МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____Є.О. Давиденко

«____»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка онлайн-системи керування проєктами

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 − 409.21810911

Студент _____О. С. Дурнєв

підпис

«__»_____2022 р.

Керівник: д-р. техн. наук, доцент _____А. В. Швед

підпис

«__»_____2022 р.

Консультант: канд. техн. наук, доцент _____А. О. Алексєєва

підпис

«__»_____2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедри
_____Є. О. Давиденко
«___»_____2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук
Дурнєву Олександру Сергійовичу .
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи**:**
Розробка онлайн-системи керування проєктами.

Затверджена наказом по ЧНУ ім. П.Могили від « 01 » грудня 2021 р. № 314

2. Строк представлення кваліфікаційної роботи: «___»_____202___р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є онлайн-система керування проєктами.

4. Перелік питань, що підлягають розробці:

- визначення функцій системи;
- моделування;
- створення бази даних;
- проєктування програмного застосунку;
- створення інтерфейсу користувача;
- кодування та тестування системи.

5. Перелік графічних матеріалів:
презентація.

6. Завдання до спеціальної частини

Сформувати перелік вимог до робочого місця, організації та обладнання робочих місць, санітарно-гігієнічних вимог, вимоги щодо освітлення, електробезпека та пожежної безпеки.

7. Консультанти:

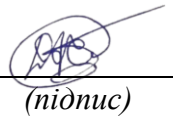
Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А.О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи Д-р. тенх. наук, доцент Швед Альона Володимирівна
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Дурнєв Олександр Сергійович
(прізвище, ім'я, по батькові студента)


(підпис)

Дата видачі завдання «____»_____202___р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема кваліфікаційної роботи:

Розробка онлайн-системи керування проєктами.

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	01.12.2021р.	05.12.2021р	виконано
2.	Оглад літератури за темою роботи	06.12.2021р.	08.12.2021р.	виконано
3.	Складання календарного плану КРБ	09.12.2021р.	10.12.2021р.	
4.	Аналіз предметної області	13.12.2021р.	20.12.2021р.	виконано
5.	Розробка проєктних рішень	11.01.2021р.	22.01.2021р.	виконано
6.	Моделювання та конструювання ПЗ	25.01.2021р.	19.02.2021р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	23.02.2021р.	20.04.2022р.	виконано
8.	Розробка спеціальної частини з охорони праці	26.04.2022р.	09.05.2022р.	виконано
9.	Відгук керівників КРБ	10.05.2022р.	15.05.2022р.	виконано
10.	Оформлення КРБ та презентаціїї	16.05.2022р.	19.05.2022р.	виконано
11.	Попередній захист	20.05.2022р.	20.05.2022р.	виконано
12.	Рецензування	22.05.2022р.	05.06.2022р.	виконано
13.	Завершення оформлення КРБ та презентації	07.06.2022р.	21.06.2022р.	виконано
14.	Захист кваліфікаційної роботи	30.06.2022р.	30.06.2022р.	

Розробив студент Дурнєв Олександр Сергійович

(прізвище, ім'я, по батькові) *(підпис)*

«____»_____202___р.

Керівник роботи Швед Альона Володимирівна

(посада, прізвище, ім'я, по батькові) *(підпис)*

«____»_____202___р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Розробка онлайн-системи керування проєктами»

Студент 409 гр.: Дурнєв Олександр Сергійович

Керівник: д-р. тенх. наук, доцент Швед Альона Володимирівна

Кваліфікаційна робота бакалавра спрямована на розробку онлайн-системи керування проєктами.

Об’єкт роботи процесс розробки онлайн-системи керування проєктами.

Предметом роботи є інформаційні технології та інструментальні засоби розробки програмного забезпечення онлайн-систем керування проєктами.

Метою кваліфікаційної роботи є автоматизація процесу цілеспрямованої координації роботи по результативному та ефективному досягненні поставлених цілей.

Дипломна робота складається зі вступу, фахової частини, яка включає чотири розділи, висновки, списку використаних джерел та 1 додатку. У вступі визначено актуальність теми, мета, об’єкт, предмет, сфера застосування та завдання даної бакалаврської роботи.

У першому розділі розглянуто аналіз предметної сфери автоматизації безнес-процесів керування проєктами. Проведено опис предметної сфери керування проєктами. Проаналізовано та поравняно характеристики різних аналогів онлайн-систем керування проєктами. Визначено специфікацію вимог до програмного забезпечення онлайн-системи керування проєктами.

У дрогому розділі проводиться проєктування та моделування програмного забезпечення онлайн-системи керування проєктами. Розглянуто діаграми прецедентів. Побудовано use case діаграму з декількома варіантами використання для онлайн-системи керування проєктами. Спроєктовано базу даних для онлайн-системи керування проєктами. Досліджено патерни для розробки онлайн-системи керування проєктами.

У третьому розділі наведено вибір засобів розробки онлайн-системи керування проектами. Розглянуто технології розробки клієнтської частини та

серверної частини онлайн-системи керування проєктами. Ознайомлено зі засобами реалізації інформаційного забезпечення. Розглянуто та обрано тип СКБД на якій буде розроблятися база даних онлайн-системи керування проєктами.

У четвертому розділі міститься програмна реалізація програмного забезпечення онлайн-системи керування проєктами. Створено діаграму класів для онлайн-системи керування проєктами. Побудовано діаграму компонентів для онлайн-системи керування прєктами. Написано інструкцію для користувача онлайн-системи керування проєктами.

Висновки мають аналіз виконаної роботи й отримані результати досліджень та розробки. У додатках наведено список основних класів програми.

В цілому бакалаврська робота без додатків містить 54 сторінки, 30 рисунків, 1 таблицю, 24 джерел посилання.

Ключові слова: *керування проєктами, патерн MVC, система керування проєктами, фреймворк laravel, архітектурні патерни, система керування базами даних, база даних MySQL.*

ABSTRACT

of the Bachelor's Thesis

"Development of an online project management system"

Student: Durnev Alexander Sergeevich

Supervisor: dr. technical Sciences, Associate Professor Swede Alyona Vladimirovna

The bachelor's qualification work is aimed at developing an online project management system.

The object of work is the process of developing an online project management system.

The subject of the work is information technologies and tools for software development of online project management systems.

The purpose of qualification work is to automate the process of purposeful coordination of work on the effective and efficient achievement of goals.

Thesis consists of an introduction, a professional part, which includes four sections, conclusions, a list of sources used and 1 appendix. The introduction identifies the relevance of the topic, purpose, object, subject, scope and objectives of this bachelor's thesis.

The first section considers the analysis of the subject area of automation of business processes of project management. The description of the subject area of project management is carried out. The characteristics of various analogues of online project management systems are analyzed and compared. The specification of requirements for the software of the online project management system is determined.

The second section deals with the design and modeling of online project management system software. Diagrams of precedents are considered. A use case diagram with several uses for an online project management system has been built. A database for an online project management system has been designed. Patterns for the development of an online project management system have been studied.

The third section presents the choice of tools for developing an online project management system. The technologies of developing the client part and the server part of the online project management system are considered. Acquainted with the means of implementing information support. The type of DBMS on which the database of the online project management system will be developed is considered and selected.

The fourth section contains the software implementation of the online project management system. A class diagram has been created for the online project management system. The component diagram for the online project management system is built. An instruction manual for an online project management system has been written.

Conclusions have an analysis of the work done and the results of research and development. The appendices list the main classes of the program.

In total, the bachelor's thesis without appendices contains 54 pages 30 figures 1 table 24 reference sources.

Keywords: *project management, MVC pattern, project management system, laravel framework, architectural patterns, database management system, MySQL database.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ КЕРУВАННЯ ПРОЄКТАМИ 7
1.1 Опис предметної сфери керування проєктами 7
1.2 Аналіз та порівняльна характеристика онлайн-системи керування проєктами 8
1.3 Специфікація вимог до програмного забезпечення онлайн-системи керування проєктами 14
Висновки до розділу 1 17
2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ 18
2.1 Діаграма прецедентів 18
2.2 Архітектура онлайн-системи керування проєктами 21
2.3 Проєктування інформаційного забезпечення онлайн-системи керування проєктами 25
Висновки до розділу 2 28
3 ВИБІР ЗАСОБІВ РОЗРОБКИ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ 29
3.1 Вибір технологій розробки клієнтської частини онлайн-системи керування проєктами 29
3.2 Вибір технологій розробки серверної частини онлайн-системи керування проєктами 31
3.3 Вибір засобів реалізації інформаційного забезпечення технологій реалізації бази даних 35
Висновки до розділу 3 38
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ 39

4.1 Діаграма класів 39
4.2 Діаграма компонентів 42
4.3 Інструкція користувача онлайн-системи керування проєктами 44
Висновки до розділу 4 51
ВИСНОВКИ 52
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 53
ДОДАТОК 56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

СКП – система керування проєктами

CRUD – основні операції управління даними, тобто створення (create), читання (read), редагування (update) та видалення (delete)

GUI − графічний інтерфейс користувача (graphical user interface)

ВСТУП

На сьогодні прогрес в світі інформаційних технологій забезпечує людство різними програмними інструментами для полегшення його існування. Прогрес інформаційних технологій не стоїть на місці. Кожного дня з’являються нові програми, комп’ютерні ігри, соціальні мережі, різні онлайн-системи і таке інше. Для того щоб це все розробляти потрібно не тільки апаратне забезпечення, а й відповідні програмні інструменти.

Актуальність теми. В сфері інформаційних технологій розробка системи керування проєктами є актуальною на сьогоднішній день. В сучасному світі інформаційних технологій будь то компанія що створює власне програмне забезпечення чи не велика група розробників яка займається розробкою власного програмного застосунку, потребують сучасні методи керування та контролю за процесом розробки. Системи керування проєктами (СКП) - це програми та онлайн-сервіси, які допомагають планувати завдання, складати розклади, розподіляти ресурси, відстежувати ефективність роботи як особисто кожного спеціаліста так і всієї команди. Керування проєктами це професійна діяльність, заснована на використанні сучасних наукових знань, навичок, методів, засобів та технологій орієнтованих на отримання ефективних результатів. Технічна складова в керуванні проєктами - це реалізація певного технічного проєкту із заданими вимогами якості у задані терміни. Економічна складова керування проєктами пов'язана з певним бюджетом, планом доходів та видатків, рентабельністю та комерціалізацією проєктів.

Об’єкт роботи процес розробки онлайн-системи керування проєктами.

Предметом роботи є інформаційні технології та інструментальні засоби розробки програмного забезпечення онлайн-систем керування проєктами.

Метою роботи є автоматизація процесу цілеспрямованої координації роботи по результативному та ефективному досягнення поставлених цілей. Для досягнення визначеної мети необхідно вирішити наступні **завдання**:

– проаналізувати систему, що розробляється та дослідити існуючі програмні рішення: користувачів, структуру та сценарії роботи системи, засоби апаратної та програмної реалізації;

– визначити архітектуру для проєктування програмного забезпечення;

– змоделювати програмне забезпечення: діаграми використання для системи керування проєктами з урахуванням усіх акторів та діаграми класів, станів та діяльності;

– розробити front-end частину онлайн-системи керування проєктами на базі технологій: HTML, CSS, JavaScript;

– розробити back-end частину онлайн-системи керування проєктами на базі технологій: Laravel, MySQL, PHP;

Сфера застосування. Проєкт може застосовуватись в загальному користувацькому просторі з метою відслідковування виконання кожного завдання, координування роботи кількох людей, слідкування за термінами та зберігання всієї необхідної інформації в одному місці.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ КЕРУВАННЯ ПРОЄКТАМИ

1.1 Опис предметної сфери керування проєктами

Керування проєктами – це цілеспрямована координація роботи по результативному та ефективному досягненні поставлених цілей [5]. Тобто це організація роботи для досягнення запланованого результату у межах затвердженого часу та бюджету. Докладає зусиль, досвіду, інструментів для задоволення вимог, встановлених замовником. Від масштабу, складності цілі залежить спосіб управління процесом. Під час проведення маркетингової компанії чи запуску нового програмного продукту використовуватимуться різні методи.

Діяльність будь-якої компанії можна поділити на проєкти. Їхньою метою є отримання результату у вигляді нового продукту, технології або послуги, що запускається на ринок. В сфері інформаційних технологій набула популярність система керування проєктами. Сьогодні практично всі, починаючи з великих популярних IT компаній та закінчуючи звичайними розробниками що займаються розробкою програмного продукту, користуються різним програмним забезпеченням для керування проєктами. **Програмне забезпечення для керування проєктами** - комплексне програмне забезпечення, що включає додатки для планування завдань, складання розкладу, контролю ціни та управління бюджетом, розподілу ресурсів, спільної роботи, спілкування, швидкого управління, документування та адміністрування системи, яка використовується спільно для управління великими проєктами [6].

Система керування проєктами – це програма, що включає комплекс технологічних та організаційних інструментів для більш ефективної роботи над завданнями та проєктами [7]. Основні цілі програми:

– підвищення ефективності командної роботи;

– поліпшення якості контролю та управління з боку керівників;

– скорочення часу виконання кожного проєкту;

Основні завдання, які допомагають вирішувати СКП:

– оперативний обмін документами та безпека їх зберігання, всі дані зберігаються у хмарі;

– систематизація діяльності команди, проєкт можна розбити на окремі категорії, делегуючи повноваження конкретним виконавцям із контролем часу виконання та якості роботи;

– ефективне спілкування всі учасники проекту мають можливість обмінюватись повідомленнями, розміщувати коментарі до завдань проєкту.

Всі СКП за набором функцій поділяються на два типи:

– Системи календарного планування та контролю (СКПК). Це окремі програми, в яких можна ставити завдання для команди та конкретних фахівців, складати календарні плани та графіки, планувати витрати, формувати звіти в міру досягнення поставленої мети.

– Професійні системи керування проєктами. Це спеціальні комплекси, що складаються з утиліт та модулів, призначених для вирішення різних, у тому числі й специфічних задач. Наприклад, за допомогою інструментів ПСКП можна моделювати та аналізувати ризики, виділяти з команди фахівців, які краще впораються з поставленими завданнями, визначати терміни реалізації проекту за різних фінансових вкладень.

1.2 Аналіз та порівняльна характеристика онлайн-системи керування проєктами

Контроль проєкту за допомогою системи керування проєктами є важливим механізмом любого розробляємого продукту сьогодні. Відповідно, існує попит на розробку програмного забезпечення для автоматизації процесу керування проєктами. Під час розвитку технологій у сфері розробки

програмного забезпечення, з’явилося багато продуктів для автоматизації процесу керування проєктами.

Більшість програмних продуктів мають схожі основні функції. Системи мають функціональні можливості для планування, оформлення завдань, а також їх призначення та для відслідковування робочого процесу. Деякі сервіси мають аутентифікацію та створення так званих дошок, списків і карток, що в свою чергу є проєктами, завданнями і коментарями до них.

Існуючі програмні рішення предметної області:

Аналог 1:

– **Назва**: Trello.

– **Розробник**: Fog Creek Software

– **Архітектура**: Web application, Mobile App (Android, IOS)

– **Мова реалізації**: Node.js, Backbone.js, JavaScript

– **Перелік функцій, характеристик**:

1. Створення своїх дошок, списків та карток.
2. Можливість додавати учасників до дошки.
3. Планування та публікування поточних завдань, систематизування їх та відслідковування за виконанням.
4. Додавання нотаток.
5. Додавання медіа-файлів.
6. Додавання файлів задач.
7. Календар.
8. Чек-листи.
9. Встановлення термінів виконання.
10. Управління правами користувачів.
11. Управління доступом.
12. Призначення термінів та виконавців

– **Переваги**:

1. Великий, постійно оновлюваний перелік поліпшень (Power-Ups), розбитий на категорії.
2. Підходить для невеликих команд.
3. Обмін файлами.
4. Цілі встановлюються для кожного співробітника.
5. Доступний та зрозумілий для розуміння інтерфейс.
6. Можливість підлаштуватися під будь-який проект.
7. Швидкий пошук.
8. Можна підключати сторонні послуги та надбудови.

– **Недоліки**:

1. У безкоштовній версії Trello можна включати обмежену кількість покращень для кожної дошки.
2. Обмежені функціональні можливості.
3. Не підходить для довгострокових проектів – легко заплутатися у величезній кількості дощок та карток.
4. Відсутня діаграма Ганта, що не дозволяє оцінювати виконання проекту за часом.
5. Незручне горизонтальне прокручування - не підходить для маленьких екранів.
6. Для реалізації масштабних проектів базових інструментів замало — доведеться сплачувати передплату.

– **Джерело**: https://www.atlassian.com /ru /software/trello

– **Призначення ПЗ**: керування проєктами.

– **Ролі користувачів системи**: Творець, учасник.

– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.1)**:

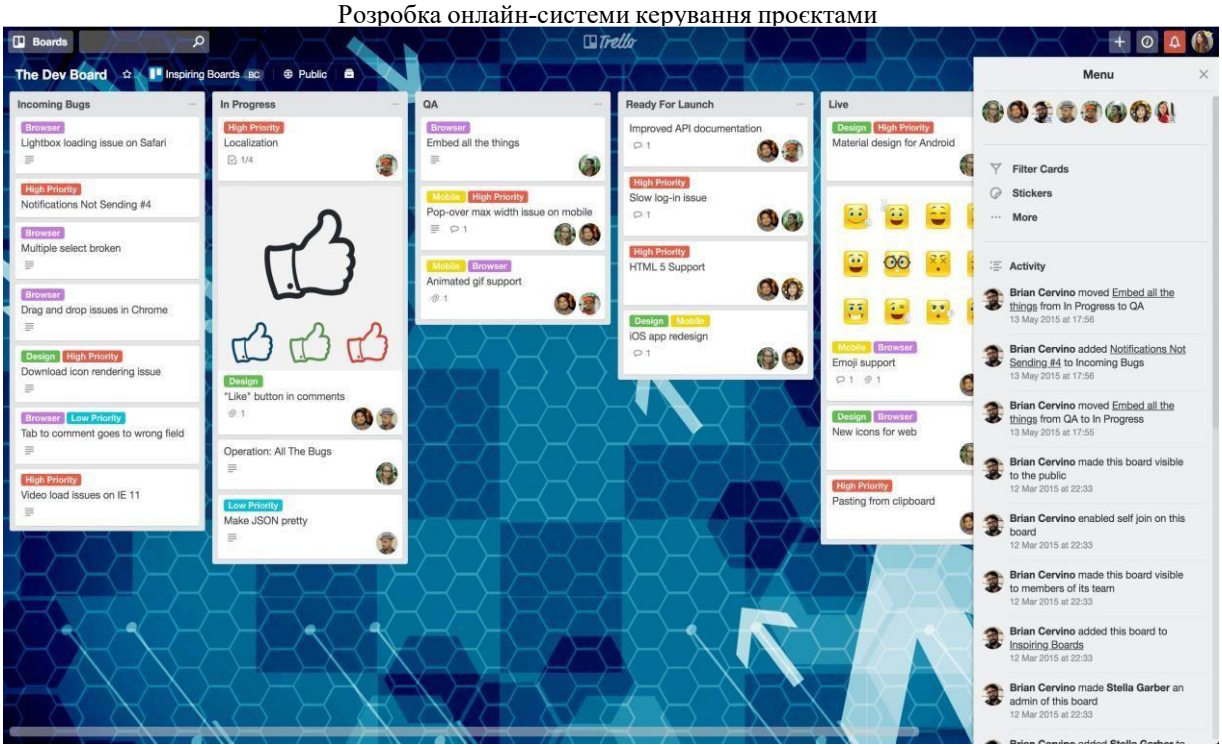


Рисунок 1.1 - Зовнішній вигляд інтерфейсу ПЗ Trello

Аналог 2:

– **Назва**: Jira.

– **Розробник**: Atlassian

– **Архітектура**: Web application, Mobile App (Android, IOS)

– **Мова реалізації**: Java

– **Перелік функцій, характеристик**:

1. Планування та оформлення завдань різного типу.
2. Призначення та відстеження завдань.
3. Складання звітності та аналітика.
4. Відстеження багів - шаблон для управління завданнями та дефектами ПЗ.

– **Переваги**:

1. Широкий функціонал під різні завдання.
2. Великий вибір інтеграцій та інструментів для розширення можливостей.
3. Кастомізація робочого простору.
4. Широкий вибір мов інтерфейсу.

5. Наявність мобільного додатка.

6. Велике ком'юніті та достатню кількість супроводжуючої інформації.

7. Широкий вибір мов інтерфейсу.

– **Недоліки**:

1. Перевантажений інтерфейс.

2. Тяжкий процес впровадження.

3. Безкоштовний тариф лише для невеликих команд (до 10 користувачів).

4. Наявність платних доповнень.

– **Джерело**: https://www.atlassian. com/ software/jira

– **Призначення ПЗ**: керування проєктами.

– **Ролі користувачів системи**: Творець, учасник.

– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.2)**:

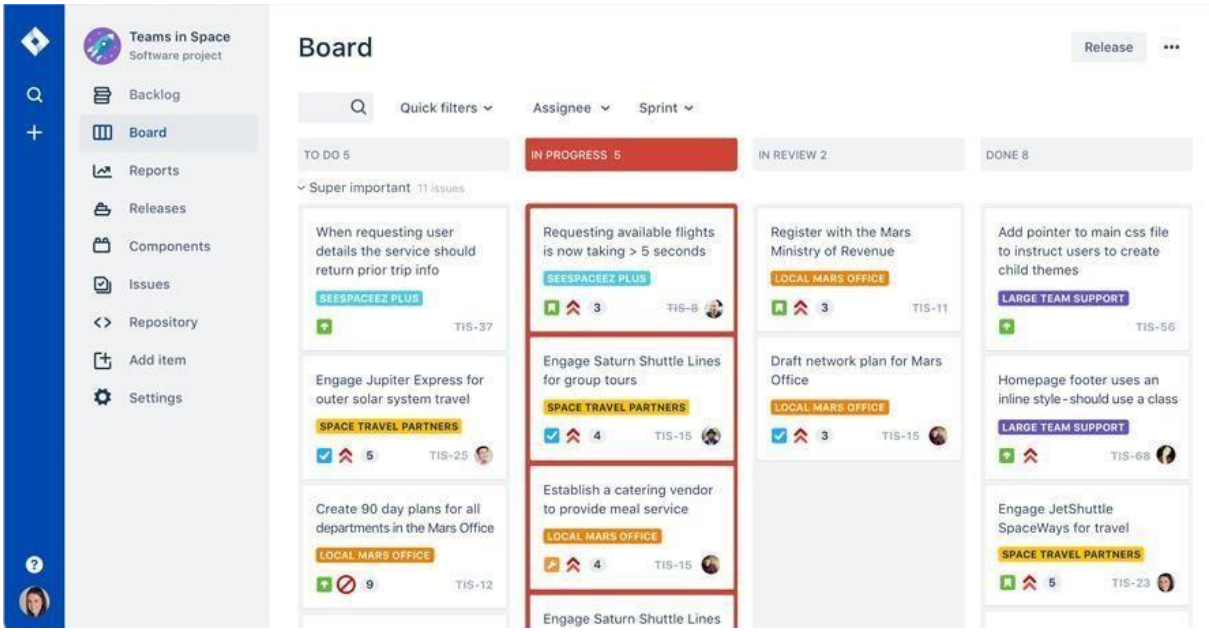


Рисунок 1.2 - Зовнішній вигляд інтерфейсу ПЗ Jira

Аналог 3:

– **Назва**: Asana.

– **Розробник**: Дастін Аарон і Московиц Джастін Розенштейн

– **Архітектура**: Web application, Mobile App (Android, IOS)

– **Мова реалізації**: Ruby

– **Перелік функцій, характеристик**:

1. Детальне відслідковування робочого процесу, гарантуючи чітке та своєчасне виконання поставлених завдань.
2. Планування, створення завдань та можливість відстежувати час їх виконання та формувати звіти.
3. Пошук завдань/проектів, співробітників, тегів у системі.
4. Звітність – повідомлення, графіки, навантаження, експорт проектів.
5. Комунікації – коментарі та лайки до завдань, обговорення із завдань/проектів.
6. Створення календаря контенту.
7. Відстеження власного прогресу.
8. Організування завдань.
9. Голосові завдання - створення завдання з урахуванням розпізнавання голосу.

– **Переваги**:

1. Управління проєктами без класичного застосування електронної пошти.
2. Для запрошення нового співробітника достатньо зазначити його email.
3. Мінімалістичний та інтуїтивно зрозумілий інтерфейс.
4. Можливість деталізації завдань.
5. Галерея готових шаблонів різних відділів (команд) організації — маркетинг, HR, IT, дизайн;
6. Гарячі клавіші та зручний пошук.
7. Інтеграція з хмарними сервісами та сховищами даних.

– **Недоліки**:

1. Неможливо поєднати особисті та загальні календарі.

2. Багато автоматичних оповіщень поштою, які не можна відключити.

3. Немає можливості перетворити список на дошку і навпаки.

4. Відсутня візуалізація прогресу (у відсотках чи діаграмах).

5. Не можна обмінюватися файлами і листуватися з колегами у межах сервісу.

– **Джерело**: https://asana.com/ru

– **Призначення ПЗ**: керування проєктами.

– **Ролі користувачів системи**: Творець, учасник.

– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.3)**:

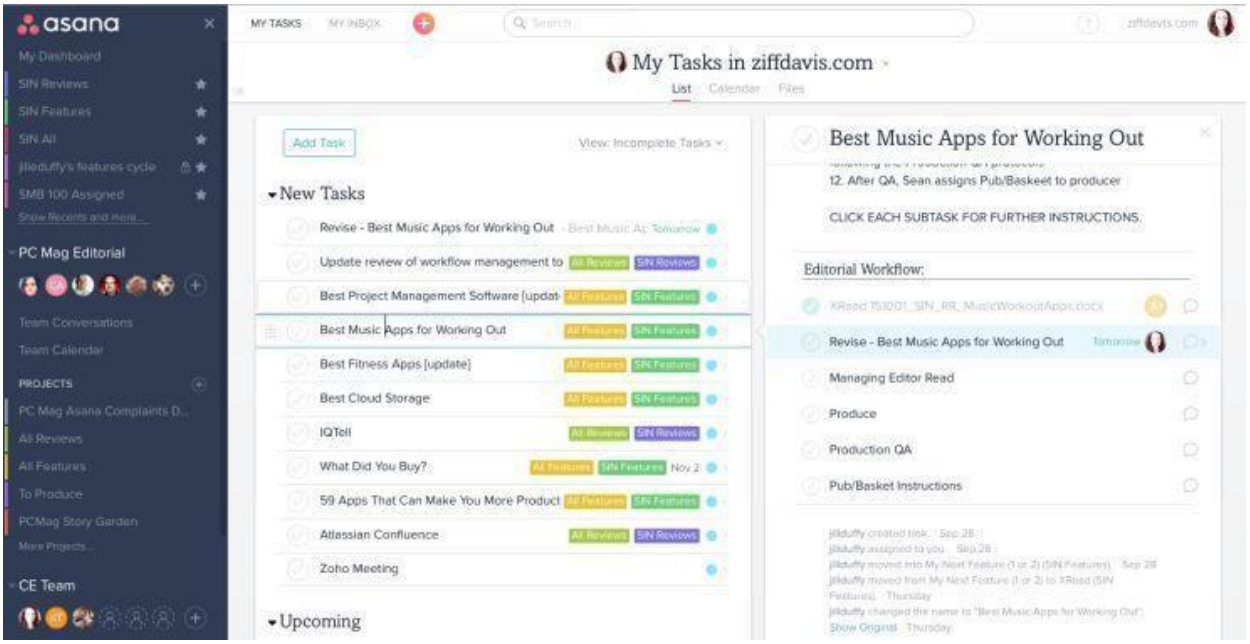


Рисунок 1.3 - Зовнішній вигляд інтерфейсу ПЗ Asana

1.3 Специфікація вимог до програмного забезпечення онлайн-системи керування проєктами

Проаналізувавши функції та характеристики програмних продуктів в області автоматизації керування проєктами, було визначено призначення та функції програмного продукту, що розроблюється.

Система керування проєктами призначена спростити процес відслідковування контролю робочого процесу. На основі аналізу результатів

керування проєкту розробнику проєкту надається можливість покращити процес керування проєкту. Створення дошок проєктів, складання списків із завданнями та додавання карток з самими завданнями має бути зручним, мультиплатформенним та інтуїтивно зрозумілим.

До основних функцій системи відносяться:

1. Реєстрація для ідентифікації користувача.
2. Редагування профілю.
3. Створення дошок, списків та карток.
4. Редагування та видалення створених дошок, списків та карток.
5. Можливість додавання дошки до обраних.
6. Можливість налаштування різного рівня доступу до дошки - тільки для себе, працюючій команді або всім.
7. Копіювання списку, переміщення та відправлення його до архіву.
8. Можливість опису завдання в картці.
9. Можливість зробити картку технічним завданням чек-листамии, посиланнями і файлами.
10. Додавання учасників до дошки.
11. Зміна фону дошки.
12. Додавання учасників до картки.
13. Додавання міток та термінів на картки.
14. Можливість залишити коментар під карткою.
15. Переміщення між колонками.
16. Нагадування про те що потрібно зробити.

Неавторизованому користувачу будь яка взаємодія з системою не доступна, і система не може бути використана ним. Авторизований користувач має можливість використовувати в повному обсязі функціонал системи. Суть програмного продукту не в тому, щоб розподілити ролі користувачам, а надавати можливість як створювати, так і керувати проєктами. Таким чином, користувач має нагоду через один обліковий запис

в системі використовувати повний функціонал системи.

Умовно ролі користувача можна поділити на дві частини: творець, учасник. Творець має право створювати та редагувати дошку, складувати до неї списки, додавати до списку картки зі завданнями та прикріпляти до них учасників. Творець має можливість ставити терміни до виконання завдань та надавати рівні доступу учасникам до карток та всієї дошки. Учасник, якому надали доступ до проєкту, бачить список карток із завданнями які необхідно виконати.

Сценарії роботи системи наступні:

1. Вхід в систему – створення дошки – створення списку – додавання карток до списку – формулювання завдань в картках – встановлення термінів на виконання завдань – додавання учасників до дошки – розподілення учасників по карткам – слідкування за процесом роботи.
2. Вхід в систему – перегляд списку карток із завданнями – перегляд термінів виконання завдань – виконання завдань.
3. Вхід в систему – перегляд карток інших учасників – аналіз процесу виконання завдань – редагування карток учасників.
4. Вхід в систему – пошук картки учасника, що цікавить зі заданого списку – видалення картки – перегляд результату.
5. Вхід в систему – перегляд заданого списку – додавання нової картки до списку – опис завдання нової картки - планування часу виконання завдання, виставлення терміну виконання завдання картки – прикріплення учасника до новоствореної картки – перегляд результатів.
6. Реєстрація в системі – вхід в систему – приєднання до дошки - пошук списку карток із завданнями, які потрібно виконати – створення власної дошки із власним списком карток - додавання завдання в картку.

Сценарій під номером 6 показує, що користувачу при реєстрації не наділяється конкретна роль. Користувач має доступ до всіх функцій системи. Наприклад, працівник реєструється в системі, керівник проєкту дає доступ до

своєї вже існуючої дошки з учасниками та завдання які працівник повинен виконати, працівник в свою чергу для полегшення виконання роботи створює свою власну дошку із списком карток.

Таким чином, система є універсальною як для творця дошки, так і для учасника, який має можливість створити власну дошку.

Висновки до розділу 1

В першому розділі було розглянуто опис предметної сфери керування проєктами. Визначено актуальність теми кваліфікаційної роботи в сфері інформаційних технологій. Розглянуто поняття керування проєктами.

Проаналізовано та порівняно аналоги онлайн-системи керування проєктами. Прикладами існуючого програмного рішення предметної області стали застосунки Trello, Asana та Jiro. В аналогах порівнювалось назва, розробник, архітектура, мова реалізації, перелік функцій, переваги та недоліки, призначення програмного забезпечення, ролі користувачів та зовнішній вигляд інтерфейсу.

Сформульовано специфікацію вимог до програмного забезпечення онлайн-системи керування проєктами. Було визначено призначення та функції програмного продукту, що розроблюється. Описано сценарії роботи системи. Визначено ролі користувача та основні функції системи.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ

2.1 Діаграма прецедентів

Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами вони ж користувачі і прецедентами вони ж випадки використання та описує функціональні аспекти системи тобто бізнес логіку. Діаграми прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проєкту і потенційними користувачами. Діаграми прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проєкту на всіх його фазах, а саме:

- зародження;
- дизайн;
- програмування;
- тестування;
- документування.

Добре продумані і завершені специфікації прецедентів легко перевтілюються у тестові випадки.

Діаграма **варіантів використання** (англ. use-case diagram) - діаграма, що описує, який функціонал програмної системи, що розробляється, доступний кожній групі користувачів [8].

Варіант використання (use case) — послідовність дій, які система чи інша сутність можуть виконувати у процесі взаємодії з акторами (у багатьох джерелах варіант використання називають прецедентом, не треба лякатися і плутатися – це одне й те саме) [9].

Елементи діаграми варіантів використання:

- актор він же користувач;

– прецедент тобто випадок використання, дія що позначається овалом;

– граничні межі системи які охоплюють усі випадки використання у системі та позначається прямокутником.

Елементи взаємодії діаграми варіантів використання:

– використовує, користувач виконує дію;

– включає, один прецедент використовує іншого;

– розширює, представлення дочірніх прецедентів;

– вимагає, наступний прецедент вимагає виконання попереднього;

– схожий, прецеденти подібні, але описують різну функціональність;

– рівнозначний, подібна функціональність, але користувач сприймає, як різну.

Всі вище описані елементи діаграми варіантів використання зображено на рисунку 2.1.

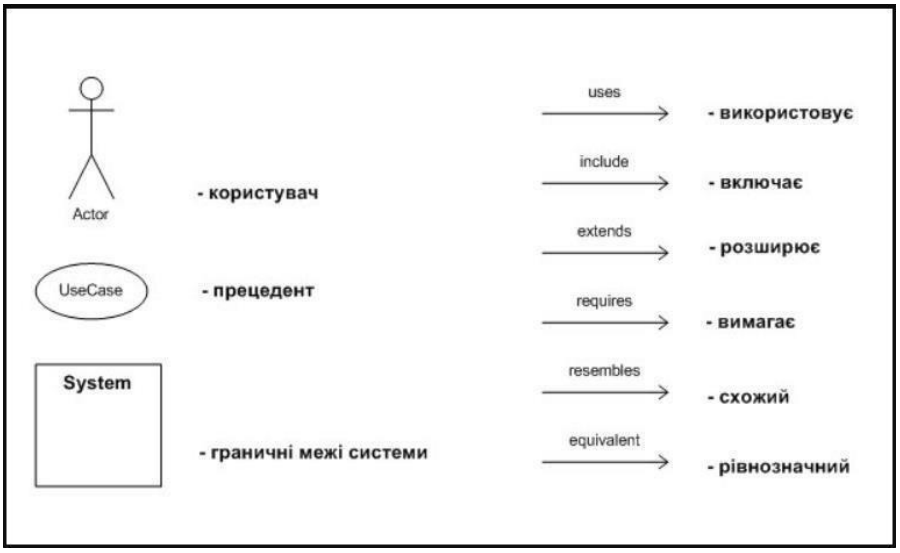


Рисунок 2.1 – Елементи діаграми варіантів використання

Для розробки даного проєкту побудуємо Usecase діаграму з декількома варіантами використання системи. Спочатку зобразимо чотири групи акторів вони ж є користувачі системи, це адміністратор, творець, учасник та користувач. Оскільки кожна група користувачів має певні функції даної

системи, створимо варіанти використання. Щоб створити варіанти використання для акторів потрібно визначити особисто для кожної групи акторів певні функції. Почнемо з сутності користувач:

– авторизація;
– керування дошкою.

Функція керування дошкою в свою чергу розширюється на такі функції:

– редагувати дошку;
– створити дошку;
– встановити термін виконання картки;
– пошук дошки, картки.

Далі опишемо адміністратора, він має такі функції:

– робота з користувачами;
– налаштування підключення.

Потім опишемо творця:

– перевірити картку, що включає додаткову функцію переглянути картку;
– керування учасниками.

Керування учасниками в свою чергу розширюється на такі функції:

– редагувати учасника;
– додати учасника;
– прикріпити учасника до картки;
– надати права доступу учаснику.

За подібною логікою опишемо функції учасника:

– виконати картку.

Функція актора виконати картку включає додаткову функцію переглянути картку.

Всі вище описані актори та їх функцію зображено на рисунку 2.2.

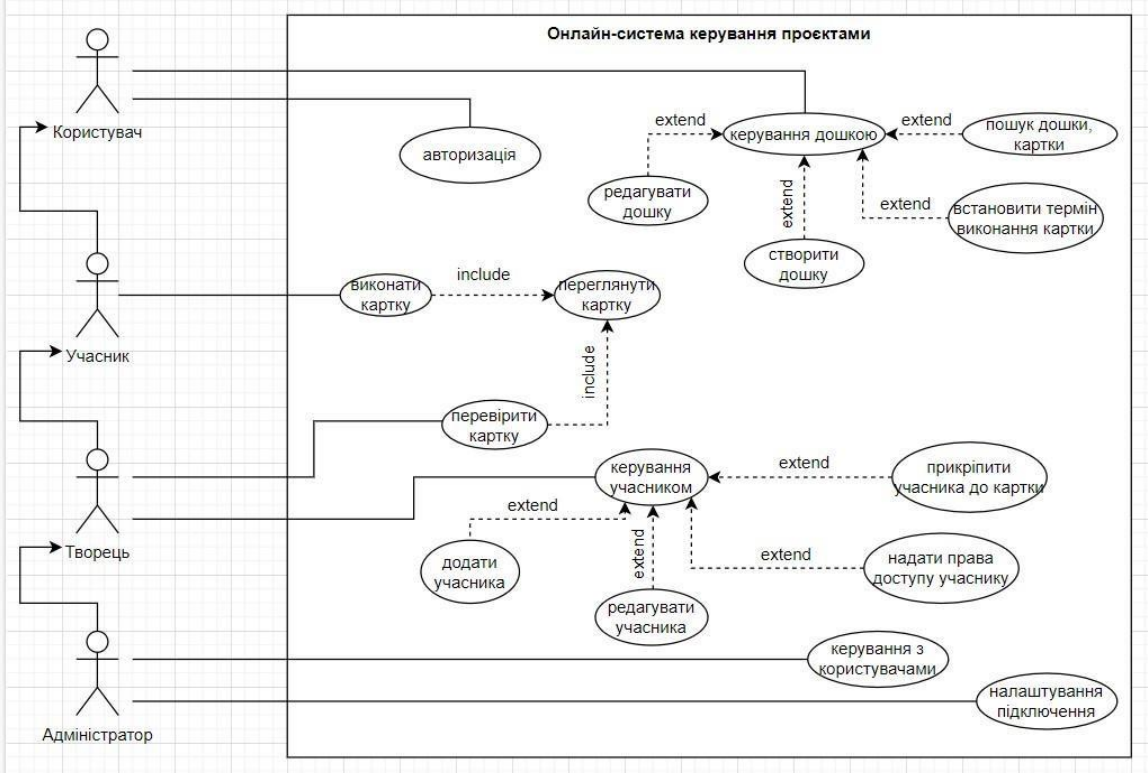


Рисунок 2.2 – Діаграма варіантів використання онлайн-системи керування проєктамии

На даній діаграмі варіантів використання зображено в цілому логіку роботи онлайн-системи керування проєктами.

2.2 Архітектура онлайн-системи керування проєктами

Для створення будь якого програмного забезпечення, будь то невелика за розміром програма для введення списку літератури чи динамічний сайт інтернет магазину, розробка на програмному рівні потребує алгоритм дій. Сучасні методи розробки на програмному рівні передбачують так звані патерни, їх використовують для вирішення тривіальних завдань.

Паттерни - це спосіб побудови (структуризації) програмного коду спеціальним чином [10]. Найчастіше вони використовуються програмістами у тому, щоб вирішити якусь проблему. У цьому випадку передбачається, що існує певний перелік загальних формалізованих проблем, причому ці проблеми трапляються відносно часто. Тому саме патерни, реалізують способи вирішення цих проблем.

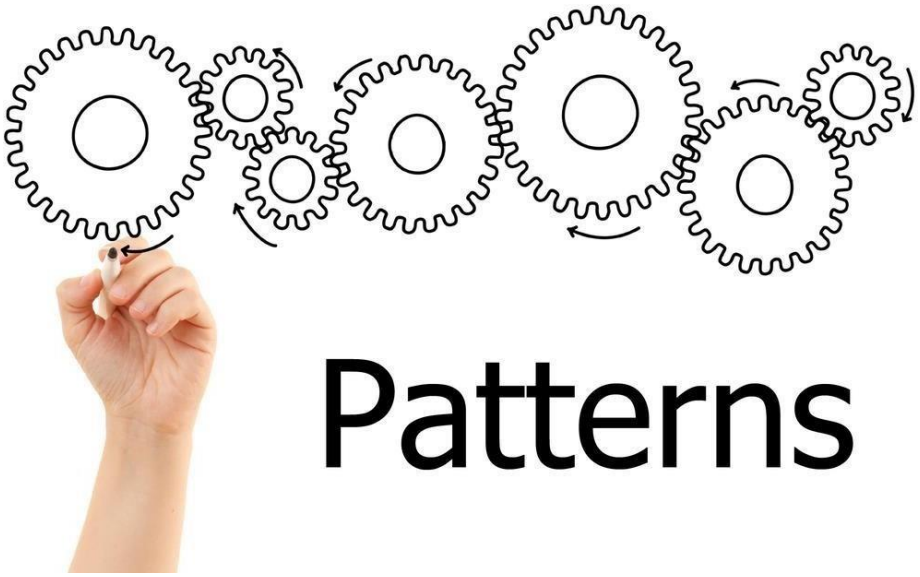


Рисунок 2.3 - Паттерни

На сьогодні існує безліч різних патернів, які вирішують різноманітні питання і виконують різні завдання. Тому їх класифікують за групами. В основі такої класифікації та об'єднання в групи лежатимуть поставлені цілі та розв'язувані завдання. В цілому існує три типи шаблонів, це породжувальні, структурні та поведінкові. Залежно від того, відноситься патерн до об'єктів або класів, існують інші класифікації патернів.

В даній роботі в розробці проєкту буде розглянуто архітектурній патерн. **Архітектурний патерн** це концепція, яка вирішує і окреслює деякі істотні взаємопов'язані елементи архітектури програмного забезпечення. Архітектурний патерн вирішує проблеми, пов'язані з архітектурними стилями [11]. Приклади проблем, які вирішують архітектурні патерни:

– як багато рівнів (tiers) буде у клієнт-серверній архітектурі;

– як рівні взаємодіятимуть;

– які модулі найвищого рівня будуть у сервіс-орієнтованій архітектурі.

Приклади архітектурних патернів:

– MVC (Model-View-Controller)

– EBI (Entity-Boundary-Interactor)

– Трирівневий (3-tier)

Розглянемо архітектурний патерн MVC, даний патер буде використовуватися в подальшій розробці онлайн-системи керування проєктами.

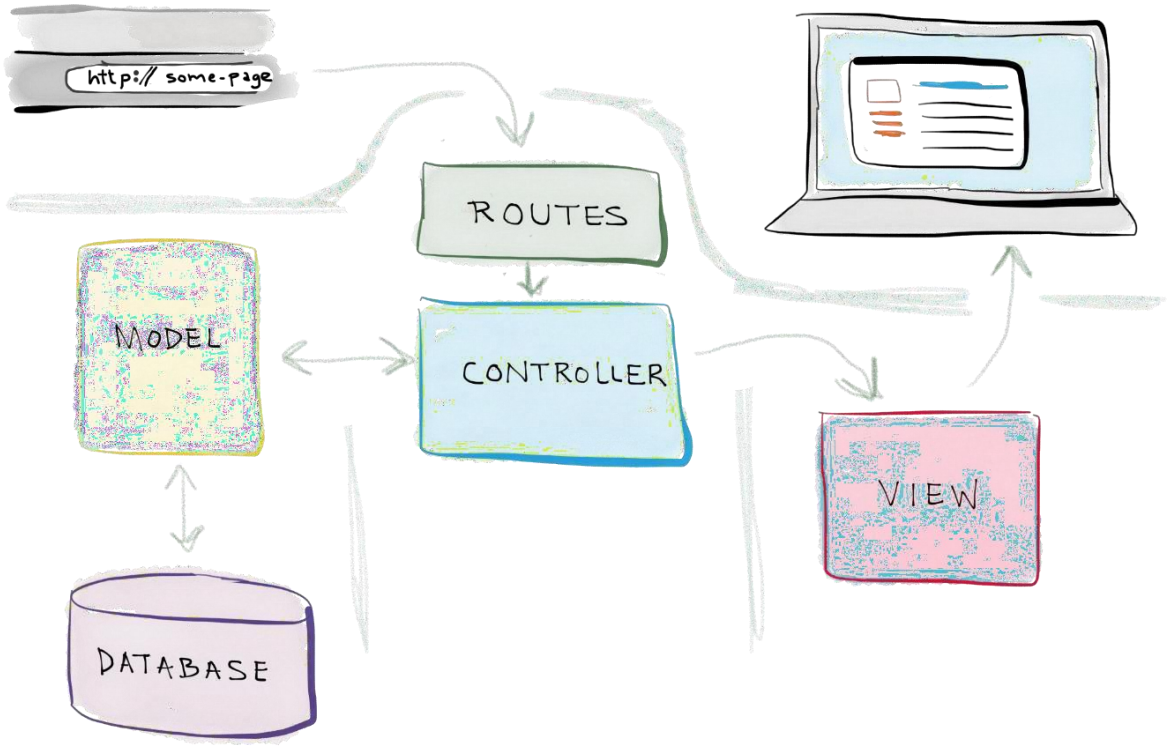


Рисунок 2.4 – Загальний вид MVC

MVC (скорочення від Model-View-Controller) - це архітектурний патерн, який ділить модулі на три групи:

– Model (модель). Отримує дані від контролера, виконує необхідні операції та передає їх у вигляд.

– View (вид або уявлення). Отримує дані від моделі та виводить їх для користувача.

– Controller (контролер). Обробляє дії користувача, перевіряє отримані дані та передає їх моделі [12]..

Модель може бути представлена об'єктом чи структурою об'єктів.

Особливості MVC:

– View використовує об'єкти даних безпосередньо з Model для відображення цих даних.

– Коли дані Model змінюються, спрацьовує подія, негайно оновлює View.

– Один View зазвичай прив'язаний до одного Controller.

– Кожен екран може мати кілька пар View-Controller.

– Controller може бути пов'язаний з кількома переглядами.

Отже цей патерн розробки потрібен у тому, щоб розділити логічні частини програми та створювати їх окремо друг від друга. Тобто писати незалежні блоки коду, які можна як завгодно міняти, не торкаючись інших.

Для того щоб використати в проєкті патерн MVC потрібно визначити моделі, контролер та вид. Моделі це частина програми що працює з даними БД, тому визначемо такі моделі:

– Board – працює з даними дошок.

– Card - працює з даними карток.

– Comment - працює з даними коментарів.

– User - працює з даними користувачів.

– List - працює з даними списків.

Далі для взаємодії з вище визначеними моделями визначимо такі контролери:

– BoardsController.

– CardsController.

– CommentsController.

– UsersController.

– ListsController.

Вище було описано лише основні моделі та контролери, тому в подальшій розробці системи можливо будуть ще додані додаткові моделі та контролери. Оскільки вид це вже реалізована html сторінка тому він буде визначений та реалізований вже на стадії самої розробки проєкту тобто на програмному рівні.

2.3 Проєктування інформаційного забезпечення онлайн-системи керування проєктами

Зберігання та обробка даних – mission-critical завдання будь-якої комп'ютерної системи. Для вирішення даного завдання існує база даних.

База даних є місцем для зберігання даних. Використовується у тому числі у клієнт-серверній архітектурі. Це все інтернет-магазини, сайти кінотеатрів або авіаквитків і таке інше. Тобто клієнт робить замовлення, а система зберігає його дані у базі. Клієнт це людина що робить запит, а система це сервер що обробляє його. В сфері комп’ютерних технологій БД це сховище, куди програма складає свої дані. Якщо програма невелика, окрема база не потрібна. Але потім це стає зручніше та вигідніше з погляду пам'яті.

Такий спосіб зберігання інформації називається реляційна база даних. **Реляційна база даних** - набір таблиць, що зберігаються в одному просторі. Така база даних побудована на основі реляційної моделі, тобто БД, що має табличний спосіб вистави даних, а на зовнішньому рівні, що задається набором однорідних таблиць [13]. Кожний об’єкт записується рядком у таблиці. Рядок називається записом. Запис складається з полів різного типу. Таких таблиць може бути безліч і саме цим і зручна така модель зберігання даних. Замість того щоб створювати велику таблицю з великою кількістю полів та інформацією всередині, краще створити папку з окремими файлами які будуть зберігати таблиці. Подібний алгоритм міститься в деревовидній системи керування каталогами в віндоус.

Після ознайомлення з поняттям бази даних перейдемо до її проєктування. Проєктувати базу даних даного застосунку будемо в програмному забезпечені DBDesigner. DBDesigner – це вільно поширена CASE-система, призначена для проєктування, моделювання, створення та підтримки інформаційних систем [14].

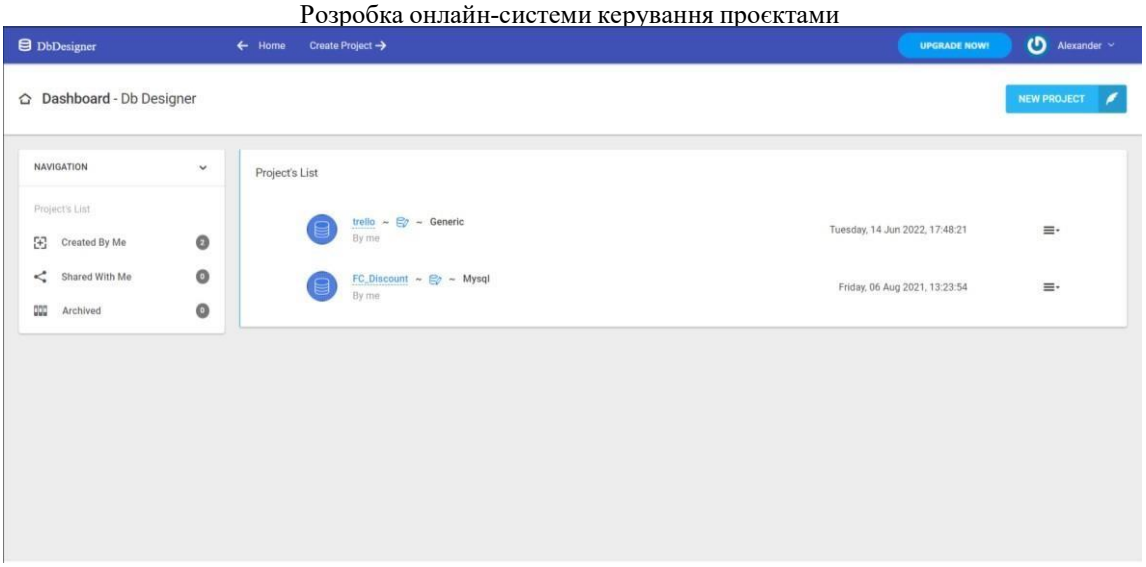


Рисунок 2.5 – Інтерфейс DBDesigner

Спочатку створимо таблиці в яких будуть міститися дані про користувачів, дошки, картки, коментарі, ролі, контакти, картинки для фону користувача та дошки. Таблиця users буде мати такі поля:

– id – первинний ключ;
– avatar – аватар користувача;
– name – ім’я користувача;
– email – пошта користувача;
– password – пароль користувача.

Таблиця boards буде мати наступні поля:

– id – первинний ключ;
– user_id – зовнішній ключ, що посилається на таблицю users;
– name - ім’я дошки;
– picture – задній фон дошки.

Таблиця list буде мати наступні поля:

– id – первинний ключ;
– board_id – зовнішній ключ, що посилається на таблицю boards;
– name - назва списку;

Таблиця cards з такими поля:

– id - первинний ключ;

– list_id – зовнішній ключ, що посилається на таблицю list;
– name - ім’я картки;
– description – вміст картки.

Таблиця comments з такими поля:

– id - первинний ключ;
– card_id - зовнішній ключ, що посилається на таблицю cards;
– content – вміст коментара.

Таблиця storage_file з такими поля:

– id - первинний ключ;
– picture - картинка.

Таблиця storage_board з такими поля:

– id - первинний ключ;
– board_id - зовнішній ключ, що посилається на таблицю boards;
– picture_id - зовнішній ключ, що посилається на таблицю storage_file.

При проєктувані таблиць було також створено первинні ключі. **Первинний ключ** – це один з потенційних ключів відносини, вибраний як основний ключ (або ключ за замовчуванням) [15]. Коротко кажучи він потрібен для зберігання унікальності та взаємодіє з зовнішнім ключем.

В усіх вище створених таблицях було створено поля зі зовнішніми ключами які в свою чергу посилаються на інші таблиці. **Зовнішній ключ** – це ключ, який використовується для зв'язування двох таблиць один з одним, тобто це поле (або набір полів) в одній таблиці, яке посилається на первинний ключ в іншій таблиці.

Таблиця, що містить зовнішній ключ, називається дочірньою таблицею, а таблиця, що містить вибраний ключ, називається посилання або батьківської таблиці. Після пройдених операцій маємо наступний вигляд спроєктованої вже бази даних (рис.2.6).

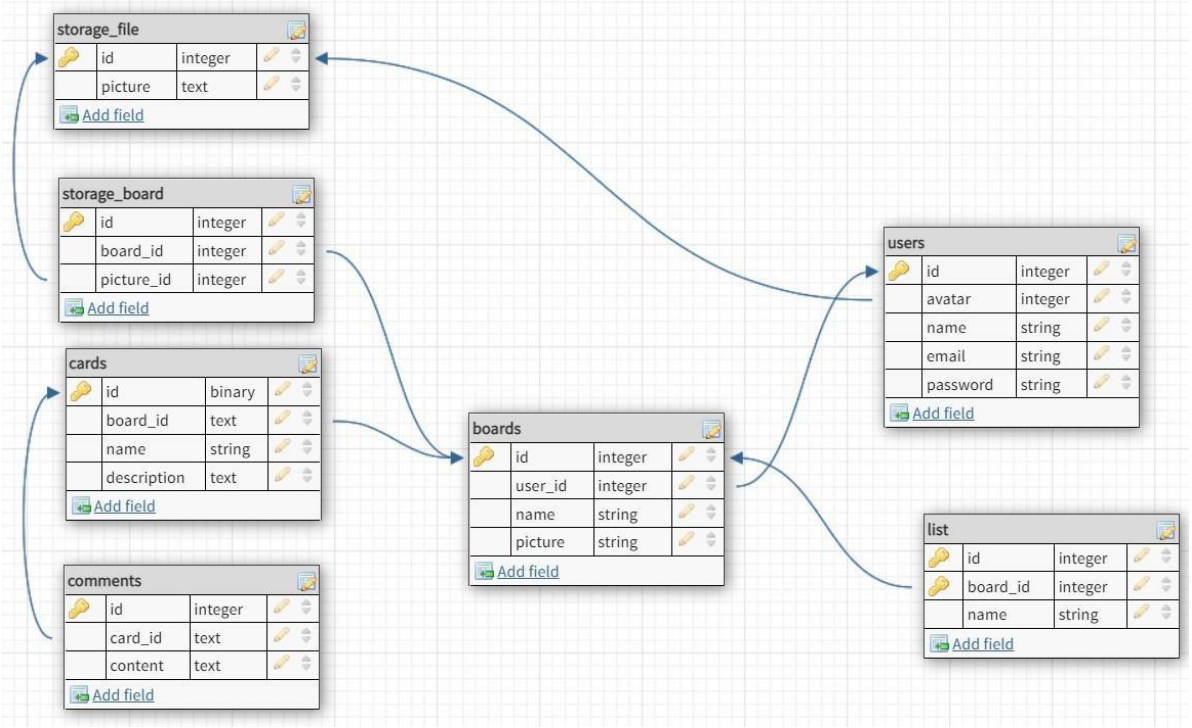


Рисунок 2.6 – Спроєктована БД онлайн-системи керування проектами

В подальшій розробці проєкту вище створена модель бази даних нашої онлайн-системи керування проєктами може змінюватися, але основні таблиці було створено які будуть вже в подальшому більш детальніше реалізовано на програмному рівні.

Висновки до розділу 2

В другому розділі було розглянуто діаграму прецидентів, а саме діаграму виріантів використання. Побудовано use case діаграму з декількома варіантами використання для онлайн-системи керування проєктами. Ознайомлено з програмним забезпеченням DBDesigner. Спроєктовано базу даних для онлайн-системи керування проєктами в програмному застосунку DBDesigner. Дано визначення для реляційних баз даних. Досліджено та обрано патерн MVC для розробки онлайн-системи керування проєктами. Визначено моделі та контролери для онлайн-системи керування проєктами.

3 ВИБІР ЗАСОБІВ РОЗРОБКИ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ

3.1 Вибір технологій розробки клієнтської частини онлайн-системи керування проєктами

Frontend — це розробка інтерфейсу користувача та функцій, які працюють на клієнтській стороні веб-сайту або програми [16]. Це все, що бачить користувач, відкриваючи веб-сторінку із чим він взаємодіє.

Компоненти frontend-розробки:

– **HTML** (HyperText Markup Language) – мова розмітки документів для створення структури сторінки: заголовки, абзаци, списки тощо [16].

– **CSS** (Cascading Style Sheets) — мова для опису та стилізації зовнішнього вигляду документа [16]. Завдяки CSS-коду браузер розуміє, як відображати елементи. CSS задає кольори та параметри шрифтів, визначає, як розташовуватимуться різні блоки сайту, і так далі. Ще він дозволяє виводити той самий документ у різних стилях, наприклад, для друку (звичайним або шрифтом Брайля), виведення передачі на екран або читання голосом.

– **JavaScript** — це мова, яку створювали, щоб оживити веб-сторінки. Його завдання – реагувати на дії користувача, обробляти кліки мишкою, переміщення курсору, натискання клавіш. Ще він посилає запити на сервер і завантажує дані без перезавантаження сторінки, дозволяє вводити повідомлення та багато іншого.

CSS-препроцесори - надбудови над самим CSS, що відкривають нові можливості мови і роблять процес роботи простішим і доступнішим для розробника за допомогою спеціальних конструкцій. Програмісти називають їх «синтаксичним цукром».

«Синтаксичний цукор» - конструкції, які не вносять нічого принципово нового в технологію, але роблять роботу з нею зручнішою, простішою і людянішою [17].

Тому використовувати один із двох популярних препроцесорів — зручне рішення:

– SASS - компілюється за допомогою Ruby;
– LESS – використовує JavaScript або Node.js.

Схеми іменування CSS допомагають написання правильного CSS-коду методології іменування:

– БЕМ;
– CSS Modules;
– SMACSS;
– OOCSS.

Популярність препроцесорів для CSS вплинула створення таких технологій і для HTML. Їхня мета абсолютно ідентична: спростити написання коду і скоротити час на нього, врятувати програміста від багаторазових повторень рядків і однакових тегів. Ось найвідоміші з HTML-препроцесорів:

– Haml;
– Jade;
– Slim.

Автоматизація JavaScript - таку можливість у сучасній розробці надають менеджери задач Gulp та Grunt, які працюють через NPM – Node Package Manager. Також зручно працювати з збирачом модулів Webpack.

Завдяки цим інструментам зручно автоматизувати складання деяких файлів.

Універсального JavaScript-фреймворку, який підходив би під вимоги будь-якого проекту, немає і бути не може зі зрозумілих причин: кожне

завдання індивідуальне. Тому будемо використовувати один із трьох найбільш популярних та затребуваних:

– React;
– Angular.js;
– Vue.js.

Існують звичайні редактори для вихідного коду та IDE - інтегровані середовища розробки з безліччю додаткових функцій. Обидва типи програм бувають безкоштовними та комерційними.

Безкоштовні редактори:

– Atom;
– Visual Studio Code;
– Brackets;
– Sublime Text;
– Notepad++,
– Netbeans.

Платні IDE:

– WebStorm;
– Zend Studio та Aptana;
– Komodo IDE.

Після огляду технологій розробки клієнтської частини, було вирішено front-end частину онлайн-системи керування проєктами писати в інтегрованому середовищі розробки IDE WebStorm з використанням таких компонентів як CSS, HTML, JavaScript та фреймворку Vue.js.

3.2 Вибір технологій розробки серверної частини онлайн-системи керування проєктами

Backend розробка - це набір апаратно-програмних засобів, за допомогою яких реалізовано логіку роботи сайту [16]. Просто кажучи, це те, що приховано від очей користувача і відбувається поза його браузером та

комп'ютером. Backend-розробник застосовує інструменти, що доступні на його сервері. Він має право вибрати будь-яку з універсальних мов програмування, наприклад, Ruby, PHP, Python, Java. Все залежить від конкретного проєкту та завдання замовника.

В нашому випадку було обрано мову програмування PHP, адже в даній роботі описується процес розробки онлайн-системи. **PHP** (рекурсивний акронім словосполучення PHP: Hypertext Preprocessor) – це поширена мова програмування загального призначення з відкритим вихідним кодом [18].

Рисунок 3.1 - PHP

PHP спеціально сконструйований для веб-розробок, і його код може впроваджуватися безпосередньо в HTML.

Замість рутинного виведення HTML-коду командами мови (як це відбувається, наприклад, в Perl або C), скрипт PHP містить HTML із вбудованим кодом. Код PHP відокремлюється спеціальними початковими та кінцевими тегами <?php і ?>, які дозволяють "перемикатися" в "PHP-режим" і виходити з нього.

PHP відрізняється від JavaScript тим, що PHP-скрипти виконуються на сервері та генерують HTML, який надсилається клієнту.

Розробляти на програмному рівні будь-який застосунок на чистому PHP не дуже зручно. Для зручності та в подальшому полегшення написання

коду програмісти використовують фреймворки. **Фреймворк** - програмна платформа, що визначає структуру програмної системи; програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекту [19]. Найпопулярніші PHP-фреймворк це Yii, CodeIgniter, Symfony, Laravel та Phalcon PHP. Використовувати в розробці проєкту вирішено PHP-фреймворк Laravel.

Laravel - це безкоштовний PHP-фреймворк з відкритим вихідним кодом, спеціально розроблений для створення складних сайтів та веб-додатків [20]. Дозволяє спростити автентифікацію, маршрутизацію, сесії, кешування, архітектуру, роботу з базою даних. Laravel розробили як помічник при створенні складних веб-ресурсів та програм. З його допомогою фахівці спрощують процес аутентифікації, а також роботу з БД, кешування, сесії, структуру застосування, маршрутизацію та інші не менш важливі процеси.

Рисунок 3.2 - Laravel

Можливостей у платформи Laravel є чимало. Одна з них – побудова логічної архітектури для проектів будь-якої складності та типу. Платформа характеризується:

– високою продуктивністю;
– можливістю інтеграції з іншими платформами та бібліотеками;
– чималою кількістю цікавих можливостей для розробників сайтів та додатків.

Тепер потрібно обрати середовище розробки.

Інтегроване середовище розробки, ICP (англ. Integrated development environment - IDE), також єдине середовище розробки, ECP - комплекс програмних засобів, використовуваний програмістами для розробки програмного забезпечення [21].

IDE (інтегроване середовище розробки) – це не просто текстовий редактор. У той час як текстові редактори для коду, такі як Sublime або Atom, пропонують безліч зручних функцій, таких як підсвічування синтаксису, інтерфейс і розширені засоби навігації, що дозволяють настроювати, вони дозволяють тільки писати код [22]. Для створення функціонуючих програм, як мінімум, потрібен компілятор і відладчик. Також існує безліч IDE найпопулярніші це Microsoft Visual Studio, PyCharm, IntelliJ IDEA, Xcode, PhpStorm.

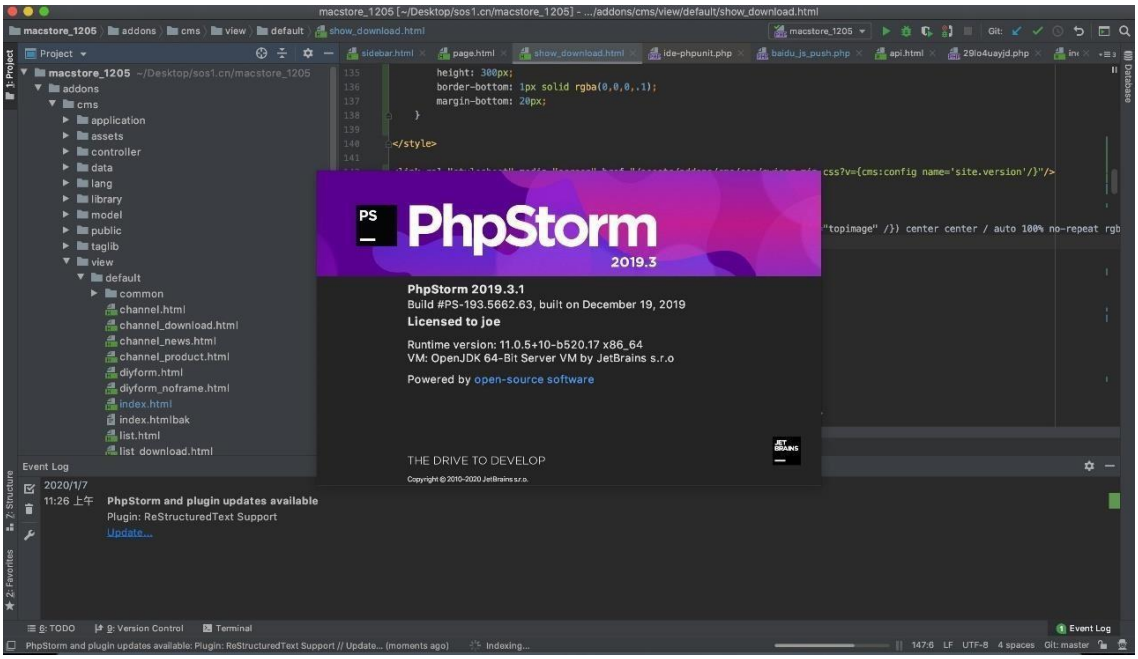


Рисунок 3.3 – Інтерфейс PhpStorm.

Розробляти нашу онлайн-систему будемо в інтегрованій середовищі розробки PhpStorm. IDE забезпечує першокласну підтримку PHP 5.3–8.1, миттєво запобігає помилкам, надає точне автодоповнення та безпечні рефакторинги, а також можливість редагування коду на HTML, CSS та JavaScript. Також PhpStorm підтримує роботу з базами даних MySQL, PostgreSQL, SQLite та SQL Server.

3.3 Вибір засобів реалізації інформаційного забезпечення технологій реалізації бази даних

Майже кожен сучасний веб-додаток взаємодіє з базою даних. Для правильної роботи сайту потрібні не лише файли з кодом сторінок, а й бази даних. Для взаємодії з БД застосовуються системи керування базами даних (СКБД).

Система керування базами даних, скор. СКБД (англ. Database Management System, скор. DBMS) - сукупність програмних та лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням та використанням баз даних [23].

Головна функція СКБД – це керування даними (які може бути як у зовнішній, і у оперативної пам'яті). СКБД обов'язково підтримує мови баз даних, а також відповідає за копіювання та відновлення інформації після будь-яких збоїв.

Для керування реляційними базами даних використовується спеціальна мова програмування – SQL. Скорочення розшифровується як «Structured query language», у перекладі українською – «мова структурованих запитів». Команди, що використовуються в SQL, поділяються на:

– маніпулюючі даними;
– визначальні дані;
– керуючі даними.

Схема роботи з базою даних виглядає так (рис. 3.4):

Рисунок 3.4 – Схема роботи з БД

При виборі типу СКБД слід, перш за все, виходити з типу розв'язуваних завдань, типів даних, перспектив зростання і масштабування. Тобто кожний тип СКБД використовується для певної задачі реалізувати базу даних.

Таблиця 3.1 – Використання СКБД

Тип СКБД	**Коли використовувати**	**Приклади популярних СКБД**
Реляційні	Потрібна транзакційність; висока нормалізація; велика частка операцій на вставку	Oracle, MySQL, Microsoft SQL Server, PostgreSQL
Ключ-значення	Завдання кешування та брокери повідомлень	Redis, Memcached
Документні	Для зберігання об'єктів у одній сутності, але з різною структурою; зберігання структур на основі JSON	CouchDB, MongoDB, Amazon DocumentDB
Графові	Завдання подібні до соціальних мереж; системи оцінок та рекомендацій	Neo4j, Amazon Neptune, InfiniteGraph, InfoGrid
Колонкові	Сховища даних; вибірки із складними аналітичними обчисленнями; кількість рядків у таблиці перевищує сотні мільйонів	Vertica, ClickHouse, Google BigTable, Sybase \ SAP IQ, InfoBright, Cassandra

З наведеної вище таблиці можна зробити висновок що для розробки бази даних для нашої онлайн-системи підходить тип СКБД реляційної бази

даних. В цьому проєкті будемо реалізовувати базу даних онлайн-системи за допомогою саме бази даних MySQL.

MySQL є однією з найпопулярніших і найпоширеніших СУБД, яка використовується в багатьох компаніях (наприклад, Facebook, Wikipedia, Twitter, LinkedIn, Alibaba та інших). MySQL є реляційною СУБД, яка відноситься до вільного програмного забезпечення: вона поширюється на умовах GNU Public License. Як правило, цю систему управління базами даних визначають як хорошу, швидку та гнучку, рекомендовану до застосування у невеликих або середніх проектах.

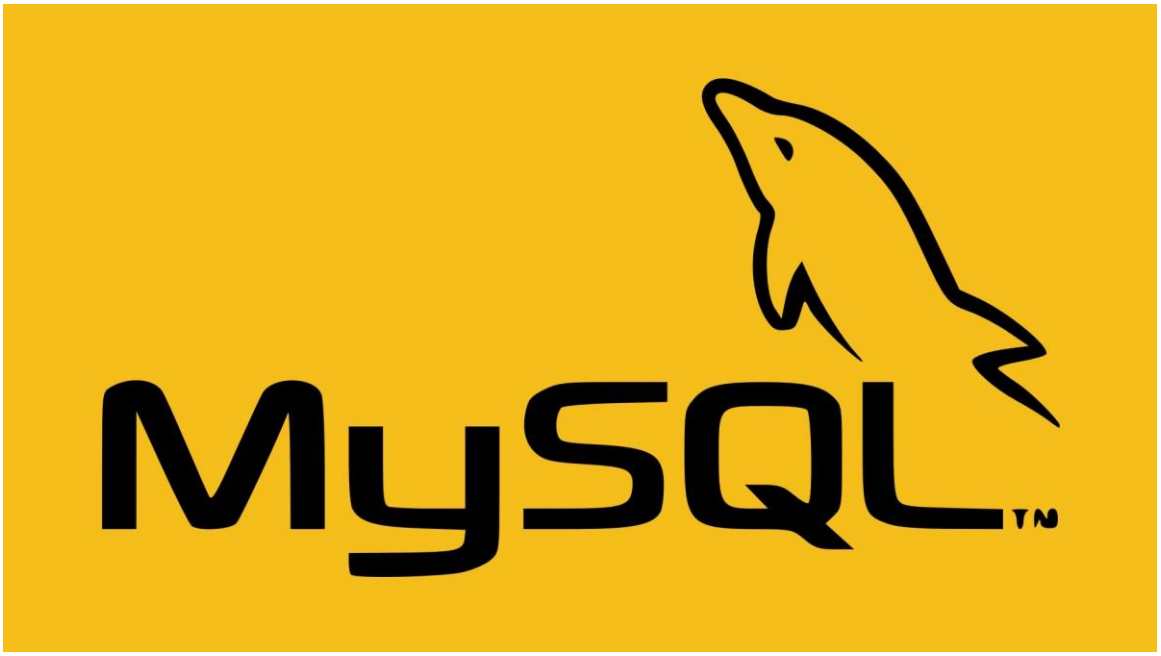


Рисунок 3.5 - MySQL

MySQL має безліч різних переваг. Наприклад, вона підтримує різні типи таблиць – як відомі MyISAM та InnoDB, так і більш екзотичні HEAP та MERGE. Крім того, кількість типів, що підтримуються, постійно зростає. MySQL виконує всі команди швидко – можливо, зараз це найшвидша СКБД із усіх існуючих. З цією системою управління базами даних може одночасно працювати необмежену кількість користувачів, а число рядків у таблицях може сягати 50 мільйонів. Так як у порівнянні з деякими іншими системами MySQL підтримує меншу кількість можливостей, то і працювати з нею

значно простіше, ніж, наприклад, з тією ж самою PostgreSQL. Тому для нашого проєкту вистачить саме цієї бази даних.

Висновки до розділу 3

В третьому розділі було розглянуто технології розробки клієнтської частини та серверної частини онлайн-системи керування проєктами. Для розробки front-end частини було обрано технології HTML, CSS, JavaScript та фреймворку Vue.js. Для реалізації back-end частини було обрано мову програмування PHP, фреймворку Laravel та інтегроване середовище розробки PhpStorm. Також було ознайомлено зі засобами реалізації інформаційного забезпечення і обрано тип СКБД реляційної бази даних MySQL на якій буде розроблятися база даних онлайн-системи керування проєктами.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН-СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ

4.1 Діаграма класів

Діаграма класів (від англ. "class diagram") призначена для представлення внутрішньої структури програми у вигляді класів та зв'язків між ними [24].

Усі сутності реального світу, з якими збирається працювати програміст, мають бути представлені об'єктами класів у програмі. При цьому у кожного класу має бути лише одне призначення та унікально осмислене ім'я, яке буде пов'язане з цією метою.

Клас - елемент діаграми, що позначає безліч об'єктів, що мають однакову внутрішню структуру, поведінку і відносини з об'єктами інших класів [24]. Зображується клас на діаграмі у вигляді прямокутника (рис. 4.1), поділеного на три секції:

– ім'я класу;
– список полів класу;
– список методів класу.

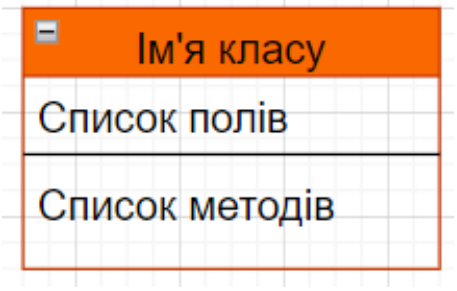


Рисунок 4.1 – Зображення класу на діаграмі

Взаємозв'язок - це особливий тип логічних відносин між сутностями, показаних на діаграмах класів та об'єктів.

Існує декілька видів взаємозв'язків між класів та об'єктів зображених на рисунку 4.2:

Class Diagram Relationship Type	Notation
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	

Рисунок 4.2 – Види взаємозв'язків

Відношення **асоціації** використовують, щоб показати, що між класами (наприклад, між двома класами) існує певний зв'язок. Зазвичай з допомогою на діаграмі класів показують, що один клас користується функціоналом іншого класу.

Відношення **залежності** використовують, щоб показати, що зміна одного класу потребує зміни іншого класу.

Відношення **успадкування** використовується, щоб показати, що один клас є батьком (базовим класом або суперкласом) для іншого класу (нащадок, похідного класу).

Відношення **агрегації** між двома класами показує, що один з них включає інший клас в якості складової частини. При цьому клас-частина може і існувати окремо від класу-цілого.

Відношення **композиції** є окремим випадком відношення агрегації. Однак у нього є одна відмінність - класи-частини, які він поєднує з класом-цілим, не можуть бути окремо.

Після ознайомлення теорії про діаграми класів приступимо до її проєктування для нашої онлайн-системи керування проєктами. Спочатку створимо та опишемо класи зі списками полів та методів цієї системи.

Опис класу User:

- name - поле, містить ім’я коистувача;
- email - поле, містить почту користувача;
- password - пароль.

Опис класу Board:

- user_id – поле містить посилання на таблицю User;
- name - поле, містить назву дошки;
- picture - поле, картинку (фон) дошки.

Опис класу List:

- board_id - поле містить посилання на таблицю Board;
- name - поле, містить назву списку.

Опис класу Card:

- list_id - поле містить посилання на таблицю List;
- name - поле, містить назву картки;
- description – поле містить опис картки.

Опис класу Comment:

- card_id - поле містить посилання на таблицю Card;
- content – поле містить вміст коментара.

Опис класу CRUD:

- create() – метод створення;
- show() – метод перегляду;
- edit() – метод редагування;
- update() - метод оновлення;
- search() - метод пошуку;
- destroy() – метод видалення.

Всі вище створені класи зі списками полів та їх методами представлені на рисунку 4.3:

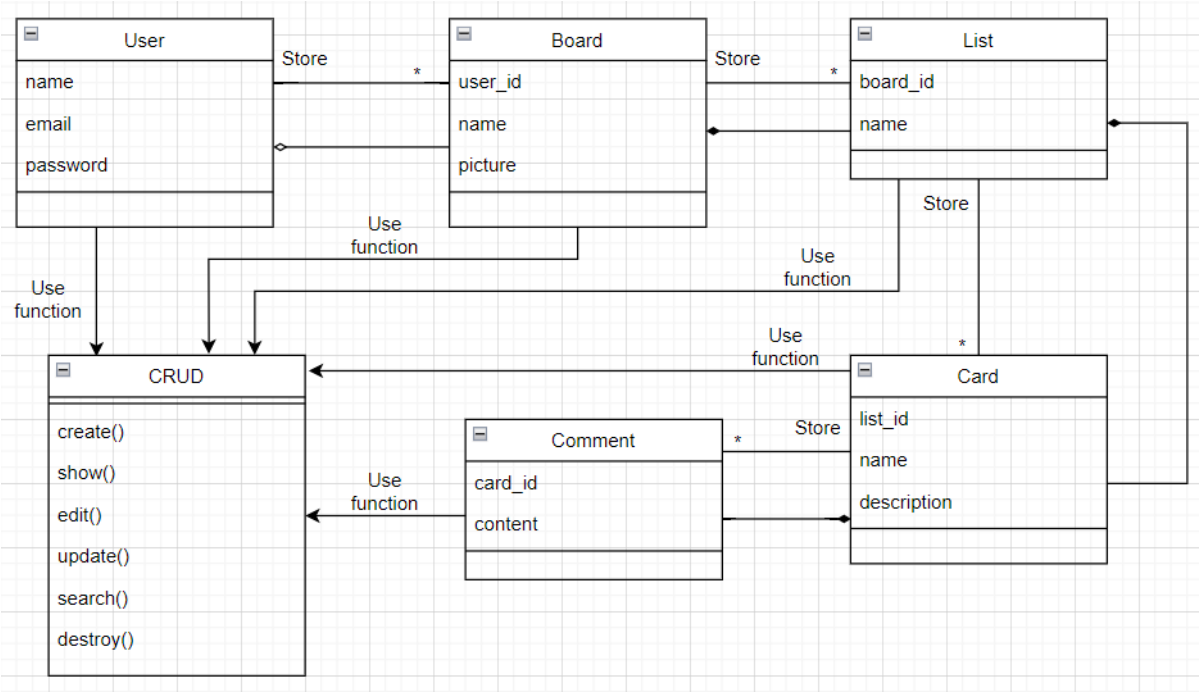


Рисунок 4.3 – Діаграма класів онлайн-системи керування проєктами

На вище зображеній на рисунку 2.5 діаграмі класів було встановлено взаємозв’язки між класами. Між класом User та Board було встановлено відношення агрегації тобто клас User може існувати й без класу Board. Всі інші класи крім CRUD мають відношення композиції тобто якщо клас контейнер видалено то й всі до нього пов’язані теж будуть видалені. Та також було встановлено відношення асоціації множинності між всіма класами крім CRUD що дало змогу зв’язати всі об’єкти класів. Клас CRUD має всі загальні методи для всіх класів тому він пов’язаний відношенням наслідування.

4.2 Діаграма компонентів

На мові уніфікованого моделювання діаграма **компонентів** показує, як компоненти з'єднуються разом для формування більших компонентів або програмних систем. Вона ілюструє архітектури компонентів програмного забезпечення та залежності між ними. Ці програмні компоненти включають

компоненти часу виконання, виконувані компоненти, а також компоненти вихідного коду. Компонентами можуть бути програмні компоненти, такі як база даних або інтерфейс користувача; або апаратні компоненти, такі як схема, мікросхема або пристрій; або бізнес-підрозділ, такий як постачальник, платіжна відомість чи доставка.

Компонентні діаграми:

– використовуються в компонентно-орієнтованих розробках для опису систем із сервіс-орієнтованою архітектурою;

– показати структуру самого коду;

– може використовуватися для фокусування на відносинах між компонентами, приховуючи деталізацію специфікації;

– допомога в інформуванні та роз'ясненні функцій створюваної системи зацікавленим сторонам.

Для організації коду, логічногого розміщення компонентів використовують діаграму компонентів. Нічого не заважає розмістити усі файли в одному чи двох пакетах або директоріях, але це негативно позначиться на якості супроводу системи та при наявності різних прав доступу до коду може порушити конфіденційність. Також, зараз робота майже усіх фреймоворків організована пакетно - модульно, і при необхідності необхідно під’єднати лише його.

На рисунку 4.5 було спроєктовано діаграму компонентів онлайн-системи керування проєктами.

В діаграмі представлено чотири компоненти:

– онлайн-система керування проєктами;

– користувацький web-інтерфейс;

– інтерфейс БД;

– сама БД.

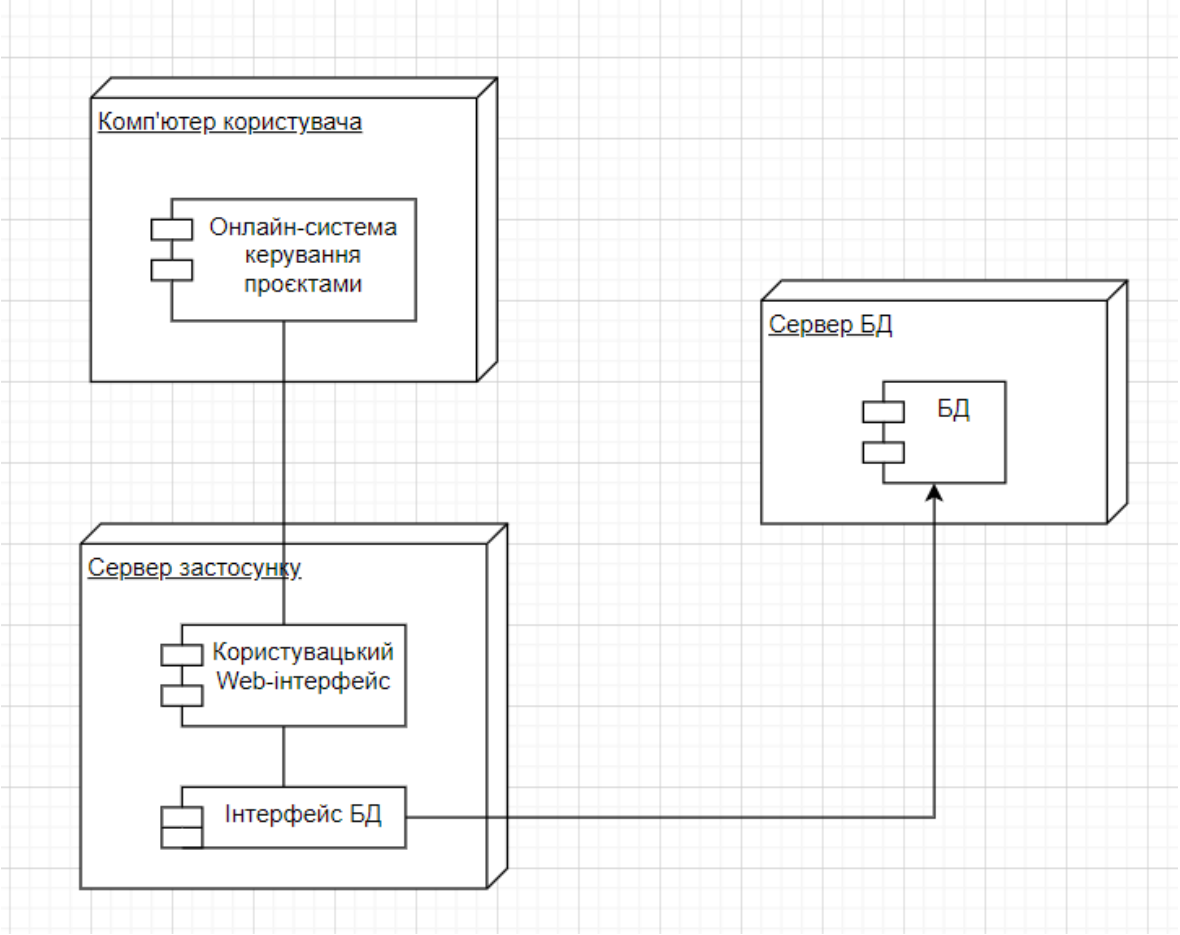


Рисунок 4.5 – Діаграма компонентів онлайн-системи керування проєктами

Вище описані компоненти находяться у вузлах:

– комп’ютер користувача;
– сервер застосунку;
– сервер БД.

4.3 Інструкція користувача онлайн-системи керування проєктами

Щоб почати користуватися вже розробленою онлайн-системою керування проєктами, як і в багатьох інших онлайн-системах, спочатку потрібно пройти реєстрацію. (рис. 4.6):

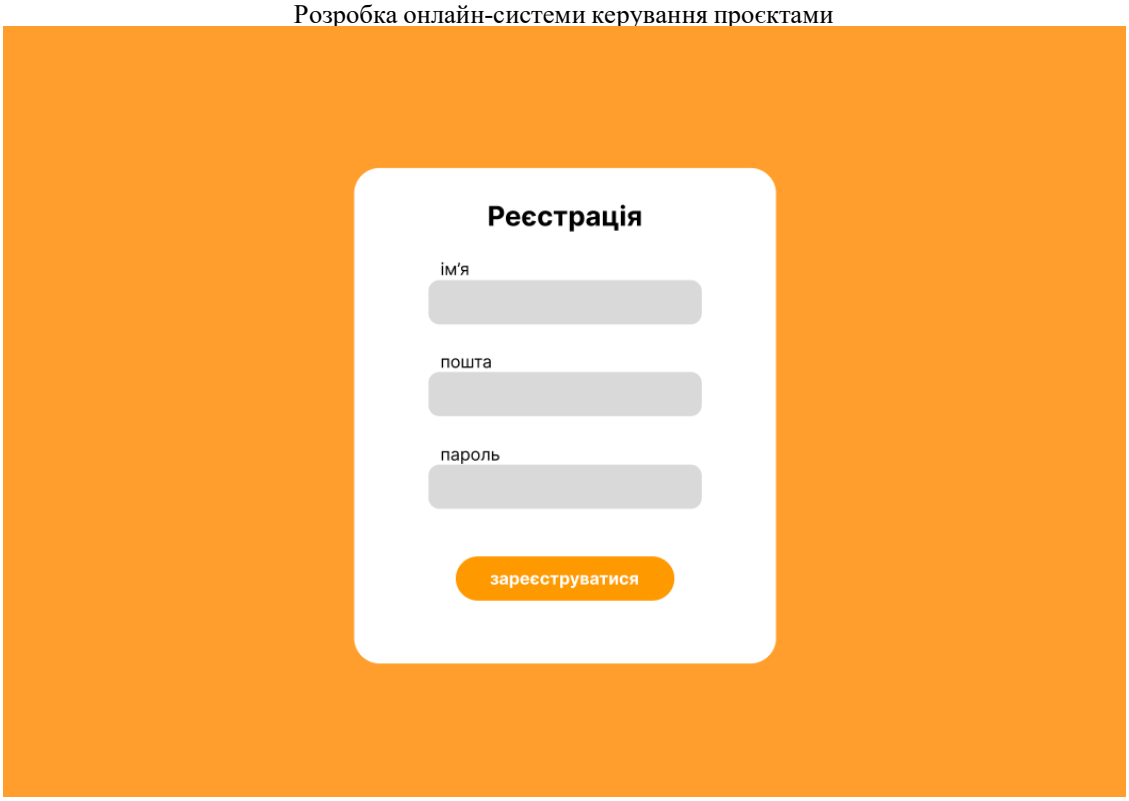


Рисунок 4.6 – Форма реєстрації

В цій формі потрібно вказати ім'я користувача, адресу електронної пошти та пароль. В разі вказування не коректних даних користувач не зможе зареєструватися. Після успішної реєстрації користувач онлайн-системи потрапляє на головну сторінку системи (рис.4.7):

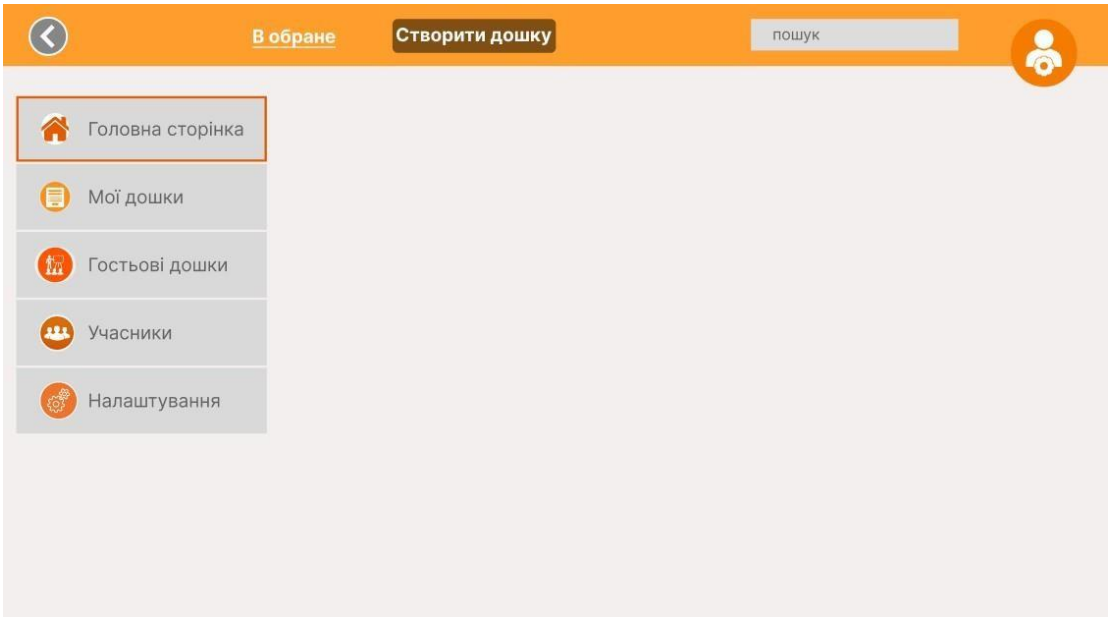


Рисунок 4.7 – Головна сторінка

На головній сторінці зображено з лівого боку головне меню в яке входить:

– мої дошки де зберігаються власні дошки користувача;
– гостьові дошки де зберігаються інші дошки;
– учасники де знаходяться учасники дошок користувача;
– налаштування.

На верхній панелі сайту містяться такі елементи:

– в обрані – це місце де зберігаються всі обрані дошки користувачем;
– кнопка створити дошку;
– поле для пошуку дошок та карток;
– кнопка що містить дані про користувача.

Далі користувач створює дошку натисканням на відповідну кнопку на верхній панелі керування сайту і в що з’явилося формі вказує дані що вимагають (рис. 4.8):

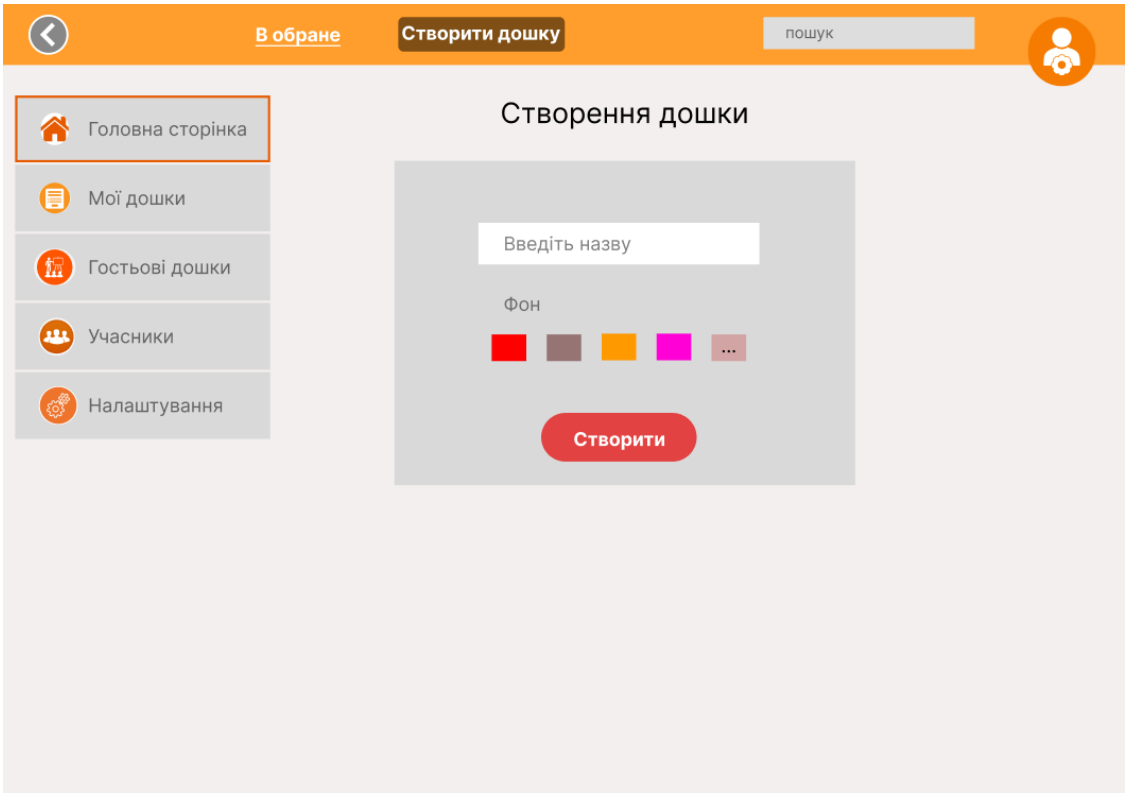


Рисунок 4.8 – Створення дошки

В формі користувач вказує назву дошки, обирає фон дошки та натискає відповідну клавішу створити.

Далі користувач потрапляє вже на сторінку з новоствореною дошкою, формою для створення списоку для карток та кнопкою запросити учасника (рис.4.9):

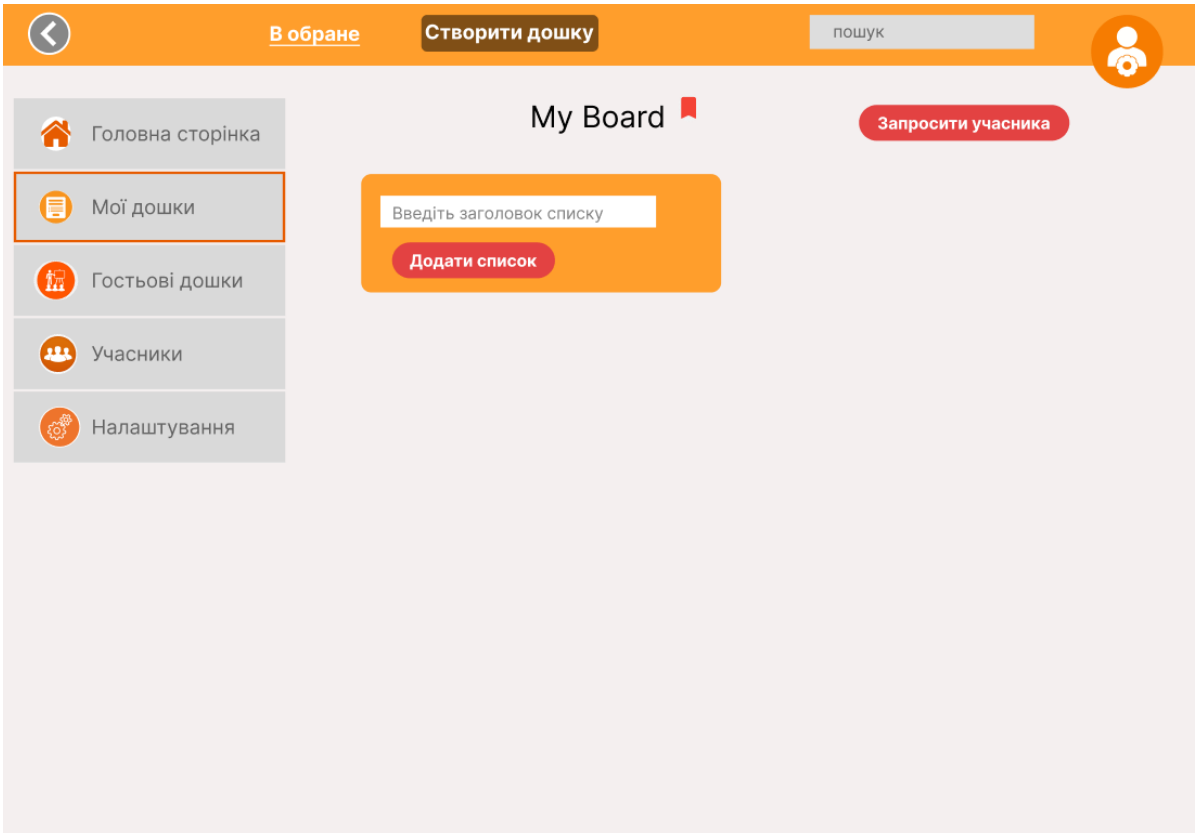


Рисунок 4.9 – Форма створення списку

В формі користувач вказує заголовок списку та натискає копку додати список.

Після вище перелічених дій користувач потрапляє на сторінку зі створеним списком та формою для додавання картки до цього списку. В формі для додавання картки користувач вказує заголовок для картки та натискає кнопку додати картку.

Форма для додавання картки наведена на рисунку 4.10.

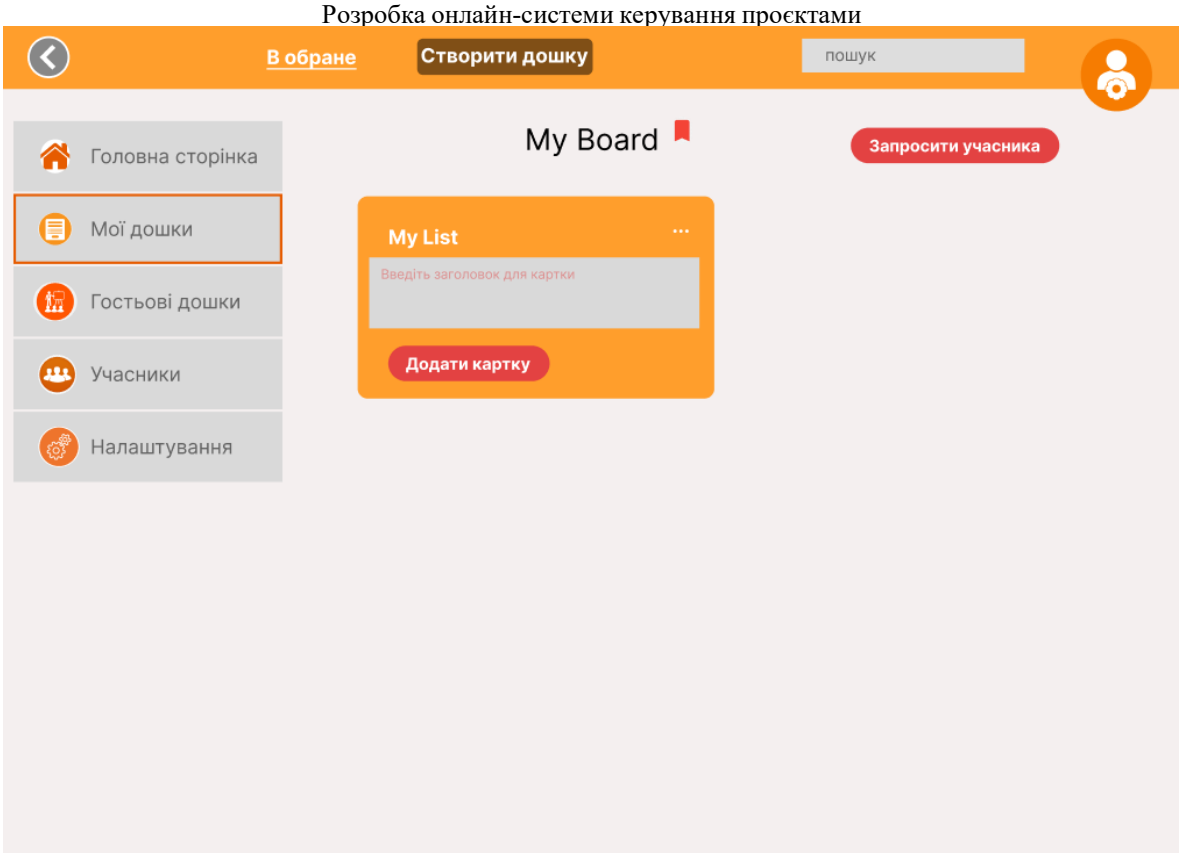


Рисунок 4.10 – Форма додавання карток

Потім користувач потрапляє на сторінку з створеною ним карткою та змогою її редагувати (рис. 4.11):

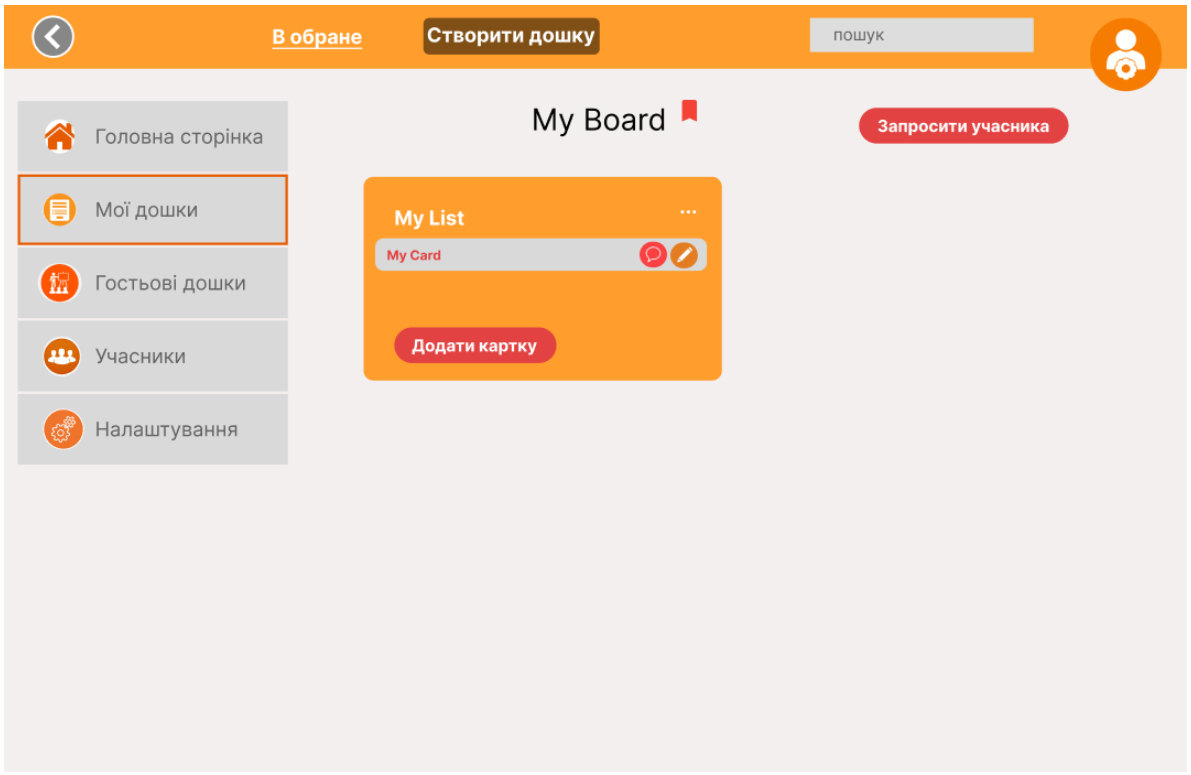


Рисунок 4.11 – Вигляд картки

Після створення картки користувач має змогу залишити коментар під карткою натиснувши кнопку коментування з правої сторони картки. Далі з’являється в нижньому правому кутку сайту не велике вікно коментування де в полі для коментування користувач має змогу написати коментар та залишити його натиснувши на кнопку зберегти (рис. 4.12):

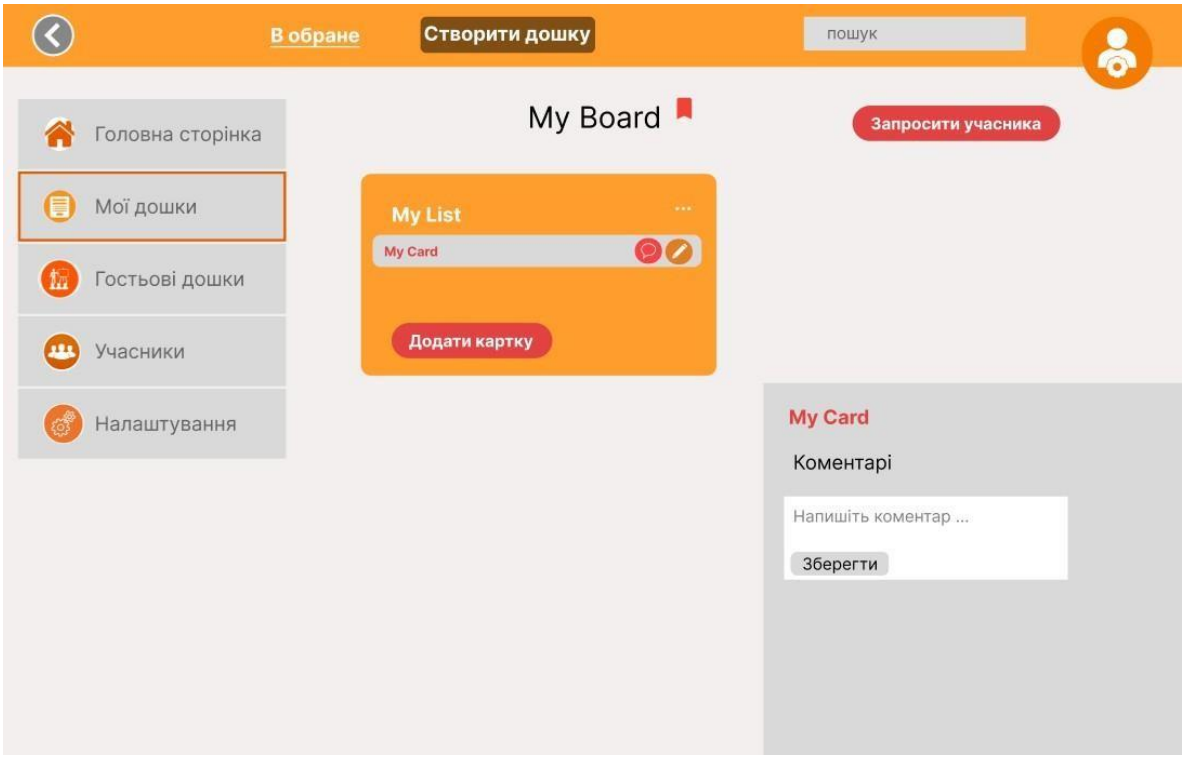


Рисунок 4.12 Вікно коментара

Після вище здійснених операцій нижче з’являється збережений коментар користувача (рис. 4.13):

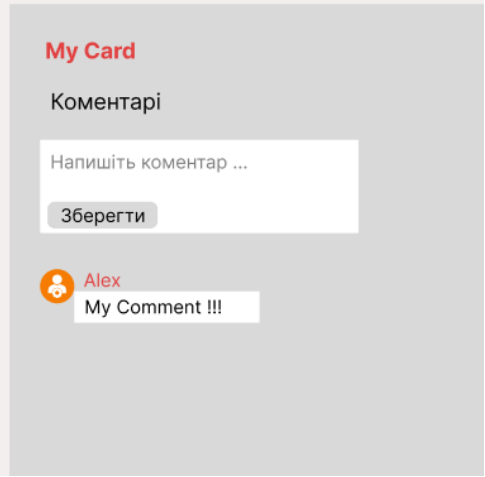


Рисунок 4.13 – Збережений коментар

Також користувач може редагувати картку завдяки кнопці з олівцем, яка розташована з правої сторони картки після кнопки коментування.

Далі користувач натискає кнопку запросити учасника та в щойно з'явилося вікні (рис. 4.14) вказує пошту потрібного йому учасника для подальшої з ним співпраці.

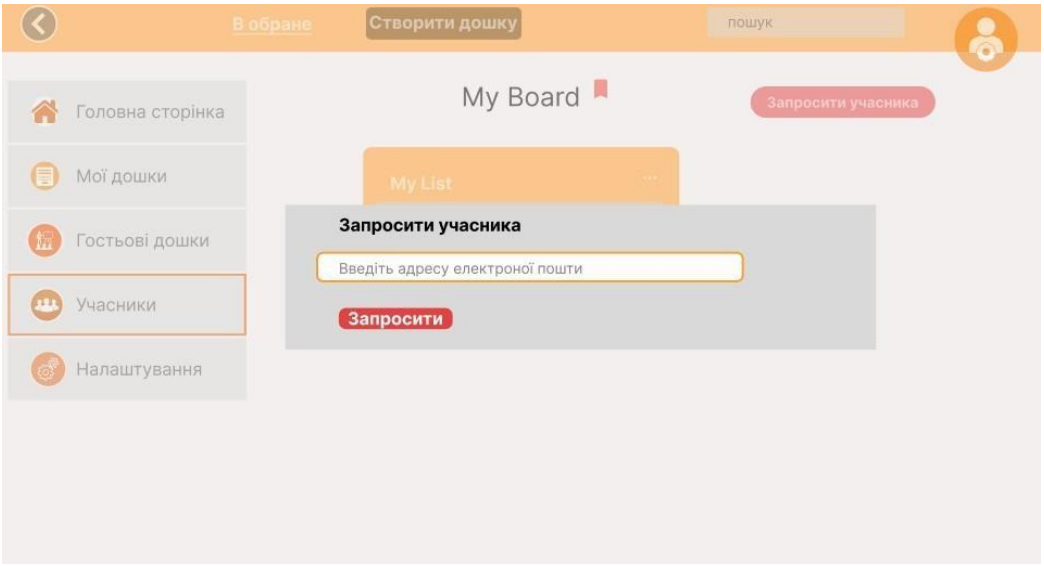


Рисунок 4.14 – Форма запиту учасника

Після запрошення учасника до дошки користувач обирає картку та натискає в картці копку з олівцем (рис. 4.15):

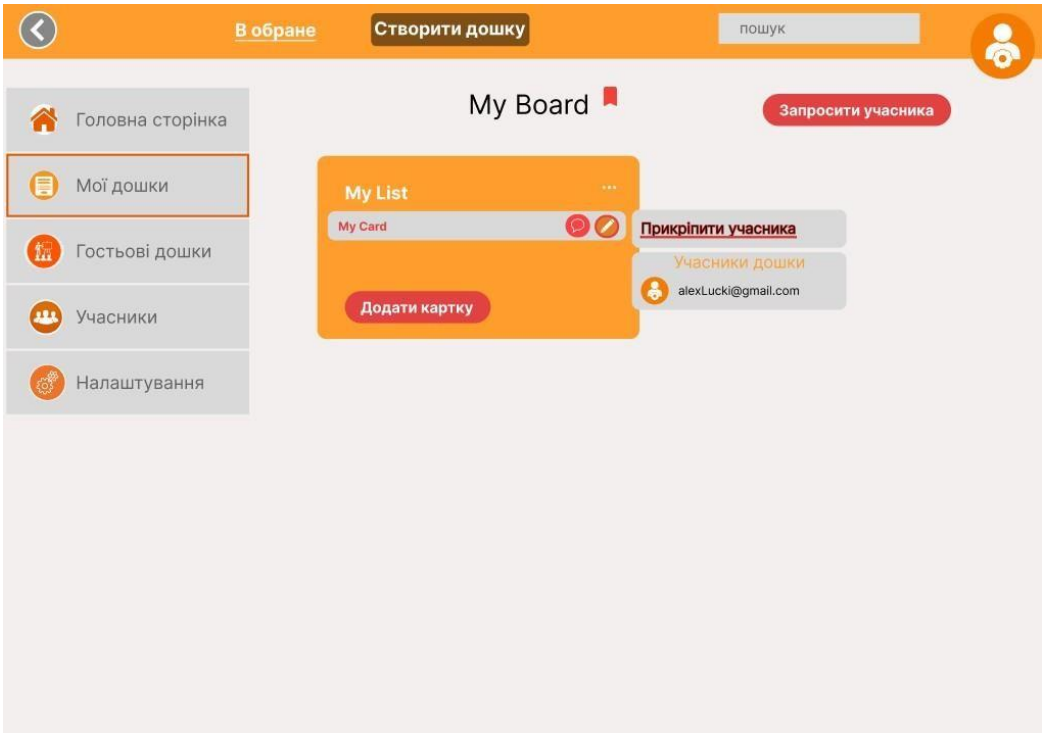


Рисунок 4.15 – Прикріплення учасника до картки

Потім натискає прикріпити учасника, обирає учасника та натискає на нього для того щоб прикріпити його до обраної картки. Вигляд картки після прикріпленого учасника зображено на рисунку 4.16.

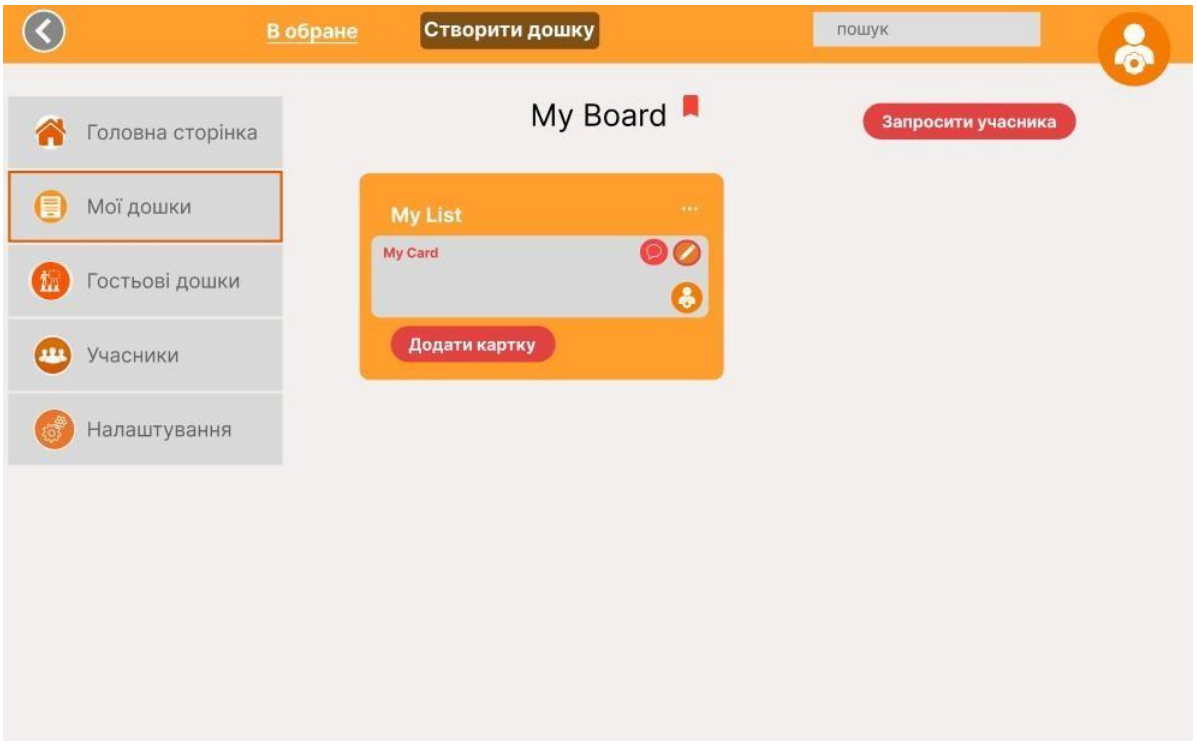


Рисунок 4.16 – Вигляд картки з учасником

Після вище пройдених операцій прикріплення учасника до картки на самій картці з правої сторони під кнопкою редагування з’являється аватар учасника.

Висновки до розділу 4

В четвертому розділі було розглянуто елементи діаграми класів. Створено діаграму класів для онлайн-системи керування проєктами. Описано їх атрибути та методи. Встановлено відношення агрегації та відношення композиції між цими класами. Побудовано діаграму компонентів для онлайн-системи керування прєктами в якій створено чотири компоненти та три вузли. Написано інструкцію для користувача онлайн-системи керування проєктами.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було проаналізовано систему, що розроблялася. Досліджено існуючі програмні рішення для користувачів, структур та сценарії роботи системи.

Розглянуто засоби апаратної та програмної реалізації. Визначено архітектуру для проєктування програмного забезпечення. Досліджено та обрано патерн MVC для розробленої онлайн-системи. Визначено моделі та контролери для онлайн-системи керування проєктами.

Змодельовано програмне забезпечення, а саме спроєктовано діаграму варіантів використання для системи керування проєктами з урахуванням усіх акторів та їх функцій. Спроєктовано діаграму класів з описаними атрибутами, методами та встановлено відношення між ними. Змодельовано діаграму компонентів в якій створено чотири компоненти та три вузли для цієї онлайн-системи.

Також було ознайомлено зі засобами реалізації інформаційного забезпечення і обрано тип СКБД реляційної бази даних MySQL на якій було розроблено базу даних онлайн-системи керування проєктами. Спроєктовано базу даних для онлайн-системи керування проєктами в програмному застосунку DBDesigner.

Було розглянуто технології розробки клієнтської частини та серверної частини онлайн-системи керування проєктами. Розроблено front-end частину онлайн-системи на базі технологій: HTML, CSS, JavaScript. Та розроблено back-end частину цієї онлайн-системи на базі технологій: Laravel, MySQL, PHP.

Написано інструкцію користувача онлайн-системи керування проєктами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мет Зандстра. Характеристики PHP 8: об’єкти, шаблони и методики програмування. Діалектика-Вільямс. 2021 р. 866 с.

2. Система керування проєктами Asana: вебсайт. URL: https://asana.com/ru (дата звернення: 15.04.2022).

3. Система керування проєктами Trello: вебсайт. URL: https://www.atlassian.com /ru /software/trello (дата звернення: 15.04.2022).

4. Система керування проєктами Jira: вебсайт. URL: https://www.atlassian. com/ software/jira (дата звернення: 15.04.2022).

5. Керування проєктами: вебсайт. URL: https://www.leadertask.ru/blog/upravlenie-proektami (дата звернення: 10.05.2022).

6. Програмне забезпечення для керування проєктами: вебсайт. URL: https://ru.wikipedia.org/wiki/Программное_обеспечение_для_управления_проектами (дата звернення: 10.05.2022).

7. Система керування проєктами: вебсайт. URL: https://www.business.ru/article/2519-sistema-upravleniya-proektami (дата звернення: 10.05.2022).

8. Використання діаграми варіантів використання: вебсайт. URL: https://habr.com/ru/post/566218/ (дата звернення: 13.05.2022).

9. Основи проєктування Use-Case діаграми: вебсайт. URL: https://github.com/kolei/PiRIS/blob/master/articles/5_1_1_10_uml_use_case.md (дата звернення: 17.05.2022).

10. Патерни в розробці: вебсайт. URL: https://otus.ru/journal/patterny-v-razrabotke-chto-eto/ (дата звернення: 18.05.2022).

11. Архітектурні патерни: вебсайт. URL: https://en.wikipedia.org/wiki/Architectural_pattern (дата звернення: 18.05.2022).

12. Загальне про MVC: вебсайт. URL: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami (дата звернення: 20.05.2022).

13. Бази даних та СКБД: вебсайт. URL: https://timeweb.com/ru/community/articles/bazy-dannyh-i-subd-1 (дата звернення: 20.05.2022).

14. Основні функції DBdesigner: вебсайт. URL: https://studbooks.net/2248041/informatika/vozmozhnosti_programmy_dbdesigner (дата звернення: 20.05.2022).

15. Первинний ключ: вебсайт. URL: https://ru.wikipedia.org/wiki/Первичный_ключ (дата звернення: 20.05.2022).

16. Frontend і Backend розробка: вебсайт. URL: https://skillbox.ru/media/code/frontend_i_backend_razrabotka/ (дата звернення: 22.05.2022).

17. Синтаксичний цукор: вебсайт. URL: https://thecode.media/sugar/ (дата звернення: 22.05.2022).

18. Основи PHP: вебсайт. URL: https://www.php.net/manual/ru/intro-whatis.php (дата звернення: 22.05.2022).

19. Фреймворк: вебсайт. URL: https://ru.wikipedia.org/wiki/Фреймворк (дата звернення: 24.05.2022).

20. Laravel: вебсайт. URL: https://blog.skillfactory.ru/glossary/laravel/ (дата звернення: 24.05.2022).

21. Інтегроване середовище розробки: вебсайт. URL: https://ru.wikipedia.org/wiki/Интегрированная_среда_разработки (дата звернення: 28.05.2022).

22. IDE для розробки: вебсайт. URL: https://www.internet-technologies.ru/articles/10-luchshih-ide.html (дата звернення: 28.05.2022).

23. Система керування базами даних: вебсайт. URL: https://ru.wikipedia.org/wiki/Система_управления_базами_данных (дата звернення: 28.05.2022).

24. Використання діаграми класів: вебсайт. URL: https://habr.com/ru/post/572234/ (дата звернення: 28.05.2022).

ДОДАТОК

Файл Board.php:

```
class Board extends Model
{
    use HasFactory;

    protected $fillable = [
        'id',
        'user_id',
        'name',
        'picture',
    ];

    public function cards()
    {
        return $this->hasMany(Card::class);
    }

    public function users()
    {
        return $this->hasMany(User::class, user_id);
    }
}
```

Файл Card.php:

```
class Card extends Model
{
    use HasFactory;

    protected $fillable =[
        'id',
        'list_id',
        'name',
        'description',
    ];

    public function list()
    {
        return $this->belongsTo(List::class, 'list_id');
    }

    public function comments()
    {
        return $this->hasMany(Comment::class, 'card_id');
    }
}
```

Файл Comment.php:

```
class Comment extends Model
{
    use HasFactory;

    protected $fillable =[
        'id',
        'card_id',
```

```
        'content',
    ];

    public function card()
    {
        return $this->belongsTo(Card::class, 'card_id');
    }
}
```

Файл User.php:

```
class User extends Authenticatable
{
    use HasFactory, Notifiable, HasRolesAndPermissions;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'id',
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];


    public function boards()
    {
        return $this->belongsToMany(Board::class);
    }

}
```

Файл List.php:

```
class List extends Model
{
    use HasFactory;
```

```
    protected $fillable = [
        'id',
        'board_id',
        'name',
        'picture',
    ];

    public function cards()
    {
        return $this->hasMany(Card::class);
    }

    public function boards()
    {
        return $this->hasMany(User::class, board_id);
    }
}
```

Файл BoardsController.php:

```
class BoardsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {

        $boards = Board::query()->where('user_id','=',Auth::user()->id)-
>get();

        return view('boards.index', compact('boards'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {

        return view('boards.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(BoardRequest $request)
    {


//        dd(Auth::user()->id);
```

```
        Board::create([
            'user_id' => Auth::user()->id,
            'name' => $request->input('name'),
            'picture' => $request->input('picture'),
        ]);

        return redirect()->route('boards.index')->with('success','Board
created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show(Board $board)
    {
        return view('boards.show',compact('board'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit(Board $board)
    {
        return view('boards.edit', compact('board'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(BoardRequest $request, Board $board)
    {
        $board->update($request->all());

        return redirect()->route('boards.index')->with('success','Board
updated successfully');
    }

    public function search(Request $request, Board $boards)
    {

        $name = $request->input('name');

        $boards = Board::query()
            ->where('boards.name','LIKE', '%'.$name.'%')
            ->orwhereHas('cards',function (Builder $query) use ($name){
                $query->where('cards.name', 'LIKE', '%'.$name.'%');
            })
            ->orWhereHas('cards.comments',function (Builder $query) use
($name){
                $query->where('comments.content', 'LIKE','%'.$name.'%');
            })
            ->get();
```

```
        return view('boards.index',compact('boards'));

    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Board $board)
    {
        $board->delete();

        return redirect()->route('boards.index',$board->user_id)
            ->with('success','board deleted successfully');
    }
}
```

Файл CardsController.php:

```
class CardsController extends Controller
{
    public function index($card_id)
    {
        $cards = Card::query()->where('board_id',$card_id)->latest()-
>simplePaginate(15);

        return view('cards.index',compact('cards'));
    }


    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create($board_id)
    {
        return view('cards.create', ['board_id' => $board_id]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(CardRequest $request)
    {
        Card::create([
            'board_id' => $request->input('board_id'),
            'name'=>$request->input('name'),
            'description'=>$request->input('description'),
        ]);

        return redirect()->route('cards.index',['board_id'=>$request-
>input('board_id')])->with('success','Card created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function show(Card $card)
    {
        return view('cards.show',compact('card'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit(Card $card)
    {
        return view('cards.edit', compact('card'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Card $card)
    {
        $request->validate([
            'name' => 'required',
            'description' => 'required',
        ]);

        $card->update($request->all());

        return redirect()->route('cards.index')->with('success','Card updated
successfully');
    }

    public function search(Request $request, Card $cards)
    {
        $name = $request->input('name');
        $cards = Card::query()->where('name', 'LIKE', '%'.$name.'%')-
>paginate();

        return view('cards.index', compact('cards'));
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Card $card)
    {
        $card->delete();

        return redirect()->route('cards.index',$card->board_id)
            ->with('success','card deleted successfully');
    }

}
```

Файл CommentsController.php:

```
class CommentsController extends Controller
{
    public function index($card_id)
    {
        $comments = Comment::where('card_id',$card_id)->get();

        return view('comments.index', compact('comments'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create($card_id)
    {
        return view('comments.create', ['card_id' => $card_id]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(ComentRequest $request)
    {


//        dd($request->input('card_id'));

         Comment::query()->create([
            'card_id' => $request->input('card_id'),
            'content'=>$request->input('content'),
        ]);


        return response()->json();

        return redirect()->route('comments.index',['card_id'=>$request-
>input('card_id')])->with('success','Comment created successfully.');

    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show(Comment $comment)
    {
        return view('comments.show',compact('comment'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
```

```
    public function edit(Comment $comment)
    {
        return view('comments.edit', compact('comment'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Comment $comment)
    {
        $request->validate([
            'content' => 'required',
        ]);

        $comment->update($request->all());

        return redirect()->route('comments.index',$comment->card_id)-
>with('success','Card updated successfully');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Comment $comment)
    {
        $comment->delete();

        return redirect()->route('comments.index',$comment->card_id)
            ->with('success','card deleted successfully');
    }
}
```

Файл UsersController.php:

```
final class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //$user = User::find(1);
        //dd($user->hasRole('developer')); //вернёт true
        //dd($user->hasRole('project-manager')); //вернёт false
        //dd($user->givePermissionsTo('manage-user')); //выдаём разрешение
        //dd($user->hasPermission('manage-user')); //вернёт true

        //Gate::allows('manage-user');

        $users = User::all();

        return view('users.index', compact('users'));
    }

    /**
```

```
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {

        return view('users.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(UserRequest $request)
    {


//        dd(Auth::user()->id);


        User::create([
            'name' => $request->input('name'),
            'email' => $request->input('email'),
            'password' => $request->input('password'),
        ]);

        return redirect()->route('users.index')->with('success','User created
successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show(User $user)
    {
        //return view('boards.show',compact('board'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit(User $user, Role $roles)
    {
        $roles = Role::all();

        return view('users.edit', compact('user','roles'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

```
     */
    public function update(UserRequest $request, User $user)
    {
        $user->update([
            'name' => $request->input('name')
        ]);
        foreach ($user->roles as $role){
            $user->roles()->detach($role);
        }

        $roles = $request->input('roles');
        foreach ($roles as $roleId) {
            $role = Role::query()->where('id', '=', $roleId)->first();
            if (!$user->hasRole($role->slug)){
                $user->roles()->attach($role);
            }
        }

        return redirect()->route('users.index')->with('success','User updated
successfully');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(User $user)
    {
        $user->delete();

        return redirect()->route('users.index',$user->id)
            ->with('success','user deleted successfully');
    }
}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка онлайн-системи керування проєктами

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ ОХОРОНА ПРАЦІ ПРИ КОРИСТУВАННІ ЕКРАННИМИ ПРИСТРОЯМИ

Спеціальність «Інженерія програмного забезпечення»
121 – КРБ.1 – 408.21810911

Студент  **О. С. Дурнєв**
підпис
«__»_____2022 р.

Консультант к.т.н., доцент _____**А. О. Алексєєва**
підпис
«__»_____2022 р.

Миколаїв – 2022

ЗМІСТ

ВСТУП 4
1 ОХОРОНА ПРАЦІ 5
1.1 Вимоги безпеки під час безпосередньої роботи за ЕОМ 5
1.2 Опис приміщення 6
1.3 Ергономіка робочого місця 7
1.4 Освітленість 11
1.5 Мікроклімат 15
1.6 Випромінювання монітору 17
1.7 Виробничий шум та вібрації 17
Висновки до спеціальної частини 19
ВИКОРИСТАНА ЛІТЕРАТУРА 20

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ЕОМ	–	Електронна обчислювальна машина
дБа	–	Одна десята Бела

ВСТУП

Впровадження обчислювальної техніки на виробництво дає позитивний соціально-економічний ефект, який призводить до зростання продуктивності, зниження частки рутинної, монотонної праці, підвищення швидкості розрахунків швидкості, обміну інформацією.

Охорона праці - це система правових, соціально-економічних, організаційно-технічних заходів, а так само санітарно-гігієнічних і лікувально-профілактичних засобів, направлених на збереження здоров'я і працездатності людини в процесі праці. (згідно закону України "про охорону праці" ст.1)

Метою охорони праці є виключення впливу на людину небезпечних та шкідливих виробничих факторів, тобто забезпечити безпеку виробничого процесу та виробничого обладнання, оптимізувати трудові процеси та виробничу обстановку. На основі такого впливу визначаються небезпечні ділянки виробництва, можливі аварійні ситуації і розробляються заходи щодо їх усунення.

Закон України «Про охорону праці» визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

Ці нормативні документи зобов'язують до створення на робочому місці умов праці, при яких вплив небезпечних і шкідливих чинників на працюючих або усунуто зовсім, або знаходиться в допустимих межах.

На робочому місці повинні бути передбачені заходи захисту від можливого впливу небезпечних і шкідливих факторів виробництва. Рівні цих факторів не повинні перевищувати граничних значень, обумовлених правовими, технічними і санітарно-технічними нормами.

1 ОХОРОНА ПРАЦІ

1.1 Вимоги безпеки під час безпосередньої роботи за ЕОМ

Користувачі ЕОМ повинні слідкувати за тим, щоб відеотермінали, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ були справними і випробуваними відповідно до чинних нормативних документів.

Щоденно перед початком роботи необхідно проводити очищення екрану відеотерміналу від пилу та інших забруднень.

Після закінчення роботи відеотермінал та персональна ЕОМ повинні бути відключені від електричної мережі.

У разі виникнення аварійної ситуації необхідно негайно відключити відеотермінал та ЕОМ від електричної мережі.

При потребі, для захисту від електромагнітних, електростатичних та інших полів можуть застосовуватися спеціальні технічні засоби, що мають відповідний сертифікат або санітарно-гігієнічний висновок акредитованих органів щодо їх захисних властивостей.

Є неприпустимими такі дії:

- виконання обслуговування, ремонту та налагодження ЕОМ безпосередньо на робочому місці користувача ЕОМ;
- зберігання біля відеотерміналу та ЕОМ паперу, дискет, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;
- відключення захисних пристроїв, самочинне проведення змін у конструкції та складі ЕОМ, устаткування або їх технічне налагодження;
- робота з відеотерміналами, в яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;
- праця на матричному принтері зі знятою (трохи піднятою) верхньою кришкою.

1.2 Опис приміщення

Під час роботи над створенням дипломного проєкту інженер працюватиме в приміщенні відповідної установи. В даній роботі було надано робоче місце зі стаціонарним комп’ютером для однієї людини, що працювала над проєктом. Схема приміщення зображена на рисунку 1.

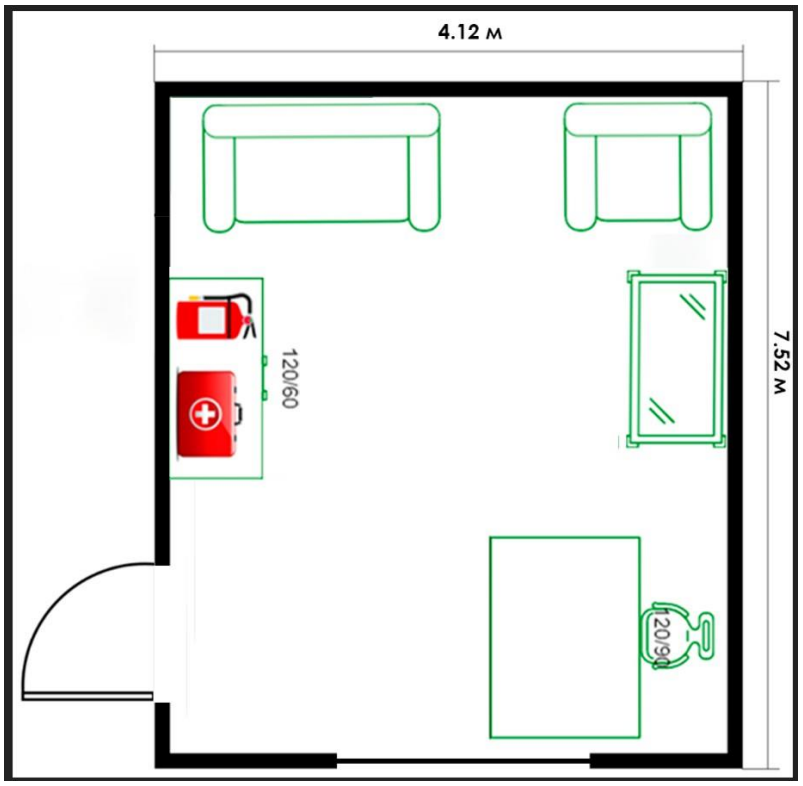


Рисунок 1 – План комп’ютерного приміщення

Виробниче приміщення споруджене згідно з вимогами [1]. Приміщення обладнане одним робочим місцем для одного інженера. Геометричні розміри якого зазначені в таблиці 1.

Таблиця 1 – Розміри приміщення

Назва	Значення
Довжина, м	7,52
Ширина, м	4,12
Висота, м	2,95
Площа, $м^2$	30,98
Об’єм, $м^3$	91,39

Виробничий процес інженера полягає в розробці алгоритмів, технічної документації та написанні програмного забезпечення, що потребує використання ЕОМ. Згідно з [2] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

В комп'ютерної лабораторії розташоване одно робоче місце обладнане ПК з рідкокристалічним дисплеєм і приєднане до локальної мережі.

Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціювання.

Для забезпечення потрібного рівного освітленості кімната має одне вікно та систему загального рівномірного освітлення, що встановлена на стелі. В приміщенні відсутня спеціальна вентиляція і звукоізоляція.

1.3 Ергономіка робочого місця

Проєктування робочих місць відноситься до числа найважливіших проблем ергономічного проєктування в області обчислювальної техніки.

Ергономіка - це наука про зручність, про організацію робочого простору для комфортної та ефективної праці працівника, виходячи з фізичних та психологічних особливостей людського організму. Фактично це зведення основних правил для забезпечення безпеки праці та комфорту в умовах сучасного офісу [12].

Ергономічними аспектами проєктування робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до розташування документів на робочому місці (наявність і розміри підставки для документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу, можливість регулювання елементів робочого місця [1].

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця

програміста повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що входить до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення.

Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і сталість розміщення предметів, засобів праці і документації. Те, що потрібно для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле – простір робочого місця, в якому можуть здійснюватися рухові дії людини.

Максимальна зона досяжності рук – це частина моторного поля робочого місця, обмеженого дугами, що описуються максимально витягнутими руками при русі їх в плечовому суглобі.

Оптимальна зона – частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем.

Розглянемо оптимальне розміщення предметів праці і документації в зонах досяжності рук:

а) дисплей розміщується в зоні а (у центрі);

б) клавіатура – у зоні г;

в) системний блок розміщується в зоні б (ліворуч);

г) література і документація, необхідна при роботі – в зоні в (ліворуч).

Для комфортної роботи стіл повинен задовольняти наступним умовам [1]:

– висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;

– нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, ні змушений підбирати ноги;

– поверхня стола повинна мати властивості, що виключають появу відблисків в полі зору програміста;

– конструкція столу повинна передбачати наявність висувних ящиків.

– висота робочої поверхні рекомендується в межах 680-760мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути біля 650мм.

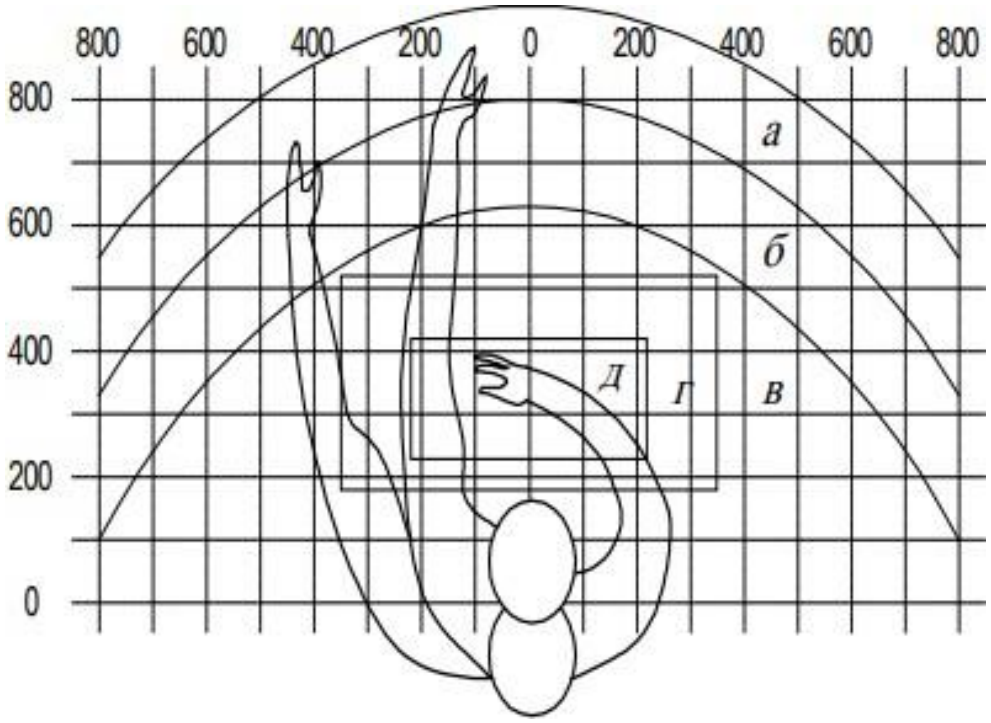


Рисунок 2 – Зони досяжності рук у горизонтальній площині: а – зона максимальної досяжності; б – зона досяжності пальців при витягнутої руці; в – зона легкої досяжності долоні; р – оптимальний простір для грубої ручної роботи; д – оптимальний простір для тонкої ручної роботи.

Велике значення надається характеристикам робочого крісла. Так, рекомендована висота сидіння над рівнем підлоги знаходиться в межах 420 – 550мм. Поверхня сидіння м'яка, передній край закруглений, а кут нахилу спинки – регульований.

Положення екрану визначається:

– відстанню зчитування (0,6 ... 0,7 м);

– кутом зчитування, напрямком погляду на 20 ° нижче горизонталі до центру екрану, причому екран перпендикулярний цьому напряму.

Повинна також передбачатися можливість регулювання екрану:

– по висоті +3 см;

– по нахилу від -10 ° до + 20 ° щодо вертикалі;

– в лівому і правому напрямках.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози користувача відеотермінала наступні:

– голова не повинна бути нахилена більш ніж на 20 °;

– плечі повинні бути розслаблені;

– лікті – під кутом 80 ° ... 100 °;

– передпліччя і кисті рук – в горизонтальному положенні.

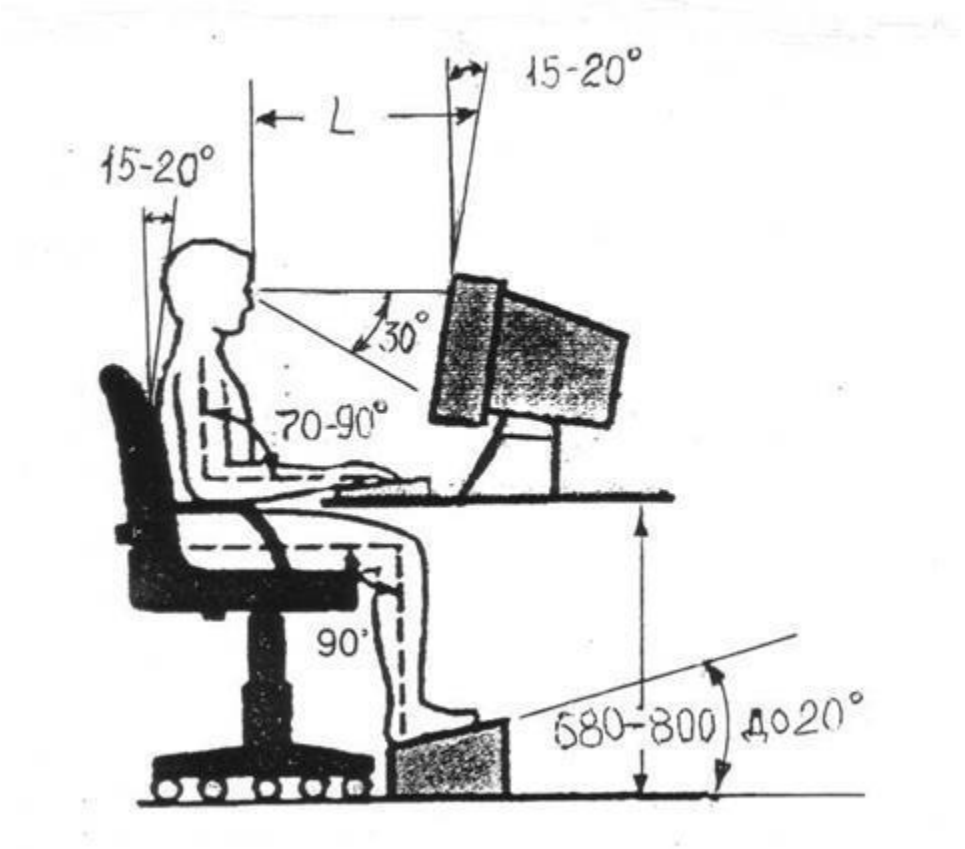


Рисунок 3 – Зони досяжності рук у горизонтальній площині.

Причина неправильної пози користувачів обумовлена наступними факторами: немає гарної підставки для документів, клавіатура знаходиться дуже

високо, а документи – низько, нікуди покласти руки і кисті, недостатній простір для ніг.

З метою подолання зазначених недоліків даються загальні рекомендації: краще пересувна клавіатура; повинні бути передбачені спеціальні пристосування для регулювання висоти столу, клавіатури і екрану, а також підставка для рук [1].

Під час користування комп'ютером медики радять встановлювати монітор на відстані 50–60 см від очей. Фахівці також вважають, що верхня частина дисплея повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться вниз. За рахунок цього площа огляду значно збільшується, викликаючи обезводнення очей. До того ж якщо екран встановлений високо, а очі широко відкриті, порушується функція моргання. Це означає, що очі не закриваються повністю, не омиваються слізною рідиною, не отримують достатнього зволоження, що приводить до їх швидкої стомлюваності.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці.

1.4 Освітленість

Недостатність освітлення призводить до напруги зору, послаблює увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає осліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому настільки важливий правильний розрахунок освітленості.

Існує три види освітлення – природне, штучне і поєднане (природне і штучне разом) [4].

Природне освітлення – освітлення приміщень денним світлом, що потрапляє через світлові прорізи в зовнішніх огороджувальних конструкціях приміщень. Природне освітлення характеризується тим, що змінюється в широких межах залежно від часу дня, пори року, характеру області і ряду інших чинників.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішаним освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, у свою чергу, може бути загальним або комбінованим. Загальне – освітлення, при якому світильники розміщуються у верхній зоні приміщення рівномірно або стосовно до розташування обладнання. Комбіноване – освітлення, при якому до загального додається місцеве освітлення.

Згідно СНиП II-4-79 в приміщень обчислювальних центрів необхідно застосувати систему комбінованого освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкта розрізнення 0,3 ... 0,5 мм) величина коефіцієнта природного освітлення (КПО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкта розрізнення 0,5 ... 1,0 мм) КЕО повинен бути не нижче 1,0%. Як джерела штучного освітлення звичайно використовуються світильники, які повинні розташовуватися рівномірно над робочими поверхнями [5].

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300лк, а комбінована – 750лк; аналогічні вимоги при виконанні робіт середньої точності – 200 і 300лк відповідно.

Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Іншими словами, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими, тому що яскраве світло в районі периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності.

Забарвлення приміщень і меблів повинна сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою.

Джерела світла, такі як світильники і вікна, які дають віддзеркалення від поверхні екрану, значно погіршують точність знаків і тягнуть за собою перешкоди фізіологічного характеру, які можуть виразитися в значній напрузі, особливо при тривалій роботі. Відображення, включаючи відображення від вторинних джерел світла, повинне бути зведено до мінімуму. Для захисту від надмірної яскравості вікон можуть бути застосовані штори і екрани [5].

Залежно від орієнтації вікон рекомендується наступне забарвлення стін і підлоги:

- вікна орієнтовані на південь: – стіни зеленувато-блакитного або світло-блакитного кольору; підлога – зелений;
- вікна орієнтовані на північ: – стіни світло-оранжевого або оранжево-жовтого кольору; підлога – червонувато-оранжевий;
- вікна орієнтовані на схід: – стіни жовто-зеленого кольору;
- підлога зелена або червонувато-оранжевий;
- вікна орієнтовані на захід: – стіни жовто-зеленого або голубувато-зеленого кольору; підлога зелена або червонувато-оранжевий.

У приміщеннях, де знаходиться комп'ютер, необхідно забезпечити наступні величини коефіцієнта віддзеркалення: для стелі: 60 ... 70%, для стін: 40 ... 50%, для підлоги: близько 30%. Для інших поверхонь і робочих меблів: 30 ... 40%.

В даному приміщенні передбачається використання комбінованого природного і штучного освітлення. У світильниках загального освітлення використовуються світлодіодні лампи. Розрахунок освітленості робочого місця

зводиться до вибору системи освітлення, визначенню необхідного числа світильників, їхнього типу і розміщення. Виходячи з цього, розрахуємо параметри штучного освітлення [6].

Відповідно до вибраного розрядом зорових робіт допустиме значення освітленості робочої поверхні приймається Е = 300 лк. Для розрахунку освітлення КЛ скористаємося методом світлового потоку [7].

Розрахунок освітлення проводиться для кімнати площею 30,98 м^{2,} ширина якої 4,12 м, довжина – 7,52 м, висота – 2,95 м.

Для визначення кількості світильників визначимо світловий потік, що падає на поверхню по формулі 1

$$F = \frac{E \cdot k \cdot S \cdot Z}{\eta}, \quad (1)$$

де F – розраховується світловий потік, Лм;

Е – нормована мінімальна освітленість, Е = 300 Лм;

k – коефіцієнт запасу, враховує зменшення світлового потоку лампи в результаті забруднення світильників у процесі експлуатації К = 1,5;

S – площа освітлюваного приміщення S = 30,98 м^2;

Z – відношення середньої освітленості до мінімальної Z = 1,1;

η – коефіцієнт використання світового потоку від світильника.

Для знаходження значення η необхідно обчислити індекс приміщення за формулою 2

$$I = \frac{S}{h \cdot (A+B)}, \quad (2)$$

де S – площа приміщення, S = 30,98 м$^{2;}$

h – розрахункова висота підвісу, h = 2.9 м;

A – ширина приміщення, А = 4,12 м;

B – довжина приміщення, В = 7,52 м.

Підставивши отримані значення у формулу 3 підрахуємо індекс для заданого приміщення.

$$I = \frac{30,98}{2,9 \cdot (4,12+7,52)} = 0,917. \quad (3)$$

При індексі приміщення I, знаходимо η = 0,34. Підставивши всі значення у формулу для отримуємо світловий потік F.

$$F = \frac{300 \cdot 1,5 \cdot 13,3 \cdot 1,1}{0,34} = 45\ 103. \quad (4)$$

У світильнику робочого приміщення використовується світлодіодна лампа Vestum T160 E27-60W-6500K світловий потік якої 6000 Лм. Розрахуємо необхідну кількість ламп за формулою 1.9.

$$N = \frac{F}{F_л}, \quad (5)$$

де N – обумовлене число ламп;

F – світловий потік, F = 19 363 Лм;

$F_л$ – світловий потік лампи, $F_л$ = 6000 Лм.

В ході обчислювань було встановлено, що для приміщення необхідно однин світильник, в якому міститься 3 лампи.

1.5 Мікроклімат

Параметри мікроклімату можуть змінюватися в широких межах, в той час як необхідною умовою життєдіяльності людини є підтримка постійності температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в

приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату.

У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. Табл. 2) [3].

Таблиця 2 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22-24	40-60	0,1
Тепла	легка-1 а	23-25	40-60	0,1

Обсяг приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути менше 19,5м3 / людина з урахуванням максимального числа одночасно працюючих в зміну.

Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в табл. 3.

Таблиця 3 – Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери [6]

Характеристика приміщення	Об'ємна витрата подається в приміщення свіжого повітря, м ^{3} / на одну людину в годину
Об'єм до 20м ^{3}на особу	Не менше 30
20 ... 40м ^{3} на особу	Не менше 20
Більш 40м ^{3} на особу	Природна вентиляція

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби,

чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціонування повітря, опалювальна система).

1.6 Випромінювання монітору

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера представлені в таблиці 4

Таблиця 4 – Допустимі значення параметрів неіонізуючих електромагнітних випромінювань

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відеомонітору	10 В/м
Напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відеомонітору	0,3 А/м
Напруженість електростатичного поля не повинна перевищувати: для дорослих користувачів для дітей дошкільних установ і що навчаються середніх спеціальних і вищих навчальних закладів	20кВ/м 15кВ/м

Нормованим параметром невикористаного рентгенівського випромінювання виступає потужність експозиційної дози. На відстані 5 см від поверхні екрану монітору її рівень не повинен перевищувати 100 мкР/год. Максимальний рівень рентгенівського випромінювання на робочому місці оператора комп'ютера зазвичай не перевищує 20 мкР/год.

1.7 Виробничий шум та вібрації

Шум як несприятливий чинник виробничого середовища наявний на більшості промислових підприємств, на транспорті, у сільському господарстві. Джерелами шуму можуть бути системи вентиляції та кондиціонування повітря, аерогазодинамічні установки, двигуни, верстати, молоти, дробарки

тощо. Інтенсивний виробничий шум може стати причиною таких професійних захворювань, як туговухість або глухота. Крім того, у працівників, які щодня перебувають під його впливом: знижується продуктивність праці; ослаблюється увага та уповільнюється реакція, спостерігається запаморочення, дратівливість, знижується працездатність, гострота зору; зростає кров'яний тиск, змінюється ритм дихання та серцевої діяльності, порушується працездатність клітин кори головного мозку тощо. Звичним для людини є шумовий фон з рівнем звукового тиску в частотах 15-35 децибел (дБ). При збільшенні рівня звукового тиску до 40-70 дБ спостерігається деяке зниження продуктивності праці та погіршення самопочуття (голосна музика, шум технічного устаткування та інше). Рівень звукового тиску в межах 75-120 дБ спричиняє враження органів слуху і серцево-судинної системи. Постійний шум з рівнем звукового тиску понад 120 дБ може призвести до акустичної травми (значне зниження слуху). Те, як проявляються патологічні зміни в організмі, які спричинив шум, залежить від його параметрів (інтенсивність і частотний склад), стажу роботи, тривалості дії протягом робочого дня, індивідуальної чутливості організму, поєднання з іншими професійними чинниками.

У табл. 5 вказані граничні рівні звуку залежно від категорії тяжкості і напруженості праці, що є безпечними відносно збереження здоров'я і працездатності.

Таблиця 5 – Граничні рівні звуку, дБ, на робочих місцях [8]

Категорія напруженості праці	Категорія важкості праці			
	I. Легка	II. Середня	III. Важка	IV. Дуже важка
I. Мало напружений	80	80	75	75
II. Помірно напружений	70	70	65	65
III. Напружений	60	60	–	–
IV. Дуже напружений	50	50	–	–

Рівень шуму на робочому місці інженерів і операторів відеоматеріалів не повинен перевищувати 50дБА, а в залах обробки інформації на обчислювальних машинах – 65дБА. Для зниження рівня шуму стіни і стеля приміщень, де встановлені комп'ютери, можуть бути облицьовані звукопоглинальними матеріалами.

Джерелом вібрації може бути пневматичне, механічне, а також електричне обладнання та інструменти обертального або ударного типу. Розрізняють локальну вібрацію, яка діє на окремі частини тіла людини, і загальну вібрацію, яка передається на організм (тіло) через сидіння або ноги працівника тощо. Загальна вібрація, своєю чергою, поділяється на транспортну (виникає під час руху транспортних засобів) і технологічну, яка існує при роботі на стаціонарному обладнанні або ж передається відповідним чином на робочі місця, які не мають власних джерел вібрації. Локальна вібрація, зокрема, спостерігається при виконанні робіт пневматичними відбійними молотками, використанні технологічного обладнання з вібраційними властивостями (ковальсько-пресове устаткування) тощо. Рівень вібрації в приміщеннях обчислювальних центрів може бути понижений шляхом встановлення устаткування на спеціальні віброізолятори.

Висновки до спеціальної частини

У даному розділі було описано вимоги до робочого місця для інженера програмного забезпечення. Було розглянуто заходи які потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам та було комфортним і безпечним для робітника. Приведені вимоги безпеки під час безпосередньої роботи за ОЕМ та організації робочого місця. Створені умови повинні забезпечувати комфортну роботу. Описано допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп’ютера. На підставі розглянутої літератури по даній темі, були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Апостолюк С. О., Джигирей В. С., Соколовський І. А. Безпека праці: ергономічні та естетичні основи : навч. посіб. Київ : Знання, 2007. 216с.

2. Санітарні норми мікроклімату виробничих приміщень : ДСН 3.3.6.042-99. – [Чинний від 2000-01-01]. – К. : МОЗ України, 2000. – 42 с. – (Національні стандарти України).

3. Безопасность жизнедеятельности. /Под ред. Н.А. Белова – М.: Знание, 2000 – 364с.

4. Довідкова книга для проєктування електричного освітлення. / Під ред. Г.Б. Кнорринга. – Л.: Енергія, 1976.

5. Самгин Э.Б. Освещение рабочих мест. – М.: МИРЭА, 1989. – 186с.

6. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень. − Затвердж. постановою Головного держсанлікаря України від 01.12.1999, № 42.

7. Мотузко Ф.Я. Охорона праці. – М.: Вища школа, 1989. – 336с.

8. Борьба с шумом на производстве: Справочник / Е.Я. Юдин, Л.А. Борисов; Под общ. ред. Е.Я. Юдина – М.: Машиностроение, 1985. – 400с., ил.

9. Ергономічні умови праці при роботі з ПК : вебсайт. URL: https://pidru4niki.com/1314011239071/bzhd/yak_stvoriti_komfortni_ergonomichni_umovi_pratsi_pri_roboti_kompyuterom (дата звернення: 13.06.2022).

10. Охорона праці, основні поняття : вебсайт. URL: https://studfile.net/preview/4293310/ (дата звернення: 13.06.2022).

11. Законодавство України : вебсайт. URL: https://zakon.rada.gov.ua/laws/show/269412/ed20110625#Text (дата звернення: 13.06.2022).

12. Ергономіка : вебсайт. URL: http://www.office-solutions.ru/ergonomics/work-space (дата звернення: 13.06.2022).

13. Засоби захисту від шуму та вібрації : вебсайт. URL:https://www.sop.com.ua/article/1071-zasobi-zahistu-vd-shumu-vbrats (дата звернення: 13.06.2022).