Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Метод та програмний засіб управління метриками якості при виборі технологій front end розробки
(тема)

Виконав студент групи СПд-2
Куніц В.В.
(прізвище, ініціали)
спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
освітньо-наукової програми
(тип програми)
Інженерія програмного забезпечення
(повна назва освітньої програми)
Керівник доц. Бойко І.В.

Допускається до захисту

Зав. кафедри, проф. _____ М.Р. Петрик

2020 р.

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Факультет комп'ютерно-інформаційних систем та програмної інженерії

Кафедра _____Програмної інженерії _____

Рівень вищої освіти - другий (магістерський)

Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма_Інженерія програмного забезпечення_

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Куніцу Василю Володимировичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Метод та програмний засіб управління метриками якості при виборі технологій front end розробки_____

затверджена наказом університету від "___" _____20 ___ р № _____

2. Термін подання студентом роботи до екзаменаційної комісії

24 грудня 2020 р.

3. Вихідні дані до роботи *алгоритм вибору технологій front end розробки, технологія і середовище об'єктно-орієнтованого програмування та проектування, метрики якості програмного забезпечення, рекомендації стандартів якості*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної області і постановка задачі, аналіз метрик якості та їх обґрунтування при оцінюванні технологій front end розробки, метод оцінювання якості технологій front end розробки, архітектура програмного засобу управління метриками якості, реалізація web-додатку управління метриками, апробація запропонованого методу і розробленого засобу*_____

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Охорона праці та			
безпека в надзвичайних ситуаціях			

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз особливостей і застосування технологій front end розробки	10 жовтня 2020 р.	
2.	Обгрунтування моделі та розробка методу управління метриками якості технологій front end розробки	30 жовтня 2020 р.	
3.	Програмний засіб управління метриками якості технологій front end розробки	24 листопада 2020 р.	
4.	Охорона праці та безпека в надзвичаних ситуаціях	1 грудня 2020 р.	
5.	Підготовка пояснювальної записки	6 грудня 2020 р.	
6.	Підготовка презентації та доповіді	8 грудня 2020 р.	
7.	Попередній захист	10 грудня 2020 р.	
8.	Нормоконтроль, рецензування	12 грудня 2020 р.	
9.	Занесення диплома в електронний архів	20 грудня 2020 р.	
10	Допуск до захисту у зав. кафедри	24 грудня 2020 р.	

Дата видачі завдання _____ _____2020 р.

Студент _____ *Куніц В.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *доц. Бойко І.В.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 112 с., 27 рис., 8 табл., 25 джер

FRONT END, МЕТРИКА, УПРАВЛІННЯ, ПРОГРАММНИЙ ЗАСІБ, ЯКІСТЬ.

Метою дипломної роботи є дослідження методів та інструментальних засобів управління метриками якості технологій побудови інтерфейсів інтерактивної взаємодії користувача з програмним забезпеченням.

У дипломній роботі проаналізовано сферу застосування і термінологічні особливості front end технологій у різних галузях інформаційних технологій, проведено аналіз характеристик та метрик якими можна скористатись при оптимальному виборі технологій front end розробки програмного забезпечення.

Визначено і класифіковано за характеристиками елементарні властивості якості FE технологій, запропоновано процедуру та реалізовано алгоритм оцінювання якості FE технологій на різних рівнях ієрархічного дерева моделі якості. Реалізовано програмний засіб підтримки запропонованих рішень.

KEYWORDS: FRONT END, METRIC, MANAGEMENT, SOFTWARE TOOLS, QUALITY.

The purpose of the thesis is to study the methods and tools for managing the quality metrics of technologies for building interfaces for interactive user interaction with software. The thesis analyzes the scope and terminological features of front end technologies in various fields of information technology, analyzes the characteristics and metrics that can be used in the optimal choice of front end technology software development. The elementary properties of FE technology quality are determined and classified according to the characteristics, the procedure is offered and the algorithm of quality assessment of FE technologies at different levels of the hierarchical tree of the quality model is implemented. A software tool to support the proposed solutions has been implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
ПЗ	Програмне забезпечення
BE	Back End
CASE	Computer Aided Software Engineering
ER	Entity Relations
FE	Front End
UML	Unified Modeling Language

ВСТУП

Актуальність теми. Розвиток інженерії програмного забезпечення супроводжується інтенсивним застосуванням новітніх, інноваційних технологій створення програмних продуктів і стимулює їхній розвиток та розгалуження різних середовищ виконання та функціонування. Одним із середовищ функціонування програмного забезпечення, що інтенсивно розвивається, є web-простір. Відповідно, з'явився новий сектор та напрям досліджень web-engineering, що досліджує методи та інструменти інтерактивної взаємодії користувача з програмним продуктом на основі визначених правил доступу до даних з однієї сторони і на рівні логіки формування запитів до бази даних з іншого. Усталеними термінами, які вживаються при розробці web-додатків є «front end (FE)» і «back end (BE)».

FE технології забезпечують інтерактивність інтерфейсу взаємодії кінцевого користувача системи з функціональністю, що реалізована на рівні сервера. Окрім цього, дані технології сприяють формуванню дружнього користувацького інтерфейсу, шляхом побудови деякої абстракції логіки виконання програмного коду.

Теперішні технології FE проєктування володіють потужними засобами створення інтерактивності користувацького інтерфейсу і використовують різні засоби доступу до даних. Однак, враховуючи широкий вибір таких технологій, різну функціональну повноту та зручність використання, постає питання щодо вибору оптимальної технології для реалізації певного конкретного проєкту. Тобто набувають все більшої актуальності задачі підвищення якості технологій FE розробки, які вимагають наукового обґрунтування моделей дослідження, побудови відповідного методу аналізу та оцінювання властивостей якості цих технологій, визначення метрик й автоматизації процесу оптимального вибору технологій для кожного конкретного випадку розробки ПЗ для web-простору.

На сьогодні багато наукових праць як українських, так і закордонних учених присвячено дослідженню якості програмного забезпечення. У цій галузі варто відмітити наукові здобутки таких учених як Лавріщева К.М., Андон П.І.,

Харченко О.Г., Харченко В.П., Яцишин В.В., Боднарчук І.О., Б. Боєм, І. Соммервіл та ряду інших.

Основна увага у працях цих учених приділяється побудові та аналізу моделей, методів та інструментів забезпечення, управління та контролю якості ПЗ з точки зору кінцевого програмного продукту. При цьому якості самих FE технологій приділяється дуже мало уваги. У зв'язку з цим, важливими та актуальними задачами у галузі інженерії ПЗ є аналіз та вдосконалення існуючих моделей представлення якості FE технологій, обґрунтування кількісних мір для вибору оптимальних технологій та автоматизація процесу управління метриками якості таких технологій.

Метою дипломної роботи є дослідження методів та інструментальних засобів управління метриками якості технологій побудови інтерфейсів інтерактивної взаємодії користувача з програмним забезпеченням.

Для досягнення сформульованої мети, у дипломній роботі розв'язуються **наступні задачі**:

– аналітичний огляд наукових праць і використання FE технологій проєктування ПЗ;

– обґрунтування моделі для вибору оптимальної технології реалізації інтерактивної взаємодії користувача програмного забезпечення;

– обґрунтування набору метрик для кількісного вираження якості FE технологій;

– створення методу оптимального вибору FE технологій;

– проєктування та імплементація програмного інструменту для управління метриками при виборі FE технологій.

Методи дослідження. Для досягнення мети використано наступні методи: аналіз та узагальнення – при проведенні аналізу існуючих моделей і метрик якості програмного забезпечення, технологій front end розробки; метричного аналізу – при побудові моделі і визначенні метрик якості; проектування, програмування і тестування – при розробці та апробації програмного засобу управління метриками.

Об’єктом дослідження є процеси управління метриками при виборі технологій Front end розробки.

Предметом дослідження є метрики, методи і засоби оцінювання якості технологій Front end розробки.

Наукова новизна одержаних результатів.

– уперше для оцінювання якості технологій front end розробки побудовано модель до складу якої входить два класи Design і Runtime, що сукупно включає 7 характеристик якості (6 стандартизованих+«Конфігурованість») та обгрунтовано використання метрик програмного коду, що дало змогу адекватно у кількісному вигляді представити якість таких технологій і прийняти рішення щодо вибору оптимальної;

– набув подальшого розвитку метод вибору технологій front end розробки, який базується на оцінках ієрархічних рівнів моделі якості і дає можливість враховувати важливість атрибутів і їх прийнятність

Практична цінність результатів дослідження. Впровадження програмного засобу управління метриками якості FE технологій дозволяє оптимізувати процес їх оптимального вибору на основі одержаних оцінок на рівні атрибутів і характеристик побудованої моделі.

Публікації. Основні результати дослідження апробовано на IX міжнародній науково - технічній конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя та на VIII науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року) у вигляді тез конференцій.

1. Куніц В.В., Бойко І.В. Аналіз особливостей технологій front end розробки. Матеріали IX міжнародної науково - технічної конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: ТНТУ. 2020. С. 34.

2. Куніц В.В., Бойко І.В. Алгоритм роботи програмного засобу управління метриками при оцінюванні технологій front end розробки. Матеріали VIII науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року). Тернопіль: ТНТУ. 2020. С. 150.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, 4 розділів, висновків, списку використаної літератури та додатків. Обсяг роботи: пояснювальна записка – 112 арк. формату А4, графічна частина – 32 слайди.

1 АНАЛІЗ ОСОБЛИВОСТЕЙ І ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

1.1. Аналіз сфери застосування технологій Front end розробки

У галузі інформаційних технологій та інженерії ПЗ під термінами «front end» і «back end» розуміють шляхи розподілу функціональності між рівнями надання даних кінцевому користувачу та фізичного чи абстрактного доступу до інформації у відповідних сховищах.

В загальному випадку, FE представляє собою результат використання технології забезпечення комунікації між кінцевим користувачем (інтерактивний інтерфейс користувача) і логікою функціонування програмного забезпечення на рівні сервера. FE та BE можуть бути розмежованими між фізично чи логічно різними середовищами виконання та апаратними ресурсами.

Основне завдання і призначення FE технологій створення ПЗ полягає у забезпеченні дружнього інтерфейсу при взаємодії користувача з системою, а також організації абстракції при переході з одного рівня, наприклад апаратного, до більш високих програмних рівнів.

Аналізуючи найбільш поширені архітектури програмного забезпечення, для прикладу архітектура Model-View-Controller, можна помітити чітку тенденцію до поділу на інтерактивну та серверну частини. У даному випадку, View є FE компонентою ПЗ, а Model – BE, оскільки безпосередньо працює з об’єктами бази даних на фізичному рівні.

Перевага застосування і поділу на front і back дає змогу спростити процес імплементації програмного забезпечення і забезпечити розподілену підтримку окремих частин програмного забезпечення.

На практиці, для означення front end користуються емпіричним правилом – будь-який компонент ПЗ на який має вплив кінцевий користувач відноситься до інтерактивної front end частини. Програмний код логіки функціонування чи опрацювання запитів користувачів міститься на сервері. Тому середовищем у

якому працюють, наприклад веб-дизайнери над HTML, не повинен бути сервер, і навпаки, back end розробники основну роботу виконують на сервері. Для створення повнофункціонального інтерактивного web-додатку необхідне залучення обох фахівців. Проведемо аналіз типів front end у різних прикладних сферах і застосуваннях.

Перший приклад front end частини у комп'ютерних системах – графічний файловий менеджер, що є «мостом» між користувачем та операційною системою. У даному випадку, кінцевий user, використовуючи файловий менеджер взаємодіє від front end частини комп'ютерної системи тим самим формує запит до back end, запускаючи на виконання сервіси операційної системи. Порівнюючи графічну оболонку файлового менеджера з командним рядком, можна зробити висновок, що перший інструмент взаємодії з операційною системою вимагає знання відповідних команд і термінології, а другий – є інтерактивним зручним інструментом для користувача.

Інший приклад та рівень front end – псевдографічний інтерфейс ncurses у Unix подібних ОС. Особливістю цього інтерфейсу є те, що на його рівні більшість потоко-орієнтованих програмних систем працюють як своєрідні фільтри, що, в залежності від ситуації, можна інтерпретувати як FE, так і BE для інших взаємодіючих програм.

У комп'ютерних мережах в ролі FE можуть виступати будь-які апаратні пристрої з вбудованим програмним забезпеченням, які здійснюють оптимізацію або захист трафіку на мережевому рівні. Такі пристрої виконують фільтрацію пакетів даних, перед тим, як вони починають циркулювати у комп'ютерній мережі.

Для CMS (Content Management System) характерною особливістю є те, що інтерфейс з яким працює звичайний користувач належить до front end частини, а засоби керування з відповідним інтерфейсом адміністративної частини формують back end частину.

Якщо розглянути принцип роботи компіляторів, то їхня FE частина виконує перетворення вихідного програмного коду у деяку проміжну структуру з якою вже пізніше взаємодіє BE частина для генерації коду на машинній мові.

BE є оптимізованим з точки зору швидкості генерації та виконання машинного коду. Для прикладу, у Visual Studio наявний FE, який дає змогу аналізувати код, написаний різними мовами програмування і перетворювати його у деяку проміжну мову (CLR). Після цього BE перетворює цей код у машинний і виконує.

Якщо розглядати сферу синтезу мовлення, то у даному випадку front end виконує функцію аналізу і конвертації тексту у вигляді деякого символьного та фонетичного образу. Відповідно, back end транслює цей образ в аудіо-звуки.

Сучасний етап розвитку ПЗ, орієнтованих під веб – простір, вимагає від розробників не тільки знань мов програмування, але і вміння та здатності використовувати різні платформи, фреймворки і препроцесори. Їхнє застосування дозволяє значно спростити та пришвидшити процес реалізації ПЗ.

В загальному випадку, software framework представляє собою сукупність готових програмних рішень, які можуть містити елементи дизайну, функціональних компонентів, що реалізують певну логіку виконання.

Функціональність фреймворків розширюється шляхом забезпечення здатності до інсталяції додаткових бібліотек, компонентів повторних використання, оновлень та ін.

Однією з основних переваг при застосуванні фреймворків і каркасних додатків, є те, що вони дають змогу уніфікувати структуру коду як при програмуванні FE частини, так і BE. За допомогою каркасних рішень набагато спростився процес автоматичної генерації інтерфейсів за рахунок стандартизації структури програмного коду. При цьому, досить ефективно використовуються властивості об’єктно-орієнтованого програмування, зокрема принцип успадкування.

Для підготовки даних і забезпечення здатності їхнього використання іншими програмами можуть бути використані препроцесори, як окремі компоненти компіляторів. Результат роботи і принцип аналізу препроцесорів залежить від їхнього функціонального призначення. Для прикладу, одні препроцесори можуть виконувати функції синтаксичного аналізатора програмного коду, інші –

представляти код програми, написаної на деякій мові, у програмний код іншої мови.

У результаті проведеного аналізу, визначено роль, функціональне призначення і принципи застосування технологій front end розробки ПЗ для web-простору.

1.2. Аналіз технологій front end розробки програмного забезпечення для web-простору

Сьогодні спостерігається стрімкий розвиток і використання технологій front end для реалізації програмного забезпечення під web. Ці технології включають не тільки використання статичних CSS при верстці html-сторінок та мінімального динамічного контенту на основі javascript/jquery, але й інструменти і засоби управління проектами. Для прикладу, FE технології дозволяють збирати проекти за допомогою таких засобів як grunt та webpack, процес тестування підтримується такими технологіями для написання автоматизованих тестів як jasmine і chai, які можна запускати на стороні front end за допомогою браузерів.

Для забезпечення продуктивної інтерактивності інтерфейсів використовують бібліотеки React.js, ES6, Node.js – платформа, що забезпечує можливість реалізації front end і back end частин проекту за допомогою JavaScript.

1.2.1. AngularJS

AngularJS на сьогодні є широко використовуваним і популярним фреймворком, який показав ефективність при створенні та супроводі складних веб-додатків. Даний фреймворк з відкритим вихідним кодом позиціонується як розширення HTML для побудови складних веб-орієнтованих додатків. На рисунку 1.1 показано вікно документації AngularJS.

Рисунок 1.1 – Вікно сайту з документацією AngularJS

Фреймворк AngularJS містить велику кількість інструментів, які дають змогу пришвидшити та підвищити ефективність розробки програмного забезпечення. До найбільш використовуваних інструментів цього фреймворку належать:

– фільтри;
– директиви;
– прив'язка даних.

На початку AngularJS орієнтувався на створення односторінкових веб додатків, основна логіка роботи яких полягала у тому, що спочатку завантажується одна сторінка з відповідним контентом, а інший контент підвантажується по мірі необхідності. Це дає змогу знизити навантаження на трафік та серверну обробку подій.

В основі AngularJS лежить мова програмування JavaScript, що дозволяє використовувати її функції в якості контролерів, сервісів чи фільтрів, а також додаткові методи з інших сторонніх проектів.

AngularJS дає змогу скоротити час реалізації веб-додатків за рахунок вбудованих механізмів. Один з них є шаблонізатор (template engine). Шаблонізатор дозволяє виконувати вбудовані команди AngularJS для виведення даних (лістинг 1.1).

Лістинг 1.1 – Приклад використання шаблонізатора

```
<Div class = "container">
  <Ul class = "list">
    <! - {{item.name}} - це частина шаблонізатора AngularJS
в бразуер користувача вона буде замінена на назву товару ->
    <Li ng-repeat = "item in products"> {{item.name}} </ li>
  </ Ul>
</ Div>
```

Інший інструмент фреймворку AngularJS забезпечує управління HTML тегами, тобто DOM (Document Object Model). При використанні jQuery розробник постійно повинен вручну оновлювати всі дані на екрані користувача, то у випадку інструменту AngularJS цього робити не потрібно, оскільки достатньо вказати які елементи до яких об'єктів даних прив'язані і при кожній зміні стану на екрані буде виводитися оновлена інформація.

AngularJS використовує об’єктний підхід для розробки, зокрема архітектурний патерн MVC, а точніше його різновид – MVVM (Model View - View Model). У результаті застосування такої архітектури програмний код стає більш структурованим, що сприяє простоті його підтримки та управління [3].

AngularJS володіє і підтримує властивість вбудованості, що дозволяє розробникам гнучко використовувати сторонні технології. При цьому, на відміну від більшості фреймворків, які вимагають повного контролю над сторінкою, AngularJS дозволяє запуск кількох додатків на одній сторінці без використання iframe.

Для забезпечення зв’язності компонентів програмного коду в AngularJS застосовуються залежності, які дають змогу задекларувати зв’язки між взаємодіючими частинами програмного забезпечення. Власне Dependency injection є перевагою фреймворку AngularJS, оскільки компонент здатний гнучко змінювати свої складові.

Ще однією з переваг даного фреймворку є те, що його архітектура дозволяє відокремити представлення даних кінцевому користувачу та поведінку системи, що забезпечує структурування та паралельність процедур тестування.

Підтримка готових рішень, тобто застосування повторно-використовуваних компонентів, дає змогу гнучко та ефективно проектувати власні, орієнтовані під специфіку предметної області користувача, рішення.

Для прикладу, найбільш популярними модулями, які використовуються практично у будь-яких проектах є модулі маршрутизації ui-router, модулі представлення даних у таблицях ui-grid, ng-table і т.д..

Окрім цього, AngularJS дозволяє створювати гібридні додатки, які одночасно орієнтовані під web та мобільні додатки, а гнучкість структури проекту забезпечує реалізацію додатків будь-якої складності.

1.2.2. KnockoutJS

KnockoutJS є ще однією з бібліотек JavaScript, що забезпечує ефективність побудови доволі складних інтерактивних користувацьких інтерфейсів. Характерними особливостями даної бібліотеки є те, що вона при проектуванні інтерфейсів, код задовольняє властивість clean code, є добре структурованим та здатним до розширення, а також зручним для читання.

Основна задача щодо функціональної придатності KnockoutJS – це вирішення задачі автоматичного оновлення користувацького інтерфейсу при оновленні елементів моделі. На рис. 1.2 показано інтерфейс документації KnockoutJS.

KnockoutJS може взаємодіяти з різними бібліотеками, однак найбільш сумісною вона є з jQuery, а роль основної системи шаблонізатора виконує jQuery.tmpl.

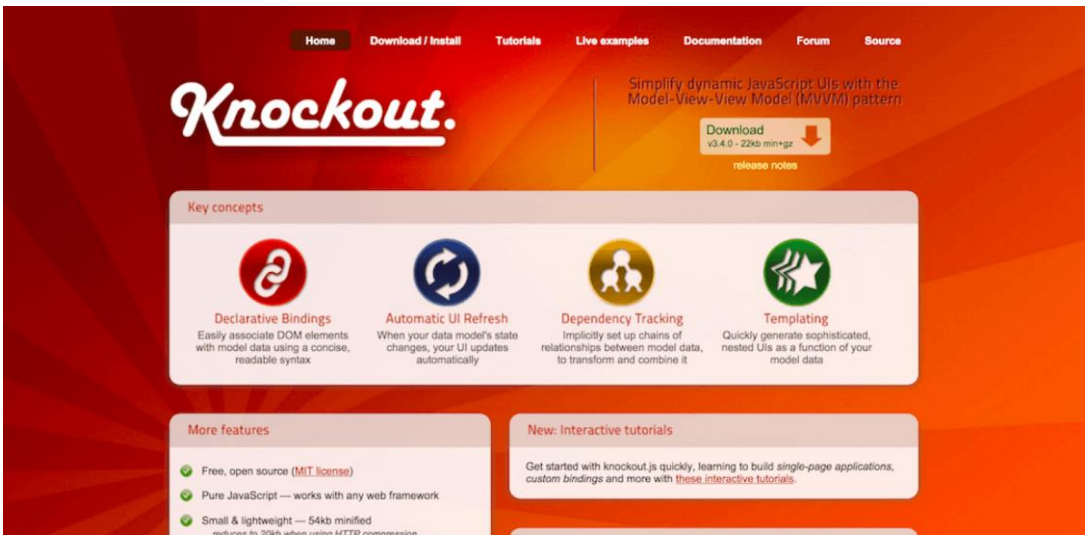


Рисунок 1.2 – Документація KnockoutJS

Бібліотека KnockoutJS містить ряд інструментів, які дають змогу забезпечити інтерактивність інтерфейсу веб-додатку, його динамічність, сортування даних та ряд інших.

1.2.3. Bootstrap

У 2011 році автори соціальної мережі Twitter створили ефективний фреймворк Bootstrap з відкритим вихідним кодом. Ком'юніті Bootstrap весь час зростає та розвивається, що зумовлено потужною підтримкою зі сторони розробників та випуском нових оновлень та функціоналу. До складу даного фреймворку входять модулі для роботи з HTML (Hyper Text Markup Language), CSS (Cascade Style Sheets) і JavaScript, забезпечено можливість побудови адаптивного дизайну.

До складу Bootstrap додано різноманітні теми, які постачаються у відповідності до вимог і стандартів material design, оновлені та оптимізовані теми для забезпечення підтримки Sass.

До переваг Bootstrap, в порівнянні з іншими фреймворками, належать:

– здатність забезпечити адаптивний дизайн;

– наявність добре структурованої документації.

До недоліки фреймворку Bootstrap можна віднести наступні :

– наявність тем, які рідко використовуються, але значно збільшують розмір компонентів;

– велика кількість програмних класів, методів і DOM компонентів, які створюють незручності та порушують принцип clean code.

1.2.4. CSS препроцесори

Одними з найбільш популярних і використовуваних препроцесорів CSS є SASS та LESS, що дозволяють забезпечити динамічність коду.

SASS представляє собою препроцесорну мову і дає змогу розширити функціональність базової CSS за рахунок змінних, застосування вкладених правил, інтегрованого імпорту та ряду інших і при цьому повністю сумісний з з CSS.

Основною перевагою SASS є те, що даний препроцесор забезпечує строгу організацію стилів і тим самим дозволяє досягти чіткості і компактності стилів. Разом з бібліотекою compass, SASS нівелює залежність від префіксів CSS3 при роботі у браузерах і робить його зручним інструментом при проектуванні front end.

Синтаксично SASS подібний до мови ruby (старіші версії) та до CSS (нові версії), що робить його зрозумілим і зручним при верстці web-додатків.

Термін "SASS" можна застосовувати як до технології, так і до синтаксису. Файли SASS можуть також бути файлами SCSS, між ними немає особливих відмінностей, місцями SCSS просто більш нагадує звичний CSS.

LESS подібний до SASS і відрізняється лише синтаксисом, хоча по суті вони обидві не є окремими мовами, а тільки розширенням CSS. Код SASS / LESS компілюється в звичайний CSS. Найбільша відмінність між SASS і LESS, крім синтаксису, те, як вони працюють. SASS компілюється за допомогою Рубі, в той час як LESS - використовує JavaScript (або Node.js). Потрібно звернути увагу на те, що попередня обробка CSS дійсно вимагає невеликого знання команд Terminal /

CLI. Не обов'язково бути експертом, але повинні бути хоча б базові знання роботи в командному рядку [8].

1.3. Аналіз критеріїв якості front end розробки

Серед критеріїв якості front end розробки розрізняють наступні:

– зв’язність;
– зчеплення;
– ефективність;
– продуктивність;
– супроводжуваність;
– реактивність.

Проаналізуємо кожен з критеріїв в контексті його ваги і з точки зору вибору оптимальної технології front end розробки.

Розрізняють два види зв’язності: зв’язність за даними та зв’язність в процесі керування. У випадку критерію зв’язності за даними, досліджується якість передачі інформації від одного модуля до іншого. Даний вид зв’язності оцінюється за способом чи каналом передачі даних та якістю самих даних. Якщо передача даних від одного модуля до іншого відбувається прямим шляхом, тобто дані виступають як параметри функцій, то сила зв’язку оцінюється кількістю аргументів функції та їхнім типом. При цьому рекомендовано передавати прості типи даних, а не структури чи об’єкти.

У випадку непрямого зв’язку між модулями, передбачається, що різні модулі спільно використовують одні і ті ж дані. Прикладом непрямого зв’язку може бути організація бази даних чи іншого виду сховища, вміст якої використовує деяка сукупність програмних модулів. У такій ситуації говорять про атрибут і метрику зв’язності за загальною зоною [9]. Якщо ж модулі взаємодіють таким чином, що один модуль може використовувати дані, які знаходяться або опрацьовуються іншим модулем, то такий вид зв’язності є зовнішнім. При розробці програмного забезпечення, не рекомендують використовувати зовнішню зв’язність, оскільки

підвищується структурна складність програмного коду, а також зростає кількість елементів контролю доступності даних при виконанні операцій зчитування та запису.

Під зв’язністю за критерієм управління, в загальному випадку, розуміють передачу керуючих даних з одного модуля до іншого і встановлення порядку та алгоритму його виконання. Зростання кількості даних управління може спричинити складнощі та збільшити трудомісткість щодо налаштування та відлагодження програмного забезпечення.

Незалежність модулів характеризується відсутністю спільних параметрів та викликів. У програмуванні повністю незалежні модулі практично не зустрічаються. Залежність між модулями, або по іншому зв’язність чи зчеплення між модулями, характеризується кількістю інформації якою володіє один модуль про інший. Чим більше даних про модуль міститься в іншому компоненті – тим більше зчеплення між модулями. При розробці програмного забезпечення дотримуються принципів мінімізації зчепелення між компонентами, що обумовлено простотою внесення зміни у такі модулі без необхідності їх внесення у пов’язаних модулях.

Зі ступенем міжмодульного зчеплення пов’язана така характеристика якості як супроводжуваність, яка є комплексною мірою якості процесу супроводу програмного забезпечення.

Задачами структурного проектування архітектури програмного забезпечення є встановлення такого зчеплення між модулями, щоб дані можна було передавати з одного модуля в інший за допомогою явного представлення і простих типів даних. При цьому повинна забезпечуватись певна функціональність, а не управління.

Зв’язність всередині модуля характеризує його «міцність». Вважається, що один модуль повинен реалізовувати одну функціональність (функцію), а його структурні елементи є необхідними компонентами для правильного опрацювання цієї функції. З формальної точки зору, зв’язність всередині модуля можна вважати достатньою, якщо функціональність, яку він реалізує можна описати одним стверджувальним реченням.

Для забезпечення якості програмного забезпечення необхідно мінімізувати зчеплення між модулями та максимізувати зв’язність всередині нього.

При оцінюванні незалежності модулів, окрім зчеплення та зв’язності використовуються й інші метрики (атрибути):

– кількість стрічок коду модуля;
– застосування наперед визначених компонентів;
– принцип і структура управління модулем;
– застосування внутрішніх функцій.

Кількість стрічок програмного коду, або по іншому його розмір, безпосередньо впливає на ступінь зрозумілості і читабельності програми, складність відлагодження та внесення змін у модуль.

У наукових працях [6-8] і практиках програмування [7-10] прийнятним є наступний інтервал щодо кількості стрічок коду R: $R \in [10,50]$.

Гранична кількість стрічок коду на рівні 50 пов’язана із психологічними особливостями людини, тобто це є кількістю інформації, яка ефективно опрацьовується і підлягає аналізу розробником програмного забезпечення.

Мінімальна кількість стрічок на рівні 10 обумовлена тим, що значення, менші за заданий рівень, значно знижують показники доцільності та раціональності використання часу, тобто спостерігається збільшення операційних часових ресурсів через підвищену кількість звернень до сторонніх компонентів за межами модуля.

Наперед визначені компоненти представляють собою модулі, стан і виконання яких не залежить від їхнього попереднього використання і не порушують технологію та алгоритм виконання процедур при розв’язку задачі. Використання таких компонентів забезпечує необхідний ступінь надійності програмних модулів, а формування вихідних результатів варто організовувати так, щоб модуль, який формує результат викликався на стороні модуля, який власне проводить обчислення результату.

Застосування внутрішніх функцій передбачає використання функціонально і структурно завершених процедур, які містяться у програмному модулі. При цьому

ускладнюється процес відлагодження модуля у зв’язку з нездатністю незалежного використання функцій і доступу до них зовнішніх модулів. Застосування внутрішніх функцій є доцільною у випадку обмежених часових ресурсів та наявності достатнього об’єму пам’яті.

До критеріїв якості та відповідних метрик front end розробки належить також ефективність, що представляє собою сукупність атрибутів, які виражають міру залежності між ступенем функціональності і продуктивності програмного забезпечення та ресурсів, які для цього необхідні.

Критерій супроводжуваність представляє собою набір атрибутів якості, що характеризують зусилля, які потрібно використовувати для проведення модифікацій та адаптацію ПЗ у нових умовах чи середовищі виконання, а також при зміні вимог до програмного забезпечення [12].

До критеріїв адаптивності при розробці web-орієнтованих програмних продуктів належить браузерна незалежність, або по іншому кросбраузерність та кросплатформність. Під кросбраузерністю розуміють здатність web-додатку забезпечувати однакову поведінку, функціональність та відображення незалежно від типу використовуваного браузера. У випадку, коли відсутні зміщення графічних блоків і не знижена читабельність сторінки web-сайту, вважають, що даний програмний продукт задовольняє критерію кросбраузерності.

Під кросплатформністю розуміють здатність програмного забезпечення однаково правильно функціонувати на різних програмно-апаратних платформах. Властивість кросплатформності дає змогу значно знизити часові і фінансові ресурси при розробці нового чи адаптації уже створеного ПЗ.

Ще одна характеристика технологій front end розробки – продуктивність. Дана характеристика дає оцінку кількості інформації, яка опрацьовується системою за одиницю часу [11].

Окрім цього, важливим атрибутом web-додатків, а відповідно і технологій їх розробки, є реактивність системи, що вимірюється часом від поступлення даних на вхід та одержанні інформації на виході.

На основі зв’язності всередині модуля та зчеплення між модулями впроваджено ще одну метрику пов’язаність, що дає змогу більш ширше оцінювати властивості технологій front end розробки. Дані метрики засновані на кращих практиках розробки програмного забезпечення, які знижують затрати при оновленні та підтримці програмних продуктів.

У даному розділі проведено аналітичний огляд існуючих рішень щодо front end технологій розробки програмного забезпечення, визначено їхні переваги та недоліки, а також проаналізовано критерії та деякі метрики якості як технологій front end проектування, так і web-додатків.

Основні результати даного розділу полягають в наступному:

1. Проаналізовано сферу застосування і термінологічні особливості front end технологій у різних галузях інформаційних технологій, що дало змогу виявити їхні спільні риси і чітко встановити, що технології front end розробки програмного забезпечення під web-простір забезпечують інтерактивність користувацького інтерфейсу, а метрики оцінювання якості потребують проведення додаткових досліджень.

2. Проведено аналіз найбільш популярних технологій front end розробки у результаті якого встановлено, що для вибору оптимальних технологій необхідно розробити універсальну та загальноприйняту систему метрик, оскільки існуючі критерії є індивідуальними для кожної з технологій.

3. Проведено аналіз характеристик та метрик якими можна скористатись при оптимальному виборі технологій front end розробки програмного забезпечення, у результаті якого встановлено, що деякі метрики якості потрібно формалізувати, доповнити для забезпечення більш повної картини опису таких технологій.

2 ОБГРУНТУВАННЯ МОДЕЛІ ТА РОЗРОБКА МЕТОДУ УПРАВЛІННЯ МЕТРИКАМИ ЯКОСТІ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

2.1. Обґрунтування моделі якості при виборі технологій front end розробки

Для формалізації процесу оцінювання якості технологій front end розробки необхідно для початку обґрунтувати модель якості, що містить відповідні набори як комплексних характеристик якості, так і елементарних атрибутів, які можна оцінити за допомогою кількісних метрик. Аналізуючи розвиток цих моделей, зокрема моделі Маккола, Боема, CUPRIDSMO та ін., можна зробити висновок про те, що результатом еволюції є стнадартизована модель, структуру і характеристики якої описані у стандарті ISO/IEC 25010.

Модель стандарту ISO/IEC 25010, як і ISO/IEC 9126, має ієрархічну структуру. На верхньому рівні ієрархії знаходяться характеристки якості, після цього іде рівень атрибутів, а на найнижчому рівні знаходяться метрики, які дають змогу кількісно виразити значення атрибутів. На рисунку 2.1 наведено структуру даної моделі.

$$H = \{H_1, H_2 ... H_n\}$$

$$\updownarrow \quad \updownarrow \qquad\qquad \updownarrow \quad \updownarrow$$

$$A_1 = \{A_{11}, A_{12}, ..., A_{1k}\} \qquad A_n = \{A_{n1}, A_{n2}, ..., A_{ns}\}$$

$$\updownarrow \quad \updownarrow \quad \updownarrow \qquad\qquad \updownarrow \quad \updownarrow \quad \updownarrow$$

$$\{M_{11}, M_{12}, ..., M_{1k}\} \ \ldots\ldots\ldots\ldots \ \{M_{n1}, M_{n2}, ..., M_{ns}\}$$

Рисунок 2.1 – Архітектура моделей якості

Стандарт ISO/IEC 25010 описує три моделі якості, що дає змогу оцінювати програмний продукт на трьох рівнях: рівень експлуатаційної якості, який відчувається кінцевим користувачем системи, рівень зовнішньої якості, який

описує якість архітектури програмного забезпечення та рівень внутрішньої якості, що дозволяє оцінити якість програмних модулів на фізичному рівні.

Для оцінювання технологій front end розробки програмного забезпечення пропонується скористатись моделлю, що використовується для оцінювання якості архітектури. Схематично її представлено на рисунку 2.2.

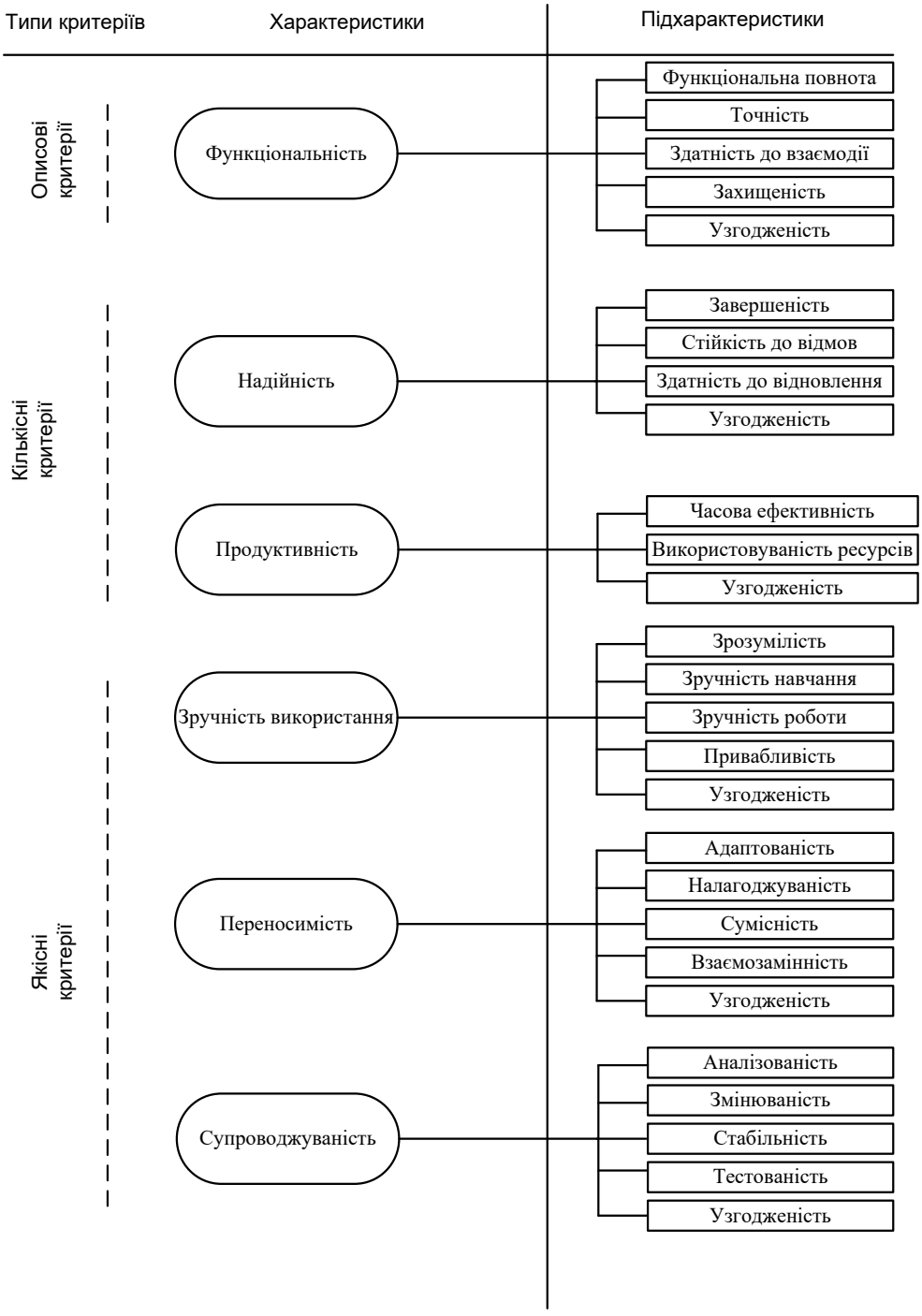


Рисунок 2.2 – Структура моделі зовнішньої якості

Кожна з наведених на рисунку 2.2 характеристик містить свій набір атрибутів, які можна кількісно оцінити. Основне призначення характеристик якості цієї моделі наступне:

1. Функціональність – інтерпретує здатність технологій front end розробки забезпечувати реалізацію деякого функціоналу.
2. Надійність – представляє собою комплексну сутність щодо підтримки нормального функціонування компонентів розробки при заданих умовах функціонування.
3. Продуктивність – визначає спроможність технологій реалізовувати деякий функціонал за критеріями апаратних ресурсів та часової ефективності.
4. Зручність використання – рівень зрозумілості та зручності щодо застосування інструментів front end розробки при реалізації web-додатків.
5. Переносимість – виражається здатністю технологій працювати у різних середовищах і забезпечувати однакову поведінку та функціональну стабільність.
6. Супроводжуваність – характеристика, що описує зручність і здатність підтримувати в актуальному та працездатному стані платформу front end розробки, можливість встановлення оновлення, контролю версію і т.п.

До цієї моделі пропонується додати ще одну характеристику – конфігурованість, яка описує здатність фреймворку чи платформи забезпечити ізольованість рівнів для проведення, наприклад, регламентних робіт.

Таким чином, для оцінювання якості технологій front end розробки, обгрунтовано модель зовнішньої якості, що містить 6 базових характеристик та додано сьому характеристику – конфігурованість.

Для кількісного вираження елементарних властивостей (атрибутів), які формують комплексні характеристики якості, необхідно проаналізувати та визначити відповідні набори метрик.

2.2. Визначення атрибутів та обґрунтування метрик якості при виборі технологій front end розробки

Для того, щоб провести вибір адекватної технології front end розробки, необхідно визначити сукупність атрибутів, які будуть описувати їх якість. До таких атрибутів належать:

– функціональна придатність – проявляється у здатності технології забезпечувати реалізацію передбачених функціональних вимог;

– точність – визначає міру відповідності одержаних результатів на запит користувача (розробника);

– здатність до взаємодії – атрибут, що описує можливість взаємодії з іншими технологіями та сторонніми сервісами;

– відповідність стандартам – може утворювати множину атрибутів щодо підтримки та відповідності визначеним стандартам галузі;

– захищеність – проявляється у наявності і здатності технології реалізувати процедури ідентифікації та авторизації користувачів;

– завершеність – оцінка функціоналу щодо правильної роботи компонентів технології front end розробки на визначеному проміжку часу;

– стійкість до відмов – проявляється у здатності інструментів технології забезпечувати певний рівень стабільної роботи в умовах виникнення непередбачуваних чи аварійних ситуацій;

– здатність до відновлення – описує спроможність системи щодо переходу в нормальний режим роботи протягом визначеного часу після її збою або відмови;

– конфігурованість визначеного набору параметрів – наявність інструментів, що необхідні для модифікації системи шляхом зміни значень параметрів з визначеного набору;

– конфігурованість визначеного набору базових об'єктів – наявність інструментарію для модифікації ПЗ, що досягається шляхом перекомпонування визначеного набору процесів, сутностей і службових процедур.

– конфігурованість реалізації нових базових об'єктів – атрибут, що описує здатність до забезпечення розширюваності набору процесів і сутностей;

– конфігурованість нової реалізації системи – наявність інструментів щодо встановлення і налаштування системи з нуля;

– відповідність стандартам надійності – атрибут щодо підтримки відповідності визначеним стандартам надійності;

– зрозумілість – атрибут (множина атрибутів), які описують здатність розробника щодо простоти використання інструментів технологій front end розробки для вирішення поставлених задач;

– зручність навчання – атрибут, який описує повноту документації та інших навчальних матеріалів і їхнє сприйняття розробником;

– зручність роботи – атрибут (атрибути), які описують зручність використання інструментів, які впливають на ефективність та продуктивність роботи розрозробників;

– привабливість – наявність компонентів та інструментів, які становлять цінність для кінцевих користувачів продукту і є привабливим для них;

– відповідність стандартам зручності використання – оцінювання відповідності стандартам ергономіки та зручності використання як розробниками, так і кінцевими користувачами;

– часова ефективність – атрибут, що описує здатність компонентів технологій front end розробки надавати вчасно очікувані результати за визначених для цього ресурсів даних;

– ефективність використання ресурсів – атрибут, або множина атрибутів, які описують залежність між функціоналом та ресурсами, які використовуються (процесорний час, об’єм пам’яті, швидкість передачі даних і т.п.)

– відповідність стандартам продуктивності – атрибут, що оцінює відповідність залучених ресурсів при забезпеченні необхідного рівня функціональності до стандартів, які існують в певній конкретній галузі;

– аналізованість – атрибут, що визначає придатність технологій front end до рев’ю коду, аналізу помилок або дефектів.

– зручність внесення змін – показник, що визначає затрати трудових ресурсів при внесенні змін у конфігурацію ПЗ;

– стабільність – здатність показувати сталі результати роботи протягом певного періоду часу і зворотний до ризику появи випадкових ефектів у випадку зміни компонентів чи оновлення системи;

– зручність перевірки – атрибут, який показує зручність тестування і є оберненим до трудозатрат при виконанні процесу відлагодження програмного забезпечення у випадку внесення змін та одержанні потрібних результатів;

– відповідність стандартам зручності супроводу – атрибут, що описує відповідність стандартизованим критеріям супроводу програмного забезпечення;

– адаптованість – атрибут щодо здатності технологій і програмного забезпечення працювати у різних середовищах із застосуванням наперед передбачених дій;

– зручність інсталяції – наявність механізмів щодо встановлення або розгортання інструментів розробки або самого ПЗ у визначеному середовищі;

– здатність до співіснування – атрибут, що описує здатність технології або ПЗ працювати в одному середовищі з іншими сервісами чи програмами і при цьому використовувати спільні ресурси;

– зручність заміни – представляє собою здатність до заміни одного ПЗ або компонентів іншими для розв’язання одних і тих задач;

– відповідність стандартам переносимості – оцінювання процесу переміщення ПЗ чи технології розробки у відповідності до наведених у стандарті критеріїв.

Для збільшення інформативності і групування характеристик якості базової моделі якості пропонується виділити дві категорії комплексних характеристик:

– Runtime – представляє групу характеристик з відповідним набором атрибутів, що описують проектовано систему (web-додаток).

– Design – характеристики, які стосуються особливостей використання технологій front end розробки на етапі створення програмного забезпечення.

На рисунку 2.3 наведено класифікацію характеристик якості, які належать до класу Runtime.

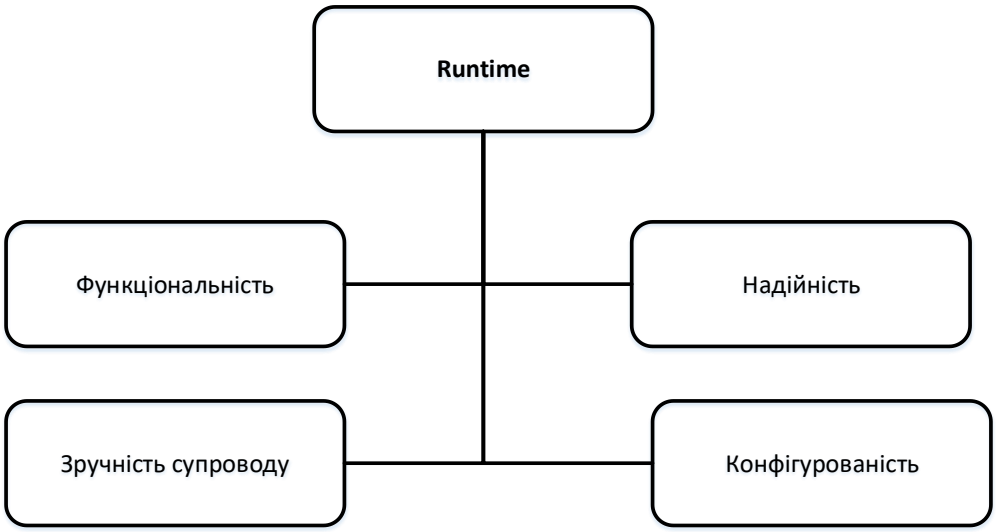


Рисунок 2.3 – Структура класу Runtime

Категорію Runtime формує 4 характеристики, які стосуються функціональності, надійності, здатності до конфігурації та зручності супроводу web-додатку. На рисунку 2.4 показано структуру класу Design.

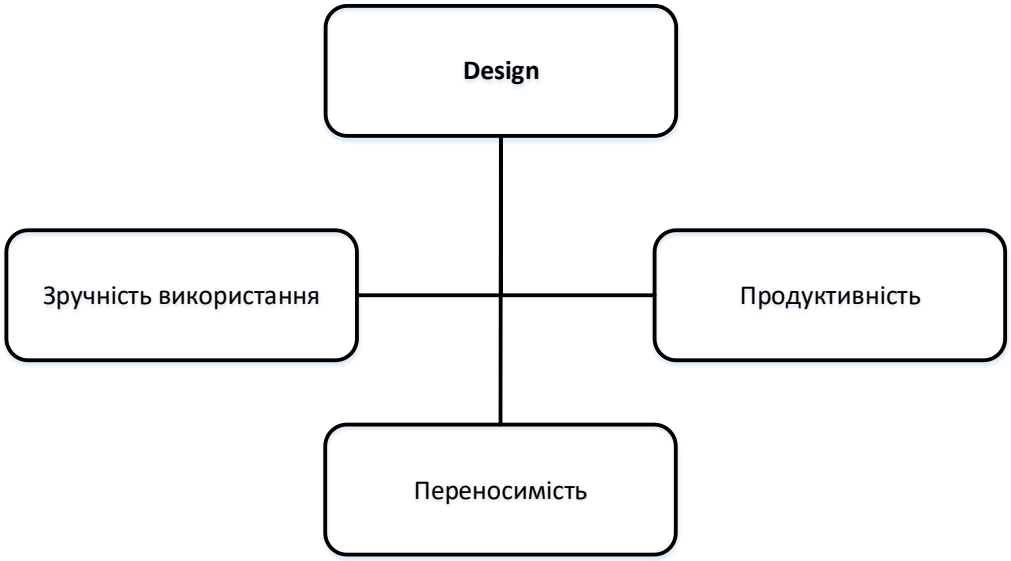


Рисунок 2.4 – Структура класу Design

Характеристику функціональність формують групи атрибутів, які наведено на рисунку 2.5.

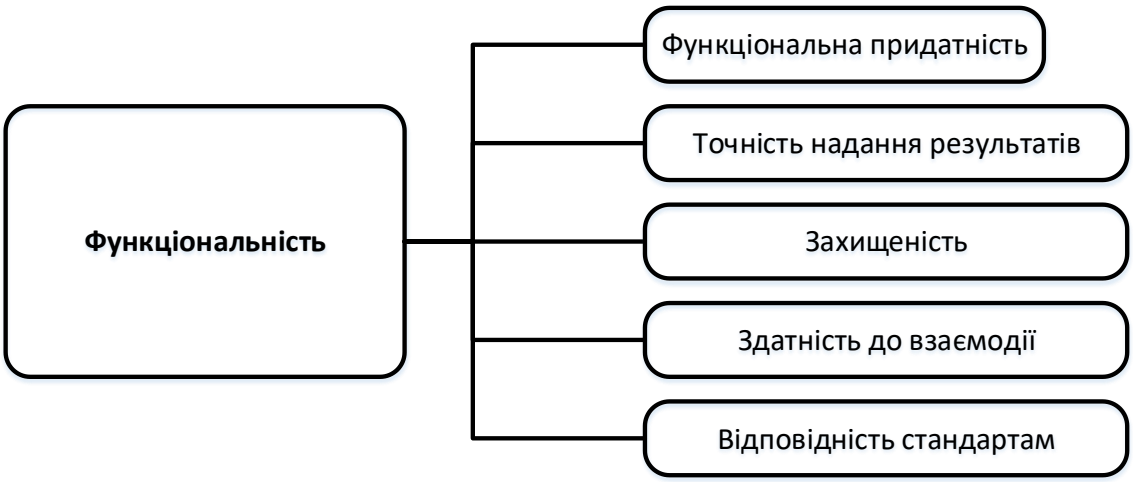


Рисунок 2.5 – Атрибути характеристики «Функціональність»

До характеристики надійність належить множина атрибутів, які показано на рисунку 2.6.

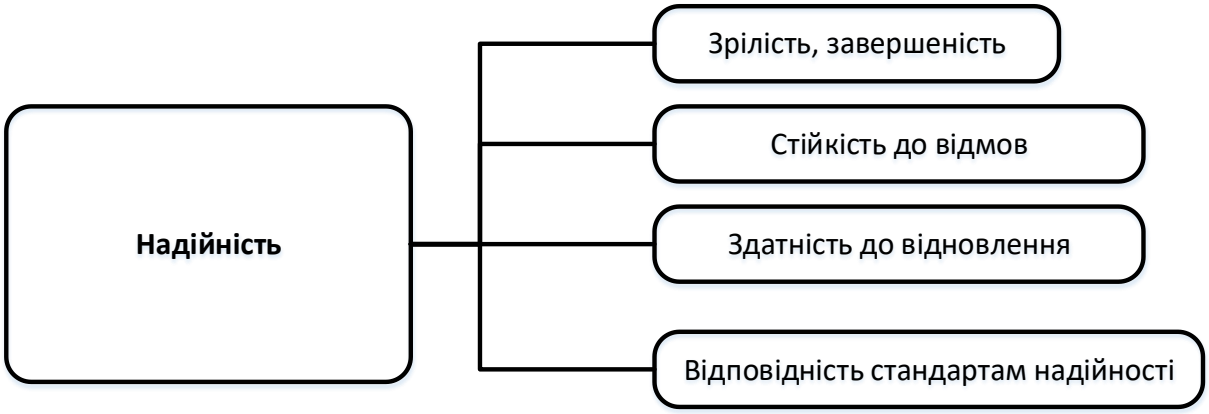


Рисунок 2.6 – Атрибути, що формують характеристику «Надійність»

На рисунку 2.7 показано структуру характеристики «Зручність використання».

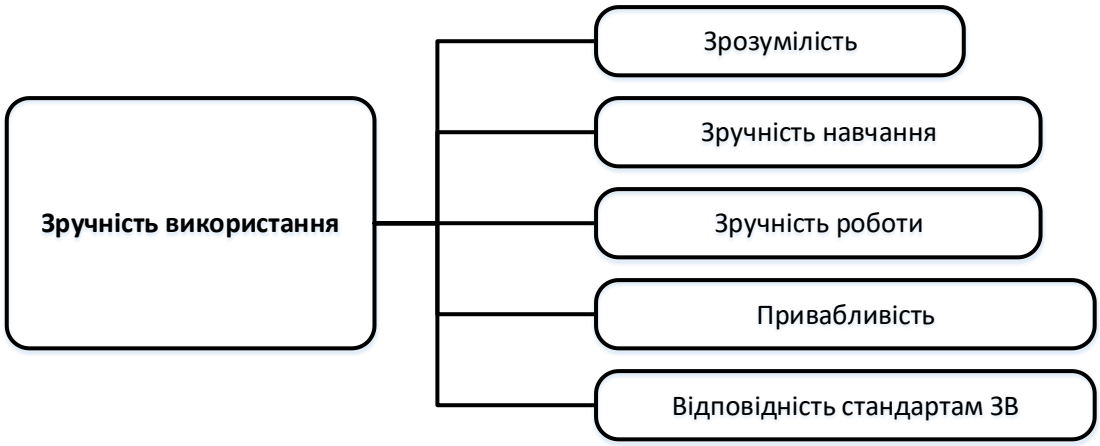


Рисунок 2.7 – Атрибути, що формують характеристику «Надійність»

На рис. 2.8 представлено структуру характеристики «Продуктивність», а на рис. 2.9 – «Зручність супроводу»

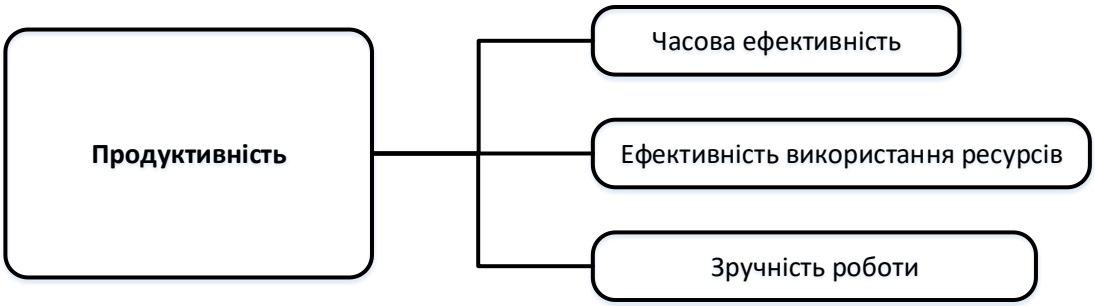


Рисунок 2.8 – Характеристика «Продуктивність»

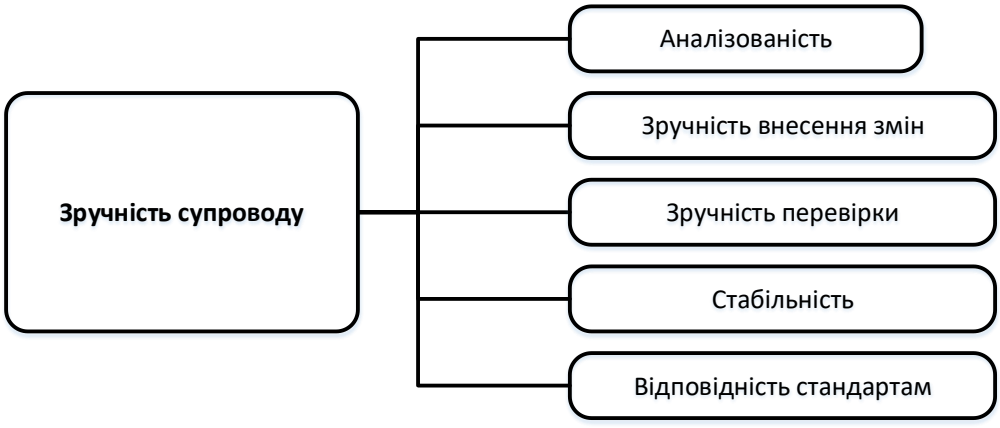


Рисунок 2.9 – Атрибути характеристики «Зручність супроводу»

Для представлення атрибутів якості за характеристиками «Переносимість» та «Конфігурованість» використано представлення, як показано на рисунку 2.10 та рисунку 2.11.

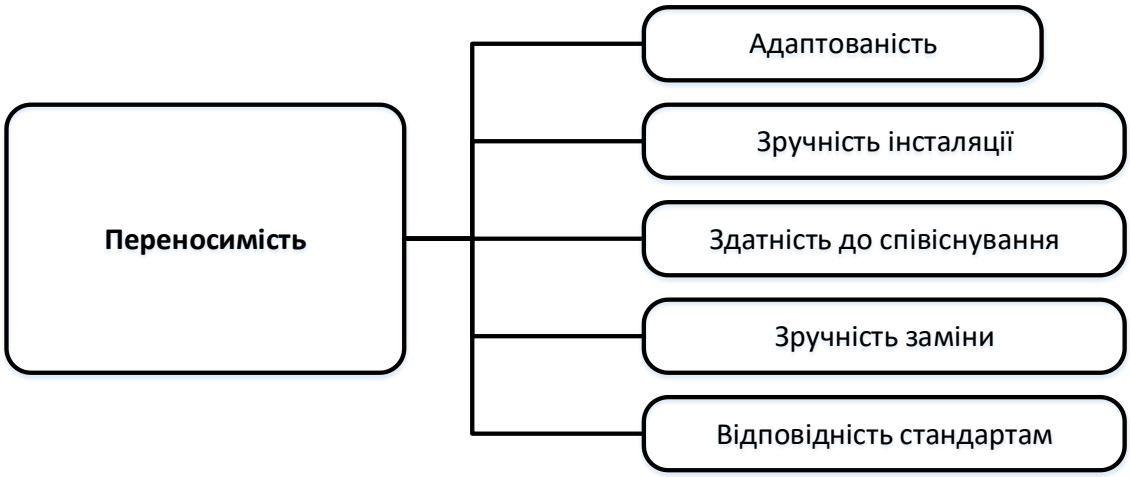


Рисунок 2.10 – Властивості характеристики «Переносимість»

Рисунок 2.11 – Атрибути характеристики «Конфігурованість»

У результаті проведеного дослідження визначено класи характеристик якості, доповнено модель якості характеристикою «Конфігурованість», встановлено атрибути для кожної з характеристик. Наступний крок полягає у дослідженні метрик для кількісного вираження якості для кожної групи, або для кожного атрибуту.

2.3. Обґрунтування метрик оцінювання атрибутів якості FE технологій

При оцінюванні атрибутів якості FE технологій можна скористатись стандартизованими наборами метрик, які наведені у ISO/IEC 25010 та ISO/IEC 9126. Однак більшість з низ орієнтовані для оцінювання готових програмних продуктів як загального користування, так і програмного забезпечення розробленого на замовлення конкретного замовника. Для технологій front end розробки їх доволі складно адаптувати і вони є не настільки інформативними в контексті їхнього оптимального застосування. Тому введемо для кожного набору атрибутів типи метрик, які можна застосувати на основі кількісних показників, зокрема розміру програмного коду, кількості унікальних операторів, цикломатичної складності та ін.

Для цього пропонується використати чотири категорії метрик:

– метрики Чепіна;
– метрики Холстеда;
– метрики Маккейба;
– метрики Джилба.

При аналізі метрик програмного забезпечення можливе застосування декількох різновидів метрик Чепіна. Як приклад метрики Чепіна можна розглянути оцінювання інформаційної міцності деякого програмного модуля. Така метрика базується на аналізі способу використання змінних при комунікації одного модуля з інших, тобто кількість параметрів, які передаються у модуль з інших зовнішніх компонентів, та кількість параметрів даного модуля, які є вхідними для сторонніх.

Метрики цикломатичної складності є основою для обчислення значень трудозатрат, часу і вартості виконання проекту, а також на їх базі можна сформувати потенційні ризики та прийняти ефективні управлінські рішення.

Найбільш широко використовуваними на практиці є метрики Холстеда. Вони включають набір оцінок, які кількісно характеризують величну операторів програмного модуля, стрічок коду та ін. На основі них можна формувати показники структурної складності програмного продукту, потоків керування та ряду інших.

Перевагою застосування метрик Холстеда є здатність до візуалізації результатів оцінювання за допомогою графу управління. Представниками метрик Холстеда, які описано у [9-11] є:

– кількість неповторюваних операторів (NUOprtr) – метрика, що формує словник унікальних операторів у програмному коді;

– кількість унікальних операндів (NUOprnd) – метрика, яка оцінює кількість неповторюваних, використаних у програмі, операндів;

Обчисливши загальну кількість операторів (NOprtr) та операндів (NOpnd), які існують у програмному забезпеченні, на основі їх співвідношення з NUOprtr та NUOprnd можна говорити про складність і доцільність використання тих чи інших технологій front end розробки. Зокрема, оперують такими поняттями, як повнота словника програмного забезпечення, його розмір, зрозумілість, модульність та ін.

Окрім, метрик Холстеда, на практиці часто застосовують метрики Маккейба, які характеризують цикломатичну складність програм. Даний показник може застосовуватись при оцінюванні окремих модулів, програмних методів чи інших структурних елементів програмного забезпечення. Суть використання даної метрики полягає у визначенні потрібної кількості проходження перевірки для забезпечення повноти покриття на структурі сильнозв’язного графу і дає змогу оцінити повноту тестових сценаріїв. У результаті застосування метрики Маккейба можна говорити про функціональну придатність FE технології, або програмного забезпечення, яке створено із їх застосуванням.

Провівши аналіз метрик і можливість їхнього застосування при оцінюванні якості та виборі оптимальної FE технології, необхідно визначити вагові коефіцієнти для кожного атрибуту якості. Таку процедуру пропонується виконати із застосуванням експертних оцінок. У таблиці 2.1 наведено повну структуру моделі оцінювання якості FE технологій із вказанням приналежності метрик до атрибутів та атрибутів до характеристик. Визначення вагових коефіцієнтів для кожного атрибуту якості моделі якості технологій front end розробки проводиться за допомогою експертної оцінки: C_n – характеристика, n – номер характеристики

моделі якості. $G_n[0;1]$ – ваговий коефіцієнт, n – номер атрибуту якості характеристики моделі якості.

Таблиця 2.1 – Повна модель якості при виборі технологій front end розробки

Характеристики	Атрибути	Метрики	Ваговий коефіцієнт
Функціональність	Функціональна придатність	Метрика Чепіна	0,7
	Точність	Метрика Холстеда	0,7
	Здатність до взаємодії	Метрика Джилба	0,9
	Відповідність стандартам	Метрика Холстеда	0,9
	Захищеність		0,9
Надійність	Завершеність	Метрика Холстеда	0,8
	Стійкість до відмов		1
	Здатність до відновлення		0,9
	Відповідність стандартам		0,9
Зручність використання	Зрозумілість	Метрика Холстеда	0,8
	Зручність навчання	Метрика Джилба	0,7
	Зручність роботи		0,9
	Привабливість	Метрика Холстеда	0,8
	Відповідність стандартам		0,9
Продуктивність	Часова ефективність	Метрика Чепіна	1
	Ефективність використання ресурсів		0,98
	Відповідність стандартам	Метрика Холстеда	0,9
Зручність супроводу	Аналізованість	Метрика Джилба	0,9
	Зручність внесення змін		0,9
	Стабільність		0,8
	Зручність перевірки		0,7
	Відповідність стандартам	Метрика Холстеда	0,9

Продовження таблиці 2.1.

Характеристики	Атрибути	Метрики	Ваговий коефіцієнт
Переносимість	Адаптованість	Метрика Джилба	0,9
	Зручність установки		0,7
	Здатність до співіснування		0,6
	Зручність заміни		0,8
	Відповідність стандартам	Метрика Холстеда	0,9
Конфігурованість	Конфігурованість визначеного набору параметрів	Метрика Холстеда	0,8
	Конфігурованість визначеного набору базових об'єктів		0,8
	Конфігурованість реалізації нових базових об'єктів	Метрика МакКейба	0,9
	Конфігурованість нової реалізації системи		0,8

Визначивши повністю усі компоненти, необхідно розробити процедуру та алгоритм оцінювання FE технологій.

2.4. Розробка процедури та алгоритму оцінювання при виборі FE технології

Процедура оцінювання та алгоритм її проведення представлено на рисунку 2.12. Вона передбачає формування вимог до оцінювання, яка включає в себе визначення цілей і мети оцінювання, ідентифікацію і класифікацію виду програмного забезпечення, яке проектується на основі FE технологій та специфікації характеристик якості.

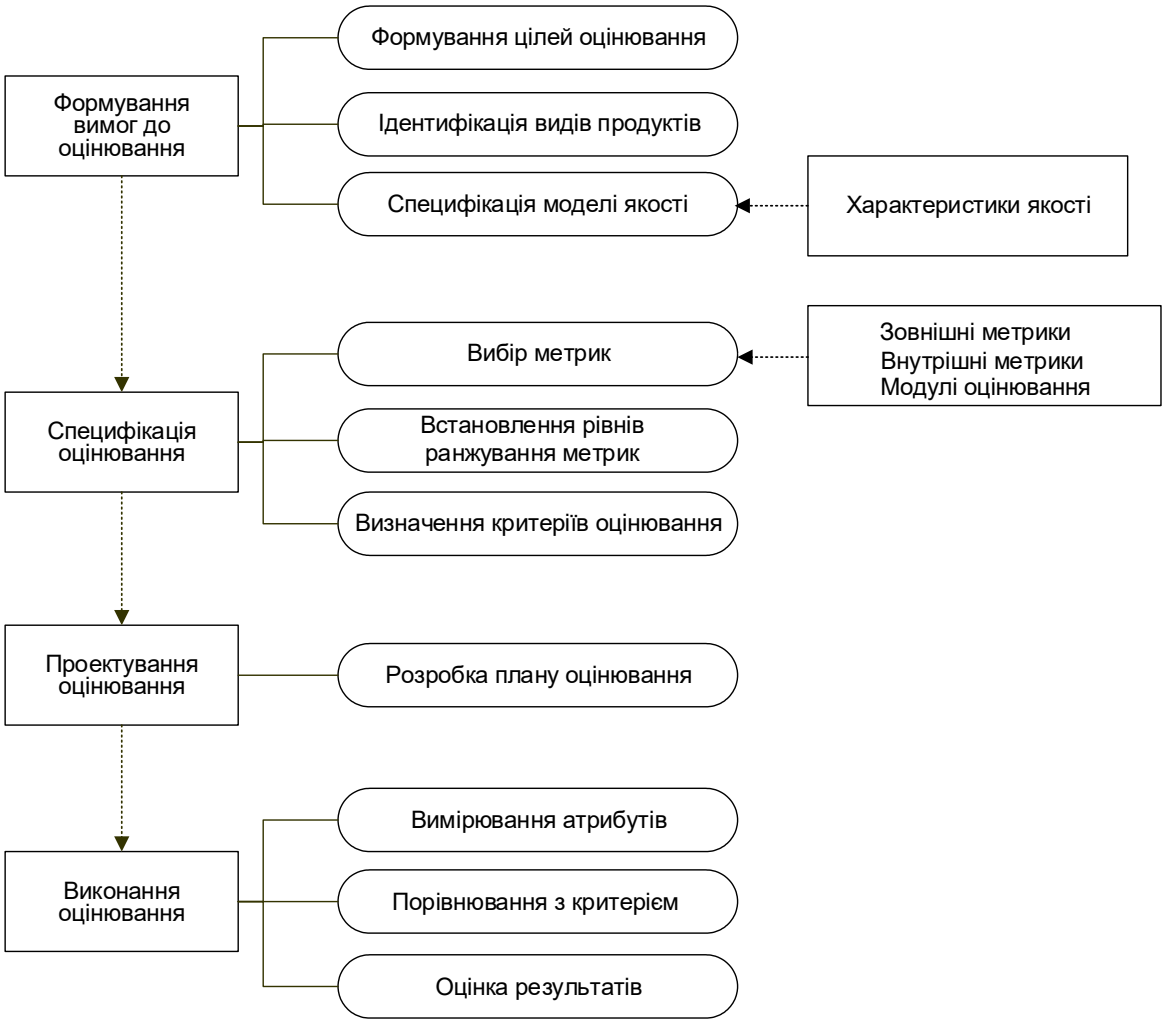


Рисунок 2.12 – Процедура оцінювання FE технологій

Наступний процес, який слідує за формуванням вимог до оцінювання, передбачає розробку специфікації. Специфікація оцінювання передбачає визначення атрибутів, які підлягають оцінюванню, вибір метрик і їхньої ваги для кожного атрибуту.

Третій етап загального процесу оцінювання FE технологій полягає у розробці плану оцінювання. План оцінювання включає в себе порядок проведення процедур кількісного вираження характеристик якості, підготовки середовища та відповідних даних.

Завершальною стадією процесу оцінювання є власне вимірювання показників якості на основі встановлених метрик та прийняття рішення щодо застосування конкретної технології.

Для характеристики «Функціональність» атрибут «Функціональна придатність» визначають за формулою (2.1), «Точність» (y_2), «Відповідність стандартам» (y_4) і «Захищеність» (y_5) за формулою (2.2), «Здатність до взаємодії» – за формулою (2.3).

$$y_1 = \mathrm{P} + 2 * \mathrm{M} + 3 * \mathrm{C} + 0.5 * \mathrm{T} \quad (2.1)$$

де y_1 – атрибут «Функціональна придатність»;

P – кількість змінних, що використовуються при обчисленнях та виводі інформації;

M – кількість змінних, які можуть бути модифікованими;

C – змінні, що виконують управління потоками;

T – змінні, що не використовуються у програмних модулях.

$$y_2 = (2\, n_2)/\,(n_1 N_2) \quad (2.2)$$

де y_2 – атрибут «Точність»;

n_2 – кількість унікальних операндів програми;

n_1 – кількість унікальних операторів програми;

N_2 – загальна кількість операндів у програмі.

$$y_3 = N4_{SV} \,/\, L_{mod} \quad (2.3)$$

де y_3 – атрибут «Здатність до взаємодії»;

$N4_{SV}$ – зв’язків між модулями;

L_{mod} – кількість модулів або підсистем;

Для оцінювання характеристики «Надійність», зокрема атрибутів «Здатність до відновлення» (y_6) та «Відповідність стандартам» (y_7) пропонується скористатись метрикою (2.2). Значення якості атрибуту «Завершеність» (y_8) обчислюють за формулою (2.4)

$$y_8 = \mathrm{V} \,/\, \mathrm{D} \tag{2.4}$$

де y_8 – атрибут «Завершеність»;

V – розмір програми.

D – трудозатрати на написання коду.

Атрибут «Стійкість до відмов» (y_9) обчислюється за наступною формулою

$$y_9 = N log_2 \mathrm{n} \tag{2.5}$$

де y_9 – атрибут «Стійкість до відмов»;

N – загальна кількість операторів та операндів програми;

n – словник програми (унікальні оператори та операнди).

Зручність використання описується атрибутами «Зрозумілість» (y_{10}), «Зручність навчання» (y_{11}), «Зручність роботи» (y_{12}), «Привабливість» (y_{13}) та «Відповідність стандартам» (y_{14}). Для обчислення значення y_{10} використаємо метрику, яка визначається наступним чином:

$$y_{10} = \mathrm{V} / L^* \tag{2.6}$$

де y_{10} – атрибут «Зрозумілість»;

V – розмір програми;

L^* – рівень якості програмування.

Атрибути y_{11} та y_{12} обчислюються на основі метрики (2.3), а y_{13} та y_{14} – на основі метрики (2.2).

Для обчислення значень атрибутів характеристики «Продуктивність» використовуються метрики (2.1) – для атрибутів «Часова ефективність» (y_{15}) та «Ефективність використання ресурсів» (y_{16}), метрика (2.2) – для оцінки атрибуту «Відповідність стандартам» (y_{17}).

Чотири атрибути характеристики «Зручність супроводу»: «Аналізованість» (y_{18}), «Зручність внесення змін» (y_{19}), «Стабільність» (y_{20}) та «Зручність перевірки» (y_{21}) обчислюються на основі метрики (2.3). Атрибут «Відповідність стандартам» (y_{22}) – на основі метрики (2.2).

По аналогії до характеристики «Зручність супроводу», атрибути характеристика «Перносимість»: «Адаптованість» (y_{23}), «Зручність інсталяції» (y_{24}), «Здатність до співіснування» (y_{25}) та «Зручніть заміни» (y_{26}) описуються метрикою (2.3), а «Відповідність стандартам» – метрикою (2.2).

Атрибути «Конфігурованість визначеного набору параметрів» (y_{27}) характеристики «Конфігурованість» можна оцінити за допомогою наступної метрики

$$y_{27} = 1/\ L^* \tag{2.7}$$

де L^* - рівень якості програмування ($L^* = (2\ \mathrm{n2})/\ (\mathrm{n1} * \mathrm{N2})$, $\mathrm{n2}$ – словник операндів, $\mathrm{n1}$ – словник ператорів, $\mathrm{N2}$ – загальна кількість операндів.

Метрику (2.7) можна застосувати і для оцінювання атрибуту «Конфігурованість визначеного набору базових об'єктів» (y_{28}).

Атрибут «Конфігурованість реалізації нових базових об'єктів» (y_{29}) обчислюється за допомогою метрики Маккейба

$$y_{29} = \lambda\ (\mathrm{G})\ + 1\ =\ \mathrm{m}\ -\ \mathrm{n}\ +\ 2p \tag{2.8}$$

де $\lambda\ (\mathrm{G})$ – кількість лінійно незалежних контурів у сильнозв’язному графі управління програмного забезпечення;

m – кількість дуг графу;

n – кількість вершин графу;

p – кількість компонент зв’язності.

Для кількісної оцінки атрибуту «Конфігурованість нової реалізації системи» ((y_{30}) також використовується метрика (2.8).

Визначенні вище метрики якості FE технологій дають змогу кількісно виразити значення атрибутів, однак вони нічого не говорять про те, чи достатній рівень якості кожного з них. Тому для ранжування рівня якості вводиться коефіцієнт прийнятності, який можна обчислити наступним чином

$$k_i = \begin{cases} 1, \text{якщо } x \to x_{max} \\ 0.6, \text{якщо } \frac{x_{max}}{2} \leq x < x_{max} \\ 0.4, \text{якщо } \frac{x_{max}}{3} \leq x < \frac{x_{max}}{2} \\ 0, \text{в іншому випадку} \end{cases} \quad (2.9)$$

В загальному випадку, оцінку якості атрибуту FE технології, запропоновано обчислювати за формулою

$$Q(Attr_i) = k_i G_i y_i \quad (2.10)$$

де $Q(Attr_i)$ – оцінка якості атрибуту $Attr_i$, $i = 1..M$ – кількість атрибутів якості;

k_i – коефіцієнт прийнятності і-го атрибуту;

G_i – ваговий коефіцієнт і-го атрибуту;

y_i – метрика і-го атрибуту;

Оскільки, оцінка якості атрибуту є елементарною, то для представлення якості за конкретною характеристикою запропоновано скористатись наступною формулою

$$Q(Ch_j) = \frac{1}{C}\sum_{j=1}^{C} Q(Attr_i) \quad (2.11)$$

де $Q(Ch_j)$ – значення якості за характеристикою Ch_j, $j = 1..7$ – кількість характеристик;

C – кількість атрибутів характеристики Ch_j;

Для представлення інтегрального показника якості FE технології запропоновано використати формулу

$$\mathrm{Q}(FE) = \frac{1}{D}\sum_{j=1}^{K} Q(Ch_j) \qquad (2.12)$$

де $\mathrm{Q}(FE)$ – інтегральний показник якості FE технології;

D – кількість характеристик моделі якості технології FE.

Таким чином, побуддовано модель та розроблено метод оцінювання якості FE технологій розробки програмного забезпечення, орієнтованого під web-простір.

Основні наукові і практичні результати, які одержано у даному розділі полягають в наступному:

1. Для представлення якості технологій front end розробки ПЗ для web-простору обгрунтовано застосування моделі якості, яка є стандартизиваною згідно ISO/IEC 25010, володіє ієрархічною структурою і доповнено її характеристикою «Конфігурованість», що дало змогу більш ширше та адекватніше описати властивості FE технологій, порівнюючи з іншими моделями.

2. Визначено і класифіковано за характеристиками елементарні властивості якості FE технологій, що дало змогу обчислювати їх кількісне значення на основі як стандартних метрик якості, так і метрик коду програмного забезпечення.

3. Обгрунтовано застосування метрик Чепіна, Холстеда, Джилба та МакКейба для оцінювання атрибутів якості технологій FE розробки, що дало змогу спростити сприйняття та підвищити зрозумілість кількісних значень якості відповдіних технологій.

4. Запропоновано процедуру та реалізовано алгоритм оцінювання якості FE технологій на різних рівнях ієрарахічного дерева моделі якості, що дало змогу керувати як метриками оцінювання, так і процесом прийняття рішень щодо якості елементарних критерів, характеристик та інтегральної якості.

3 ПРОГРАМНИЙ ЗАСІБ УПРАВЛІННЯ МЕТРИКАМИ ЯКОСТІ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

3.1. Визначення та аналіз вимог до програмного засобу управління метриками при оцінюванні технологій front end розробки

Визначення вимог до програмного засобу це одна з невід'ємних складових загального процесу розробки ПЗ. Отже, перед початком розробки такого web-додатку слід визначити вимоги, яким повинна відповідати розроблювана система.

Розроблений web-додаток повинен надавати наступні можливості:

- реєстрацію та авторизацію користувачів;
- зберігання та обробку інформації;
- управління компонентами моделі якості, зокрема метриками якості;
- автоматизацію процесу визначення якості технологій FE розробки;
- надійний канал передачі даних;
- безпеку і конфіденційність інформації про користувача.

Доступ до основних функцій сервісу буде надаватися тільки зареєстрованим та авторизованим користувачам відповідно до їх прав в системі. Не зареєстрований користувач не буде мати можливості використовувати основні функції web-додатку.

Також, однією з основних вимог є зручний та інтуїтивно зрозумілий інтерфейс користувача, який надавав би усі необхідні користувачу інструменти при проведенні оцінювання якості та виборі FE технологій розробки ПЗ, а також керувати власним профілем.

Щоб зберігати напрацювання користувачів необхідно спроектувати та налаштувати документо – орієнтовану базу даних для збереження всієї необхідної інформації для роботи в системі, доступ до якої буде здійснюватися через систему керування базою даних(СКБД).

Основними вимогами, що висуваються до бази даних є такі:

- мінімальна надлишковість даних;

- забезпечення цілісності бази даних;
- авторизований доступ до бази даних;
- несуперечність даних;
- прозорість доступу до даних.

Розроблюваний web-дотаток повинен надавати можливість подальшої модернізації шляхом масштабування.

Найбільш важливим параметром для будь-якого програмного забезпечення та web-сторінок є безпека їхньої експлуатації та захист від несанкціонованого доступу до даних користувача.

У web-додатку управління метриками якості, показники цілісності та доступності покладено на СКБД, а для забезпечення конфіденційності та моніторингу будуть використані механізми ідентифікації та автентифікації.

Також одним з важливих кроків вдалого проектування системи є правильне визначення сутностей і атрибутів предметної області.

Правильне визначення сутностей та атрибутів дозволяє зменшити надлишковість даних при їх зберіганні та використанні, а також покращує розуміння предметної області для якої будується база даних.

Сутностями web-додатку підтримки процесу вибору FE технологій є компоненти моделі якості, зокрема атрибути, характеристики і метрики.

Деякі сутності, які містять невелику кількість атрибутів в даній предметній області, але все ж їх краще винести як окремі сутності, а не вносити до переліку атрибутів іншої. Такий підхід дозволяє зменшити надлишковість даних, а також покращити їхню структурованість. Тепер не потрібно дублювати одинакові атрибути в різних сутностях, що значно підвищує продуктивність при роботі з базою даних. В теорії баз даних це називається нормалізацією бази даних.

Повний перелік сутностей разом з їхніми атрибутами подано у таблиці 3.1.

Таблиця 3.1 – Сутності та атрибути предметної області

№ з/п	Сутність	Атрибути
1	Якість	Дата
		Результат
2	Характеристика моделі якості	Назва
		Атрибути якості
3	Атрибут якості	Назва
		Значення
4	Користувач	Ім’я
		Електронна пошта
		Пароль
		Роль
5	Роль	Guest
		User
		Admin

В UML актори зображуються так, як показано на рисунку 3.1, вони можуть бути зображені у вигляді піктограми класу із зазначенням стереотипу «Actor» або у вигляді піктограми «анімаційний чоловічок». Допускаються обидві форми, але багато розробників моделей використовують «чоловічка» для тих ролей, які, швидше за все, будуть виконуватися людьми, а піктограми класу для ролей, які будуть відображати інші системи .

Користувач

Рисунок 3.1 – Варіанти позначення актора

Прецедент — це дія, яку повинна виконувати система за зверненням актора. Це «варіант використання» системи конкретним актором:

– прецеденти завжди створюються актором;

– прецеденти завжди описуються з точки зору акторів.

Зазвичай, прецеденти розглядаються на рівні системи, але згідно визначенням вони можуть застосовуватися також і для опису «варіанту використання »підсистеми (частини системи) або навіть окремого класу [14].

На рисунку 3.2 показана піктограма UML для прецедентів. Ім'я прецеденту може бути написано усередині овалу або під ним.

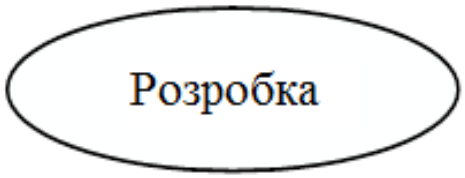


Рисунок 3.2 – Приклад піктограми прецеденту

Щоб побудувати діаграму прецедентів, необхідно визначитися з типами акторів, які будуть взаємодіяти з системою, а також описати прецеденти, що будуть доступними кожному типу актора.

Перелік ролей та їх прецедентів наведено у таблиці 3.2.

Таблиця 3.2 – Перелік ролей та прецедентів

№ з/п	Актор	Прецеденти
1	Guest	Реєстрація
		Вхід в систему
2	User	Вхід в систему
		Зміна персональних даних
		Робота з системою

Продовження таблиці 3.2

№ з/п	Актор	Прецеденти
3	Admin	Вхід в систему
		Робота з системою
		Керування атрибутами
		Керування користувачами

На рисунку 3.3 наведено побудовану use case діаграму, яка інтерпретує та візуалізує функціональні можливості розроблюваного засобу управління метриками FE технологій.

Рисунок 3.3 – Діаграма прецедентів «Система управління метриками якості»

У випадку web-додатку управління метриками FE технологій можна виділити наступні види акторів: щойно зареєстрований користувач, гість, адміністратор. Для

всіх акторів будуть доступні прецеденти входу в систему та зміни персональних даних, усі інші допустимі варіанти використання будуть розподілені за ролями.

3.2. Побудова архітектури та розробка алгоритму роботи системи управління метриками якості FE технологій

Проектований web-додаток призначений для автоматизації, оптимізації та забезпечення адекватності оцінювання якості FE технологій розробки ПЗ на основі запропонованої моделі якості.

Схема типової архітектури web-додатку, заснованого на взаємодії його основних компонентів показаний на рисунку 3.4.

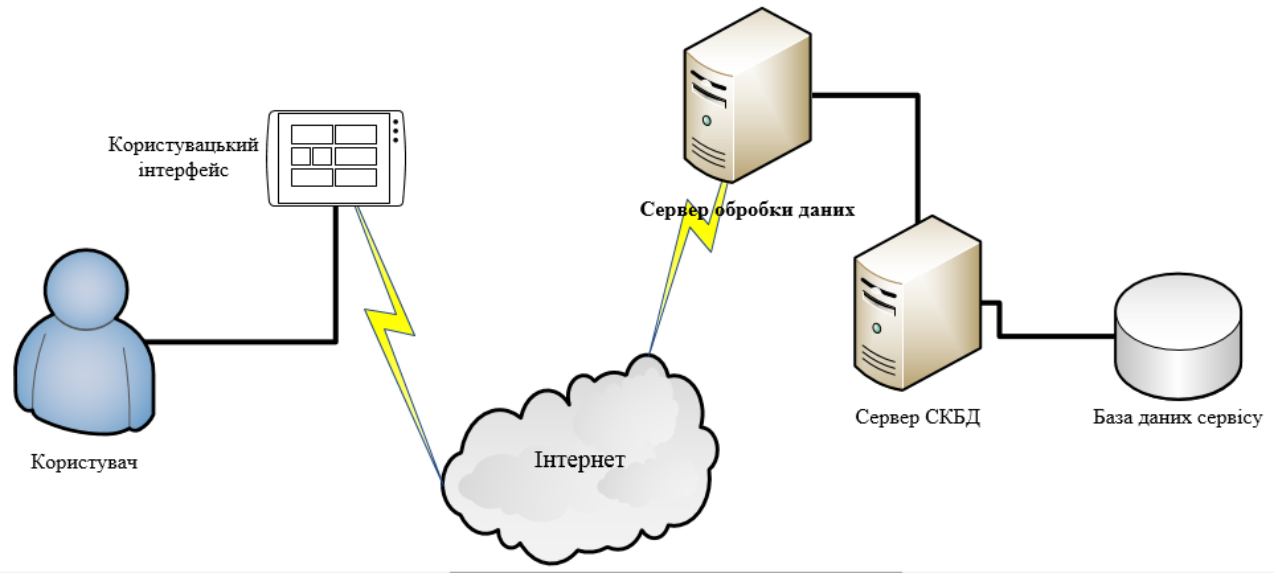


Рисунок 3.4 – Типова архітектура web-додатку

Типова архітектура відображає повну взаємодію користувача з системою в цілому, а також зв'язок всіх компонентів в системі між собою.

Згідно типової архітектури, показаної на рисунку 3.4 випливає наступний принцип взаємодії компонентів web-додатку:

– користувач взаємодіючи з користувацьким інтерфейсом викликає певні події;

– запит події через мережу Інтернет потрапляє на сервер обробки даних;

– сервер обробки даних обробляє отриманий запит і/або формує відповідь та надсилає її назад клієнтові, і/або звертається до сервера СКБД;

– якщо до сервера СКБД надійшов запит, то відбувається його взаємодія з базою даних та відправка даних у відповідь;

– дані, що надійшли з сервера СКБД потрапляють на сервер обробки даних, який їх структурує та відправляю назад користувачеві;

– відповідь проходить через мережу Інтернет та відображається на користувацькому інтерфейсі.

Враховуючи загальну концепцію проектування web-додатків, архітектуру програмного засобу управління метриками якості побудовано у відповідності до вимог та наведено на рисунку 3.5.

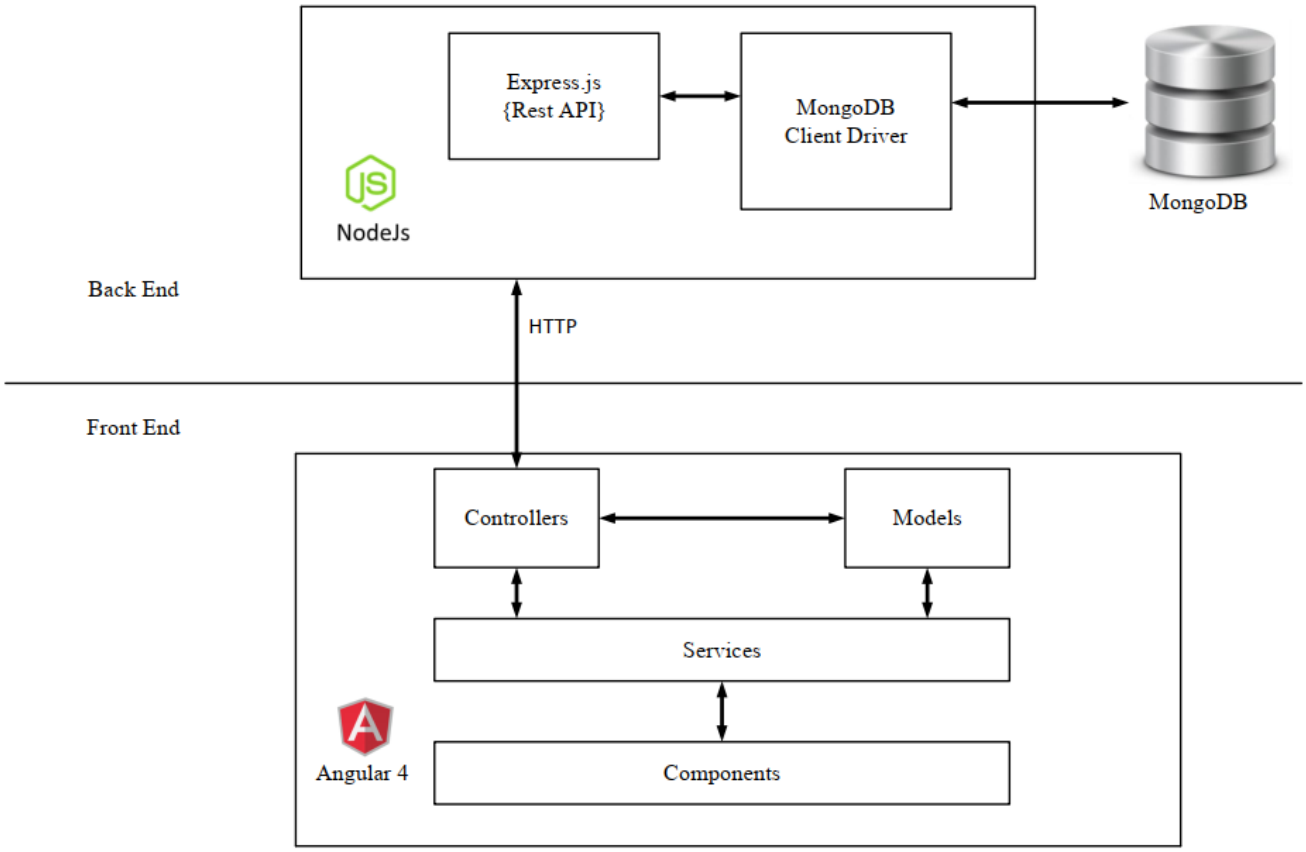


Рисунок 3.5 – Архітектура розроблюваного програмного засобу

Як видно з рис. 3.5, архітектура додатку складається з двох частин: front end та back end. Front end частина реалізується за допомогою фреймворку Angular 4, в основі якого лежить об’єктно-орієнтований патерн проектування MVC.

Back end містить механізм Rest API та документо-орієнтовану базу даних Mongo DB. Обидві складові реалізуються за допомогою платформи Node JS. Взаємодія між Front end та Back end виконується за допомогою HTTP протоколу.

У процесі аналізу функціональних вимог та спроектованої архітектури, розроблено алгоритм роботи програмного засобу управління метриками якості FE технологій, який наведено на рисунку 3.6.

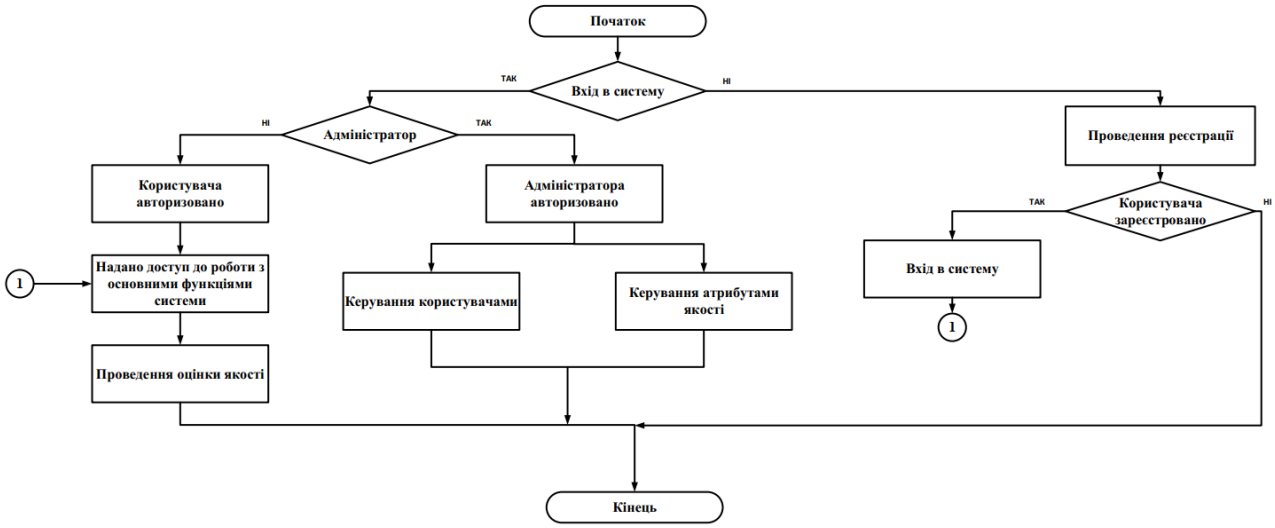


Рисунок 3.6 – Алгоритм роботи програмного засобу управління метриками якості

Як видно з побудованого алгоритму, він враховує усі аспекти, що були представлені у вигляді use case діаграми (рис. 3.3) та архітектури програмного засобу (рис. 3.5).

Для реалізації запропонованої архітектури та алгоритму необхідно обрати та налаштувати середовища написання програмного коду. Послідовність виконання дій наступна:

– встановити середовище розробки програмного забезпечення WebStorm;

– встановити і налаштувати СКБД MongoDB [18];

– створити новий проект під назвою Qualitizer;

– створити файл з набором необхідних пакетів для розробки ПЗ package.json (додаток Б);

– за допомогою команди npm install встановити усі необхідні файли;

– створити директорію для клієнтської частини client та добавити в корінь цієї директорії інші необхідні директорії, які використовуються на клієнтській стороні;

– створити директорію для серверної частини server;

– розробити усі необхідні компоненти для проведення оцінювання якості FE технологій.

Після виконання вище зазначених кроків можна переходити до фази програмування. Для початку необхідно розробити сервер використовуючи платформу NodeJS, та фреймворк Express. Файл app.ts з серверною частиною розміщується в директорії Server. В лістингу 3.1 розміщено код серверної частини веб – додатку. Сервер повинен приймати запити від клієнта опрацьовувати їх і надавати відповідь клієнту, зокрема, за допомогою сервера буде здійснюватись підключення до бази даних MongoDB.

Лістинг 3.1 – Програмний код сервера

```
import * as bodyParser from 'body-parser';
import * as dotenv from 'dotenv';
import * as express from 'express';
import * as morgan from 'morgan';
import * as mongoose from 'mongoose';
import * as path from 'path';
import setRoutes from './routes';
const app = express();
dotenv.load({ path: '.env' });
app.set('port', (process.env.PORT || 3000));
app.use('/', express.static(path.join(__dirname, '../public')));
```

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
let mongodbURI;
if (process.env.NODE_ENV === 'test') {
  mongodbURI = process.env.MONGODB_TEST_URI;
} else {
  mongodbURI = process.env.MONGODB_URI;
  app.use(morgan('dev'));
}
mongoose.Promise = global.Promise;
const mongodb = mongoose.connect(mongodbURI, { useMongoClient: true });
mongodb
  .then((db) => {
    console.log('Connected to MongoDB on', db.host + ':' + db.port);
    setRoutes(app);
    app.get('/*', function(req, res) {
      res.sendFile(path.join(__dirname, '../public/index.html'));
    });
    if (!module.parent) {
      app.listen(app.get('port'), () => {
        console.log('Angular Full Stack listening on port ' + app.get('port'));
      });
    }
  })
  .catch((err) => {
    console.error(err);
});

export { app };
```

Розробивши серверну частину, далі необхідно реалізувати логіку входу та реєстрації нового користувача. Для початку слід розробити модель користувача User.ts. У лістингу 3.2 наведено програмну реалізацію моделі користувача.

Лістинг 3.2 – Модель користувача

```
 export class User {
  _id: string;
  username: string;
  email: string;
```

```
  role?: string;
}
```

Наступним кроком потрібно розробити компонент для реєстрації register.component, компонент для логінізації login.component, logout.cocmponent. Лістинги даних компонентів приведено у додатку Б.

Розробивши компоненти потрібно розробити сервіс user.service.ts який буде проміжною ланкою між сервером та компонентами і призначений для роботи з маніпуляцією нових користувачів. Лістинг user.service.ts приведено у лістингу 3.3.

Лістинг 3.3 – Сервіс для керування користувачами

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { User } from '../shared/models/user.model';
@Injectable()
export class UserService {
  constructor(private http: HttpClient) { }
  register(user: User): Observable<User> {
    return this.http.post<User>('/api/user', user);
  }
  login(credentials): Observable<any> {
    return this.http.post<any>('/api/login', credentials);
  }
  getUsers(): Observable<User[]> {
    return this.http.get<User[]>('/api/users');
  }
  countUsers(): Observable<number> {
    return this.http.get<number>('/api/users/count');
  }
  addUser(user: User): Observable<User> {
    return this.http.post<User>('/api/user', user);
  }
  getUser(user: User): Observable<User> {
    return this.http.get<User>(`/api/user/${user._id}`);
  }
  editUser(user: User): Observable<string> {
   return this.http.put(`/api/user/${user._id}`, user, { responseType: 'text' });}
```

```
deleteUser(user: User): Observable<string> {
return this.http.delete(`/api/user/${user._id}`, { responseType: 'text' });  } }
```

Для входу в систему і аутентифікації користувача потрібно розробити ще 3 сервіси:

– auth.service.ts – відповідатиме за вхід/вихід з системи, визначення поточного користувача в системі;

– auth.guard.admin.service.ts – призначений для визначення адміністратора системи та активації сторінок призначених лише для Адміністратора системи;

– auth.guard.login.service.ts – призначений для визначення успішного входу в систему користувача і активації необхідних вкладок які призначені лише для простого користувача.

Лістинг розроблених сервісів приведено в додатку Б.

Наступний етап – розробити компонент який призначений лише для користувача з роллю Адміністратор і виконуватиме такі завдання, як додавання, видалення, редагування параметрів якості на основі яких буде проводитись оцінювання якості front end розробки програмного забезпечення, також за допомогою логіки розроблюваного компоненту можна буде керувати зареєстрованими користувачами. Фрагмент коду компоненту admin.component наведено у додатку Б.

Розробивши компоненти системи, які слугуватимуть для підготовки даних на основі яких буде проводитись оцінювання якості front end розробки, необхідно розробити компонент quolity.component який буде виконувати власне саму оцінку якості на основі приведених даних (додаток Б).

Одним з найважливіших кроків для запуску програми є проведення конфігурування файлу app.module.ts, в якому проводиться імпорт усіх розроблених компонентів системи. Без коректного конфігурування даного файлу запуск системи просто не відбудеться так як в нього за допомогою механізму Dependency Injection інкапсулюються всі компоненти які потрібні для роботи системи. Код сконфігурованого файлу app.module.ts наведено у лістингу 3.4.

Лістинг 3.4 – Код конфігураційного файлу app.module.ts

```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { RoutingModule } from './routing.module';
import { SharedModule } from './shared/shared.module';
import { QuolityService } from './services/quolity.service';
import { UserService } from './services/user.service';
import { AuthService } from './services/auth.service';
import { AuthGuardLogin } from './services/auth-guard-login.service';
import { AuthGuardAdmin } from './services/auth-guard-admin.service';
import { AppComponent } from './app.component';
import { QuolityComponent } from './quolity/quolity.component';
import { AboutComponent } from './about/about.component';
import { RegisterComponent } from './register/register.component';
import { LoginComponent } from './login/login.component';
import { LogoutComponent } from './logout/logout.component';
import { AccountComponent } from './account/account.component';
import { AdminComponent } from './admin/admin.component';
import { NotFoundComponent } from './not-found/not-found.component';
@NgModule({
  declarations: [
    AppComponent,
    QuolityComponent,
    AboutComponent,
    RegisterComponent,
    LoginComponent,
    LogoutComponent,
    AccountComponent,
    AdminComponent,
    NotFoundComponent
  ],
  imports: [
    RoutingModule,
    SharedModule
  ],
  providers: [
    AuthService,
    AuthGuardLogin,
    AuthGuardAdmin,
    QuolityService,
    UserService
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
```

```
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Розробивши всі необхідні компоненти програмного засобу управління метриками технологій front end розробки перейдемо до проектування користувацьких інтерфейсів і тестування системи.

3.3. Розробка користувацьких інтерфейсів і тестування програмного засобу управління метриками якості FE технологій

Перед початком роботи в системі управління метриками якості FE технологій необхідно здійснити вхід у систему, якщо користувач вже зареєстрований. Вікно входу в систему зображено на рисунку 3.7.

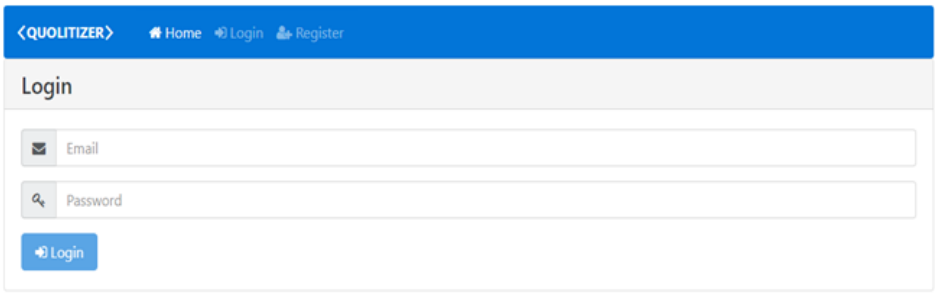


Рисунок 3.7 – Вхід у систему

Якщо користувача успішно аутентифіковано, то відкриється домашня сторінка сервісу, яка показана на рисунку 3.8.

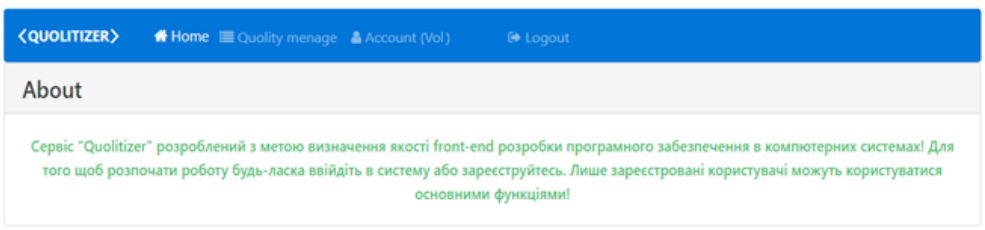


Рисунок 3.8 – Успішний вхід в систему

Для того, щоб відкрити вкладку управління параметрами якості, необхідно натиснути клавішу + , що знаходиться біля напису Quality parameters. Натиснувши на дану кнопку панель для керування параметрами якості стане доступною. Щоб додати нову характеристику моделі якості, необхідно перейти до вкладки “Add new quality parameter”, ввести назву створюваної характеристики, а також додати атрибути якості, які будуть міститись в даній характеристиці. Після цього натиснути кнопку “Save quality characteristic”. Новостворену характеристику буде записано в базу даних. На рисунку 3.9 наведено панель управління параметрами якості.

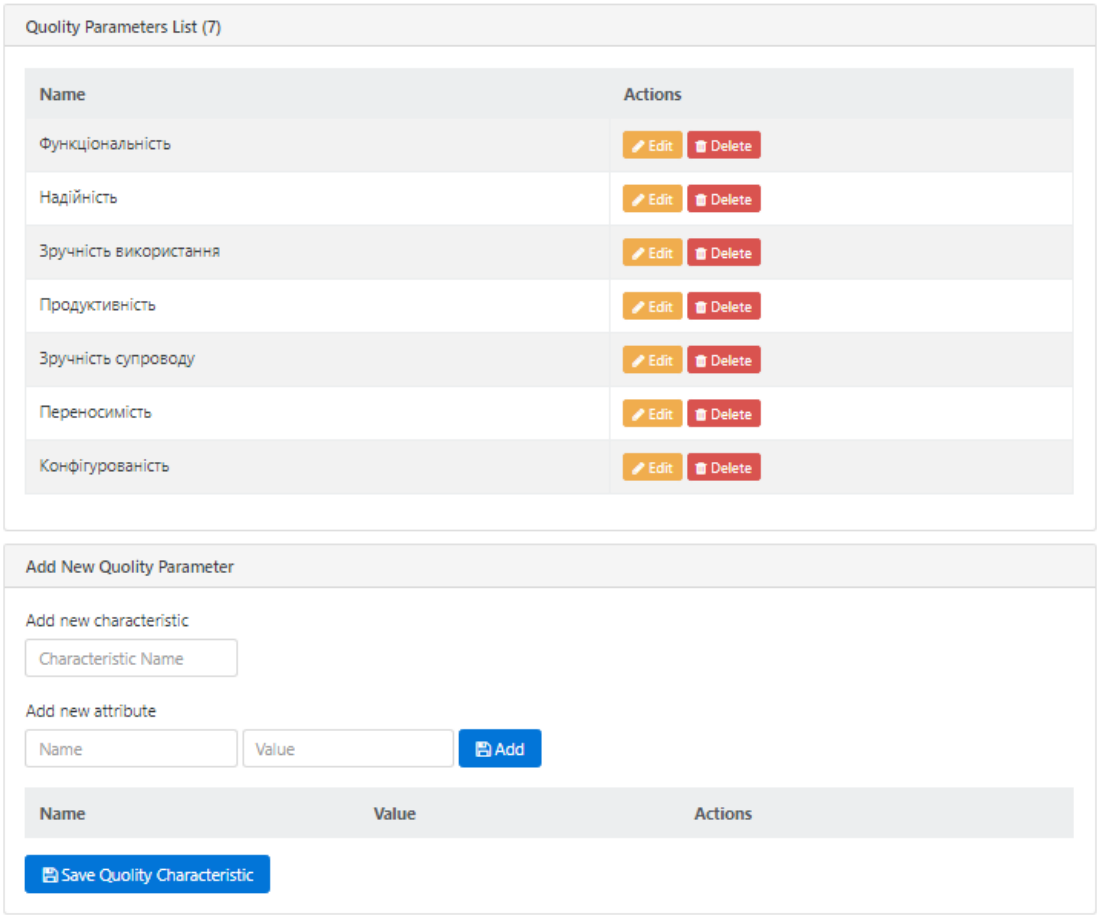


Рисунок 3.9 – Панель управління параметрами якості

Для того, щоб вийти з системи потрібно в навігаційній панелі натиснути кнопку Logout.

Процедура, яку повинен виконати користувач для оцінювання якості технології FE передбачає виконання наступних кроків:

– увійти в систему;

– перейти на вкладку “Quality Manage”;

– обрати характеристику зі списку наявних характеристик;

– обрати атрибути, що містить характеристика;

– обравши атрибути, перейти на панель “Results” та ввести необхідні відомості про оцінювану систему;

– два попередні пункти повторити стільки разів, скільки міститься в списку характеристик для того, щоб отримати максимально точну оцінку;

– пройшовши всі перелічені кроки, натиснути кнопку завершення оцінювання якості і після цього буде виведено результат у відсотках.

Визначивши всі кроки, які слід пройти для того, щоб оцінити якість технології FE можна переходити до більш детального тестування розробленої системи оцінювання якості. Тестування web – додатка оцінювання якості front end розробки програмного забезпечення. Для того, щоб здійснити оцінювання якості, користувачу слід спочатку увійти в систему і отримати певні права для роботи з системою. Увійшовши успішно в систему, необхідно обрати пункт меню “Quality manage”. Після цього відкриється сторінка для роботи з оцінюванням якості, на якій буде розміщено кількість доступних характеристик якості “Avaliable Quality Parameters()”. Обрати потрібну характеристику з випадного списку. Сторінка для оцінювання якості показана на рисунку 3.10.

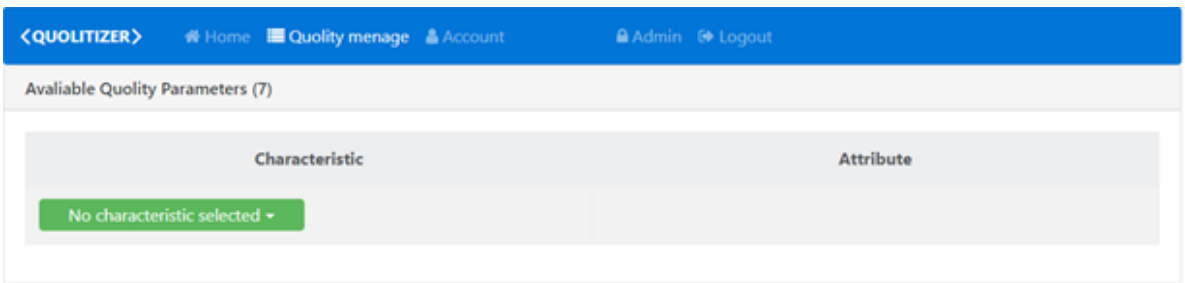


Рисунок 3.10 – Сторінка оцінювання якості

Обиравши одну з характеристик побудованої моделі якості, доступним буде перелік атрибутів, що містить дана характеристика. Вікно додатку з обраною характеристикою якості показано на рисунку 3.11.

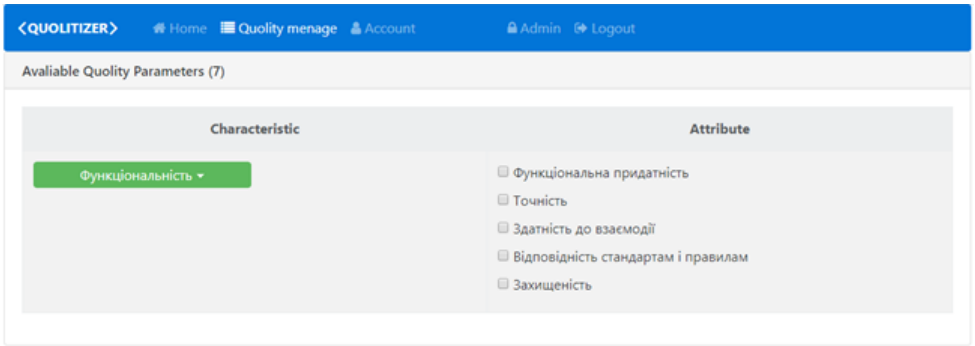


Рисунок 3.11 – Вікно додатку з обраною характеристикою якості

Обравши характеристику потрібно позначити атрибути цієї характеристики на основі яких буде проводитись оцінка. Якщо позначити хоча б один атрибут, то з’явиться кнопка переходу до пункту “Results”. Вікно з позначеними атрибутами якості зображено на рисунку 3.12.

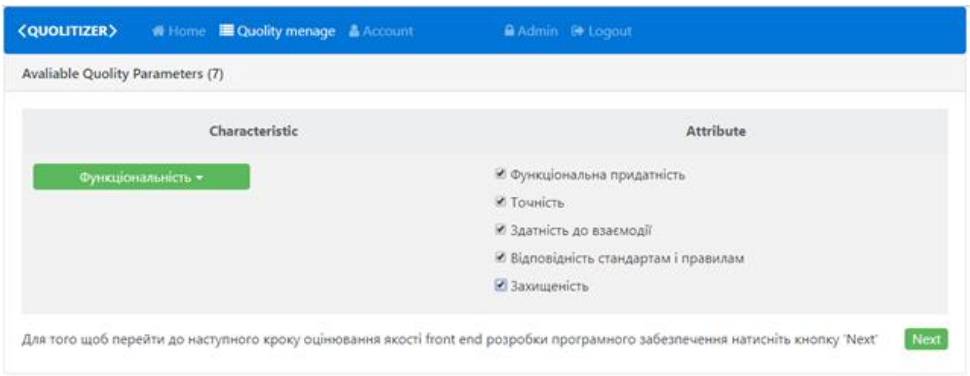


Рисунок 3.12 – Сторінка оцінювання якості з обраними атрибутами

Для того, щоб перейти до наступного пункту, необхідно натиснути кнопку “Next”. Якщо позначені атрибути не відповідають загальній кількості атрибутів обраної характеристики, то при натисненні на кнопку “Next” відкриється вікно

підтвердження. Вікно підтвердження показано на рисунку 3.13. В іншому випадку, буде здійснено перехід до панелі “Results”. Панель “Results” зображена на рисунку 3.14.

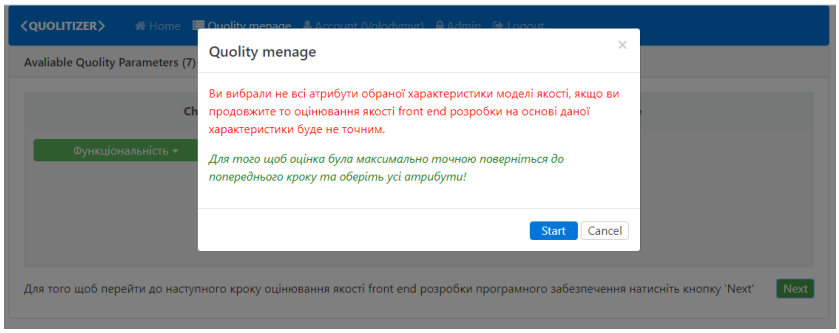


Рисунок 3.14 – Вікно підтвердження дії

Натиснувши на кнопку “Start”, у вікні підтвердження користувача (рисунок 3.15) буде перенаправлено на вкладку “Results”.

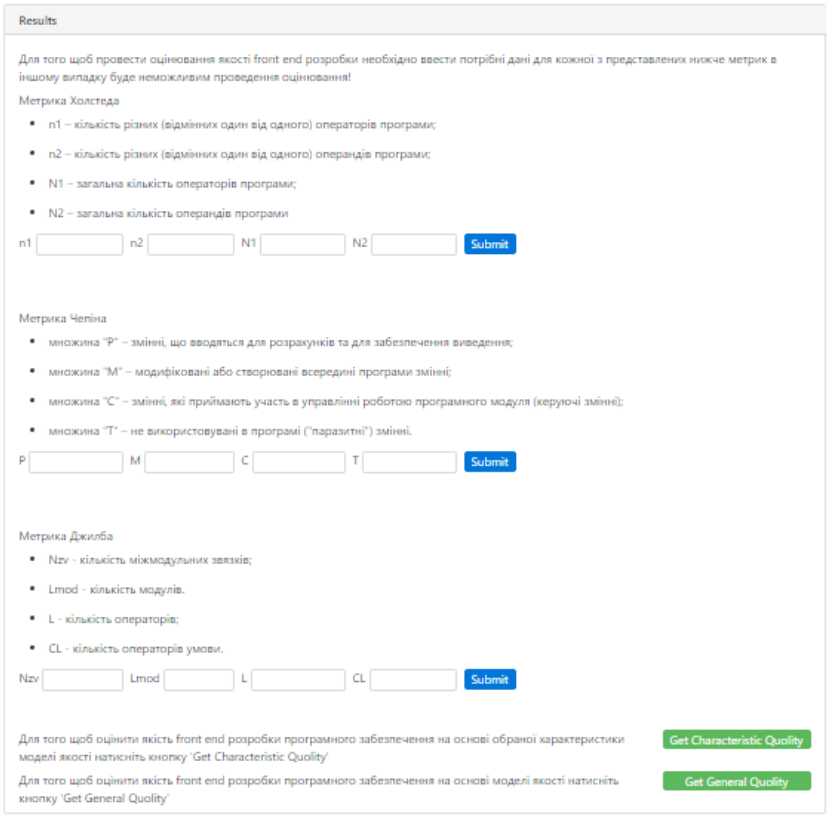


Рисунок 3.15 – Панель “Results”

Кількісна оцінка атрибутів якості обраної характеристики проводиться за допомогою метрик. Для того, щоб провести оцінку необхідно ввести параметри, які вимагає кожна метрика.

На панелі “Results” розміщуються поля для вводу необхідних параметрів метрик за допомогою яких визначаються кількісні характеристики обраних атрибутів. Для кожної з метрик потрібно ввести усі необхідні параметри і натиснути кнопку “Submit”. Після цього параметри будуть записані в систему. Перелік відображуваних метрик на панелі залежить від обраних атрибутів, якщо метрика визначає кількісні характеристики обраного атрибута, то відповідно вона є доступною на панелі “Results”. В іншому випадку – не доступна для користувача. Вікно зі встановленими параметрами для кожної з метрик зображено на рисунку 3.16.

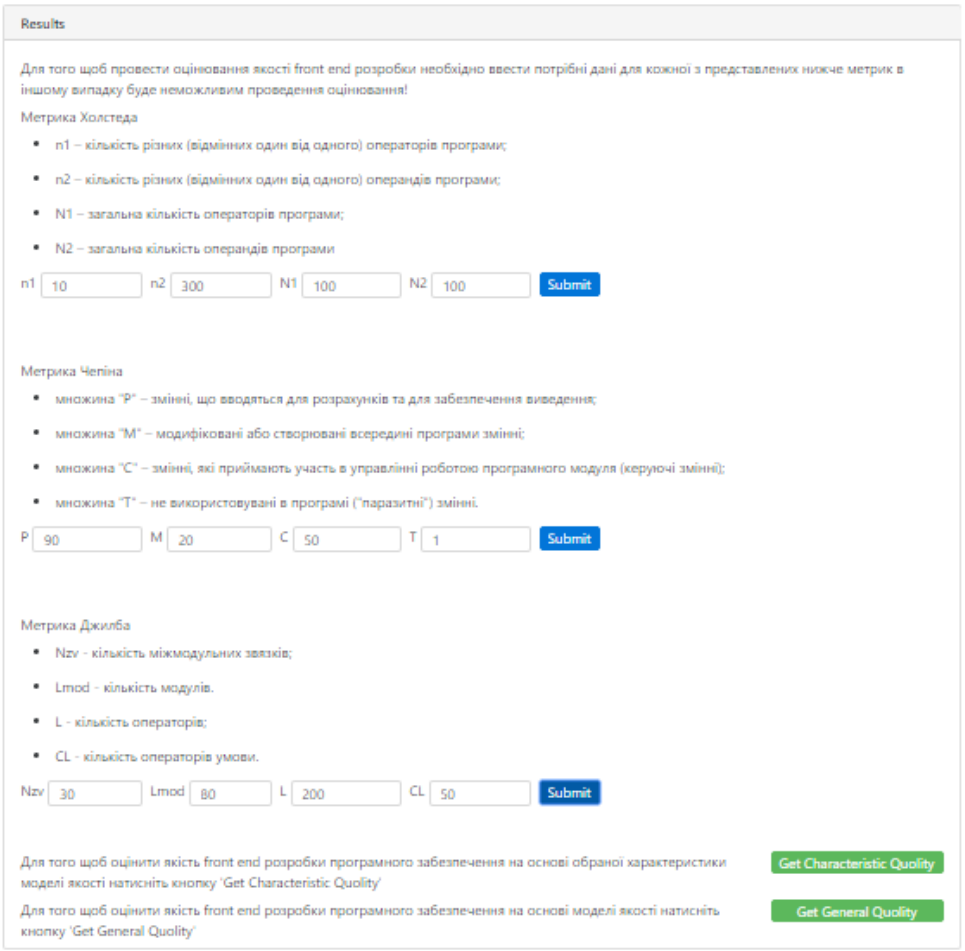


Рисунок 3.16 – Вікно зі встановленими параметрами для кожної з метрик

Також на обраній панелі внизу розташовані дві кнопки “Get Characteristic Quolity” та “Get General Quolity”. Натиснувши на кнопку “Get Characteristic Quolity” буде проведена оцінка обраної характеристики оцінюваної системи, результат даної оцінки буде записано в систему. Таких кроків з оцінкою характеристики потрібно провести рівно стільки ж, скільки доступно характеристик.

Натиснувши кнопку “Get General Quolity” буде проведена загальна оцінка системи на основі попередньо оцінених характеристик. Якщо кількість попередньо оцінених характеристик не співпадає з загальною кількістю характеристик, то відкриється вікно підтвердження. Вікно підтвердження проведення загальної оцінки системи показано на рисунку 3.17.

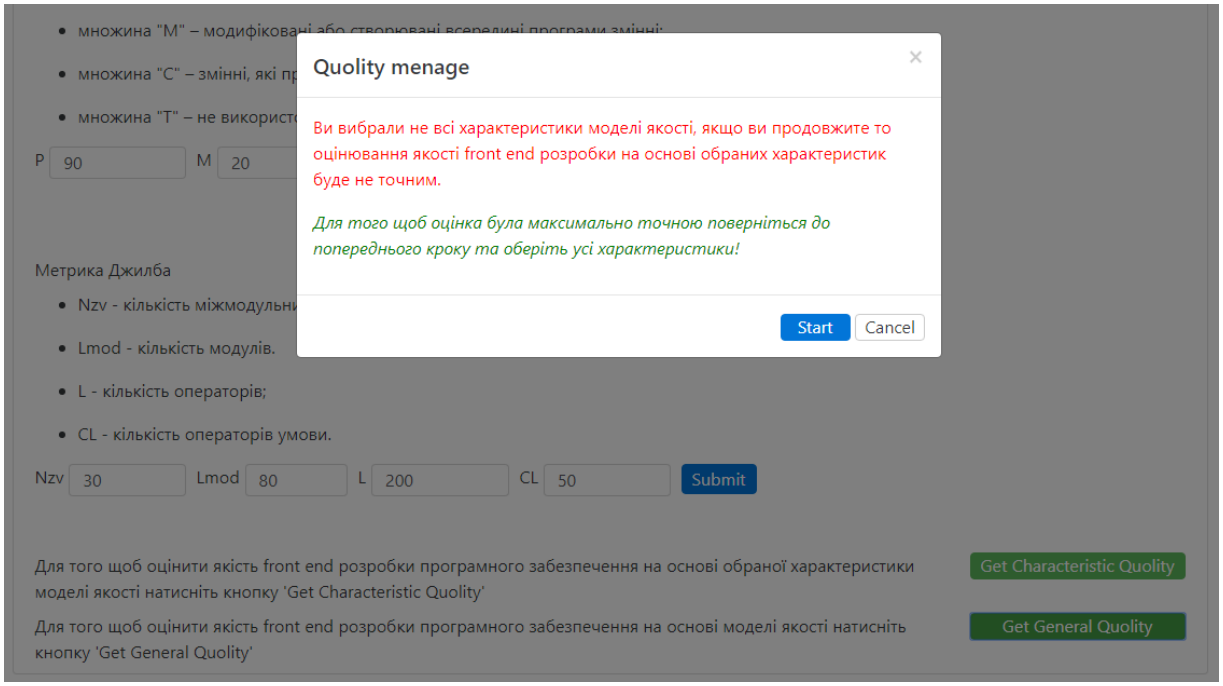


Рисунок 3.17 – Вікно підтвердження проведення загальної оцінки системи

Якщо натиснути на кнопку “Start” у вікні підтвердження, то буде проведена загальна оцінка системи. Після цього результат проведення оцінювання буде виведено у модальному вікні. Неповний результат оцінювання якості технології front end на основі однієї характеристики приведено на рисунку 3.18.

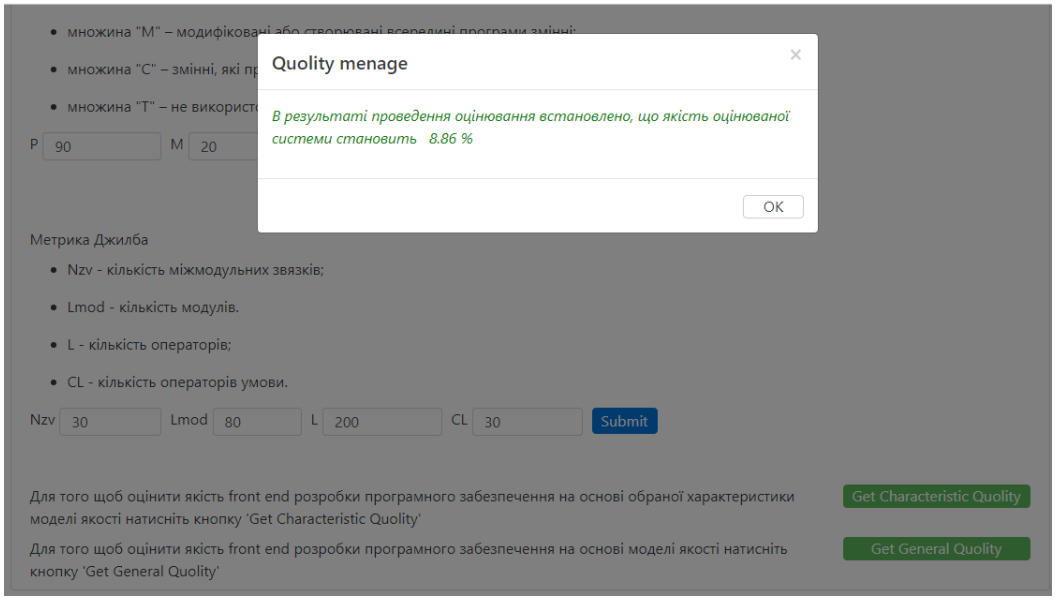


Рисунок 3.18 – Неповний результат оцінювання якості

Отже, для більш повної оцінки системи, потрібно покроково визначити кількісні характеристики кожної з доступних в системі характеристик, а вже тоді проводити повну оцінку системи. Вікно з результатом повної оцінки системи на основі всіх доступних характеристик наведено на рисунку 3.19.

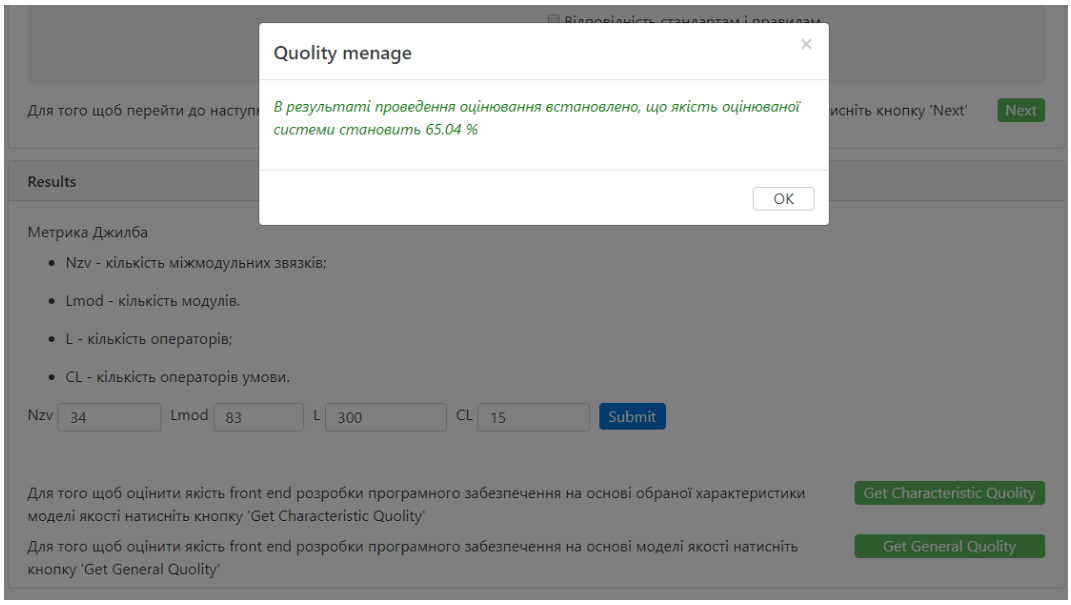


Рисунок 3.19 – Вікно повного результату оцінки системи

Отже, провівши покрокове тестування розробленого програмного засобу управління метриками якості FE технологій, можна зробити висновок про те, що його реалізація повністю відповідає поставленим вимогам. Розроблений додаток дозволяє користувачеві в повній мірі провести оцінку розробленої або розроблюваної системи та одержати результат якому відповідає якість оцінюваної системи.

У даному розділі дипломної роботи магістра одержано наступні практичні результати:

1. Визначено сукупність функціональних вимог до програмного засобу управління метриками якості FE технологій розробки ПЗ, що дозволило визначити шляхи побудови та тип його архітектури.

2. Побудовано архітектуру програмного засобу із поділом на рівень front end та back end опрацювання даних, а для його реалізації використано платформу NodeJS та фреймворк AngularJS, що дало змогу ефективно реалізувати об’єкти запропонованої моделі якості та провести вимірювання показників якості атрибутів і характеристик даної моделі.

3. Протестовано програмний засіб управління метриками якості FE технологій у результаті якого одержано кількісні показники якості на різних рівнях моделі якості і підтверджено працездатність програмного засобу.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Дипломна робота магістра присвячена розробці програмного засобу управління метриками якості технологій front end розробки. Оскільки, розроблене програмне забезпечення передбачає використання електронно-обчислювальної техніки, то важливим є дотримання вимог з охорони праці і техніки безпеки. Проаналізуємо основні правила і норми, яких необхідно дотримуватись при експлуатації комп'ютерів та периферійних пристроїв.

В загальному випадку, поняття охорони праці в галузі інформаційних технологій, представляє собою дотримання всіх вимог і нормативів, що присутні в законодавчих актах про охорону праці. Закони цієї області спрямовані на якісну і безпечну експлуатацію робочих приладів і приміщень, дотримання санітарно-гігієнічних умов праці і захист від інших небезпечних чинників на підприємстві [24]. Ці засоби є складовими дослідження методів і засобів управління метриками якості front end технологій розробки. В основних законодавчих актах про охорону праці приділяється велика увага поліпшенню умов праці в усіх галузях господарства, впровадженню сучасних засобів техніки безпеки і забезпечення санітарно - гігієнічних умов, що запобігають виробничому травматизму і професійним захворюванням.

Охорона життя і здоров'я людини є пріоритетним напрямком соціальної політики держави. В Україні прийнято закон прямої дії «Про охорону праці», який регламентує захист конституційного права працівників на безпечні умови праці. Законодавство України про охорону праці складається із загальних законів України та спеціальних законодавчих актів [24]. Загальними законами України, що визначають основні положення з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю (КЗпП), Закон України «Про загальнообов'язкове державне соціальне страхування від нещасного випадку

на виробництві та професійного захворювання, які спричинили втрату працездатності» [24].

При розробці програмного засобу управління метриками якості, які передбачали використання ПК, площа та об'єм для одного робочого місця оператора визначається згідно норм НПАОП 0.00-7.15-18, зокрема площа повинна становити не менше 6,0 квадратних метрів, об'єм - не менше 20,0 кубічних метрів [25].

Згідно вимог НПАОП 0.00-7.15-18 стіни, стеля та підлога приміщень, в яких розміщені комп’ютери, повинні бути виготовлені з матеріалів, дозволених для оформлення приміщень органами державного санітарно-епідеміологічного нагляду.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), повинні бути надійно захищені діелектричними щитками та сітками з метою недопущення потрапляння працівника під напругу.

Організація робочого місця оператора повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам НПАОП 0.00-7.15-18.

У приміщенні, де одночасно експлуатуються понад п'ять електронно-обчислювальних машин, на помітному та доступному місці мають бути встановленні аварійні резервні вимикачі, які можуть повністю вимкнути електричне живлення приміщення, крім освітлення [26].

Одним із найбільш важливих нормативних документів щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98 [25].

Дотримання даних правил значно знижує наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відео-дисплейними матеріалами, зокрема можливість зорових, нервово-емоційних переживань, серцево-судинних захворювань. Виходячи з цього, роботодавець

повинен забезпечити гігієнічні й ергономічні вимоги щодо організації робочих приміщень для експлуатації електронно-обчислювальних машин (ЕОМ) з ВДТ, робочого середовища, робочих місць з ЕОМ, режиму праці і відпочинку при роботі з ЕОМ тощо, які викладені у нормах НПАОП 0.00-7.15-18.

Відповідно до встановлених гігієнічно-санітарних вимог роботодавець зобов'язаний забезпечити в приміщеннях з оптимальні параметри виробничого середовища [25].

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів персонального комп’ютера і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Вимогам нормативних актів з охорони праці мають відповідати:

– умови праці на кожному робочому місці;

– безпека технологічних процесів, машин, механізмів, обладнання й інших засобів виробництва;

– стан засобів колективного та індивідуального захисту;

– санітарно-побутові умови.

Приміщення з ЕОМ мають бути оснащені переносними вуглекислотними вогнегасниками з розрахунку 1 на 50 м^2, але не менше 2 на приміщення. Підходи до засобів пожежогасіння повинні бути вільними [28].

При облаштуванні робочих місць необхідно забезпечувати належні умови освітлення приміщення і робочого місця, оптимальні параметри мікроклімату, ергономічних характеристик основних елементів робочого місця, а також враховувати наявність шуму і вібрації, м'якого рентгенівського випромінювання, електромагнітного випромінювання, ультрафіолетового та інфрачервоного випромінювання, електростатичного поля між екраном і оператором, відсутність пилюки, озону, оксидів азоту та аероіонізації.

Отже, при розробці програмного засобу управління метриками якості технологій front end розробки, проаналізовано та враховано необхідні вимоги щодо охорони праці при використанні електронно-обчислювальної техніки і забезпечено умови для зручної та ефективної роботи працівників.

4.2. Підвищення стійкості роботи об'єктів господарської діяльності у воєнний час

На основі вивчення факторів, які впливають на стійкість роботи об'єктів господарської діяльності, і оцінки стійкості елементів і галузей виробництва проти уражаючих факторів ядерної, хімічної і біологічної зброї, стихійних лих і виробничих аварій, необхідно завчасно організувати і провести організаційні, інженерно-технічні й технологічні заходи для підвищення стійкості роботи.

Здійснення організаційних заходів передбачає завчасну підготовку всіх структур цивільного захисту, служб і формувань до надзвичайних ситуацій, в тому числі і військових дій [22].

Вжиттям технологічних заходів підвищується стійкість роботи об'єктів шляхом змінювання технологічних процесів, режимів, можливих в умовах різних надзвичайних ситуацій.

Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу уражаючих факторів під час військових дій [22].

При проведенні цих заходів необхідно враховувати конкретні умови об'єкта народного господарства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

Одним з найбільш важливих завдань в умовах воєнного часу і надзвичайних ситуацій є забезпечення захисту людей та їх життєдіяльності.

Для підвищення стійкості об’єктів господарювання та захисту людей, згідно [22] необхідно:

– створити на об'єкті надійну систему оповіщення про загрози нападу противника, радіоактивне забруднення, хімічне і біологічне зараження, загрозу стихійного лиха і виробничої аварії.

– організувати розвідку і спостереження за радіоактивним забрудненням, хімічним і біологічним зараженням;

– організувати гідрометеорологічне спостереження за рівнем води, напрямком і швидкістю вітру, рухом і поширенням хмари радіоактивного забруднення, сильнодіючих отруйних речовин і отруйних речовин.

– створити фонд захисних споруд ЦО, запасів засобів індивідуального захисту і забезпечення своєчасної видачі їх населенню.

– завчасно підготуватись до масової санітарної обробки населення і знезаражування одягу;

– організувати взаємодію з установами охорони здоров'я для медичного обслуговування населення в умовах воєнного часу.

Також в умовах воєнного часу необхідно провести підготовку до евакуації населення, розміщеного в зонах можливих руйнувань і катастрофічного затоплення. Це передбачає завчасну підготовку місць евакуації, організацію прийому евакуйованого населення на територію населених пунктів.

Окрім цього, необхідно забезпечити постачання продуктів харчування, питної води, предметів першої необхідності та провести заходи щодо морально-психологічної підготовки населення до виживання в умовах воєнного часу, забезпечити процес чіткого інформування про обстановку та правила дій і поведінки населення в надзвичайних ситуаціях воєнного часу [22].

Для забезпечення стійкості роботи об'єктів повинні проводитись інженерно-технічні заходи на мережах комунального господарства з метою захисту джерел тепла із заглибленням у ґрунт комунікацій. Котельні слід розміщувати в спеціальному окремо розміщеному приміщенні.

Якщо об'єкт одержує тепло з міської теплоцентралі, необхідно провести заходи для забезпечення стійкості трубопроводів і розподільних пристроїв, підведених до об'єкта [22].

Теплова мережа має будуватися за кільцевою системою з прокладанням труб у спеціальних каналах зі з'єднанням паралельних ділянок. Для відключення пошкоджених ділянок мають бути встановлені запірно-регулючі засувки, вентилі та ін. Ці пристосування необхідно розміщувати в оглядових колодязях, на території, що не завалюється при руйнуванні будівель.

Система каналізації має будуватись окремо: одна для дощових, друга для промислових і господарських вод. На об'єкті має бути не менше двох виводів з підключенням до міських каналізаційних колекторів, а також виводи і колодязі з аварійними засувками на об'єктових колекторах з інтервалом 50 м на території, що не завалюється, для аварійного скидання неочищеної води в найближчі штучні та природні заглиблення [22].

На деяких промислових об'єктах є системи для забезпечення технології виробництва: для подання кисню, аміаку, стиснутого повітря та інших рідких і газових реактивів. Для цих систем розробляють заходи для попередження виникнення вторинних факторів зброї, стихійних лих та виробничих аварій і катастроф.

Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні.

З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги. Повітряні лінії електропостачання замінити на підземно-кабельні.

Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ [22].

Для підвищення стійкості забезпечення водою слід провести такі заходи. Необхідно створити основні і резервні джерела водопостачання. Як резервне джерело краще мати артезіанську свердловину, яку необхідно підключити до системи водопостачання. Крім того, воду можна брати з близько розміщеної природної водойми або спорудити штучну водойму чи резервуари з обладнанням пристроїв для збору і перекачування води.

Підвищенню стійкості забезпечення водою сприяє подавання води безпосередньо в мережу поза водонапірними баштами, спорудження обвідних ліній для подання води поза пошкодженими спорудами.

Завчасне вжиття заходів захисту вододжерел, водопровідних споруд, свердловин і шахтних колодязів від забруднення радіоактивними речовинами, зараження хімічними і біологічними засобами [22].

Для забезпечення виробництва продукції необхідні електроенергія, паливо, мастила, засоби захисту рослин, міндобрива, профілактичні й лікувальні препарати ветеринарної медицини, запасні частини, сировина та інші матеріально-технічні засоби. Забезпечення об'єктів цими ресурсами дасть можливість випускати необхідну продукцію в надзвичайних умовах мирного і воєнного часу. Тому повинні проводитись такі заходи, які б забезпечили стійкість постачання і сприяли підвищенню захисту мережі електро-, водо-, газопостачання, транспортних комунікацій і джерел постачання всім необхідним для забезпечення функціонування галузей сільського господарства в надзвичайних умовах.

Для забезпечення надійності системи управління і зв'язку потрібно організувати захищений пункт управління, забезпечити його засобами зв'язку, які б дали можливість швидко доводити сигнали ЦЗ до всіх виробничих підрозділів і населення у місцях проживання. При цьому необхідно здійснити планування збору даних про обстановку, передачу команд і розпоряджень в умовах впливу на об'єкт уражаючих факторів. Для підвищення стійкості системи управління і зв’язку в умовах воєнного часу необхідно організувати використання радіозасобів, засобів телефонного зв'язку, а також забезпечити зв’язок із колонами евакуйованого населення, що перебувають у дорозі, і відповідальними особами, які супроводжують їх під час евакуації, забезпечити дублювання ліній і каналів зв'язку.

ВИСНОВКИ

У дипломній роботі магістра одержано наступні наукові і практичні результати.

1. Проаналізовано сферу застосування і термінологічні особливості front end технологій у різних галузях інформаційних технологій, що дало змогу виявити їхні спільні риси і чітко встановити, що технології front end розробки програмного забезпечення під web-простір забезпечують інтерактивність користувацького інтерфейсу, а метрики оцінювання якості потребують проведення додаткових досліджень.

2. Проведено аналіз найбільш популярних технологій front end розробки у результаті якого встановлено, що для вибору оптимальних технологій необхідно розробити універсальну та загальноприйняту систему метрик, оскільки існуючі критерії є індивідуальними для кожної з технологій.

3. Проведено аналіз характеристик та метрик якими можна скористатись при оптимальному виборі технологій front end розробки програмного забезпечення, у результаті якого встановлено, що деякі метрики якості потрібно формалізувати, доповнити для забезпечення більш повної картини опису таких технологій.

4. Для представлення якості технологій front end розробки ПЗ для web-простору обгрунтовано застосування моделі якості, яка є стандартизованою згідно ISO/IEC 25010, володіє ієрархічною структурою і доповнено її характеристикою «Конфігурованість», що дало змогу більш ширше та адекватніше описати властивості FE технологій, порівнюючи з іншими моделями.

5. Визначено і класифіковано за характеристиками елементарні властивості якості FE технологій, що дало змогу обчислювати їх кількісне значення на основі як стандартних метрик якості, так і метрик коду програмного забезпечення.

6. Обгрунтовано застосування метрик Чепіна, Холстеда, Джилба та МакКейба для оцінювання атрибутів якості технологій FE розробки, що дало змогу

спростити сприйняття та підвищити зрозумілість кількісних значень якості відповідних технологій.

7. Запропоновано процедуру та реалізовано алгоритм оцінювання якості FE технологій на різних рівнях ієрархічного дерева моделі якості, що дало змогу керувати як метриками оцінювання, так і процесом прийняття рішень щодо якості елементарних критеріїв, характеристик та інтегральної якості.

8. Визначено сукупність функціональних вимог до програмного засобу управління метриками якості FE технологій розробки ПЗ, що дозволило визначити шляхи побудови та тип його архітектури.

9. Побудовано архітектуру програмного засобу із поділом на рівень front end та back end опрацювання даних, а для його реалізації використано платформу NodeJS та фреймворк AngularJS, що дало змогу ефективно реалізувати об’єкти запропонованої моделі якості та провести вимірювання показників якості атрибутів і характеристик даної моделі.

10. Протестовано програмний засіб управління метриками якості FE технологій у результаті якого одержано кількісні показники якості на різних рівнях моделі якості і підтверджено працездатність програмного засобу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию. М.: «Вильямс». 2012. 320с.

2. ДеМарко Т., Листер Т. Вальсируя с медведями: управление рисками в проектах по разработке программного обеспечения. Компания p.m. Office. М. 2015. 190 с.

3. Основные положения по разработке и применению систем сетевого планирования и управления. М. 1974. 216 с.

4. Алферов П. Роль бизнес-заказчика в ИТ-проекте. Управление проектами. №4. 2008. URL: http://www.pmmagazine.ru/document.asp?ob_no=771 (дата звернення 15.11.2019 р.).

5. Фатрелл Р., Шафер Д., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. М.: Издательский дом «Вильямс». 2013. 1136 с.

6. Брауде Э. Технология разработки программного обеспечения. СПб: Питер, 2014. 655 с.

7. Зыль С. Проектирование, разработка и анализ програмного обеспечения критического назначения. СПб: БХВ-Петербург. 2010. 336 с.

8. Соммервилл И. Инженерия программного обеспечения. Издательский дом “Вильямс”. 2015. 624 с.

9. ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes

10. ISO/IEC 33001:2015. Information technology – Process assessment

11. Кантор М. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения. М.: Вильямс. 2012. 176 с.

12. Симонов С. Технологии и инструментарий для управления рисками. Jet Info. № 2. 2003. 34 с.

13. Fenton N., Neil M. Decision Support Software for Probabilistic Risk Assessment Using Bayesian Networks. IEEE Software. 2014. № 31 (2). P. 21-26.

14. Fenton N., Neil M. Risk Assessment and Decision Analysis with Bayesian Networks. CRC Press. 2012. – P. 33-38.

15. Soumya Krishnan M. Software Development Risk Aspects and Success Frequency on Spiral and Agile Model. Int. Journal of Innovative Research in Computer and Comm. Eng. 2015. Vol. 3. P. 122-129.

16. Abdullahi M., Basri Sh., Osman Ali H. Strength and Weakness of Software Risk Assessment Tools. International Journal of Software Engineering and Its Applications. 2014. Vol. 8. № 3. P. 389-398.

17. Woody C., Ellison R. Supply-Chain Risk Management: Incorporating Security into Software Development. Software Engineering Institute Carnegie Mellon University. 2010. P. 166-178.

18. Britkin A.I. Model estimate the duration of the iterative software development process. Open education. 2009. №4. P. 75.

19. Seacord R. Secure Coding in C and C++ 2nd Edition by Addison-Wesley Professional. Part of the SEI Series in Software Engineering series Published. 2013. 256 p.

20. William R. Project Management Body of Knowledge. PMI Standard Committee. 2006. 182 p.

21. Sommerville I. Software Engineering. Addison-Wesley Publ. Company. 2011. 866 p.

22. Желібо Є., Заверуха Н., Зацарний В. Безпека життєдіяльності. К.: 2001. 483 с.

23. НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації ЕОМ». Наказ Держгірпромнагляду від 26.03.2010 No 6.

24. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Київ. 2018. URL: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php (дата звернення: 10.12.2019).

25. Бедрій Я. Основи охорони праці користувачів персональних комп’ютерів: навчальний посібник для студентів ВНЗ та інженерів-практиків. Навчальна книга-Богдан. 2014. 144 с.

ДОДАТОК А

Апробація результатів роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів.
Актуальні задачі сучасних технологій – Тернопіль 25-26 листопада 2020.

УДК 004.055
І.В. Бойко, канд. техн. наук, В.В. Куніц
Тернопільський національний технічний університет імені Івана Пулюя, Україна

АНАЛІЗ ОСОБЛИВОСТЕЙ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

I.V. Boiko PhD, V.V. Kunits
ANALYSIS OF FRONT END DEVELOPMENT TECHNOLOGIES

Розвиток інженерії програмного забезпечення сприяє розвитку нових технологій розробки програмних продуктів і формує розгалуження щодо середовищ їх функціонування. На даний час спостерігається тенденція до зростання ролі напряму web-engineering, що передбачає необхідність побудови правил доступу до даних на рівні їх представлення кінцевому користувачу та відповідно на фізичному рівні доступу до бази даних. При розробці програмного забезпечення використовують термінологію front end та back end частин. Front end представляє собою своєрідний інтерфейс взаємодії між кінцевим користувачем системи і власне логікою програми, що виконується на сервері (back end). З іншої сторони, Front end можна розглядати як деяку абстракцію, що забезпечує користувачу дружній інтерфейс при роботі з системою. Для побудови сучасних web-інтерфейсів можна скористатись високопродуктивною бібліотекою React.js, ES6, Node.js – універсальна платформа, що дає змогу реалізовувати як front end частину, так і back end засобами javascript та ін. AngularJS є популярним фреймворком з відкритим вихідним кодоьм, який широко використовується для створення та підтримки складних веб-додатків, позиціонується як розширення HTML. Спочатку AngularJS був орієнтований на створення односторінкових веб додатків, що передбачало завантаження додаткового контенту в залежності від потреб та поведінки користувача. Перевагою такого підходу є економія трафіку та зниження навантаження на сервер. AngularJS дає змогу скоротити час розробки програмного забезпечення за рахунок вбудованих механізмів, для прикладу шаблонізаторів (template engine), забезпечення таких характеристик як вбудованість, зв'язність, наявність готових рішень та простота тестування.

Інша бібліотека, що дає змогу реалізувати front end частину ПЗ – KnockoutJS. Дана бібліотека володіє засобами створення складних інтерфейсів та забезпечує «чистоту коду», здатність до розширюваності та читабельність коду. Головною перевагою і завданням KnockoutJS є автоматичне оновлення користувацького інтерфейсу у випадку зміни властивості, що реалізована у моделі.

Ще одним open-source front end фреймворком, який набув широкої популярності є Bootstrap. Даний фреймворк володіє ефективними засобами розробки логіки та забезпечення адекватної поведінки користувача на основі HTML, CSS і JS. Bootstrap підтримує стандарт адаптивного веб-дизайну і не залежить від розмірів і складності користувацького інтерфейсу. До переваг фреймворку Bootstrap належить: підтримка адаптивного веб-дизайну (можна відключити за бажанням); повнота документації, що інструментів проектування. До недоліків Bootstrap можна віднести: у випадку використання рідко застосовуваних стилів зростає розмір стандартного фреймворку; надмірна кількість HTML класів і DOM елементів «забруднюють» код і вводять в оману розробників.

Отже, у результаті аналізу фреймворків front end розробки програмного забезпечення, важко кількісно оцінити їхні переваги та недоліки, тому актуальною задачею є розробка методів і метрик при їх виборі для реалізації конкретних задач.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004.055
І.В. Бойко, канд. техн. наук, В.В. Куніц
(Тернопільський національний технічний університет імені Івана Пулюя)

АЛГОРИТМ РОБОТИ ПРОГРАМНОГО ЗАСОБУ УПРАВЛІННЯ МЕТРИКАМИ ПРИ ОЦІНЮВАННІ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

UDC 004.055
I.V. Boiko PhD, V.V. Kunits

THE ALGORITHM OF SOFTWARE OF METRICS MANAGEMENT IN EVALUATING FRONT END DEVELOPMENT

Програмний засіб управління метриками якості призначений для полегшення, оптимізації, та швидкого оцінювання технологій розробки інтерактивних інтерфейсів користувача шляхом вибору відповідних метрик.

Реалізація програмного засобу передбачає розробку його алгоритму, який дає змогу обирати для кожного атрибуту метрику, що визначена для вимірювання тієї чи іншої характеристики моделі якості. На рис. 1 наведено алгоритм роботи розробленого програмного засобу.

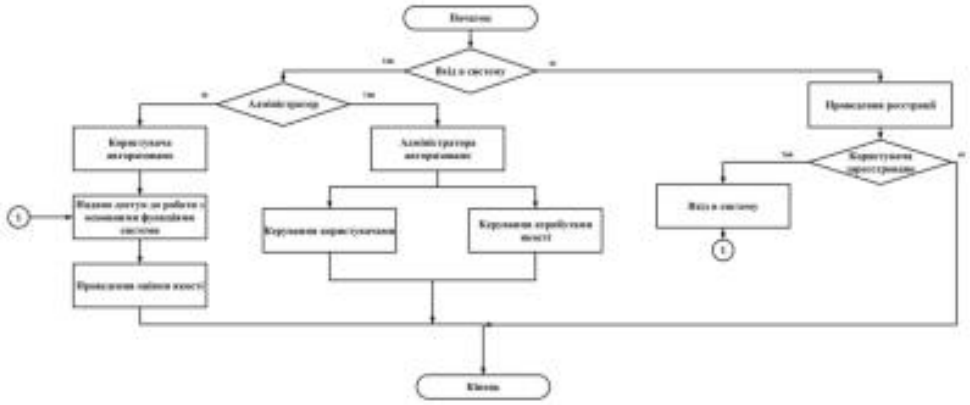


Рисунок 1. Алгоритм роботи програмного забезпечення управління метриками якості при оцінюванні якості технологій front end розробки

Розробивши алгоритм системи слід виконати наступні кроки:

- встановити середовище розробки програмного забезпечення WebStorm;
- встановити і налаштувати СКБД MongoDB;
- створити новий проект під назвою QMetrics;
- створити файл з набором необхідних пакетів для розробки ПЗ package.json;
- за допомогою команди npm install встановити усі необхідні файли;
- створити директорію для клієнтської частини client та добавити в корінь цієї директорії інші необхідні директорії, які необхідні на клієнтській частині;
- створити директорію для серверної частини server.

У результаті імплементації запропонованого алгоритму з використанням наведених вище технологій, реалізовано програмний засіб, який забезпечує гнучкість вибору технологій front end розробки при проектуванні різних програмних продуктів.

ДОДАТОК Б

Фрагменти програмного коду програмного засобу управління метриками якості технологій front end розробки

Б.1 – Вміст файлу package.json

```
{
  "name": "Quolitizer",
  "version": "1.0.0",
  "license": "MIT",
  "author": "Naida Volodymyr",
  "description": "Quolitizer project built using Angular 2+,
Express, Mongoose and Node.",
  "angular-cli": {},
  "engines": {
    "node": "6.11.4",
    "npm": "3.10.10"
  },
  "scripts": {
    "ng": "ng",
    "build": "ng build",
    "start": "node dist/server/app.js",
    "predev": "tsc -p server",
    "dev": "concurrently \"mongod\" \"ng serve -pc proxy.conf.json -
-open\" \"tsc -w -p server\" \"nodemon dist/server/app.js\"",
    "prod": "concurrently \"mongod\" \"ng build -aot -prod && tsc -p
server && node dist/server/app.js\"",
    "test": "ng test",
    "testbe": "tsc -p server && mocha dist/server/test --exit",
    "lint": "ng lint",
    "lintbe": "tslint server/**/**.ts{,x}",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.0.3",
```

```
    "@angular/common": "^5.0.3",
    "@angular/compiler": "^5.0.3",
    "@angular/core": "^5.0.3",
    "@angular/forms": "^5.0.3",
    "@angular/http": "^5.0.3",
    "@angular/platform-browser": "^5.0.3",
    "@angular/platform-browser-dynamic": "^5.0.3",
    "@angular/router": "^5.0.3",
    "angular2-jwt": "^0.2.3",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.18.2",
    "bootstrap": "4.0.0-alpha.5",
    "core-js": "^2.5.1",
    "dotenv": "^4.0.0",
    "express": "^4.16.2",
    "font-awesome": "^4.7.0",
    "jquery": "^3.2.1",
    "jsonwebtoken": "^8.1.0",
    "mongoose": "^4.13.5",
    "morgan": "^1.9.0",
    "rxjs": "^5.5.2",
    "tether": "1.4.0",
    "zone.js": "^0.8.18"
  },
  "devDependencies": {
    "@angular/cli": "1.5.4",
    "@angular/compiler-cli": "^5.0.3",
    "@angular/language-service": "^5.0.3",
    "@types/express": "^4.0.37",
    "@types/jasmine": "~2.5.54",
    "@types/jasminewd2": "~2.0.3",
    "@types/node": "~6.0.92",
    "chai": "^4.1.2",
    "chai-http": "^3.0.0",
    "codelyzer": "~3.2.2",
    "concurrently": "^3.5.1",
```

```
    "jasmine-core": "~2.6.2",
    "jasmine-spec-reporter": "~4.1.0",
    "karma": "~1.7.1",
    "karma-chrome-launcher": "~2.1.1",
    "karma-cli": "~1.0.1",
    "karma-coverage-istanbul-reporter": "^1.2.1",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "^0.2.2",
    "mocha": "^4.0.1",
    "nodemon": "^1.12.1",
    "protractor": "~5.1.2",
    "ts-node": "~3.2.0",
    "tslint": "~5.7.0",
    "typescript": "~2.4.2"
  }
}
```

Б.2 – Код сервера

```
import * as bodyParser from 'body-parser';
import * as dotenv from 'dotenv';
import * as express from 'express';
import * as morgan from 'morgan';
import * as mongoose from 'mongoose';
import * as path from 'path';

import setRoutes from './routes';

const app = express();
dotenv.load({ path: '.env' });
app.set('port', (process.env.PORT || 3000));

app.use('/', express.static(path.join(__dirname, '../public')));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

let mongodbURI;
```

```
if (process.env.NODE_ENV === 'test') {
  mongodbURI = process.env.MONGODB_TEST_URI;
} else {
  mongodbURI = process.env.MONGODB_URI;
  app.use(morgan('dev'));
}

mongoose.Promise = global.Promise;
const mongodb = mongoose.connect(mongodbURI, { useMongoClient: true
});

mongodb
  .then((db) => {
    console.log('Connected to MongoDB on', db.host + ':' + db.port);

    setRoutes(app);

    app.get('/*', function(req, res) {
      res.sendFile(path.join(__dirname, '../public/index.html'));
    });

    if (!module.parent) {
      app.listen(app.get('port'), () => {
        console.log('Angular Full Stack listening on port ' +
app.get('port'));
      });
    }

  })
  .catch((err) => {
    console.error(err);
});

export { app };
```

Б.3 – Сервіс для керування користувачами

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';

import { User } from '../shared/models/user.model';

@Injectable()
export class UserService {

  constructor(private http: HttpClient) { }

  register(user: User): Observable<User> {
    return this.http.post<User>('/api/user', user);
  }

  login(credentials): Observable<any> {
    return this.http.post<any>('/api/login', credentials);
  }

  getUsers(): Observable<User[]> {
    return this.http.get<User[]>('/api/users');
  }

  countUsers(): Observable<number> {
    return this.http.get<number>('/api/users/count');
  }

  addUser(user: User): Observable<User> {
    return this.http.post<User>('/api/user', user);
  }

  getUser(user: User): Observable<User> {
    return this.http.get<User>(`/api/user/${user._id}`);
  }
```

```
  editUser(user: User): Observable<string> {
    return this.http.put(`/api/user/${user._id}`, user, {
responseType: 'text' });
  }

  deleteUser(user: User): Observable<string> {
    return this.http.delete(`/api/user/${user._id}`, { responseType:
'text' });
  }

}
```

Б.4 – Код компоненту admin.component

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormControl, FormGroup, Validators } from
'@angular/forms';

import { ToastComponent } from '../shared/toast/toast.component';
import { AuthService } from '../services/auth.service';
import { UserService } from '../services/user.service';
import { User } from '../shared/models/user.model';
import { QuolityService } from '../services/quolity.service';
import { QuolityParameter } from '../shared/models/quolity-
parameter.model';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.scss']
})
export class AdminComponent implements OnInit {
  quolityParameter = new QuolityParameter();
  quolityParameters: QuolityParameter[] = [];

  users: User[] = [];
```

```
  isLoading = true;
  isAddParameters = false;
  isEditing = false;

  addParameterForm: FormGroup;
  name = new FormControl('', Validators.required);
  value = new FormControl('', Validators.required);
  weight = new FormControl('');

  constructor(public auth: AuthService,
              public toast: ToastComponent,
              private userService: UserService,
              private quolityService: QuolityService,
              private formBuilder: FormBuilder,) { }

  ngOnInit() {
    this.getUsers();
    this.getParameters();
    this.addParameterForm = this.formBuilder.group({
      name: this.name,
      value: this.value,
      weight: this.weight
    });
  }

  getUsers() {
    this.userService.getUsers().subscribe(
      data => this.users = data,
      error => console.log(error),
      () => this.isLoading = false
    );
  }

  deleteUser(user: User) {
    if (window.confirm('Are you sure you want to delete ' +
user.username + '?')) {
```

```
      this.userService.deleteUser(user).subscribe(
        data => this.toast.setMessage('user deleted successfully.',
'success'),
        error => console.log(error),
        () => this.getUsers()
      );
    }
  }

  toggleAddParameters() {
    this.isAddParameters = !this.isAddParameters;
  }


  getParameters() {
    this.quolityService.getParameters().subscribe(
      data => this.quolityParameters = data,
      error => console.log(error),
      () => this.isLoading = false
    );
  }

  addParameter() {

this.quolityService.addParameter(this.addParameterForm.value).subscr
ibe(
      res => {
        this.quolityParameters.push(res);
        this.addParameterForm.reset();
        this.toast.setMessage('item added successfully.',
'success');
      },
      error => console.log(error)
    );
  }
```

```
  enableEditing(qParam: QuolityParameter) {
    this.isEditing = true;
    this.quolityParameter = qParam;
  }

  cancelEditing() {
    this.isEditing = false;
    this.quolityParameter = new QuolityParameter();
    this.toast.setMessage('item editing cancelled.', 'warning');
    // reload the quolity to reset the editing
    this.getParameters();
  }

  editParameter(qParam: QuolityParameter) {
    this.quolityService.editParameter(qParam).subscribe(
      () => {
        this.isEditing = false;
        this.quolityParameter = qParam;
        this.toast.setMessage('item edited successfully.',
'success');
      },
      error => console.log(error)
    );
  }

  deleteParameter(qParam: QuolityParameter) {
    if (window.confirm('Are you sure you want to permanently delete
this item?')) {
      this.quolityService.deleteParameter(qParam).subscribe(
        () => {
          const pos = this.quolityParameters.map(elem =>
elem._id).indexOf(qParam._id);
          this.quolityParameters.splice(pos, 1);
          this.toast.setMessage('item deleted successfully.',
'success');
        },
```

```
        error => console.log(error)
      );
    }
  }
}
```

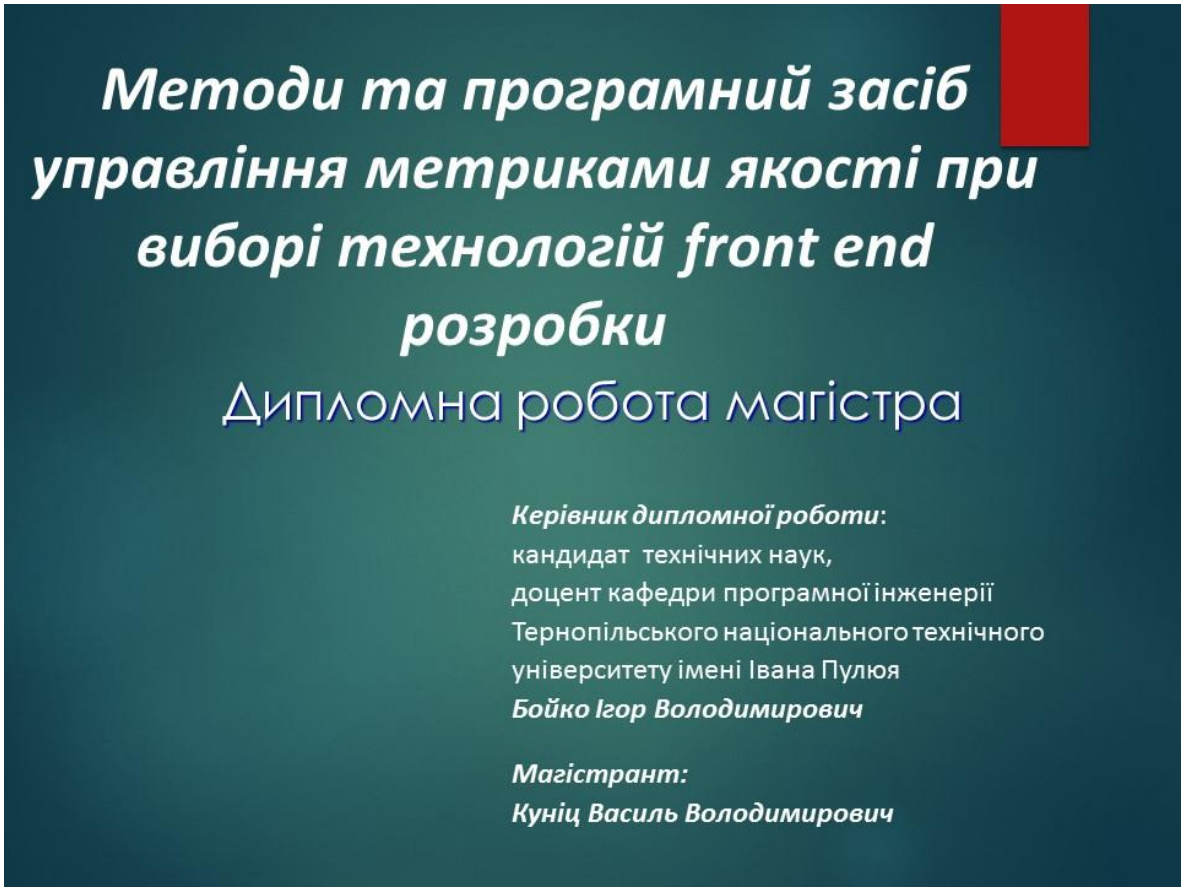
Б.5 – Код конфігураційного файлу app.module.ts

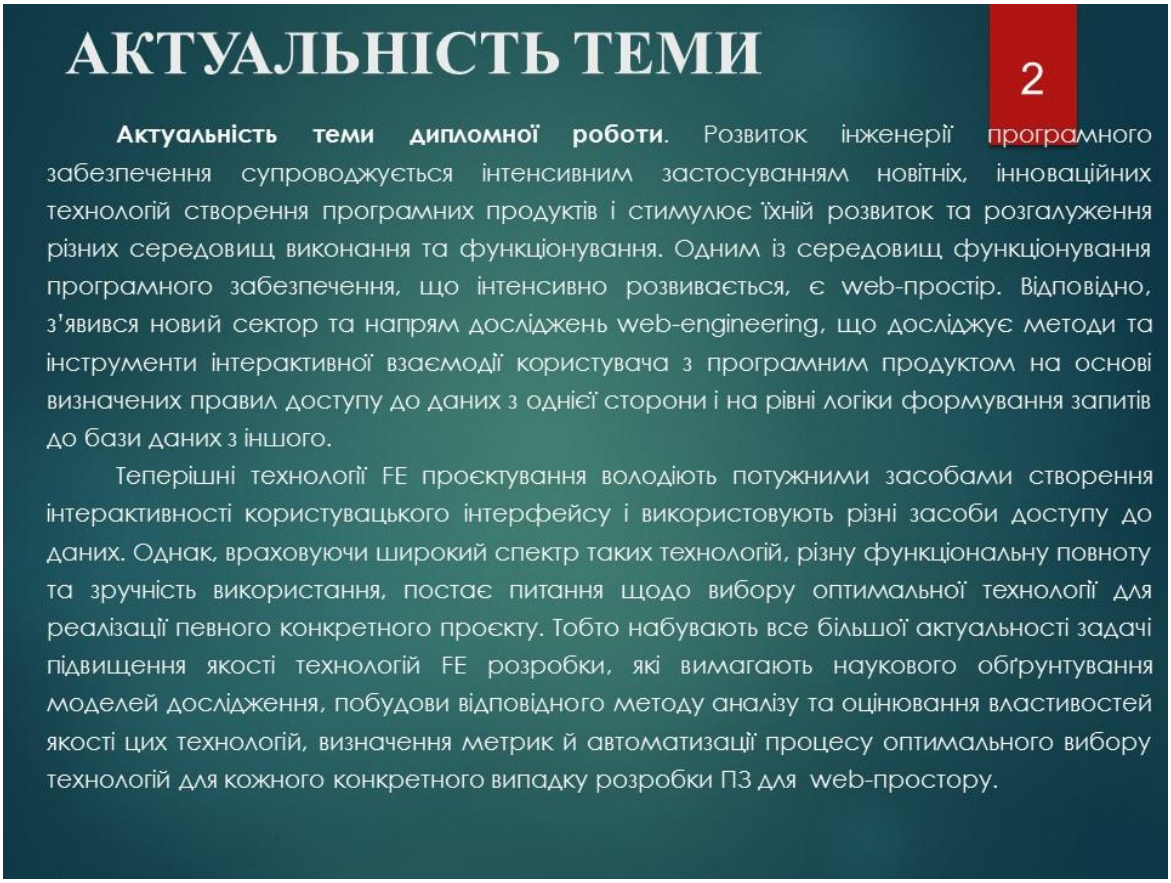
```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { RoutingModule } from './routing.module';
import { SharedModule } from './shared/shared.module';
import { QuolityService } from './services/quolity.service';
import { UserService } from './services/user.service';
import { AuthService } from './services/auth.service';
import { AuthGuardLogin } from './services/auth-guard-
login.service';
import { AuthGuardAdmin } from './services/auth-guard-
admin.service';
import { AppComponent } from './app.component';
import { QuolityComponent } from './quolity/quolity.component';
import { AboutComponent } from './about/about.component';
import { RegisterComponent } from './register/register.component';
import { LoginComponent } from './login/login.component';
import { LogoutComponent } from './logout/logout.component';
import { AccountComponent } from './account/account.component';
import { AdminComponent } from './admin/admin.component';
import { NotFoundComponent } from './not-found/not-found.component';

@NgModule({
  declarations: [
    AppComponent,
    QuolityComponent,
    AboutComponent,
    RegisterComponent,
    LoginComponent,
    LogoutComponent,
    AccountComponent,
```

```
    AdminComponent,
    NotFoundComponent
  ],
  imports: [
    RoutingModule,
    SharedModule
  ],
  providers: [
    AuthService,
    AuthGuardLogin,
    AuthGuardAdmin,
    QuolityService,
    UserService
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

ДОДАТОК В

Слайди презентації





МЕТА, ОБ'ЄКТ І ПРЕДМЕТ ДОСЛІДЖЕННЯ

3

Мета дипломної роботи магістра *є дослідження методів та інструментальних засобів управління метриками якості технологій побудови інтерфейсів інтерактивної взаємодії користувача з програмним забезпеченням*

Об'єкт дослідження: *процеси управління метриками при виборі технологій Front end розробки.*

Предметом дослідження *є метрики, методи і засоби оцінювання якості технологій Front end розробки.*

ЗАДАЧІ ДОСЛІДЖЕННЯ

4

Для досягнення мети у дипломній роботі магістра були поставлені і розв'язані **наступні задачі**:

- аналітичний огляд наукових праць і використання FE технологій проєктування ПЗ;
- обґрунтування моделі для вибору оптимальної технології реалізації інтерактивної взаємодії користувача програмного забезпечення;
- обґрунтування набору метрик для кількісного вираження якості FE технологій;
- створення методу оптимального вибору FE технологій;
- проєктування та імплементація програмного інструменту для управління метриками при виборі FE технологій.

МЕТОДИ ДОСЛІДЖЕННЯ

5

Для вирішення задач дипломної роботи використано:

- аналіз та узагальнення – при проведенні аналізу існуючих моделей і метрик якості програмного забезпечення технологій front end розробки;
- метричного аналізу – при побудові моделі і визначенні метрик якості;
- проектування, програмування і тестування – при розробці та апробації програмного засобу управління метриками.

НАУКОВА НОВИЗНА

6

Наукова новизна одержаних результатів полягає в наступному:

- уперше для оцінювання якості технологій front end розробки побудовано модель до складу якої входить два класи Design і Runtime, що сукупно включає 7 характеристик якості (6 стандартизованих+«Конфігурованість») та обгрунтовано використання метрик програмного коду, що дало змогу адекватно у кількісному вигляді представити якість таких технологій і прийняти рішення щодо вибору оптимальної;
- набув подальшого розвитку метод вибору технологій front end розробки, який базується на оцінках ієрархічних рівнів моделі якості і дає можливість враховувати важливість атрибутів і їх прийнятність

АНАЛІЗ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

7

AngularJS на сьогодні є широко використовуваним і популярним фреймворком, який показав ефективність при створенні та супроводі складних веб-додатків. Даний фреймворк з відкритим вихідним кодом позиціонується як розширення HTML для побудови складних веб-орієнтованих додатків.

В основі AngularJS лежить мова програмування JavaScript, що дозволяє використовувати її функції в якості контролерів, сервісів чи фільтрів, а також додаткові методи з інших сторонніх проектів. AngularJS дає змогу скоротити час реалізації веб-додатків за рахунок вбудованих механізмів. Один з них є шаблонізатор (template engine).

АНАЛІЗ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

8

KnockoutJS є ще однією з бібліотек JavaScript, що забезпечує ефективність побудови доволі складних інтерактивних користувацьких інтерфейсів. Характерними особливостями даної бібліотеки є те, що вона при проектуванні інтерфейсів, код задовольняє властивість clean code, є добре структурованим та здатним до розширення, а також зручним для читання.

Основна задача щодо функціональної придатності KnockoutJS – це вирішення задачі автоматичного оновлення користувацького інтерфейсу при оновленні елементів моделі.

АНАЛІЗ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

9

У 2011 році автори соціальної мережі Twitter створили ефективний фреймворк Bootstrap з відкритим вихідним кодом. Ком'юніті Bootstrap весь час зростає та розвивається, що зумовлено потужною підтримкою зі сторони розробників та випуском нових оновлень та функціоналу.

До складу даного фреймворку входять модулі для роботи з HTML (Hyper Text Markup Language), CSS (Cascade Style Sheets) і JavaScript, забезпечено можливість побудови адаптивного дизайну.

КРИТЕРІЇ ЯКОСТІ FRONT END РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

10

Серед основних критеріїв якості front end розробки розрізняють наступні:

- Зв'язність – міра зв'язності програмних елементів всередині одного модуля;
- Зчеплення – міра зв'язності між модулями;
- Ефективність – відношення між очікуваним і фактичним результатом застосування технологій front end розробки;
- Продуктивність – часові та апаратні ресурси для ефективного виконання задач у платформах і фреймворках front end розробки;
- Супроводжуваність –зручність супроводу програмного забезпечення написаного за допомогою front end технології;
- Реактивність – здатність системи швидко реагувати на зміни.

СТРУКТУРА МОДЕЛЕЙ ЯКОСТІ СТАНДАРТУ ISO/IEC 25010 (ISO/IEC 9126)

11

Модель стандарту ISO/IEC 25010, як і ISO/IEC 9126, має ієрарахічну структуру. На верхньому рівні ієрархії знаходяться характеристки якості, після цього іде рівень атрибутів, а на найнижчому рівні знаходяться метрики, які дають змогу кількісно виразити значення атрибутів. Структура моделі якості показана на даному слайді.

$$H = \{H_1, H_2 ... H_n\}$$

$$\updownarrow \quad \updownarrow \qquad\qquad \updownarrow \quad \updownarrow$$

$$A_1 = \{A_{11}, A_{12}, ..., A_{1k}\} \qquad A_n = \{A_{n1}, A_{n2}, ..., A_{ns}\}$$

$$\updownarrow \quad \updownarrow \quad \updownarrow \qquad\qquad \updownarrow \quad \updownarrow \quad \updownarrow$$

$$\{M_{11}, M_{12}, ..., M_{1k}\} \quad ..... \quad \{M_{n1}, M_{n2}, ..., M_{ns}\}$$

СТРУКТУРА МОДЕЛЕЙ ЯКОСТІ СТАНДАРТУ ISO/IEC 25010 (ISO/IEC 9126)

12

Для оцінювання технологій front end розробки програмного забезпечення пропонується скористатись моделлю, що використовується для оцінювання якості архітектури, тобто модель внутрішньої якості. Схематично її представлено на даному слайді.

Функціональність – інтерпретує здатність технологій front end розробки забезпечувати реалізацію деякого функціоналу.

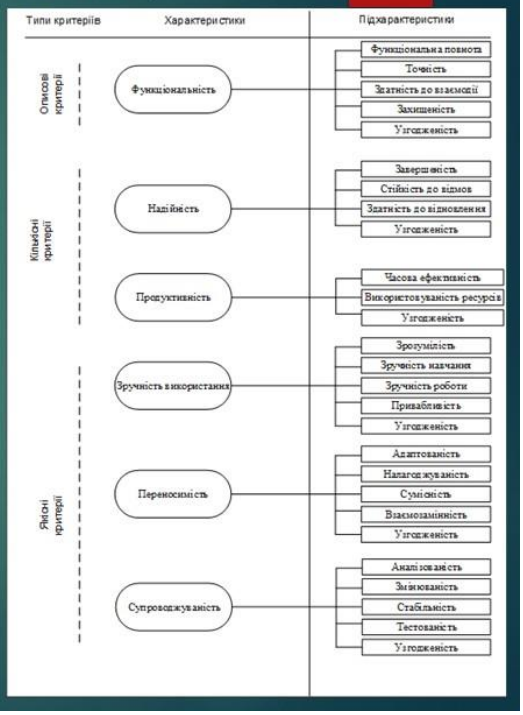
Надійність – представляє собою комплексну сутність щодо підтримки нормального функціонування компонентів розробки при заданих умовах функціонування.

Продуктивність – визначає спроможність технологій реалізовувати деякий функціонал за критеріями апаратних ресурсів та часової ефективності.

Зручність використання – рівень зрозумілості та зручності щодо застосування інструментів front end розробки при реалізації web-додатків.

Переносимість – виражається здатністю технологій працювати у різних середовищах і забезпечувати однакову поведінку та функціональну стабільність.

Супроводжуваність – характеристика, що описує зручність і здатність підтримувати в актуальному та працездатному стані платформу front end розробки, можливість встановлення оновлення, контролю версію і т.п.



ЗАПРОПОНОВАНА МОДЕЛЬ ЯКОСТІ ДЛЯ ОЦІНЮВАННЯ ТЕХНОЛОГІЙ FRONT END РОЗРОБКИ

13

Для збільшення інформативності і групування характеристик якості базової моделі якості пропонується виділити дві категорії комплексних характеристик:

Runtime – представляє групу характеристик з відповідним набором атрибутів, що описують проектовану систему (web-додаток).

Design – характеристики, які стосуються особливостей використання технологій front end розробки на етапі створення програмного забезпечення.

ХАРАКТЕРИСТИКА «ФУНКЦІОНАЛЬНІСТЬ»

14

Характеристику функціональність формують групи атрибутів, які наведено на даному слайді

- функціональна придатність – проявляється у здатності технології забезпечувати реалізацію передбачених функціональних вимог;
- точність – визначає міру відповідності одержаних результатів на запит користувача (розробника);
- здатність до взаємодії – атрибут, що описує можливість взаємодії з іншими технологіями та сторонніми сервісами;
- відповідність стандартам – може утворювати множину атрибутів щодо підтримки та відповідності визначеним стандартам галузі;

МОДЕЛЬ ЯКОСТІ З МЕТРИКАМИ І ВАГОВИМИ КОЕФІЦІЄНТАМИ ДЛЯ КОЖНОГО АТРИУБУТУ 15

Характеристики	Атрибути	Метрики	Ваговий коефіцієнт
Функціональність	Функціональна придатність	Метрика Чепіна	0,7
	Точність	Метрика Холстеда	0,7
	Здатність до взаємодії	Метрика Джилба	0,9
	Відповідність стандартам	Метрика Холстеда	0,9
	Захищеність		0,9
Надійність	Завершеність	Метрика Холстеда	0,8
	Стійкість до відмов		1
	Здатність до відновлення		0,9
	Відповідність стандартам		0,9
Зручність використання	Зрозумілість	Метрика Холстеда	0,8
	Зручність навчання	Метрика Джилба	0,7
	Зручність роботи		0,9
	Привабливість	Метрика Холстеда	0,8
	Відповідність стандартам		0,9
Продуктивність	Часова ефективність	Метрика Чепіна	1
	Ефективність використання ресурсів		0,9
	Відповідність стандартам	Метрика Холстеда	0,9
Зручність супроводу	Аналізованість	Метрика Джилба	0,9
	Зручність внесення змін		0,9
	Стабільність		0,8
	Зручність перевірки		0,7
	Відповідність стандартам	Метрика Холстеда	0,9

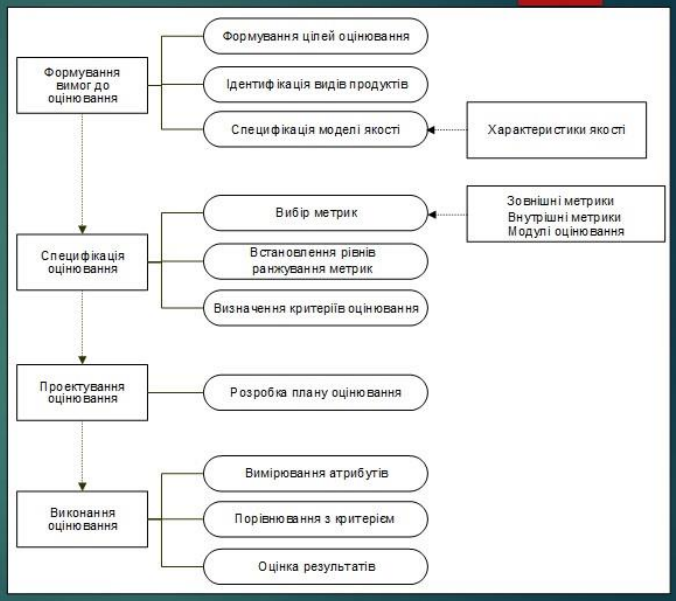
МОДЕЛЬ ЯКОСТІ З МЕТРИКАМИ І ВАГОВИМИ КОЕФІЦІЄНТАМИ ДЛЯ КОЖНОГО АТРИУБУТУ 16

Характеристики	Атрибути	Метрики	Ваговий коефіцієнт
Переносимість	Адаптованість	Метрика Джилба	0,9
	Зручність установки		0,7
	Здатність до співіснування		0,6
	Зручність заміни		0,8
	Відповідність стандартам	Метрика Холстеда	0,9
Конфігурованість	Конфігурованість визначеного набору параметрів	Метрика Холстеда	0,8
	Конфігурованість визначеного набору базових об'єктів		0,8
	Конфігурованість реалізації нових базових об'єктів	Метрика МакКейба	0,9
	Конфігурованість нової реалізації системи		0,8

МОДЕЛЬ ЯКОСТІ З МЕТРИКАМИ І ВАГОВИМИ КОЕФІЦІЄНТАМИ ДЛЯ КОЖНОГО АТРИУБУТУ

17

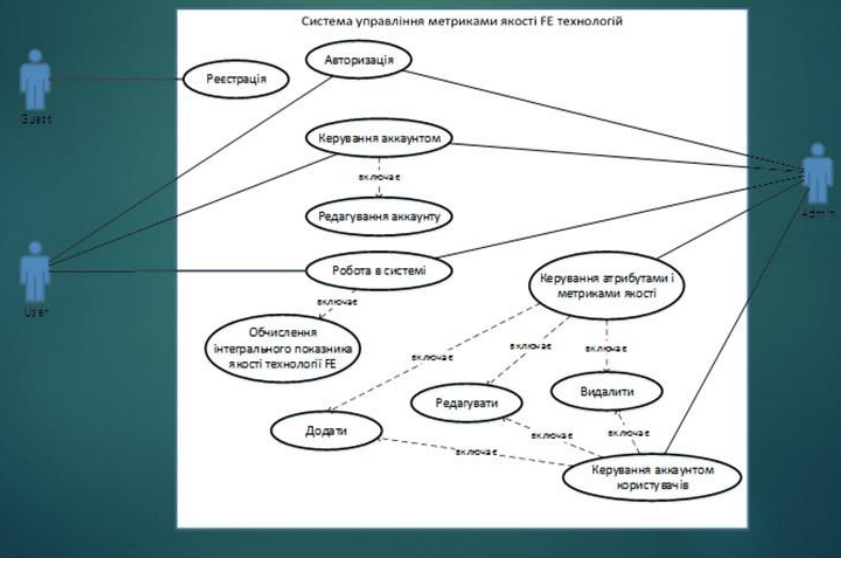
Процедура оцінювання та алгоритм її проведення представлено на даному слайді. Вона передбачає формування вимог до оцінювання, яка включає в себе визначення цілей і мети оцінювання, ідентифікацію і класифікацію виду програмного забезпечення, яке проектується на основі FE технологій та специфікації характеристик якості.



USE CASE ДІАГРАМА

18

Визначення вимог до програмного засобу це одна з невід'ємних складових загального процесу розробки ПЗ. Отже, перед початком розробки такого web-додатку слід визначити вимоги, яким повинна відповідати розроблювана система.



СУТНОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ

19

Сутностями web-додатку підтримки процесу вибору FE технологій є компоненти моделі якості, зокрема атрибути, характеристики і метрики.

№ з/п	Сутність	Атрибути
1	Якість	Дата
		Результат
2	Характеристика моделі якості	Назва
		Атрибути якості
3	Атрибут якості	Назва
		Значення
4	Користувач	Ім'я
		Електронна пошта
		Пароль
		Роль
5	Роль	Guest
		User
		Admin

СУТНОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ

20

Перелік ролей та їх прецедентів наведено у таблиці на даному слайді

№ з/п	Актор	Прецеденти
1	Guest	Реєстрація
		Вхід в систему
2	User	Вхід в систему
		Зміна персональних даних
		Робота з системою
3	Admin	Вхід в систему
		Робота з системою
		Керування атрибутами
		Керування користувачами

ТИПОВА АРХІТЕКТУРА WEB-ДОДАТКУ

21

Згідно типової архітектури, показаної на даному слайді випливає наступний принцип взаємодії компонентів web-додатку:

– користувач взаємодіючи з користувацьким інтерфейсом викликає певні події;

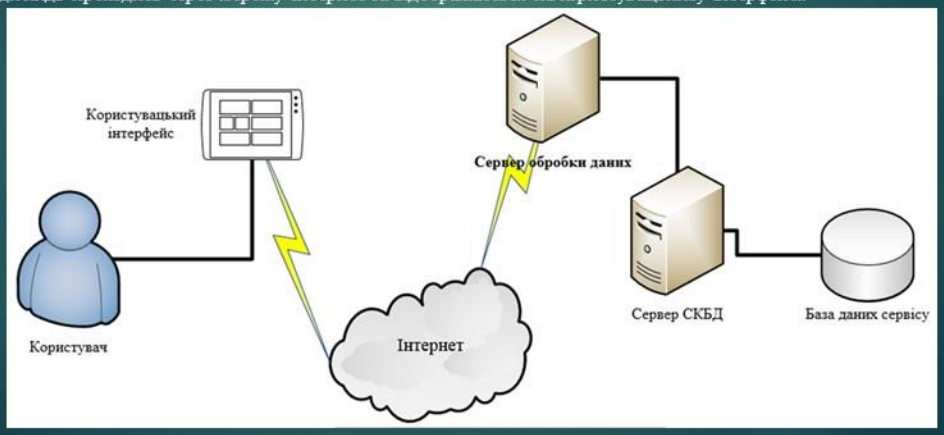
– запит події через мережу Інтернет потрапляє на сервер обробки даних;

– сервер обробки даних обробляє отриманий запит і/або формує відповідь та надсилає її назад клієнтові, і/або звертається до сервера СКБД;

– якщо до сервера СКБД надійшов запит, то відбувається його взаємодія з базою даних та відправка даних у відповідь;

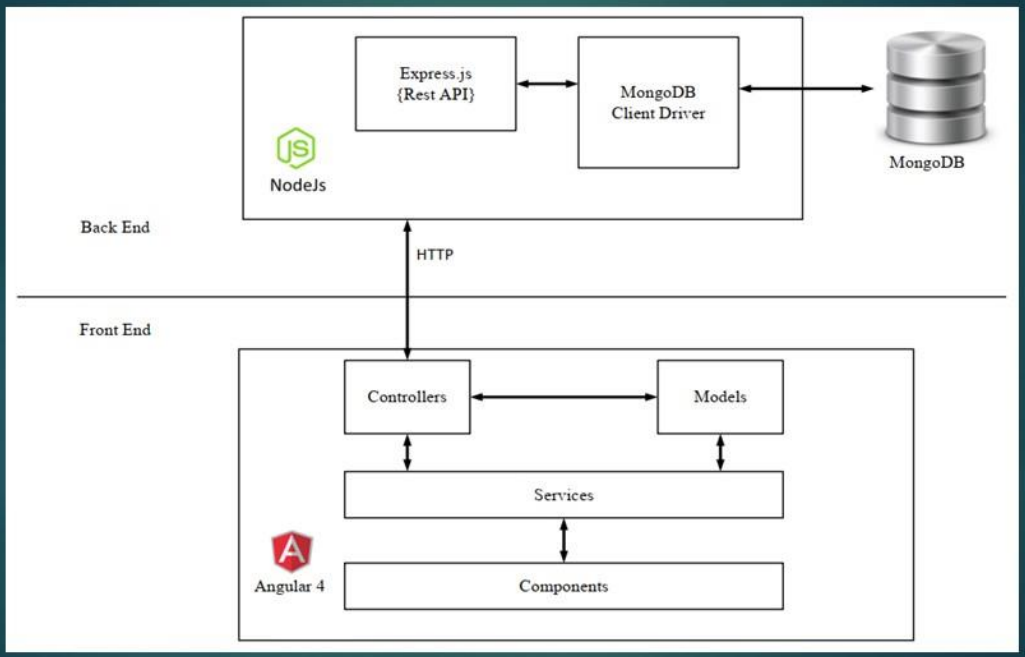
– дані, що надійшли з сервера СКБД потрапляють на сервер обробки даних, який їх структурує та відправляю назад користувачеві;

– відповідь проходить через мережу Інтернет та відображається на користувацькому інтерфейсі.



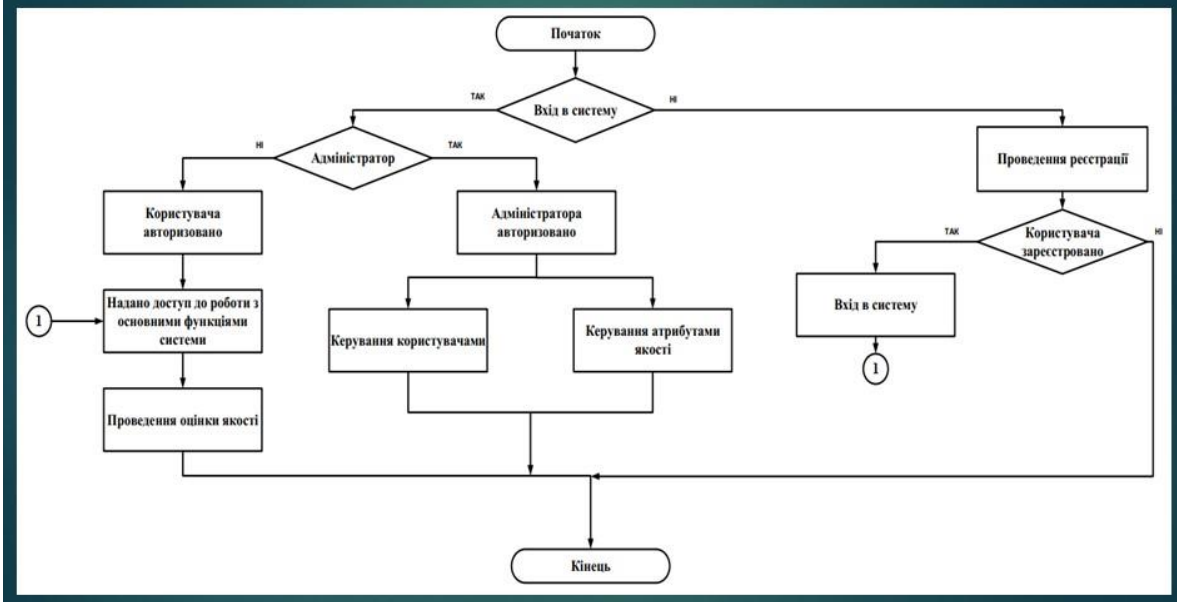
СПРОЕКТОВАНА АРХІТЕКТУРА WEB-ДОДАТКУ

22



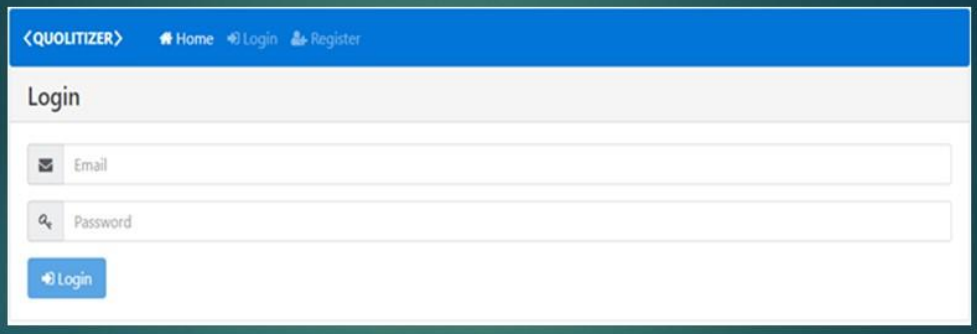
АЛГОРИТМ РОБОТИ WEB-ДОДАТКУ

23



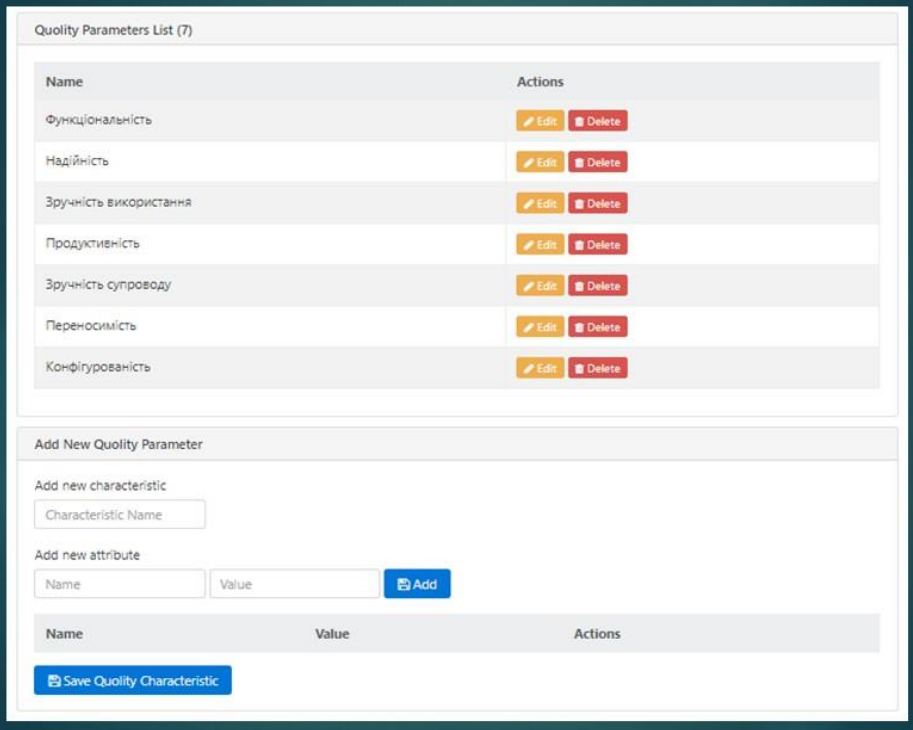
ФОРМА ВХОДУ У СИСТЕМУ

24



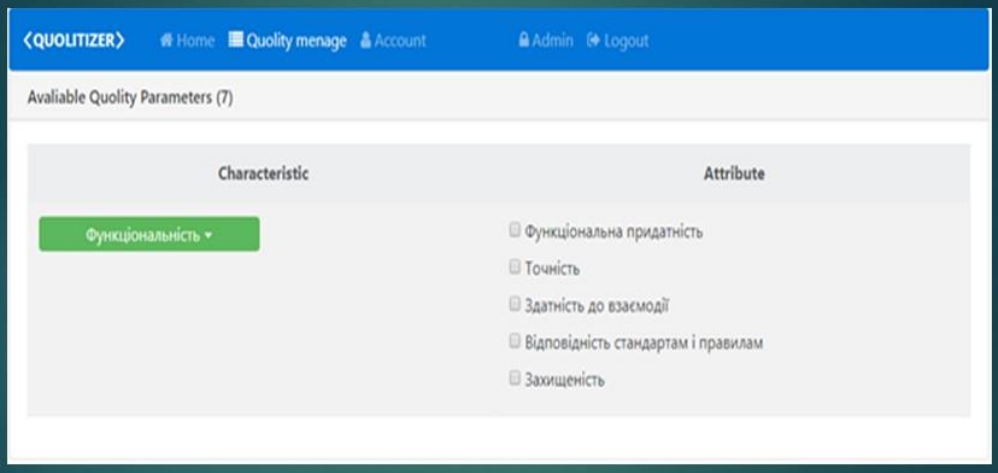
ПАНЕЛЬ УПРАВЛІННЯ ПАРАМЕТРАМИ ЯКОСТІ

25



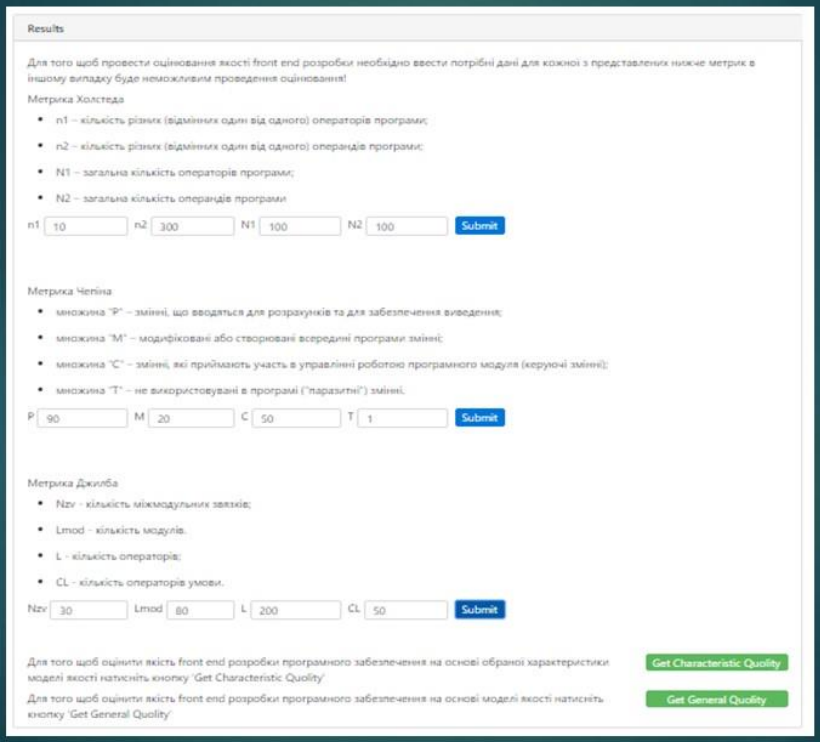
ВІКНО ДОДАТКУ З ОБРАНОЮ ХАРАКТЕРИСТИКОЮ ЯКОСТІ

26



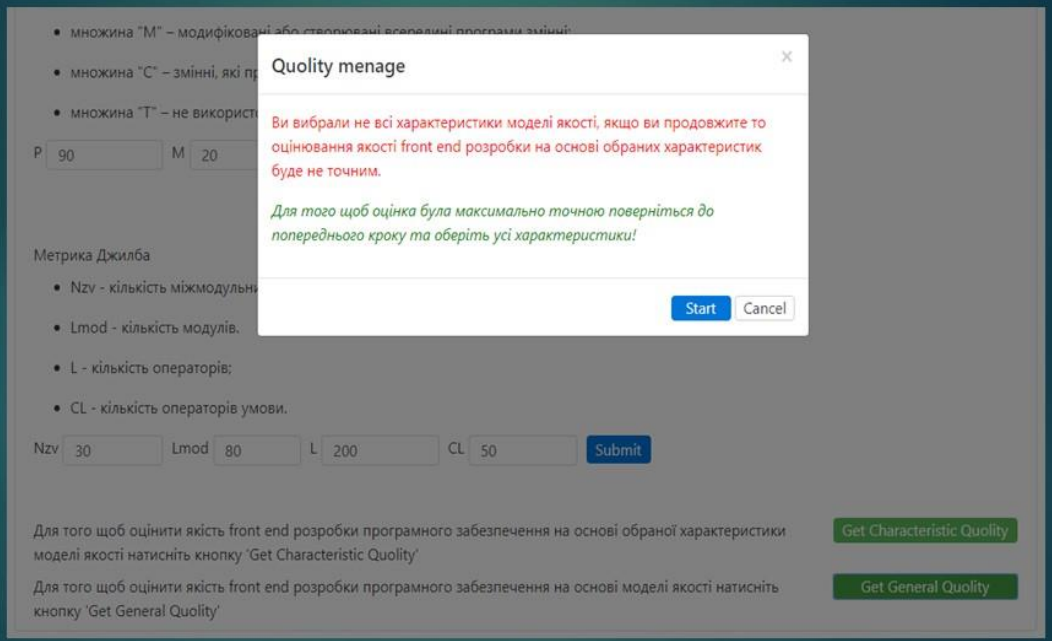
УПРАВЛІННЯ МЕТРИКАМИ ЯКОСТІ

27



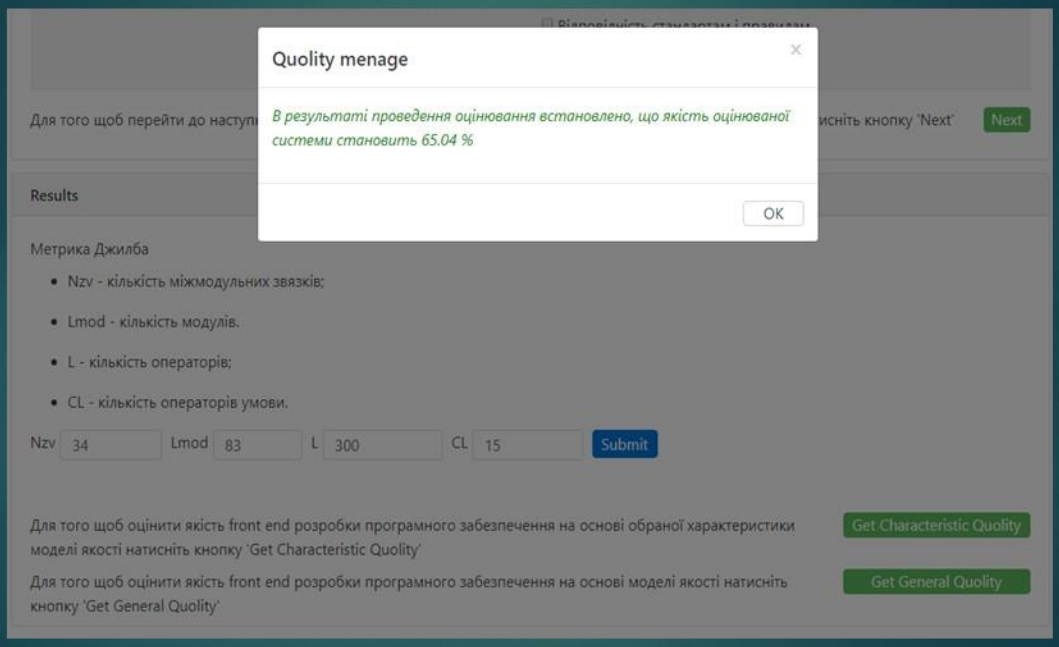
ПОВІДОМЛЕННЯ ПРИ ВИБОРІ НЕ ВСІХ АТРИБУТІВ ЧИ ХАРАКТЕРИСТИК ЯКОСТІ

28



РЕЗУЛЬТАТ ОЦІНЮВАННЯ ТЕСТОВОЇ ПЛАТФОРМИ FRONT END РОЗРОБКИ

29



ВИСНОВКИ

30

У дипломній роботі магістра, одержано наступні основні наукові і практичні результати.

1. Проаналізовано сферу застосування і термінологічні особливості front end технологій у різних галузях інформаційних технологій, що дало змогу виявити їхні спільні риси і чітко встановити, що технології front end розробки програмного забезпечення під web-простір забезпечують інтерактивність користувацького інтерфейсу, а метрики оцінювання якості потребують проведення додаткових досліджень.
2. Проведено аналіз найбільш популярних технологій front end розробки у результаті якого встановлено, що для вибору оптимальних технологій необхідно розробити універсальну та загальноприйняту систему метрик, оскільки існуючі критерії є індивідуальними для кожної з технологій.
3. Проведено аналіз характеристик та метрик якими можна скористатись при оптимальному виборі технологій front end розробки програмного забезпечення, у результаті якого встановлено, що деякі метрики якості потрібно формалізувати, доповнити для забезпечення більш повної картини опису таких технологій.

ВИСНОВКИ

31

4. Проаналізовано сферу застосування і термінологічні особливості front end технологій у різних галузях інформаційних технологій, що дало змогу виявити їхні спільні риси і чітко встановити, що технології front end розробки програмного забезпечення під web-простір забезпечують інтерактивність користувацького інтерфейсу, а метрики оцінювання якості потребують проведення додаткових досліджень.

5. Для представлення якості технологій front end розробки ПЗ для web-простору обгрунтовано застосування моделі якості, яка є стандартизиваною згідно ISO/IEC 25010, володіє ієрархічною структурою і доповнено її характеристикою «Конфігурованість», що дало змогу більш ширше та адекватніше описати властивості FE технологій, порівнюючи з іншими моделями.

6. Визначено і класифіковано за характеристиками елементарні властивості якості FE технологій, що дало змогу обчислювати їх кількісне значення на основі як стандартних метрик якості, так і метрик коду програмного забезпечення.

ВИСНОВКИ

32

7. Для представлення якості технологій front end розробки ПЗ для web-простору обгрунтовано застосування моделі якості, яка є стандартизиваною згідно ISO/IEC 25010, володіє ієрархічною структурою і доповнено її характеристикою «Конфігурованість», що дало змогу більш ширше та адекватніше описати властивості FE технологій, порівнюючи з іншими моделями.

8. Визначено і класифіковано за характеристиками елементарні властивості якості FE технологій, що дало змогу обчислювати їх кількісне значення на основі як стандартних метрик якості, так і метрик коду програмного забезпечення.

9. Обгрунтовано застосування метрик Чепіна, Холстеда, Джилба та МакКейба для оцінювання атрибутів якості технологій FE розробки, що дало змогу спростити сприйняття та підвищити зрозумілість кількісних значень якості відповідних технологій.

10.Запропоновано процедуру та реалізовано алгоритм оцінювання якості FE технологій на різних рівнях ієрархічного дерева моделі якості, що дало змогу керувати як метриками оцінювання, так і процесом прийняття рішень щодо якості елементарних критеріїв, характеристик та інтегральної якості.