

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ВІДСЛІДКУВАННЯ ПОМИЛОК
ERROR TRACKING SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-218сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Сардак Е.М.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сардака Едуарда Михайловича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення відслідковування помилок
Error Tracking Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 20 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “10” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Сардак Е.М.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 27 рис., 1 табл., 3 дод., 17 джерел.

ВІДСЛІДКОВУВАННЯ ПОМИЛОК, ВЕБЗАСТОСУНОК, PHP, MODEL-VIEW-CONTROLLER.

Об'єкт роботи – процес відслідковування помилок. Предмет роботи – програмне забезпечення відслідковування помилок. Мета роботи – розробити програмне забезпечення відслідковування помилок для підтримки роботи команди.

Матеріали, методи та технічні засоби: мова програмування PHP, система керування базами даних MySQL.

Результати. Виконано проєктування і розроблення програмного забезпечення, охоплюючи вибір технології програмування для реалізації програмного забезпечення, в якості якої було обрано мову PHP, визначення функціональних вимог до програми, проєктування бази даних, визначення архітектури застосунку на основі шаблону проєктування MVC, реалізацію відповідних класів моделей, контролерів та вебсторінок з динамічним вмістом.

Висновки. Розроблене програмне забезпечення призначене для підтримки роботи над усуненням помилок зі сторони команди учасників програмних проєктів. Програма дозволяє повідомляти про виявлені помилки, усувати помилки в коді проєкту, перевіряти стан проєктів, помилки яких усунені розробниками, призначати відповідальних розробників за усунення помилок.

Галузь використання. Розроблення програмного забезпечення, підтримка керування програмними проєктами відносно відслідковування помилок.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 27 figures, 1 table, 3 appendixes, 17 sources.

ERROR TRACKING, WEB APPLICATION, PHP, MODEL-VIEW-CONTROLLER.

The object of the work is the process of tracking errors. The subject of the work is error tracking software. The purpose of the work is to develop error tracking software to support the work of the team.

Materials, methods and technical means: PHP programming language, MySQL database management system.

Results. Software design and development were performed. The processes include the choice of programming technology for software implementation, for which PHP language was chosen, definition of functional requirements for the software, database design, definition of application architecture based on MVC design template, implementation of appropriate classes of models, controllers and web pages with dynamic content.

Conclusions. The developed software is designed to support the elimination of errors by the team of participants of software projects. The program allows to report detected errors, eliminate errors in the project code, check the status of projects which errors have been eliminated by the developers, appoint responsible developers for troubleshooting.

Field of use. Software development, software project management support for error tracking.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Проблема відслідковування помилок	9
1.2 Огляд програмних аналогів.....	13
1.3 Висновки за розділом 1	17
2 Проєктування програмного забезпечення	19
2.1 Вибір технології програмування	19
2.2 Вимоги до програмного забезпечення	20
2.3 Проєктування бази даних	23
2.4 Визначення архітектури програмного забезпечення.....	31
2.5 Висновки за розділом 2	32
3 Основні рішення щодо реалізації компонентів системи.....	33
3.1 Опис логіки роботи програми.....	33
3.2 Опис логіки реалізації.....	35
3.3 Висновки за розділом 3	42
4 Експлуатація, тестування та експериментальне дослідження програми	43
4.1 Призначення програми	43
4.2 Умови виконання програми	43
4.3 Тестування сценаріїв роботи з програмою.....	43
4.4 Висновки за розділом 4	52
Висновки	53
Перелік джерел посилення	54
Додаток А Технічне завдання	56
Додаток Б Текст програми	60
Додаток В Слайди презентації.....	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- HTML – HyperText Markup Language;
MVC – Model-View-Controller;
PHP – Personal Home Page.

ВСТУП

Розроблення програмного забезпечення – багатоступеневий процес, який не завершується навіть після впровадження програмного забезпечення. На етапі тестування та на етапі використання програми відповідно тестувальники або кінцеві користувачі програми можуть виявити помилки в її роботі, над якими мають працювати розробники. Усе це триває до кінця життєвого циклу програмного забезпечення. Без якісного інструменту відслідковування помилок цей процес реалізувати ефективно не можна, якщо програма має велику кількість функціональних можливостей до того ж, якщо таких проєктів розроблено компанією багато. Кожну помилку потрібно розподілити між учасниками, визначити відповідального, розв'язати, вносячи зміни в програму, відтестувати, повідомити користувачів про результати та впевнитися в досягненні результатів. Відповідно зробити це під час великої кількості випадків не можна без програмного забезпечення.

Сучасні системи відслідковування помилок є достатньо зручними, але рідко інтегрують до свого складу завдання, пов'язані з роботою цілої команди, адже вони передбачають визначення відповідальних осіб, а всі інші учасники команди розробки при цьому не отримують відповідні дані. До того ж вони передбачають сплату певних коштів за використання послуг, мають пені обмеження зокрема на обсяг даних, кількість учасників тощо, що для корпоративного сектору може викликати деякі незручності. До того ж все це не сприяє конфіденційності даних. Саме на усунення таких незручностей і направлено дану роботу, в якій вирішується актуальне завдання розроблення програмного забезпечення відслідковування помилок.

У результаті метою роботи було визначено розробити програмне забезпечення відслідковування помилок для підтримки роботи команди.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема відслідковування помилок

Звичайно, що у кожній команді розробників є бажання виявити помилки у своєму вебзастосунку до того, як користувачі стикнуться з ними, адже перше враження є основою для сприйняття, тому привітання з повідомленням про помилку – це вірний спосіб відлякати нових відвідувачів, окрім того часто користувачі не повідомляють про те, що застосунок зазнав аварію, але при цьому можуть повідомити про це в соціальній мережі, тому відстеження помилок можна розглядати як процес попереднього виявлення проблем та їх якнайшвидшого усунення, при цьому необхідно враховувати наступне:

- моніторинг помилок може ускладнитися, оскільки ви маєте справу з помилками, про які повідомляють користувачі та ваша виробнича команда, тому ефективне відстеження помилок із самого початку є настільки важливим;

- легше якомога швидше вирішити дрібні проблеми, щоб вони не стали більшими проблемами, таким чином, впровадження відстеження помилок з самого початку циклу розробки заощадить команді багато часу;

- помилки бекенда або сторони сервера включають такі речі, як винятки сторінок та помилки JavaScript: не всі серверні помилки впливають на взаємодію з користувачем, але відстеження їх може бути корисним під час налаштування застосунку;

- не всі помилки коду помітні для користувачів: незалежно від того, якими інструментами команда користується для відстеження винятків, найскладнішою частиною моніторингу помилок є фільтрація всіх повідомлень, щоб зосередитись на реальних проблемах, тому тестери-люди завжди будуть потрібні;

- старі інструменти моніторингу помилок в основному були зосереджені на серверних помилках, при цьому відстеження помилок

бекенда є відносно простим, оскільки весь код на стороні сервера працює в одному і тому ж місці, у свою чергу моніторинг помилок інтерфейсу вимагає трохи більше роботи, оскільки продуктивність інтерфейсу сильно залежить від апаратного забезпечення, програмного забезпечення та підключення користувача, потрібно переконатись, що вебсайт або програма працює безперебійно в понад десятку веббраузерів та сотнях різних пристроїв, що є непростю задачею;

- помилки інтерфейсу включають збої в користувацькому інтерфейсі, повільну продуктивність та пошкоджені інтерфейси: окрім збору даних, необхідно чітко розуміти, як помилки впливають на взаємодію з користувачем;

- відтворення помилок у веброзробці означає бачити помилки самотійно, відтворюючи умови, за яких вони виникають: якщо помилку не вдасться відтворити, то швидше за все її і вирішити не вдасться;

- кожен вебсайт або застосунок повинен надавати користувачам можливість повідомляти про помилки;

- звичайно, бувають випадки, коли проблема не у програмі, а в підключенні користувача або конфігурації мережі: певні рішення відстеження помилок дозволяють записувати та відтворювати сеанси користувача, щоб можна було на власні очі побачити, що саме сталося у користувача [1].

Раніше, коли інструменти відстеження помилок не були настільки поширеними і до того ж вдосконаленими, багато розробників, як правило, налаштовували сповіщення електронною поштою для інформування про помилки, у такому випадку, в основному, коли виникає нова помилка, скрипт надсилає необроблені винятки у папку вхідних листів розробника, але цей процес ускладнювався одразу, як програма ставала великою, і міг контролюватися таким чином тоді, коли програма навпаки була невеликою і відповідно генерувала мінімальні помилки, у той же час як тільки додавались різні фрагменти, сповіщення стають надто активними, а це призводить до

того, що помилки викликають менше цікавості, менше уваги, а відповідно і результат погіршується, адже в такому випадку можна було б пропустити важливу помилку, приховану серед купи помилок, які насправді взагалі не мали б хвилювати, але на даному етапі розвитку інструменти відслідковування помилок стали значно більш поширеними, об'єднують між собою помилки, які сприймаються як подібні, що у свою чергу не тільки зменшує кількість сповіщень, але також дозволяє краще визначити пріоритетність серйозності помилок, це все спрямоване на створення робочого циклу для розробників програмного забезпечення, щоб вони були більш продуктивними та швидко виправляли помилки, перш ніж вони спричинять більші проблеми [2].

Система відстеження помилок, яку також називають системою відстеження дефектів, – це програмний застосунок, який фіксує, повідомляє та керує даними про помилки та винятки, що виникають у програмному забезпеченні, щодо якого можна визначити наступне:

- часто система відстеження помилок інтегрована з програмним забезпеченням для кейс-менеджменту, воно автоматизує процес моніторингу і відстеження помилок та інших проблем, які можуть заважати ефективному функціонуванню технологічної та інформаційної інфраструктури компанії;

- система відстеження дефектів є життєво важливою частиною розробки програмного забезпечення, провідні розробники та команди програмного забезпечення постійно ним користуються, більшість систем відстеження помилок дають кінцевим користувачам можливість безпосередньо вводити звіти про помилки [3].

До основних етапів відстеження помилок належать:

- виявлення помилок: помилки виявляються командою розробників на етапі тестування продукту, вони також можуть бути виявлені та повідомлені кінцевими користувачами;

- звітування про помилку: розробник ідентифікує помилку та реєструє час її виявлення у звіті про помилку;

- виправлення помилок: розробники намагаються виправити помилку різними підходами;

- повторне тестування програмного забезпечення: програмне забезпечення тестується неодноразово, щоб переконатися, що воно працює належним чином;

- збір даних: усі дані, пов'язані з помилкою, записуються у звіт про помилку, щоб уникнути того ж самого в майбутньому [3].

За час свого існування помилка може пройти декілька стадій або станів, що включають:

- активний: дослідження триває;

- тест: помилка виправлена і готова до тестування;

- перевірено: результат виконання протестовано і перевірено системою забезпечення якості;

- закрито: помилка може бути закрыта після повторного тестування якості або якщо вона більше не вважається дефектом;

- повторно відкрито: помилку відновлено [4].

Управління помилками здійснюється на основі пріоритету та серйозності (критичності) помилки, при цьому рівні серйозності допомагають визначити відносний вплив проблеми на випуск продукту, кількість цих класифікацій може різнитися, але вони, як правило, включають певну форму (рівень):

- катастрофічний: спричиняє повний збій програмного забезпечення або непоправну втрату даних, за даного випадку немає обхідних шляхів, і продукт не може бути випущений;

- порушення функціональності: можливо, існує обхідний шлях, але він незадовільний, програмне забезпечення не може бути випущено;

- вихід з ладу некритичних систем: існує достатньо задовільний обхідний шлях, продукт може бути випущений, якщо помилка задокументована;

– зовсім незначний: є обхідний шлях, або проблему можна ігнорувати, в результаті помилка не впливає на випуск продукту [4].

Вимогами до систем відслідковування помилок є:

– забезпечення процесу реєстрації помилок: програмне забезпечення для відстеження помилок має вміти правильно описувати помилку, це повинно дозволити розробникам чітко зрозуміти помилку, але якщо про помилку не повідомляється або обов’язкові поля є неповними, це негативно впливає на процес розробки програмного забезпечення;

– шаблон відстеження помилок: члени команди розробників працюють разом над подібним шаблоном відстеження помилок, оскільки використання різних платформ може спричинити розбіжності, таким чином, щоб уникнути цієї плутанини, усі розробники використовують однакові шаблони, щоб спростити звітування;

– рівні пріоритету: команди тестування та розробки призначають рівні пріоритети кожній помилці, яка з’являється в застосунку, це дозволяє спочатку працювати над питаннями з високим пріоритетом, а потім звертатися до менш важливих помилок, програмне забезпечення для відстеження помилок працює ефективно лише тоді, коли команди ефективно визначають ці завдання;

– інструменти відстеження помилок: система управління дефектами представляє інструменти відстеження помилок, які дозволяють тестувальникам ефективно виконувати свою роботу, для випуску якісного програмного забезпечення тестувальники вимагають інструментів відстеження помилок для своїх проєктів [5].

1.2 Огляд програмних аналогів

Backlog [6] (рис. 1.1) – це онлайн-програма для управління проєктами, яка побудована передовсім для програмістів: наприклад, має перевірку

різниці між файлами та інструменти для контролю версій і об'єднання фрагментів коду [7].

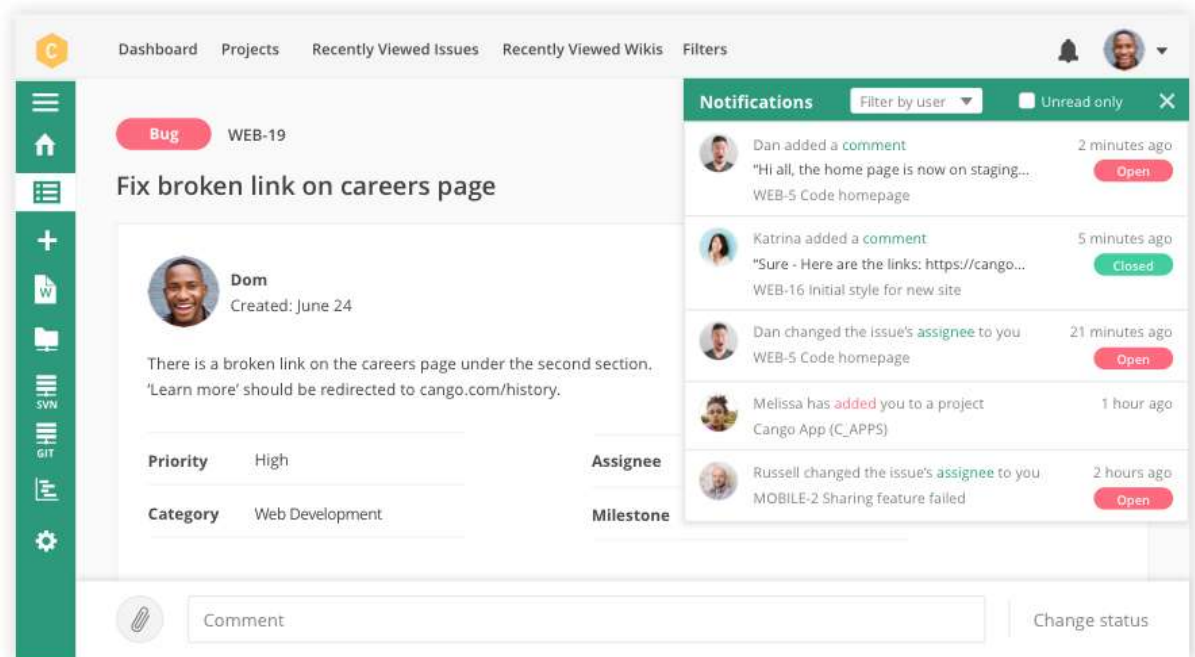


Рисунок 1.1 – Інтерфейс програми Backlog [8]

У Backlog використовується поняття проблеми, а не помилки. Це пов'язано з тим, що засоби, які створюються програмою, є ширшими і включають окрім помилки ще й завдання, запити.

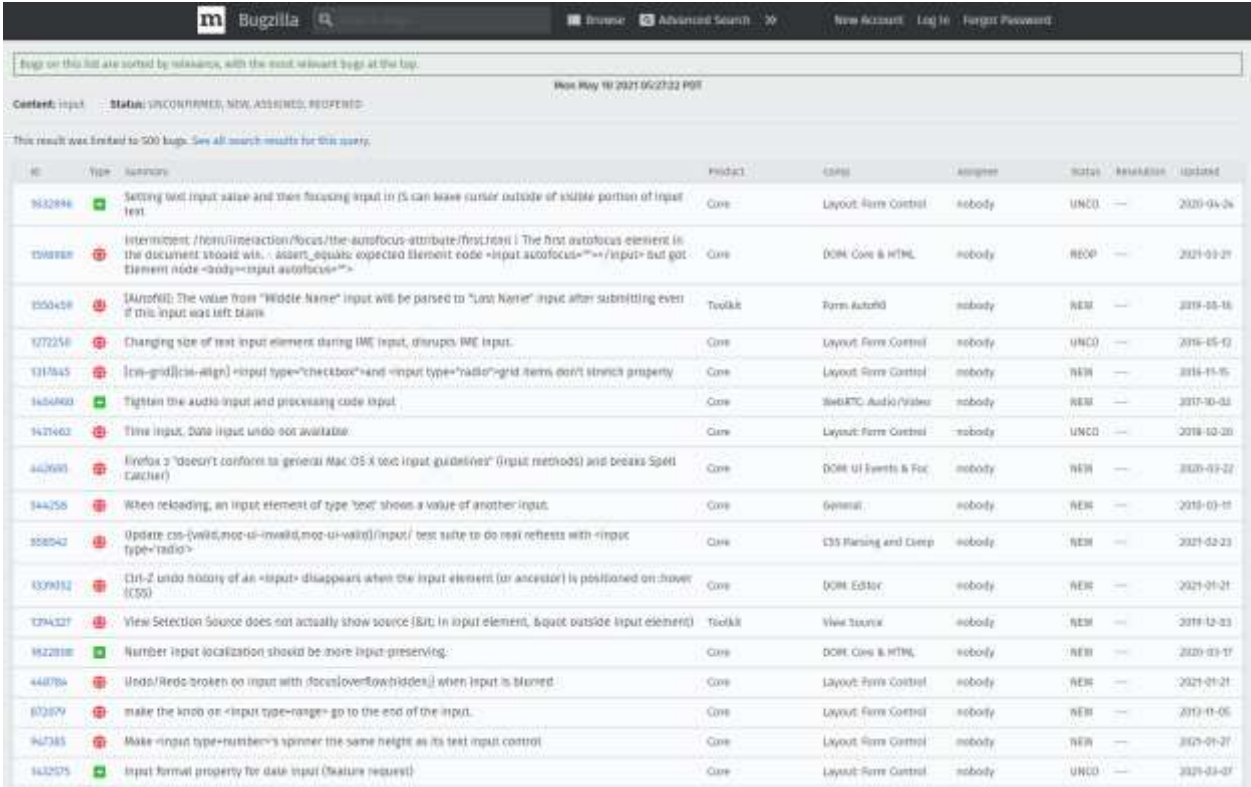
Програма дозволяє класифікувати проблему, додати рядок теми, написати опис, використовуючи відформатований текст, оцінити її пріоритет та додати відповідальну особу, послідовників, віху, версію, дату початку, термін виконання, приблизні години та фактичні години, також можна завантажувати файли, окрім того під час коментування проблеми можна заповнити поле для того, кого слід повідомити [7].

Backlog має наступні варіанти сплати за послуги:

– безкоштовний план: підтримує 10 користувачів, один проєкт та надає 100 МБ сховища файлів, пропонує такі функції, як управління командою, відстеження проблем, підтримка чату та електронної пошти, а також мобільні програми;

– платні плани: початковий план коштує 35 доларів для 30 користувачів (плата відбувається щорічно) і включає п'ять проєктів та 1 ГБ місця для зберігання, пропонує додаткові функції управління проблемами [7].

Bugzilla [9] (рис. 1.2) – це система відстеження дефектів або система відстеження помилок: системи відстеження дефектів дозволяють командам розробників ефективно відстежувати виправлені помилки, проблеми, вдосконалення та інші запити на зміни у своїх продуктах, прості можливості відстеження дефектів часто вбудовані в інтегровані середовища управління вихідним кодом, такі як Github або інші віддалено або локально встановлені еквіваленти [9].



ID	Type	Summary	Product	Component	Assignee	Status	Resolution	Updated
963284	NEW	Setting tool input value and then focusing input in JS can leave cursor outside of visible portion of input text	Core	Layout: Form Control	nobody	UNCO	---	2020-04-26
130888	NEW	Intermittent / dom/interaction/focus/the-autofocus-attribute/first-focus The first autofocus element in the document should wth... assert_equals expected element code <input autofocus="" /> but got element node <body=>input autofocus="">	Core	DOM: Core & HTML	nobody	NEW	---	2021-03-21
1550458	NEW	[Autofill] The value from "Middle Name" input will be parsed to "Last Name" input after submitting even if this input was left blank	Toolkit	Form Autofill	nobody	NEW	---	2019-05-16
1272258	NEW	Changing size of text input element during IME input, disrupts IME input.	Core	Layout: Form Control	nobody	UNCO	---	2019-05-02
1317845	NEW	[css-grid][css-align] <input type="checkbox"> and <input type="radio"> grid items don't stretch properly	Core	Layout: Form Control	nobody	NEW	---	2019-11-15
1458460	NEW	Tighten the audio input and processing code input	Core	WebRTC: Audio/Video	nobody	NEW	---	2017-10-02
1421462	NEW	File input, Data input undo not available	Core	Layout: Form Control	nobody	UNCO	---	2018-10-08
443681	NEW	Firefox 3 "doesn't conform to general Mac OS X text input guidelines" (input methods) and breaks Spell Catcher	Core	DOM: UI Events & Foc	nobody	NEW	---	2008-07-27
344256	NEW	When reloading, an input element of type "text" shows a value of another input.	Core	General	nobody	NEW	---	2010-03-11
308042	NEW	Update css-(valid,moz- -invalid,moz- -valid)/input/ best suite to do real reflows with <input type="radio">	Core	CSS Parsing and Comp	nobody	NEW	---	2021-02-23
433932	NEW	Ctrl-Z undo history of an <input> disappears when the input element (or ancestor) is positioned on:drawer (CSS)	Core	DOM: Editor	nobody	NEW	---	2021-01-21
1294327	NEW	View Selection Source does not actually show source (< in input element, " outside input element)	Toolkit	View Source	nobody	NEW	---	2019-12-23
962288	NEW	Number input localization should be more input-preserving.	Core	DOM: Core & HTML	nobody	NEW	---	2020-03-17
448784	NEW	Undo/Redo broken on input with focus/overflow/hidden when input is blurred	Core	Layout: Form Control	nobody	NEW	---	2021-01-21
832879	NEW	make the knob on <input type="range"> go to the end of the input.	Core	Layout: Form Control	nobody	NEW	---	2013-01-05
962385	NEW	Make <input type="number">'s spinner the same height as its text input control	Core	Layout: Form Control	nobody	NEW	---	2020-01-27
1432575	NEW	Input format property for date input (feature request)	Core	Layout: Form Control	nobody	UNCO	---	2019-03-07

Рисунок 1.2 – Інтерфейс програми Bugzilla для Mozilla Firefox

Bugzilla надає користувачам наступні функціональні можливості:

– пошук помилок, що включає базовий пошук помилок, схожий на Google, простий для нових користувачів та такий, що здійснює пошук

повного тексту помилки, та вдосконалену систему пошуку, де можна створити будь-який потрібний пошук, включаючи пошук за часом (наприклад, «показати помилки, де пріоритет змінився за останні 3 дні») та інші запити;

- сповіщення електронною поштою, керовані налаштуваннями користувача: можна отримати електронний лист про будь-які зміни, внесені в Bugzilla;

- списки помилок у декількох форматах (Atom, iCal тощо): при пошуку помилок можна отримати результати у багатьох різних форматах, а не тільки у простому базовому HyperText Markup Language (HTML)-макеті, доступно багато форматів, включаючи формат для імпорту в електронні таблиці Excel та різні формати XML;

- заплановані звіти (щоденні, щотижневі, щогодинні тощо) електронною поштою: Bugzilla має систему, яка надішле користувачеві або групі користувачів, що була вказана, результати пошуку за розкладом, який було вказано;

- відстеження часу: можна підрахувати, скільки годин знадобиться для виправлення помилки, також можна встановити кінцевий термін, до якого повинна бути вирішена помилка [10].

Zoho BugTracker [11] (рис. 1.3) – це інструмент відстеження помилок, який дозволяє фіксувати деталі, такі як те, що потрібно виправити, кому призначений запит і коли він буде виправлений, також можна переглянути ці дані у форматі списку або картки Kanban, можна налаштувати вигляд картки за допомогою фільтрів, також можна експортувати цю інформацію як файл електронної таблиці та отримувати сповіщення електронною поштою про стан вирішення помилок, програма може використовуватися на основі наступних тарифних планів:

- безкоштовний план: включає трьох користувачів, один проєкт, 10 Мб місця для зберігання, основні звіти та такі функції, як відстеження помилок, управління документами та імпорт помилок;

– платні плани: перший рівень коштує 3 долари на користувача на місяць для 10 користувачів, він включає 10 проєктів, 5 Гб місця для зберігання та додаткові функції, такі як шаблони проєктів, відстеження часу, експорт помилок, автоматичний збір деталей помилок з електронних листів, стан помилок та розширені звіти про помилки [7].

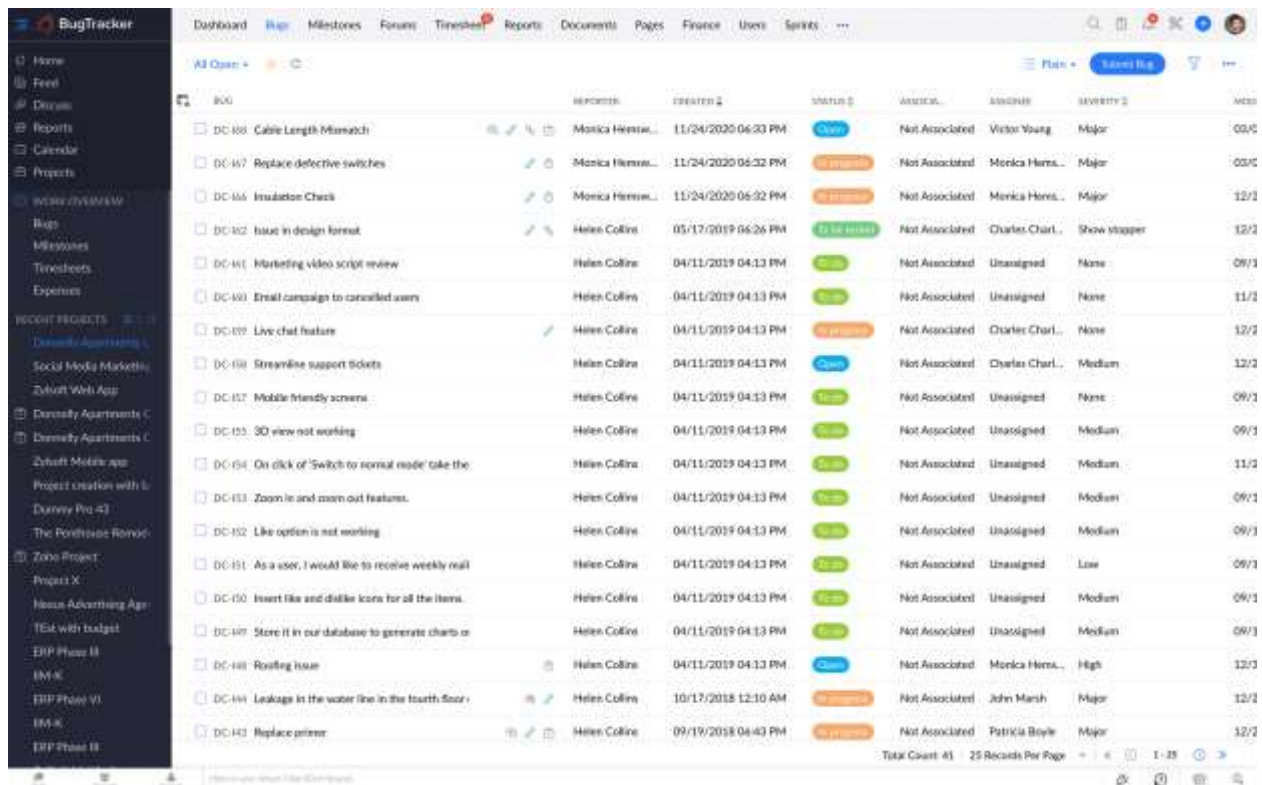


Рисунок 1.3 – Інтерфейс програми Zoho [12]

1.3 Висновки за розділом 1

У розділі проаналізовано проблему відслідковування помилок, зокрема розглянуто передумови створення систем відслідковування помилок, актуальний стан проблеми, етапи відслідковування помилок, стани помилок, їх рівні критичності, вимоги до систем. Проаналізовано програмні аналоги програмного забезпечення, що розробляється, визначено основні їх функціональні можливості.

Для того щоб досягти поставлену мету, в роботі було визначено наступні завдання:

- визначення вимог до програмного забезпечення відслідковування помилок;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір технології програмування

Найбільш придатним рішенням для розроблення програмного забезпечення відслідковування помилок є створення вебзастосунку. Це спростить доступ різних типів користувачів, зокрема і звичайних користувачів, які працюють з програмою, а не тільки безпосередніх учасників проєкту, що дозволить розширити використання програми на подальші етапи життєвого циклу проєкту.

Технологіями, які дуже часто використовуються для створення серверної сторони програмного забезпечення, є інструменти Personal Home Page (PHP) [13]-[14] та Node.js. У табл. 2.1 виконано порівняння даних технологій.

Таблиця 2.1 – Вибір технології програмування

Критерій	PHP	Node.js
Тип	Скриптова мова серверної сторони	Середовище виконання
Швидкість завершення проєкту	Більша	Менша
Легкість доступу до баз даних	Більша	Менша
Продуктивність	Менша	Більша
Бібліотека готових модулів	Більша	Менша
Веброзробка	Так	Так

Порівняння технологій за основними критеріями показало:

– PHP відзначається більшою швидкістю завершення проєкту, оскільки мова загалом є простішою у використанні, має менше специфічних

засобів, характерних для JavaScript, пов'язаних зокрема з функціональним програмуванням, створити і запустити готовий проєкт мовою PHP швидше;

- легкість доступу до бази даних більша у PHP, оскільки код з'єднання і запитів прямо вбудовується в код програми, до того ж не потрібні додаткові інтерфейси;

- рішення, створені мовою PHP, мають меншу швидкість виконання, але варто відзначити, що для проєкту, який створюється в роботі, це не є критичним, адже потенційно він не буде перевантаженим учасниками: основними учасниками будуть учасники проєктів, якими займається компанія;

- більше готових модулів, що підтримують створення коду для різних задач, існує для мови PHP.

Зважаючи на описану логіку, саме мова програмування PHP була обрана для створення вебзастосунку відстеження помилок.

2.2 Вимоги до програмного забезпечення

Для підтримки процесу відслідковування помилок програма повинна забезпечувати виконання наступних функціональних вимог:

- авторизація;
- реєстрація;
- внесення нової помилки;
- перегляд усіх помилок, внесених користувачем;
- перегляд усіх внесених помилок;
- перегляд помилок за проєктом;
- перегляд інформації про помилку;
- перегляд пов'язаних помилок з даною;
- встановлення пріоритету помилки;
- встановлення критичності помилки;
- перегляд помилок за заданим станом;

- пошук помилок за назвою;
- відображення помилок, за які відповідальний розробник;
- призначення розробника, відповідального за усунення помилки;
- визначення строків усунення помилки;
- перегляд помилок, не призначених розробникам;
- перегляд помилок, готових до тестування;
- перегляд історії оновлення помилки;
- створення коментарів до помилки;
- перегляд коментарів до помилки;
- перегляд оновлень помилок розробником;
- внесення і редагування проєктів;
- внесення версій проєктів;
- створення команд розробників;
- зв'язування помилки з існуючою помилкою;
- перегляд адресованих коментарів;
- повторне відкриття помилки;
- визначення успішного усунення помилки;
- визначення успішної перевірки помилки;
- закриття помилки;
- визначення неможливості відтворення помилки.

Усі користувачі програми розглядаються як учасники (учасники програмних проєктів). Окремо серед них можна виділити такі групи користувачів (рис. 2.1):

- розробники, які в даному процесі відповідальні за усунення виявлених помилок у програмах;
- тестувальники, які можуть виявляти помилки, а також є відповідальними за перевірку усунення помилок після роботи, виконаної розробниками;
- менеджери, які відповідальні за розподілення відповідальності в межах роботи над усуненням помилок;

– звичайні користувачі, які можуть вносити інформацію про помилки та відслідковувати розв’язання їх проблем.



Рисунок 2.1 – Діаграма прецедентів

Розробники у процесі роботи над помилками можуть переглядати

необхідні їм дані про помилки та редагувати їх стан, визначаючи вирішення проблеми або неможливість повторення виявленої ситуації. Це їх основна відповідальність.

Менеджери можуть редагувати помилки шляхом встановлення відповідальних та визначення параметрів помилок.

Тестувальники можуть вносити дані про виявлені помилки, переглядати помилки, їх стан, а також редагувати їх стан.

Звичайні користувачі можуть тільки вносити дані про виявлені помилки і переглядати дані тільки про власні внесені помилки, на цьому їх роль в системі обмежена.

Варто зауважити, що незважаючи на спільне використання сценарію перегляду помилок, цей сценарій по-різному відображається різними акторами. Окрім того і змінювання стану помилки є спільним прецедентом, але різні значення стану помилки можуть встановити різні актори.

2.3 Проєктування бази даних

Проєктування бази даних виконано на основі використання системи управління базами даних MySQL тому, що зв'язка мови програмування PHP та системи MySQL відзначається наступними перевагами:

- відкрита платформа;
- кросплатформність;
- частота оновлень.

За результатами проєктування бази даних було виділено наступні основні сутності предметної області:

- сутність учасника (користувача) системи User, яким може виступати розробник, тестувальник, користувач;
- сутність проєкту Project;
- сутність версії проєкту Version;
- сутність помилки Bug;

- сутність коментаря до помилки Comment;
- сутність оновлення помилки Update;
- сутність операційної системи OS;
- сутність мови програмування Language;
- сутність команди з розробників Team, що працює над проєктом.

Таблиця Participant містить наступні поля:

- id – номер учасника;
- surname – прізвище учасника;
- name – ім'я учасника;
- patronymic – по батькові учасника;
- email – електронна пошта учасника;
- password – пароль учасника;
- position – посада учасника;
- rights – сторона, яку представляє учасник (розробник, тестувальник).

Таблиця Project містить наступні поля:

- id – номер проєкту;
- name – назва проєкту;
- team_id – команда розробників, що працює над проєктом;
- created – дата створення проєкту;
- path – розташування проєкту.

Поле team_id є зовнішнім ключем таблиці Team, визначає відношення один-до-багатьох між таблицями, оскільки кожен проєкт може обслуговуватися однією командою, а кожна команда може працювати над різними проєктами.

Таблиця Bug містить наступні поля:

- id – номер помилки;
- name – назва помилки;
- description – текстовий опис помилки;
- conditions – умови, за яких спостерігалась помилка;
- device – пристрій, на якому виявлено помилку;

– software – програма, в якій встановлено помилку (для вебзастосунків);

– os_id – операційна система, під час роботи в якій було виявлено помилку;

– duedate – дата, до якої має бути вирішена проблема;

– state – стан помилки;

– priority – пріоритет помилки;

– criticality – рівень критичності помилки;

– steps – кроки, які потрібно виконати для вирішення проблеми;

– waitedres – очікуваний користувачем результат;

– obtainedres – отриманий фактично користувачем результат (різниця між waitedres і obtainedres і визначає помилку за думкою користувача, який її виявив);

– obtained – час, коли було виявлено помилку;

– updated – момент останнього оновлення даних помилки;

– responder_id – відповідальний за вирішення помилки;

– founder_id – тестувальник (або користувач), який виявив помилку;

– project_id – проєкт, до якого належить помилка.

Поле os_id є зовнішнім ключем таблиці OS, визначає відношення один-до-багатьох між таблицями, оскільки кожна помилка може бути виявлена у конкретній операційній системі, але до однієї операційної системи може бути віднесено різні помилки.

Поле responder_id є зовнішнім ключем таблиці Participant, визначає відношення один-до-багатьох між таблицями, оскільки для кожної помилки може бути встановлено одного відповідального, але кожен розробник може відповідати за усунення різних помилок.

Поле founder_id є зовнішнім ключем таблиці Participant, визначає відношення один-до-багатьох між таблицями, оскільки повідомити про помилку має конкретний користувач, але кожен користувач може повідомити про різні помилки.

Поле `project_id` є зовнішнім ключем таблиці `Project`, визначає відношення один-до-багатьох між таблицями, оскільки кожна помилка може стосуватися тільки одного конкретного проєкту, але при цьому до одного проєкту можуть відноситися різні помилки.

Таблиця `Version` містить наступні поля:

- `id` – номер версії;
- `project_id` – проєкт;
- `number` – номер версії;
- `path` – шлях до версії з програмою.

Поле `project_id` є зовнішнім ключем таблиці `Project`, визначає відношення один-до-багатьох між таблицями, оскільки кожна версія проєкту може відноситися тільки до одного конкретного проєкту, але кожен проєкт може мати різні версії.

Таблиця `Comment` містить наступні поля:

- `id` – номер;
- `participant_id` – учасник, який створив коментар;
- `bug_id` – помилка, до якої належить коментар;
- `created` – час створення помилки;
- `quot` – текст коментаря.

Поле `participant_id` є зовнішнім ключем таблиці `Participant`, визначає відношення один-до-багатьох між таблицями, оскільки кожен коментар може належати тільки одному користувачу, але кожен користувач може створити різні коментарі.

Поле `bug_id` є зовнішнім ключем таблиці `Bug`, визначає відношення один-до-багатьох між таблицями, оскільки кожен коментар може стосуватися тільки однієї конкретної помилки, але кожна помилка може мати різні коментарі.

Таблиця `CommentMessage` містить наступні поля:

- `id` – номер;
- `comment_id` – коментар;

– participant_id – користувач, якому має бути адресовано коментар.

Таблиця CommentMessage направлена на збереження адресатів коментарів, що дозволяє відправляти їх як повідомлення розробникам за необхідності, надаючи можливість працювати одразу з декількома учасниками. Таблиця реалізує зв'язок один-до-багатьох між таблицями Comment і Message, оскільки кожен коментар може адресуватися різним учасникам, до того ж кожен учасник може отримати асоціацію з різними коментарями.

Таблиця Update містить наступні поля:

- id – номер оновлення помилки;
- bug_id – номер помилки;
- state – новий стан помилки;
- priority – новий пріоритет помилки;
- criticality – нова критичність помилки;
- updated – час оновлення помилки;
- responder_id – розробник, який виконав оновлення;
- description – опис оновлення;
- version_id – версія програми, за якої сталося оновлення помилки.

Поле responder_id є зовнішнім ключем таблиці Participant, визначає відношення один-до-багатьох між таблицями, оскільки кожне оновлення може виконуватися тільки одним учасником (розробником, тестувальником або менеджером).

Поле bug_id є зовнішнім ключем таблиці Bug, визначає відношення один-до-багатьох між таблицями, оскільки кожне оновлення стосується однієї конкретної помилки, але кожна помилка може мати різні оновлення.

Поле version_id є зовнішнім ключем таблиці Version, визначає відношення один-до-багатьох між таблицями, оскільки кожне оновлення стосується конкретної версії проєкту, яка в даний момент є поточною, але кожна версія може мати різні оновлення.

Таблиця OS містить наступні поля:

- id – номер операційної системи;
- name – назва операційної системи.

Таблиця Language містить наступні поля:

- id – номер мови програмування;
- name – мова програмування.

Таблиця Team містить наступні поля:

- id – номер команди;
- name – назва команди;
- manager_id – менеджер, який у команді відповідальний за розподіл

відповідальності за усунення помилок між учасниками команди.

Поле manager_id є зовнішнім ключем таблиці Manager, визначає відношення один-до-багатьох між таблицями, оскільки кожна команда повинна мати одного менеджера, а кожен менеджер може брати участь в роботі різних команд.

Таблиця ParticipantTeam містить наступні поля:

- id – номер запису;
- team_id – номер команди;
- participant_id – номер учасника, який належить до команди.

Таблиця ParticipantTeam реалізує зв'язок один-до-багатьох між таблицями Participant і Team, оскільки кожен користувач може брати участь в різних командах, до того ж одна команда складається з різних учасників.

Таблиця ProjectOS підтримує можливість реалізації кросплатформних проєктів, адже у іншому випадку створюється окрема версія, тоді окремий застосунок можна визначити як окремий проєкт і зв'язати між собою різні застосунки, що реалізують версії для різних операційних систем. Для кросплатформної програми таке зробити не можна, тому виділено дану таблицю, що реалізує зв'язок багато-до-багатьох між таблицями Project і OS: кожна операційна система може мати різну кількість проєктів, кожен проєкт може мати версії для різних операційних систем.

Таблиця містить наступні поля:

- id – номер запису;
- project_id – проєкт;
- os_id – операційна система, в якій працює проєкт.

Таблиця ConnectedBugs визначає пов'язані або подібні помилки, що дозволяє опрацьовувати помилки комплексно і витрачати менше часу або знаходити рішення, яке вже було створено раніше.

Таблиця містить наступні поля:

- id – номер запису;
- predecessor_id – помилка, до якої відноситься дана помилка;
- bug_id – помилка.

Таблиця ProjectLanguage підтримує можливість створення проєктів, написаних різними мовами програмування. Кожна мова програмування може мати різну кількість проєктів, кожен проєкт може бути написаний різними мовами програмування, тому знадобилася дана таблиця для реалізації зв'язку багато-до-багатьох.

Таблиця містить наступні поля:

- id – номер запису;
- project_id – проєкт;
- language_id – мова програмування, якою написано проєкт.

Таблиця ProjectTechnology підтримує можливість створення проєктів, використовуючи різні технології. Кожна технологія може мати різну кількість проєктів, кожен проєкт може бути створений на основі різних технологій, тому знадобилася дана таблиця для реалізації зв'язку багато-до-багатьох.

Таблиця містить наступні поля:

- id – номер запису;
- project_id – проєкт;
- technology_id – технологія, використовуючи яку написано проєкт.

Створена в результаті схема бази даних представлена на рис. 2.2.

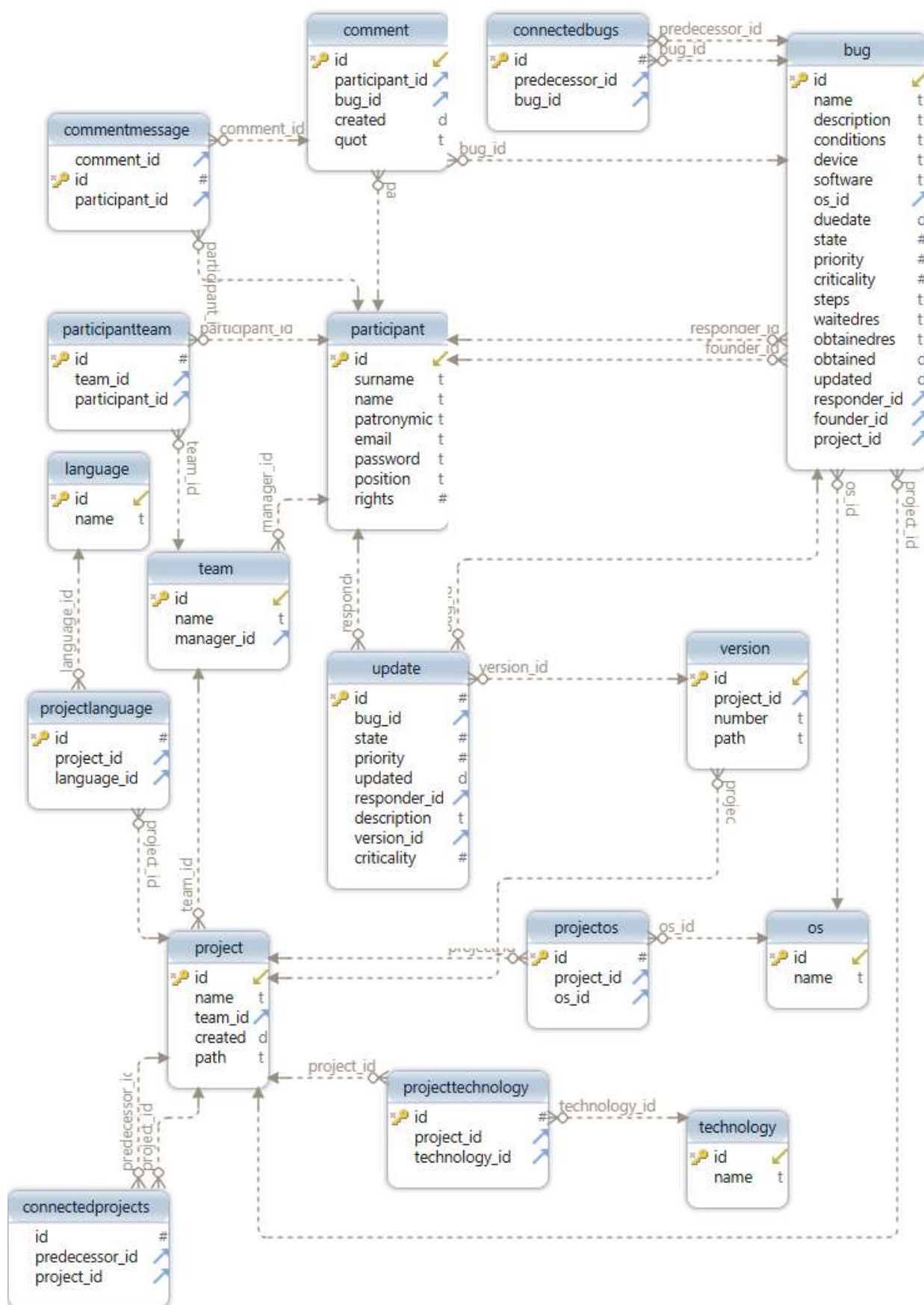


Рисунок 2.2 – Схема бази даних

Таблиця ConnectedProjects визначає пов'язані проєкти (підпроєкти, версії для різних операційних систем).

Таблиця містить наступні поля:

- id – номер запису;
- predecessor_id – пов'язаний з даним проєкт;
- project_id – проєкт.

2.4 Визначення архітектури програмного забезпечення

Архітектуру програмного забезпечення буде побудовано на основі використання шаблону проєктування Model-View-Controller (MVC). Використання даного шаблону (рис. 2.3) має з одного боку дозволити структурувати проєкт, а з іншого боку, використовуючи цю структурування, досягти легкості подальшої підтримки, адже всі частини програмного забезпечення будуть розділені за тією відповідальністю, яка їм призначена, відповідно знайти ділянку, де виникла проблема, застосувати засоби рефакторингу коду, а за необхідності і доповнити проєкт іншими функціями (що актуально для розробки на дипломну кваліфікаційну роботу, адже в даному випадку проєкт створюється за обмежень у часі та кількості розробників), стає простіше. Окрім того за використання шаблону простіше підтримувати проєкт команді розробників.

Основна ідея MVC полягає в тому, що кожен компонент проєкту має своє призначення, і ці призначення різні, частина коду містить дані програми, частина коду робить застосунок таким, яким він виглядає перед користувачем, а частина коду контролює, як функціонує програма, загалом MVC – це спосіб впорядкувати основні функції коду у власні, акуратно організовані компоненти, компонентами MVC є:

- модель: Код моделі зазвичай відображає реальні речі, цей код може містити необроблені дані або він визначатиме основні компоненти застосунку;

– представлення: код представлення складається з усіх функцій, які безпосередньо взаємодіють з користувачем, це код, завдяки якому застосунок виглядає так, як виглядає, він визначає, як ваш користувач бачить і взаємодіє з застосунком;

– контролер: код контролера діє як зв'язок між моделлю та представленням, отримуючи введення користувача та вирішуючи, що з ними робити, це частина, що пов'язує модель і представлення [15].

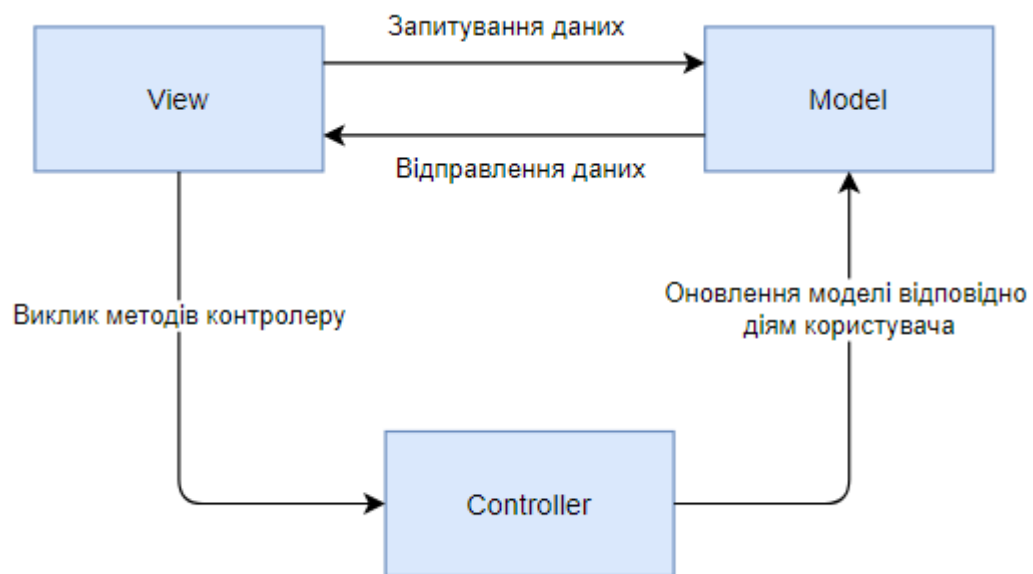


Рисунок 2.3 – Шаблон проєктування MVC

2.5 Висновки за розділом 2

У даному розділі виконано проєктування програмного забезпечення, охоплюючи вибір технології програмування для реалізації програмного забезпечення, в якості якої було обрано мову PHP, визначення функціональних вимог до програми, проєктування бази даних, визначення архітектури застосунку на основі шаблону проєктування MVC.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис логіки роботи програми

Логіка роботи програми побудована на принципі розподілу відповідальності між учасниками проєкту. Для кожного учасника проєкту створено свій варіант інтерфейсу, який контролює логіку роботи з програмою. Єдиним у взаємодії всіх учасників проєкту з програмою є робота над помилками. Стани помилки з'єднують логіку роботи учасників різного типу в єдине ціле, адже кожен з них має певні інструменти, які по суті змінюють той чи інший стан об'єкта помилки.

Спочатку користувач як кінцевий споживач або тестувальник повідомляє про помилку, в результаті чого розпочинається життєвий цикл помилки, яка спочатку знаходиться в стані «зафіксована». У цьому стані помилка знаходиться до того моменту, поки менеджер не виконає розподіл між розробниками, що є учасниками команди, яка працює над проєктом. Після цього помилка виявиться в стані «призначена».

Працюючи над помилкою, розробник, якому вона призначена, може виявити, що йому не вдалося відтворити помилку. У такому випадку він змінює її стан на «не відтворена». У цьому стані помилка буде знаходитися до того моменту, поки тестувальник, користувач або менеджер не уточнять її деталі. Якщо це вдалося, то помилка знову стає «призначеною» і відображається для роботи розробника.

Як тільки розробнику або команді розробників вдасться вирішити помилку, вона буде знаходитися в стані «вирішена».

Тоді розпочинається робота тестувальника і його зона відповідальності. Якщо тестувальник за результатами тестування підтвердив розв'язання проблеми, то помилка переходить до стану «протестована». Якщо проблему було виявлено знову, то помилка переходить до стану «призначена», і розробник знову працює над її усуненням.

Якщо помилка протестована і менеджер підтвердив отримання успішного результату, то помилка переходить в стан «закрита».

Результуюча діаграма станів помилки представлена на рис. 3.1.

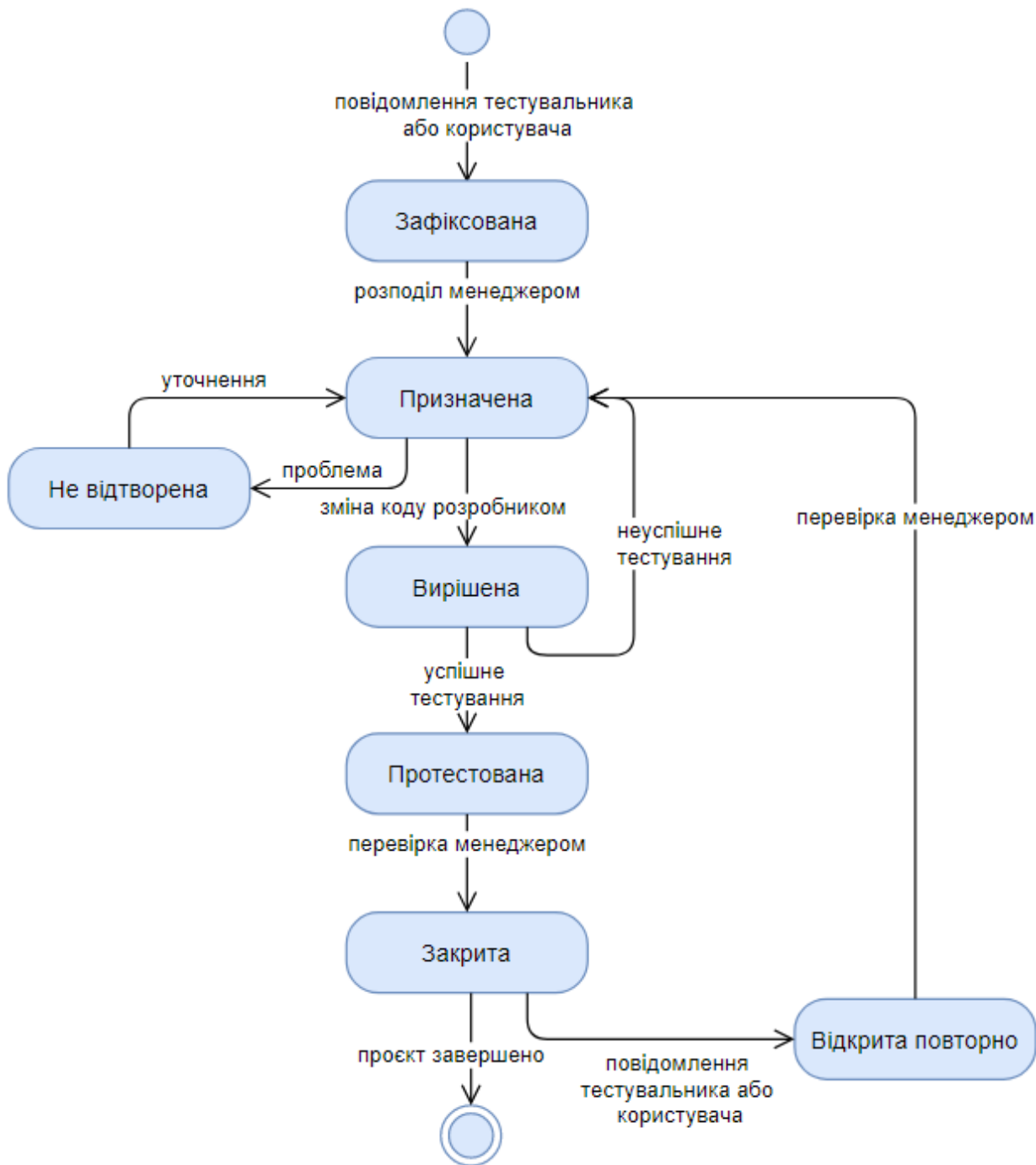


Рисунок 3.1 – Діаграма станів помилки

Якщо після закриття в певний момент проєкт завершує своє існування, то на цьому життєвий цикл помилки завершується.

До цього моменту тестувальник або користувач можуть виявити повторно, тоді вона відкривається заново і переходить до стану «відкрита повторно».

Якщо менеджер підтвердив це, то вона знову виявляється в стані «призначена», і над нею може працювати розробник.

3.2 Опис логіки реалізації

У відповідності з шаблоном проєктування MVC увесь проєкт було розбито на складові (рис. 3.2), що охоплюють:

- моделі застосунку (папка Model);
- контролери застосунку (папка Controller);
- представлення застосунку (папка View);
- налаштування асоціацій між адресами вебзастосунку (папка Config, скрипт Routes.php);
- створення підключення до бази даних та загальних засобів роботи з базою даних (папка Helper, скрипт Dbase.php);
- ресурси застосунку (папка Resources), де знаходяться каскадні таблиці стилів, зображення, скрипти javascript.

Скрипти javascript з папки Resources охоплюють засоби, необхідні для організації даних у відповідності з заданими параметрами, зокрема таким чином організовано відображення помилок під час пошуку за назвою, відображення помилок за обраним станом. Тобто це дозволяє не звертатися повторно до сервера з запитом на завантаження даних заново, а продовжувати працювати з отриманим набором, відфільтровуючи його потрібним чином. Це рішення щодо реалізації знижує в результаті навантаження на сервер.

На рис. 3.2 приведені основні складові розробленого проєкту, зокрема кількість розроблених представлень більша, ніж винесено на схему.

Для створення представлень вебзастосунку було використано вільний HTML-шаблон [16].

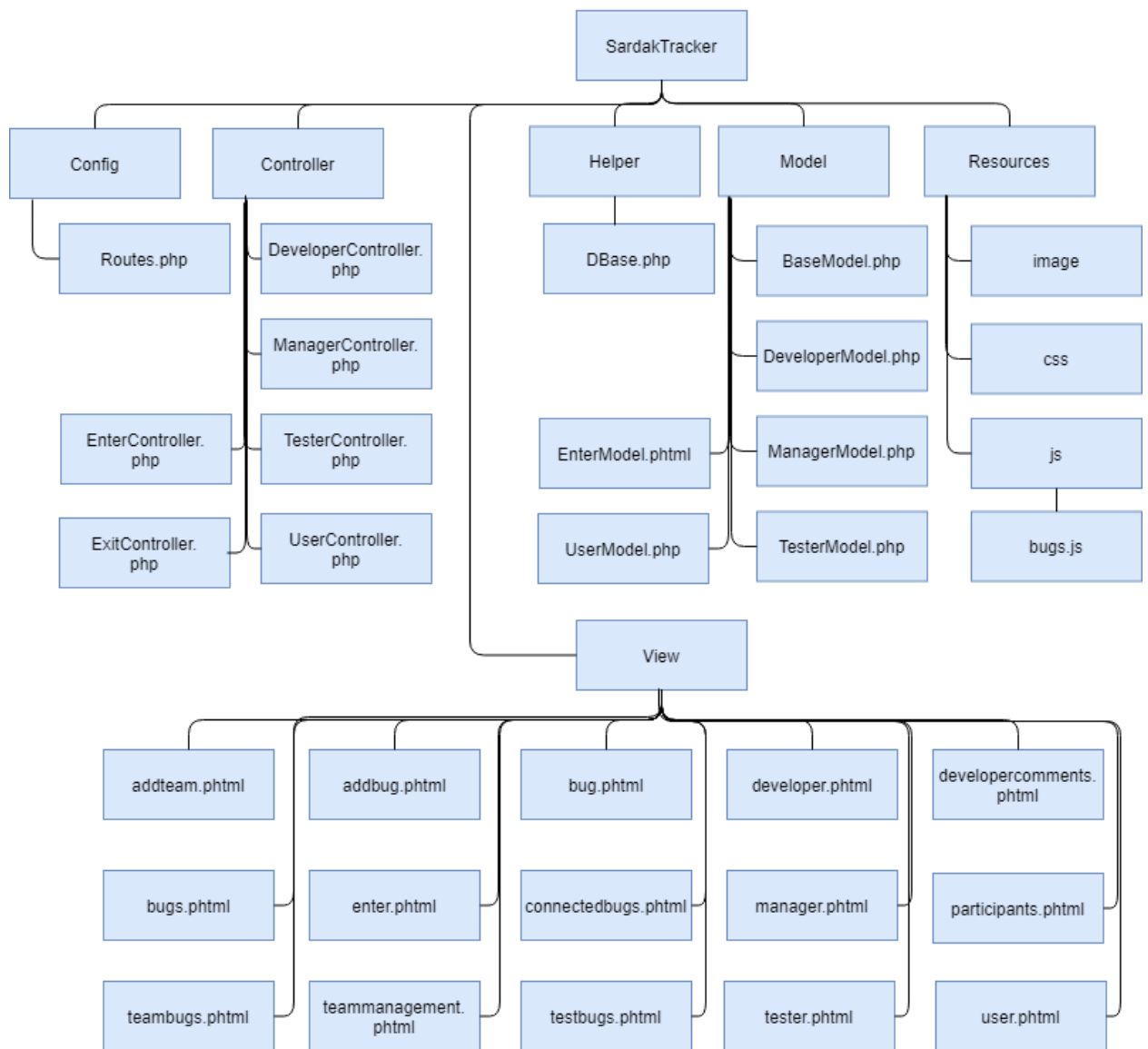


Рисунок 3.2 – Структурна схема програми

Представлення зокрема охоплюють:

- addteam.phtml – вебсторінка створення команди розробників;
- addbug.phtml – вебсторінка внесення помилки;
- bug.phtml – вебсторінка перегляду повної інформації про помилку;
- developer.phtml – основна вебсторінка розробника;
- developercomments.phtml – вебсторінка перегляду призначених розробнику коментарів;

- bugs.phtml – вебсторінка перегляду всіх помилок;
- enter.phtml – вебсторінка авторизації користувача;
- connected.phtml – вебсторінка перегляду помилок, пов'язаних з даною;
- manager.phtml – основна вебсторінка менеджера;
- participants.phtml – вебсторінка перегляду учасників, що надає менеджеру засоби контролю за роботою з відслідковування помилок у команді;
- teambugs.phtml – вебсторінка перегляду помилок, що віднесені до проєктів команд, в яких бере участь авторизований користувач;
- teammanagement.phtml – вебсторінка керування складом команди;
- testbugs.phtml – вебсторінка перегляду помилок, які потрібно відтестувати;
- tester.phtml – основна вебсторінка тестувальника;
- user.phtml – основна вебсторінка звичайного користувача;
- manheader.phtml – заголовна частина вебсторінки, що підключається до сторінок, з якими працює менеджер, в результаті менеджер може переходити тільки між тими сторінками, що належать до функціональних засобів його роботи;
- devheader.phtml – заголовна частина вебсторінки, що підключається до сторінок, з якими працює розробник, в результаті розробник може переходити тільки між тими сторінками, що належать до функціональних засобів його роботи, аналогічно створені розширення для сторінок інших користувачів.

Представлення виводять отримані дані у відповідних позиціях розмітки, формуючи таким чином вебсторінку, що відображається користувачу.

Моделі застосунку:

– BaseModel – базова модель, що містить засоби роботи з даними, які використовуються усіма типами учасників або є засобами, які стосуються даних загалом;

- DeveloperModel – модель для підтримки роботи розробника;
- ManagerModel – модель для підтримки роботи менеджера;
- TesterModel – модель для підтримки роботи тестувальника;
- UserModel – модель для підтримки роботи користувача;
- EnterModel – модель для підтримки авторизації учасника.

Клас UserModel створений шляхом наслідування класу BaseModel і визначає наступні методи своєї роботи.

Метод createBug(\$bug) виконує підтримку внесення помилки, що характеризується інформацією параметра bug, в базу даних. Для цього спочатку вносяться потрібні значення, які не задаються користувачем. Наприклад, станом помилки стає стан з номером 0, тобто зафіксований, вноситься інформація про авторизованого користувача як про автора, а дані про операційну систему та проєкт, до якого належить дана помилка, перетворюються до ідентифікаційного номеру шляхом звертання до засобів як раз базового класу BaseModel:

```
$bug['state'] = 0;
$bug['founder_id'] = $_SESSION['user'];
$bug['os_id'] = $this->getOSId($bug['os_id']);
$bug['project_id'] = $this->getProjectId($bug['project_id']);
```

Після формування повних даних помилки далі реалізується внесення цих даних в базу даних, що є основною відповідальністю моделі [17]. Для цього реалізується встановлення з'єднання з базою даних через клас DBase і його метод createConn(), далі формується запит на внесення даних помилки в базу даних, а після того виконується даний запит:

```
$conn = DBase::createConn();
```

```
$bugsql = $conn->prepare('INSERT INTO Bug (name, description,
conditions, device, software, os_id, duedate, state, priority, criticality, steps,
waitedres, obtainedres, obtained, updated, founder_id, project_id)
```

```
VALUES (:name, :description, :conditions, :device, :software, :os_id,
:duedate, :state, :priority, :criticality, :steps, :waitedres, :obtainedres, :obtained,
:updated, :founder_id, :project_id)');
```

```
$bugsql->execute($bug);
```

Метод `getUserBugs()` повертає перелік помилок, які були визначені авторизованим користувачем. Для цього реалізується запит до бази даних на витягання даних про помилки, включаючи дані розробника, який відповідальний за дану помилку, та назву проєкту:

```
$bugs = $conn->prepare('SELECT Bug.id as bug_id, Bug.name as
bug_name, Bug.priority as bug_priority, Bug.criticality as bug_criticality,
Bug.obtained as bug_obtained, Bug.updated as bug_updated, Bug.state as
bug_state, Prt.id as participant_id, Prt.email as participant_login, P.name
```

```
FROM Bug
```

```
INNER JOIN Participant Prt on Bug.responder_id = Prt.id
```

```
INNER JOIN Project P on Bug.project_id = P.id
```

```
WHERE founder_id = :founder_id
```

```
ORDER BY Bug.id desc');
```

```
$bugs ->bindParam('founder_id', $_SESSION['user']);
```

```
$bugs ->execute();
```

Після цього виконується витягання всіх записів про відповідні помилки, а інформація про пріоритет помилки, рівень її критичності, стан перетворюється з числового вигляду на текстовий:

```

$bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
$bugs = $this->textPriority($bugs);
$bugs = $this->textCriticality($bugs);
$bugs = $this->textState($bugs);

```

Метод `editBug($bug_id, $newInfo)` забезпечує редагування інформації про помилку, при цьому не змінюється стан помилки, її пріоритет, критичність, а змінюється тільки загальна інформація, що необхідно для випадків, коли користувач невірно ввів дані помилки і бажає її виправити:

```

$bugsql = $conn->prepare('UPDATE Bug SET name = :name, description
= :description, conditions = :conditions, device = :device, software = :software,
os_id = :os_id, steps = :steps, waitedres = :waitedres, obtainedres = :obtainedres,
updated = :updated, project_id = :project_id');
$newbug = $bugsql ->execute($newInfo);
$this->changeBug($bug_id, $newInfo, $pos);

```

Метод `reopenBug($bug)` виконує внесення даних про зміну стану помилки `bug` на повторно відкритий.

Клас `TesterModel` створений шляхом наслідування класу `UserModel` і визначає наступні методи роботи тестувальника.

Метод `getToTestBugs()` повертає перелік помилок, які були створені авторизованим тестувальником і знаходяться у вирішеному стані, тобто проєкт перероблений розробником і потребує тестування.

Метод `createBug($bug)` виконує внесення даних про помилку `bug` в базу даних, додаючи аналогічно розглянутій вище моделі параметри за замовчуванням. Тестувальнику знадобилось перевизначити даний метод, оскільки він, на відміну від звичайного користувача, може також встановити самостійно критичність та пріоритет помилки.

Метод `reopenBug($bug)` забезпечує відкриття помилки повторно.

Метод `editBug($bug_id, $newInfo)` виконує редагування автором даних помилки.

Метод `getUnrecognizedBugs()` повертає перелік помилок, які не були відтворені розробником, через що не вдалося досягти успіху.

Метод `testedBug($bug)` визначає стан помилки `bug` як протестований.

Аналогічно організовані інші класи моделей.

Клас моделі `BaseModel` підтримує методи визначення ролі учасника за його обліковим записом, отримання всього переліку помилок (доступний даний перелік буде за рахунок контролерів тільки менеджерам, тестувальникам і розробникам), визначення текстових значень характеристик за числовим представленням, отримання переліків усіх коментарів за помилкою, отримання переліку пов'язаних помилок з даною тощо.

Варто зазначити, що контроль за всіма засобами моделей виконується контролерами, так звичайному користувачу не доступний перелік усіх помилок, які містяться в системі, тобто він не може звернутися до цих засобів, адже вони зокрема не підтримуються його контролером.

Контролери застосунку:

- `DeveloperController` – контролер, що виконує керування взаємодією розробника зі своїми сторінками;

- `ManagerController` – контролер, що виконує керування взаємодією менеджера зі своїми сторінками;

- `TesterController` – контролер, що виконує керування взаємодією тестувальника зі своїми сторінками;

- `UserController` – контролер, що виконує керування взаємодією користувача зі своїми сторінками;

- `EnterController` – контролер, що виконує керування взаємодією користувача зі сторінкою авторизації.

Контролер `UserController` підтримує визначення відповідної моделі, через яку відбувається наповнення представлення даними, шляхом створення конструктору, визначення методів заповнення даними, створення,

редагування помилки, відтворення переліків помилок, додавання коментарів, зв'язування помилок.

3.3 Висновки за розділом 3

Описано логіку роботи програми на основі взаємодії користувачів з програмою для роботи над об'єктами помилок, в результаті чого побудовано діаграму станів. Описано розроблені класи моделей, контролерів, включаючи їх структуру та логіку реалізації. Представлено структурну схему програми.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Розроблене програмне забезпечення призначене для підтримки роботи над усуненням помилок зі сторони команди учасників програмних проєктів. Програма дозволяє звичайним користувачам та тестувальникам повідомляти про виявлені помилки, тестувальникам – перевіряти стан проєктів, помилки яких усунені розробниками, розробникам – усувати помилки в коді проєкту, менеджерам – призначати відповідальних розробників за усунення помилок.

4.2 Умови виконання програми

Сервер має забезпечувати виконання наступних апаратних вимог:

- процесор з 4 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 8 Гб;
- 500 Мб доступного простору на жорсткому диску.

Обов'язковим є підключення до високошвидкісного інтернету (не менше ніж 100 Мбіт/с, бажано – 1 Гбіт/с).

4.3 Тестування сценаріїв роботи з програмою

Головне меню тестувальника включає наступні розділи:

- «Відслідковування», що призначено для перегляду даних усіх помилок, які були внесені в систему авторизованим у даний момент тестувальником;
- «Тестування», що призначено для роботи з помилками, які були оброблені розробником і вимагають перевірки, тобто тестування, при цьому відображаються всі помилки, які знаходяться в стані «вирішена»;

– «Усі помилки», що призначено для відображення всіх помилок, з якими працює компанія, варто звернути увагу на те, що вебзастосунок передбачає підтримку роботи однієї окремої компанії, тобто кожна компанія, яка бажає використовувати вебзастосунок має скористатися версією для себе;

– «За проєктами», що дозволяє переглядати помилки, але вибираючи при цьому з випадуючого переліку проєкт, до якого ці помилки належать;

– «Внести помилку», що дозволяє тестувальнику внести в систему інформацію про нову виявлену помилку, тобто розпочати життєвий цикл об'єкта помилки;

– «Вихід», що дозволяє змінити або просто вийти з облікового запису.

У режимі роботи «Тестування» тестувальник може виконувати безпосередньо свої обов'язки (рис. 4.1).

Програмне забезпечення відслідковування помилок: тестувальник

Тестування

Помилки, готові до тестування

Пошук:

Пріоритет: ▼

#	Назва	Пріоритет	Критичність	Створено	Оновлено	Проект	Розробник	Результат	Редагувати
43	Помилка оновлення пошти	Середній	Незначна	08.05.2021	11.05.2021	JustPlanner	vinnsk@developer.com	Успіх	Редагувати
41	Неможливість помилку за замовчуванням	Низький	Незначна	08.05.2021	10.05.2021	SmartApp	zotiv@developer.com	Успіх	Редагувати
38	Постійне збої запуску	Високий	Критична	08.05.2021	11.05.2021	JustPlanner	vinnsk@developer.com	Успіх	Редагувати
37	Кріш за відсутності інтернету	Високий	Значна	08.05.2021	10.05.2021	FinManager	zotiv@developer.com	Успіх	Редагувати
35	Неможливість відкрити де-факт	Середній	Значна	08.05.2021	10.05.2021	MovieCollector	vinnsk@developer.com	Успіх	Редагувати
34	Помилка не оновлюється триває	Високий	Критична	07.05.2021	10.05.2021	MusicRecommender	arsenjev@developer.com	Успіх	Редагувати
33	Некоректний фокус на пол	Низький	Незначна	07.05.2021	09.05.2021	MovieCollector	arsenjev@developer.com	Успіх	Редагувати
31	Втрата курсора на мить	Низький	Тривіальна	07.05.2021	09.05.2021	FinManager	zotiv@developer.com	Успіх	Редагувати

Copyright © 2021 НУ "Запорізька політехніка"

Рисунок 4.1 – Перегляд помилок, готових до тестування

Тестувальник може ввести в полі пошуку назву або частину назви помилки, щоб здійснити пошук. При цьому не потрібно виконувати введення

і натискати на кнопку, дані актуалізуються одразу. Тобто тестувальник вводить частину назви помилки, пошук якої він виконує, а таблиця одразу змінює свій вміст.

Тестувальник також може обрати відповідний, потрібний йому, рівень критичності помилок, що будуть відображені. Для цього потрібно з випадającego переліку «Критичність» над таблицею обрати потрібне значення.

За кожною помилкою в таблиці доступна наступна інформація:

- номер помилки, який потрібен зокрема для того, щоб виконати зв'язування помилок: для цього потрібно знати номер помилки;

- назва помилки, яка при натисканні на неї дозволяє перейти до сторінки перегляду детальної інформації про помилку, на цій же сторінці можна виконати зв'язування помилки з іншими помилками (звичайний користувач виконати цю дію не може);

- пріоритет помилки: визначає також колір рядка з помилкою (зелений для низького пріоритету, сірий – для середнього, червоний – для високого);

- критичність помилки;

- дата внесення інформації про помилку;

- дата оновлення інформації про помилку;

- проєкт, до якого належить помилка: на назву проєкту можна натиснути і перейти до переліку помилок проєкту;

- електронна пошта розробника, який відповідальний за усунення помилки.

Стовпчик «Результат» дозволяє вказати на результат проведеного тестування. Якщо помилку було усунено, то потрібно натиснути на кнопку «Успіх». Якщо помилка знову була виявлена, то потрібно натиснути на стрілку на кнопці і обрати з випадającego переліку пункт «Повернути», після чого натиснути на дану кнопку.

Якщо необхідно відредагувати дані помилки, наприклад, змінити її опис або інші характеристики (не включаючи стан), то потрібно натиснути на кнопку «Редагувати» в останньому стовпці таблиці.

У випадку відкриття розділу меню «Внести помилку», тестувальник починає працювати зі сторінкою, на якій розташовано форму внесення даних помилки (рис. 4.2).

Програмне забезпечення відслідковування помилок: тестувальник

Внести помилку

Внесення даних виявленої помилки

Назва помилки

Повний опис помилки

Очікуваний пріоритет: Середній

Очікувана критичність: Незначна

Проект: Обрати

Пристрій, на якому виявлено помилку

Очікуваний результат

Отриманий результат

Кроки для відтворення помилки

Операційна система

Програма, через яку відбувалось з'єднання (необов'язкове поле)

Рисунок 4.2 – Внесення даних нової помилки

Форма внесення даних помилки містить наступні поля:

- назва помилки;
- повний опис помилки, який може містити достатньо великий текст;
- очікуваний пріоритет, що дозволяє обрати значення з випадаючого переліку, при цьому значення буде встановлено по суті як рекомендоване, тобто безпосередньо розробнику, а значить і тестувальникам після виконання

роботи розробником, з даним рівнем пріоритету буде передано тільки тоді, коли це значення підтвердить або змінить менеджер;

- очікувана критичність, що дозволяє обрати значення з випадającego переліку, також вимагає підтвердження менеджером;

- проєкт (програма), при роботі з яким сталася помилка: потребує вибору з випадającego переліку;

- версія проєкту, в якій було встановлено помилку (якщо відомо);

- пристрій, на якому було виявлено посилку: потребує введення значення вручну;

- очікуваний результат: результат роботи програми, який сподівався отримати користувач або тестувальник;

- отриманий результат: результат роботи програми, який фактично отримав користувач або тестувальник;

- кроки для відтворення помилки: необхідні, щоб розробник потім зміг відтворити помилку і мати більше шансів досягти успіху;

- операційна система, при роботі з якою було виявлено помилку, містить випадający перелік;

- програма, через яку було виявлено помилку: в основному використовується для вебсайтів, передбачає визначення браузера в такому випадку;

- умови, за яких виявлено помилку (будь-яка додаткова інформація).

Рівні пріоритетів, які можна встановити помилці:

- високий;

- середній;

- низький.

Рівні критичності, які можна встановити помилці:

- блокуюча;

- критична;

- значна;

- незначна;

– тривіальна.

Рівень пріоритету може встановити тільки менеджер. Усі інші можуть тільки переглядати встановлений рівень пріоритету, тобто менеджер визначає порядок роботи розробників та тестувальників над помилками.

Звичайний користувач має подібну функціональність за логікою до тестувальника, але при цьому обмежену. Наприклад, він не може встановлювати критичність і пріоритет помилки. Окрім того йому доступна менша кількість розділів меню, тобто він має меншу кількість засобів перегляду помилок. До того ж він може працювати тільки з власними помилками.

Розділи меню звичайного користувача включають:

- «Відслідковування» для перегляду тільки тих помилок, які вніс авторизований користувач;
- «Внести помилку» для внесення даних нової помилки, при цьому не можна вказати пріоритет та критичність;
- «Вихід» для виходу з системи.

Під час перегляду власних помилок він зокрема не може переходити до всіх помилок проєкту (рис. 4.3), тому назва проєкту відображається без посилання. Користувач не зможе перейти і до всіх помилок в системі. Користувачу не доступна інформація і про розробника, який працює над усуненням помилки, тому що це закрита інформація для зовнішніх учасників проєкту.

Користувач може переглядати інформацію про стан помилок, які він виявив. Для цього він може скористатися пошуком, вибором стану помилки, якщо помилка, яку він вніс, достатньо багато.

У користувача є можливість відкрити повторно помилку, яку було успішно вирішено раніше. Окрім того він має можливість відредагувати описові дані помилки, щоб уточнити їх розробнику, при цьому не має на увазі повне редагування.

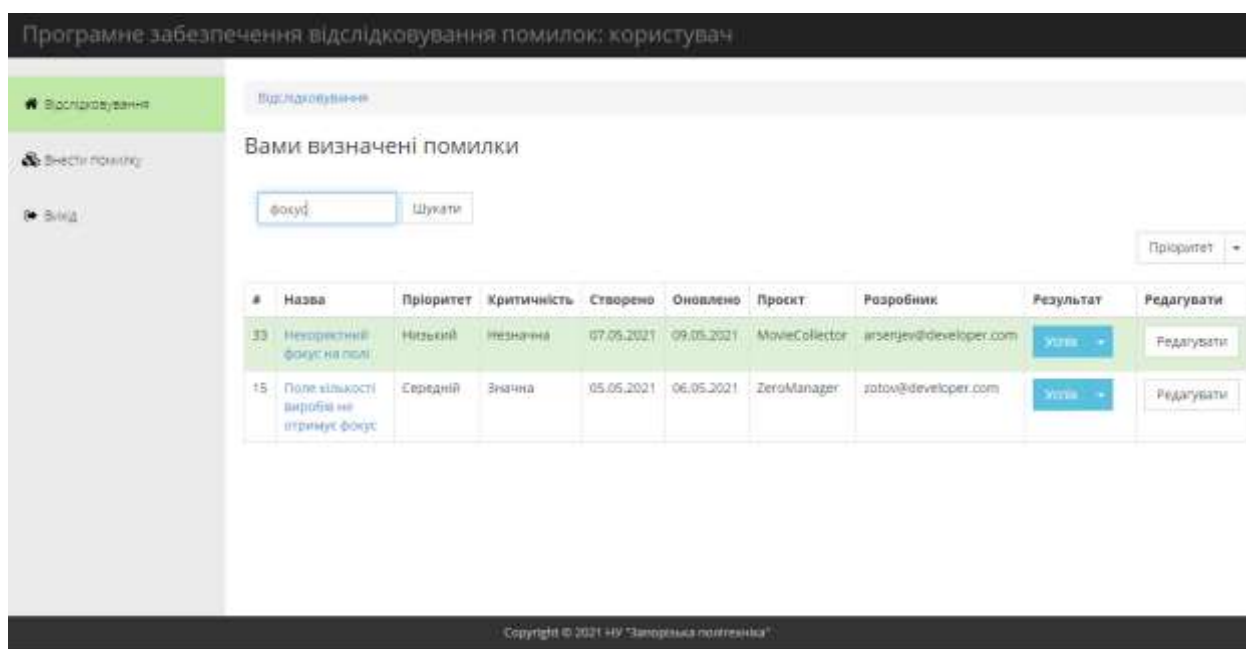


Рисунок 4.3 – Пошук помилок серед усіх внесених користувачем

Розробнику доступні наступні розділи меню:

- «Відслідковування», що дозволяє переглянути помилки, які були призначені для усунення авторизованому розробнику;
- «Усі помилки», що дозволяє переглянути перелік усіх наявних в системі помилок;
- «Командні помилки», що дозволяє переглянути помилки проєктів, над якими працюють командди, до складу яких входить розробник: даний варіант роботи підтримує командний режим роботи над проєктами, тобто будь-який учасник команди може працювати над помилками, але при цьому остаточно відповідальним за результат є розробник, якого призначив менеджер;
- «Адресовані коментарі», що дозволяє переглянути коментарі, які були тестувальники направлені особисто авторизованому користувачу (для цього під час створення коментаря щодо помилки потрібно обрати адресата, всі такі коментарі будуть відображені користувачу в системі, відсортовані за датою появи від поточного часу).

При перегляді помилок (рис. 4.4) розробник отримує детальну інформацію на окремій сторінці, що включає повні дані цієї помилки,

можливість переходу на сховища проєкту або відповідної версії проєкту, дані виконаних оновлень, перелік прив'язаних помилок, які вважаються схожими, коментарі.

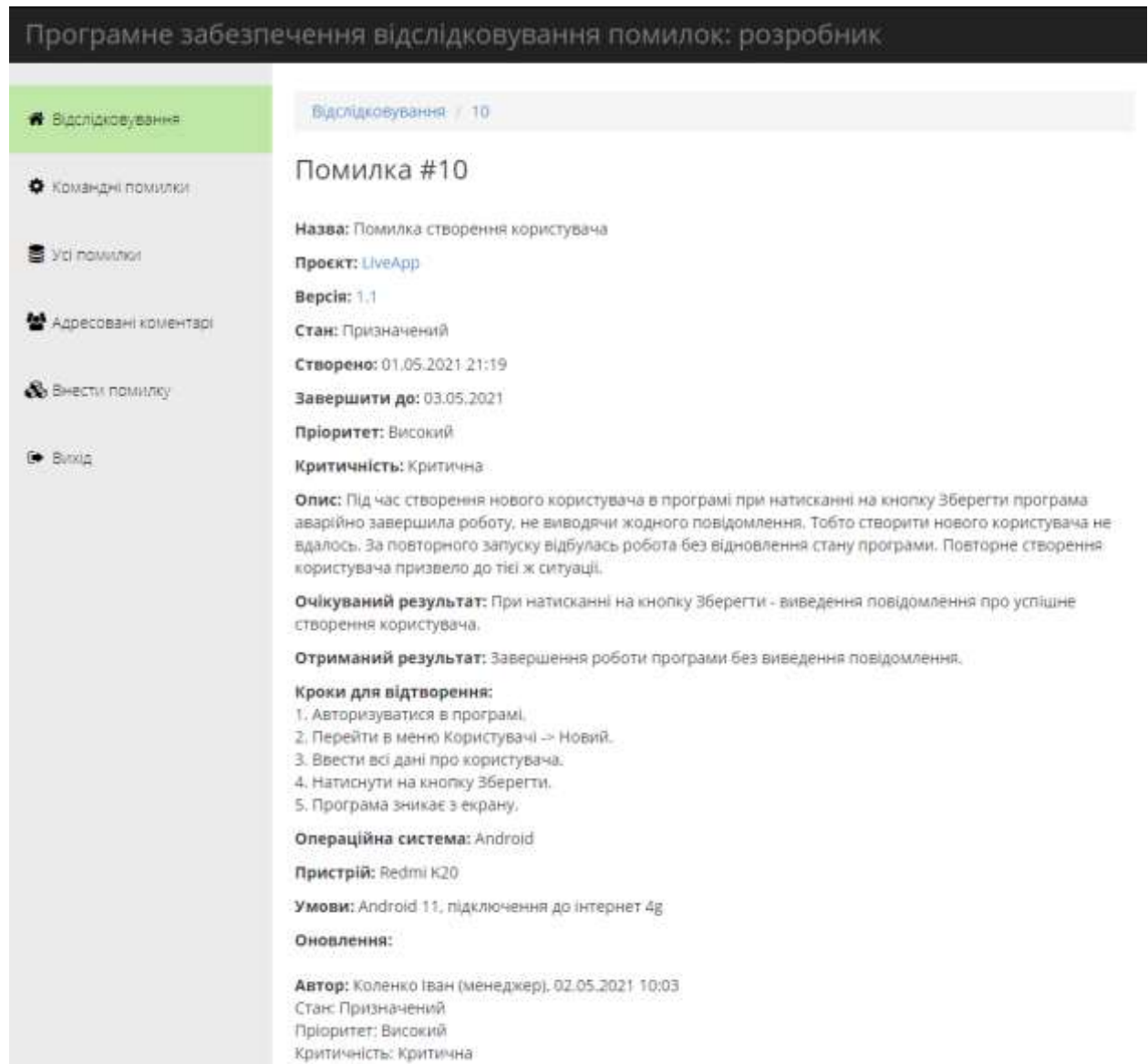


Рисунок 4.4 – Перегляд інформації про помилку розробником

Менеджеру доступні наступні розділи меню:

- «За проєктами», що дозволяє переглядати помилки за проєктами: спочатку відображається перелік проєктів, потрібно обрати потрібний, тоді відкриється сторінка зі всіма помилками, пов'язаними з даним проєктом;
- «Усі помилки», що дозволяє переглянути усі помилки, які було внесено в систему;

– «Проекти», що дозволяє керувати проектами, звідси можна додавати нові проекти, редагувати існуючі;

– «Команди», що дозволяє керувати комендами розробників, створюючи нові, редагуючи склад існуючих команд, пов'язувати їх з проектами (для цього відкривається сторінка зв'язаних з проектом помилок);

– «Розробники», що дозволяє отримувати інформацію про роботу розробників над помилками: відображається перелік розробників і за кожним можна перейти до сторінки перегляду його оновлень над всіма помилками.

Базовим є розділ «За проектами»: після вибору конкретного проекту відображається сторінка з помилками за ним, відсортованими за номером (рис. 4.5). Перейти до вибору проектів можна за допомогою верхнього меню через пункт «Проекти».

Програмне забезпечення відслідковування помилок: користувач

Відслідковування

Вам визначені помилки

фокус

Пріоритет ▼

#	Назва	Пріоритет	Критичність	Створено	Оновлено	Проект	Стан	Відновлення	Редагувати
33	Некоректний фокус на полі	Низький	Незначна	07.05.2021	09.05.2021	MovieCollector	Вирішена		Редагувати
15	Поле кількості виробів не отримує фокус	Середній	Значна	05.05.2021	06.05.2021	ZeroManager	Закрита		Редагувати

Copyright © 2021 НУ "Запорізька політехніка"

Рисунок 4.5 – Керування помилками менеджером

Для кожної помилки можна вибрати пріоритет та критичність для випадків, якщо помилку ще не закрито. Окрім того можна вибрати відповідального розробника з випадаючого переліку (для закритих проектів

також недоступно), з календаря обрати дату, до якої має бути завершено роботу над помилкою.

Для збереження змін, які були внесені щодо помилки в рядку таблиці, призначена кнопка «Зберегти». Кнопка «Прив'язати» дозволяє прив'язати обраний проєкт до іншого існуючого. Кнопка «Закрити» активна тільки для випадків, коли помилка знаходиться у відтестованому стані, тобто робочий цикл за нею завершено, тоді менеджер може її закрити. Кнопка «Редагувати» дозволяє перейти до повного редагування помилки, включаючи визначення тих параметрів, що відображаються в таблиці, але в більш розширеному вигляді, тобто поля зокрема займають в інтерфейсі більшу кількість місця.

4.4 Висновки за розділом 4

Описано сценарії роботи звичайного користувача, тестувальника, менеджера та розробника з програмою щодо реалізації різних етапів відслідковування помилок.

ВИСНОВКИ

У дипломній кваліфікаційній роботі бакалавра розглянуто проблему розроблення програмного забезпечення відслідковування помилок. Проаналізовано проблему відслідковування помилок, зокрема розглянуто передумови створення систем відслідковування помилок, актуальний стан проблеми, етапи відслідковування помилок, стани помилок, їх рівні критичності, вимоги до систем. Проаналізовано програмні аналоги програмного забезпечення, що розробляється, визначено основні їх функціональні можливості.

У роботі виконано проєктування програмного забезпечення, охоплюючи вибір технології програмування для реалізації програмного забезпечення, в якості якої було обрано мову PHP, визначення функціональних вимог до програми, проєктування бази даних, визначення архітектури застосунку на основі шаблону проєктування MVC, та програмну реалізацію класів моделей, контролерів і вебсторінок з динамічним вмістом для взаємодії з користувачем. У результаті виконання даних етапів роботи створено діаграми прецедентів і станів, схему бази даних і структурну схему програми.

Розроблене програмне забезпечення призначене для підтримки роботи над усуненням помилок зі сторони команди учасників програмних проєктів. Програма дозволяє звичайним користувачам та тестувальникам повідомляти про виявлені помилки, тестувальникам – перевіряти стан проєктів, помилки яких усунені розробниками, розробникам – усувати помилки в коді проєкту, менеджерам – призначати відповідальних розробників за усунення помилок. Інтерфейс роботи для кожного користувача відповідної ролі створений окремо для обмеження його можливостей у відповідності зі сценаріями його роботи.

Усі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Arsenault, C. Error Tracking – Top Suggestions and Tools [Electronic resource] / C. Arsenault. – Access mode : <https://www.keycdn.com/blog/error-tracking>
2. Harley, N. What to Look For in a Great Error Tracking Service [Electronic resource] / N. Harley. – Access mode : <https://raygun.com/blog/what-to-look-for-in-a-great-error-tracking-service/>
3. Bug Tracking System – What It Is and Why You Need It [Electronic resource]. – Access mode : <https://kissflow.com/case/bug-tracking-system/>
4. What is bug tracking? [Electronic resource]. – Access mode : <https://www.ibm.com/topics/bug-tracking>
5. Parker, R. Common Problems That Occur In A Bug Tracking Software [Electronic resource] / R. Parker. – Access mode : https://dev.to/ray_parker01/common-problems-that-occur-in-a-bug-tracking-software-233b
6. Online Project Management Software for Developers | Backlog [Electronic resource]. – Access mode : <https://backlog.com/>
7. Duffy, J. Backlog by Nulab Review | PCMag [Electronic resource] / J. Duffy. – Access mode : <https://www.pcmag.com/reviews/backlog-by-nulab>
8. Backlog's Issue Tracking and Project Management Features | Backlog [Electronic resource]. – Access mode : <https://backlog.com/features/>
9. Home Bugzilla [Electronic resource]. – Access mode : <https://www.bugzilla.org/>
10. Features Bugzilla [Electronic resource]. – Access mode : <https://www.bugzilla.org/features/>
11. Bug Tracking Software | Free Issue Tracker System – Zoho BugTracker [Electronic resource]. – Access mode : <https://www.zoho.com/bugtracker/>
12. Bugs View | Custom Views, Milestones, Track Bugs [Electronic

resource]. – Access mode : <https://www.zoho.com/bugtracker/bugs-view.html>

13. Nixon, R. Learning PHP, MySQL & JavaScript [Text] : With jQuery, CSS & HTML5 / R. Nixon. – Sebastopol : O'Reilly, 2018. – 832 p.

14. Tatroe, K. Programming PHP [Text] : Creating Dynamic Web Pages / KL. Tatroe, P. MacIntyre. – Sebastopol : O'Reilly, 2020. – 544 p.

15. MVC [Electronic resource] : Model, View, Controller | Codecademy. – Access mode : <https://www.codecademy.com/articles/mvc>

16. Free Template 415 Dashboard [Electronic resource]. – Access mode : <https://templatemo.com/tm-415-dashboard>

17. Ulman, L. PHP and MySQL for Dynamic Web Sites [Text] : Visual QuickPro Guide / L. Ulman. – Berkeley : Peachpit Press, 2017. – 704 p.

ДОДАТОК А
Технічне завдання

Вступ

Відслідковування помилок є важливим супутнім процесом розроблення програмного забезпечення. У ньому беруть ту чи іншу участь тестувальники, розробники, менеджери. Програмне забезпечення відслідковування помилок має стати ефективним засобом їх комунікації щодо розв’язання проблеми усунення помилок у будь-який момент роботи над проєктом, виявлення залежностей між помилками.

A.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Програмне забезпечення відслідковування помилок» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для підтримки роботи над усуненням помилок зі сторони команди учасників програмних проєктів. Програма дозволяє звичайним користувачам та тестувальникам повідомляти про виявлені помилки, тестувальникам – перевіряти стан проєктів, помилки яких усунені розробниками, розробникам – усувати помилки в коді проєкту, менеджерам – призначати відповідальних розробників за усунення помилок.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має надавати такі функціональні можливості:

- авторизація;

- реєстрація;
- внесення нової помилки;
- перегляд усіх помилок, внесених користувачем;
- перегляд усіх внесених помилок;
- перегляд помилок за проєктом;
- перегляд інформації про помилку;
- перегляд пов'язаних помилок з даною;
- встановлення пріоритету помилки;
- встановлення критичності помилки;
- перегляд помилок за заданим станом;
- пошук помилок за назвою;
- відображення помилок, за які відповідальний розробник;
- призначення розробника, відповідального за усунення помилки;
- визначення строків усунення помилки;
- перегляд помилок, не призначених розробникам;
- перегляд помилок, готових до тестування;
- перегляд історії оновлення помилки;
- створення коментарів до помилки;
- перегляд коментарів до помилки;
- перегляд оновлень помилок розробником;
- внесення і редагування проєктів;
- внесення версій проєктів;
- створення команд розробників;
- зв'язування помилки з існуючою помилкою;
- перегляд адресованих коментарів;
- повторне відкриття помилки;
- визначення успішного усунення помилки;
- визначення успішної перевірки помилки;
- закриття помилки;
- визначення неможливості відтворення помилки.

А.3.2 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп'ютером не висувається.

Для забезпечення роботи бази даних на сервері необхідні вміння системного адміністратора для підтримки роботи вебсервера сервера системи керування базами даних MySQL.

А.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби серверної сторони повинні включати щонайменше:

- процесор з 4 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 8 Гб;
- 500 Мб доступного простору на жорсткому диску.

А.4 Порядок контролю та приймання

Принципи контролю виконання роботи визначаються календарним планом, розташованим у завданні на дипломну кваліфікаційну роботу, де вказано назви етапів, строки їх виконання та результат.

ДОДАТОК Б
Текст програми

Б.1 Текст файла UserModel.php

```

<?php

class UserModel extends BaseModel
{
    public function createBug($bug)
    {
        $bug['state'] = 0;
        $bug['founder_id'] = $_SESSION['user'];
        $bug['os_id'] = $this->getOSId($bug['os_id']);
        $bug['project_id'] = $this->getProjectId($bug['project_id']);
        $bug['priority'] = 0;
        $bug['criticality'] = 0;
        $today = date('Y-m-d H:i:s');
        $bug['obtained'] = $today;
        $bug['updated'] = $today;
        unset($bug['submit']);

        $conn = DBase::createConn();
        $bugsql = $conn->prepare('INSERT INTO Bug (name,
description, conditions, device, software, os_id, duedate, state,
priority, criticality, steps, waitedres, obtainedres, obtained,
updated, founder_id, project_id)
VALUES (:name, :description, :conditions, :device,
:software, :os_id, :duedate, :state, :priority, :criticality, :steps,
:waitedres, :obtainedres, :obtained, :updated, :founder_id,
:project_id)');
        $bugsql->execute($bug);
        $bug_id = $conn->lastInsertId();

        $updbug['bug_id'] = $bug_id;
        $updbug['state'] = 0;
        $updbug['priority'] = 0;
        $updbug['criticality'] = 0;
        $today = date('Y-m-d H:i:s');
        $updbug['updated'] = $today;
        $updbug['responder_id'] = $_SESSION['user'];
        $updbug['description'] = '';
    }
}

```

```

        $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

        $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
                VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
        return $bugsql->execute($updbug);

    }

    public function getUserBugs()
    {
        $conn = DBase::createConn();
        $bugs = $conn->prepare('SELECT Bug.id as bug_id,
Bug.name as bug_name,
Bug.priority as bug_priority,
Bug.criticality as bug_criticality,
Bug.obtained as bug_obtained,
Bug.updated as bug_updated,
Bug.state as bug_state,
Prt.id as participant_id,
Prt.email as participant_login,
P.name
        FROM Bug
        INNER JOIN Participant Prt on Bug.responder_id =
Prt.id

        INNER JOIN Project P on Bug.project_id = P.id
        WHERE founder_id = :founder_id
        ORDER BY Bug.id desc');
        $bugs ->bindParam('founder_id', $_SESSION['user']);
        $bugs ->execute();
        $bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
        $bugs = $this->textPriority($bugs);
        $bugs = $this->textCriticality($bugs);
        $bugs = $this->textState($bugs);
        return $bugs;
    }

    public function editBug($bug_id, $newInfo)
    {
        unset($newInfo['submit']);

```

```

        $newInfo['os_id'] = $this->getOSId($bug['os_id']);
        $newInfo['project_id'] = $this-
>getProjectId($newInfo['project_id']);
        $today = date('Y-m-d H:i:s');
        $newInfo ['updated'] = $today;
        $pos = $this->isNew($newInfo, $bug_id);
        if(!$pos)
            return true;

        $conn = DBase::createConn();
        $bugsql = $conn->prepare('UPDATE Bug SET name = :name,
description = :description, conditions = :conditions, device =
:device, software = :software, os_id = :os_id, steps = :steps,
waitedres = :waitedres, obtainedres = :obtainedres, updated =
:updated, project_id = :project_id');
        $newbug = $bugsql ->execute($newInfo);
        $this->changeBug($bug_id, $newInfo, $pos);
        return $newbug;
    }

    public function isNew($newInfo, $bug_id)
    {
        $pos = [];
        $conn = DBase::createConn();
        $bugsql = $conn->prepare("SELECT name, description,
conditions, device, software, os_id, steps, waitedres, obtainedres,
project_id FROM Bug WHERE id = :id");
        $bugsql->bindParam(':id', $bug_id);
        $bugsql->execute();
        $bug = $data->fetch();

        foreach ($bug as $key => $b) {
            if($b != $newInfo [$key]) {
                $pos[$key] = $key;
            }
        }
        if(count($pos)) {
            return $pos;
        }
        return false;
    }

    public function reopenBug($bug)

```

```

{
    $bug['state'] = 6;
    $today = date('Y-m-d H:i:s');
    $bug['updated'] = $today;
    unset($bug['submit']);

    $conn = DBase::createConn();
    $bugsql = $conn->prepare('UPDATE Bug SET description =
:description, conditions = :conditions, device = :device, software =
:software, steps = :steps, waitedres = :waitedres, obtainedres =
:obtainedres, updated = :updated, state = :state WHERE id = :id');
    $bugsql->execute($bug);

    $updbug['bug_id'] = $bug_id;
    $updbug['state'] = 6;
    $today = date('Y-m-d H:i:s');
    $updbug['updated'] = $today;
    $updbug['responder_id'] = $_SESSION['user'];
    $updbug['description'] = '';
    $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

    $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, updated, responder_id, description, version_id)
VALUES (:bug_id, :state, :priority, :updated,
:responder_id, :description, :version_id)');
    return $bugsql->execute($updbug);

}

}

```

Б.2 Текст файла TesterModel.php

```

<?php

class TesterModel extends UserModel
{
    public function getToTestBugs()
    {
        $conn = DBase::createConn();

```

```

        $bugs = $conn->prepare('SELECT Bug.id as bug_id,
        Bug.name as bug_name,
        Bug.priority as bug_priority,
        Bug.criticality as bug_criticality,
        Bug.obtained as bug_obtained,
        Bug.updated as bug_updated,
        Bug.state as bug_state,
        Prt.id as participant_id,
        Prt.email as participant_login,
        P.name
        FROM Bug
        INNER JOIN Participant Prt on Bug.responder_id =
Prt.id

        INNER JOIN Project P on Bug.project_id = P.id
        WHERE founder_id = :founder_id AND bug_state = 3
        ORDER BY Bug.id desc');
$bugs ->bindParam('founder_id', $_SESSION['user']);
$bugs ->execute();
$bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
$bugs = $this->textPriority($bugs);
$bugs = $this->textCriticality($bugs);
$bugs = $this->textState($bugs);
return $bugs;
}

public function createBug($bug)
{
    $bug['state'] = 0;
    $bug['founder_id'] = $_SESSION['user'];
    $bug['os_id'] = $this->getOSId($bug['os_id']);
    $bug['project_id'] = $this->
>getProjectId($bug['project_id']);
    $bug['priority'] = $this->getPriority($bug['priority']);
    $bug['criticality'] = $this->
>getCriticality($bug['criticality']);
    $today = date('Y-m-d H:i:s');
    $bug['obtained'] = $today;
    $bug['updated'] = $today;
    unset($bug['submit']);

    $conn = DBase::createConn();
    $bugsql = $conn->prepare('INSERT INTO Bug (name,
description, conditions, device, software, os_id, duedate, state,

```

```

priority, criticality, steps, waitedres, obtainedres, obtained,
updated, founder_id, project_id)
    VALUES (:name, :description, :conditions, :device,
:software, :os_id, :duedate, :state, :priority, :criticality, :steps,
:waitedres, :obtainedres, :obtained, :updated, :founder_id,
:project_id)');
    $bugsql->execute($bug);
    $bug_id = $conn->lastInsertId();

    $updbug['bug_id'] = $bug_id;
    $updbug['state'] = 0;
    $updbug['priority'] = $bug['priority'];
    $updbug['criticality'] = $bug['criticality'];
    $today = date('Y-m-d H:i:s');
    $updbug['updated'] = $today;
    $updbug['responder_id'] = $_SESSION['user'];
    $updbug['description'] = '';
    $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

    $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
        VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
    return $bugsql->execute($updbug);
}

public function reopenBug($bug)
{
    unset($bug ['submit']);
    $bug['state'] = 6;
    $bug['criticality'] = $this-
>getCriticality($bug['criticality']);
    $today = date('Y-m-d H:i:s');
    $bug['updated'] = $today;
    unset($bug['submit']);

    $conn = DBase::createConn();
    $bugsql = $conn->prepare('UPDATE Bug SET description =
:description, conditions = :conditions, device = :device, software =
:software, steps = :steps, waitedres = :waitedres, obtainedres =

```

```

:obtainedres, updated = :updated, state = :state, criticality =
:criticality WHERE id = :id ');
    $bugsql->execute($bug);

    $updbug['bug_id'] = $bug['id'];
    $updbug['state'] = 6;
    $updbug['priority'] = $bug['priority'];
    $updbug['criticality'] = $bug['criticality'];
    $today = date('Y-m-d H:i:s');
    $updbug['updated'] = $today;
    $updbug['responder_id'] = $_SESSION['user'];
    $updbug['description'] = $bug['why'];
    $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

    $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
        VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
    return $bugsql->execute($updbug);

}

public function editBug($bug_id, $newInfo)
{
    unset($newInfo ['submit']);
    $newInfo['os_id'] = $this->getOSId($newInfo['os_id']);
    $newInfo['project_id'] = $this->getProjectId($newInfo
['project_id']);
    $newInfo['criticality'] = $this-
>getCriticality($newInfo['criticality']);
    $today = date('Y-m-d H:i:s');
    $newInfo ['updated'] = $today;
    $pos = $this->isNew($newInfo, $bug_id);
    if(!$pos)
        return true;
    $conn = DBase::createConn();
    $bugsql = $conn->prepare('UPDATE Bug SET name = :name,
description = :description, conditions = :conditions, device =
:device, software = :software, os_id = :os_id, steps = :steps,
waitedres = :waitedres, obtainedres = :obtainedres, updated =
:updated, project_id = :project_id, criticality = :criticality');

```

```

        $newbug = $bugsql->execute($newInfo);
        $this->changeBug($bug_id, $newInfo, $pos);
        return $newbug;
    }

    public function getUnrecognizedBugs()
    {
        $conn = DBase::createConn();
        $bugs = $conn->prepare('SELECT Bug.id as bug_id,
        Bug.name as bug_name,
        Bug.priority as bug_priority,
        Bug.criticality as bug_criticality,
        Bug.obtained as bug_obtained,
        Bug.updated as bug_updated,
        Bug.state as bug_state,
        Prt.id as participant_id,
        Prt.email as participant_login,
        P.name
        FROM Bug
        INNER JOIN Participant Prt on Bug.responder_id =
Prt.id
        INNER JOIN Project P on Bug.project_id = P.id
        WHERE founder_id = :founder_id AND bug_state = 2
        ORDER BY Bug.id desc');
        $bugs ->bindParam('founder_id', $_SESSION['user']);
        $bugs ->execute();
        $bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
        $bugs = $this->textPriority($bugs);
        $bugs = $this->textCriticality($bugs);
        $bugs = $this->textState($bugs);
        return $bugs;
    }

    public function testedBug($bug)
    {
        $bug['state'] = 4;
        $today = date('Y-m-d H:i:s');
        $bug['updated'] = $today;
        unset($bug['submit']);

        $conn = DBase::createConn();
        $bugsql = $conn->prepare('UPDATE Bug SET updated =
:updated, state = :state');

```

```

$bugsql->execute($bug);

$updbug['bug_id'] = $bug['id'];
$updbug['state'] = 4;
$updbug['priority'] = $bug['priority'];
$updbug['criticality'] = $bug['criticality'];
$updbug['updated'] = $today;
$updbug['responder_id'] = $_SESSION['user'];
$updbug['description'] = $bug['why'];
$updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

$bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
return $bugsql->execute($updbug);

}

public function returnBug($bug)
{
    $bug['state'] = 1;
    $today = date('Y-m-d H:i:s');
    $bug['updated'] = $today;
    unset($bug['submit']);

    $conn = DBase::createConn();
    $bugsql = $conn->prepare('UPDATE Bug SET updated =
:updated, state = :state');
    $bugsql->execute($bug);

    $updbug['bug_id'] = $bug['id'];
    $updbug['state'] = 1;
    $updbug['priority'] = $bug['priority'];
    $updbug['criticality'] = $bug['criticality'];
    $updbug['updated'] = $today;
    $updbug['responder_id'] = $_SESSION['user'];
    $updbug['description'] = $bug['why'];
    $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

```

```

        $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
        VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
        return $bugsql->execute($updbug);

    }

}

```

Б.3 Текст файла DeveloperModel.php

```

<?php

class DeveloperModel extends BaseModel
{
    public function solvedBug($bug)
    {
        $bug['state'] = 3;
        $today = date('Y-m-d H:i:s');
        $bug['updated'] = $today;
        unset($bug['submit']);

        $conn = DBase::createConn();
        $bugsql = $conn->prepare('UPDATE Bug SET updated =
:updated, state = :state');
        $bugsql->execute($bug);

        $updbug['bug_id'] = $bug['id'];
        $updbug['state'] = 3;
        $updbug['priority'] = $bug['priority'];
        $updbug['criticality'] = $bug['criticality'];
        $updbug['updated'] = $today;
        $updbug['responder_id'] = $_SESSION['user'];
        $updbug['description'] = $bug['why'];
        $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);
    }
}

```

```

        $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)

```

```

        VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
        return $bugsql->execute($updbug);

```

```

    }

```

```

public function getDeveloperBugs()

```

```

{

```

```

    $conn = DBase::createConn();

```

```

    $bugs = $conn->prepare('SELECT Bug.id as bug_id,
Bug.name as bug_name,

```

```

Bug.priority as bug_priority,

```

```

Bug.criticality as bug_criticality,
Bug.obtained as bug_obtained,

```

```

Bug.updated as bug_updated,

```

```

Bug.state as bug_state,

```

```

Prt.id as participant_id,

```

```

Prt.email as participant_login,

```

```

P.name

```

```

    FROM Bug

```

```

    INNER JOIN Participant Prt on Bug.responder_id =

```

```

Prt.id

```

```

    INNER JOIN Project P on Bug.project_id = P.id

```

```

    WHERE responder_id = :responder_id

```

```

    ORDER BY Bug.id desc');

```

```

    $bugs ->bindParam('responder_id', $_SESSION['user']);

```

```

    $bugs ->execute();

```

```

    $bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);

```

```

    $bugs = $this->textPriority($bugs);

```

```

    $bugs = $this->textCriticality($bugs);

```

```

    $bugs = $this->textState($bugs);

```

```

    return $bugs;

```

```

}

```

```

public function problemBug($bug)

```

```

{

```

```

    $bug['state'] = 2;

```

```

    $today = date('Y-m-d H:i:s');

```

```

    $bug['updated'] = $today;

```

```

    unset($bug['submit']);

```

```

        $conn = DBase::createConn();
        $bugsql = $conn->prepare('UPDATE Bug SET updated =
:updated, state = :state');
        $bugsql->execute($bug);

        $updbug['bug_id'] = $bug['id'];
        $updbug['state'] = 2;
        $updbug['priority'] = $bug['priority'];
        $updbug['criticality'] = $bug['criticality'];
        $updbug['updated'] = $today;
        $updbug['responder_id'] = $_SESSION['user'];
        $updbug['description'] = $bug['why'];
        $updbug['version_id'] = this-
>getProjectVersionId($bug['project_id'], $bug['version']);

        $bugsql = $conn->prepare('INSERT INTO Update (bug_id,
state, priority, criticality, updated, responder_id, description,
version_id)
        VALUES (:bug_id, :state, :priority, :criticality,
:updated, :responder_id, :description, :version_id)');
        return $bugsql->execute($updbug);

    }

    public function getTeams()
    {
        $conn = DBase::createConn();
        $bugs = $conn->prepare('SELECT Team.id as team_id,
Team.name as team_name,
        FROM Team
        WHERE Team.id = ParticipantTeam.team_id AND
ParticipantTeam.participant_id = :participant_id
        ORDER BY Bug.id desc');
        $bugs ->bindParam('participant_id', $_SESSION['user']);
        $bugs ->execute();
        $bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
        $bugs = $this->textPriority($bugs);
        $bugs = $this->textCriticality($bugs);
        $bugs = $this->textState($bugs);
        return $bugs;
    }

```

```

public function editBug($bug_id, $newInfo)
{
    unset($newInfo ['submit']);
    $newInfo ['os_id'] = $this->getOSId($newInfo['os_id']);
    $newInfo ['criticality'] = $this-
>getCriticality($newInfo['criticality']);
    $today = date('Y-m-d H:i:s');
    $newInfo ['updated'] = $today;
    $pos = $this->isNew($newInfo, $bug_id);
    if(!$pos)
        return true;

    $conn = DBase::createConn();
    $bugsql = $conn->prepare('UPDATE Bug SET name = :name,
description = :description, conditions = :conditions, device =
:device, software = :software, os_id = :os_id, steps = :steps, updated
= :updated, criticality = :criticality');
    $newbug = $bugsql ->execute($newInfo);
    $this->changeBug($bug_id, $newInfo, $pos);
    return $newbug;
}

public function getTeamProjects()
{
    $conn = DBase::createConn();
    $bugs = $conn->prepare('SELECT Bug.id as bug_id,
Bug.name as bug_name,
Bug.priority as bug_priority,
Bug.criticality as bug_criticality,
Bug.obtained as bug_obtained,
Bug.updated as bug_updated,
Bug.state as bug_state
FROM Bug, Project, Team, ParticipantTeam
WHERE Bug.project_id = Project.id AND
Project_team_id = ParticipantTeam.team_id AND
ParticipantTeam.participant_id = :participant_id
ORDER BY Bug.id desc');
    $bugs ->bindParam('participant_id', $_SESSION['user']);
    $bugs ->execute();
    $bugs = $bugs->fetchAll(PDO::FETCH_ASSOC);
    $bugs = $this->textPriority($bugs);
    $bugs = $this->textCriticality($bugs);
    $bugs = $this->textState($bugs);

```

```

        return $bugs;
    }

    public function getActiveComments()
    {
        $conn = DBase::createConn();
        $comments = $conn->prepare('SELECT Comment.id as
comment_id,
        Comment.created as comment_created,
        Comment.quot as comment_quot,
        Bug.id as bug_id,
        Bug.name as bug_name
        FROM Comment
        INNER JOIN Bug on Comment.bug_id = Bug.id
        INNER JOIN CommentMessage on
CommentMessage.comment_id = Comment.id
        WHERE CommentMessage.participant_id = :user_id
        ORDER BY Bug.id desc');
        $comments ->bindParam('user_id', $_SESSION['user']);
        $comments ->execute();
        $comments = $bugs->fetchAll(PDO::FETCH_ASSOC);
        return $comments;
    }

}

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд № 1

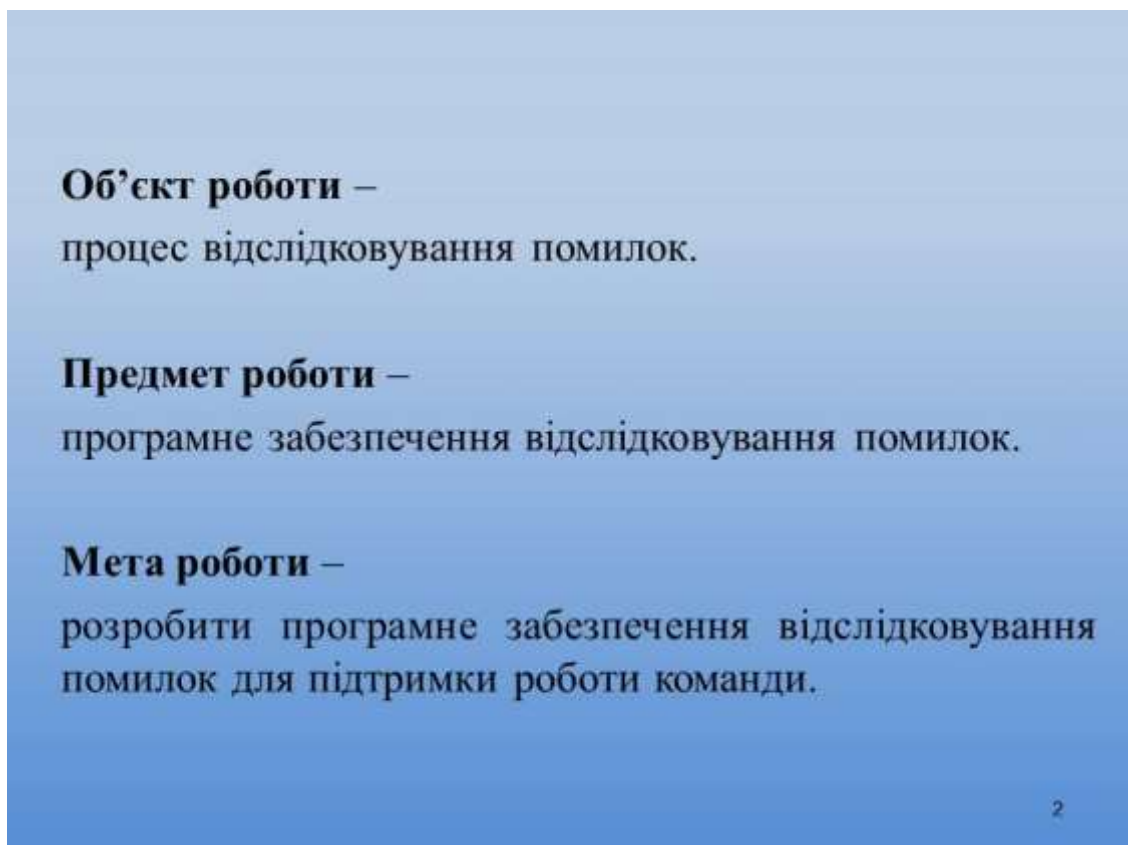


Рисунок В.2 – Слайд № 2

Завдання дипломної кваліфікаційної роботи

- визначення вимог до програмного забезпечення відслідковування помилок;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

3

Рисунок В.3 – Слайд № 3

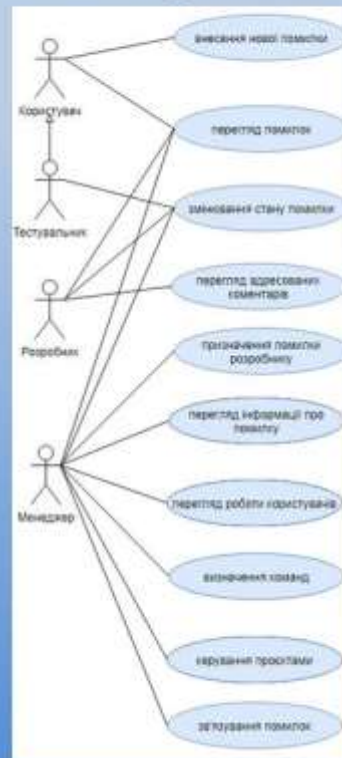
Вибір технології програмування

Критерій	PHP	Node.js
Тип	Скриптова мова серверної сторони	Середовище виконання
Швидкість завершення проєкту	Більша	Менша
Легкість доступу до баз даних	Більша	Менша
Продуктивність	Менша	Більша
Бібліотека готових модулів	Більша	Менша
Веброзробка	Так	Так

4

Рисунок В.4 – Слайд № 4

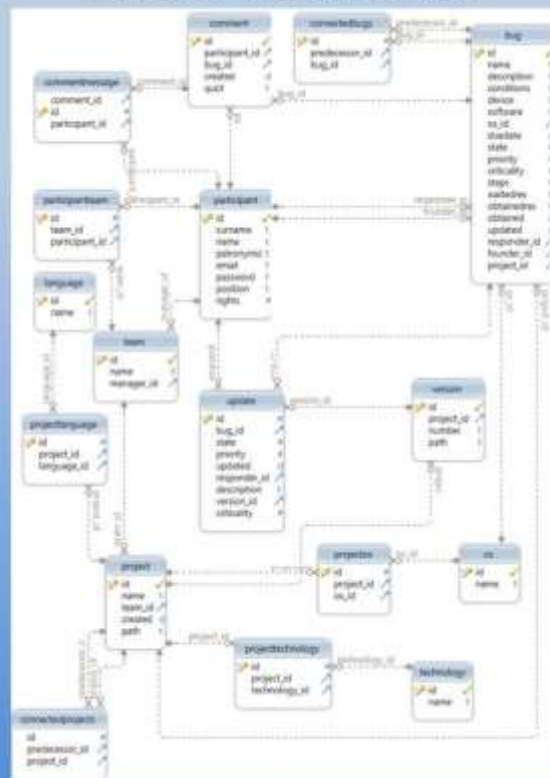
Діаграма прецедентів



5

Рисунок В.5 – Слайд № 5

Схема бази даних



6

Рисунок В.6 – Слайд № 6

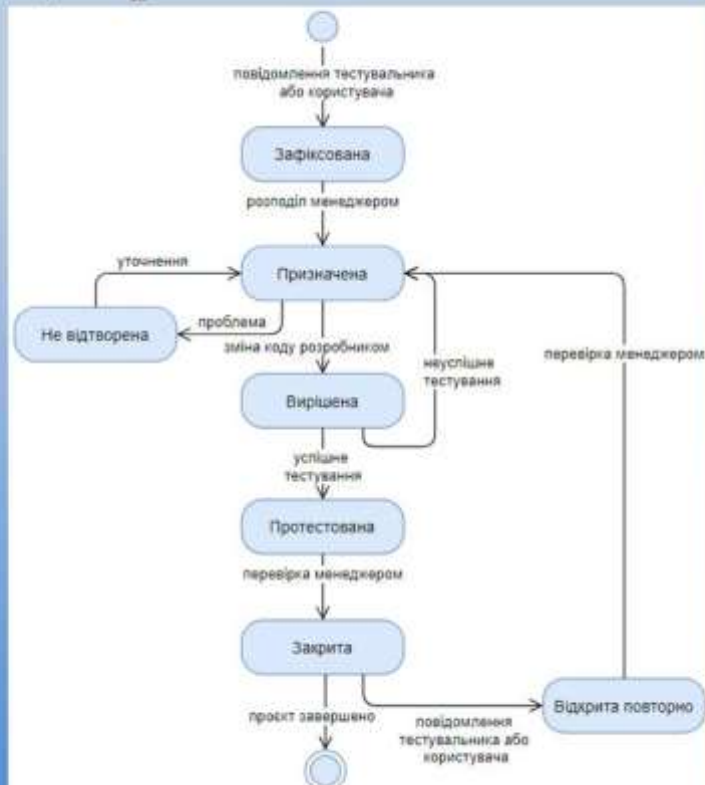
Шаблон проектування Model-View-Controller



7

Рисунок В.7 – Слайд № 7

Діаграма станів помилки



8

Рисунок В.8 – Слайд № 8



Рисунок В.9 – Слайд № 9

Робота з програмою

Програмне забезпечення відслідковування помилок: тестувальник.

Варіанти

Тестування

Усі помилки

За проектами

Фінансові помилки

Вихід

Тестування

Помилки, готові до тестування

Примітки

#	Назва	Пріоритет	Критичність	Створено	Оновлено	Проект	Розробник	Результат	Редагувати
43	Помилка оновлення даних	Середній	Незначна	08.05.2021	11.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
41	Неможливість встановити налаштування	Низький	Незначна	08.05.2021	10.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
38	Помилка при встановленні налаштувань	Високий	Критична	08.05.2021	11.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
37	Помилка при встановленні налаштувань	Високий	Критична	08.05.2021	10.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
35	Неможливість встановити налаштування	Середній	Незначна	08.05.2021	10.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
34	Помилка при встановленні налаштувань	Високий	Критична	07.05.2021	10.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
33	Неможливість встановити налаштування	Низький	Незначна	07.05.2021	09.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати
31	Помилка при встановленні налаштувань	Низький	Незначна	07.05.2021	09.05.2021	TestManager	testmgr@developer.com	Різка	Редагувати

Copyright © 2021 "Інформаційні технології"

Рисунок В.10 – Слайд № 10

Режим роботи тестувальника з програмою

Програмне забезпечення відслідковування помилок: тестувальник

- Відслідковування
- Тестування
- Усі помилки
- За проектом
- Внести помилку**
- Вихід

Внесення даних виявленої помилки

Назва помилки

Повний опис помилки

Очікуваний пріоритет: Середній

Очікувана критичність: Невисока

Проект: Обреша

Пристрій, на якому виявлено помилку

Очікуваний результат

Отриманий результат

Кроки для відтворення помилки

Операційна система Програма

Рисунок В.11 – Слайд № 11

Режим роботи менеджера з програмою

Програмне забезпечення відслідковування помилок: менеджер

- За проектом**
- Усі помилки
- Проекти
- Розробники
- Кандидати
- Вихід

Помилки, внесені за проектом #10 JustPlanner

Назва

Пріоритет

#	Назва	Пріоритет	Критичність	Стан	Відповідальний	Звернення	Звернути	Прийняти	Закрити	Редагувати
43	Помилка розробки модуля	Середній	Невисока	Виконано	Сидан	11.05.2021	Звернути	Прийняти		Редагувати
40	Не відтворено помилки користувачем	Середній	Невисока	Закрито	Сидан	12.05.2021		Прийняти		Редагувати
38	Помилка збоїв програми	Високий	Критична	Протестована	Винос	11.05.2021	Звернути	Прийняти	Закрити	Редагувати
35	Не відтворено помилки користувачем	Середній	Невисока	Протестована	Тодорова	13.05.2021	Звернути	Прийняти		Редагувати

Copyright © 2021 МП "Інформаційні технології"

Рисунок В.12 – Слайд № 12

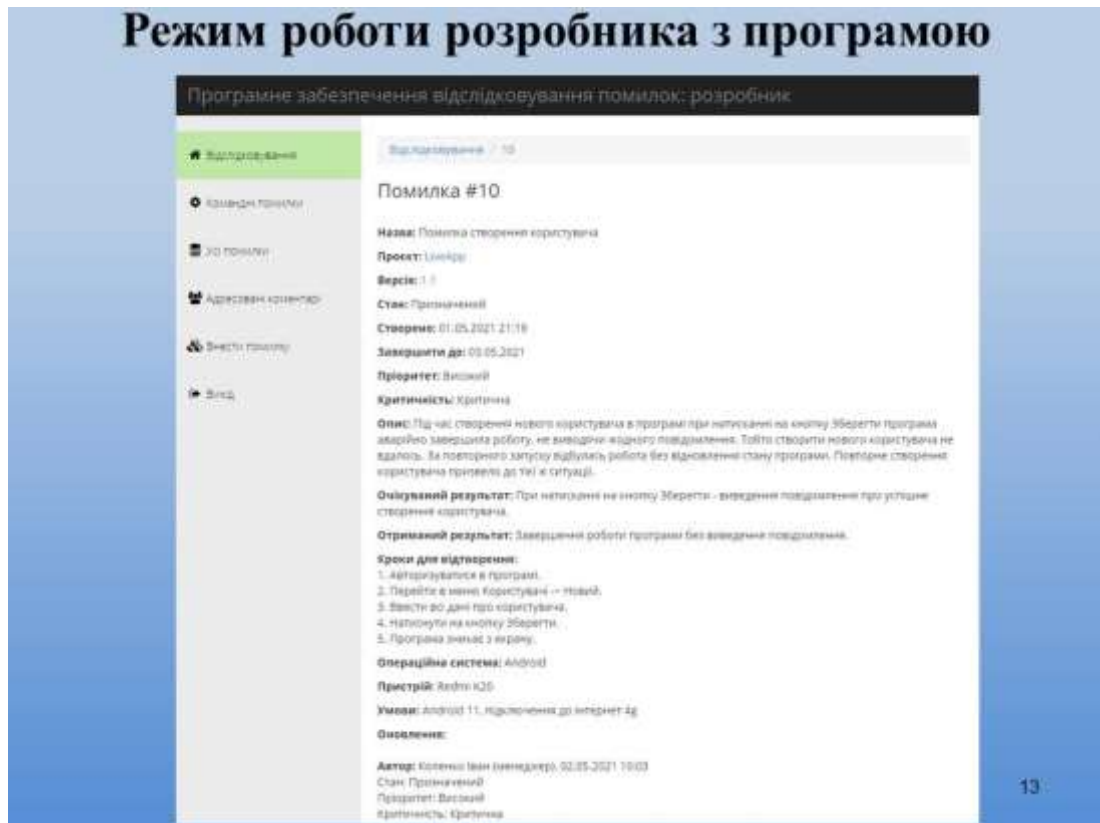


Рисунок В.13 – Слайд № 13

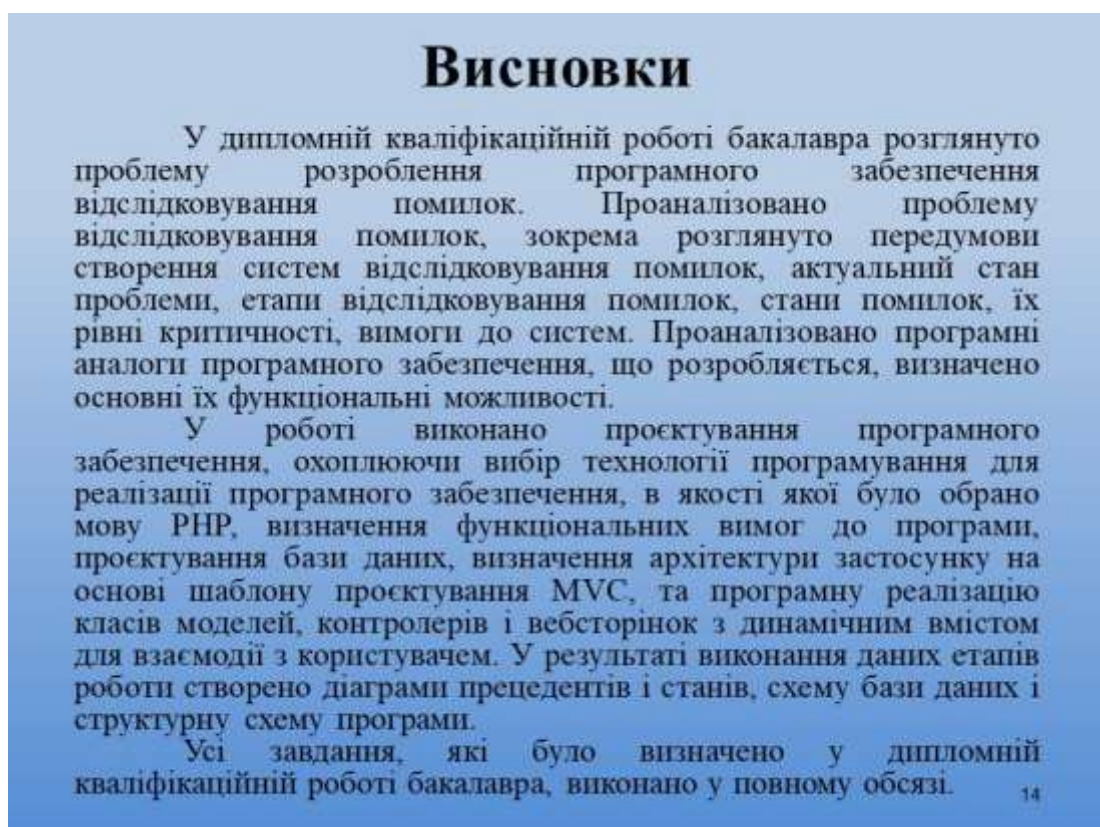


Рисунок В.14 – Слайд № 14