту з інформаційних ресурсів, вимогам до мов і засобів розробки, вимог до апаратного забезпечення;

– обмежень інтерфейсу інноваційних програмних проектів, що накладаються на взаємодію із зовнішніми системами;

– функціональних вимог до виконання заявленого в специфікації набору операцій;

– відповідності ергономічним вимогам, забезпечення виконання необхідних вимог по ліцензуванню інформаційного проекту.

Концептуальною основою моделі якості FURPS / FURPS + є декомпозиція характеристик програмного забезпечення в розрізі двох категорій: функціональної (F) та нефункціональної (URPS).

Такий підхід до розробки зазначених моделей дозволяє використовувати ці категорії у вигляді вимог для оцінки якості програмного забезпечення, а також у вигляді основних показників при оцінці якості програмних систем.

Використання даної моделі пояснюється тим, що в ній міститься найбільш універсальний перелік характеристик для оцінки якості програмного забезпечення.

У моделі К. Гецці і його співавторів використовуються різні підходи до визначення якості програмного забезпечення і процесу його експлуатації в складі програмного проекту [6]. Відповідно до цієї моделі якість програмного забезпечення описується за допомогою таких показників:

– цілісність;
– надійність;
– стійкість;
– продуктивність;
– верифікованість;
– супроводжуваність;
– практичність;
– можливість багаторазового використання;
– мобільність;
– зрозумілість;
– можливість взаємодії;

– ефективність;

– своєчасність реакції на дії користувача.

В основу моделі G.R. Dromey [4] покладені критерії оцінки характеристик якості і їх під характеристик. Головне призначення цієї моделі полягає в оцінці якості програмного проекту в цілому як інформаційної системи з урахуванням тих обставин, що оцінка якості кожного такого програмного проекту в силу відомих причин буде відрізнятися від оцінок якості інших інноваційних програмних проектів.

Дану модель доцільно використовувати для виявлення можливої наявності помилок в програмному забезпеченні і пошуку тих його складових, в яких потенційно можлива поява помилок. Використовуючи відносини між характеристиками і під характеристиками якості, ця модель дозволяє отримати оцінки якості окремих властивостей і характеристик як програмного забезпечення і його складових, так і інноваційного програмного проекту в цілому.

В Центрі забезпечення якості програмного забезпечення NASA використовується модель метрик SATC (Software Assurance Technology Center). Принципова відмінність цієї моделі від всіх розглянутих раніше полягає в тому, що в ній спочатку оцінки якості визначаються для кожної складової інноваційного програмного проекту окремо. При цьому спочатку оцінюється якість розробки вимог специфікацій, програмного забезпечення і його складових, документації на інформаційний проект, тестування його складових, а також виконання операцій. Далі на основі отриманих оцінок формується інтегральний показник якості розробленого інноваційного програмного проекту. Завдяки такому підходу ця модель забезпечує не тільки оцінку ризиків інформаційного проекту, але і ефективність виконуваних операцій, а значить і якість розробленого інноваційного програмного проекту. Для цього в моделі якості SATC формується набір цілей, які пов'язані з інноваційним програмним проектом і атрибутами

виконуваних ним операцій відповідно до структури моделі якості програмного забезпечення, розробленої в рамках стандарту ISO 9126 [11].

Ієрархічну модель оцінки якості об'єктно-орієнтованого проектування інформаційних проектів (QMOOD) запропонували J. Bansiya і C. Davis [15]. Ця модель істотно розширює методологію оцінки якості, використовувану в моделі G.R. Dromey. Вона ґрунтується на чотирьох рівнях оцінки якості інноваційного програмного проекту:

– на показниках якості, які представлені у вигляді набору якісних атрибутів проекту і використовуються для опису його основних характеристик:

– функціональності;

– ефективності;

– зрозумілості;

– розширюваності;

– можливості багаторазового використання;

– гнучкості;

– на об'єктно-орієнтованих властивостях інноваційного програмного проекту, які можуть бути визначені в ході оцінки якості його структури (внутрішньої і зовнішньої), функціональності окремих складових і атрибутів. У QMOOD як структурної об'єктно-орієнтованої множини зазвичай використовуються:

– розмір і ієрархічна структура;

– пов'язаність складових частин;

– поліморфізм;

– складність обміну інформацією;

– на об'єктно-орієнтованих метриках інноваційного програмного проекту, які призначені для оцінки його якості;

– на об'єктно-орієнтованих властивостях інноваційного програмного проекту, які використовуються для формування його архітектури.

У 2003 р L. Bass, P. Clements, R. Kazman представили модель оцінки інноваційних програмних проектів, засновану на двох різних підходах до вибору показників для оцінки його якості, з огляду на тривалість усього життєвого циклу використовуваного в ньому програмного забезпечення [1]. При цьому автори моделі виділили дві групи основних характеристик якості:

– ефективність, безпеку, доступність і функціональність;

– модифікованість, мобільність, можливість багаторазового використання, успадкованість і тестованість.

У 2005 р дослідники K. Khosravi і Y. Gueheneuc запропонували модель оцінки якості програмного забезпечення, засновану на багаторазовому використанні однієї з глобальних характеристик якості [17]. Крім того, вона допускає можливість багаторазового використання таких характеристик, як зрозумілість, гнучкість, модульність, надійність, масштабованість і зручність використання.

Оцінка якості в цій моделі полягає в послідовному розв'язанні двох задач:

– у виборі глобальної характеристики;

– у виборі підхарактеристик, пов'язаних з глобальною характеристикою.

У цій моделі для зв'язку показників якості та їх підхарактеристик використовувалися базові визначення стандартів IEEE, ISO / IEC і ряду інших моделей оцінки якості.

У 2008 р дослідники C. Chang, C. Wu, H. Lin запропонували підхід для оцінки якості програмного забезпечення на основі використання методології нечітких множин і методу аналізу ієрархій [3]. Основні принципи цього підходу були ними реалізовані в рамках моделі оцінки якості стандарту ISO 9126. Зокрема, ними використовувалися характеристики і підхарактеристики основних показників моделі оцінки якості програмного забезпечення стандарту ISO 9126.

У тому ж році вчені A. Sharma, R. Kumar і P.S. Grover [14] запропонували компонентно-орієнтовану модель для оцінки якості програмного забезпечення, в яку були включені не тільки все характеристики і під характеристики моделі оцінки якості стандарту ISO 9126, а й запропоновано цілий ряд нових під характеристик, до яких можна віднести придатність до повторного використання, гнучкість, складність, масштабованість. Для оцінки якості інноваційних програмних проектів в цій моделі запропоновано використовувати добре відомий метод аналізу ієрархій.

Проведений аналіз моделей оцінювання якості програмного забезпечення дав змогу сформувати узагальнений алгоритм функціонування таких моделей (на основі ISO/IEC 9126-1:2001. Software Engineering – Software Product Quality – Part 1: Quality Model), який зображений на рис. 2.2.

Рисунок 2.2 – Узагальнений алгоритм функціонування моделі оцінювання якості програмного забезпечення

2.2. Визначення критеріїв оцінки системи Front-end розробки

В основі узагальненого алгоритму моделі оцінювання якості програмного забезпечення лежить визначення показників, які використовуються в кожній окремій моделі.

Загалом методи визначення показників якості ПЗ можна класифікувати:

– за способами отримання інформації про ПЗ - вимірювальний, реєстраційний, органолептичний, розрахунковий;

– за джерелами отримання інформації – традиційний, експертний, соціологічний.

Вимірювальний метод ґрунтується на отриманні інформації про властивості та характеристики ПЗ з використанням інструментальних засобів. Наприклад, з використанням цього методу визначається обсяг ПЗ - число рядків вихідного тексту програм і число рядків - коментарів, число операторів і операндів, число виконаних операторів, число гілок в програмі, число точок входу (виходу), час виконання гілки програми, час реакції і інші показники [18, 25].

Реєстраційний метод ґрунтується на отриманні інформації під час випробувань або функціонування ПЗ, коли реєструються і підраховуються певні події, наприклад, час і число збоїв і відмов, час передачі управління інших модулів, час початку і закінчення роботи.

Органолептичний метод ґрунтується на використанні інформації, одержуваної в результаті аналізу сприйняття органів почуттів (зору, слуху), і застосовується для визначення таких показників як зручність застосування, ефективність тощо.

Розрахунковий метод ґрунтується на використанні теоретичних і емпіричних залежностей (на ранніх етапах розробки), статистичних даних, що накопичуються при випробуваннях, експлуатації та супроводі ПЗ. За

допомогою розрахункового методу визначаються тривалість і точність обчислень, час реакції, необхідні ресурси.

Експертний метод використовується у випадках, коли завдання не може бути вирішена ніяким іншим з існуючих способів, або інші способи є значно більш трудомісткими. Експертний метод рекомендується застосовувати при визначенні показників наочності, повноти і доступності програмної документації, легкості освоєння, структурності. Визначення значень показників якості ПЗ експертним методом здійснюється групою експертів-фахівців, компетентних у вирішенні даного завдання, на базі їх досвіду і інтуїції.

Соціологічні методи використовують спеціальні анкети-опитування.

Як вже відмічалося в попередньому розділі, відмінною рисою будь-якого веб-додатка є наявність front-end, який відіграє суттєву роль в загальній оцінці додатку. До того ж для більшості користувачів саме front-end виступає основним структурним елементом, за яким вони оцінюють весь програмний комплекс

Тому для саме для Front-end зараз є важливим, ніж будь-коли, вимірювання та оцінювання якості.

Вихідними даними і вищим пріоритетом при виборі показників якості в більшості випадків є призначення, функції та функціональна придатність відповідного програмного засобу.

Процеси вибору і встановлення метрик і шкал для опису характеристик якості програмних засобів можна розділити на два етапи [27]:

– вибір і обґрунтування набору вихідних даних, що відображають загальні особливості і етапи життєвого циклу проекту програмного продукту і його споживачів, кожен з яких впливає на певні характеристики якості комплексу програм;

– вибір, встановлення конкретних метрик і шкал вимірювання характеристик і атрибутів якості проекту для їх подальшої оцінки і

зіставлення з вимогами специфікацій в процесі кваліфікаційних випробувань або сертифікації на певних етапах життєвого циклу програмного засобу.

Будь-який веб-додаток складається з клієнтської, серверної та інформаційної (контент) частин. Для кожної з них можна розробити свою модель оцінювання якості, де одні і ті ж характеристики будуть мати різні ваги або взагалі будуть відсутні.

Визначимо, які характеристики якості є важливими для Front-end веб-додатку.

Як зазначалось в попередньому розділі модель якості програмного продукту в основному включає такі характеристики – функціональна придатність, ефективність функціонування, сумісність, зручність використання (практичність), надійність, захищеність, супроводжуваність і мобільність.

Зараз інтерфейс веб-додатків складніший, ніж будь-коли. Середня вага веб-сторінка перевищує 3 Мб і викликає понад 100 ресурсів на кожен клік користувача. Крім того відбувається постійне оновлення контенту, до додатку звертається велика кількість користувачів, що заходять на веб-сторінки з різних браузерів, пристроїв, від різних Інтернет-провайдерів та локацій.

Характеристики, які можна виключити з моделі якості клієнтської частини веб-додатків – це ефективність функціонування, захищеність і мобільність, оскільки вони можуть використовуватися для оцінки якості ПЗ серверної частини веб-додатка.

Таким чином, до основних характеристик якості клієнтської частини веб-додатків можуть бути віднесені функціональна придатність, сумісність, зручність використання, надійність і супроводженість.

Придатність є основною характеристикою і не може бути виключена. Так, наприклад, на основі тестування функціональної придатності визначається готовність продукту до передачі замовнику.

Продуктивність також є важливою для клієнтської частини, оскільки вона визначає швидкість завантаження веб-сторінок, час обробки запитів користувачів.

Зручність використання є однією з основних характеристик якості продукту з точки зору користувачів. Тому дана характеристика також є важливою для клієнтської частини.

Надійність клієнтської частини важлива в контексті її завершеності. Оцінка завершеності заснована на підрахунку кількості помилок в коді клієнтської частини веб-додатки.

Супроводженість також важлива для клієнтської частини. Прикладом цього є шаблон MVC (model-view-controller), який вимагає відділення компонент, що відповідають за генерацію веб-сторінок, від вмісту веб-сторінок. Тоді особа, що розробляє клієнтську частину, може вносити зміни, які не впливають на серверну частину.

З урахуванням вищесказаного, пропонується модель якості front-end веб-додатків, яка наведена на рис. 2.3.

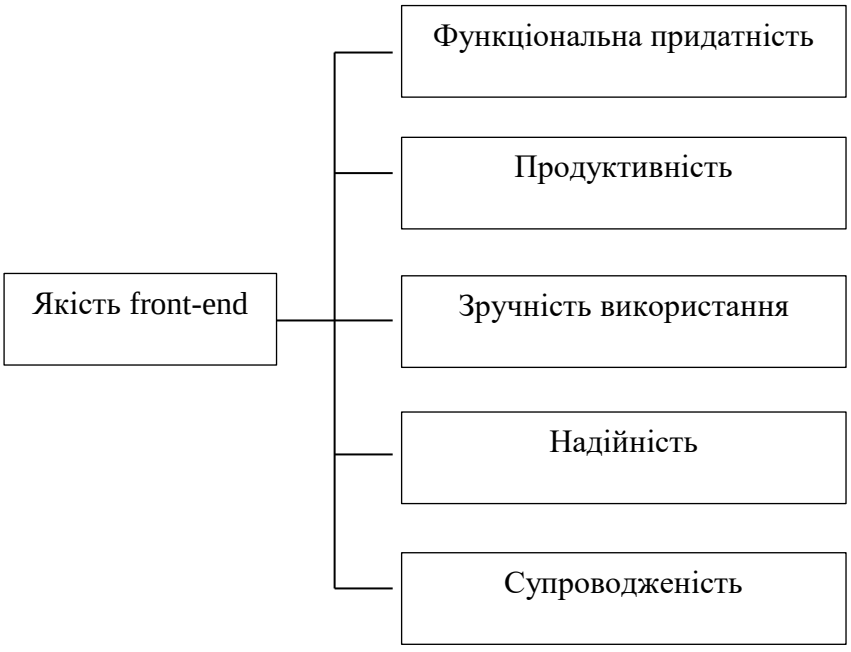

Рисунок 2.3 – Характеристики якості Front-end

На основі сформованої моделі якості та з врахуванням особливостей Front-end розробки визначимо нижні рівні системи показників.

1. Функціональність;

1.1. навігація та перегляд інформації;

1.1.1. головна навігація (головне меню);

1.1.2. контекстне меню;

1.2. пошук та отримання інформації;

1.2.1. глобальний пошук;

1.2.2. тематичний пошук;

1.3. керування інформаційним наповненням;

1.3.2. редагування інформації;

1.3.3. редагування сторінок та розділів сайту;

1.3.4. редагування головного меню;

1.3.5. редагування контекстного меню;

1.3.6. редагування стилів;

1.4. точність;

1.4.1. точність подання інформації;

1.5. захищеність;

1.5.1. захищеність доступу до системи керування інформаційним наповненням;

1.5.2. захищеність доступу до сховища даних;

1.5.3. захищеність доступу до веб-сервера;

2. Надійність;

2.1. завершеність;

2.1.1. відсоток ізольованих сторінок;

2.1.2. відсоток ушкоджених посилань (неіснуючих, недоступних) ;

2.1.5. невідповідності і неточності в різних Web-браузерах;

2.1.7. наявність орфографічних та пунктуаційних помилок;

2.2. стійкість до відмов;

2.2.1. наявність помилок при виконанні;

2.2.2. наробіток на відмову;

2.3. здатність до відновлення;

2.3.1. відновлення працездатності після збою;

2.4. узгодженість (відповідність стандартам) ;

2.4.1. відповідність коду вимогам W3C;

2.4.2. відповідність стилів CSS вимогам W3C;

3. зручність використання;

3.1. зрозумілість;

3.1.1. зрозумілість загальної структури сайту;

3.1.2. карта сайту;

3.2. зручність навчання;

3.2.1. наявність та якість довідки та загальної підтримки;

3.2.2. наявність та якість довідки та підтримки тематичної;

3.2.3. питання, що часто задають;

3.2.4. віртуальний тур;

3.3. зручність роботи;

3.3.1. мітка поточної позиції;

3.3.2. зручність і наочність навігації;

3.3.3. зручність та наочність подання інформації при різних розмірах екрану;

3.4. зворотний зв'язок;

3.4.1. гостьова книга;

3.4.2. опитування;

3.4.3. форум;

3.4.4. коментарі та пропозиції;

3.5. привабливість (інтерфейс та естетичні особливості);

3.5.1. однорідність стильового оформлення;

3.5.2. групування головних елементів управління;

3.5.3. однорідність кольорового оформлення посилань;

3.5.4. сталість елементів керування;

3.5.5. підтримка різних мов;

4. Продуктивність (ефективність) ;

4.1. часова ефективність;

4.1.1. швидкість завантаження (розмір) статичних і динамічних сторінок;

4.1.2. час відгуку (виконання запитів) ;

4.2. доступність;

4.2.1. зручність при відключенні зображень у браузері;

4.2.2. заголовки в зображеннях;

4.2.3. спливаючі підказки;

4.3. ефективність використання ресурсів;

4.3.1. використання оперативної пам'яті;

4.3.2. використання дискового простору;

5. зручність супроводу (супроводженість) ;

5.1. аналізованість;

5.1.1. ведення лог-файлів;

5.1.2. ведення інших інформаційних протоколів та журналів;

5.1.3. зручність читання лог-файлів;

5.1.4. інформативність повідомлень, що виводяться при збої;

5.2. стабільність;

5.2.1. програмна обробка помилок;

5.2.2. збереження працездатності після відмови при внесенні змін;

5.4. тестованість;

5.4.1. тестованість модулів;

5.4.2. тестованість web-интерфейсу.

Запропоновані показники в цілому характеризують Front-end з двох сторін: зі сторони розробників, так і зі сторони користувачів. Таким чином, на основі цих показників можна отримати достатню повну характеристику front-end веб-додатку.

2.3. Розробка методики інтегрального оцінювання систем Front-end розробки

Показники якості об'єднані в ієрархічну систему з чотирьох рівнів. кожен вищий рівень містить в якості складових показники нижчих рівнів. Допускається вводити додаткові показники на кожному з рівнів.

Для отримання інтегральної оцінки за групами показників якості використовують фактори якості (1-й рівень): функціональність, надійність, зручність використання, продуктивність, супроводженість.

Кожному фактору якості відповідає певний набір критеріїв якості (комплексні показники 2-го рівня). Критерії якості визначаються однією або декількома метриками (3-й рівень). Метрики складаються з оціночних елементів (одиничних показників – 4-й рівень), що визначають задану в метриці властивість. Число оціночних елементів, що входять в метрику не обмежена і може коливатися в різну сторону.

В процесі оцінки якості front-end на кожному рівні (крім рівня оціночних елементів) здійснюється обчислення показників якості ПЗ, тобто визначення кількісних значень абсолютних показників (P_{ij}), де j – порядковий номер показника даного рівня для і- го показника вищого рівня).

Показники якості, які були визначені для оцінювання Front-end розробки мають як числовий характер (наприклад, швидкість завантаження сторінки, кількість використовуваної оперативної пам’яті тощо), так логічний або оцінювальний характер (наприклад, тестованість модулів, наявність різних мов тощо). Для того, щоб привести всі вхідні ознаки до однакового вузького діапазону пропонується використовувати таку непряму шкалу

$$h(X)=\begin{cases}1, & якщо\ X=0\\ \dfrac{X_{\max}-X}{X_{\max}}, & якщо 0\le X\le X_{\max}\\ 0, & якщо\ X\ge X_{\max}\end{cases} \qquad (2.1)$$

Використання елементарних функцій дозволяє у відсотковому відношенні інтерпретувати якість задоволення вимог відповідних атрибутів. При цьому, згідно з [9], шкали та метрики нормуються. Для забезпечення об’єктивності процесу оцінювання, шкала ранжується на три рівні прийнятності:

– 0 – 39% – незадовільний рівень;

– 40% – 59% – граничний рівень;

– 60% – 100% – задовільний рівень.

Це дозволить звести разом в одній моделі різні за природою змінні і забезпечить коректну роботу обчислювального алгоритму

Для вимірювання показників пропонується використовувати непряму шкалу, яка може враховувати ступінь задоволення відпо

Існує декілька способів побудови інтегральної оцінки [8, 9].

Метод сум, тобто сумування всіх фактичних значень показників.

$$Ip_i=\sum_{j=1}^{n}\frac{x_{ij}^{ф}}{x_{ij}^{б}} \qquad (2.2)$$

де $x_{ij}^{ф}, x_{ij}^{б}$ – відповідно фактичне і базисне значення j-го показника для і-го об’єкта

Метод геометричних середніх, який використовується для оцінки показників, значення y_{ij} яких знаходиться в межах від 0 до 1:

$$Ip_i = \left[\prod_{j=1}^{n} y_{ij} \right]^{\frac{1}{n}} \quad (2.3)$$

Метод суми місць передбачає попереднє ранжування всіх об'єктів за окремими показниками. Кожному показнику відповідає новий параметр, який визначає місце кожного серед інших по і-му показнику.

Метод відстаней, за допомогою якого розраховується відстань всіх об’єктів до певного еталону

Однак, в усіх цих методах фактично вважається, що всі показники є рівноправними для дослідника. Крім того, відбувається порівняння з певним об’єктом-еталоном.

При побудові інтегральної оцінки front-end розробки зазвичай достатньо важко визначитися з базовими показниками. Крім того, для різних веб-додатків різні показники можуть набувати різної ваги. Так для корпоративного порталу на перше місце виступають надійність, продуктивність, для веб-сайту Інтернет-магазину більш важливими є зручність використання та ефективність.

Тому для врахування переваг дослідника необхідно корегувати відповідні формули вводячи ваги показників, наприклад,

$$Ip_i = \sum_{j=1}^{n} k_j \frac{x_{ij}^{ф}}{x_{ij}^{б}}, \quad (2.4)$$

де k_j – вага j-го показника, $\sum_{j=1}^{n} k_j = 1$

Можна запропонувати такий метод побудови інтегрального показника, що враховує переваги дослідника:

$$Ip_i = \sqrt{k_1 x_{1i}^2 + k_2 x_{2i}^2 + ... + k_n x_{ni}^2} \quad (2.5)$$

Даний вид інтегрального показника має перевагу в тому, що дозволяє надати більшу значимість показникам, які мають більшу вагу.

Отже, наведемо алгоритм методики оцінювання front-end розробки:

1) для кожної j-ї метрики m_j визначається ваговий коефіцієнт V_j. Сума вагових коефіцієнтів всіх метрик, що відносяться до однієї і тієї ж групи характеристик, постійна і дорівнює 1.

$$\sum_{j=1}^{J} V_j^M = 1 \tag{2.6}$$

де J – кількість метрик для цієї групи, M – ознака метрики.

2) здійснюється оцінка кожної i-ї під характеристики якості $S_i^П$ за формулою

$$S_i^П = \frac{\sum_{j=1}^{J1} (m_j V_j^M)}{\sum_{j=1}^{J1} V_j^M} \tag{2.7}$$

де J1 – кількість метрик, що реально використовується при оцінці i-ї під характеристики, П – ознака під характеристики.

3) для кожної i-ї під характеристики якості $S_i^П$ визначається ваговий коефіцієнт $V_i^П$. Сума ваговий х коефіцієнтів всіх під характеристик, що відносяться до однієї під характеристики якості, постійна і дорівнює 1

$$\sum_{i=1}^{i} V_i^П = 1 \tag{2.8}$$

де I – кількість під характеристик деякої характеристики.

4) розраховується оцінка кожної k-ї характеристики якості C_k^X

$$C_k^X = \frac{\sum_{j=1}^{I1} (S_i^П V_j^П)}{\sum_{j=1}^{I1} V_j^П} \tag{2.9}$$

де I1 – кількість під характеристик, що реально використовуються при оцінці k-ї характеристики.

5) для кожної k-ї характеристики якості C_k^X визначається ваговий коефіцієнт V_k^X. Сума вагових коефіцієнтів всіх характеристик дорівнює 1.

6) інтегральна оцінка якості обчислюється за формулою

$$Q = \sqrt{\sum_{k=1}^{K} (C_k^X)^2 V_k^X} \qquad (2.10)$$

Алгоритм реалізації оцінювання якості Front-end додатків містить три фази: вимірювання показників реалізації web-ресурсу, елементарне оцінювання та частинне або загальне оцінювання web-ресурсу. На рисунку 2.4 зображено взаємозв'язок між цими етапами, проміжні та вихідні дані процесу реалізації оцінювання.

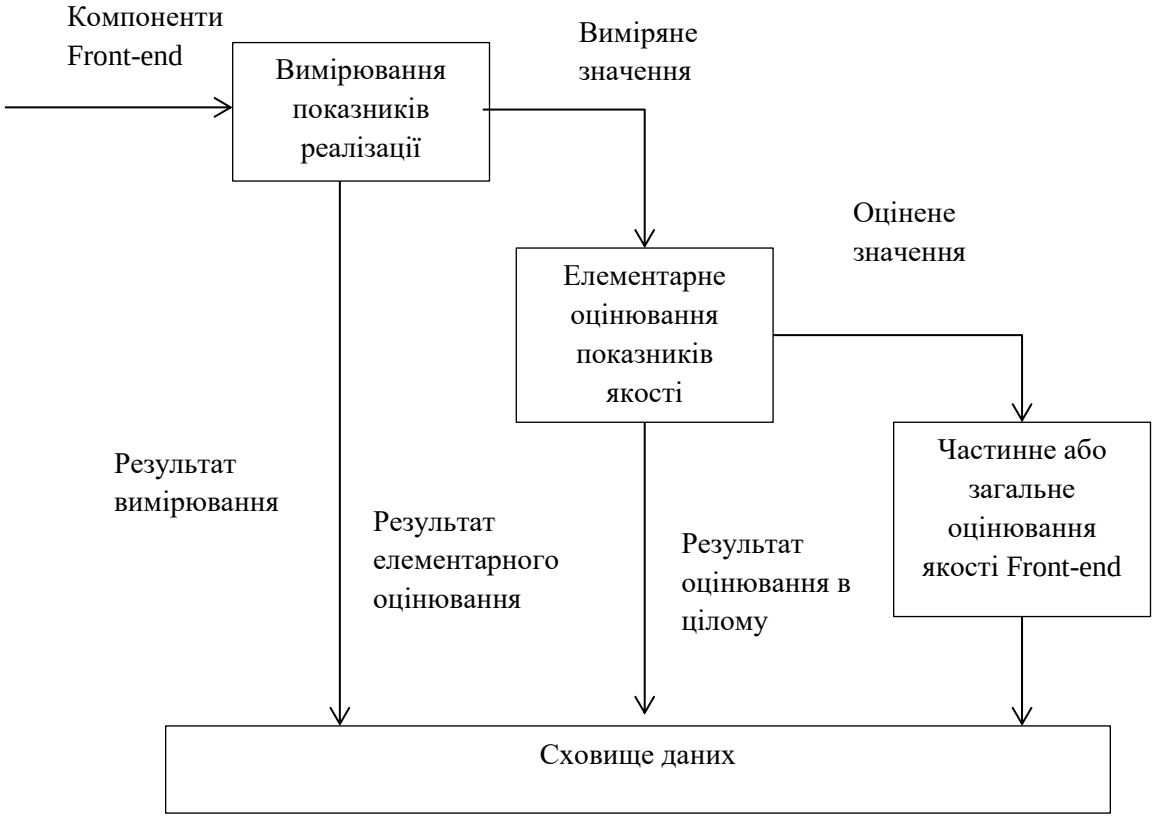

Рисунок 2.4 – Узагальнена реалізація процесу обчислення оцінки Front-end

Вибір оціночних елементів залежить від функціонального призначення Front-end розробки та визначається з урахуванням даних, отриманих при

проведенні експериментів різних видів, а також результатів експлуатації системи.

Побудована методика оцінювання Front-end розробки є достатньо простою в застосуванні. Їй властива певна гнучкість, яка дає змогу враховувати різні типи розробок та різні умови проведення досліджень, тому її можна використовувати для моніторингу Front-end розробки.

Суть моніторингу полягає в зборі необхідної інформації та ретельному її аналізі. Регулярне проведення моніторингу забезпечує своєчасне виявлення помилок і, відповідно, їх виправлення в найкоротші терміни.

Системи моніторингу здатна підвищити ефективність IT-служби компанії, знизити бізнес-ризики, пов'язані з недосконалістю інформаційних технологій і забезпечити доступність і якість основних послуг компанії.

Розроблена методика може стати важливим елементом забезпечення якісного функціонування Front-end компоненти веб-проекту компанії. Застосування даної методики не вимагає спеціальних навичок і дає можливість визначити, на скільки якість даної компоненти змінюється з часом, і що саме впливає на ці зміни.

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ОЦІНЮВАННЯ FRONT-END СИСТЕМ

3.1. Визначення бізнес-вимог до системи оцінювання

В основі розроблюваної системи оцінювання покладений На основі побудований методу оцінювання Front-end розробки.

Призначенням програми є збір і обробка інформації з оцінювання Front-end розробки.

Метою розробки системи є створення єдиного банку даних характеристик програмних проектів і централізованої системи оцінювання Front-end розробки.

Інформаційна система для оцінювання програмної розробки повинна забезпечувати наступне:

– можливість входу в систему для менеджера. Менеджер повинен мати всі права на редагування, зміну і доповнення інформації, а також розрахунку оцінок програмної розробки.

– можливість відображення інформації про характеристики програмних розробок та результати їх оцінювання.

– можливість аналізу оцінювання програмних розробок в динаміці [23].

Програма повинна представляти собою десктопний додаток.

Апаратне забезпечення системи повинно ефективно використовувати наявні технічні засоби. У складі апаратних засобів повинен бути персональний комп'ютер, який буде працювати як в якості сервера, так і в якості робочого місця, він повинен мати такі мінімальні характеристики:

– частота процесора не менше 1,6 ГГц (x86 або x64);

– об’єм оперативної пам'яті 1024 МБ;

– об'єм жорсткого диска 120 ГБ;

– пристрої периферії (миша, клавіатура);

– монітор.

Програмне забезпечення, що використовується при розробці і бібліотеки програмних кодів, повинні мати широке поширення, бути загальнодоступними і використовуватися в промислових масштабах.

Базовою програмною платформою повинна бути операційна система MS Windows.

Для роботи програми необхідно, щоб на комп'ютері були встановлені:

– операційна система з серії windows 7, 10;

– microsoft office;

– microsoft.NET framework.

Інформаційна система з оцінювання Front-end розробок забезпечує:

– збереження характеристик для оцінюваного програмного продукту;

– швидкий і зручний спосіб доступу до наявних результатів оцінювання;

– відображення результатів проходження тестів;

– можливість автоматизованого розрахунку оцінки Front-end проекту та занесення результатів оцінювання в базу даних;

– можливість виведення статистичних даних.

3.2. Проектування системи оцінювання

На основі аналізу вимог до системи було виділено такі передбачувані режими роботи програми:

– режим занесення і збереження даних;

– режим розрахунку;

– режим статистичної обробки інформації;

– режим пошуку даних.

Побудуємо діаграму варіантів використання для програми, що проектується наступним чином (рис. 3.1).

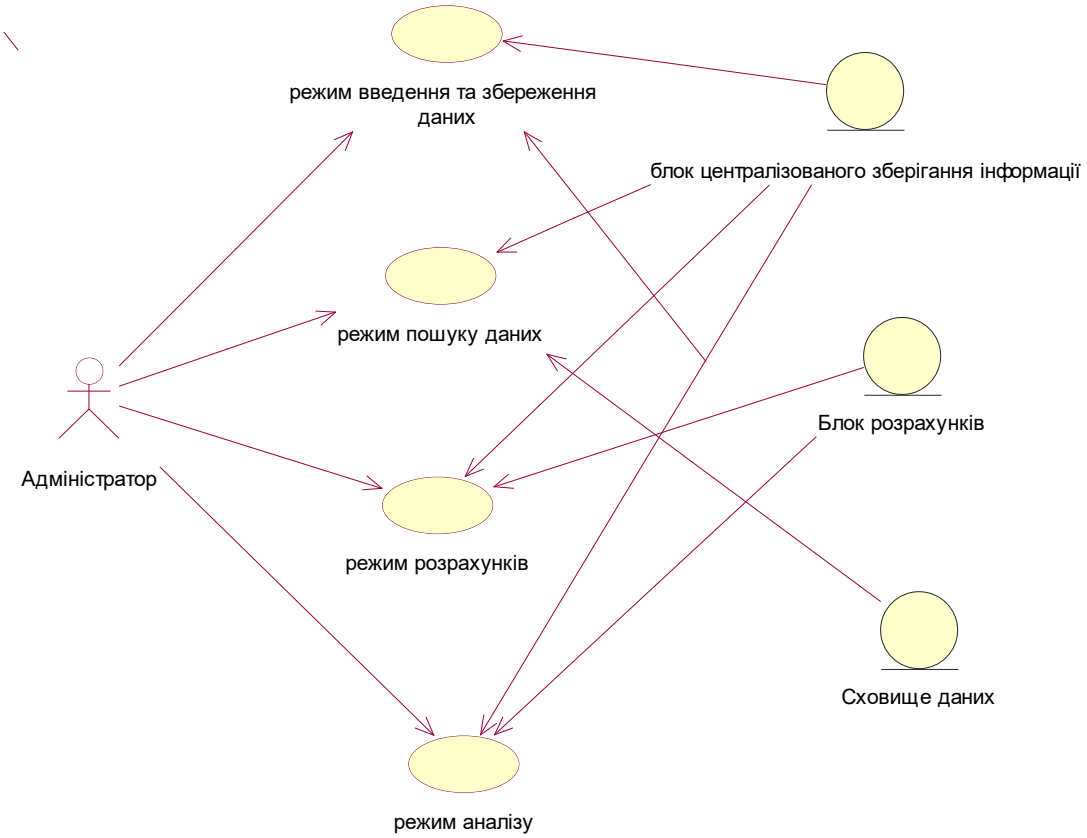

Рисунок 3.1 – Діаграма варіантів використання

Виходячи з вимог до системи і передбачуваних режимів її роботи, в якості її основних функціональних блоків були визначені: блок централізованого зберігання інформації, блок організації тестування, блок обробки інформації, сховище даних.

Блок централізованого зберігання інформації забезпечує можливість автоматизованого занесення результатів вимірювань в базу даних в базу даних.

Сховище даних забезпечує:

– наявність даних з оцінених front-end розробках;

– швидкий і зручний спосіб доступу до наявних результатів розрахунків.

Блок обробки інформації забезпечує

– розрахунок інтегральної оцінки Front-end розробки;

– можливість виведення статистичних даних про оцінювання Front-end розробок.

Функціональна схема системи відображена на рис.3.2.

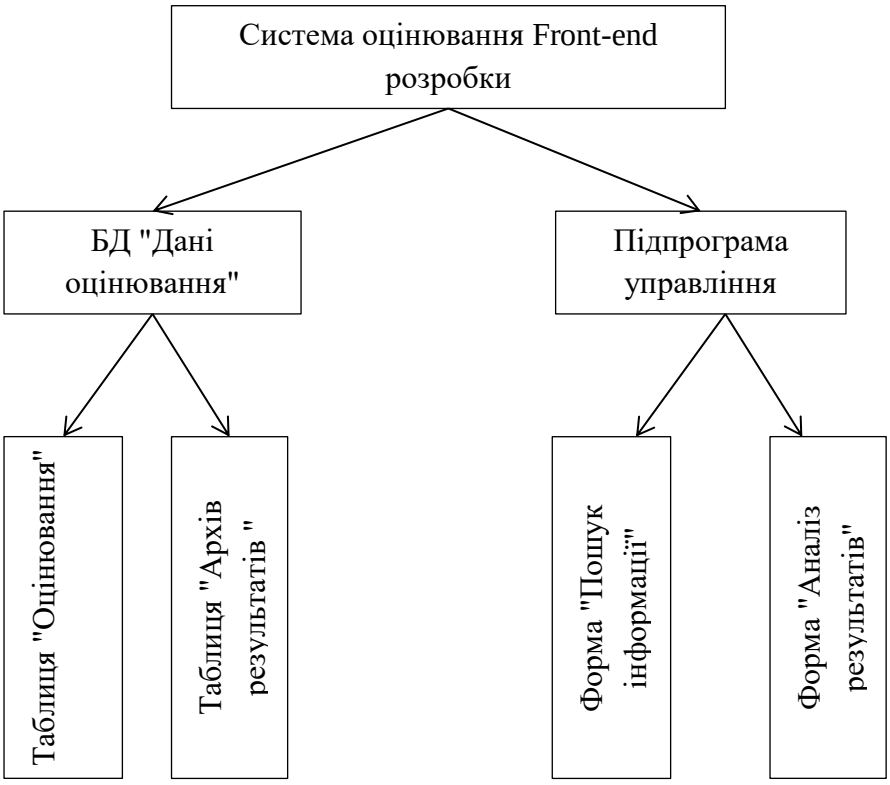

Рисунок 3.2 – Функціональна схема системи

Загальний алгоритм функціонування системи представлений на рис. 3.3.

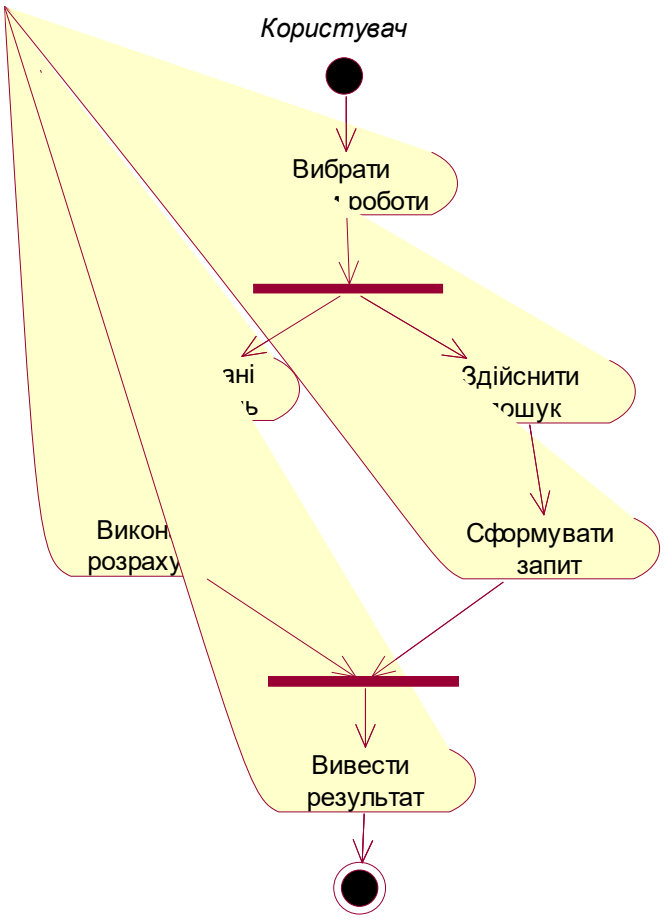

Рисунок 3.3 – Загальний алгоритм функціонування системи

Розглянемо даталогічну модель даних. Для даної системи була розроблена модель даних, яка складається з:

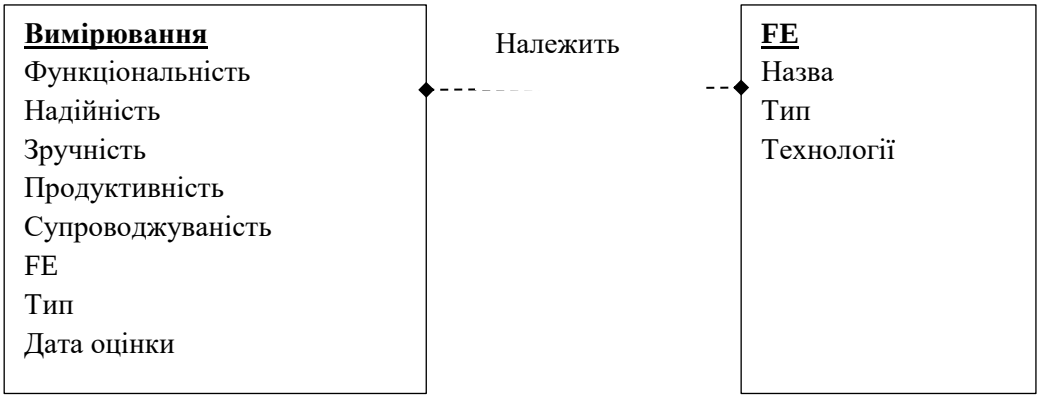

Рисунок 3.4 – Даталогічна модель даних

3.3. Реалізація проекту системи оцінювання

На основі визначених вимог до системи, її функціональної схеми, логічної схеми моделі даних для реалізації програмної розробки було вибрано наступне:

– блок обробки інформації, спроектований на базі додатків Windows, що гарантує можливість зручної програмної взаємодії з додатками;

– блок розрахунків створений на базі додатків Windows, який забезпечує можливість зручного програмної взаємодії з блоком зберігання інформації;

– блок централізованого зберігання інформації спроектований у вигляді бінарного файла визначеної структури. Такий вибір обумовлений декількома причинами. По-перше, бінарні файли можуть містити інформацію будь-якого вигляду та типу. По-друге, розмір такого файла значно менший від розміру файла будь-якої СУБД, оскільки він не містить додаткової інформації. По-третє, для обробки бінарного файла не потрібно використовувати спеціальні засоби, тому він може передаватися до інших програм для використання внесених даних. Таким чином, обраний варіант дає змогу створювати незалежне сховище даних для зберігання результатів оцінювання Front-end, які потім можна використовувати в подальшій системі моніторингу.

Фізична модель даних, реалізована в структурі бінарного файла, зображена на рис. 3.6.

baza.bin

Ім’я	Тип
nazva	string
func	double
usab	double
rel	double
perf	double
supr	double

Рисунок 3.6 – Фізична модель

Для розробки блоків використовувалась VisualStudio у вигляді багато віконного додатка WindowsForms на мові C #.

Вхідне вікно системи зображено на рис. 3.7.

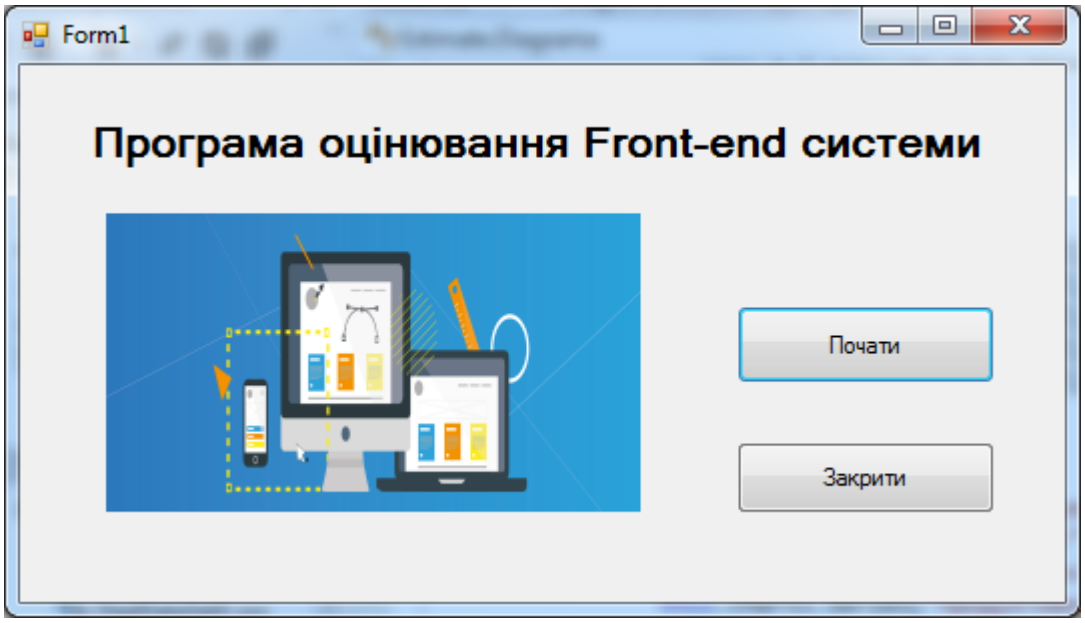

Рисунок 3.7 – Вхідне вікно системи

Після натискання кнопки Почати ми потрапляємо до основного вікна системи, де запропоновані основні режими роботи з програмою.

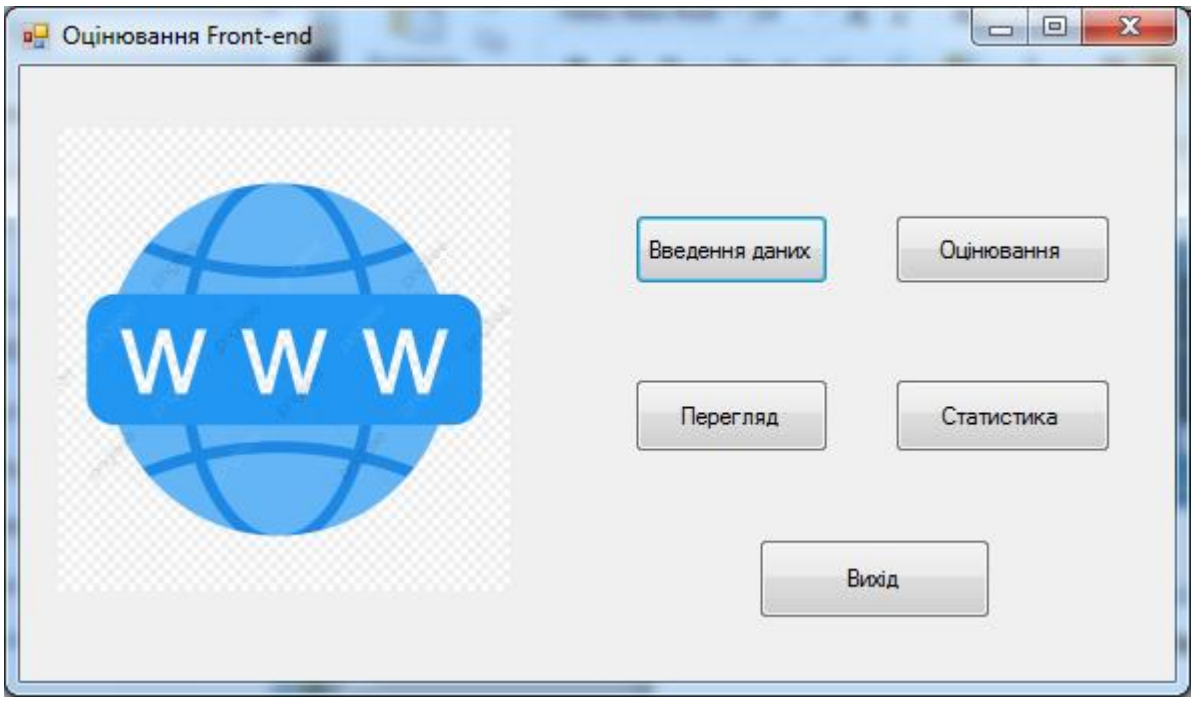

Рисунок 3.8 – Робоче вікно системи

Введення даних здійснюється в окремому діалоговому вікні. Користувачу потрібно ввести ім’я оцінюваної системи та її основні характеристики (рис.3.9)

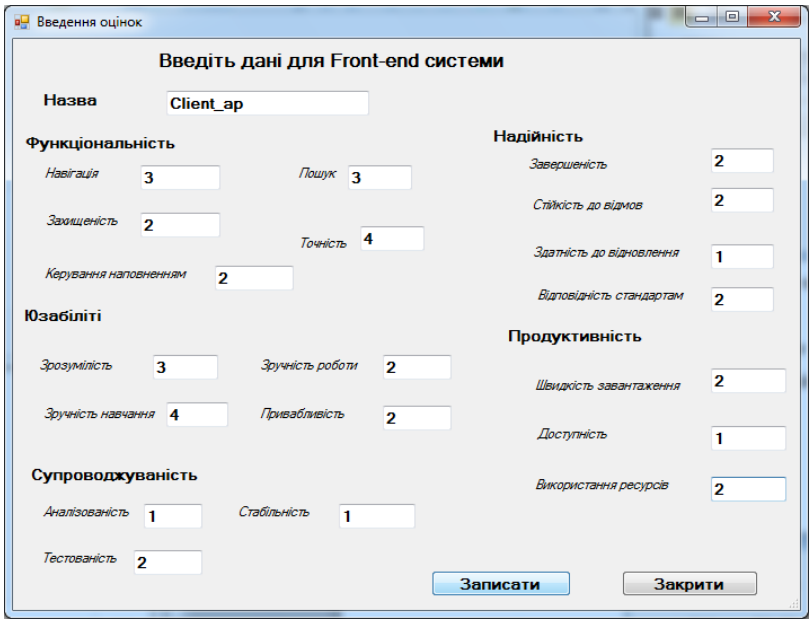

Рисунок 3.9 – Введення характеристики системи

Переглянути сховище даних з розрахованими характеристиками можна, використовуючи кнопку Перегляд (3.10).

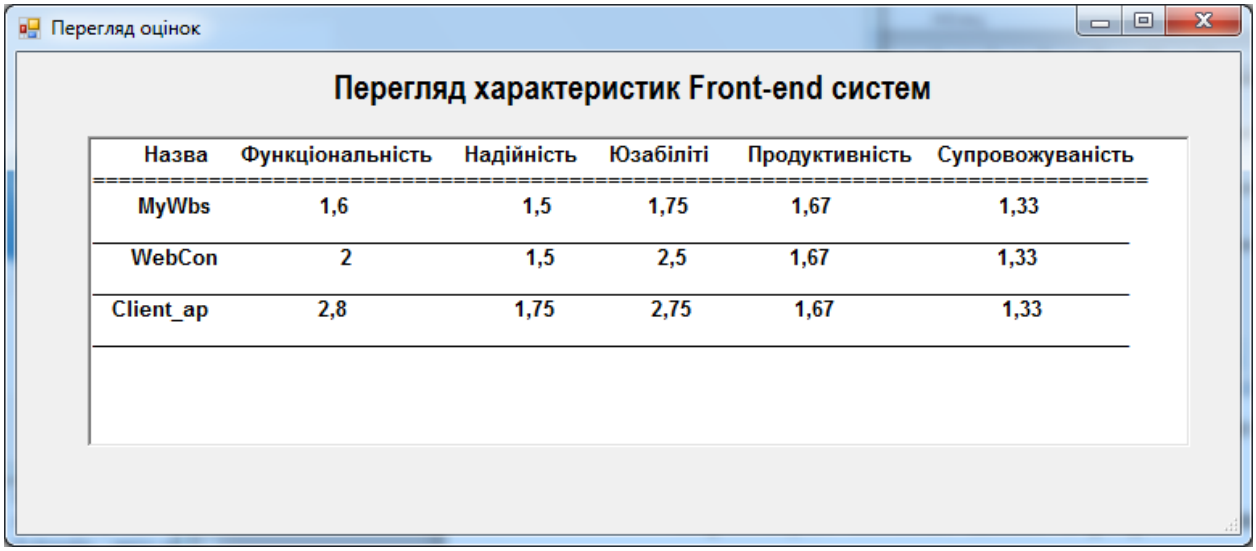

Рисунок 3.10 – Сховище даних

Для оцінювання Front-end системи користувачу необхідно ввести вагові коефіцієнти, які вказують на його вподобання (рис.3.11).

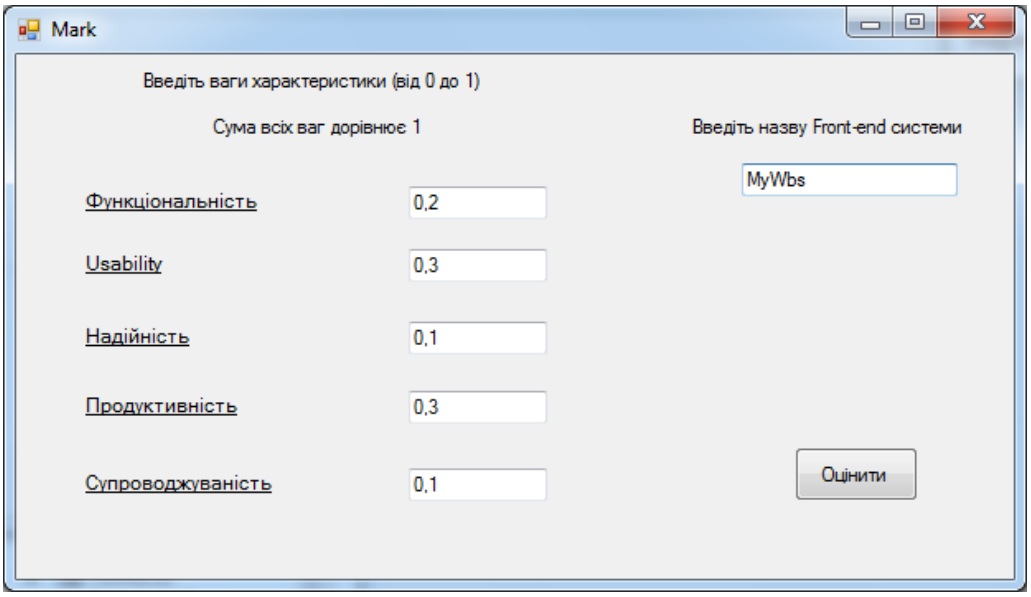

Рисунок 3.11 – Оцінювання Front-end системи

Результатом буде інтегральна оцінка (рис.3.12).

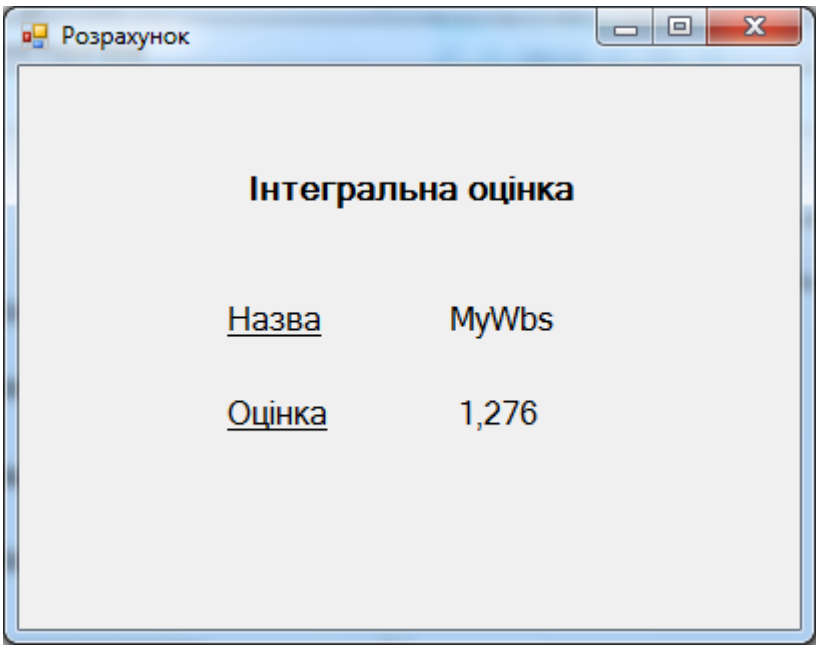

Рисунок 3.12 – Інтегральна оцінка Front-end системи

Якщо оцінювана система ще не введена в сховище, то програма видасть повідомлення про помилку (3.13)

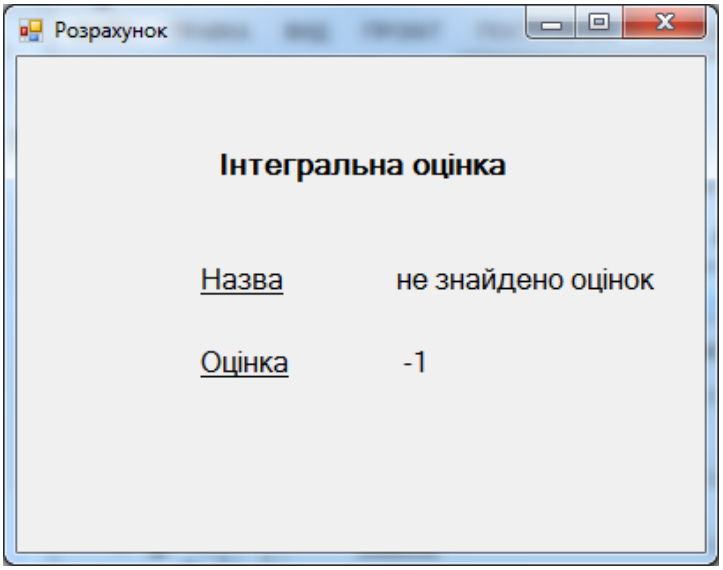

Рисунок 3.13 – Відповідь програми при відсутності оцінюваної системи

Для порівняльного аналізу характеристик оцінюваних систем використовується кнопка Статистика. Результат видається у вигляді діаграми (рис. 3.14).

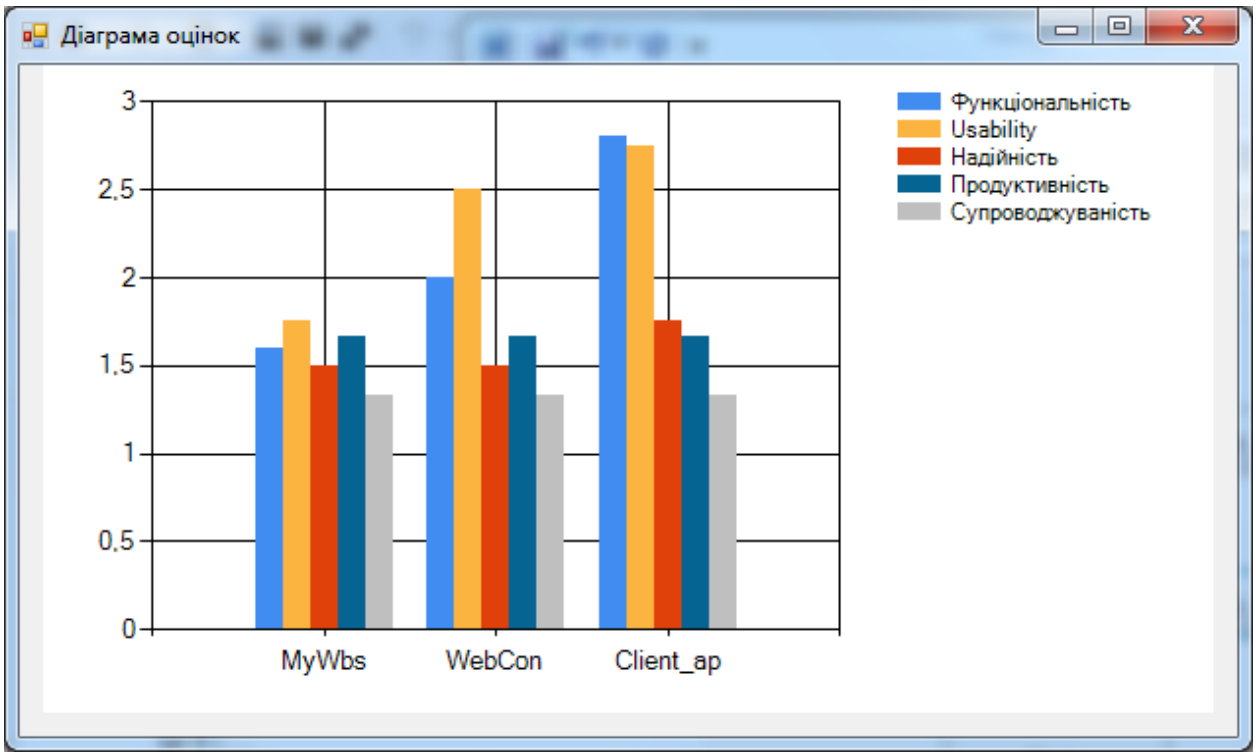

Рисунок 3.14 – Порівняльний аналіз характеристик Front-end систем

ВИСНОВКИ

Зараз широке розповсюдження отримали глобальні мережі. Різні організації почали використовувати Інтернет в своїй діяльності як засіб комунікації з партнерами, клієнтами. Тому для більшості людей веб-сайт є візитівкою компанії і саме він є основним інструментом для залучення клієнтів. Враховуючи велику конкуренцію в Інтернеті, кожна компанія має прагнути мати якісний веб-сайт, який відповідає як вимогам з точки зору розробки програмного забезпечення, так і вимогам з точки зору користувачів, які хочуть мати зручний інструмент для вирішення своїх задач.

В ході виконання даного дослідження були отримані такі результати:

– сучасні розробники при проектуванні веб-сайтів використовують компонентний підхід, який дозволяє відділити бізнес-логіку додатку від його інтерфейсу. З одного боку це дозволяє підвищити надійність програмного забезпечення, його стійкість, масштабованість. З іншого боку це ставить перед розробниками Front-end частини програмної системи задачі щодо використання відповідних стандартів для створення інтерфейсу, який буде задовольняти користувачів.

– для розробки Front-end систем використовують різні засоби та технології, які допомагають дотримуватися вимог щодо розробки програмного забезпечення. Основою для створення якісного програмного забезпечення виступають моделі якості.

– більшість моделей якості представляють собою ієрархічні системи, які складаються з окремих оцінювальних елементів, метрик та критеріїв. Їх використовують для оцінювання якості програмних продуктів, однак вони не враховують особливості функціонування Front-end систем.

– запропонована модель якості для Front-end систем дозволяє врахувати ці особливості, основною з яких є оцінювання usability системи. Відмінною особливістю даної моделі є те, що оцінку якості можна отримати

від двох сторін, що реалізують процеси життєвого циклу веб-додатків. При цьому в даній моделі, крім розробників коду, враховуються думки користувачів (розробників контенту). Запропонована оцінка є достатньо гнучкою, оскільки дозволяє враховувати лише ті параметри, які є апжливими для конкретної Front-end системи.

– дана модель може бути врахована в процесі розробки моделі якості веб-додатків. Її можна використати, як джерело даних щодо функціонування Front-end системи і відповідно на основі цих даних будувати стратегії щодо подальшого розвитку системи

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bass L., Clements P., Kazman R. Software Architecture in Practice. Addison Wesley, 2003, 528 p.
2. Boehm B.W., Brown J.R., Kaspar H., Lipow M., MacLeod G.J., Merritt M.J. Characteristics of Software Quality. TRW Series of Software Technology, Amsterdam, North Holland, 1978, 166 p.
3. Chang C., Wu C., Lin H. Integrating Fuzzy Theory and Hierarchy Concepts to Evaluate Software Quality. Software Quality Control, 2008, vol. 16, iss. 2, pp. 263–267.
4. Dromey G.R. A Model for Software Product Quality. Transactions of Software Engineering, 1995, vol. 21, iss. 2, pp. 146–162.
5. Fitzpatrick R. Software Quality: Definitions and Strategic Issues. Staffordshire University, School of Computing Report, 1996, 34 p. URL: https://www.saylor.org/site/wp-content/uploads/2012/11/Software-quality-definitions-and-strategic-issues.pdf (дата звернення: 24.04.2020)
6. Ghezzi C., Jazayeri M., Mandrioli D. Fundamental of Software Engineering. Prentice-Hall, NJ, USA, 1991, 543 p.
7. Grady R.B., Caswell D.L. Software Metrics: Establishing a Company-Wide Program. Prentice-Hall, 1987, 275 p.
8. IDE Метрики качества программного обеспечения URL: http:// www.pmprofy.ru/content/rus/67/672-article.asp (дата звернення: 26.04.2020).
9. Khosravi K., Gueheneuc Y. On Issues with Software Quality Models. Proceedings of 9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering. 2005, pp. 70–83.
10. McCall J.A., Richards P.K., Walters G.F. Factors in Software Quality: Metric Data Collection and Validation. Final Technical Report. Vol. 2. National Technical Information Service, Springfield, 1977.

11. McCall J.A., Richards P.K., Walters G.F. Factors in Software Quality: Concept and Definitions of Software Quality. Final Technical Report. Vol. 1. National Technical Information Service, Springfield, 1977.
12. McCall J.A., Richards P.K., Walters G.F. Factors in Software Quality: Preliminary Handbook on Software Quality for an Acquisition Manager. Final Technical Report. Vol. 3. National Technical Information Service, Springfield, 1977.
13. Polillo R. Quality Models for Web [2.0] Sites: A Methodological Approach and a Proposal. [Электронный ресурс]. URL: http://gplsi.dlsi.ua.es/congresos/qwe11/fitxers/QWE11_Polillo.pdf. (дата звернення: 06.03.2020).
14. Sharma A., Kumar R., Grover P.S. Estimation of Quality for Software Components: An empirical approach. ACM SIGSOFT Software Engineering Notes, 2008, vol. 33, iss. 6, pp. 1–10.
15. Автоматизированное тестирование Веб-приложений URL: http://www.developers.org.ua/archives/max/2006/ 06/05/selenium-ide-avtomatizirovannoe-testirovanie-veb-prilozheniy-za-15-minut/ (дата звернення: 26.04.2020).
16. Вступ у JavaScript. URL: https://learn.javascript.ru/intro (дата звернення: 12.04.2020)
17. Гагарина Л. Г. Технология разработки программного обеспечения : учебное пособие для вузов / Л. Г. Гагарина. Москва: ФОРУМ: ИНФРА-М, 2008,2011. 399 с..
18. Горбаченко И.М. Оценка качества программного обеспечения для создания систем тестирования // Фундаментальные исследования. 2013. № 6-4. С. 823–827
19. Дакетт Джон. HTML и CSS. Разработка и создание веб-сайтов / Джон Дакетт [пер. с англ. М.А. Райтмана]. М.: Издательство «Э», 2013. 480 с.
20. Дакетт Джон. JavaScript и jQuery. Интерактивная веб-разработка / Джон Дакетт [пер. с англ. М.А. Райтмана]. М.: Издательство «Э», 2017. 640 с.

21. Дмитревич Г.Д., Мохсер А.А., Ларистов А.И. Архитектура Web-ориентированных САПР // ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ, №5, 2010 . С.20-23
22. Исмаил Е.Е. Современные модели качества программных средств и их особенности // МНЖ «Поиск». 2015. № 3. С. 272–282
23. Лавріщева К.М. Програмна інженерія.–К.– 2008.–319 с.
24. Лешек А. Мацяшек. Анализ требований и проектирование систем. — Вильямс, 2002.
25. Модели менеджмента при разработке программных продуктов [Электронный ресурс]. URL: http://staratel.com/iso/InfTech/ DesignPO/index.html (дата звернення: 26.04.2020).
26. Основы программной инженерии (по SWEBOK). Конструирование [Электронный ресурс]. – Режим доступа: http://swebok.sorlik.ru/3_software_construction.html (дата обращения: 01.04.2020).
27. Принципи побудови моделі якості у використанні програмних систем/ І.Е. Райчев, О.Г. Харченко// Збірник наукових праць ІПМЕ НАНУ, вип. 39, 2007.
28. Разработка требований к программному обеспечению / Карл Вигерс//Пер. с англ. М.: Издательско-торговый дом «Русская Редакция», 2004. - 576с.
29. Ларин С.Н. Оценки моделей веб-приложений /С.Н. Ларин и др. / Региональная экономика: теория и практика, 2017, т. 15, вып. 6. С. 1187–1198
30. Сучасні інструменти для веб-розробника URL: https://depix.ru/articles/sovremennye_tehnologii_verstki_i_front_end (дата звернення: 12.04.2020)
31. Технологии разработки программного обеспечения : учебник / С. Орлов. – Санкт-Петербург : Питер, 2002. – 464 с.: ил.
32. Технология разработки программного обеспечения./ Брауде Е.// СПб.: Питер, 2004. – 655 с.