МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКАРЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Олійник Олександр Олександрович

Програмний засіб контролю робочого часу та виконання поставлених завдань / Software of the working time and tasks performance control

спеціальність: 6.050102 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерні системи та мережі

Випускна квалікаційна робота

Виконав: студент групи КСМ-42/1
Олійник Олександр Олександрович

Науковий керівник
Ігнатєв І.В.

ТЕРНОПІЛЬ 2019

РЕЗЮМЕ

Дипломна робота містить 68 сторінки, 15 таблиць, 27 рисунків, список використаних джерел із 29 найменувань та 3 додатки.

Метою дипломної роботи є розробка застосунку, який забезпечує моніторинг, комунікацію, актуалізацію даних при виконання IT проектів абсолютно безкоштовним з повним функціоналом та open source.

Об’єктом дослідження є процеси розробки програмного забезпечення з використанням таск-трекінгових систем.

Предметом дослідження є сучасні методи, засоби та технологій для розробки таск-трекінгових систем.

Методи розробки базуються на технології JAVA з використанням бази даних PostgreSQL.

Одержані результати полягають в розробці системи, яка надає можливість для розробника бачити повний список завдань,які потрібно виконати, а для менеджера - побачити на якій стадії знаходиться розробка проекту для інформування замовника.

Ключові слова: ТАСК-ТРЕКІНГОВА СИСТЕМА, ПРОЕКТ, СПРИНТ, КАРТОЧКА, USER STORY, БАГ.

RESUME

Thesis contains 68 pages, 25 tables, 27 figures, list of sources with 29 titles and 3 applications.

The aim of the thesis is the development of an application that provides monitoring, communication, updating of data while executing IT projects completely free with full functionality and open source.

Object of research are software development processes using task tracking systems.

The subject of research are modern methods, tools and technologies for developing traction tracking systems.

Development methods are based on JAVA technology using the PostgreSQL database.

The resulting is to develop a system that gives developers the opportunity to see a complete list of tasks that need to be done, and for the manager to see at what stage is the development of the project to inform the customer.

Keywords: TASK-TRACKING SYSTEM, PROJECT, SPRINT, CARD, USER STORY, BUG.

ЗМІСТ

ВСТУП

На сьогоднішній день IT-компанії створюють великі комерційні проекти, в яких задіяна величезна кількість людських ресурсів. Звичайно ж, що сам процес розробки ділиться між членами команди. Для того, щоб проектний менеджер міг ефективно керувати проектом, йому необхідно знати на якому етапі знаходиться процес розробки того чи іншого модуля додатку . Перед початком розробки проекту, його керівники планують роботу таким чином, щоб усі, хто задіяні у процесі розробки, працювали разом. Проте все ж є випадки, коли деякі учасники команди працюють віддалено. У такому випадку перед керівниками постає завдання організації робочого процесу таким чином, щоб усі працівники у разі потреби могли легко між собою комунікувати. Так, можна використовувати альтернативні джерела зв'язку такі як: телефон, соціальні мережі, чи скайп. Але є значно простіший та зручніший спосіб – використання task tracking систем. У такому випадку керівник зможе з легкістю стежити за виконанням роботи певним працівником чи команди у цілому.

Насамперед, призначення цих систем – це стеження за якістю продукту, адже більшість дефектів потрібно виявляти на етапі розробки, тим самим це дає можливість на етапі підтримки не виявити критичні баги. А наступною метою є – забезпечення комунікації для усіх членів команди.

Зазвичай системи, здатні забезпечувати такий функціонал – платні, та коштуватимуть такі немало. Компанії невеликого масштабу зазвичай не мають можливості придбати такого роду інструменти. Окрім них, у процесі розробки продукту є необхідність використовувати й інші, можливо і дорожчі застосунки.

Практично усі IT – компанії використовують такого роду програмні продукти.

Актуальною задачею є розробка застосунку Lazy_Track, який забезпечує моніторинг, комунікацію, актуалізацію даних при виконання IT проектів абсолютно безкоштовним з повним функціоналом та open source.

Щоб досягнути даної мети необхідно виконати наступні завдання:

– дослідити існуючі аналоги;

– спроектувати для розроблюваного продукту функціональні та нефункціональні вимоги;

– розробити модель бази даних.

Об’єктом дослідження є розробка та реалізація програмного продукту з застосуванням task tracking систем.

Предметом дослідження є метод, який використовує засіб та технологій для того, щоб розробити task tracking систему.

Для розробки системи скористаємось технологіями необхідними для розробки:

SQL – мова, за допомогою якої можна створювати та працювати з базами даних, котрі в свою чергу є зв’язаною інформацією, що зберігається в таблицях [22].

JAVA – це мова програмування для створення програм, що виконують різні важливі і другорядні функції. Необхідно відзначити, що JAVA – це не тільки сама мова, але і платформа, на якій можливо створювати і використовувати додатки, написані на JAVA [8].

Практичне значення такого продукту використовується для того, щоб мати можливість розробнику бачити весь список поставлених задач, які необхідно реалізувати, а менеджеру - контролювати стадію розробки продукту для доповіді замовнику.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ, АЛГОРИТМІВ ТА ПРОГРАМНИХ РІШЕНЬ

1.1 Дослідження предметної області

ІТ-компанії розробляють великі проекти, необхідністю для них є залучати висококваліфікованих спеціалістів. Завдання діляться між командою. Тому, керівнику необхідно відстежувати виконання проекту та у будь який момент. Роботу сплановують так, щоб уся команда працювала разом. Але інколи працівники працюють віддалено. Тоді виникає ряд проблем зв’язаних з комунікацією, з перевіркою виконаних завдань у віддалених працівників. Є багато варіантів: соц. Мережі, мобільний зв’язок, скайп. Але найкращим шляхом є використання так званих task tracking system (з англ. – система для відстежування завдань). Такі системи дають можливість дати завдання усім працівникам, відстежувати їх виконання віддалено, вирішувати проблеми миттєво.

Формалізація та опис бізнес-процесів залучає мову моделювання IDEF0. Методологія цієї системи представляє сукупність робіт і функцій. У ній побудовано ієрархічну модель об’єктів, це спрощує розуміння предметної області. В IDEF0 представлені логічні зв'язки між роботами, а не послідовність їх виконання в часі.

Основні бізнес-процеси системи:

- написання UserStory;
- набір команди
- робота з спринтом
- закриття проекту

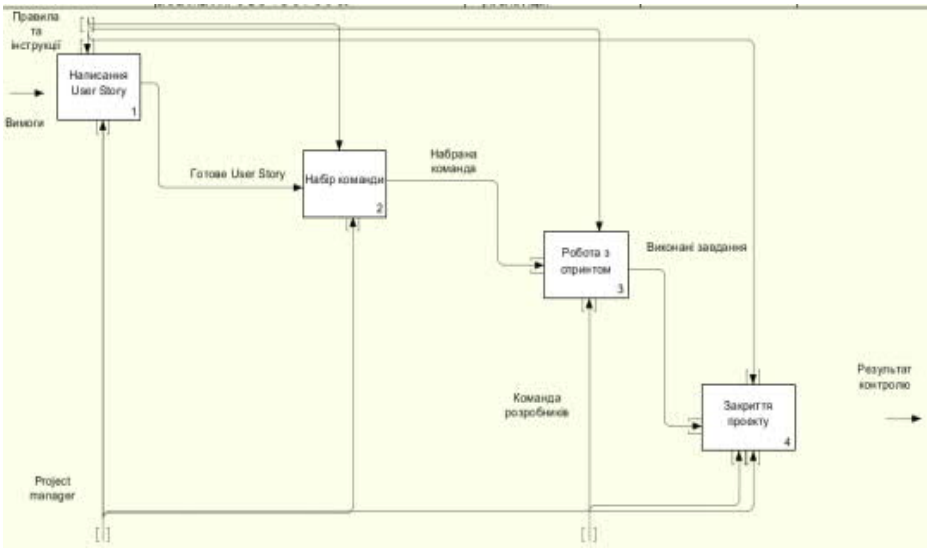

Рисунок 1.1 – Діаграма бізнес-процесів розроблюваного програмного продукту.

Розглянемо детальніше кожен з вище представлених процесів. На рисунку 1.2 представлено процес написання UserStory

Для написання User Story менеджерам необхідно проаналізувати усі умови. Усі вимоги потрібно проаналізувати у замовника, лише після того приступити до написання User Story. Характеристику процесу написання User Story представлено в таблиці 1.1

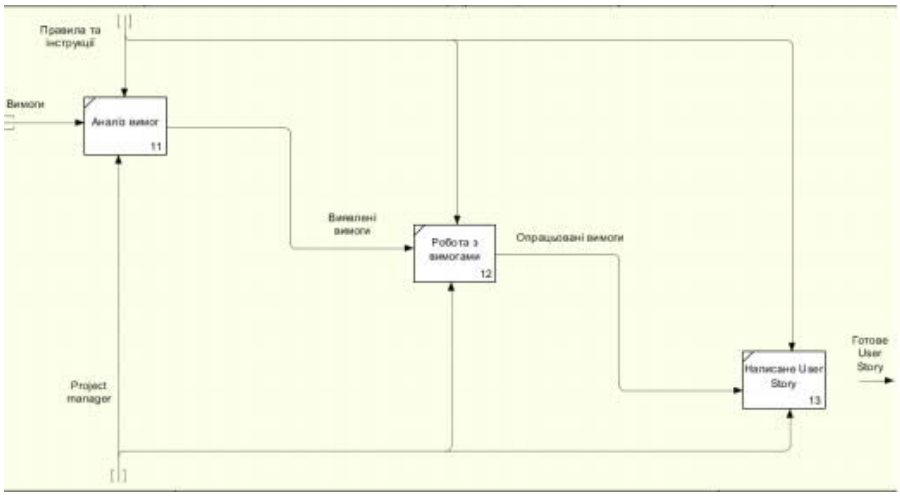

Рисунок – 1.2 Діаграма процесу написання User Story

Таблиця 1.1 – Характеристика бізнес-процесу “Написання User Story”

Назва характеристики	Значення характеристики
Ім'я процесу	Написання User Story
Основні учасники	Замовник, PM
Вхідна подія	Виявлення вимог
Вхідні документи	Список виявлених вимог
Вихідна подія	Готова User Story
Вихідні документи	Список User Stories
Клієнти процесу	Користувач

Наступним бізнес-процесом є формування команди, яка працюватиме над написанням проекту. Процес створення команди зображений на рисунку 1.3.

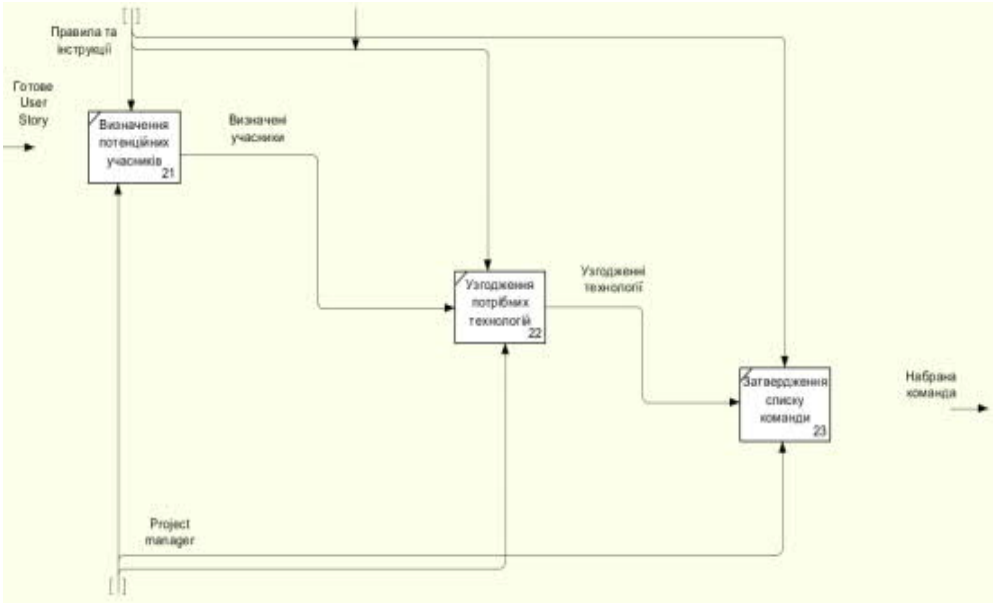

Рисунок 1.3 – Діаграма процесу набору команди

Для початку, перед набором команди, необхідно мати написані User Stories це потрібно для визначення кількості необхідних працівників.

Після визначення команди потрібно обговорити список технологій необхідних для реалізації проекту. Останній етап – це затвердження команди.

Характеристику процесу набору команди наведено в таблиці 1.2. Після затвердження команди можна починати написання проекту. Для цього створюється спринт, під час якого буде відбуватися написання проекту. Процес роботи зі спринтом зображений на рисунку 1.4.

Перед початком нового спринта необхідно розробити завдання, які мають бути реалізовані. Наступний етап – це виконання поставлених завдань. Після виконання завдань, їх необхідно перевірити на помилки, у разі виявлення – виправити.

Таблиця 1.2 – Характеристика бізнес-процесу “Набір команди”

Назва характеристики	Значення характеристики
Ім'я процесу	Набір команди
Основні учасники	PM, потенційні учасники команди
Вхідна подія	Виявлення User Stories
Вхідні документи	Список User Stories
Вихідна подія	Затвердження списку учасників команди
Вихідні документи	Список учасників команди
Клієнти процесу	Користувач

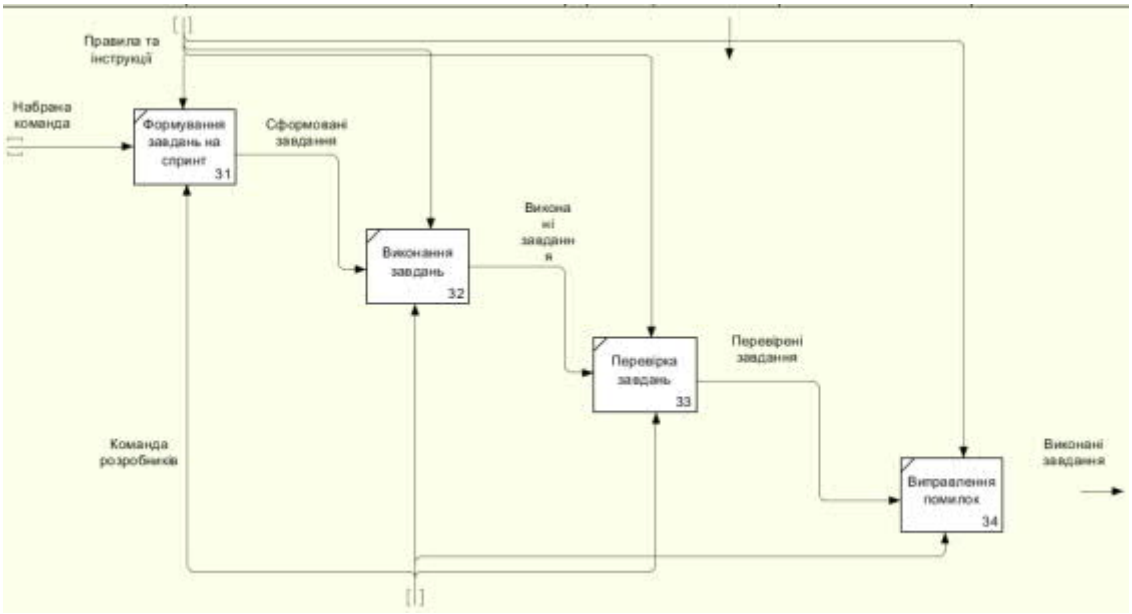

Рисунок 1.4 – Діаграма процесу роботи зі спринтом

Характеристику процесу роботи зі спринтом наведено в таблиці 1.3.

Таблиця 1.3 – Характеристика бізнес-процесу “Робота зі спринтом”

Назва характеристики	Значення характеристики
Ім'я процесу	Робота зі спритом
Основні учасники	Учасники команди
Вхідна подія	Затвердження списку учасників команди
Вхідні документи	Список учасників команди
Вихідна подія	Виконані завдання
Вихідні документи	Список виконаних завдань
Клієнти процесу	Користувач

Після виконання всіх завдань проект можна закривати. На рисунку 1.5 зображена діаграма процесу закриття проекту.

Перед закриттям проводиться кінцева перевірка виконаних завдань. Опісля перевірки розроблений проект потрібно залити на віддалений сервер, це необхідно для співпраці з замовником. Фінальним етапом є передача розроблюваного продукту безпосереднього його замовнику.

Характеристика закриття проекту наведена у таблиці 1.2.4

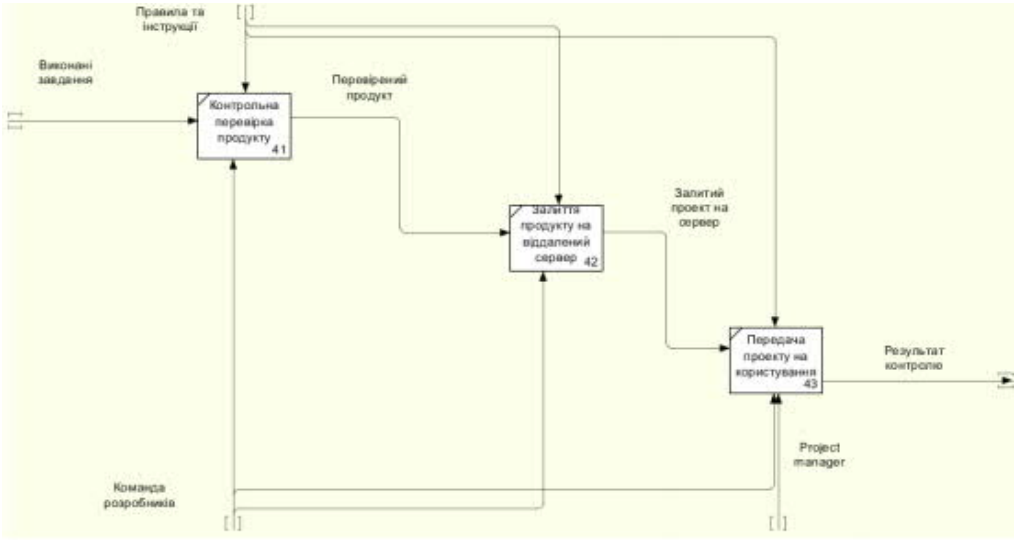

Рисунок 1.5 – Діаграма процесу закриття проекту

Таблиця 1.4 – Характеристика бізнес-процесу “Закриття проекту”

Назва характеристики	Значення характеристики
Ім'я процесу	Закриття проекту
Основні учасники	Учасники команди, PM
Вхідна подія	Виконані завдання
Вхідні документи	Список виконаних завдань
Вихідна подія	Результат контролю перевірки

1.2 Опис об’єкту дослідження

На сьогоднішній день ІТ-сфера набирає стрімких обертів надзвичайно швидко. Майже кожного дня на сьогоднішній день відкриваються нові компанії, які перебувають у цьому напрямку. Технології смаку – доволі нова компанія, яка розробляє різного роду програмне забезпечення. Заснування компанії було у 2015 році в Україні. Персонал є доволі молодим, але досвідченим та кваліфікованим. Команда налічує кандидатів наук, магістрів та бакалаврів. У компанії зосереджується увага на реалізації особливих рішень, які характеризуються унікальністю та відповідають бізнес-вимогам замовників. Персонал має досвід у розробці сучасного роду веб-додатків [5].

Складність та розміри проектів не мішає компанії забезпечувати своїх замовників новою технологією, сучасними розробками та чудовою інженерною допомогою, яку пропонує своїм замовникам.

Рисунок 1.6 – Структура оранізації ТОВ “Технології смаку”

Сьогодні компанія працює над декількома проектами у сфері email-маркетингу. Через великий об’єм проекту, керівнику необхідно стежити за виконанням усіх поставлених задач. Заради таких цілей компанія використовує

task tracking систему “YouTrack”. Поточний стан проекту можна з легкістю тримати завдяки такому продукту. Необхідно тільки моніторити сторінку, у якій відображається кількість зробленої роботи.

Комп’ютерний парк у компанії Технології смаку налічує 25 робочих машин. У кожного працівника, який приймає участь у розробці є по 2 робочі екрана фірми DELL. Всі машини використовуються в одних цілях – виконання проекту.

1.3. Аналіз існуючих програмних рішень та постановка задач проекту

На сьогоднішній день є різного роду програми для відстежування завдань. Вони створювались під особливості потреб користувача. Перед тим, як приступити до розробки такого додатку необхідно розглянути та проаналізувати вже існуючі інструменти для того, щоб зрозуміти специфікацію таких систем і все це врахувати для розробки нової технологічної системи для відстеження завдань. Для порівняння вибрано дві поширених системи, такі як YouTrack та Bugzilla. Одина з цих систем є платною, а інша – безкоштовна.

YouTrack – комерційний проект для відстеження помилок, продукт для управління проектами розроблений компанією JetBrains [25]. YouTrack підтримує пошукові запити, автодоповнення, маніпуляцію з наборами завдань, настройку набору атрибутів завдання, створення користувацьких робочих процесів і дозволяє активно використовувати клавіатуру в інтерфейсі (що є важливим для багатьох програмістів) [25].

Після завершення реєстрації, користувач переходить на нову сторінку, де можна створити новий проект. Наступним кроком буде вікно яке містить дошку та беклог, у них описані всі завдання, як необхідно виконати на проект. Це зображено на рис. 1.6, на якому зображено завдання чи описані баги.

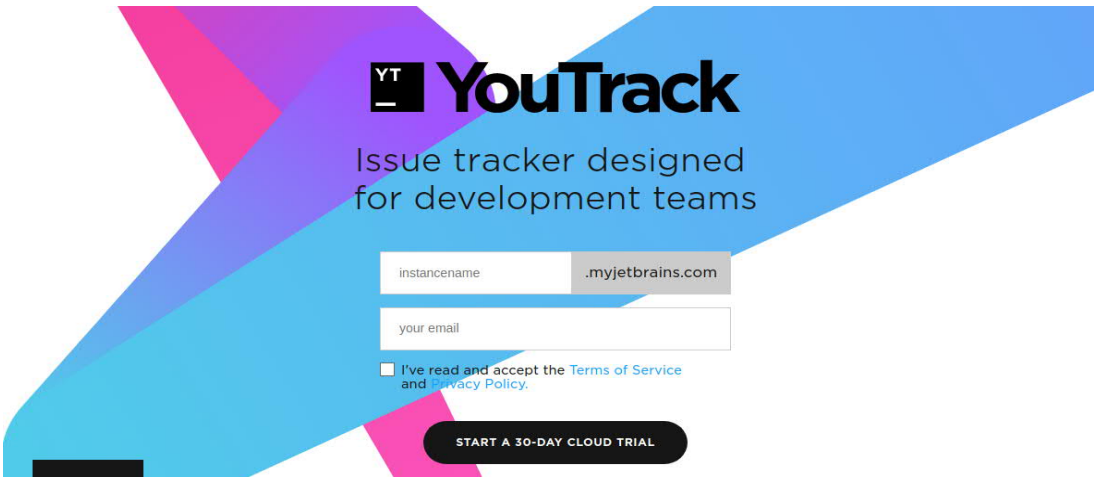

Рисунок 1.6 – Вікно реєстрації

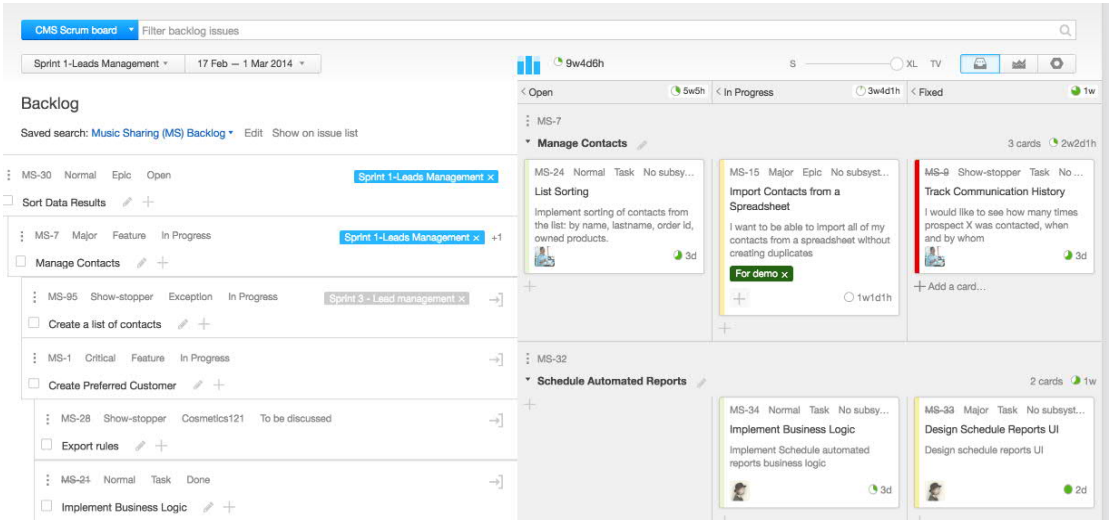

Рисунок 1.7 – Дошка з картками

Bugzilla (Багзілла) – система відстеження помилок і ведення завдань (англ. bugtracker tool) з веб-оболонкою. Bugzilla написана мовою Perl, розробляється проектом Mozilla і поширюється під вільною ліцензією MPL. Окрім Mozilla, система Bugzilla також використовується для відстежування помилок у більшості великих вільних проектів, включаючи KDE, GNOME, FreeBSD, ядро Linux, Apache, LibreOffice, в компаніях Red Hat і SUSE [3].

Рисунок 1.8 – Головна сторінка

Після етапу реєстрації користувачу необхідно створити проект. Щоб завершити цей процес необхідно вводити баги. На рисунку 1.9 зображена форма для створення звіту про помилку.

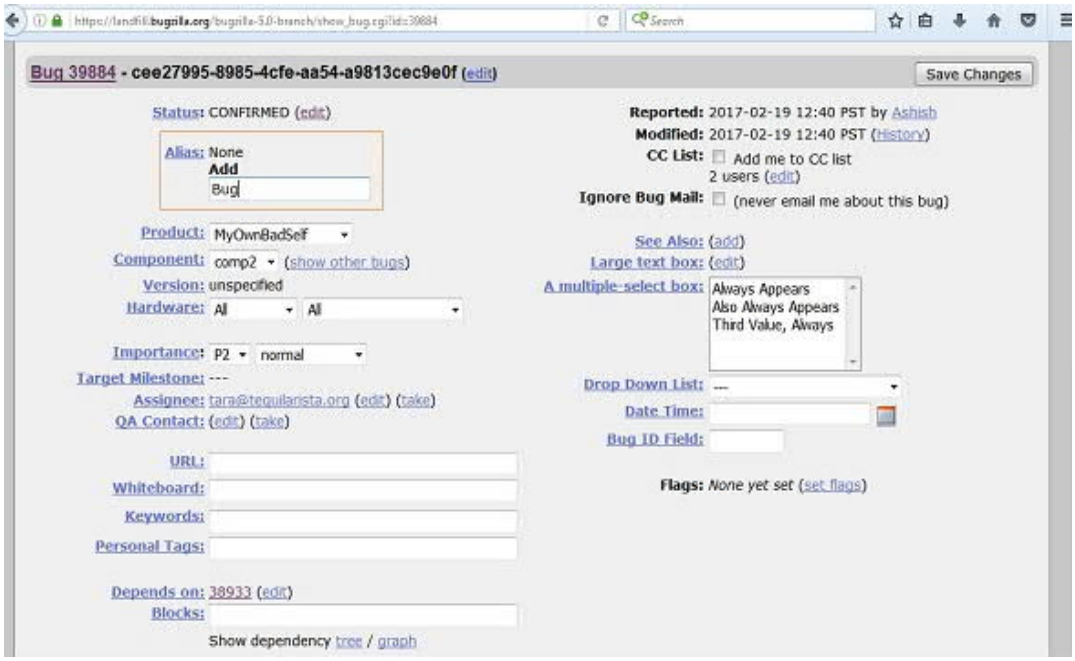

Рисунок 1.9 – Створення багу

Таблиця 1.5 – Порівняльна характеристика найбільш поширених систем для відстеження завдань

Фірма-розробник	JetBrains	Mozilla
1	2	3
Назва програмного продукту	YouTrack	Bugzilla
Остання версія	2018.1	5.1.2
Функціональність	- створення завдань; - оформлення багів; - можливість створювати Scrum та Kanban дошки для роботи; - індикатор прогресу; - пошук завдань; - експорт всіх завдань;	-розширені можливості пошуку - списки помилок у кількох форматах - заплановані звіти - звіти та діаграми - відстеження часу
Інтеграція з сторонніми сервісами	GitHub, IntelliJ IDEA, TestLink, TestRail, Subversion, CVS	TestLink, інтеграція з системою електронної пошти.
Інтерфейс користувача	Дружній та зрозумілий	Застарілий інтерфейс з низьким рівнем юзабіліті.
Допомога користувачу	Для допомоги користувачу створено багато відео-уроків, де показано як створити	Є багато документації по роботі з системою

Продовження таблиці 1.5

1	2	3
Фірма-розробник	JetBrains	Mozilla
Ціна за користування	від $200 за рік	безкоштовна
Переваги	- швидкодія; - зрозумілий інтерфейс; - інтеграція з сторонніми продуктами;	- безкоштовна - простота - автоматизація роботи з документацією - тісна інтеграція з системою електронної пошти.
Недоліки	- ціна; - відсутня можливість спілкування з командою за допомогою чату;	- обмежений функціонал; - відсутня можливість спілкування з командою за допомогою чату;

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Розроблення архітектури програмної системи

Для проектування системи було використано трирівневу архітектуру. До складу трирівневої архітектури входять такі компоненти: клієнтський додаток, який підключений до сервера додатків, та підключений до бази даних на сервері.

Структурну схему програмної системи зображено на рисунку 2.1.

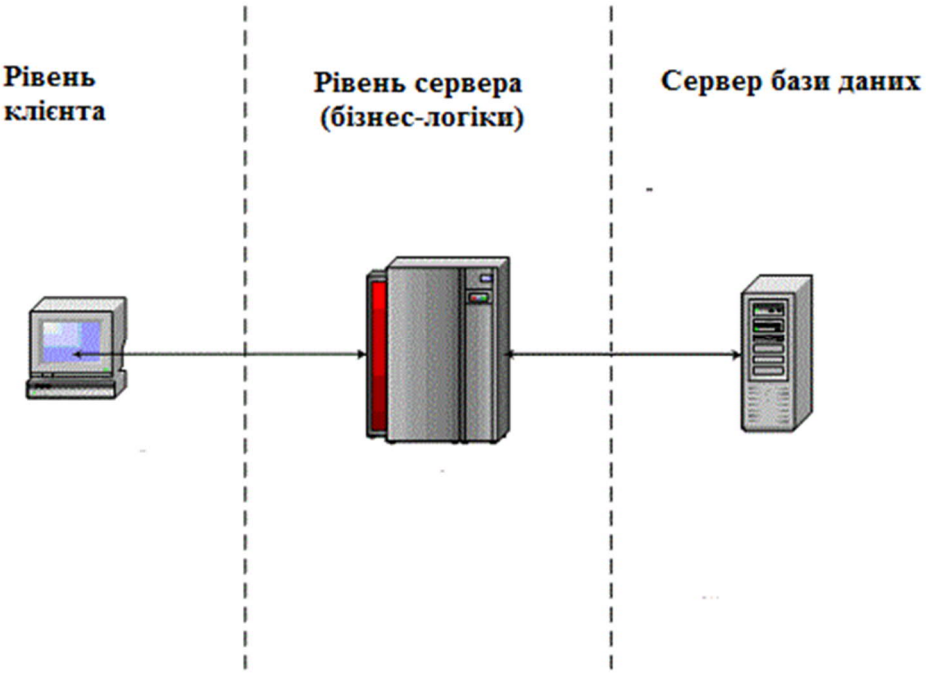

Рисунок 2.1 – Структурна схема програмної системи

Клієнтський рівень – створений та призначений для юзера. Перший рівень не повинен містити прямого зв’язку з БД, не має бути загруженим головною логікою бізнесу та не має зберігати програмний стан. У цьому рівні повинна міститись найлегша форма логіки: блок авторизації, чекінг значень, які будуть вноситись, у відповідності та відповідальності певної організації.

Другий рівень – серверний рівень. На цьому рівні відображається основна частина бізнес-логіки. На ньому перебуває програмний інтерфейс, який зв’язує частини юзера з логікою БД [2].

Гарантія збереження даних надає сервер БД, його включають до третього рівня. Він показує собою БД з процедурами, тригерами, та термінами реляційної моделі.

Було розроблено діаграму компонент для представлення статичного аспекту архітектури, яка зображена на рисунку 2.2

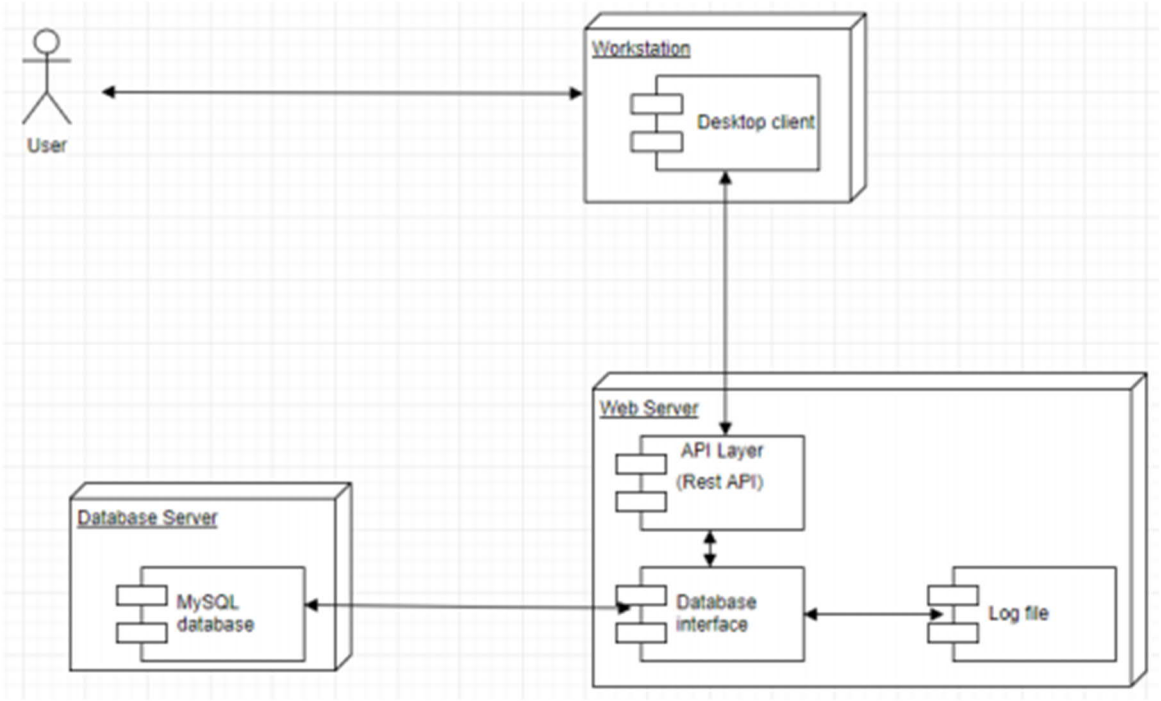

Рисунок 2.2 – Діаграма компонент

За введення клієнтом дані відповідає компонент “Workstation”.

За отримання введених даних користувачем з компонента Workstation за допомогою API, відповідає компонент “Web Server”, проводиться обробка з використанням з бізнес-логіки, та підготовка для взаємодії з базою даних. Все, що відбувається на сервері, знаходиться у файлах з логами. У компоненті “Database Server” відбувається обробка запитів, які були підготовлені на попередньому етапі.

За допомогою діаграм станів представляється складова архітектури. Діаграма станів для створення проекту зображена на рисунку 2.3.

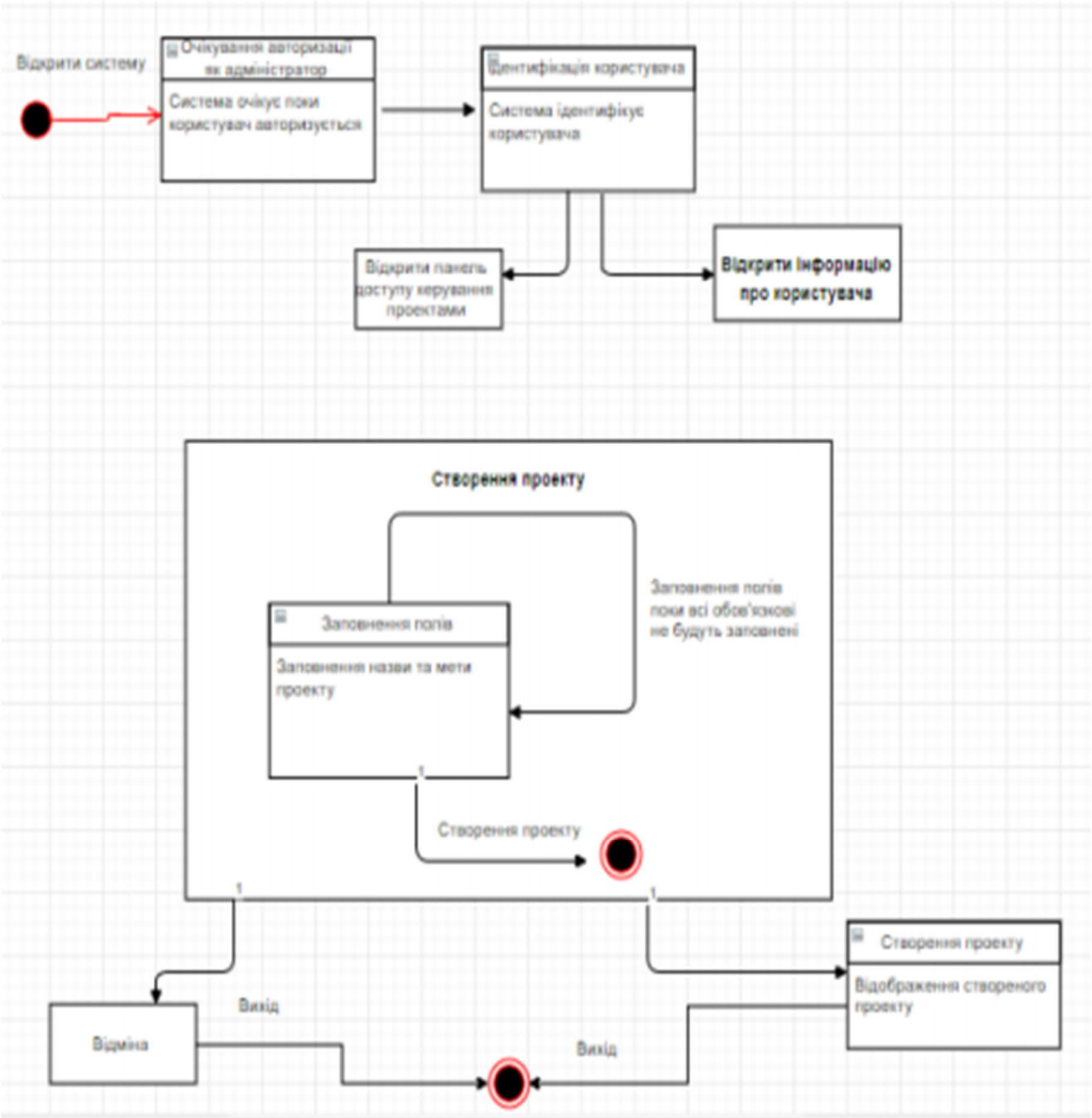

Рисунок 2.3 – Діаграма станів для створення проекту

На рисунку 2.4 зображено діаграму станів для створення спринта.

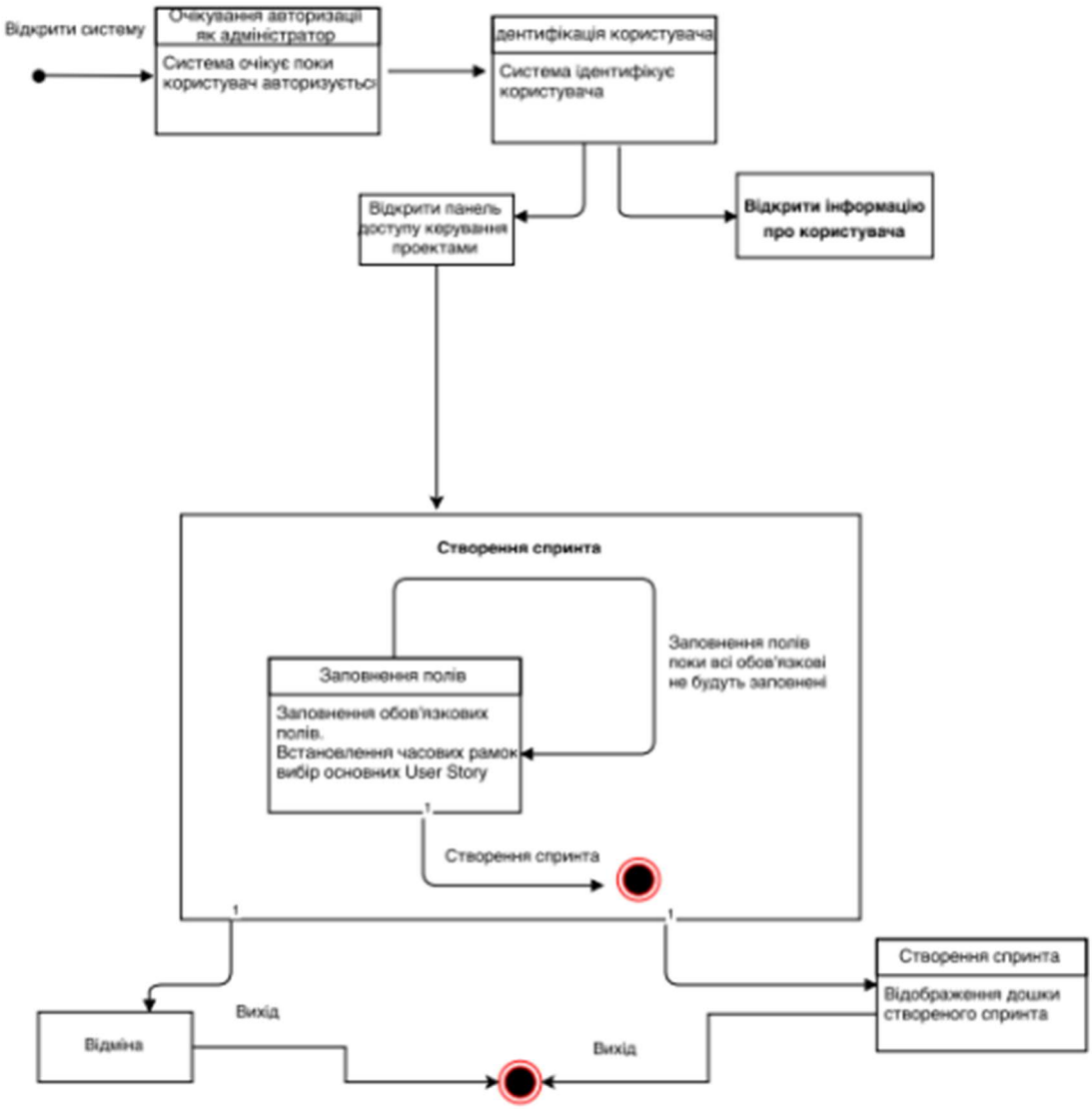

Рисунок 2.4 – Діаграма станів для створення спринта

На рисунку 2.5 зображена діаграма станів для створення картки.

Створити логічне представлення допомагає діаграма класів, на її основі створюється вихідний код описаних класів. Взаємозв’язки між класами та інтерфейсом можна побачити по значкам діаграм.

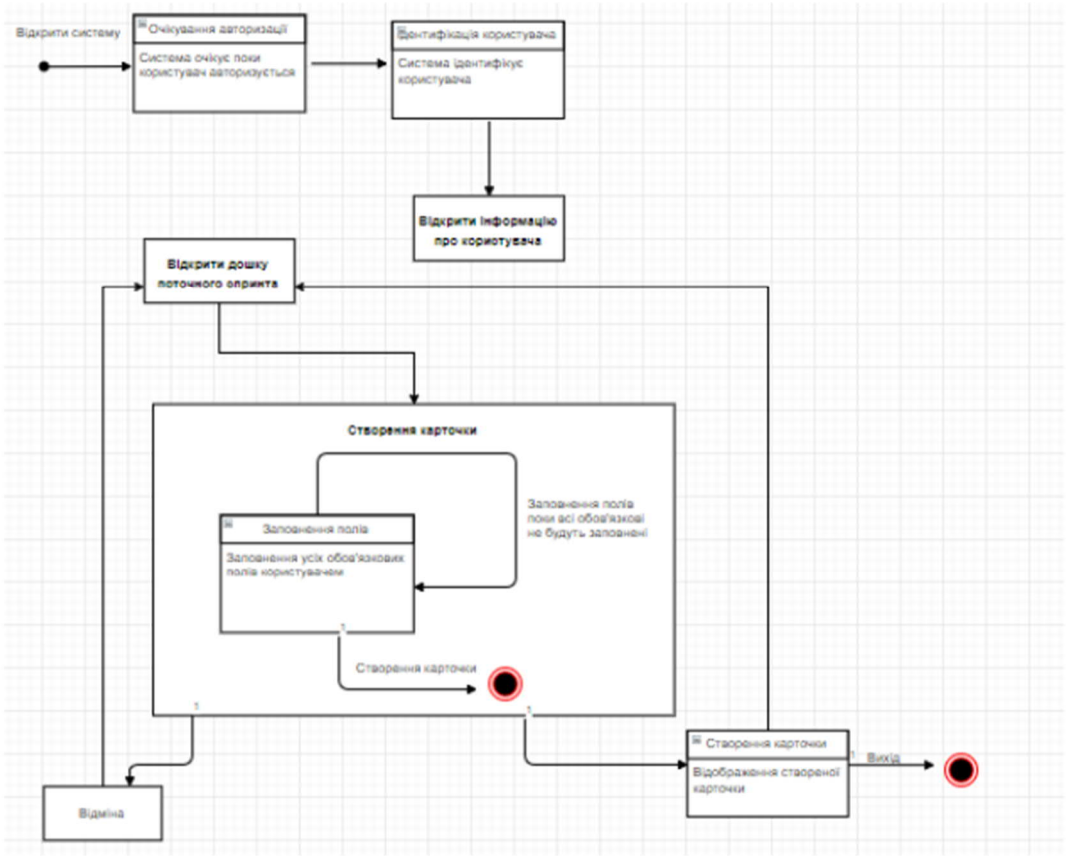

Рисунок 2.5 – Діаграма станів для створення картки

2.2 Аналіз існуючих алгоритмів розв’язання поставленої задачі

Шаблони проектування програмного забезпечення (англ. software design patterns) – ефектні способи вирішення задач проектування програмного забезпечення. Шаблон не є закінченим зразком, який можна безпосередньо транслювати в програмний код. Об'єктно-орієнтований шаблон найчастіше є зразком вирішення проблеми і відображає відношення між класами та об'єктами, без вказівки на те, як буде зрештою реалізоване це відношення.

Шаблон проектування або патерн (англ. Design pattern) в розробці програмного забезпечення – повторювана архітектурна конструкція, що представляє собою рішення проблеми проектування в рамках деякого часто виникає контексту.

Зазвичай шаблон не є закінченим зразком, який може бути прямо перетворений в код; це лише приклади розв'язання задач, який можна використовувати в різних ситуаціях. Об'єктно-орієнтовані шаблони показують відносини і взаємодії між класами або об'єктами, без визначення того, які кінцеві класи або об'єкти додатки будуть використовуватися.

«Низькорівневі» шаблони, що враховують специфіку конкретної мови програмування, називаються ідіомами. Це хороші рішення проектування, характерні для конкретної мови або програмної платформи, і тому не універсальні.

На найвищому рівні існують архітектурні шаблони, вони охоплюють собою архітектуру всієї програмної системи. Алгоритми за своєю суттю також є шаблонами, але не проектування, а обчислення, так як вирішують обчислювальні завдання.

Також існує інша група шаблонів проектування, що отримала назву GRASP – General Responsibility Assignment Software Patterns. Опис цих шаблонів наводить Крег Ларман у своїй книзі. Шаблони GRASP формулюють найбільш базові принципи розподілу обов'язків між типами. До складу шаблонів GRASP входить 9 шаблонів:

- інформаційний експерт (Information Expert);
- творець примірників класу (Creator);
- низька зв'язаність (Low Coupling);
- високе зчеплення (High Cohesion);
- контролер (Controller);
- поліморфізм (Polymorphism);
- штучний (Pure Fabrication);
- перенаправлення (Indirection);
- стійкий до змін (Protected Variations).

У порівнянні з повністю самостійним проектуванням, шаблони мають ряд переваг. Основна користь від використання шаблонів полягає в зниженні складності розробки за рахунок готових абстракцій для вирішення цілого класу

проблем. Шаблон дає рішенням своє ім'я, що полегшує комунікацію між розробниками, дозволяючи посилатися на відомі шаблони. Таким чином, за рахунок шаблонів проводиться уніфікація деталей рішень: модулів, елементів проекту – знижується кількість помилок.

Застосування шаблонів концептуально схожий на використання готових бібліотек коду. Правильно сформульований шаблон проектування дозволяє, відшукавши вдале рішення, користуватися ним знову і знову. Набір шаблонів допомагає розробнику вибрати можливий, найбільш підходящий варіант проектування.

Хоча легка зміна коду під відомий шаблон може спростити розуміння коду, на думку Стіва Макконнелла, із застосуванням шаблонів можуть бути пов'язані дві складності. По-перше, сліпе слідування деякого обраним шаблоном може привести до ускладнення програми. По-друге, у розробника може виникнути бажання спробувати деякий шаблон в справі без особливих підстав.

Багато шаблони проектування в об'єктно-орієнтованому проектуванні можна розглядати як ідіоматичне відтворення елементів функціональних мов. Пітер Норвіг стверджує, що 16 з 23 шаблонів, описаних в книзі «Банди чотирьох», в динамічно-тіпізіруемих мовах реалізуються значно простіше, ніж в C ++, або виявляються непомітні. Пол Грехем вважає саму ідею шаблонів проектування - антипаттерн, сигналом про те, що система не володіє достатнім рівнем абстракції, і необхідна її ретельна переробка. Неважко бачити, що саме визначення шаблону як «готового рішення, але не прямого звернення до бібліотеки» по суті означає відмову від повторного використання на користь дублювання. Це, очевидно, може бути неминучим для складних систем при використанні мов, які не підтримують комбінатори і поліморфізм типів, і це в принципі може бути виключено в мовах, що мають властивість гомоіконічності (хоча і не обов'язково ефективно), так як будь-який шаблон може бути реалізований в вигляді виконуваного коду.

2.3 Проектування структури бази даних

Заключним етапом перед програмною реалізацією бази даних є проектування структури БД. На цьому етапі ми визначаємо та описуємо всі поля та відношення які є у базі даних.

Використовуючи діаграму потоків даних можна показати інформаційні процеси.

Рисунок 2.7 – Діаграма потоків верхнього рівня

Діаграма декомпозиції зображена на рисунку 2.8.

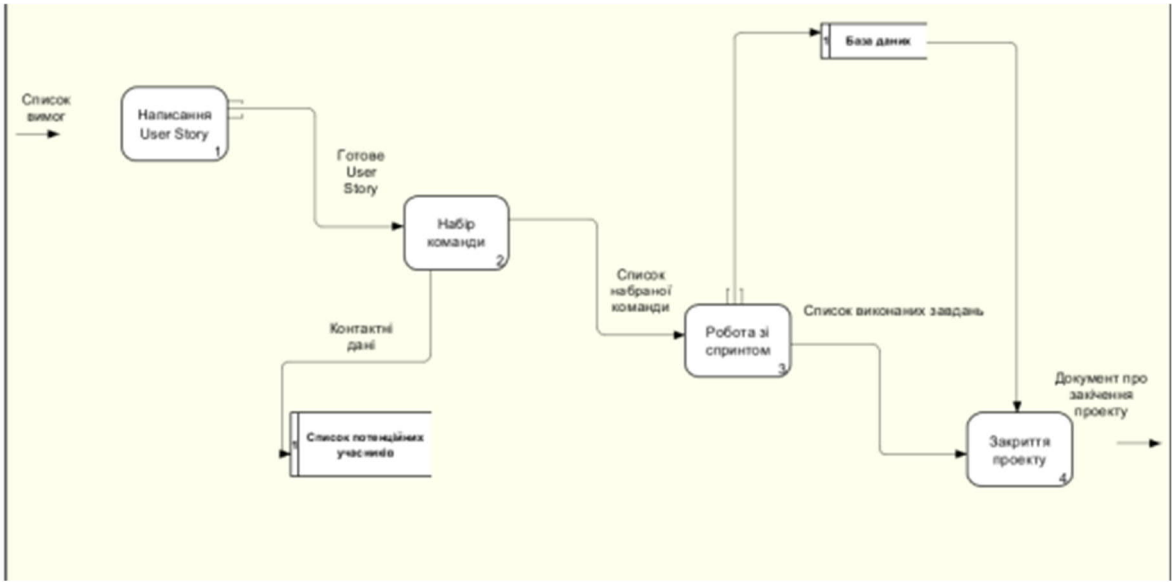

Рисунок 2.8 – Діаграма декомпозиції

ER-діаграму ми створюємо на останньому етапі проектування бази даних. (див. рис. 2.9).

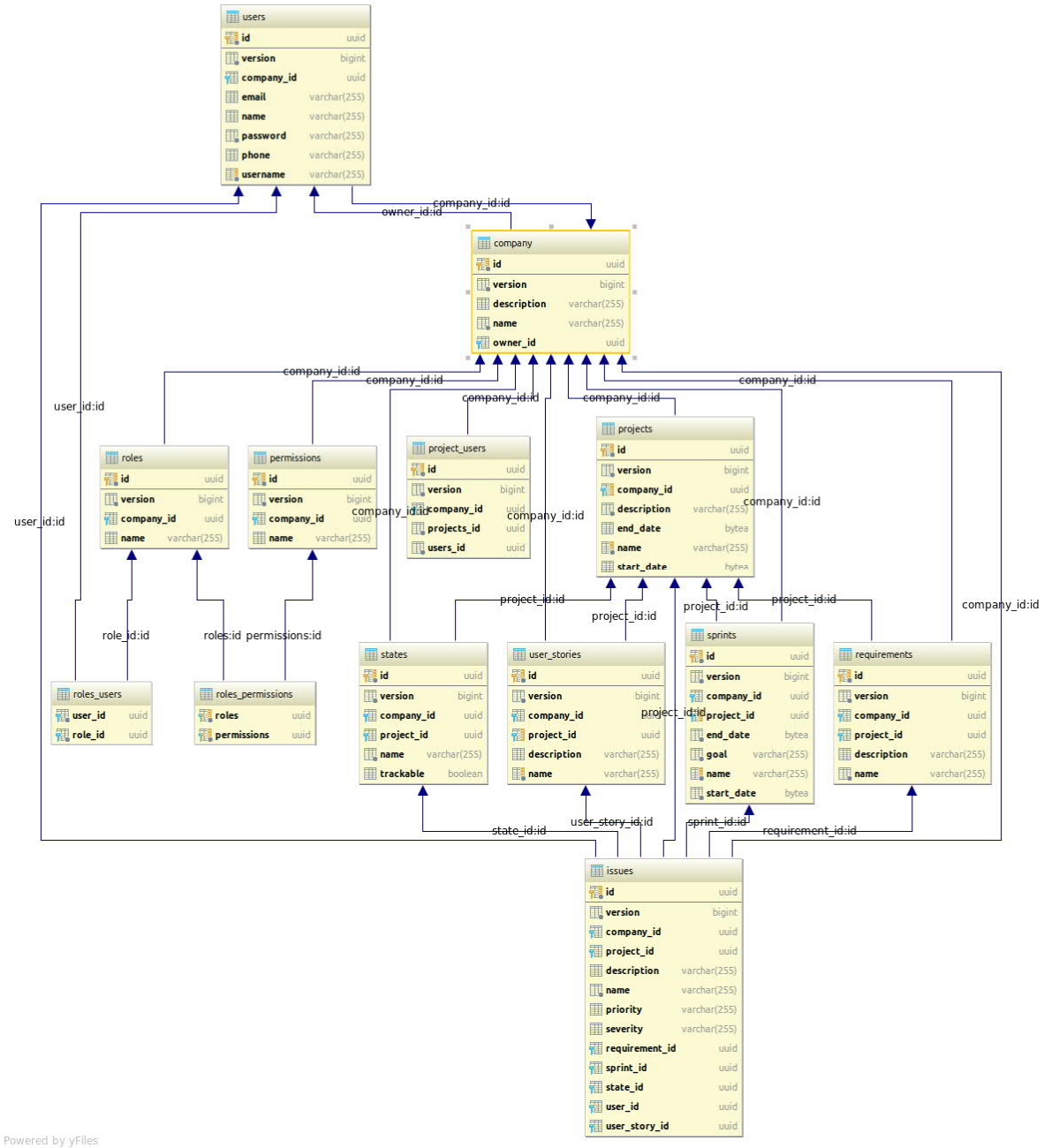

Рисунок 2.9 – ER-діаграма

Система, що розробляється повинна виконувати наступні функції:

- створення, редагування та видалення нової компанії;
- створення, редагування та видалення проекту;
- створення, редагування та видалення спринтів;
- створення, редагування та видалення карточок;

– запрошення, редагування та видалення нових користувачів.

Функція «Створення, редагування та видалення нової компанії». Мета системи – стежити за виконанням проекту. Дана функціональність розроблена для того, щоб створювати компанію, яка в подальшому буде володіти проектом.

Функція «Створення, редагування та видалення проекту» є відповідальною за додавання проекту, який буде розроблятися. Саме за ним в подальшому буде проводитися контроль виконання.

Функція «Створення, редагування та видалення спринтів» дозволяє юзеру додати проміжок часу протягом якого будуть виконуватися певні завдання.

Функція «Створення, редагування та видалення карточок» дає змогу юзеру додати картки - завдання, які необхідно виконати. Карточка також може мати тип «Баг» чи «User Story».

У наступній таблиці показано відношення між таблицями у БД.

Таблиця 2.1 – Відношення між таблицями в базі даних

Назва таблиці	Тип зв’язку	Назви таблиці до якої відношення
company	one to one	user
user	many to many	user_roles
user	many to one	company
permissions	many to one	company
roles	many to many	permissions
roles	many to one	company
projects	many to one	company
projects	many to many	users

Таблиця 2.2 – Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Користувач	Ім’я	varchar	20	User name
	Логін	varchar	20	username
	Пароль	varchar	30	password
	Електронна адреса	varchar	50	e-mail
	Номер	varchar	20	phone
Компанія	Назва	varchar	50	Company name
	Опис	varchar	50	description
Проект	Назва	varchar	200	Project name
	Опис	varchar	200	description
	Дата початку	timestamp		start_date
	Дата кінця	timestamp		end_date
Спринт	Назва	varchar	200	sprint name
	Мета	varchar	200	goal
	Дата початку	timestamp		start_date
	Дата кінця	timestamp		end_date
User Story				user_stories
	Назва	varchar	200	name
	Опис	varchar	200	description
Картка	Ім’я	varchar	200	Issues name
	Опис	varchar	200	description
	Пріоритет	varchar	200	priority
	Серйозність	varchar	200	severity

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Програмна реалізація системи

Для виконання завдання обрано мову Java та Spring, Spring security фреймворки.

Мова програмування Java взяла свій початок в 1991 році Джеймсом Гослингом, Патриком Нотоном, Крісом Уартом, Едом Франком і Майком Шериданом, вони працювали у компанії Sun Microsystems, Inc. Перша робоча версія зайняла 18 місяців часу. Першою назвою мови було "Oak" (Дуб), але в 1995 році було перейменовано на “Java”. Перед тим, як зробити першу реалізацію мови Oak в кінці 1992 р. та публічно представити про розробку Java навесні 1995 р. над розробкою працювало багато інших спеціалістів. Зокрема, Біл Джой, Артур ван Гоф та Тім Ліндхольм вклали великий внесок у розвиток основного прототипу Java.

Головними пунктами, які повпливали на вибір мови програмування на користь Java стали:

- простота;
- безпека;
- переносимість;
- об’єктна орієтованість;
- надійність;
- багатопоточність;
- архітектурна нейтральність;
- висока продуктивність;
- розподіленість;
- динамічність.

Для бек-енд частини продукту використано спрінг фреймворк. Основна перевага Spring – можливість розробки програми як набору слабо пов’язаних (loose-coupled) компонентів. Чим менше компоненти програми знають один про

одного, тим простіше розробляти новий і підтримувати існуючий функціонал додатка. Класичний приклад – управління транзакціями. Spring дозволяє вам керувати транзакціями абсолютно незалежно від основної логіки взаємодії з БД. Зміна цієї логіки не порушить транзакційність, так само як зміна логіки управління транзакціями не зламає логіку програми. Spring заохочує модульність. Компоненти можна додавати і видаляти майже незалежно один від одного. В принципі, додаток можна розробити таким чином, що він навіть не буде знати, що управляється за допомогою Spring. Також Spring помітно спрощує модульне тестування (unit-testing): в компонент, розроблений для роботи в IoC контейнері дуже легко впроваджувати фейкові залежності і перевірити роботу тільки цього компонента. Ну, і як приємне доповнення, Spring сильно полегшує ініціалізацію і налаштування компонентів додатка, дозволяючи гнучко налаштовувати додаток без істотних змін Java-коду [21].

Для того, щоб використання додатку було безпечним в дипломній роботі використано Spring Security. Spring Security – це Java / JavaEE framework, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Проект був розпочатий Беном Алексом (Ben Alex) в кінці 2003 року під ім'ям «Acegi Security», перший реліз вийшов в 2004 році. Згодом проект став офіційним дочірнім проектом Spring. Вперше публічно представлений під новим ім'ям Spring Security 2.0.0 в квітні 2008 року [15].

Розроблене програмне забезпечення дає можливість працювати з наступними функціями:

- реєстрація;
- проект;
- спринт;
- User story;
- картки-завдання/баги;
- чат;
- запрошення нового користувача.

Наведемо приклад програмного коду одного з основних класів – користувач.

```java
package com.lazytrack.core.entity;

import ...

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
@Entity
@Table(name = "users")
public class User extends CompanyAssociated {

  private String name;
  private String email;
  private String phone;
  @Column(name = "username", nullable = false, unique = true)
  private String username;

  @Column(name = "password", nullable = false)
  private String password;

  @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
  @JoinTable(name = "roles_users",
      joinColumns = @JoinColumn(name = "user_id"),
      inverseJoinColumns = @JoinColumn(name = "role_id"))
  private List<UserRole> roles = new ArrayList<>();
}
```

Рисунок 3.1 – Програмний код класу “Користувач”

Як можемо бачити, основними полями для користувача є name, email, phone. Крім цього повинен бути обов’язково username та password. Всі інші класи основного функціоналу наведені у додатку А.

PostgreSQL – це повноцінна SQL СУБД з великим списком можливостей і величезною кількістю людей по всьому світу, які використовують і розробляють цю СУБД. На відміну від ще однієї вільної СУБД MySQL, розробка якої спочатку орієнтувалася на веб, розробка PostgreSQL орієнтувалася на використання в складних додатках. Саме тому акцент завжди робився на надійність, наявність розвиненої функціональності і відповідно стандартам. При цьому, звичайно, PostgreSQL можна точно також використовувати і в веб-додатках, де дана СУБД показує незмінно відмінні результати, при кращій масштабованості і налаштованості [17].

3.2. Програмні модулі системи

3.2.1. Компоненти ПЗ

Даний програмний продукт було розроблено на мові програмування Java, а база даних, яка була використана при розробці – це PostgreSQL. Для того щоб працював додаток, потрібно встановити тільки Oracle JRE версії 8 і вище.

3.2.2. Встановлення ПЗ

Для роботи системи необхідно встановити JRE версії 8 і вище. Оскільки додаток розроблений на мові програмування Java і компілюється в байт код, то він може бути запущений на будь-які операційній системі. Опишемо встановлення JRE на популярних операційних системах таких як Windows, Linux та MacOC.

Для того щоб встановити JRE на цю операційну систему Linux потрібно:

1. Відкрити термінал.
2. Ввести команду “sudo apt-get install default-jre”. Після цього JRE буде встановлено.

Для того щоб встановити JRE на цю операційну систему Windows потрібно:

1. Скачати JRE версії 8 і вище.
2. Відкрити скачаний файл.
3. Натисніть кнопку Install (Встановити), щоб прийняти умови ліцензійної угоди і продовжити установку.
4. Компанія Oracle співпрацює з компаніями, що пропонують різні продукти. Під час установки Java може бути запропоновано встановити такі програми. Перевірте, що потрібні програми обрані, і натисніть кнопку Next (Далі) для продовження установки.

5. З'являться кілька діалогових вікон із запитами підтвердження останніх етапів установки; в останньому діалоговому вікні натисніть кнопку Close (Закрити). Процедура установки Java завершена.

Для того щоб встановити JRE на цю операційну систему MacOC потрібно:

1. Скачати JRE версії 8 і вище.
2. Відкрити скачаний файл.
3. Два рази нажміть на іконку пакета, щоб запустити майстер установки
4. Майстер установки відображає екран вітання Java. Натисніть Next (Далі).
5. Після завершення установки відображається екран підтвердження. Клацніть Закрити, щоб завершити установку.

Крім цього, необхідно ще встановити PostgreSQL. Для того щоб встановити цю БД на Linux потрібно:

1. Відкрити термінал
2. Виконати команду sudo apt-get install postgresql postgresql-contrib
3. Під’єднатись до Бази даних.

3.2.3 Базові функції ПЗ

Для адміністратора були розроблені такі базові функції:

- реєстрація;
- створення компанії;
- створення проекту;
- редагування проекту;
- видалення проекту;
- створення спринта;
- редагування спринта;
- видалення спринта;
- створення User Story;
- редагування User Story;

- видалення User Story;
- створення картки;
- редагування картки;
- видалення картки;
- відстеження часу;
- користування чатом;
- запрошення нового користувача.

Для користувача передбачені такі функції:

- авторизація;
- створення спринта;
- редагування спринта;
- видалення спринта;
- створення User Story;
- редагування User Story;
- видалення User Story;
- створення картки;
- редагування картки;
- видалення картки;
- відстеження часу;
- користування чатом.

Реєстрація зображена на рисунку 3.2

Рисунок 3.2 – Реєстрація користувача та компанії

Авторизація зображена на рисунку 3.3

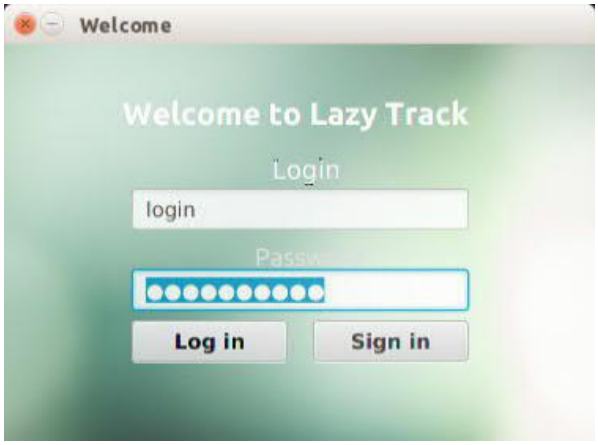

Рисунок 3.3 – Авторизація

Створення нового проекту показано на рисунку 3.4

Рисунок 3.4 – Створення нового проекту

Створення нового спринта зображено на рисунку 3.5

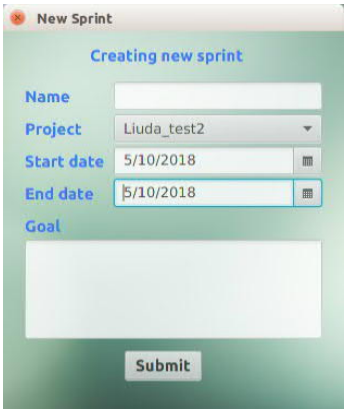

Рисунок 3.5 – Створення нового спринта

Створення User Story показано на рисунку 3.6

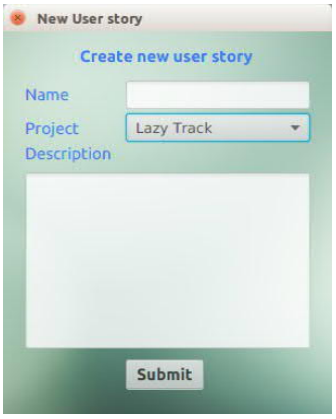

Рисунок 3.6 – Створення User Story

3.3 Тестування та верифікація розробленого програмного забезпечення

Було проведено тестування яке необхідне щоб перевірити розроблений продукт на відповідність вимог та виявлення та дефектування помилок. На цьому етапі є дуже важливим перевірка, адже це визначить спритність системи для наданого користування. Сьогодні, є різного роду тестування, кожне з них дозволяє перевірити той чи інших аспект розроблюваного продукту.

Було проведено наступні тестування, та враховано специфікацію програмного продукту:

- функціональне тестування;
- GUI тестування;
- тестування безпеки.

3.3.1 Функціональне тестування

Було проведено функціональне тестування для системи, яка розроблялася. Це дало змогу перевірити відповідність вимог, які були прописані в специфікації. Такого роду тести включають усі розроблювані функції та приділяють увагу типовим помилкам в програмі.

Було проведено 17 тестів. Усі показали позитивний результат. Тому цей тест показує, що усі вимоги було виконано. Текст функціонального тестування наведено в додатку Б.

3.3.2 GUI тестування

Кожна програма має чітко взаємодіяти з юзером, від цього безпосередньо залежить її успіх. Неправильна система злегкістю може показати низку проблем. Перевагами ручного тестування:

- контроль коректності проводиться людиною;

– пошук «косметичних» дефектів;

– аналіз успішності проходження тесту буде виконуватися не за формальними ознаками, а згідно людському сприйняттю.

Для проведення GUI тестування було складено чек-лист, згідно якого і проводилося тестування. Чек-лист знаходиться у додатку В.

За результатами перевірки згідно чек-листу ніяких дефектів виявлено не було.

3.3.3. Тестування безпеки

Тестування безпеки це дуже важлива складова для продукту що розробляється. Оскільки в системі присутні 2 ролі (клієнт та адміністратор) тому необхідно забезпечити максимальний захист кожного з учасників. Для забезпечення захисту для простого користувача було заблокований деякий функціонал, який доступний тільки для адміністратора. Наприклад, запрошення нового користувача у систему. Це може зробити тільки користувач з роллю адміністратор. На рис 3.7 зображені меню, які доступні тільки для адміністратора.

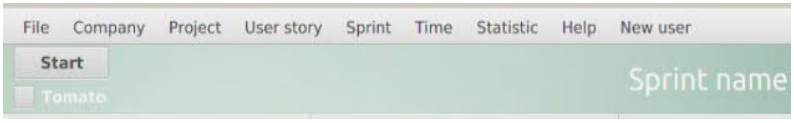

Рисунок 3.7– Функції, які доступні адміністратору

На рисунку 3.8 зображені меню, які бачить простий користувач.

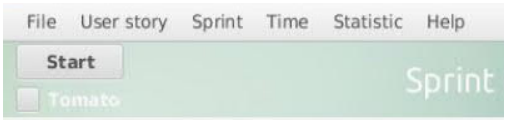

Рисунок 3.8 – Функції, доступні користувачу

Всі паролі користувачів зберігаються в базі даних у зашифрованому вигляді за допомогою BCrypt Hash. При введені пароля у форму він відображається символом «•», що не дає змогу підглянути пароль іншій особі.

Дуже важливим є процес авторизації та аутентифікації. Після проходження авторизації всі дані про користувача за допомогою токена передаються на сервер. Час життя токена 2 хвилини. Після цього часу він автоматично перегенеровується, що запобігає різним шкідливим програмам скористатися ним. Також це запобігає втратам даних. Під час тестування безпеки була протестована можливість користувала потрапити на сторінку, яка доступна тільки адміну. Для цього було використано програму Postman, в якій через API була здійснена перевірка. Всі спроби були провальними, на запит повертався HTTP код 403, який означає, що доступ заборонено. Отже, в такий спосіб зловмиснику також не вдасться отримати будь-яку інформацію.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

В цьому розділі бакалаврської роботи (БР) проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання програмного забезпечення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для моделювання роботи сенсора температури AD7814 і характеризується підвищеною ефективністю виконання алгоритму, що призводить до зменшення часу візуального представлення об’єкту дослідження.

4.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$K = K_1, + K_2 \tag{4.1}$$

де K_1 – витрати на розробку програмних засобів, грн;

K_2 – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп’ютері, грн.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($Вф$);
- витрати на покупні вироби ($Пв$);

– витрати на придбання спецобладнання для проведення експериментальних робіт (*Об*);

– накладні витрати (*Н*);

– інші витрати (*Ів*).

4.1.1 Розрахунок витрат на оплату праці

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці програмного забезпечення задіяні наступні спеціалісти – розробники, а саме – керівник проекту, студент-дипломник, консультант техніко-економічного розділу.

Таблиця 4.1 – Вихідні дані для розрахунку витрат на оплату праці

Посада виконавців	Місячний оклад, грн
Керівник ВКР, доцент	5950
Консультант техніко-економічного розділу, доцент	7293
Студент	1400

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \qquad (4.2)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду, год;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду, грн.

Середньо годинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \tag{4.3}$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ – місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год (приймаємо 168 год).

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці

Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год	Витрати на розробку, грн
Керівник ВКР, доцент	16	49,5	792
Консультант техніко-економічного розділу, доцент	2	63,8	127,6
Студент	100	8,3	830
Разом	1749		

4.1.2 Відрахування на соціальні заходи

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати:

$$B_{Ф} = \frac{20,5}{100} \cdot 1749 = 358,5 \text{ (грн)} \quad (4.4)$$

4.1.3 Розрахунок витрат на матеріали та комплектуючі

У таблиці 4.3 наведений перелік купованих виробів і розраховані витрати на них.

Таблиця 4.3 – Розрахунок витрат на матеріали та комплектуючі

Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
Папір (формат А4)	уп	90,0	1	90,00	9,0	99,0
Ручка кулькова	шт	20,0	1	20,00	2,00	22,0
Олівець простий	шт	4,0	1	4,00	0,4	4,40
Зошит, 18 арк	шт	6,0	1	6,00	0,6	6,60
Тонер для принтера	уп	50	1	50	5,0	55,0
Разом						187

4.1.4 Витрати на використання комп’ютерної техніки

Витрати на використання комп’ютерної техніки включають витрати на амортизацію комп’ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп’ютером. За даними обчислювального центру ТНЕУ для комп’ютера типу IBM PC/ATX вартість години роботи становить 5,2 грн. Середній щоденний час роботи на комп’ютері – 2 години. Розрахунок витрат на використання комп’ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4 – Розрахунок витрат на використання комп'ютерної техніки

Назва етапів робіт, при виконанні яких використовується комп'ютер	Час використання комп'ютера, год.	Витрати на використання комп'ютера, грн.
Проведення досліджень та оформлення їх результатів	60	312
Оформлення техніко-економічного розділу	8	41,6
Оформлення ВКР	12	62,4
Разом	80	416

4.1.5 Накладні витрати

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати:

$$H = 1{,}5 \cdot 1749 = 2623{,}5 \text{ (грн)} \quad (4.5)$$

4.1.6 Інші витрати

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I = 1749 \cdot 0{,}1 = 174{,}9 \text{ (грн)} \quad (4.6)$$

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_{ПВ} + H + I \quad (4.7)$$

$$K_1 = 1749 + 358{,}5 + 187 + 2623{,}5 + 174{,}9 = 5092{,}9 \text{ (грн)} \qquad (4.8)$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від} \qquad (4.9)$$

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн/год.

$t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 40 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,2 грн. Тому:

$$K_2 = 5{,}2 \cdot 80 = 416 \text{ (грн)} \qquad (4.10)$$

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення (таблиця 2.5).

Таблиця 4.5 – Кошторис витрат на розробку програмного забезпечення

Найменування витрат	Сума витрат, грн
Витрати на оплату праці	1749
Відрахування у спеціальні державні фонди	358,5
Витрати на куповані вироби	187
Накладні витрати	2623,5
Інші витрати	174,9
Витрати на відлагодження і дослідну експлуатацію програмного продукту	416
Разом	5508,9

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}} \tag{4.11}$$

де E_n – одноразові експлуатаційні витрати на ПЗ (аналог), грн;

E_{1n} – вартість підготовки даних для експлуатації ПЗ (аналогу), грн;

E_{2n} – вартість роботи комп'ютера для розробки програмного забезпечення (аналогу), грн.

Річні експлуатаційні витрати $B_{\text{еп}}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}} \tag{4.12}$$

де $N_{\text{п}}$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{n} n_i t_i c_i \tag{4.13}$$

де i – категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i – кількість працівників i-ої категорії, осіб;

t_i – трудомісткість роботи співробітників *i*-ої категорії по підготовці даних, год.;

c_i – середнього годинна ставка працівника і-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m} \tag{4.14}$$

де c_i^0 – основна місячна заробітна плата працівника і-ої категорії, грн;

b – коефіцієнт, який враховує додаткову заробітну плату;

m – кількість робочих годин у місяці, год.

Для роботи з даними як для поточного програмного забезпечення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: с° = 1400 грн. Тоді:

$$c_1 = \frac{1400(1+0{,}57)}{22*8} = 12{,}5\,(\text{грн/год}) \tag{4.15}$$

Трудомісткість підготовки даних для програмного забезпечення складає 1 год, для аналога 1,5 год.

Таблиця 4.6 – Розрахунок витрат на підготовку даних та реалізацію програмного забезпечення на комп'ютері

Час роботи співробітників, год	Середньогодинна заробітна плата, грн/год	Витрати, грн
Проектне рішення		
1	12,5	12,5
Аналог		
1,5	12,5	18,75

Витрати на експлуатацію комп’ютера визначається за формулою:

$$E_{2П} = t * S_{МГ} \quad (4.16)$$

де t – витрати машинного часу для реалізації програмного продукту, год;

$S_{МГ}$ – вартість однієї години роботи комп’ютера, грн/год.

$$E_{2n} = 1 \cdot 5{,}2 = 5{,}2 \text{ (грн)}; \; E_{2a} = 1{,}5 \cdot 5{,}2 = 7{,}8 \text{ (грн)} \quad (4.17)$$

$$E_n = 12{,}5 + 5{,}2 = 17{,}7 \text{ (грн)}; \; E_a = 18{,}75 + 7{,}8 = 26{,}55 \text{ (грн)} \quad (4.18)$$

$$B_{en} = 17{,}7 \cdot 252 = 4460{,}4 \text{ (грн)}; \; B_{ea} = 26{,}55 \cdot 252 = 6690{,}6 \text{ (грн)} \quad (4.19)$$

4.3 Розрахунок ціни споживання програмного продукту

Ціна споживання – це витрати на придбання і експлуатацію го продукту за весь строк його служби:

$$Ц_{С(П)} = Ц_{П} + B_{(E)NPV} \quad (4.20)$$

де $Ц_n$ – ціна придбання програмного продукту, грн.

$$Ц_{П} = K(1 + \frac{П_{Р}}{100}) + K_0 + K_к \quad (4.21)$$

де K – кошторисна вартість;

$П_р$ – рентабельність;

$К_о$ – витрати на прив'язку та освоєння програного забезпечення на конкретному об'єкті, грн;

$К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн.

$$Ц_Д = 5508{,}9 \cdot (1+0{,}3) = 7161{,}57 \text{ (грн).} \quad (4.22)$$

Вартість витрат на експлуатацію програмного забезпечення (за весь час його експлуатації), грн:

$$B_{епрv} = \sum_{t=0}^{T} \frac{B_{еП}}{(1+R)^t} \quad (4.23)$$

де $B_{еп}$ – річні експлуатаційні витрати, грн;

T – термін служби програмного забезпечення, років;

R – річна ставка проценту банку.

$$B_{епрv} = \sum_{t=1}^{5} \frac{4460{,}4}{(1+0{,}08)^t} = 17845{,}3 \text{ (грн)} \quad (4.24)$$

$$B_{епрv} = \sum_{t=1}^{5} \frac{6690{,}6}{(1+0{,}08)^t} = 26767{,}9 \text{ (грн)} \quad (4.25)$$

Тоді ціна споживання програмного забезпечення дорівнюватиме:

$$Ц_{сп} = 7161{,}57+17845{,}3 = 25006{,}9 \text{ (грн)} \quad (4.26)$$

Аналогічно визначається ціна споживання для аналогу:

$$Ц_{са} = 6500{,}0+26767{,}9 = 33267{,}9 \text{ (грн)} \quad (4.27)$$

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері розробки програмного продукту:

$$E_{ПР} = Ц_{П} - Ц_{А} \quad (4.28)$$

$$E_{ПР} = 7161,57 - 6500,0 = 661,57 \text{ (грн)} \quad (4.29)$$

Річний економічний ефект в сфері експлуатації:

$$E_{КС} = B_{ЕА} - B_{ЕП} \quad (4.30)$$

$$E_{кс} = 6690,6 - 4460,4 = 2229,6 \text{ (грн)} \quad (4.31)$$

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t} \quad (4.32)$$

$$\Delta E_{екс} = \sum_{t=1}^{5} 2229,6 * (1+0,08)^{5-t} = 13065,41 \text{(грн)} \quad (4.33)$$

Сумарний ефект складає:

$$E = E_{пр} + \Delta E_{екс} = 661,57 + 13065,41 = 13726,98 \text{ (грн)} \quad (4.34)$$

Таблиця 4.7 – Показники економічної ефективності програмного забезпечення

Найменування	Одиниці вимірювання	Значення показників	
		Базовий варіант	Новий варіант
Капітальні вкладення	грн.	–	5508,9
Ціна придбання	грн.	6500,0	7161,57
Річні експлуатаційні витрати	грн.	17200,15	26767,9
Ціна споживання	грн.	24721,35	33267,9
Економічний ефект в сфері проектування	грн.	–	661,57
Економічний ефект в сфері експлуатації	грн.	–	2229,6
Додатковий ефект в сфері експлуатації	грн.	–	13065,41
Сумарний ефект	грн.	13726,98	

4.5 Висновки

В даному розділі проведено розрахунок витрат на розробку програмного забезпечення. Здійснено порівняння з існуючим аналогом, і цим показано, що вказане програмне забезпечення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, зручність. Згідно проведеного економічного обґрунтування зазначене програмного забезпечення є конкурентноздатним. Крім того, отримано економічний ефект у розмірі 13726,98 грн. і тому розробка і впровадження цього програмного забезпечення є економічно доцільними.

ВИСНОВКИ

Результатом виконання дипломної роботи є створений новий програмний продукт - система управління задачами “Lazy_Track”. Перед початком роботи було досліджено предметну область та проаналізовано вже існуючі аналоги такої системи для того, щоб виявити переваги та недоліки таких інструментів та врахувати це при розробці свого додатку. Також було створено діаграму варіантів використання та здійснено розкадровку. Наступним етапом було формування функціональних та нефункціональних вимог до майбутнього продукту.

1. Визначено архітектуру програмного продукту. Далі було побудовано діаграму станів та UML діаграму класів. Під час роботи з базою даних PostgreSQL було створено ER-діаграму, також визначено зв’язки між таблицями. Завершальним на цьому етапі було створення таблиці ідентифікаторів.

2. Створено систему згідно визначених на перших етапах вимог. Додаток було створено на мові програмування Java з використанням різних фреймворків. Після написання застосунку його було протестовано. А саме проводила функціональне тестування та тестування безпеки, що є дуже важливим. Також була розроблена інструкція для користувача під популярні операційні системи. Останнім, що було виконано це визначено базові функції додатку для користувача та адміністратора.

3. Розроблено програмний продукт з тими функціями, які були визначені в специфікації вимог. Результати тестування показали, що система працює справно.

4. Результати бакалаврської роботи доповідалися на інтернет конференції “Наукова-практична конференцыя інтелектуальны комп’ютерні системи та мережі”

5. Проведено розрахунок витрат на розробку програмного забезпечення. Здійснено порівняння з існуючим аналогом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вельма А. М. Вибір системи відстеження помилок в залежності від конфігурації програмного забезпечення / А. М. Вельма, Є. Ю. Лактіонов // Вісник НТУУ "КПІ". Сер. : Інформатика, управління та обчислювальна техніка. – 2010. – Вип. 52. – С. 137-141

2. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%D0%BD%D1%82-%D1%81%D0.

3. Лог [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3.

Офіційний сайт системи Bugzilla [Електронний ресурс] – Режим доступу до ресурсу: https://www.bugzilla.org/.

4. МЕТОДИЧНІ ВКАЗІВКИ до виконання та захисту дипломної роботи / [М. П. Дивак, Я. Ш. Михайло, Р. П. Юрій та ін.]., 2014. – 58 с.

5. Офіційний сайт системи Bugzilla.Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.bugzilla.org/features/.

6. Офіційний сайт компанії ScalHive [Електронний ресурс] – Режим доступу до ресурсу: https://scalhive.com/.

7. Офіційний сайт системи YouTrack [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/youtrack/.

8. Офіційний сайт компанії Spring Securiry framework [Електронний ресурс] – Режим доступу до ресурсу: https://projects.spring.io/spring-security/.

9. Що таке JAVA? [Електронний ресурс] – Режим доступу до ресурсу: http://a-yak.com/shho-take-java/.

10. Що це таке User Stories? [Електронний ресурс] – Режим доступу до ресурсу: https://www.quality-assurance-group.com/user-stories/.

11. Вислобоков В. Что такое PostgreSQL? Для тех кто сомневается! [Електронний ресурс] / Виктор Вислобоков – Режим доступу до ресурсу: http://postgresql.ru.net/docs/about_pgsql.html.

12. Сидоров К. О роли PM и менеджменте больших масштабируемых проектов [Електронний ресурс] / Кирил Сидоров– Режим доступу до ресурсу: https://dou.ua/lenta/articles/scalable-project-management/.

13. Багтрекеры: как выбрать лучший? [Електронний ресурс]. – Режим доступу: http://qatestlab.com/ru/knowledge-center/QA-Testing-Materials/Bug-Trackers-What-to-Choose/.

14. Как установить Java с помощью Apt-Get в Ubuntu 16.04 [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/java-apt-get-ubuntu-16-04-ru.

15. Клиент – сервер [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%E2%80%94_%D1%81%D0%B5%D1%80%D0%B2%D0%B5% 80.

16. Краткий обзор Spring Security [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/post/203318/.

17. Система отслеживания ошибок [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%

18. Система управления базами данных PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: http://bourabai.kz/dbt/servers/postgresql.htm.

19. СКРАМ – ЭТО ЭФФЕКТИВНОЕ УПРАВЛЕНИЕ ПРОЕКТАМИ [Електронний ресурс] – Режим доступу до ресурсу: https://brainrain.com.ua/%D1%81%D0%BA%D1%80%D0%B0%D0%BC-%D1%8D%D1%82%D0%BE.

20. Bugzilla [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Bugzilla.

21. Java Runtime Environment [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java_Runtime_Environment.

22. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Spring_Framework.

23. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL.

24. Task management [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Task_management.

25. Tracking system [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tracking_system.

26. YouTrack [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/YouTrack.

27. Методичні вказівки до написання техніко-економічного розділу дипломних проектів освітньо-кваліфікаційного рівня «бакалавр» підготовки 6.050102 комп'ютерна інженерія/ І.Р. Паздрій Тернопіль: ТАНГ, 2014. 37 с.

28. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня "Бакалавр" напряму підготовки 6.050102 «Комп'ютерна інженерія» фахового спрямування «Комп'ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2016. 65с.

29. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.