Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Медіасистем та технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Багтрекінгова система менеджменту проектів
(тема)

Виконав:
студентка 2 курсу, групи ТЕМВм-19-1

Злидень І.О.

Спеціальності 186 Видавництво та поліграфія

Тип програми Освітньо-професійна

Освітня програма
Технології електронних мультимедійних видань

Керівник проф. Манаков В.П.

Допускається до захисту
Зав. кафедри МСТ _____ Ткаченко В.П.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Технології електронних мультимедійних видань
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
«26» жовтня 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Злидню Івану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Багтрекінгова система менеджменту проекті

Затверджена наказом по університету від 23 жовтня 2020 р. № 1432 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 грудня 2020 р.

3. Вихідні дані до роботи
Метою магістерської атестаційної роботи є дослідження стану автоматизації процесу документування знайдених невідповідностей. Об'єктом дослідження даної роботи є процес покращення автоматизації документуваня. Предмет дослідження – багтрекінгова система менеджменту проектів

4. Перелік питань, що потрібно опрацювати в роботі
Вступ. Дослідження стану автоматизації процесу документування знайдених невідповідностей. Основні засади в розробці багтрекінгових систем. Існуючі інструменти розробки ІС. Реалізація прототипу інформаційної системи баг-трекінгу. Проектування багтрекінгової системи менеджменту проектів. Економічне обґрунтування. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Презентація: Титульна сторінка, Завдання на магістерську атестаційну роботу, Актуальність дослідження, Мета та задачі дослідження, Аналіз багтрекінгових систем, Опис процесу дослідження стану автоматизації стану документування, Розробка багтрекінговой системи менеджменту проектів, Економічне обґрунтування, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Манаков В. П.		
Економічна частина	проф. Полозова Т.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу магістра		
2	Аналіз та знаходження помилок існуючих багтрекінгових систем	06.11.2020–07.11.2020	
3	Проведення теоретичних досліджень	08.11.2020–10.11.2020	
4	Розробка багтрекінгової системи	11.11.2020–18.11.2020	
5	Економічна частина	20.11.2020–22.11.2020	
6	Оформлення пояснювальної записки	22.11.2020–28.11.2020	
7	Оформлення графічної частини	28.11.2020–29.11.2020	
8	Захист атестаційної роботи	23.12.2020	

Дата видачі завдання 26 жовтня 2020 р.

Студент _____ _Злидень І.О._
(підпис)

Керівник роботи _____ ___ проф. Манаков В. П.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 61 сторінку, 9 таблиць, 14 рисунків, 49 використаних джерел, 1 додаток.

БАГТРЕКІНГОВА СИСТЕМА,АВТОМАТИЗАЦІЯ, ДОСЛІДЖЕННЯ, САЙТ.

Метою роботи – створення багтрекінгової системи менеджменту проектів та дослідження стану автоматизації процесу документування знайдених невідповідностей .

Об’єктом дослідження даної роботи є процес документування .

Предмет дослідження – стан автоматизації процесу документування знайдених невідповідностей.

Результатами атестаційної роботи є створена багтрекінгова система та покращення процесу документування.

У розділі "Економічна частина" представлене економічне обґрунтування витрат на проведення науково-дослідної роботи.

РЕФЕРАТ

Пояснительная записка содержит 61 страницу, 9 таблиц, 14 рисунков, 49 использованных источников, 1 приложение.

БАГТРЕКИНГОВА СИСТЕМА, АВТОМАТИЗАЦИЯ, ИССЛЕДОВАНИЯ, САЙТ.

Целью работы - создание багтрекинговои системы менеджмента проектов и исследования состояния автоматизации процесса документирования найденных несоответствий.

Объектом исследования данной работы является процесс документирования.

Предмет исследования – состояние автоматизации процесса документирования найденных несоответствий.

Результатами аттестационной работы является созданная багтрекингова система и улучшения процесса документирования.

В разделе "Экономическая часть" представлено экономическое обоснование затрат на проведение научно-исследовательской работы.

ABSTRACT

The explanatory note contains 61 pages, 9 tables, 14 pictures, 49 sources used, 1 application.

BAG TRACKING SYSTEM, AUTOMATION, RESEARCH, SITE.

The purpose of the work is to create a bug tracking system for project management and study the state of automation of the process of documenting the inconsistencies found.

The object of research in this work is the documentation process.

The subject of research is the state of automation of the process of documenting the found inconsistencies.

The results of the certification work are the created bug tracking system and the improvement of the documentation process.

Section "Economic part" contains economic substantiation of expenses for carrying out research work.

ЗМІСТ

С.

ВСТУП

Ні для кого не секрет, що ІТ сфера стрімко розвивається по всьому світу в тому числі і Україна. Великий спектр компаній у цій сфері надають свої послуги по всьому світу. Одним з напрямків ІТ-послуг є саме забезпечення якості ПЗ.

Забезпечення якості (Quality Assurance) – це поняття, яке являє собою сукупність заходів, що охоплюють усі етапи розробки, випуску та експлуатації програмного забезпечення без виключення. Це діяльність (активності) на усіх етапах життєвого циклу ПЗ, яка здійснюється для забезпечення необхідного рівня якості продукту, що розробляється.

QA-інженер - Quality Assurance engineer (Quality Assurance дослівно означає «забезпечення якості») - це фахівець по тестування програмного забезпечення. Його діяльність спрямована на поліпшення процесу розробки ПЗ, запобігання дефектів і виявлення помилок в роботі продукту.

В наш час інформаційних технологій QA-інженерія є досить затребуваною, адже кожен програмний продукт потребує постійної перевірки якості на усіх етапах свого життя. Тестувальнику потрібно описувати всі знайдені, баги в спеціальній системі. Він детально описує проблему, присвоює пріоритет по її усуненню, описує шлях до проблеми і ще вказує безліч деталей, які допоможуть команді розробників виправити всі невідповідності. Для систематизації знайдених помилок програмного продукту використовують спеціальні інформаційні системи, які називають багтрекерами.

Багтрекер – це інформаційна системи, що розроблена для того, щоб тестувальники та програмісти могли відстежувати історію звітів про баги і взаємодіяти один з одним під час своєї роботи. Він може розглядатись як різновид системи відстеження проблем.

Компанія з тестування ПЗ неодмінно використовує багтрекер. Дана програма є незамінним інструментом перевірки ПЗ. Багтрекер є прикладною програмою, яка розроблена спеціально з метою контролю та обліку всіх знайдених помилок і збоїв.

Система відстеження помилок сприяє економії часу всіх учасників процесу розробки. Багтрекер в простій і зрозумілій формі надає всі виявлені невідповідності та помилки.

Існує величезна кількість баг-трекерів. Всі вони переслідують одну мету, але відрізняються інтерфейсом і функціональністю. Вибір системи залежить від побажань і потреб тестувальника, а також від вимог самого замовника.

Наявність багтрекера є дуже важливим компонентом у розробці програмного забезпечення, вони широко застосовуються компаніями, що розробляють програмні продукти. Загалом, використання багтрекера є однією з

«ознак хорошої команди програмістів»

Об’єктом дослідження являється галузь QA-інженерії, а оскільки однією з основних цілей тестування є виявлення дефектів та надання інформації для прийняття рішень, то предметом дослідження є конкретний процес документування знайдених дефектів.

Завданням атестаційної роботи є:

– визначення основних засад у розробці багтрекінгових систем;

– визначення структурних особливостей системи.

– вивчення існуючих інструментів розробки іс.

– дослідження існуючих багтрекерів-аналогів.

– формулювання ключових елементів системи.

– розробка загальної структури багтрекінгових систем менеджменту проектів.

– моделювання основних процедур за допомогою модулів конструктора іс.

– опис послідовного алгоритму роботи багтрекера.

Відповідно метою є сама система з її функціоналом, допоможе систематизувати процеси тестування ПЗ.

1 ДОСЛІДЖЕННЯ СТАНУ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ДОКУМЕНТУВАННЯ ЗНАЙДЕНИХ НЕВІДПОВІДНОСТЕЙ

1.1 Основні засади в розробці багтрекінгових систем

Для початку дослідження необхідно визначити основні поняття та засади в QA-інженерії та в розробці інформаційних систем в цілому.

Взагалі, інформаційна система – це організований набір елементів, що збирає, обробляє, передає, зберігає та надає дані. До складу ІС входять: люди, обладнання, процеси, процедури, дані та операції. Інформаційна система складається з наступних компонентів:

– структура системи;

– функції кожного елемента системи;

– вхід і вихід кожного елементу і системи в цілому;

– мета і обмеження системи та її окремих елементів.

Інформаційна система представляє не тільки функціонування об’єкта управління, а й впливає на нього через органи управління. Вона являється сукупністю інформаційних процесів для задоволення потреби в інформації різних рівнів прийняття рішень. Метою ІС є продукування інформації для використання управлінським апаратом. Система такого роду забезпечує нагромадження, передачу, збереження, оброблення та узагальнення інформації “знизу вгору”, а також конкретизує інформацію “зверху донизу”. ІС призначена для опису економічного об’єкта, його станів, взаємодії, що виражається через економічні показники. Вона покликана своєчасно подавати органам управління необхідну і достатню інформацію для прийняття рішень, якість яких забезпечує високоефективну діяльність об’єкта управління та його підрозділів.

До головних завдань належать:

– виявлення джерел інформації;

– збирання, реєстрація, обробка та видача інформації, що характеризує стан виробництва та управління;

– розподіл інформації між керівниками, підрозділами та виконавцями відповідно до їх участі в управлінні.

Найбільш важливими принципами побудови ефективних інформаційних систем є наступні.

Принцип інтеграції, що полягає в тому, що оброблювані дані, одного разу введені в систему, багаторазово використовуються для вирішення великої кількості завдань.

Принцип системності, що полягає в обробці даних в різних аспектах, щоб отримати інформацію, необхідну для прийняття рішень на всіх рівнях управління.

Принцип комплексності, що полягає в механізації і автоматизації процедур перетворення даних на всіх етапах функціонування інформаційної системи.

Інформаційні системи також класифікуються:

– за функціональним призначенням: виробничі, комерційні, фінансові, маркетингові та ін.;

– по об’єктах управління: інформаційні системи автоматизованого проектування, управління технологічними процесами, управління підприємством (офісом, фірмою, корпорацією, організацією) і т.п.;

– за характером використання результатної інформації: інформаційно-пошукові, призначені для збору, зберігання і видачі інформації за запитом користувача; інформаційно-радять, пропонують користувачеві певні рекомендації для прийняття рішень (системи підтримки прийняття рішень); інформаційно- керуючі, результатная інформація яких безпосередньо бере участь у формуванні керуючих впливів.

Розвиток комп’ютерної інформаційної технології нерозривно пов’язаний з розвитком інформаційних систем. Вони використовуються в економіці для автоматизованого (людиномашинного) розв’язування економічних задач. Для того, щоб розв’язати будь-яку задачу з допомогою комп’ютера необхідно спочатку створити інформаційне забезпечення (забезпечити розрахунки потрібними даними) і математичне забезпечення (створити математичну модель розв’язування задачі, за якою складається програма для ЕОМ).

Необхідна для розв’язування інформація може надходити безпосередньо (вхідна інформація) або через систему інформаційного забезпечення, яка може поповнюватися і за рахунок нової інформації. Визначальною особливістю інформаційної системи є те, що вона забезпечує користувачів інформацією з кількох організацій.

У системах обробки інформації головними її компонентами є дані та обчислення. Більшість інформаційних систем управління інформаційними ресурсами в організаціях містять і багато інших компонентів, таких як вимоги, запити, тригери і звіти. І всі вони, зокрема, містять великі описи свого власного змісту в тій чи іншій формі. Ці описи необхідні для інтерпретації і для коректного використання наданої інформації (коли в системі немає повного опису, то передбачається, що користувачі отримують його з іншого джерела).

Для головних компонент інформації (даних і обчислень) важливе значення має така характеристика, як їх надмірність. Означення надмірності суттєво залежить від одиниці інформації. Коли одиниця вибрана, то надмірність – це просто дублювання однієї й тієї самої одиниці в системі. Важливим у виборі одиниці інформації є її розмір. Вибір занадто малої одиниці призводить до високого рівня незалежності блоків інформації, але водночас і до збільшення накладних витрат затрат на їх підтримку; у разі взяття великої одиниці неможливо виключити численне дублювання підблоків інформації

Оскільки багтрекер – це інформаційна система обліку та документообігу, то відповідно він має всі риси такого роду системи.

Інформаційне забезпечення ІС обліку передбачає створення єдиного інформаційного фонду, систематизацію та уніфікацію показників і документів (багів) та розроблення засобів формалізованого опису даних тощо.

Інформаційне забезпечення є важливим елементом автоматизованих інформаційних систем обліку, який призначений для відображення інформації, що характеризує стан керованого об’єкта і є основою для прийняття управлінських рішень.

У процесі розробки інформаційного забезпечення варто визначити:

– склад інформації, що охоплює перелік інформаційних одиниць або сукупностей, достатніх для розв’язання комплексу задач;

– структуру інформації та перетворення її, тобто формування показників документів;

– характеристики руху інформації, тобто обсяг потоків, маршрути, терміни;

– характеристику якості інформації;

– способи перетворення інформації.

Програмне забезпечення та інформаційна технологія, зорієнтованою на кінцевого користувача, ведуться паралельно з організацією інформаційного забезпечення. Основою інформаційного забезпечення ІС є інформаційна база (ІБ), що застосовується задля повного функціонування ІС. За складом та змістом вона повинна відповідати вимогам тих задач і проектувати ті системи, які розв’язуються на основі цієї ІБ.

1.2 Структурні особливості системи

Структуру інформаційної системи складає сукупність окремих її частин – підсистем. Підсистемою є частина системи, що виділена за певною ознакою. Структура будь-якої інформаційної системи може бути представлена як сукупність підсистем, що забезпечують інформаційне, технічне, математичне, програмне, організаційне і правове забезпечення (рис. Б.2).

У цій ситуації інформаційне забезпечення визначається як сукупність єдиної системи класифікації та кодування повідомлень, уніфікованих систем документації, схем інформаційних потоків, які циркулюють в організації, а також методологія побудови баз даних.

Технічне забезпечення – комплекс технічних засобів, що призначені для роботи інформаційної системи, а також відповідна документація на ці засоби й технологічні процеси.

Математичне й програмне забезпечення - сукупність математичних методів, моделей, алгоритмів і програм для реалізації цілей і завдань інформаційної системи, а також нормального функціонування комплексу технічних засобів.

Організаційне забезпечення – сукупність методів і засобів, що регламентують взаємодію працівників з технічними засобами й між собою в процесі розробки й експлуатації інформаційної системи.

Правове забезпечення – сукупність правових норм, які визначають створення, юридичний статус і функціонування інформаційних систем, що регламентують порядок одержання, перетворення й використання відомостей.

1.3 Існуючі інструменти розробки ІС

Існує достатньо широкий спектр інструментів розробки інформаційних систем. Оскільки, планується розробка інформаційної системи без особливого застосування мов програмування, то і розглядатимуться альтернативи, які являють собою алгоритмічно-комбіновані системи-конструктори з гнучким інтерфейсом та повністю або частково відкритим кодом для редагування.

WordPress – це система керування вмістом, що має відкритий кодом і, яка через свою простоту в установленні та використанні широко застосовується для створення веб-сайтів. Сфера використання є досить широкою, починаючи від блогів та закінчуючи складними за структурою веб-сайтіами. Вбудована система тем і плагінів у поєднанні з вдалою архітектурою дозволяє конструювати на основі WordPress практично будь-які веб-проекти.

Написана система за допомогою мови програмування PHP з використанням бази даних MySQL. Початковий код поширюється на умовах ліцензії GNU General Public License .

Від інших систем управління вмістом сайту WordPress відрізняється достатньо простим інтерфейсом. Програмісту це можливо не гарантує всієї повноти зручностей, зате адміністратору веб-ресурсу значно полегшує роботу.

Публікації створюються моментально, сама платформа встановлюється не більше ніж за 5 хвилин.

Уже в другій версії CMS з'явився візуальний редактор. У наступних версіях додалися автозбереження (версія 2.1), можливість вказувати власний пароль при установці і задавати свої фонові зображення, підтримка користувацького меню (версія 3.0), а також формат мобільного застосування (версія 4.1.1.).

WordPress розповсюджується в мережі вільно і безкоштовно, важить кілька мегабайт і завантажується з офіційного сайту розробників. Там же, на сайті, можна знайти російськомовний Кодекс Wordpress, що містить актуальну корисну інформацію по роботі з системою для новачків і досвідчених розробників. Нижче наведено зображення адміністративної панелі CMS WordPress

Joomla! – це система керування вмістом (CMS), яка написана на мовах PHP і JavaScript і використовує в якості сховища бази даних СУБД MySQL або інші стандартні реляційні СУБД. Система є вільним програмним забезпеченням, поширюваним під ліцензією GNU GPL.

Система управління вмістом Joomla! є, по суті, відгалуженням широко відомої CMS Mambo, яка утвориласячерез те, що команда незалежних розробників відокремилася від проекту Mambo через незгоду в економічній політиці і 16 вересня 2005 року в світ вийшла перша версія Joomla!, що по своїй суті є перейменованою Mambo 4.5.2.3 і включає в себе виправлення знайдених на той момент помилок і вразливостей.

До літа 2008 року по числу щоденних завантажень Joomla! посіла друге місце після WordPress зі значним відривом від інших подібних систем.

CMS Joomla! включає в себе мінімальний набір інструментів при початковій установці, який доповнюється в міру необхідності. Це знижує захаращення адміністративної панелі непотрібними елементами, а також знижує навантаження на сервер і економить місце на хостингу.

Joomla! надає можливість відображати інтерфейс фронтальної та адміністративної частини будь-якою мовою. Каталог розширень містить широкий спекир мовних пакетів, які встановлюються засобами адміністрування.

Доступні пакети українською, російською, білоруською та ще деякими мовами пострадянського простору.

Структурно система являє собою ядро, до якого приєднуються компоненти. Компоненти можуть складатися по функціональному призначенню з модуля, плагіна і/або шаблону:

– модулі – це елементи, які відображаються на сторінці користувачів під виглядом окремих блоків;

– плагіни – це елементи, що відображаються на адміністративній панелі інструментів і надають додаткову функціональність розроблюваній системі;

– шаблоном є елемент, який відповідає за уніфіковане оформлення всього сайту.

До складу ядра Joomla! входить:

– блок (функціональність визначається HTML-кодом всередині нього) і його різновиди з широкими можливостями керування з адміністративної панелі: банери (модулі для розміщення реклами), контакти (блок для розміщення контактної інформації та форми зворотного зв'язку), стрічка новин (RSS підписка);

– матеріал з мітками (ключові слова) і категоріями (папками), що слугують навігацією користувачів, редактором для зручного оформлення матеріалу.

Не входять до ядра вільні сторонні розширення: плагіни для резервного копіювання (backup) і відновлення, наприклад, Akeeba Backup.

Створення власного розширення здійснюється за допомогою PHP з використанням для взаємодії API Joomla!.

Wix – один з найавторитетніших конструкторів сайтів. Сервіс, який вже 12 років задає тенденції розвитку всієї ніші.

Wix найкраще підходить для створення різного роду сайтів-візиток – складних з дизайнерської і функціональної точок зору: портфоліо, бізнес-сайт, промо-сторінка або Лендінгем. Дуже вдалими виходять блоги - їх легко робити, зручно вести. eCommerce складова теж розвинена, особливо з урахуванням додатків і фірмових ноу-хау на зразок Wix Code.

Область застосування Wix можна охарактеризувати трьома факторами:

– сайти з невеликою кількістю сторінок, оскільки дизайн кожної потрібно формувати вручну;

– блоги, форуми, магазинні вітрини та інші типи сторінок, які динамічно розширюються ніяк не ускладнювати роботу над сайтом. Впливає обсяг тільки статичних сторінок на кшталт контактів, переваг, умов співпраці, історій компаній, виробництва та іншого;

– необхідність глибокої і тонкої настройки дизайну, а також введення великої кількості дрібних функціональних елементів.

Також слід зазначити, що магазин додатків Wix містить багатий асортимент додаткового функціоналу (по суті, модулів, якщо брати за аналогією з іншими системами). Кілька категорій корисних додатків (всього понад 250 найменувань) допоможуть досягти практично будь-якої розумної мети.

Wix ідеально підходить для створення маленьких і середніх за обсягом сайтів зі складним дизайном і функціональністю. Цей конструктор може бути простим, поверхневим, якщо потрібно: взяли готовий шаблон, замінили демо- контент, поправили кольору, отримали сайт. З тим же успіхом його можна використовувати для конструювання як завгодно складних сторінок з купою деталей: анімаціями, ефектами, формами для збору і пакування інформації в бази даних, інтерактивними елементами, спливаючими вікнами авторизацій різних етапів допуску до інформації, всілякими опитуваннями, табами, колонками і іншим.

Сукупні можливості Wix складаються з функціоналу редактора, загальних налаштувань, панелі управління, встановлених сторонніх додатків і опцій, які можна впровадити на сайт самостійно за допомогою інструменту Wix Code.

Будь-який розділ редактора містить підказки по роботі з ним. При наведенні курсора на кружечок «і» відобразиться коротке пояснення і посилання на повний варіант FAQ по темі. Абсолютно всі опції і віджети забезпечені поясненнями, просто в деяких потрібно натиснути на знак «?» Для виклику спливаючого вікна з інструкцією. Також можна відразу піти в Центр Підтримки - базі знань Wix зможливістю пошуку відповідей на запит. Пробігшись за категоріями бази знань

очима, ви зможете отримати уявлення про можливості конструктора - назви розділів FAQ чітко перераховують функціонал.

Фірмовий магазин додатків, які можна додати на сайт. Дуже сильна частина Wix. Використання даного розділу в кілька разів розширює стоковий функціонал платформи. Тут міститься величезна кількість (на поточний момент 252) додатків різної спрямованості: форми, робота з соцмережами, галереї, додатки для магазину і маркетингу. Можна додати інтерактивний календар, наприклад, інструменти для Email-маркетингу, опитування, Dropbox, eCommerce-примочки, SEO-інструменти та інші корисності в допомогу стандартного функціоналу конструктора. Всі додатки відсортовані за категоріями. Більшість з них розроблені командою Wix, інші, як правило, є інтеграцією можливостей сторонніх сервісів. Приблизно половина додатків зовсім безкоштовні, а інші мають як безкоштовний варіант використання, так і платний з додатковими можливостями.

1.4 Баг-трекери аналоги

Mantis Bug Tracker (MantisBT) – це безкоштовна система відстеження, у якої висхідний код є відкритим. Розповсюджується за ліцензією GNU General Public License 2. Система створена з метою задоволення загальних потреб відстеження запитів (англ. issue management system), управління проектами, та, найчастіше, для відстеження помилок в ПЗ. Інтерфейс користувача – веб-сайт.

Назва та логотип ІС являють собою англомовний каламбур. Mantis (укр. богомол) відомий, зокрема, тим, що відстежує та їсть інших комах (англ. bug). А слово «bug» також загально відоме як помилка в програмному забезпеченні.

Mantis – це напевно, це найпоширеніший представник систем стеження за багами. Він написаний на мові PHP. Його не можна назвати ідеальним багтрекером, однак він здатний вирішувати всі основні завдання, які від нього потрібні.

Тестувальник програмного забезпечення змушений працювати в даному багтрекер безпосередньо за допомогою браузера. Користувачі даного продукту

постійно нарікають на проблеми з Unicode. Загалом, даний продукт ще вимагає особливої доопрацювання.

Jira. Багтрекер написаний на мові Java. Відображає хід виконання проектів, є зручні посилання, за допомогою яких можна контролювати звіти і поточні завдання. За допомогою даної системи можна створювати проекти через e-mail. Існує можливість імпорту звітів в Excel, а також можливість Wiki-форматування. Підтримує інтеграцію з Confluence. Даний багтрекер здатний працювати через захищене з'єднання із застосуванням SSL. Потенційних клієнтів, однак, може відлякувати вартість комерційної ліцензії.

JIRA це продукт, призначений для організації процесу контролю запитів і завдань, що має частину функціональності зазвичай властивої великим і дорогим системам управління проектами.

Ключовими поняттями в JIRA є проекти і завдання. Завдання створюються в проектах, для виконання завдань призначаються виконавці. Завдання можуть бути різного типу і мати підзадачі, завдання можуть бути пов'язаними з іншими завданнями. Статус завдань змінюється в процесі їх виконання.

Ви можете організувати контроль розробки проектів, роздавши завдання виконавцям, ви можете визначити свій власний метод руху завдань - від створення до виконання і контролю результатів, конфігурувати правила повідомлення про події всіх учасників процесу, управляти правами доступу користувачів і робити багато чого іншого.

JIRA приносить великий ефект будь-якої організації, діяльність якої можна інтерпретувати як виконання будь-яких проектів і завдань, що мають тематичні і тимчасові рамки. Головна перевага цього продукту в його ні з чим незрівнянну здатності настройки під ваші потреби. Наприклад, у фінансовій сфері, ви можете організувати процес оформлення кредиту, від заявки, до введення необхідних даних, до прийняття рішення і так далі. У сфері державного управління - ви можете створити завдання, визначити терміни виконання, приєднати документи, організувати процес проходження завдання між співробітниками і проконтролювати результат.

Ключовими поняттями в JIRA є проекти і завдання. Проекти служать для групування завдань. Завдання створюються в проектах, для виконання завдань призначаються виконавці. Завдання можуть бути різного типу і мати підзадачі, завдання можуть бути пов’язаними з іншими завданнями. Статус завдань змінюється в процесі їх виконання.

У JIRA багато понять пов’язані з певними проектами. Проект пов’язаного з ним лідера проекту, URL сайту проекту, схему розсилки нотифікацій, схему контролю доступу - дуже гнучкий механізм контролю доступу користувача до завдань проекту. Кожному проекту можна зіставити свою схему руху завдання (документообіг). Кожному проекту можна порівняти свій вигляд екрану - "зібрати" потрібне вікно з можливих компонентів.

Також, JIRA формує звіти по кожному проекту.

Завдання створюються в проектах. Завдання мають типи, наприклад: Завдання, Помилка, Нова ідея. Можна створювати і свої типи завдань. При описі кожного типу завдання є можливість управління набором полів.

JIRA дозволяє відшукувати завдання за всіма критеріями і по призначених для користувача полів, створювати фільтри, які можна зберегти і використовувати знову, а також зробити загальнодоступними і організувати автоматичну розсилку результатів роботи фільтрів членам робочої групи.

Для організації роботи з користувачами JIRA має групи користувачів і ролі.

JIRA має систему контролю доступу користувачів до проектів, завдань і функцій JIRA має систему контролю доступу користувачів до проектів, завдань і функцій, засновану на членство користувачів в групах і ролях.

Так, для кожного проекту, є можливість управління доступом кожної групи користувачів до кожної дії. Також, є можливість сформувати набір допусків в "роль". Типовий найпростіший поділ ролей в JIRA включає в себе такі ролі:

– адміністратора;
– керівник проекту;
– співробітника що працює над проектом;
– інші співробітники.

Але можливості JIRA цим не обмежуються, можливе створення спеціальних ролей, наприклад таких, яким можливо тільки читання завдань одного типу але неможливо іншого. Завдання JIRA в кожен момент часу мають певний статус. Можливі дії з завданнями, які мають той чи інший статус, визначається вбудованою системою управління рухом завдань. Тут наведена схема найпростішого опису руху завдання. У JIRA є можливість створення такої складної схеми руху завдання, яка потрібна. Схема руху завдання може бути своя для кожного відділу, проекту, типу завдання.

Схема руху редагується вбудованим редактором. Редагуючи рух завдань, створюючи нові статуси завдань (події) і визначаючи можливі дії, можна організувати будь-яку роботу.

Рух завдання можна зробити залежним від умов, застосовувати логіку І / АБО, виконувати визначені дії на кожному етапі руху завдання.

Завдання JIRA в кожен момент часу мають певний статус. Користувачі інформуються по e-mail в разі будь-яких дій із завданнями, для цього служить настроюється система нотифікації користувачів. Спільно з системою управління рухом завдання і налаштованим розсилаються фільтрами це дозволяє дуже ефективно інформувати всіх зацікавлених осіб про хід виконання завдання.

Bugzilla. Безкоштовний багтрекер, розроблений Mozilla Organization. Написаний на мові Perl. Свого роду еталон, з яким порівнюються всі інші системи стеження за багами з подібною функціональністю. Забезпечує автоматизацію роботи з документацією, підтримує тісну інтеграцію з системою електронної пошти. Відсутня інтеграція з SVN. Користувачі скаржаться на огидний інтерфейс і не дуже якісний код. Останнє не дивно, так як багтрекер був випущений на ринок ще в 1998 році.

Ключовим поняттям системи є баг («Bug») – деяке завдання, запит, рекламація з приводу помилки в системі, або просто повідомлення, яке потребує зворотного зв’язку, і призначення системи.

Сутність Bug має набір атрибутів, робота з якими – редагування і запити – є основними сценаріями використання Bugzilla.

Основні структурні атрибути є такими:

– Product – основний атрибут, що задає структуру. Кожен «Product» складається з набору компонентів. Можна включити класифікацію продуктів – додатковий підрозділ продуктів на групи;

– Component – додаткова структурна класифікація. Залежно від обраного компонента, баг може мати різний набір прапорів.

Серед атрибутів життєвого циклу слід виділити такі:

– Status – основний атрибут, який визначає поточний стан бага, тобто міру його активності – від самого «початкового» стану, коли він навіть не підтверджений, як баг, до благополучного завершення, коли баг виправлений / вирішене, що підтверджено Службою Якості. Набір станів залежить від конкретної інсталяції та настройки Bugzilla, однак, стандартний набір: UNCONFIRMED («Не підтверджено»);

– New («Новий») – тільки що зареєстрований або перевірений;

– Assigned («Акцептований») – користувач, вказаний в «Assigned-to», підтвердив свою відповідальність за цей баг. Баг може бути «перенаправлення» іншій особі, і знову стати «NEW»;

– Reopened – баг вже був одного разу вирішено, однак, рішення було або неправильними, або неостаточним.

– Resolution - цей атрибут має сенс тільки для багів в станах «Resolved», «Verified», «Closed».

Також, набір «резолюцій» можна налаштовувати, але стандартний набір наступний:

– Fixed («Вирішено») – означає, що завдання виконано або баг виправлений;

– Invalid («Некоректно») – неправильна або некоректна постановка, яка не має сенсу;

– Wontfix («Проблема є, але вирішувати її не будемо»);

– Duplicate («Дубль») – описана проблема вже реєструвалася в певному багу;

– Worksforme («А у мене працює...») – не вдалося відтворити описану проблему ні емуляцією сценарію, ні аналізом коду;

– Moved – проблема перенесена в іншу систему-«tracker»;

– Later – (Також можна використовувати) Завдання зафіксована, але її рішення відкладається на невизначений термін.

Trac – це безкоштовна і проста система управління проектами, але помітно спрощує наше життя при належному терпінні. Основою для Trac є SVN репозиторій. Тому якщо ви до сих пір не використовуєте в розробці Subversion, то швидше за все Trac буде для вас нецікавим. Основні функції включають:

– управління проектом (поділ проекту на етапи, контроль виконання, всі зміни по проекту заносяться на тимчасову шкалу);

– тікети (стандартна функціональність - облік помилок, зауважень, побажань з можливістю фільтрації і занесення соответсвенно в milestone, roadmap);

– перегляд сховища (досить зручний модуль по перегляду Subversion репозиторію проекту. Дозволяє переглядати вихідний код з урахуванням ревізії, а також змін;

– управління користувачами;

– Wiki.

Варто відмітити, що для розширення функціональності є можливість підключення додаткових модулів, але вони не є безкоштовними.

Redmine – відкритий (безкоштовний) web-серверний додаток для управління проектами та завданнями (в тому числі для відстеження помилок). Поширюється згідно з ліцензією GNU General Public License.

Особливістю сертифікації системи менеджменту IRIS є те, що вимоги стандарту повинні не просто виконуватися, а повинні бути представлені свідоцтва управління процесами проектування і розробки на кожному етапі виконання проекту, при схваленні кожного етапу робіт перед переходом до наступного. Написаний на Ruby Rails. Підтримує близько 30 мов. Інтеграція з SVN, CVS, GIT, Mercurial і ін. Може підтримувати кілька проектів, на відміну від Trac. Існує пакетне редагування, засноване на AJAX. Можна встановлювати різні користувальницькі ролі в проектах. Підтримує настроювання процесів (workflow). З мінусів: обмежені можливості при роботі з файлами і документами,

немає прав на окремі типи переходів настроюються процесів, неможливо управляти також правами доступу на рівні окремих полів завдання.

Користувачі є одним з центральних понять предметної області. Модель користувача є основою для ідентифікації і аутентифікації працює з системою персоналу і клієнтів, а також для авторизації їх в різних ролях, проектах і т. д.

Ролі користувачів визначаються гнучкою моделлю визначення прав доступу користувачів. Ролі включають в себе набір привілеїв, що дозволяють розмежовувати доступ до різних функцій системи. Користувачам призначається роль в кожному проекті, в якому він бере участь, наприклад, «керівник», «розробник» або «спостерігач» та інші. Користувач може мати кілька ролей.

Кожен користувач системи може подивитися завдання, в яких він бере участь як замовник, спостерігач і виконавець. Система кольором виділяє завдання з різними по терміновості параметрами.

Проект є одним з основних понять в предметної області систем управління проектами. Завдяки цій сутності можливо організувати спільну роботу і планування декількох проектів одночасно з розмежуванням доступу різним користувачам. Проекти допускають ієрархічну вкладеність.

Завдання є центральним поняттям всієї системи, що описує якусь задачу, яку необхідно виконати. У кожного завдання в обов’язковому порядку є опис і автор, в обов’язковому порядку завдання прив’язана до трекера.

Для того, щоб додати нове завдання в проект, необхідно перейти на вкладку «Нова задача», вибрати трекер завдання і заповнити обов’язкові (*) і додаткові (в тому числі і призначені для користувача) поля завдання.

Трекери є основною класифікацією, за якою упорядковано завдання в проекті. По суті, в «Redmine» трекери представляють собою аналог підкласів класу «Завдання» та є основою для визначення і сортування різного роду завдань, дозволяючи призначати для кожного їх типу різні поля. Прикладами трекерів є «Проблема», «Поліпшення», «Проектування», «Сервіс» і т.д.

Кожне завдання має статус. Статуси представляють собою окрему сутність з можливістю визначення прав на призначення статусу для різних ролей.

За відстеження змін параметрів завдань користувачами в системі RM відповідає вкладка: «Дія» та «Змінений параметр». Запис журналу відображає одну дію користувача по редагуванню параметрів завдання і/або додавання коментаря до неї. Тобто служить одночасно інструментом ведення історії завдання і інструментом ведення діалогу. Сутність «Змінений параметр» прив’язана до окремого запису журналу і призначена для зберігання старого і нового значення зміненого користувачем параметра.

Система підтримує облік витраченого часу завдяки сутності «Витрачений час», пов’язаної з користувачами і завданням. Сутність дозволяє зберігати витрачений час, вид діяльності користувача (розробка, проектування, підтримка) і короткий коментар до роботи. Ці дані можуть бути використані, наприклад, для аналізу внеску кожного учасника в проект або для оцінки фактичної трудомісткості і вартості розробки. Хоча Redmine і є безкоштовним та все ж деякі з модулів доступні лише після оплати.

Отже, проаналізувавши всі вище перелічені багтрекер-системи слід визначити, що існує велике різноманіття ІС такого типу з широким функціоналом, більшість із них є умовно безкоштовними, адже для одержання повного функціоналу все ж необхідно платити. Також більшість таких систем не є«чистими» багтрекерами, вони призначені більше для ведення проектів в цілому, а документування дефектів є лише побічним функціоналом. З більшістю аналогів також виникає проблема освоєння інтерфейсу та логіки системи через перенавантаження різноманітністю функціоналу.

2 ПРОЕКТУВАННЯ БАГТРЕКІНГОВОЇ СИСТЕМИ МЕНЕДЖМЕНТУ ПРОЕКТІВ

2.1 Опис ключових елементів системи

Для обґрунтування ключових рішень та обраних технологій, програмна система розбивається на складові частини: сервер, веб-додаток, база даних. Серверна частина програмної системи повинна забезпечувати постійну обробку клієнтських запитів та швидку відповідь на них. Простота серверного рішення з одного боку, та можливість розгортання архітектури на віддаленому сервері з другої, призвела до створення ряду вимог до серверної частини:

а) простота при створенні архітектури;

б) швидкість обробки передачі даних до користувача.

Наглядне розподілення наданих вимог і варіантів серверних рішень для даної системи відображене у таблиці 2.1, де кожен стовпець відображає вимоги до серверного рішення відповідно до наданого списку вимог, та кожен рядок відповідає за варіанти мов програмування а платформ для створення серверного рішення.

Таблиця 2.1 – Відповідність мов програмування до серверних вимог

	А	Б	В	Г	Д
PHP	+	+	+	+	-
Node.js	+	-	+	+	-
Java	-	+	+	+	-
Phyton	+	+	-	+	-

Результатом дослідження мов програмування та платформ для створення серверного рішення, відображеного в таблиці 4.1, є твердження, що для розробки серверу для даної програмної системи потрібно використовувати PHP.

Головним фактором мови PHP є практичність. PHP повинен надати програмісту засоби для швидкого і ефективного вирішення поставлених завдань. Практичний характер PHP обумовлений п’ятьма важливими характеристиками:

– традиційністю;
– простотою;
– ефективністю;
– безпекою;
– гнучкістю.

Існує ще одна «характеристика», яка робить PHP особливо привабливим: він поширюється безкоштовно. Причому, з відкритими вихідними кодами (Open Source). Мова PHP здаватиметься знайомим програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з Ci, Perl.

Код PHP дуже схожий на той, який зустрічається в типових програмах на C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP - мова, що поєднує переваги Perl і Ci і спеціально націлений на роботу в Інтернеті, мова з універсальним (правда, за деякими застереженнями) і ясним синтаксисом.

І хоча PHP є досить молодою мовою, він знайшов таку популярність серед web-програмістів, що на даний момент є мало не найпопулярнішою мовою для створення web-додатків (скриптів).

Сценарій PHP може складатися з 10 000 рядків або з одного рядка - все залежить від специфіки вашого завдання. Вам не доведеться довантажувати бібліотеки, вказувати спеціальні параметри компіляції або що-небудь в цьому роді. Механізм PHP просто починає виконувати код після першої екрануючої послідовності (<?) І продовжує виконання до того моменту, коли він зустріне парну екранує послідовність (?>). Якщо код має правильний синтаксис, він виконується в точності так, як вказав програміст.

PHP - мова, яка може бути вбудований безпосередньо в html-код сторінок, які, в свою чергу будуть коректно оброблятися PHP -інтерпретатором. Ми можемо використовувати PHP для написання CGI-сценаріїв і позбутися від безлічі незручних операторів виведення тексту. Ми можемо залучати PHP для формування HTML-документів, позбувшись від безлічі викликів зовнішніх сценаріїв. Велика розмаїтість функцій PHP позбавлять вас від написання багаторядкових призначених для користувача функцій на C або Pascal.

Ефективність є виключно важливим фактором при програмуванні для багатокористувацьких середовищ, до числа яких належить і web.

Дуже важлива перевага PHP полягає в його «двигуну». «Двигун» PHP не є ні компілятором, ні інтерпретатором. Він є транслюється інтерпретатором. Такий пристрій «двигуну» PHP дозволяє обробляти сценарії з достатньо високою швидкістю.

За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше аналогічних їм програм, написаних на Perl. Однак, щоб не робили розробники PHP, відкомпільовані виконувані файли будуть працювати значно швидше - в десятки, а іноді і в сотні разів. Але продуктивність PHP цілком достатня для створення цілком серйозних web-додаткі. PHP надає в розпорядження розробників і адміністраторів гнучкі і ефективні засоби безпеки, які умовно поділяються на дві категорії: засоби системного рівня і засоби рівня додатки.

У PHP реалізовані механізми безпеки, що знаходяться під управлінням адміністраторів; при правильному налаштуванні PHP це забезпечує максимальну свободу дій і безпеку. PHP може працювати в так званому безпечному режимі (safe mode), який обмежує можливості застосування PHP користувачами по ряду важливих показників. Наприклад, можна обмежити максимальний час виконання і використання пам’яті (неконтрольований витрата пам’яті негативно впливає на швидкодію сервера). За аналогією з cgi-bin адміністратор також може встановлювати обмеження на каталоги, в яких користувач може переглядати і виконувати сценарії PHP, а також використовувати сценарії PHP для перегляду конфіденційної інформації на сервері.

У стандартний набір функцій PHP входить ряд надійних механізмів шифрування. PHP також сумісний з багатьма додатками незалежних фірм, що дозволяє легко інтегрувати його з захищеними технологіями електронної комерції (e-commerce). Інша перевага полягає в тому, що вихідний текст сценаріїв PHP можна переглянути в браузері, оскільки сценарій компілюється до його відправки за запитом користувача. Реалізація PHP на стороні сервера

запобігає викрадення нетривіальних сценаріїв користувачами, знань яких вистачає хоча б для виконання команди View Source.

Оскільки PHP є вбудовуваною мовою, він відрізняється винятковою гнучкістю по відношенню до потреб розробника. Хоча PHP зазвичай рекомендується використовувати в поєднанні з HTML, він з таким же успіхом інтегрується і в JavaScript, WML, XML та інші мови. Крім того, добре структуровані додатки PHP легко розширюються в міру необхідності (втім, це відноситься до всіх основних мов програмування).

Немає проблем і з залежністю від браузерів, оскільки перед відправкою клієнту сценарії PHP повністю компілюються на стороні сервера. По суті, сценарії PHP можуть передаватися будь-яких пристроїв з браузерами, включаючи стільникові телефони, електронні записники, пейджери і портативні комп'ютери, не кажучи вже про традиційні ПК. Програмісти, що займаються допоміжними утилітами, можуть запускати PHP в режимі командного рядка.

Оскільки PHP не містить коду, орієнтованого на конкретний web-сервер, користувачі не обмежуються певними серверами (можливо, незнайомими для них). Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold і Zeus - PHP працює на всіх перерахованих серверах.

Стратегія Open Source, і розповсюдження початкових текстів програм в масах, зробило безсумнівно благотворний вплив на багато проектів, в першу чергу - Linux, хоча і успіх проекту Apache сильно підкріпив позиції прихильників Open Source. Сказане стосується і до історії створення PHP, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником у розвитку проекту PHP. Ухвалення стратегії Open Source і безкоштовне розповсюдження початкових текстів PHP надало неоціненну послугу користувачам. До того ж, чуйне співтовариство користувачів PHP є свого роду «колективної службою підтримки», і в популярних електронних конференціях можна знайти відповіді навіть на найскладніші питання.

Для розробки веб-додатку буде використовуватися фреймворк Phalcon. Phalcon - PHP фреймворк з відкритим вихідним кодом, написаний на Сі. В даний

момент підтримується версія переписана на Zephir, має високу продуктивність і вживає мало ресурсів. Всі частини Phalcon мало залежні один від одного в процесі створення додатків.

Phalcon має наступні характеристики:

– всі компоненти повністю написані на мові програмування Сі;

– існують версії для різних популярних операційних систем;

– висока швидкість роботи, малі витрати серверних ресурсів;

– компоненти фреймворка слабо пов’язані між собою;

– взаємодія з базами даних реалізовано на Сі за технологією ORM.

Головні можливості фреймворка:

– можна користуватися власною базою та її окремими елементами;

– накладні витрати в додатках мінімізуються за рахунок низкоуровневой організації;

– за технологіями ORM відбувається взаємодія з базами даних, що в результаті дає дуже велику продуктивність;

– всі процеси відбуваються досить швидко, завдяки прямому зверненню фреймворка до внутрішніх структур PHP.

Phalcon за своєю суттю цікавий фреймворк, так як він встановлюється як окремий модуль PHP. Даний процес займає всього кілька хвилин.

Переваги Phalcon - це, перш за все, висока продуктивність і невелике навантаження файлової системи; при суворої тіпізірованія спостерігається меншою витрата пам’яті; часткова обробка інформації без інтерпретації..

Веб-додаток програмної системи повинен забезпечувати модульну взаємодію елементів інтерфейсу та контролерів, що структурують та відправляють запити до серверної часини. Для даного веб-додатка було складено ряд вимог:

а) простий описовий синтаксис;

б) легкість переходу між версіями;

в) документованість.

Наглядне розподілення наданих вимог і варіантів клієнтських рішень для даної системи відображене у таблиці 2.2, де кожен стовпець відображає вимоги до клієнтського рішення відповідно до наданого списку вимог, та кожен рядок відповідає за фреймворки та платформи для створення клієнтського рішення.

Таблиця 2.2 – Відповідність фреймворків та платформ до клієнтських вимог

	А	Б	В	Г
JS + Jquery	+	+	+	-
Razor	+	+	-	+

Результатом дослідження фреймворків та платформ для створення клієнтського рішення, відображеного в таблиці 2.2, є твердження, що для розробки веб-додатку для даної програмної системи найбільш підходить Jquery

jQuery - це фреймворк, бібліотека, створена для взаємодії мови програмування JavaScript і мови гіпертекстової розмітки HTML.

jQuery - це написана на JavaScript бібліотека, заснована на взаємодії мультіпарадігменної мови програмування з HTML документами і файлами XML відповідного вмісту.

Основні можливості:

– звернення до будь-яких елементів DOM для їх зміни і обробки;

– величезна бібліотека плагінів, особливо що стосуються візуального оформлення сторінок;

– підтримка роботи з подіями.

Завдяки jQuery відпадає необхідність піклуватися про синтаксис і особливості взаємодії різних браузерів і операційних систем з кодом на увазі ідентичною реалізації роботи фреймворку у всіх середовищах.

Швидкодія фреймворку досягається завдяки використанню селекторів - механізму швидкого звернення до будь-якого об’єкту HTML-документа.

База даних даного програмного продукту повинна відповідати ряду вимог, пов’язаних з функціональними особливостями різних СКБД та зберігаємих в ній структур даних.

Відповідно до цього, для бази даних було складено ряд вимог:

– простота розгортання;

– розширюваність;

– підтримка бази даних розробниками системи;

– статистика по базі даних;

– простота побудови запитів.

Було відібрано декілька кандидатів, а саме три серед яких MySQL, MongoDB та RethinkDB. Було обрані як реляційні так і не реляційні бази зважаючи на досвід використання обох підходів.

Наглядне розподілення наданих вимог і варіантів баз даних для даної системи відображене у таблиці 2.3, де кожен стовпець відображає вимоги до бази даних відповідно до наданого списку вимог, та кожен рядок відповідає за базу даних для зберігання клієнтських даних, що обробляються сервером.

Результатом дослідження баз даних для зберігання клієнтських даних, відображеного в таблиці 2.3, є твердження, що для розробки веб-додатку для даної програмної системи найбільш підходить MySQL.

Таблиця 2.3 – Відповідність баз даних до наданих вимог

	А	Б	В	Г	Д
MySQL	+	+	+	+	+
MongoDB	+	-	+	+	-
RethinkDB	+	+	+	-	-

MySQL - вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам.

Зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. MySQLнадає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання даних.

MySQL – характеризується великою швидкістю, стійкістю і простотою використання, була розроблена для підвищення швидкодії обробки великих баз даних. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатоканальності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість рядків у таблицях може досягати 50 млн.;

– висока швидкість виконання команд;

– наявність простої і ефективної системи безпеки.

Серед основних переваг MySQL відзначають наступні:

– масштабованість;

– переносність;

– зв’язаність;

– безпека;

– швидкість функціонування;

– зручність експлуатації;

– відкритий код.

MySQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в Yahoo!, Google,HP, Associated Press. Згідно документації, що додається до MySQL, деякі БД, що використовуються компанією MySQL AB (розробником MySQL), зберігають до 50 млн. записів. MySQL працює на різних платформах, серед яких Unix, Linux, Windows, OS/2,Solaris, Mac OC. Окрім того, MySQL працює на різних платформах.

MySQL має мережеву структуру. До MySQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MySQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API), які дозволяють встановлювати з’єднання з MySQL із додатків, написаних на таких

мовах як C, C++, Perl, PHP, Java, Python. MySQL має систему контроля доступу до даних, забезпечує шифрування даних при передаванні. MySQL досить зручно встановлюється та реалізується, легко адмініструється.

Для зручної роботи з MySQL буде використовуватись Workbench. MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення та експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL.

Можливості MySQL Workbench:

– дозволяє наочно представити модель бази даних в графічному вигляді;

– наочний і функціональний механізм установки зв’язків між таблицями, в тому числі «багато до багатьох» із створенням таблиці зв’язків;

– відновлення структури таблиць з вже існуючої на сервері БД (зв’язки відновлюються в InnoDB, при використанні MyISAM зв’язки необхідно встановлювати вручну);

– зручний редактор SQL запитів, що дозволяє відразу ж відправляти їх серверові і отримати відповідь у вигляді таблиці;

– дожливість редагування даних у таблиці в візуальному режимі.

Для розробки даного програмного продукту було перевірено декілька середовищ розробки. Було поставлено задачу знайти середовище розробки, відповідне до ряду вимог, пов’язаних з функціональними особливостями даних середовищ. Відповідно до цього, для середовищ розробки було складено ряд вимог:

а) простота та швидкість;

б) велика кількість додаткових розширюючи модулів;

в) документованість;

г) розпізнавання різних форматів файлів та синтаксису.

Наглядне розподілення наданих вимог і варіантів середовищ розробки для даної системи відображене у таблиці 2.4, де кожен стовпець відображає вимоги до середовища розробки відповідно до наданого списку вимог, та кожен рядок відповідає за середовище розробки.

Таблиця 2.4 – Відповідність середовищ розробки до наданих вимог

	А	Б	В	Г
PhpStorm	+	+	+	+
Sublime Text	+	+	+	+
WebStorm	-	-	+	+

Результатом дослідження середовищ розробки, відображеного в табл. 2.4, є твердження, що для вибору середовища розробки більш за все підходить PhpStorm. Основною перевагою даного середовища розробки э легкість та простота. Для завантаження необхідних модулів потрібно скористатися офіційним сайтом даного середовища розробки.

Також під час проектування виявилось, що у кінцевому продукті потрібно буде робити операції у певний час, саме тому було потрібно обрати програмний засіб виконувати певні операції за часом. Для цих операцій було обрано Cron.

Cron - класичний демон (комп'ютерна програма в системах класу UNIX), що використовується для періодичного виконання завдань в певний час. Регулярні дії описуються інструкціями, поміщеними в файли crontab і в спеціальні каталоги. Cron застосовують для автоматизації виконання поставлених завдань на сервері. Він працює на Linux і відмінно підходить для того, щоб запустити програму або скрипт на певний час і з певною періодичністю, запланувавши це дію заздалегідь.

Єдине обмеження, яке може відноситься саме до Cron - це обмеження на періодичність запуску. Наприклад щоб не перевантажувати сервер, провайдер може дозволяти запуск Cron-завдань не частіше, ніж один раз на годину або півгодини. В іншому на ці завдання діють всі ті ж обмеження, що і на сервер в цілому (пам'ять ОЗУ, розмір файлу, час його виконання).

2.2 Розробка загальної структури багтрекінгової сиситеми

На початкових етапах створення необхідно мати уявлення про те, як працює організація, яку ми збираємося автоматизувати. Для опису і візуального

представлення необхідно побудувати модель. Саме для побудови функціональної моделі (або моделі процесів) і призначено програмний продукт BPwin.

Найбільш зручною мовою моделювання бізнес-процесів є IDEF0, запропонована більш 20 років тому Дугласом Россом.

Під моделлю в IDEF0 розуміють опис системи (текстове і графічне), яке має дати відповідь на деякі заздалегідь певні питання. Процес моделювання якої-небудь системи в IDEF0 починається з визначення контексту, тобто найбільш абстрактного рівня опису системи в цілому. У контекст входить визначення суб'єкта моделювання, цілі і точки зору на модель. Під суб'єктом розуміється сама система, при цьому необхідно точно встановити, що входить в систему, а що лежить за її межами, іншими словами, ми повинні визначити, що ми будемо в подальшому розглядати як компоненти системи, а що як зовнішній вплив. На визначення суб'єкта системи буде істотно впливати позиція, з якої розглядається система, і мета моделювання - питання, на які побудована модель повинна дати відповідь. IDEF0-модель передбачає наявність чітко сформульованої мети, єдиного суб'єкта моделювання і однієї точки зору (рис. 2.1).

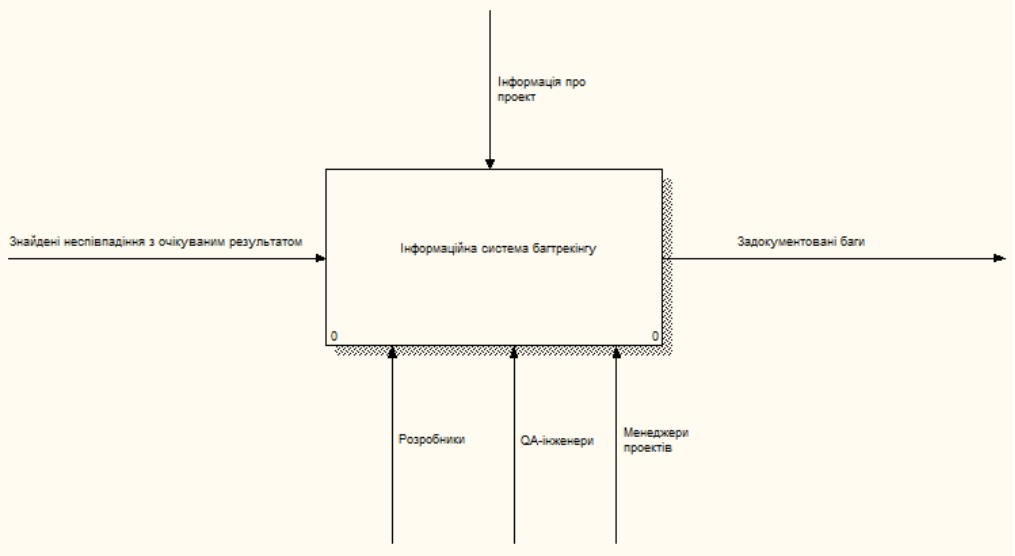

Рисунок 2.1 – Представлення ІС баг-трекінгу за допомогою IDEF0

Всі неспівпадіння будуть вноситися на основі інформації про проект QA-інженерами і назначатися на виконання розробникам. Все це буде відбуватися

під контролем менеджера проекту. Ну і в результаті ми отримаємо задокуменентований баг з усіма необхідними критеріями.

Далі проведемо декомпозицію моделі за допомогою методології DFD.

Діаграми потоків даних (DFD) являються основним засобом моделювання функціональних вимог майбутнього ПЗ. З їх допомогою ці вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, зв’язаної потоками даних. Основною метою таких засобів є демонстрація кожного процесу, який перетворює свої вхідні дані у вихідні, а також виявляє відносини між цими процесами. При використанні DFD моделі систему представляють як ієрархію діаграм потоків даних, що описують процес перетворення інформації з моменту введення в систему до видачі користувачеві і цей процес є асинхронним. На кожному наступному рівні ієрархій відбувається уточнення процесів, поки черговий процес не буде визнаний елементарним.

Моделі потоків даних були запропоновані спочатку Е. Йорданом (1975), а потім Ч. Гейном і Т. Сарсоном (1979). На таких моделях засновані класичні методології структурного аналізу і проектування програмного забезпечення. Така ж модель використовується в методології структурного аналізу і проектування SSADM (Structured Systems Analysis and Design Method) прийнятою у Великобританії як національний стандарт розробки ІС.

Основою моделі є поняття зовнішньої сутності, процесу, сховища (накопичувача) даних і потоку даних.

Зовнішня сутність – це матеріальний об’єкт або фізична особа, що виступають в якості приймачів інформації або її джерел, наприклад, замовники, персонал, постачальники, клієнти банк і тому подібне.

Процес – перетворення вхідних потоків даних у вихідні відповідно до певного алгоритму. Кожен процес в системі має свого виконавця, який здійснює дане перетворення, і свій особистий номер, який пов’язаний з виконавцем. Так само як у разі функціональних діаграм фізично перетворення може здійснюватися комп’ютерами, уручну або спеціальними пристроями. На верхніх рівнях ієрархії, доки процеси ще не є визначеними, замість поняття «процес»

використовують поняття «система» і «підсистема», які позначають систему в цілому або її функціонально закінчену частину відповідно.

Сховище даних – абстрактний пристрій який призначено для зберігання інформації. Деталізувати тип пристрою і способи розміщення, вилучення і зберігання для такого пристрою не потрібно. Фізично це може бути база даних, файл, таблиця в оперативній пам’яті, картотека на папері і тому подібне.

Потік даних – процес передачі інформації від джерела до приймача. Фізично процес передачі інформації може відбуватися вручну за участю пристроїв або людей зовні проектованої системи або по кабелях під управлінням програми або програмної системи.

Таким чином, діаграма демонструє як потоки даних, породжені деякими зовнішніми сутностями, трансформуються відповідними процесами (або підсистемами), зберігаються накопичувачами даних і передаються іншій зовнішній суті – приймачам інформації. В результаті ми отримуємо мережеву модель зберігання/обробки інформації (рис. 2.2).

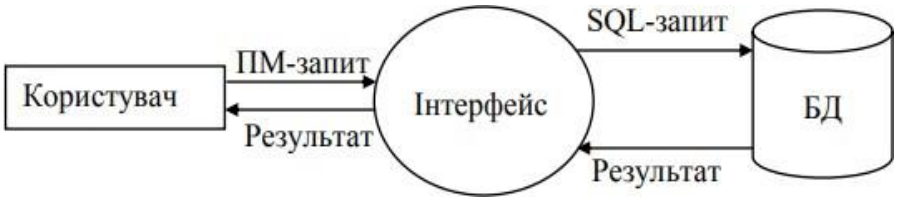

Рисунок 2.2 – Загальна схема потоків даних

Знайдені неспівпадіння тестувальник спочатку вводить в відповідну форму з урахуванням інформації про проект (специфікацій). З форми введені дані потрапляють до БД знайдених багів. З БД всі скомпоновані дані виводяться на сайт за запитом будь-кого з користувачів (тестувальника, розробника чи менеджера проекту). І відповідно ці скомпоновані дані являють собою задокументовані баги, які ми отримуємо на виході. Також слід зазначити, що існує можливість доповнення знайдених невідповідностей після виправлень розробника, адже не завжди баги виправляються с першого разу.

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ БАГ-ТРЕКІНГУ

3.1 Постановка задачі

Багтрекінгова система менеджменту проектів повинна надавати простий доступ до ведення проектів. Найкращий та найпростіший доступ – це використання браузера, тобто система у вигляді веб-застосунку. Функціонал, який надається системою, повинен бути чітким і зрозумілим, при цьому покриваючи основні потреби користувачів в системі, а саме:

– користувач може реєструватися у системі та входити у неї за рареєстрованими електронною адресою та паролем;

– користувач може проглядати детальну інформацію о своїх проектах у вигляді дашборду (інформація по проекту, учасники, статистичну інформацію о відкритих/теперешніх/закінчених/закритих завданнях, бачити завдання, які він має виконати, мати можливість переходити та бачити списки завдань за категоріями та завдання самі);

– користувач може створювати нові проекти;

– користувач може редагувати детальну інформацію по своїм проектам, які знаходяться на дашборді та архівувати їх;

– користувач може шукати проекти за неповним ім’ям та проглядати списки завдань за знайденними проектами;

– користувач може створювати, редагувати завдання та змінювати їх статус та детальну інформацію, якщо він належить до проекту, у якому містяться ці завдання;

– користувач може проглядати детальну інформацію по завданням, залишати коментарі та причипляти документи, щоб мати змогу обговорювати завдання, ділитися думками та усією потрібною інформацією;

– користувач може редагувати/вдаляти тільки свої коментарі та вдаляти причеплині документи.

3.2 UML-моделювання

Моделювання розроблюваної системи проводиться з використанням мови UML для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи. UML – мова графічного опису для об’єктного моделювання в галузі розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи. Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки додатків. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Нижче представлена діаграма варіантів використання (рис. 3.1), яка відображає можливості користувача в системі.

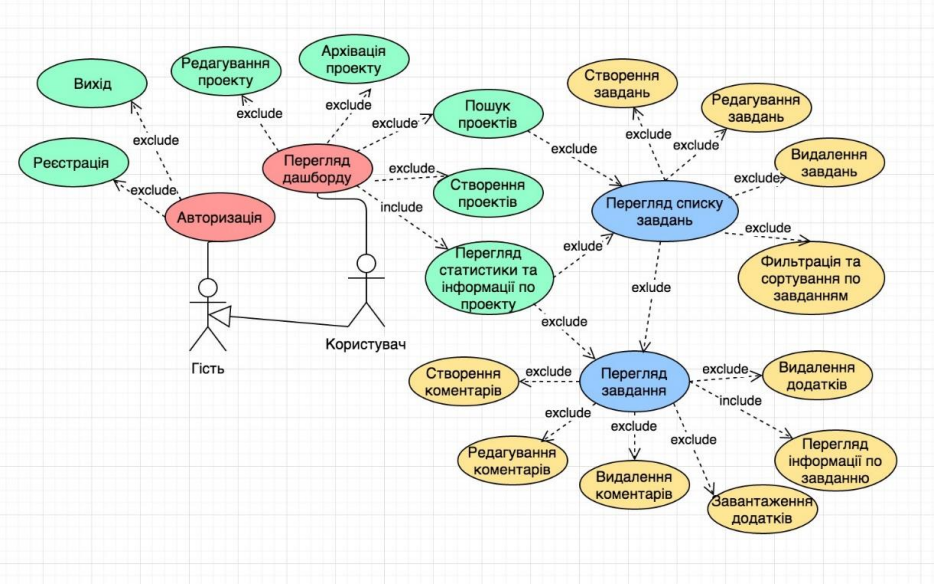

Рисунок 3.1 – UML діаграма

На ній зображено різні рівні деталізації – Cloud level, Kitlevel, Sea level та Fish level. Кожен рівень деталізації помічен кольором. Cloud level – найвищий рівень абстракції, Fish level – найнижчий рівень деталізації.

3.3 Проектування бази даних

Для зберігання даних сервісу було обрано NoSQL СКБД MongoDB у якості бази даних. При проектуванні бази даних було використано Robomongo – візуальний інтерфейс доступу до MongoDB.

Основними сутностями у системі, що розробляється, є сутності «Користувач», Проект» та «Завдання». Незважаючи на те, що стутність «Коментар» є окремою єдиницею, він не зберігається у базі даних окремою таблицею, він зберігається напряму у сутності

«Завдання», що полегшую їх показ. Необхідно визначити їх атрибути та зв’язки між ними.

У ході концептуального моделювання було встановлено, що «Користувач» має наступні атрибути:

– унікальний номер користувача в системі – ідентифікатор користувача;
– персональні дані користувача – повне ім’я;
– адреса електронної пошти;
– пароль;
– список проектів, у якому бере участь користувач;
– поля для відстеження успішної регестрації та сесії користувача у системі. Сутність «Проект» повинна мати такі атрибути:
– унікальний номер проекту у системі – ідентифікатор проекту;
– дата створення;
– назва;
– опис;
– флажок, який вказує архівований проект чи ні.

Основна сутність «Завдання» повинна мати такі атрибути:

– унікальний номер завдання у системі – ідентифікатор завдання;
– назва;
– версія;
– пріорітет завдання;
– статус завдання;
– категорія завдання;
– опис;
– проект, якому належить завдання;
– творець завдання;
– уповноважений виконати завдання;
– список коментарів.

Сутність «Коментар» повинна мати такі атрибути:

– унікальний номер коментара у системі – ідентифікатор коментара;
– коментар;
– відправник коментара.

Використання NoSQL бази даних вирішило проблему залежностей «один к багатьом/багато к одному», оскільки структура зберігання відрізняється від SQL баз даних і дозволяє збергіти дані у виді списків (у даному випадку JSON документів) та дозволяє які завгодно рівні вкладеності, тобто можливо зберігати сутності у сутностях. Це спрощує проектування бази даних, зберігання даних та керування ними у подальшому використанні бази даних.

Нижче наведено рисунок 3.2 зі схемою бази даних, на якій вказані усі таблиці, що використовуються, а також типи даних атрибутів. Оскільки використовується NoSQL база даних, вирішено використовувати діаграму класів як ER-діаграму. Вона дозволяє виявити та наглядно продемонструвати залежності між сутностями-класами. На схемі зображені всі атрибути сутностей, котрі можуть бути. Лініями зв’язків показані відношення між таблицями. Цілісність даних забезпечується первинними ключами сутностей. Для оптимізації запитів було додано індекси до деяких полів.

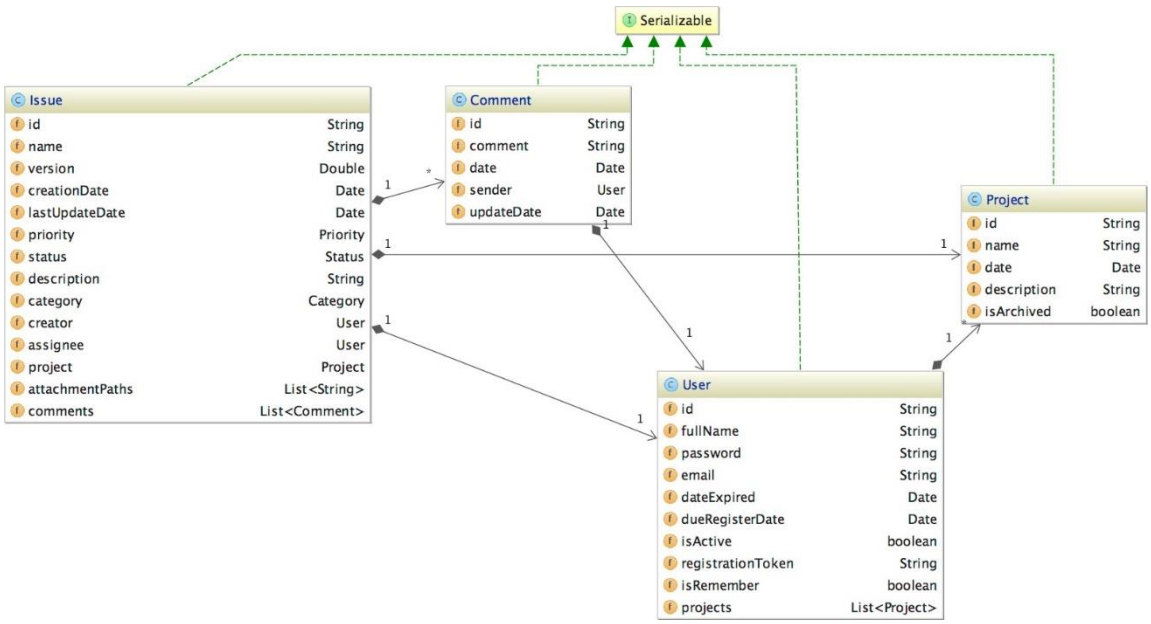

Рисунок 3.2 – Схема бази даних

4 ОПИС РОБОТИ ВЕБ-ЗАСТОСУНКУ

Система має англомовний інтерфейс. Користувач починає свій шлях з основної сторінки, де вводить свої дані для авторизації і після підтвердження адміністратором авторизується в системі (рис. 4.1).

Рисунок 4.1 – Головна сторінка

На некоректну поведінку користувача система реагує повідомленнями червоним кольором та підсвічує проблемні поля (рис. 4.2).

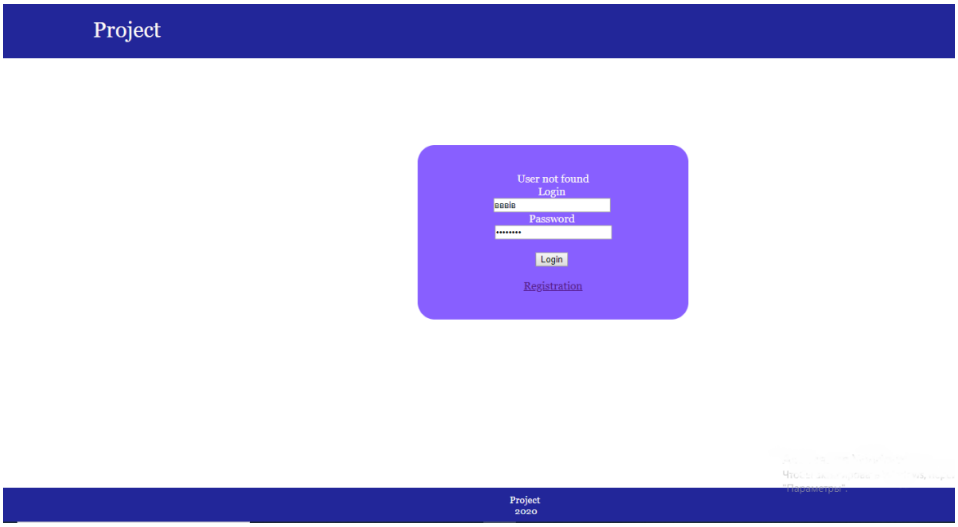

Рисунок 4.2 – Некотректна поведінка

Нажавши кнопку Register (Зареєструватися), можно попасти на строрінку реєстрації (рис. 4.3). Після реєстрації на вказану електронну адресу приходить письмо з посиланням.Після проходження за цим посиланням регістрація завершуеться та користувач заноситься у систему.

Після входу у систему з’являється дашборд (рис. 4.4), на якому відображуються усі проекту користувача.

Рисунок 4.3 – Сторінка реєстрації

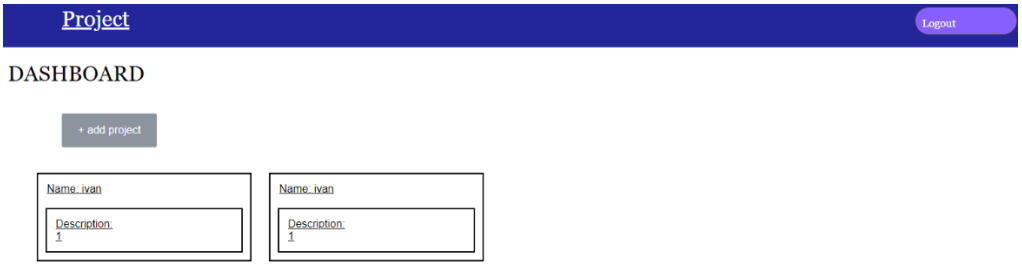

Рисунок 4.4 – Дашборд користувача

Користувач має право редагувати та архівувати свої проекти. На всіх екранах додатку є навігаційне меню. У ньому розташовані кнопка створення проекту, поле для пошуку проекту та кнопка виходу із системи (по нажаттю на своє ім’я).

На кожному дашборді є кнопки створення завдання (ticket), кнопка редагування проекта та кнопка архівації проекта. Також на панелі розташовані чотири прямокутники, на яких виводяться статистичні числа, які показують скількі завдань на проекті відкрито/у прогресі/виконано/зроблено/закрито. Під кожним прямокутником є лінк (View details), який веде на сторінку із завданнями по категоріям. У прямокутники Participants зобрані учасники проекту; показують поточні завдання користувача з відповідними посиланнями, які ведуть на сторінки з детальною інформацією по кожному завданню; Description виводить інформацію по проекту.

Нижче показана сторінка із завданнями по проекту. Людина, розташована у проекті, може створювати, редагувати та вдаляти завдання. У таблиці із завданнями працює фільтрація по усім полям, пошук та сортування. Нажавши поле, розташований на кожному рядку, можна побачити інформацію більш детально (рис. 4.5).

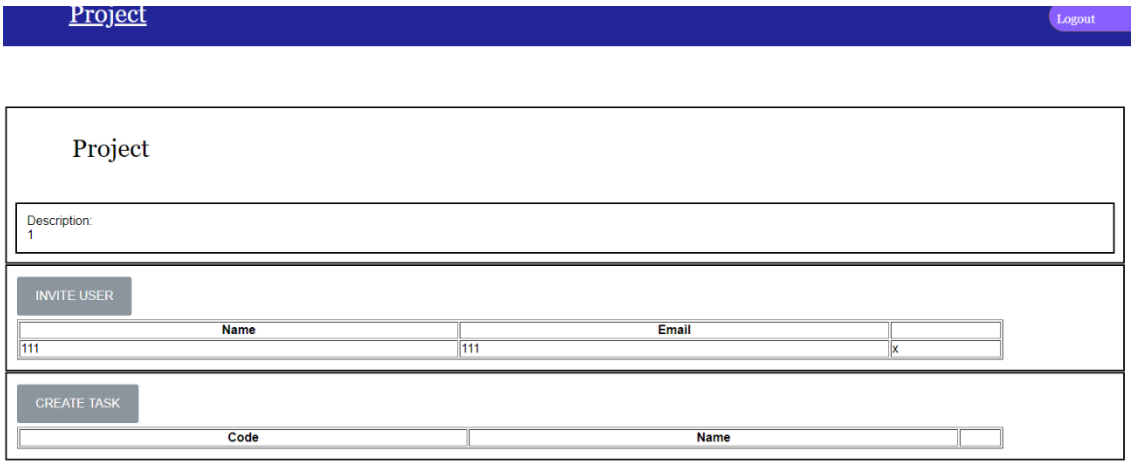

Рисунок 4.5 – Список завдань та детальна інформація

Усі редагування та видалення у системі (крім коментарів) працюють через модальні вікна (рис. 4.7). Приклад редагування проекту розташований на рисунку. Назначати завдання можливо тільки користувачам, які приймають участь у проекті. У графі, де потрібно назначати користувачів, працює пошук, аналогічний до пошуку проектів.

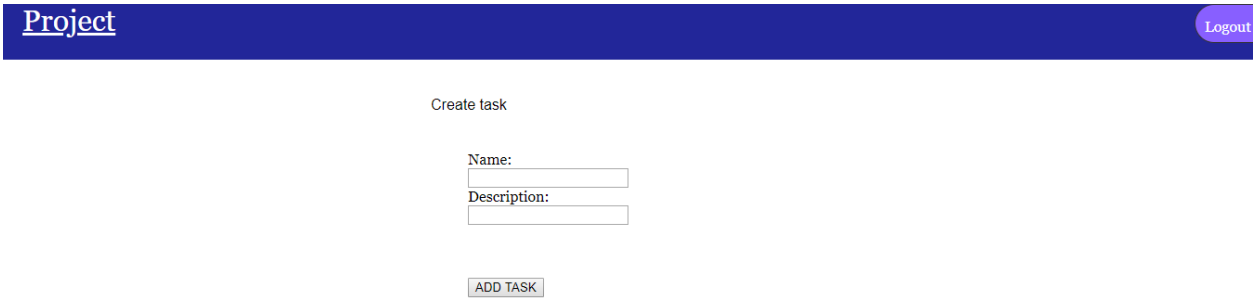

Рисунок 4.7 – Створення та Редагування завдання

Коментарі може вдалити чи редагувати тільки власник. Причеплені документи можуть вдаляти які-завгодно користувачі (рис. 4.8). Коментарі працюють у режимі реального часу, тобто результати додавання, редагування чи виделення коментарів одразу відображаються у відповідній формі, сторінку не треба перезавантажувати.

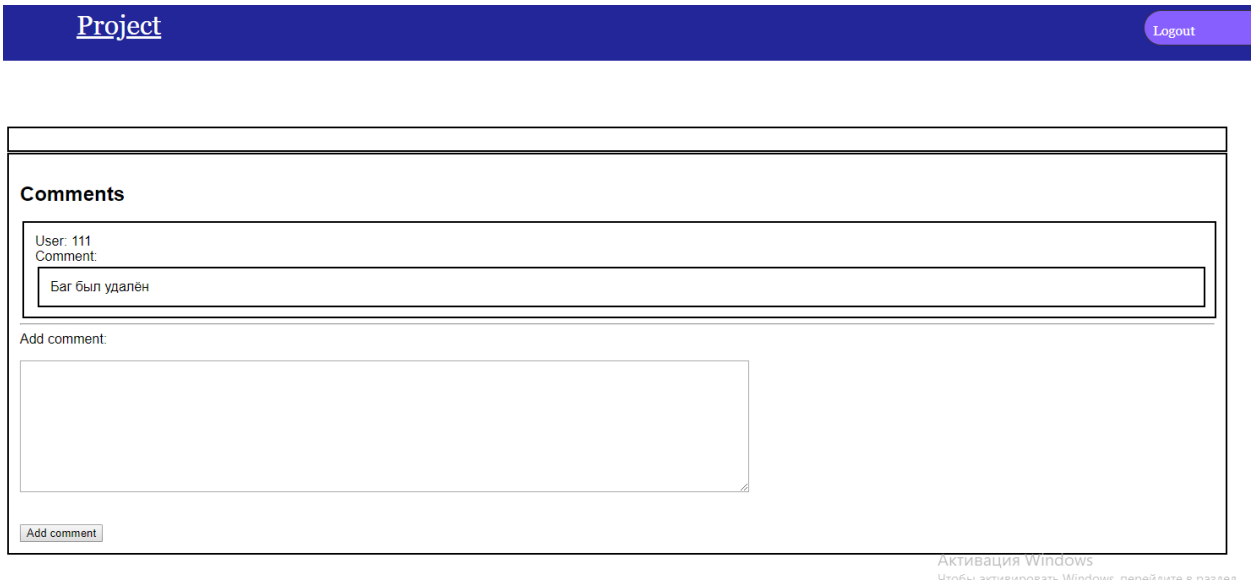

Рисунок 4.8 – Сторінка коментарів

5 ТЕСТУВАННЯ

Тестування програмного забезпечення є процесом технічного дослідження, призначеного для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає у тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів, а також покриття тестами основної частини функціоналу системи. Як результат програмне забезпечення тестується стандартним виконанням програми з метою виявлення дефектів, тому для тестування системи, що розроблюється у даній роботі, було обрано метод мануального, тобто ручного тестування.

У розроблюваній системі є дві частини, що потребують тестування – серверна частина та клієнтська. Бізнес-логіка серверної частини тестувалася шляхом перевірки JSON-відповідей на певні JSON-запити з використанням розширення для браузера Google Chrome – Postman.

На стороні веб-клієнту проводилося тестування інтерфейсу користувача для виявлення помилок у функціональності за допомогою інтерфейсу, необроблених виключеннях при взаємодії з інтерфейсом, втрати або перекручення даних, передачу даних через елементи інтерфейсу, а також помилок в інтерфейсі, таких як відсутність елементів.

Більшість тестів, проведених з системою, виконувалися у якості димового тестування (Smoke testing) – явного тестування на помилки та працездатність модулів. Поняття димового тестування пішло з інженерного середовища: «При введенні в експлуатацію нового обладнання вважалося, що тестування пройшло вдало, якщо з установки не пішов дим».

6 ЕКОНОМІЧНА ЧАСТИНА

6.1 Характеристика науково-дослідної роботи

В економічній частині атестаційної роботи обґрунтовано доцільність використання багтрекінгової системи менеджменту проектів.

В атестаційній роботі проведено дослідження стану автоматизації процесу документування знайдених невідповідностей з очікуваним результатом.

6.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

У процесі виконання науково-дослідної роботи був проведений огляд існуючих засобів для оцінки сайту, на основі аналізу спеціальної літератури розглянуті принципи роботи багтрекінгу, на яких будується відстеження історії звітів про баги та принцип формування критеріїв для їх оцінки. Після чого проведено експеримент та дослідження стану автоматизації процесу документування. У результаті оцінено ефективність ІС багтрекінгу.

Розділимо науково-дослідну роботу (НДР) на етапи: початковий, основний і заключний.

На стадії виконання підготовчого етапу були виконані підбір і аналіз інформації для проведення відповідних до постановки задачі робіт. Проведено пошук інформації в Internet та у відповідній літературі.

На етапі виконання основної частини НДР були виконані такі роботи:

– визначення основних засад в розробці ІС багтрекінгу;

– дослідження стану автоматизації документації;

– дослідження структурних особливостей системи;

– вибір існуючих інструментів розробки ІС;

– тестування та вибір методології створення багтрекінгової системи;

– доведення роботи створеної методики автоматизації процесу документування.

У заключній частині здійснюється оцінка ефективності виконання НДР, складання звіту по НДР, захист звіту.

Для виконання НДР було сформовано команду з 16 осіб. У команду ввійшли:

– маркетолог – 1 особа, заробітна плата 10 000 грн./міс.;

– керівник юзабіліті-тестування – 1 особа, заробітна плата 15 000 грн./міс.;

– керівник фокус-групи – 1 особа, заробітна плата 15 000 грн./міс.;

– учасники фокус-групи – 5 осіб, заробітна плата 9 000 грн./міс.;

– учасники юзабіліті-тестування – 5 осіб, заробітна плата 9 000 грн./міс.;

– веб-дизайнер – 1 особа, заробітна платня 17 000 грн./міс.;

– SEO-спеціаліст – 1 особа, заробітна плата 22 000 грн./міс.;

– адміністратор проекту – 1 особа, заробітна плата 18 000 грн./міс.;

Розрахуємо трудові витрати та заробітну плату виконавців.

Середня заробітна плата за робочий день розраховується за формулою:

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \quad (6.1)$$

де $З_{ср.міс.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n = 21).

Етапи виконання НДР, перелік та зміст робіт, трудомісткість їх виконання, заробітна плата працівника представлені в табл. 6.1.

Таблиця 6.1 – Розрахунок трудовитрат і заробітної плати працівника

Перелік робіт	Кіль-кість вико-навців	Посада виконавця	Трудоміст-кість робіт, люд.-днів	Середньо-денна заробітна плата, грн.	Сума заробітної плати, грн.
1	2	3	4	5	6
1. Початковий етап					
1.1. Розробка та затвердження ТЗ	1	Адміністратор проекту	3	857,14	2571,42
1.1.2.Досліди цільової аудиторії та аналіз конкурентів	1	Маркетолог	1	476,19	476,19
1.1.3.Аналіз ризиків	1	Маркетолог	1	476,19	476,19

Продовження таблиці 6.1

1	2	3	4	5	6
1.3. Підготовка довідкових матеріалів та даних для виконання НДР	1	Адміністратор проекту	3	857,14	2571,42
2. Основний етап					
2.1 Постановка задачі	1	Адміністратор проекту	3	857,14	2571,42
2.2 Проведення опитування (фокус-група)	5	Учасник фокус-групи	2	428,57	857,14
2.3 Обробка результатів фокус-групи	1	Керівник фокус-групи	1	714,28	714,28
2.4 Розробка дизайну	1	Веб-дизайнер	10	809,52	8095,2
2.5 Розробка системи	1	SEO-спеціаліст	8	1047,61	8381,36
2.5.1.Розробка БД	1	SEO-спеціаліст	6	1047,61	6286,02
2.5.Тестування системи	1	Керівник юзабіліті-тестування	1	714,28	714,28
2.5.1 Група юзабіліті-тестування	5	Учасники юзабіліті-тестування	1	428,57	428,57
2.6 Обробка результатів екперементу	1	Адміністратор проекту	1	857,14	857,14
3. Заключна частина					
3.1 Оцінка ефективності виконання НДР	1	Адміністратор проекту	1	857,14	857,14
3.2 Складання звіту по НДР	1	Адміністратор проекту	5	857,14	4285
3.3 Захист звіту	1	Адміністратор проекту	1	857,14	857,14
Усього			48		40999,89

6.3 Розрахунок одноразових витрат на розробку НДР

Розрахунок собівартості відбувається відповідно до існуючих нормативних актів України. До розрахунків входять такі статті витрат:

– витрати на оплату праці;

– єдиний соціальний внесок;

– матеріальні витрати;

– амортизація основних засобів (вартість машинного часу);

– витрати на спожиту електроенергію;

– комунальні витрати.

Матеріальні витрати – це вартість матеріалів, які необхідні для виконання роботи з урахуванням цін, що діють на момент розрахунку.

Матеріальні витрати розраховуються за такою формулою:

$$M = \sum_{j=1}^{n} Q_j \times Ц_j, \qquad (6.2)$$

де M – сумарні витрати на матеріали, в тому числі малоцінні предмети, що швидко зношуються (носії, папір, канцелярське приладдя тощо);

Q_j – кількість використаних одиниць j-го виду матеріалів, $j = (1 \div n)$;

$Ц_j$ – ціна одиниці j-го виду матеріалів.

Розрахунок матеріальних витрат представлено в таблиці 6.2.

Таблиця 6.2 – Розрахунок матеріальних витрат

Найменування	Од. вим.	Кількість од.	Ціна, грн	Сума, грн.
Олівець	шт.	3	11,00	33,00
Ручки	шт.	5	25,00	100,00
Папір	уп.	2	115,00	230,00
Степлер	шт.	1	30,00	30,00
Скріпки для степлеру	уп.	2	7,00	14,00
Усього				407

Витрати на оплату праці працівників формуються виходячи з того скільки часу їм необхідно на виконання роботи та середньомісячної заробітної плати. Відповідно до проведених розрахунків витрати на оплату праці виконавців складають 40999,89 грн.

Єдиний соціальний внесок на загальнодержавне страхування (ЄСВ) – обов’язковий платіж до системи загальнообов’язкового державного соціального страхування, що справляється в Україні з метою забезпечення страхових виплат за поточними видами загальнообов’язкового державного соціального страхування. Для об’єкта дослідження ставка єдиного соціального внеску дорівнює 22% від витрат на оплату праці. В нашому випадку розмір ЄСВ дорівнює 9019.9 грн. При виконанні НДР застосовувалось наступне обладнання: 6 робочих комп’ютерів вартістю 12000 грн.

Вищенаведене устаткування є власністю виконавців, тому доцільно розрахувати суму амортизаційних відрахувань на період виконання НДР:

$$AB = \sum_{k=1}^{L} \frac{BO_k}{TE_k} \times T\,, \tag{6.3}$$

де AB – сума амортизаційних відрахувань, нарахованих під час проведення науково-дослідницької роботи;

BO_k – вартість основних засобів k-го виду;

TE_k – термін експлуатації основних засобів k-го виду, днів;

T – термін науково-дослідницької роботи, днів;

L – кількість видів обладнання.

Витрати на використану обладнанням електроенергію розраховуються:

$$З_е = M \cdot t \cdot T_{кВт}, \tag{6.4}$$

де M – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$T_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

Споживна потужність комп’ютера складає 0,75 кВт за годину. Тариф споживачів за першою категорією (тобто 100 кВт та більше) складає 2,1 грн./кВт за годину (з ПДВ).

До комунальних витрат відносяться такі статті:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20% від витрат на оплату праці;

– вартість оплати послуг Інтернету становитиме – із розрахунку 7 грн. за день. Всього 336 грн. за 48 днів виконання НДР.

Для виконання НДР використовувалося безкоштовне ПЗ для розробки документації – Libra Office, яка знаходиться в безкоштовному доступі. Результати розрахунку кошторису витрат на виконання НДР наведені в табл. 6.3.

Таблиця 6.3 − Кошторис витрат на розробку НДР

№ з/п	Стаття витрат	Сума, грн.
1	Заробітна плата	40999,89
2	Єдиний соціальний внесок	9019.9
3	Матеріальні витрати	407
4	Амортизація основних засобів	3000
5	Витрати на спожиту електроенергію	1156,05
6	Адміністративні витрати	8199,97
7	Вартість послуг Інтернету	336
	Усього витрати	63117,91

Таким чином, кошторис витрат на виконання даної НДР відбиває сумарні витрати за статтями та складає 63117,91 грн.

6.4 Оцінка результатів науково-дослідної роботи

Результат – це наслідок послідовності дій виконаних при НДР, виражений якісно або кількісно. В загальному випадку оцінка результатів НДР – це визначення ефективності отриманих рішень порівняно з сучасним науково-технічним рівнем. У якості результату впровадження НДР є підвищення ефективності та створення методики автоматизації процесу документування.

Результат впровадження НДР визначається за такою формулою:

$$\Delta P_j = \left| Хб_j - Хн_j \right|, \tag{6.5}$$

де ΔP_j – покращення *j*-ої характеристики досліджуваного процесу за рахунок впровадження результатів НДР (*j=1*);

$Хб_j$ – базове значення *j*-ої характеристики;

$Хн_j$ – нове значення *j*-ої характеристики після впровадження НДР.

У якості досліджуваної характеристики виступає час, використаний на досягнення мети користувачем (викладення документації про стан дефекту). Виходячи з того, що розроблені прототип мають урізану функціональність форми, було вирішено отримувати час досягнення мети без урахування часу, використаного на введення текстової інформації в поля. Отримані результати тестування наведені у таблиці 6.4.

Таблиця 6.4 – Показник конверсії на сайті

	Існуючій сайт	Прототип 1
Час виконання завдання (хв.)	3,4	2,5

Прототип 1(створена багтрекінгова система) по результатам тестування виявився більш зручним та зрозумілим у користуванні, що дозволило скоротити час досягнення мети користувачем на 0,9 хвилини.

6.5 Визначення економічної ефективності результатів НДР

Для визначення економічної ефективності результатів НДР необхідно порівняти витрати на розробку НДР з отриманими результатами.

Основним показником економічної ефективності науково-дослідної роботи є коефіцієнт «ефект-витрати», який розраховується за формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \qquad (6.6)$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн.;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня

витрат НДР змінює *j*-ту характеристику досліджуваного процесу.

У результаті проведених досліджень, можна зробити висновок про те, що кожна гривня витрат на розробку НДР забезпечує зниження затрат часу на 0,00014 % швидше дізнатися та видалити дефект. Дана науково-дослідна робота має позитивний показник економічної ефективності. Роботу у цілому можна враховувати ефективною або такою, що має науковий та технічний рівень.

ВИСНОВКИ

Отже, багтрекінговою системою є організований набір елементів, що збирає, обробляє, передає, зберігає та надає дані. Інформаційна система складається з людей, обладнання, процесів, процедур, даних та операцій. Кожна інформаційна система включає в себе наступні компоненти:

– структура системи;
– функції кожного елемента системи;
– вхід і вихід кожного елементу і системи в цілому;
– мета і обмеження системи та її окремих елементів.

Забезпечення якості (Quality Assurance) являє собою сукупність заходів, охоплюючих абсолютно усі етапи розробки, випуску та експлуатації програмного забезпечення. Це активності на усіх етапах життєвого циклу ПЗ, які вживаються для забезпечення необхідного рівня якості випускаємого продукту.

Багтрекером є інформаційна системи, розроблена, щоб допомогти тестувальникам та програмістам відстежувати історію звітів про баги та взаємодіяти один з одним під час своєї роботи. Він може розглядатись як різновид системи відстеження проблем.

Оскільки, об’єктом дослідження являється галузь QA-інженерії, а однією з основних цілей тестування є виявлення дефектів та надання інформації для прийняття рішень, то предметом дослідження був конкретний процес документування знайдених дефектів. На основі цього було виконано основне завдання атестайійної роботи, а саме, розробка ІС баг-трекінгу та досягнуто мети роботи. Відповідно метою є сама система з її функціоналом, яка допоможе систематизувати процеси тестування ПЗ.

Завданням атестайійної роботи є:

– визначення основних засад у розробці багтрекінгових систем;
– визначення структурних особливостей системи;
– вивчення існуючих інструментів розробки ІС;
– дослідження існуючих багтрекерів-аналогів;

– формулювання ключових елементів системи;

– розробка загальної структури багтрекінгових систем менеджменту проектів;

– моделювання основних процедур за допомогою модулів конструктора ІС;

– опис послідовного алгоритму роботи багтрекера.

Тобто було створено повноцінний прототип, який наповнено всім необхідним функціоналом для документування знайдених невідповідностей з очікуваним результатом у програмах та сайтах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Луценко Є.С. Багтрекер, як інструмент контролю за процесом тестування // Збірник наукових праць студентів спеціальностей «Інформаційні управляючі системи і технології», «Комп'ютерний еколого-економічний моніторинг» / редкол.: В.С. Пономаренко [ті ін.]. Харків: ХНЕУ, 2011. 308 с.

2. Калбертсон Р., Браун К., Кобб Г. Швидке тестування. Вильямс, 2012. 374 с.

3. Louise Tamres, Introducing Software Testing. USA, 2012.

4. Judy McKay, Managing the Test People: A Guide to Practical Technical Management. USA, 2008.

5. Frank P. Ginac, Customer Oriented Software Quality Assurance. USA, 2012.

6. Michael S. Deutsch, Software quality engineering. USA, 2008.

7. H. Kan, Metrics and Models in Software Quality Engineering. USA, 1995.

8. Philip B. Crosby, Quality Is Free. USA, 1979.

9. John W. Horch, Practical guide to software quality management. USA, 2011.

10. Криспін Л., Грегорі Д., Гнучке тестування: практичне керівництво для тестувальників ПО і гнучких команд. Вільямс, 2010. 464 с.

11. Інформаційні системи та їх роль в управлінні економікою. Одеса, 2017. URL: https://www.uzhnu.edu.ua/uk/infocentre/get/6742 (дата звернення: 28.11.2020).

12. Луценко Є.С. Сучасні підходи до розробки програмного забезпечення // Збірник наукових праць студентів спеціальностей «Інформаційні управляючі системи і технології», «Комп'ютерний еколого- економічний моніторинг». Харків ХНЕУ, 2010. 342 с.

13. Ситник В. Основиінформаційних систем. Науковий посібник. Київ, 2015.

14. Cheng Hsu. Information Systems. The Connection of People and Resources for Innovation. USA New York, January 2013.

15. Терещенко Л.О., Матієнко-Зубенко І.І. Інформаційнісистеми і технології в обліку. К.: КНЕУ, 2004. 187 с.

16. Сініцин С.В., Налютин Н.Ю., Верифікація програмного Забезпечення. Біном, 2008. 368 с.

17. K.Rubin. Developing object-oriented software // IBM Object-Oriented. Technology Center, USA, 2014. 22-31 с.

18. Буйницька О.П. Інформаційні технології та технічні засоби навчання. Навч. посіб. – К.: Центр учбової літератури, 2012. 240 с.

19. Офіційний сайт Wordpres. URL: https://uk.wikipedia.org/wiki/WordPress (дата звернення: 28.11.2020).

20. Опис CMS Wordpress та оцінки користувачів. URL: http://wordpress. cmsmagazine.ru (дата звернення: 28.11.2020).

21. Дэн Рамел. Joomla! для профессионалов = Advanced Joomla!. М.: «Вильямс», 2014. 448 с.

22. Огляд та відгуки про конструктора сайтів Wix. URL: https://uguide.ru/ konstruktor-sajtov-wix-obzor-otzyvy-primery-sajtov (дата звернення: 28.11.2020).

23. Багтрекери, як вибрати кращий. URL: http://ru.qatestlab.com/ knowledge-center/qa-testing-materials/bug-trackers-what-to-choose/ (дата звернення: 28.11.2020).

24. Vikrant Bhateja, Bao Le Nguyen, Nhu Gia Nguyen, Suresh Chandra Satapathy, Dac-Nhuong Le. Information Systems Design and Intelligent Applications. Proceedings of Fourth International Conference. India, 2017.

25. Jay M. Gould Input/Output Databases. Uses in Business and Government. Chicago, April 2018.

26. Кумскова І. О. База даних. Навчальний посібник. Москва, 2016.

27. Бідюк П.І., Гожий О.П., Коршевнюк Л.О. Комп’ютерні системи підтримки прийняття рішень. Київ, 2010. 890 с.

28. Larry E. Wood, User Interface Design Bridging the Gap from User Requirements to Design. USA, May 2018.

29. Brad A. Myers, Languages for Developing User Interfaces. USA, 2011.

30. Бірман І. М. Інтерфейс користувача. Москва. 2013.

31. Duckett J. Web Design with HTML, CSS, JavaScript and JQuery Set. USA, 2017.

32. David Love, Tkinter GUI Programming by Example.

33. Learn to create modern GUIs using Tkinter by building real-world projects in Python. USA, April 2018.

34. Chris Aquino and Todd Gandee, Front-End Web Development: The Big Nerd Ranch Guide. USA, 2016.

35. Andreas Schwarz, Dave Thomas, David Heinemeier Hansson, James Duncan Davidson, Justin Gehtland, Leon Breedt, and Mike Clark, Agile Web Development with Rails. USA, 2010.

36. Створення моделі процесів у BPwin (IDEF0). URL: http:// www.interface.ru/public/caseall/caseall3.htm (дата звернення: 28.11.2020).

37. August-Wilhelm Scheer, Henrik von Scheel, and Mark von Rosing. The Complete Business Process Handbook. 1st Edition. Body of Knowledge from Process Modeling to BPM, Volume 1. USA, December 6 2014.

38. Matthias Weidlich, Mathias Weske. Business Process Modeling Notation: Second International Workshop, BPMN 2010, Potsdam. Germany, October 13-14, 2010 Proceedings.

39. Діаграми потоків даних – Студопедія. URL: https://studopedia.info/1-113597.html (дата звернення: 28.11.2020).

40. David Le Blanc, John Viega, and Michael Howard, 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them, USA, 2009.

41. Geetha Manjunath and Sitaram. Moving to the Cloud: Developing Apps in the New World of Cloud Computing. Canada, December 1, 2011.

42. Capers Jones, Olivier Bonsignour, The Economics of Software Quality. USA, July 14 2011.

43. Ільїн В. В., Управління ефективністю впровадження інформаційних систем. Росія, Москва, 2014

44. Філімонова Є. В., Інформаційні технології в професійній діяльності. Росія, Москва. 2016.

45. Черніков Б. В., Лексикологічний синтез документів в комплексах інформаційних систем. Росія Санкт-Петербург, 2016.