**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою “Інженерія програмного забезпечення комп’ютерних систем”**
**спеціальності 121 “Інженерія програмного забезпечення”**

на тему: “Клієнт-серверний додаток для створення груп та організації процесу праці над проектом”

Виконав: студент 4 курсу, групи ІП-74
(шифр групи)

Чирко Ярослав Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач Алещенко О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф. д. т. н. Сімоненко В. П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)
Освітньо-професійна програма
“Інженерія програмного забезпечення комп’ютерних систем”
спеціальності 121 “Інженерія програмного забезпечення”

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

**ЗАВДАННЯ**
на бакалаврський дипломний проєкт студента

*Чирка Ярослава Юрійовича*

1. Тема проєкту *Клієнт-серверний додаток для створення груп та організації процесу праці над проектом*
керівник проєкту Алещенко Олексій Вадимович, ст. викладач, затверджені наказом
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
по університету від 11.05.2021 року № 1139-с
2. Термін здачі студентом закінченого проєкту 01.06.2021.
3. Вихідні дані до проєкту *технічна документація. Клієнт-серверна система. Spring, Java.Thymeleaf.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
*Опис предметної області, дослідження методики побудови клієнт-серверних систем за принципом MVC, дослідження необхідних технологій, розробка системи для створення груп та організації процесу праці над проектом.*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень):
структурна схема, схема взаємодії, функціональна схема.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Нормоконтроль | проф. Сімоненко В. П. | | |

7. Дата видачі завдання_____

Календарний план

| № п/п | Найменування етапів дипломного проєкту | Терміни виконання етапів проєкту | Примітки |
|---|---|---|---|
| 1. | *Затвердження теми проєкту* | *20.12.2020* | |
| *2.* | *Вивчення та аналіз завдання* | *20.12.2020-15.01.2021* | |
| *3.* | *Розробка архітектури та загальної структури системи* | *15.01.2021-11.03.2021* | |
| *4.* | *Розробка структур окремих підсистем* | *11.03.2021-17.04.21* | |
| *5.* | *Програмна реалізація системи* | *23.05.2021* | |
| 6. | *Оформлення пояснювальної записки* | *31.05.2021* | |
| *7.* | *Передзахист* | *03.06.2021* | |
| *8.* | *Захист* | *17.06.2021* | |

Студент-дипломник Ярослав ЧИРКО
(підпис)

Керівник проєкту Олексій АЛЕЩЕНКО
(підпис)

## АНОТАЦІЯ

В бакалаврській дипломній роботі було реалізовано клієнт-серверний додаток для створення груп та організації процесу праці.

Система дозволяє користувачам створювати групи для спільної роботи та надає функції, необхідні для ефективної організації процесу праці, зокрема можливість додавання завдань для користувачів, додавання нових членів команди, спілкування в межах групи та надання необхідних для роботи прав співробітникам.

## ANNOTATION

In this bachelor’s thesis client-server application for groups creation and workflow management has been implemented.

This System allows users to create groups for working together and provides necessary tools for effective workflow management, in particular abilities to add tasks for users, add new team members, communication within the group and granting necessary user rights to employees.

# Опис альбому

**бакалаврського дипломного проєкту**

**на тему:**

“Клієнт-серверний додаток для створення груп та організації процесу праці над проектом”

| Справки | Формат | Значення | Найменування | Кількість листів | № екземпляра | Додаток |
|---|---|---|---|---|---|---|
| | А4 | | Завдання на дипломний проєкт | 2 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.001 ОА | Опис альбому дипломного проєкту | 1 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.002 ТЗ | Технічне завдання проєкту | 4 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.003 ПЗ | Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Пояснювальна записка | 65 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.004 Д1 | Схема структурна — діаграма класів | 1 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.005 Д2 | Схема функціональна — блок-схема алгоритму створення групи | 1 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.006 Д3 | Схема взаємодії | 1 | | |
| | | | | | | |
| | А4 | ІАЛЦ.467100.007 Д4 | Ключові елементи коду програми | 11 | | |

| Змн. | Арк. | ПІБ | Підпис | Дата | ІАЛЦ.467100.001 ОА | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Чирко Я.Ю. | | | Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Опис альбому. | Літ. | Арк. 1 / Аркушів 1 |
| Перевірив | | Алещенко О.В. | | | | | |
| Н/Контр | | Сімоненко В.П. | | | | КПІ ім. Ігоря Сікорського ФІОТ ІП-74 | |
| Зав.Каф | | Стіренко С.Г. | | | | | |

# Технічне завдання

**бакалаврського дипломного проєкту**

**на тему:**

“Клієнт-серверний додаток для створення груп та організації процесу праці над проектом”

# ЗМІСТ

## 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Це технічне завдання поширюється на розробку Клієнт-серверного додатку для створення груп та організації процесу праці над проектом. Область застосування: різноманітні проєкти вже створених груп та заплановані проєкти без визначеної групи.

## 2. ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проєкту “Клієнт-серверний додаток для створення груп та організації процесу праці над проектом”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

## 3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка клієнт-серверного додатку, котрий надасть користувачам можливість створення груп та необхідні інструменти для організації процесу праці.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в періодичних виданнях та публікації на тему розробки клієнт-серверних додатків в мережі Інтернет.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1. Вимоги до розробленого продукту

- Можливість реєстрації та авторизації користувачів;
- Можливість створення користувацьких груп;
- Можливість додавання користувачів до груп зручним способом;
- Можливість створення завдань для користувачів групи;
- Можливість спілкування в межах групи.

### 5.2. Вимоги до програмного забезпечення

- Операційна система Windows 7/8/8,1/10 або Unix-подібна;
- PostgreSQL;
- Сучасний Веббраузер;
- Java Virtual Machine 11.

### 5.3. Вимоги до апаратної частини

- Підключення до мережі Інтернет;
- Процесор з частотою 1 ГГц або більше;
- Оперативна пам’ять об’ємом 1 Гб або більше;
- 16 Гб вільної пам’яті на носії інформації або більше.

## 6. ЕТАПИ РОЗРОБКИ

| Етап | Дата |
|---|---|
| 6.1 Вивчення літератури | 20.12.2021 |
| 6.2 Складання і узгодження технічного завдання | 15.01.2021 |
| 6.3 Аналіз структури системи, що розробляється | 11.03.2021 |
| 6.4 Створення модулів системи | 23.05.2021 |
| 6.5 Перевірка правильності роботи розроблених модулів | 24.05.2021 |
| 6.6 Відлагодження і виправлення помилок | 24.05.2021 |
| 6.7 Оформлення документації дипломного проєкту | 31.05.2021 |

# Пояснювальна записка

**бакалаврського дипломного проєкту**

**на тему:**

“Клієнт-серверний додаток для створення груп та організації процесу праці над проектом”

# ЗМІСТ

# ВСТУП

Проблема підвищення ефективності працівників під час роботи є головною проблемою для нових команд та проєктів, проте вона зустрічається і у більш досвідчених групах. Для її вирішення було створено багато методів для полегшення планування та надання зручного для розуміння представлення даних. З використанням сучасних технологій процес створення потрібного користувачеві плану значно спростився, програми що було розроблено з цією метою наразі можуть використовувати отриману від користувача інформацію для генерації різних представлень даних, а можливість зберігання великої кількості даних дозволяє дозволяє легко отримати потрібні для розрахунку певної статистики.

Окрім цього проблема пошуку потрібних членів команди є актуальною для малих проєктів, стартапів або для груп, що мають на меті створення команд для самонавчання, заняття певним хобі. Пошук таких користувачів може бути довгим процесом та відтермінувати початок роботи нової команди. Також, потрібно буде шукати додатки для спілкування між членами команди та надання можливості планування робочого процесу, у простому для розуміння співробітниками форматі.

Завданням даного проєкту було створення системи, котра дозволить користувачам створювати команди для виконання різних цілей, знаходити користувачів для співпраці та ефективно організовувати робочий процес. З огляду на це система що розробляється повинна бути легко зрозумілою для новачків та мати на меті допомогу в підвищенні ефективності праці користувачів. Також слід враховувати різні типи груп, котрі можуть використовувати дану систему. Головною ціллю є пошук потрібних користувачів або створених груп та забезпечення необхідного функціоналу для їх роботи.

# РОЗДІЛ 1

# АКТУАЛЬНІСТЬ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

## 1.1 Актуальність розробки продукту

Програмне забезпечення для організації роботи команд (роботи над проєктом) наразі є дуже актуальними, оскільки чимало компаній на даний час переводять своїх працівників на віддалений тип працевлаштування. При такому переведенні компанії часто стикаються з проблемами, котрі стосуються налагодження процесу праці, у зв'язку з цим може погіршитись працездатність співробітників а процес праці ускладнитись зважаючи на погіршення можливостей комунікації між працівниками. Для вирішення цих проблем та покращення можливостей роботи в таких умовах було створено різноманітні програми для організації робочого процесу команд. Таке програмне забезпечення зазвичай має велику кількість різних функцій задля надання простого та зрозумілого представлення даних щодо завдань, планів, проєктів команди та обов'язків кожного з її членів. Також для забезпечення потреб комунікації команд в таких програмах додають текстові, аудіо- та відео чати, це може надати можливості проведення зборів або конференцій для вирішення важливих питань, усуне неточності та покращить розуміння справи.

Таке програмне забезпечення часто орієнтується на вже створені команди, тобто для того щоб бути доданим до групи потрібно мати запрошення. Також воно створюється для використання в суворо робочому колективі, тобто хоч воно й має велику кількість функцій, вони рідко використовуються більшою частиною команди але така кількість різних функцій може заважати новачкам та тим, хто влаштовує команди, метою яких є не комерційна діяльність.

Можливість знаходити нових членів команди або вже створені команди зі списку є важливою для певних користувачів. Прикладом можуть слугувати

різні хобі або стартапи. Для них важливо мати можливість знайти користувачів незалежно від їх місця розташування проте з необхідними навичками, знанням відповідної мови та бажанням співпрацювати. Також це може бути корисним для груп метою яких є навчання, оскільки вони зможуть знайти однодумців з схожою метою та ставити завдання для правильного та вчасного засвоєння матеріалу а, також, допомагати іншим зрозуміти складний чи незрозумілий матеріал завдяки можливості чату.

Окрім цього таке програмне забезпечення може допомогти менеджерам проєктів та покращити їх роботу оскільки в такі програмні продукти зазвичай додають різні функції, котрі можуть покращити розуміння можливостей команди менеджером. Наприклад це може бути вчасність виконання завдань певним членом команди або час, котрий він зазвичай затрачує на виконання певних типів завдань. Завдяки цій інформації можна вносити зміни до робочого плану або вираховувати готовність усього проєкту або певного його етапу. Не використовуючи такі додатки отримана інформація мала б меншу точність та могла б не бути зібрана взагалі, це ускладнило б роботу менеджера.

Якщо ж брати до уваги малі компанії або стартапи, то вони за відсутності проєктного менеджера могли б отримувати ці дані та власноруч або за допомогою сторонніх програм аналізувати проблемні місця шукати шляхи їх подолання та з певною точністю розуміти на якій стадії знаходиться проєкт.

Для усунення непорозумінь у наступних розділах слід розкрити значення словосполучення “організації процесу праці над проектом”. В даному випадку розуміється можливості користувачів планувати роботу команди над проєктом шляхом створення завдань для кожного користувача на спеціальній сторінці та обговорення планів за допомогою вбудованого чату.

Створення завдань для користувачів допоможе в розумінні роботи, котру потрібно буде виконати саме даному конкретному члену групи. Це дозволить легше планувати його час на її виконання, покроково розуміти дії, потрібні

для отримання бажаного результату та мати можливість переглянути їх у будь-який час в зручному форматі. Окрім цього, можливість перегляду вже виконаних завдань надасть змогу користувачам бачити обсяг завдань, котрі вже було виконано в проєкті, розуміти з якими саме завданнями виникали труднощі, котрі з завдань було виконано раніше ніж заплановано, а котрі вже після визначеного терміну та чи вплинули на це ті завдання, котрі виконувались перед ними. Це повинно полегшити розуміння того, як саме слід організовувати робочий процес для усіх співробітників в подальшому під час роботи над даним та наступними проєктами, як можна покращити продуктивність праці та зробити передбачення більш точними а плани найбільш ефективними з можливих.

Можливість використання чату в даній програмі також може допомогти ефективно організувати робочий процес, оскільки за умови виникнення труднощів або непорозумінь їх можливо буде усунути за допомогою обговорення в чаті. Окрім цього чат може використовуватись для обміну інформацією потрібною для підтримки зв’язку в інших програмах як, наприклад додатках для проведення відео та аудіо конференцій. Також, вбудований чат покращить розуміння цілей та навичок членів команди до початку роботи, тобто при створенні команди можна буде надіслати повідомлення користувачам, що зацікавили керівника команди з питаннями щодо їх зацікавленості в проєкті та наявності необхідних для групи навичок якщо про такі не було згадано в анкеті користувача. Зважаючи на це, наявність чату в такій програмі може сильно покращити розуміння завдань, полегшити процес створення команди та сприятиме обговоренню оптимального плану робіт зважаючи на членів команди та їх можливості.

Наявність цих компонентів дозволить підвищити ефективність праці персоналу, надасть змогу швидкого вирішення проблем які можуть виникнути в процесі роботи, дозволить отримувати дані про завершені завдання та обмінюватись думками, пропозиціями або зауваженнями до певних частин робочого процесу в текстовому форматі. Завдяки тому, що дані

функції є вбудованими в програмний продукт немає необхідності перемикатись з одного додатку в інший, це дозволить уникнути частих відволікань та дасть змогу користувачеві зосередитись на важливих завданнях та повідомленнях оскільки усі потрібні дані будуть знаходитись в одному програмному продукті.

Зважаючи на нинішню ситуацію можна заявити, що популярність таких додатків буде зростати оскільки чимало організацій та груп перейдуть повністю на віддалений режим роботи тому що матимуть досвід роботи в таких умовах, а ті люди в котрих раніше не було достатньо вільного часу зможуть займатись власними хобі або навчанням з дому, отже потребуватимуть відповідних програм для підвищення продуктивності праці та покращення планування власних завдань.

Звичайно такі програми вже існують та користуються популярністю в певних сферах діяльності, проте вони мають ряд відмінностей від запланованого продукту про деякі з них вже було згадано, також зважаючи на те, що вони часто створюються для комерційних проєктів чимало функцій таких програм діють лише за умови платної підписки, прикладом цього може слугувати відсутність можливості використання певних діаграм, планувальників або навіть обмеження кількості користувачів в певній групі. З огляду на це слід детальніше розглянути таке програмне забезпечення для покращення розуміння вже існуючих аналогів та їх відмінностей від запланованого продукту.

## 1.2 Аналіз існуючих рішень

### 1.2.1Monday.com

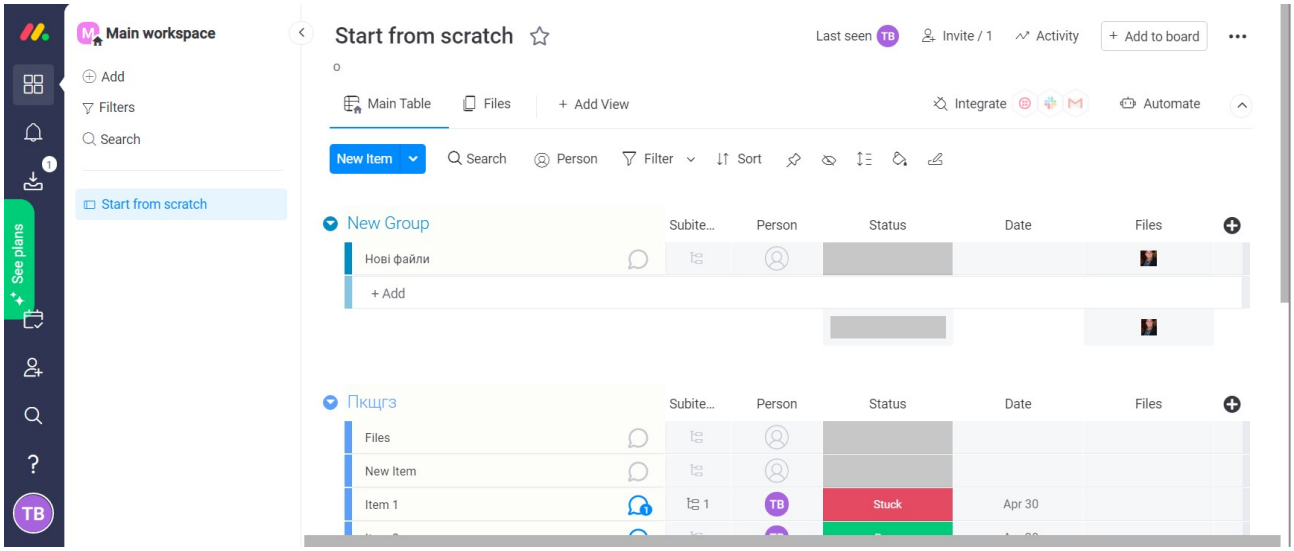



Рисунок 1.1 — Головний екран monday.com[1]

Даний сайт має інтерфейс до якого легко пристосуватись, усі потрібні функції для додавання та редагування завдань, зміни можливості перегляду таблиць та додавання нових а, також, додавання нових користувачів та призначення їх відповідальними за виконання завдань легко знайти та почати використовувати. Наявні можливості фільтрування завдань за різними критеріями, такими як їх назвою, виконавцями, групою до якої воно віднесено, часом його дедлайну, статусом завдання. Такі можливості дозволять легко знайти потрібні завдання для швидкого редагування шляхом відмічення їх та редагування потрібного поля в кожному. Є можливість розбивати одне велике завдання на підзавдання що може допомогти за потреби розділення навантаження між виконавцями та створить потрібну структуру для зменшення можливості пропуску певного етапу при виконанні завдання. Також це може допомогти при певних типах організації робочого процесу, окрім цього при розбитті завдань на менші можливо буде краще зрозуміти на якому саме етапі виконання завдання виникли труднощі що дозволить швидше їх усунути.

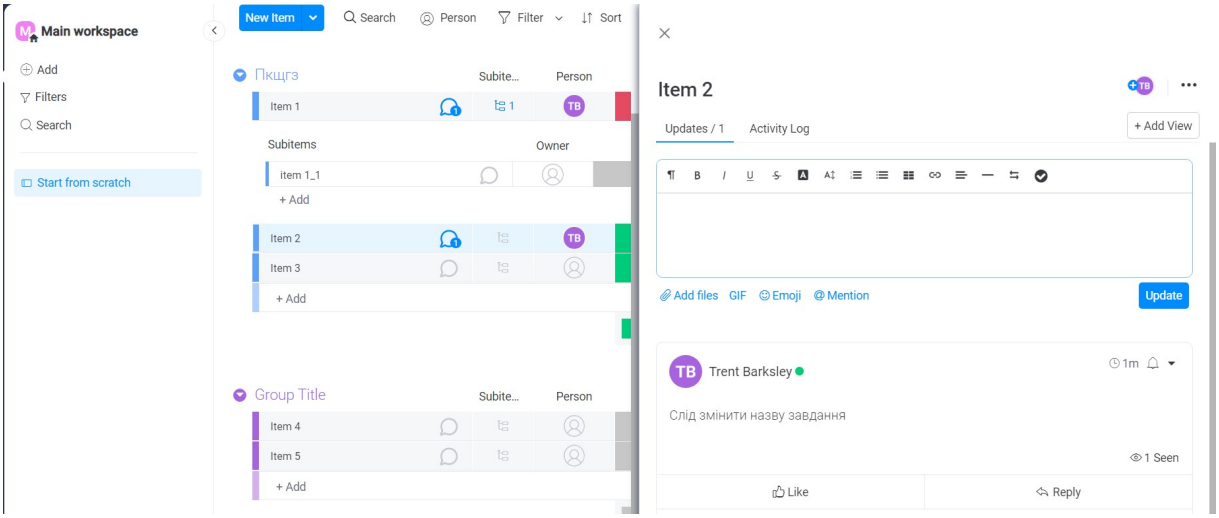



Рисунок 1.2 — Додавання відгуку до завдання в monday.com[1]

Для кожного запису користувачі можуть додавати помітки у вигляді текстових повідомлень, що може надати можливість кращого інформування виконавця або менеджера про можливі проблеми з даним завданням, його частинами або потребою змінити будь-що що здається неправильним, також сам виконавець має змогу додати посилання на результати виконання. Хоча такий підхід добре працює для одного завдання, проте за умови відсутності відповідного завдання для обговорення групі потрібно буде створити його та згодитись що саме дану тему буде обговорено в чаті даного завдання. Це може слугувати джерелом незручностей, помилок або навіть зашкодити розумінню командою їх завдань якщо такі повідомлення не будуть помічені, а як наслідок прочитані, членами групи. Звичайно цього можна уникнути за допомогою сторонніх програм для перемовин, проте це може вплинути на зручність та працездатність команди, оскільки потрібно буде часто перемикатись з однієї програми в іншу що може посприяти частим відволіканням та зменшить продуктивність користувачів. В додатку також є можливість інтеграції з різними сторонніми програмами такими як Gmail, Slack та інші, це дозволяє автоматизувати додавання завдань шляхом введення спеціальних команд в повідомлення або налаштування певних дій у відповідь на події, котрі відбулись в програмі. Це може полегшити взаємодію користувачів з програмою, пришвидшити додавання нових завдань з обговорення та надасть змогу отримувати швидкі повідомлення про

оновлення будь-якої частини завдання, його стану чи його деталей. Проте, звичайно це може призвести до частої потреби відволіктись для перевірки можливих повідомлень про оновлення що негативно вплине на працездатність користувача, також не всі подібні сторонні програми підтримуються що може стати недоліком для певних користувачів.

Також у даному додатку є можливість різного представлення користувацької інформації наприклад за допомогою діаграми Ганта, у вигляді календаря, часової стрічки тощо. Це може дозволити використання даної програми для кращого представлення даних у звітах або під час презентацій. Звичайно, для певних груп користувачів ці функції є надлишковими, проте завдяки їх розташуванню користувачі, котрі не планують використовувати їх можуть навіть не помітити їх наявність. Окрім цього в меню представлень також є можливість перегляду усіх доданих до завдань файлів.

Найбільшим недоліком даної програми для звичайних користувачів може бути неможливість додавання більш ніж одного співробітника до проєкту за умови використання безкоштовної підписки. Кількість користувачів можна збільшити, проте лише обравши тарифний план, що підходить під потреби команди ( тобто такий що дозволить додавання потрібної кількості співробітників ).

Переваги:

- ✓ Велика кількість варіантів представлення даних
- ✓ Зручний користувацький інтерфейс
- ✓ Можливість додавання файлів
- ✓ Можливість написання коментарів до завдань
- ✓ Інтеграція зі сторонніми програмами

Недоліки:

- × Лише дві людини в команді за умови використання безкоштовної версії
- × Відсутність окремих сторінок для обговорення загальних питань
- × Відсутність можливості зробити таблицю публічною

### 1.2.2 Бітрікс24

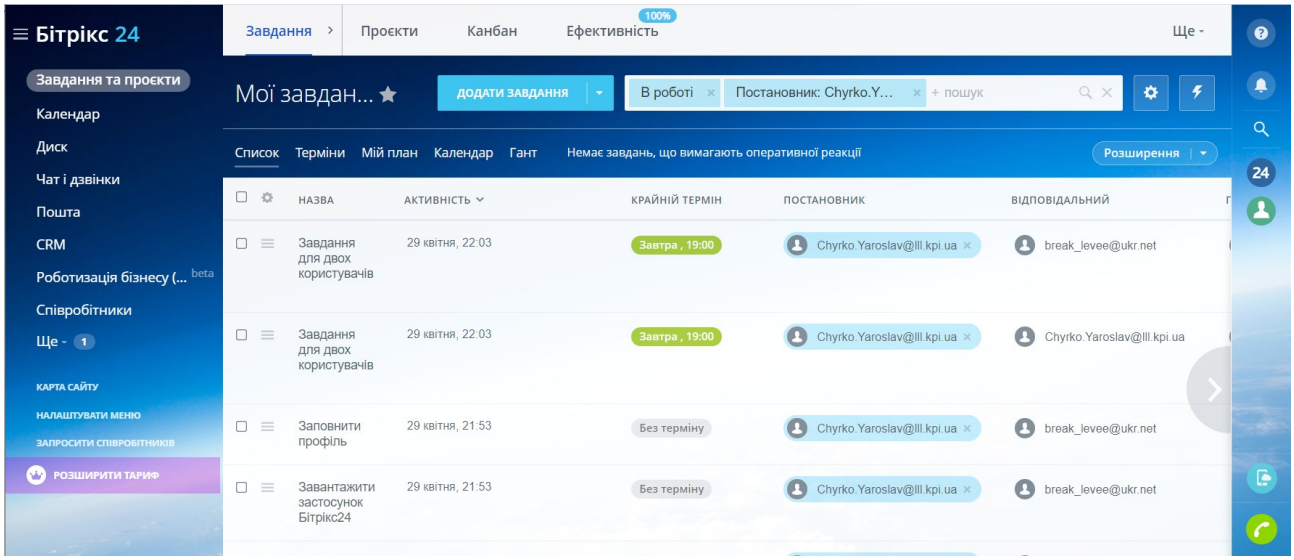



Рисунок 1.3 — Головний екран завдань в Бітрікс24[2]

Даний додаток має простий та зручний інтерфейс котрий буде інтуїтивно зрозумілий навіть для тих користувачів, котрі ніколи не взаємодіяли з подібними програмами. Для головної панелі завдань існує кілька різних представлень: списком, у вигляді календаря, у вигляді діаграми Ганта, списком що поділений залежно від терміну виконання та у вигляді плану. Кожне з цих представлень може допомогти користувачеві у розумінні або правильному представленні інформації та надасть можливість використання їх для презентацій або доповідей. Процес додавання нових завдань є простим та зрозумілим. Наявна велика кількість опцій для правильного опису завдань, можливість встановлення попередніх, часу на виконання, виконавців, постановників та спостерігачів, пов’язаних завдань, віднесення завдання до певної групи або проєкту додавати теги тощо.

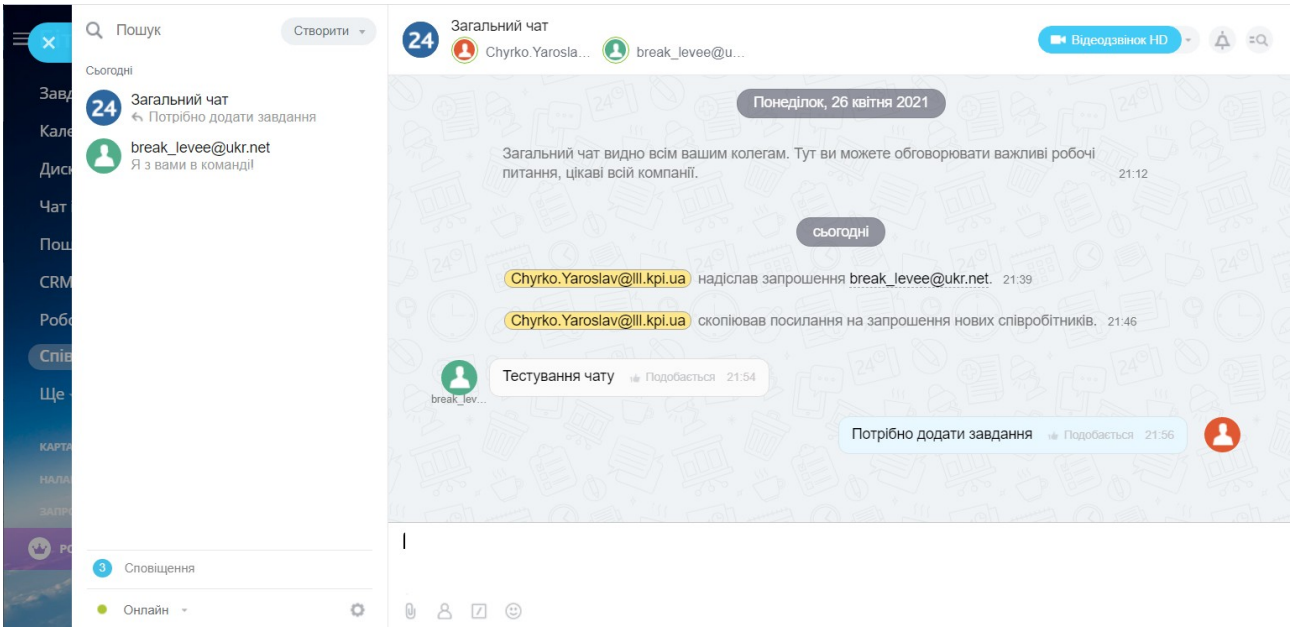



Рисунок 1.4 — Екран чату Бітрікс24[2]

Також даний продукт надає змогу використання чату для спілкування між користувачами групи, підгрупами або індивідуальними учасниками. Такий підхід надає можливість створення загального чату для групи осіб та чатів для окремих підрозділів цієї групи, наприклад окремих чатів для розробників та дизайнерів у випадку використання додатку для створення програмного забезпечення. Також є можливість здійснення аудіо та відеодзвінків.

В програмі є можливість створення груп та проєктів. Проєкти можна обирати з трьох типів: відкритого — видно всім співробітникам та усі можуть приєднатись, закритого — видно лише учасникам та приєднатись можна лише за запрошенням та зовнішнього — видимий лише учасникам, проте додати до нього можна навіть користувачів зовнішніх по відношенню до компанії. Групи можна створювати чотирьох типів, три перших повторюють типи проєктів, четвертий тип — група для зовнішніх публікацій — це група для публікації постів та новин на Бітрікс24, її видно лише членам групи та вступити до неї можна лише після схвалення модератором. Для усіх груп також є опис для розуміння їх цільового призначення. Для груп можна створювати окремі теки для збереження файлів що були завантажені саме в дану групу або використовувати загальну.

Окрім вже згаданих функцій також є різні функції пов’язані з аналітикою та можливість створити процеси котрі дозволять автоматизувати процес праці. Серед платних функції є функція створення поштових скриньок для співробітників.

Зважаючи на таку кількість різноманітних функцій також слід сказати що для нових користувачів вони можуть здатись надлишковими, слід буде вивчати для чого кожна з них може бути потрібна та як саме її слід використовувати, це може вплинути на початкову продуктивність користувача, проте надасть змогу краще працювати в подальшому.

Переваги:

- ✓ Можливість додавання співробітників без обмежень
- ✓ Велика кількість варіантів представлення даних
- ✓ Можливість використання вбудованого чату, аудіо та відеозв’язку
- ✓ Можливість створення груп та проєктів
- ✓ Можливість додавання файлів

Недоліки:

- × Відсутня можливість створення типу груп видимих для усіх користувачів.
- × Може здатись складним новим користувачам

### 1.2.3 Trello

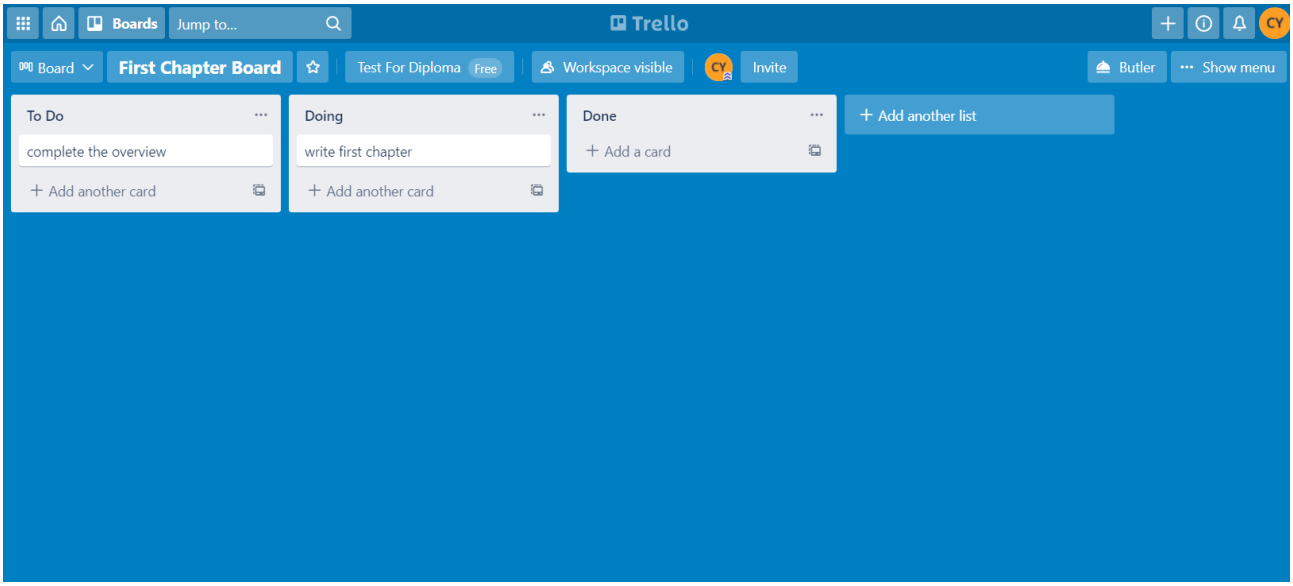



Рисунок 1.5 — Головний екран Trello[3]

Даний додаток надає можливість створення дощок для додавання на них завдань, на екрані створення можна обрати з різних готових рішень котрі були створені користувачами для певних цілей, або використати базову та змінити її відповідно до власних потреб. Це дозволяє компаніям різної спрямованості використовувати дану програму для власних проєктів.

На сторінці дошки можна додати завдання та згрупувати їх, групи можна створювати власноруч для зручності пошуку потрібного завдання. Вигляд дощок можна змінювати на календар, таблицю, графік спринтів, часову стрічку та мапу, проте дані функції доступні лише у платній версії продукту, тому за умови потреби представлення власних завдань в іншому вигляді користувач повинен буде змінити план підписки або використати сторонні додатки для створення діаграм чи таблиць. Видимість таблиць можна змінювати, можливі варіанти: приватна — лише учасники таблиці можуть її бачити та редагувати, пов'язана з робочим місцем — усі учасники робочого місця можуть бачити та редагувати її, пов'язана з організацією — усі учасники організації можуть бачити дану таблицю, публічна — усі користувачі інтернету можуть її бачити, її також можуть знаходити пошукові системи проте редагувати її можуть лише учасники таблиці.

Додавати користувачів до таблиці можна використовуючи адресу електронної пошти або відправивши посилання для приєднання.

Також у даного додатку є можливості автоматизації, котрі дозволяють автоматично виконувати певні дії за умови того, що певна подія відбулась. Це може підвищити продуктивність та усунути пені часто повторювані кроки при створенні завдань та допоможе швидко реагувати на зміни в завданнях або наближення дедлайну.

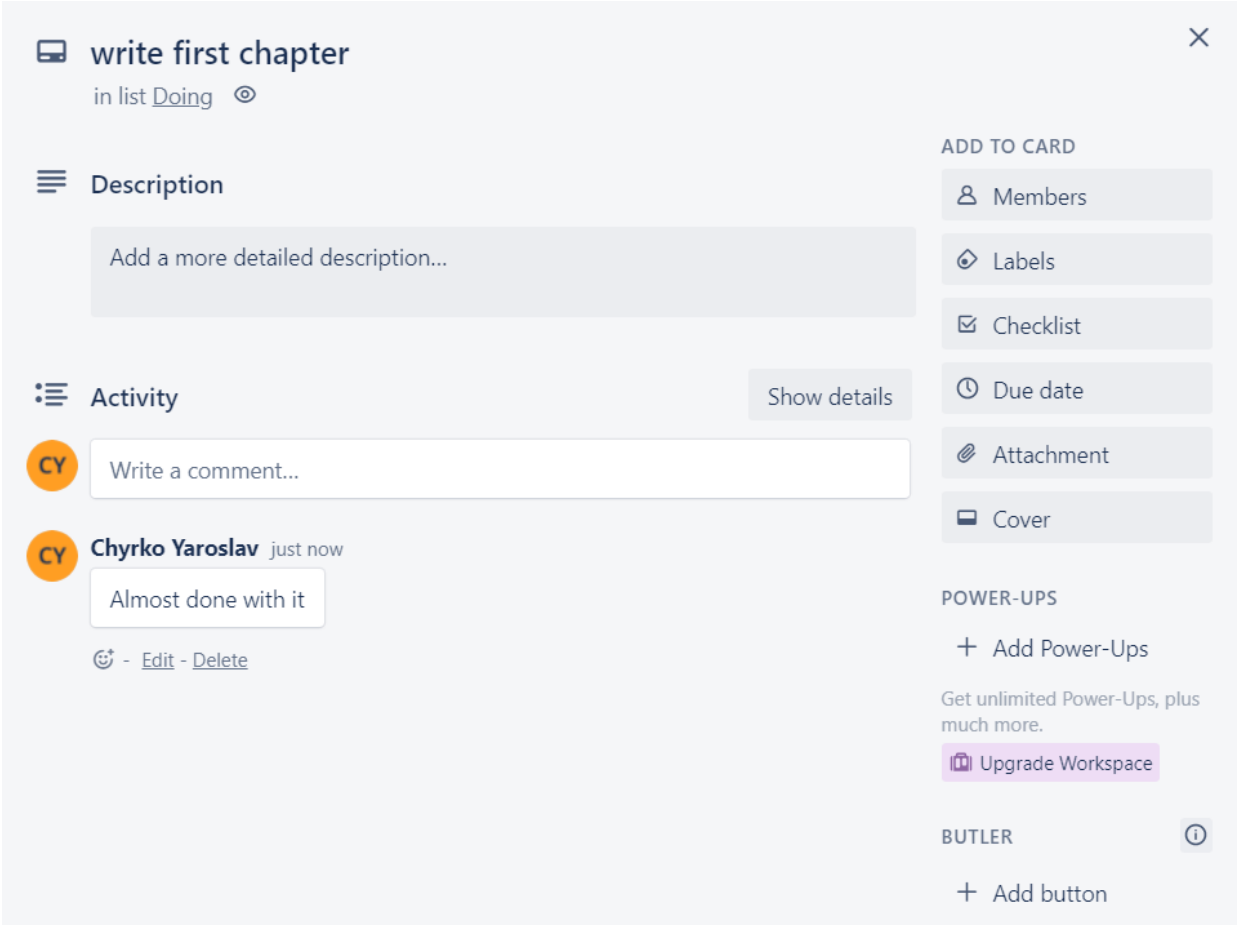



Рисунок 1.6 — Додавання коментарів до завдання в Trello[3]

Функції чату в даній програмі немає, проте є можливість додавання коментарів до певного завдання. Це може бути корисно для уточнення інформації про певне конкретне завдання або для повідомлення про помилку, проте як вже згадувалось раніше це може призвести до помилок або погіршити розуміння завдання. Можлива інтеграція даного додатку з іншими програмами, наприклад з додатком Slack для забезпечення можливості використання функції чату, проте використання сторонньої програми може

негативно вплинути на продуктивність оскільки користувачу доведеться перемикатись з однієї програми на іншу.

Окрім цього до завдання можна додати список, теги та файли. Це надасть змогу швидше знаходити потрібне завдання, мати потрібні для його виконання файли та розділити завдання на список підзавдань.

Переваги:

- ✓ Можливість додавання необмеженої кількості співробітників
- ✓ Можливість зробити таблицю видимою для усіх користувачів
- ✓ Можливість додавання файлів та коментарів до завдання
- ✓ Інтеграція зі сторонніми програмами
- ✓ Можливість використання готових шаблонів для дощок

Недоліки:

- × Різні представлення даних доступні лише за підпискою
- × Відсутність вбудованого чату

### 1.2.4 Підсумок

Розглянувши дані програми слід визначити який саме функціонал матиме система що розробляється, оскільки у рамках бакалаврської роботи не достатньо часу для того щоб реалізувати усі аспекти проєкту. Зважаючи на це та на проаналізовані системи слід визначити головні для користувачів функції та їх можливості.

Як і було зазначено раніше, у системі що розробляється повинна бути можливість обміну текстовими повідомленнями для користувачів однієї групи для надання можливості зручного обміну інформацією між ними, також повинна бути можливість знаходження груп, надсилати запити на приєднання, знаходити та додавати користувачів до вже створених груп. Пошук повинен проводитись з урахуванням мов що знає користувач та мови, котрою користуються в групі для спілкування. Окрім цього, також повинна бути можливість створення завдань для користувачів групи та можливість надання результату їх виконання у вигляді посилань на сторонні ресурси, такі як Google Drive, Git тощо. Саме такий підхід дозволить користувачам

обирати комфортну для них платформу для завантаження результатів, зменшить навантаження на систему що розробляється а різним типам груп обирати таку платформу, що найбільше підходить для їх типів завдань або використовувати одразу декілька різних.

# ВИСНОВОК ДО РОЗДІЛУ 1

У даному розділі було визначено мету створення даної програмної системи, обговорено актуальність її розробки в наш час, надано визначення організації процесу праці над проєктом в даному випадку, зазначено необхідні функції для роботи системи та розглянуто три аналогічні додатки та визначено переваги та недоліки кожного з них, що дозволить зрозуміти їх відмінності від кінцевого продукту. Також було згадано про можливі сфери застосування майбутньої програми, цільову авдиторію користувачів та обґрунтовано вибір саме даних функцій для забезпечення ефективної роботи користувачів.

Дані що було наведено в цьому розділі посприяють ефективному плануванню розробки продукту в наступних розділах.

# РОЗДІЛ 2

# ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Система, що розроблялась у ході даної роботи написана мовою програмування Java та фреймворку Spring Boot, дані що були введені користувачами зберігаються у відповідні таблиці бази даних PostgreSQL. Для представлення клієнтської частини було обрано шаблонізатор Thymeleaf, він використовує синтаксис мови HTML для розмітки сторінок, для надання сторінкам бажаного вигляду використовуються можливості таблиць стилів CSS а для надання інтерактивності сторінкам мова JavaScript та можливості її бібліотеки jQuery. Зважаючи на потребу створення чату також використовуються протоколи WebSocket та STOMP.

Далі в даному розділі буде розглянуто наведені технології, їх можливості та історія.

## 2.1 Мова програмування Java

Дану мову було обрано зважаючи на її можливості, велику кількість бібліотек та фреймворків та ознайомленість виконавця з даною технологією. Окрім цього дана мова та пов’язані з нею технології часто використовуються в комерційних проєктах отже під час виконання дипломної роботи буде отримано релевантний досвід. Дана мова часто займає провідні позиції у рейтингах використання та затребуваності.

### 2.1.1 Історія

Мова програмування Java була створена на початку 1990х років Джеймсом Гослінгом співробітником компанії Sun Microsystems. Команда почала розробку даного проєкту для використання цифровими пристроями такими як телевізори, телевізійні приставки тощо. Гослінг хотів змінити та розширити мову C++, котру спочатку планувалось використовувати для проєкту, його в подальшому назвали “OAK” (укр. “ДУБ”), проте назву змінили зважаючи на зареєстровану торгову марку Oak Technologies. [4]

Перша публічна версія Java 1.0 була представлена у 1995 році, її слоганом було “Write once, Run anywhere” (“Напишіть один раз, запустіть будь-де”) оскільки завдяки JVM (Віртуальній машині Java) програми не потребували специфічного для системи коду та перекомпіляції на пристрої оскільки спочатку Java код переводиться в спеціальний байткод що не залежить від платформи а потім виконується на JVM котра може бути різна для різних платформ. [4]

У 2009 році компанія Oracle поглинула Sun Microsystems, а отже, подальшим розвитком мови та платформи займалась дана компанія.

### 2.1.2 Особливості

Під час розробки даної мови програмування компанія дотримувалась п’яти основних принципів:

- Повинен бути використаний об’єктно-орієнтований підхід.
- Різні операційні системи повинні мати змогу запускати одну програму.
- Повинна бути можливість безпечного виконання коду з віддалених джерел.
- Повинна бути влаштована підтримка мережевих з’єднань.
- Мова повинна бути легка у використанні з огляду на те, які частини об’єктно-орієнтованих мов програмування було визначено допустимими.[6]

Оскільки Java розроблялась як розширення мови C++ використання об’єктно-орієнтованого підходу було було зрозумілим рішенням тому, що дозволяв розробникам, що використовували мову C++ легко почати використання нової мови, окрім цього такий підхід дозволяв перевикористання коду в подальшому та допомагав розробникам писати більш зрозумілий та структурований код що допомагало у підтримці великих проєктів. Незважаючи на використання об’єктно орієнтованого підходу в мові також є концепції функціонального стилю, котрі було привнесено з Java 8 для спрощення виконання певних операцій.

Можливість запуску однієї програми на різних операційних системах, як було зазначено раніше, було забезпечено за допомогою JVM, проте такий

підхід сповільнював програми у порівнянні з програмами що одразу компілювались в машинний код, для вирішення даної проблеми сучасні імплементації JVM використовують можливості JIT (just-in-time) компіляції або компілюють байт-код у машинний.

Можливість виконання коду з віддалених джерел забезпечила можливість створення аплетів для використання на сайтах, можливість їх запуску досить швидко була додана в сучасні на той час веб оглядачі, а завдяки влаштованій підтримці мережевих з'єднань мова досить швидко почала використовуватись для розробки веб додатків.

Окрім усього згаданого раніше Java також має вбудоваий механізм збору сміття Garbage Collector, котрий перевіряє Java Heap (укр. Купу) в якій зберігаються об'єкти та за умови знаходження об'єктів на які програма на даний час не має "сильних" посилань (англ. Strong reference) він їх видаляє тим самим звільняючи місце це дозволяє розробникам не використовувати сторонні збирачі сміття та не видаляти об'єкти власноруч на відміну від розробників, котрі використовують C++.

### 2.1.3 Застосування

Java використовується для розробки різних типів програм для різних операційних систем та пристроїв. Для різних потреб було створено різні версії платформ мови:

**Java SE** – має основний функціонал мови програмування Java. В ній визначені усі потрібні класи від базових типів та об'єктів до високорівневих, котрі використовуються для зв'язку з базами даних, мережевої взаємодії, парсингу XML документів, розробки графічних інтерфейсів та безпеки.

Окрім цього платформ Java SE включає в себе віртуальну машину, інструменти для розробки бібліотеки, котрі часто використовують для розробки програм мовою Java.

**Java EE** – дана платформа побудована на платформі Java SE та окрім її основного функціоналу також надає можливість розробки та запуску

великих, надійних, безпечних та багатопоточних веб додатків з можливістю їх розширення.

**Java ME** – платформа створена для запуску Java програм на малих пристроях таких як мобільні телефони. Дана платформа є підмножиною платформи Java SE з додатковими бібліотеками створеними для розробки додатків для малих пристроїв.[5]

Java часто використовується для розробки клієнт-серверних додатків, програм з графічним інтерфейсом, програм для операційної системи Android, для хмарних обчислень тощо. Зважаючи на часте використання її для написання веб додатків вона була обрана для даної дипломної роботи.

Хоча дана мова й має можливості розробки клієнт-серверних програм за допомогою технології Jakarta EE (раніше відома як Java EE), проте частіше для даних цілей використовують можливості фреймворку Spring.

## 2.2 Фреймворк Spring

Для виконання поставленої задачі з використанням обраної мови програмування слід підібрати фреймворк для полегшення завдання написання серверної частини додатку.

Для мов Java існує багато фреймворків створених з цією метою, проте переглянувши їх було вирішено використовувати Spring оскільки він є найбільш використовуваним серед інших, об’єднує в собі велику кількість проектів для різних завдань та надає можливість значно пришвидшити процес розробки. Також зважаючи на частоту його використання для нього можна часто знайти відповіді на потрібні питання та шляхи вирішення типових помилок.

### 2.2.1 Історія

Фреймворк Spring вперше був випущений під ліцензією Apache 2.0 у червні 2003 року. Ідея даного фреймворка була запропонована розробниками Юргеном Хоеллером та Яном Кароффом, вони запевнили Рода Джонсона, автора книги “Expert One-on-One J2EE Design and Development” почати

створення проєкту на основі його інфраструктурного коду з книги. В книзі Джонсон розповів про нинішній стан та проблеми Java EE та EJB а також запропонував легше рішення використовуючи звичайні Java класи (англ. POJO) та механізм впровадження залежностей (англ. dependency injection). Версія 1.0 була випущена в березні 2004 року, проте навіть до цього перша (0.9) версія користувалась популярністю у розробників. Наступні версії додавали нові функції, підтримку більш нових версій мови Java, більш зручні методи налаштування, серед них були можливості більш зручного налаштування з використанням XML, Java анотацій, та лише Java коду, підтримки вбудованих баз даних, мови виразів Spring (SpEL), вебсокетів, реактивного програмування та різного роду покращень. [8]

Окрім даного фреймворку також було розроблено проєкт Spring Boot для подальшого полегшення розробки застосунків та усунення потреби довгого налаштування програм на початку їх розробки. Розробка проєкту була почата на початку 2013 року, до цього призвів запит Майка Янгстрома котрий писав що архітектуру вебдодатків що використовують Spring можна значно спростити якщо об'єднати та вбудувати налаштування звичайних сервісів веб контейнерів в контейнері Spring котрий завантажувався би зі звичайного методу *main()*. Версія 1.0.0 була представлена в квітні 2014 року. Нові версії додавали підтримку більш нових версій мови Java, фреймворку Spring, Servlet API, серверів jetty, tomcat а також сторонніх програм. [8]

Також у фреймворку Spring є дочірній проєкт Spring Security котрий надає можливість використання розроблених механізмів автентифікації та авторизації користувачів та інші інструменти для забезпечення безпеки додатків, котрі було розроблено з використанням фреймворку Spring. Даний проєкт було започатковано в 2003 році з назвою "Acegi Security" Беном Алексом та випущено в 2004 році під ліцензією Apache License, пізніше його включили до портфоліо проєктів Spring з новою назвою, перший публічний реліз з якою був Spring Security 2.0.0 в квітні 2008 року. [11]

Найновіша на даний час версія фреймворку Spring – 5.3.6 а Spring Boot – 2.4.5

### 2.2.2 Особливості

Основні принципи Spring:

- Надання вибору на кожному етапі.
- Облаштування різних перспектив.
- Збереження сильної зворотньої сумісності.
- Встановлення високих стандартів якості коду.
- Простий та зрозумілий дизайн прикладних програмних інтерфейсів (англ. API). [7]

Ці принципи дозволяють забезпечити можливість розробникам переходити на більш сучасні версії фреймворку без потреби внесення великої кількості правок до існуючих проєктів, змінювати певні їх частини на подібні, наприклад переводити проєкт з однієї бази даних на іншу без потреби переписувати логіку (за умови використання спеціалізованих модулів), а новим користувачам легко розуміти як нещодавно додані інструменти так і ті що використовуються вже давно та знаходити інструкції та документацію котрі будуть правильними з мінімальною кількістю змін.

Фреймвок Spring складається з великої кількості різних проєктів, котрі поділені на модулі**.**

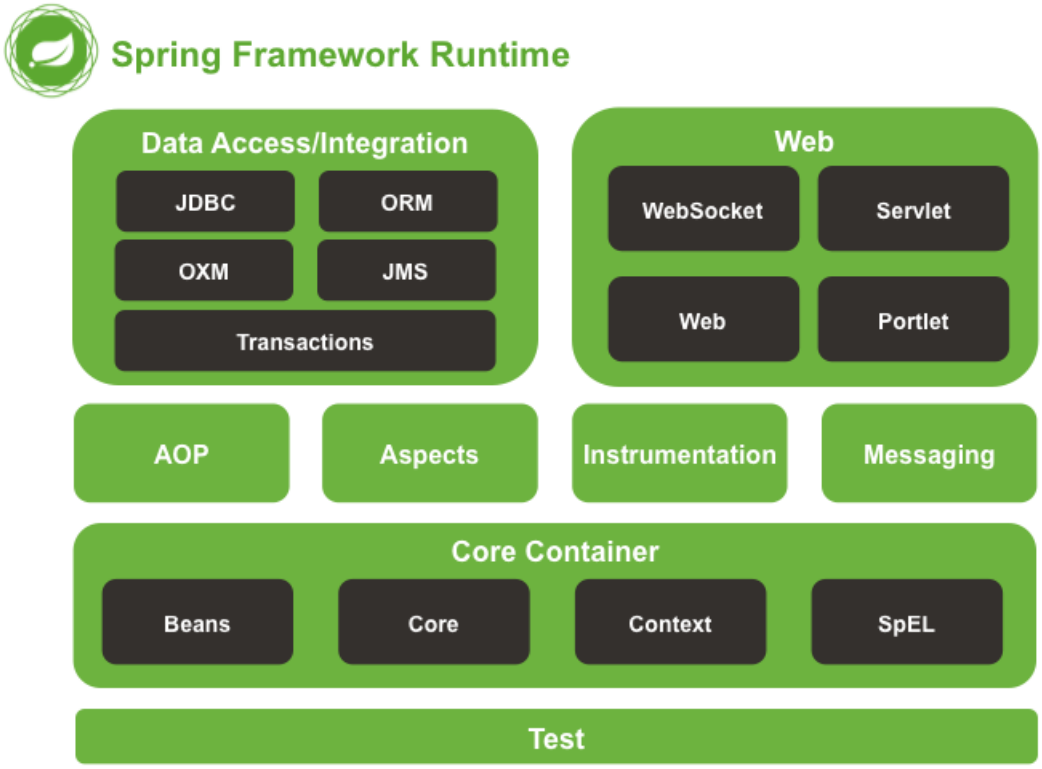



Рисунок 2.1 — Модулі фреймворку Spring[9]

Дані модулі забезпечують можливість комфортного створення різних типів додатків з використанням цього фреймворку. Вони розділені таким чином, що навіть без знайомства з їх можливостями буде зрозуміло в якому саме модулі знаходяться потрібні розробнику компоненти.

Основний контейнер (англ. Core Container) Spring – в ньому знаходяться основні компоненти архітектури Spring а саме:

- Ядро (англ. Core) - надає функції основним частинам фреймворку такі як Інверсія контролю та Впровадження залежносей;
- Bean – надає реалізацію фабричного патерну котра називається BeanFactory;
- Контекст (англ. Context) — даний модуль використовує надані модулями Ядро та Bean можливості, має права доступу до будь-яких об’єктів що він створює та налаштовує, основою частиною модуля є інтерфейс ApplicationContext;

- SpEL(мова виразів Spring) – надає можливість маніпулювання графом об’єктів під час виконання програми та використання мови запитів.

Доступ до даних/ Інтеграція (англ. Data Access/Integration) — спрощує доступ до даних, управління транзакціями, парсинг XML файлів та надсилання повідомлень, до цього контейнеру відносяться:

- JDBC (З’єднання з базою даних Java) – має рівень абстракції JDBC що усуває потребу в написанні специфічного коду для JDBC.
- ORM (Об'єктно-реляційна проекція) – хоча Spring не має власної ORM, проте даний модуль надає змогу інтеграції з популярними інструментами.
- OXM — даний модуль полегшує проекцію між об’єктами Java та XML документами, він також надає змогу інтеграції з різними популярними фреймворками.
- JMS (укр. сервіси повідомлень Java) — модуль для відправки та отримання повідомлень.
- Transactions – модуль що об’єднує кілька різних прикладних інтерфейсів для управління транзакціями**.**

Веб Контейнер (англ. Web Container) — в даному контейнері знаходяться модулі, котрі полегшують завдання розробки вебдодатків, окрім цього в модулі Web є можливість інтеграції з різними популярними фреймворками MVC (Model-View-Controller), до цього контейнера належать:

- Web — надає можливості для веб-орієнтованої інтеграції.
- Web-MVC — модуль що реалізує патерн MVC та надає імплементації потрібних для створення вебдодатків класів та функцій.
- Web-Socket — Надає можливість використання вебсокетів для створення дуплексного зв’язку між клієнтською та серверною частинами, тобто обидві частини можуть надсилати запити, також має потрібні для налаштування даного зв’язку класи та реалізацію протоколу STOMP.
- Web-Portlet – імплементує патерн MVC використовуючи середовище протлетів.

Окрім цього також є декілька важливих модулів, котрі не належать до контейнерів:

- AOP (укр. Аспектно орієнтоване програмування) — імплементує наскрізний функціонал, також він надає можливість розуміння перехоплення певних процесів перехоплювачів наприклад для додавання функціоналу під час виконання методу.
- Aspects — фреймворк для використання AOP, завдяки даному модулю можлива інтеграція з AspectJ.
- Testing — модуль що надає підтримку в написанні інтеграційних та юніт тестів з використанням відповідних фреймворків таких як JUnit або TestNG.
- Instrumentation — модуль що надає інструменти для підтримки налагодження класів, він використовується у різних серверах для додатків.
- Messaging — модуль котрий використовується для налаштування та реєстрації об'єктів повідомлень для їх використання в чергах повідомлень. [10]

### 2.2.3 Застосування

Як було зазначено раніше даний фреймворк було створено для полегшення розробки додатків Java EE, отже він дозволяє створювати клієнт-серверні додатки з можливістю легшої їх підтримки, проте даний фреймворк також може посприяти спрощенню розробки і інших типів програм, окрім цього модулі дозволять краще налагоджувати їх навіть з віддалених пристроїв, а можливість використовувати лише потрібні модулі дозволить зменшити кількість використовуваних бібліотек. Загалом використання даного фреймворку дозволяє спростити процес розробки та підтримки великих за розміром проєктів, а його розширення у вигляді Spring Boot надає можливість скоротити кількість часу потрібного для налаштування нового проєкту, окрім цього завжди є можливість змінити певні частини на аналогічні завдяки можливостям інтеграції.

Після аналізу було вирішено використовувати Spring Boot для виконання дипломного проєкту. З усіх описаних модулів, для забезпечення потрібного функціоналу у роботі використовуються Web-MVC та Web-Socket з Веб контейнеру, ORM з використанням бібліотеки Hibernate з контейнеру Доступ до даних/Інтеграція, та сам додаток буде використовувати модулі потрібні для його роботи. Окрім цього також використовується фреймворк Spring Security для забезпечення аутентифікації та авторизації користувачів.

## 2.3 Архітектура MVC

Для розробки даного проєкту було обрано архітектуру MVC зважаючи на зручність її використання та відсутність потреби створення системи, що її підтримує, оскільки як згадувалось раніше фреймворк Spring підтримує розробку веб додатків котрі використовують цю архітектуру та надає потрібні для її створення та налагодження інструменти у зручному для розробників вигляді.

### 2.3.1 Історія

Дана архітектура була винайдена Трюгве Реєнскаугом у 1978 році, тоді дана концепція називалась Thing Model View Editor (Річ-Модель-Представлення-Редактор), проте він швидко змінив назву на звичну Model View Controller (Модель-Представлення-Контролер). Спочатку цей патерн було створено з метою моделювання складних задач з реального світу для усунення прогалини між моделлю об’єктів в уявленні людини та цифровими моделями в комп’ютерах. Перша імплементація даного патерну була розроблена для мови програмування Smalltalk-80 як частина бібліотеки та використовувалась для створення користувацьких графічних інтерфейсів. На той час частини патерну визначались так:

- Модель (англ. Model) — певна інформація представлена програмою.
- Представлення (англ. View) — одне представлення інформації з моделі, з однією моделлю може бути пов’язано кілька різних представлень наприклад графіків, діаграм тощо.

- Контролер (англ. Controller) — збирає введені користувачем дані та змінює модель.

Таким чином контролер не мав зв’язку з представленням напряму, замість цього він змінював дані моделі, а модель вже впливала на представлення.

З появою веббраузерів та вебсторінок значення даного патерну змінилось. На зміну головним чином вплинула технологія JSP (JavaServer Pages) В початковій специфікації було описано два методи створення програм: Model 1 та Model 2. Саме в частині про Model 2 було описано архітектуру схожу на MVC, тут запити від браузера обробляються Сервлетом котрий генерує результат та зберігає його в компоненті, після цього він викликає сторінку JSP котра отримує доступ до даних компонента та відображає їх в браузері. В цій новій версії частини паттерну визначались так:

- Модель — Бізнес логіка та одне або більше джерел інформації.
- Представлення — користувацький інтерфейс котрий відображає інформацію моделі користувачеві.
- Контролер — механізм за допомогою якого користувач здатен взаємодіяти з додатком.

Отже в даному варіанті модель та представлення можуть взаємодіяти лише через контролер. Саме такий варіант даного патерну використовується в наш час в фреймворках та бібліотеках котрі його імплементують.[13]

### 2.3.2 Особливості

Головним завданням даної архітектури є розділення відповідальностей компонентів. Завдяки цьому створені з використанням даного патерну системи можуть легко розширятись а їх частини змінюватись без потреби зміни іастин інших компонентів. Також це надає змогу різним командам розробників в один час створювати різні частини системи (наприклад серверну окремо від клієнтської) що може значно пришвидшити процеси розробки, тестування та налагодження додатків.

Як і було зазначено раніше в даному патерні використовуються три основні компоненти: Контролер, Представлення та Модель. Взаємозв'язок між ними та їх призначення показано на рисунку 2.2

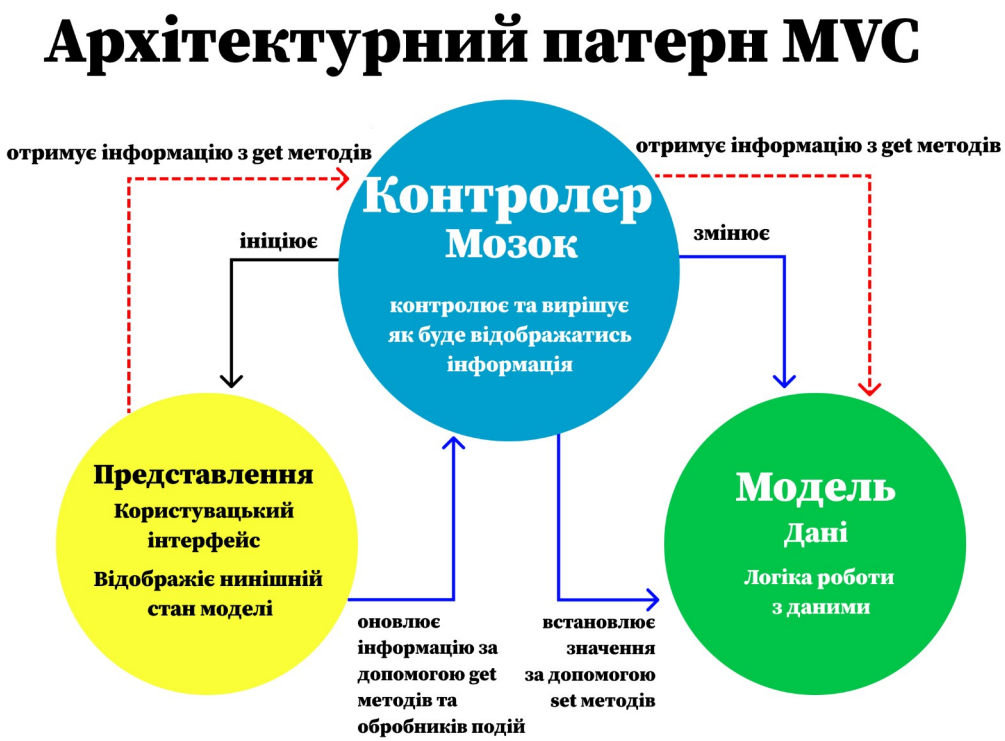



Рисунок 2.2 — Схема патерна MVC[12]

Саме на даній діаграмі компоненти описані так:

Представлення — відображає нинішній стан моделі, отримує дані за допомогою get методів через контролер та оновлює дані в контролері за допомогою set методів та обробників подій.

Контролер — контролює та вирішує як буде відображено дані, він ініціалізує представлення та направляє моделі. Окрім цього він змінює дані в моделях за допомогою методів set та отримує з них дані за допомогою get методів.

Модель — це дані та методи визначені для взаємодії з ними.

### 2.3.3 Застосування

Даний патерн використовується для полегшення та пришвидшення процесу розробки та підтримки складних проєктів та для надання можливості повторного використання коду в подальшому оскільки він

дозволяє використовувати різні представлення для одного контролера та різні контролери для одного представлення тим самим надаючи змогу зміни вигляду даних не змінюючи модель або зміни відповідей на дії користувача без зміни представлення.

На даний час існує велика кількість реалізацій даної архітектури для різних мов програмування та платформ. Звичайно, він використовується не лише для створення вебдодатків, але й для створення програм з користувацьким графічним інтерфейсом нарівні з іншими подібними патернами як наприклад MVP, MVVM та MVI.

В даному проєкті використовується можливості фреймворку Spring для створення відповідних компонентів даної архітектури за допомогою анотацій.

## 2.4 Протокол WebSocket

Для реалізації функції чату в проєкті було вирішено використовувати протокол WebSocket для надання можливості користувачам отримувати нові повідомлення без потреби оновлення сторінки через певні проміжки часу.

### 2.4.1 Історія

Даний стандарт почали розробляти Майкл Картер та Ян Хіксон в середині 2008 року для надання можливості використання повністю дуплексного з'єднання з вебсерверами. Хоча раніше отримання оновлень від веб серверів було можливим, проте воно було імплементоване за допомогою технологій polling (укр. опитування)  та long polling (укр. довге опитування), вони використовували можливості звичайного HTTP протоколу та мали недоліки та проблеми котрі розробникам доводилось вирішувати. В 2010 році з'явився перший браузер з підтримкою даної технології - Google Chrome 4, пізніше її підтримка з'явилась і в інших браузерах та на мобільних пристроях що використовують операційні системи iOS та Android. [15]

### 2.4.2 Особливості

Технологія WebSocket це тонкий транспортний прошарок побудований на стаку TCP/IP пристрою. Його головним завданням є надання прошарку зв'язку найбільш близького до TCP при цьому не забуваючи про потреби безпеки та необхідні абстракції для полегшення користування.

Для встановлення зв'язку даного типу використовується протокол потискання рук (англ. Handshake) використовуючи вже існуюче HTTP з'єднання з надсиланням заголовків оновлення (англ. update header). Зважаючи на це при перериванні такого типу зв'язку він переходить до звичайного HTTP після чого відбувається спроба його повторного встановлення. На діаграмі нижче можна побачити різницю між типами зв'язку HTTP та WebSocket.

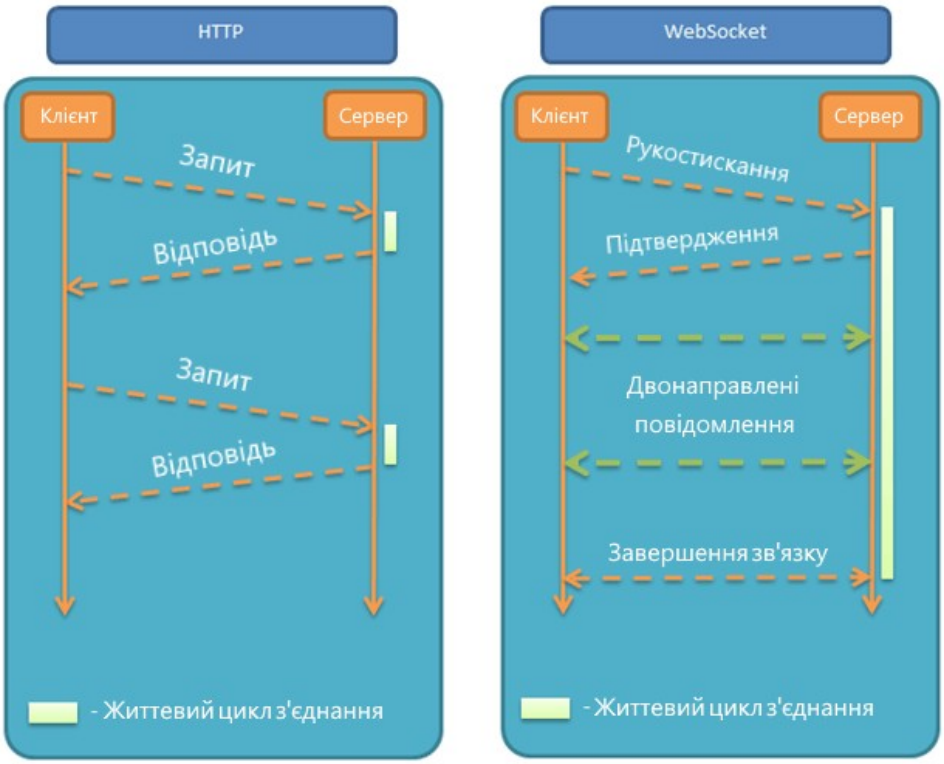



Рисунок 2.3 — Порівняння протоколів HTTP та WebSocket[14]

Як можна зрозуміти сервер котрий використовує протокол WebSocket не потребує запиту з клієнту для надсилання інформації. Якщо ж використовувати такі механізми як long polling – тобто залишення з'єднання

відкритим або звичайного polling – тобто періодичного надсилання запитів на сервер можна зіткнутись з неочікуваними проблемами.

Проте зважаючи на те, що прошарок WebSocket дуже близький до TCP він потребує додаткового протоколу (договору) для визначення того як саме інформація буде форматована, організована та інтерпретована в повідомленнях для уникнення проблем. Одним з таких протоколів є протокол STOMP, він використовує фрейми, котрі схожі на повідомлення в HTTP. Оскільки даний протокол підтримується фреймворком Spring в проєкті було використано саме його.

### 2.4.3 Застосування

В наш час дану технологію підтримують усі сучасні браузери, вона використовується на мобільних пристроях для сповіщення та оновлення інформації, окрім цього їх використовують при розробці програмного забезпечення з використанням мікросервісної архітектури.

## 2.5 Інструменти для роботи з даними

Для клієнт серверних додатків часто дуже важливим є можливість збереження введеної користувачем інформації для подальшої обробки або представлення. Для цього використовують можливості мови, бібліотек або фреймворків для створення з’єднання з обраною базою даних та надсилання запитів до неї. Окрім цього також існують різні інструмети, які дозволяють уникнути потреби написання запитів що може допомогти попередити виникнення помилок та пришвидшити процес розробки. Розробники, котрі не мають досвіду роботи з базами даних можуть створювати додатки з використанням навіть більш складних конструкцій без потреби власноруч їх писати та вивчати все що для цього потрібно. Також дані інструменти надають змогу переносити систему на новий сервер без потреби створення усіх потрібних таблиць та зв’язків між ними. Навіть при виникненні необхідності зміни використовуваної бази даних певні технології надають можливість заміни лише необхідного драйвера та деяких налаштувань для

цього що дозволяє використовувати програму без потреби переписування логіки взаємодії зі сховищем або мінімізуючи кількість коду що потрібно змінювати.

### 2.5.1 PosgreSQL

PosgreSQL – це безкоштовна система управління реляційною базою даних побудована на стандарті мови SQL та підтримую багато можливостей стандарту SQL:2011. Зважаючи на це вона надає змогу установлення різних типів зв’язків між таблицями для надання можливості реалізації певної структури відносин між різними типами даних. Окрім цього дана система підтримується різними операційними системами та має зручний та простий у використанні інсталятор. Також велика кількість мов програмування мають можливість встановлення зв’язку з цією технологією завдяки різним стороннім бібліотекам та драйверам. Звичайно, вона також підтримує можливість інтеграції з різними фреймворками що спрощує процес написання запитів та керування транзакціями.

### 2.5.2 Spring Data JPA

Дана технологія є частиною чтеку технологій Spring, вона дозволяє створювати прошарок для доступу до даних (DAO layer) що значно спрощує процес розробки додатків з використанням Spring котрим потрібна можливість доступу до даних.

Завдяки Spring Data JPA полегшується завдання створення таблиць та керування їх елементами. Для створення таблиці достатньо створити звичайний Java клас та додати до нього необхідні анотації, проте це лише за умови використання конфігурації додатку за допомогою анотацій. Після цього під час запуску проєкту буде знайдено усі подібні компоненти та створено потрібні для них поля. Назви, типи та розміри полів можна задавати за допомогою відповідних анотацій, окрім цього також можна використати анотації для встановлення можливості запису нульових значень та типу генерації первинного ключа (Id). Для виконання даних операцій

використовується ORM бібліотека Hibernate, проте його можна за бажання замінити на іншу, для уникнення проблем з анотаціями слід використовувати ті, що визначені в пакеті javax.persistence. Окрім перерахованих можливостей за допомогою анотацій можна задавати відношення між таблицями шляхом додавання відповідних анотацій над потрібним полем, ним може бути колекція у випадку відношення Один-до-Багатьох (англ. One-to-Many) або Багато-до-Багатьох (англ. Many-to-Many), в останньому випадку також можна створити композитний ключ для таблиці з'єднання.

Завдяки можливостям Spring Data JPA можна створити інтерфес, котрий буде розширяти визначений інтерфейс CrudRepository<T,ID> для обраного типу та типу ідентифікатора визначеного в ньому. Всередині цього інтерфейсу можна визначити методи використовуючи особливі назви, аргументи та типи котрі вони будуть повертати. Під час запуску проєкту фреймворк буде створювати клас для доступу до даних на основі даного методу та, розділяючи назви методів на частини за допомогою мови виразів Spring, створюватиме відповідні запити зважаючи на ці частини. Проте для зручності та просто за потреби можна використовувати й інші назви та анотацію @Query де в параметер значення (англ. value) можна передати потрібний запит написаний з використанням синтаксису HQL або SQL. Навіть без визначення методів до даного класу буде додано стандартні для збереження, отримання усіх елементів та елементу за його ідентифікатором. Для отримання об'єкту даного класу слід створити змінну типу визначеного інтерфейсу та помітити його анотацією @Autowired, завдяки цьому фреймворк створить об'єкт визначеного класу та додасть посилання на нього в дану змінну, після цього можна буде використовувати його для виконання потрібних методів.

### 2.5.3 Hibernate

Hibernate — це ORM бібліотека мови Java що має на меті надання можливостей об'єктно-реляційного представлення (англ. object-relational mapping) для мови з метою вирішення проблеми написання великої кількості

низькорівневого коду для здійснення запитів до бази даних. ЇЇ можна використовувати як з уже існуючими таблицями так і для створення нових. Вона реалізує специфікацію JPA, тому сумісна з технологіями котрі цього вимагають. Для створення представлення достатньо створити звичайний клас Java та відповідним чином позначити його в конфігурації, окрім цього вимагається наявність у такого класу стандартного конструктора, оскільки бібліотека створює об’єкт після чого викликає відповідні set методи для встановлення потрібних полів. В Spring дана бібліотека використовується за замовчанням для роботи з базами даних.

### 2.5.4 Використання

Під час розробки продукту для виконання завдання бакалаврської дипломної роботи дані технології використовувались для створення потрібних таблиць, встановлення необхідних зв’язків між ними, додавання, редагування, читання та отримання потрібних елементів таблиці. Їх використання дозволило уникнути можливих помилок та заощадити час, окрім цього надало можливість зручного форматування отриманих з таблиць даних, уникнути потреби вирішення проблем з відповідністю типів даних в таблиці та моделях та зменшити кількість коду, потрібного для реалізації системи.

## 2.6 Мова розмітки HTML

У даному дипломному проєкті мова розмітки HTML використовується для створення розмітки документів веб сторінок представлення, тобто для забезпечення графічного інтерфейсу для користувачів проте, зважаючи на статичну природу таких документів, для створення представлень використовується шаблонізатор (шаблонний двигун) для надання можливості відображення інформації з моделей та спрощення написання шаблонів розмітки.

### 2.6.1 Історія

Мова розмітки HTML була створена працівником компанії ЦЕРН (англ. CERN) Тімоті Бернесом Лі у 1991 році для оформлення документів Всесвітньої Павутини (World Wide Web), його створили як послідовника стандартної узагальненої мови розмітки (SGML), проте основною метою було створення простої для розуміння мови котра могла б бути відображена незалежно від екрану пристрою. Перша версія мови була створена у 1991 році та мала невелику кількість тегів для розмітки. В наступних версіях додавались нові теги для кращого форматування тексту, проте у версії 3.2 котра вийшла 14 січня 1997 року HTML пристосували до використання таблиць стилів CSS специфікація для яких була затверджена лише за місяць до цього. В наступній версії 4.0 частину тегів помітили як застарівшу, окрім цього додались можливості використання скриптів а деякі теги були удосконалені. 28 жовтня 2014 року завершився процес специфікації HTML 5, в цій версії була додана підтримка мультимедійних технологій а синтаксис став більш суворим у порівнянні з минулими версіями.[17]

### 2.6.2 Застосування

На даний момент мова HTML залишається стандартним інструментом для відображення інформації в браузерах, проте велика кількість різноманітних інструментів може допомогти вирішити проблеми написання великої кількості розмітки створюючи спеціальні шаблони. Також мова JavaScript, спеціальні бібліотеки та фреймворки для неї мають змогу вносити зміни в розмітку HTML документів додаючи динамічності та створюючи більш приємні вебсторінки, а в мережі Інтернет можна знайти велику кількість готових сторінок котрі надаються для використання в проєктах та різні програми для генерації розміток за наданим дизайном.

## 2.7 Шаблонізатор Thymeleaf

Шаблонізатор (також відомий як шаблонний двигун) — програмне забезпечення, створене для об’єднання шаблонів документу з даними з

моделей для перетворення їх в документи. Таке програмне забезпечення використовує власну мову відому як мова шаблону для написання шаблонів, вона часто надає можливість використання таких функцій як:

- створення змінних та функцій
- заміна тексту
- використання циклів та умовних операторів
- додавання сторонніх файлів. [16]

Таке програмне забезпечення дозволяє швидше створювати документи, котрі можна повторно використовувати з іншими даними моделей.

Thymeleaf – це шаблонізатор розроблений використовуючи мову програмування Java з метою заміни стандартних сторінок JSP. Окрім цього під час його розробки використовувався принцип натуральних шаблонів (англ. Natural Templates), тобто таких, котрі можна відкрити в браузері без потреби використання спеціального середовища і їх зміст буде відображатись правильно. Такий підхід дозволяє командам розробників мати більш чітке розуміння того як саме буде виглядати те чи інше представлення навіть за умови відсутності потрібного функціоналу. Це дозволяє зменшити час на тестування прототипів та надає можливість перевірки правильності представлень без потреби запуску усієї системи.[16]

Також дана технологія підтримує можливість створення так званих діалектів для надання користувачам можливості персоналізації шаблонів та можливості їх використання для специфічних потреб проєкту.[16]

Цей шаблонізатор може працювати як у веб середовищі так і в звичайних та використовується для створення шаблонів документів розміток XML, HTML5 та XHTML. Він найбільш пристосований для надання представлень у додатках що створені на базі архітектури MVC.[16]

Для розробки дипломного проєкту дана технологія була обрана зважаючи на її можливості інтеграції з фреймворком Spring, можливістю використання особливого діалекту для використання усіх переваг даної інтеграції – SpringStandard та простим синтаксисом, оскільки для

використання особливих функцій даного шаблонізатора використовуються лише спеціальні атрибути HTML тегів та особливий синтаксис для отримання значень з моделей фреймворку що надає змогу писати розмітку для сторінок таким чином, що вона є найбільш наближеною до звичайної розмітки HTML.

## 2.8 Таблиці стилів CSS

Таблиці стилів CSS це інструмент, що дозволяє змінити стиль відображення HTML документу таким чином, щоб він відповідав розробленому дизайну сторінки та був зручним зрозумілим та приємним для використання відвідувачами сайту. Окрім цього дана технологія надає можливість використання декількома сторінками одних стилів для відповідних елементів розмітки, тобто використовуючи її можна задати однаковий стиль для усього додатку, або використовувати для кожної сторінки власний стиль якщо того потребує дизайн. Таким чином одна сторінка HTML може виглядати зовсім по-різному залежно від застосованих для неї стилів. Також ця технологія має спеціальні функції для полегшення розробки адаптивних сторінок, тобто розміщення та властивості певних елементів розмітки будуть залежати від розміру екрану користувацького пристрою та його положення, це дозволяє уникнути потреби створнення різних розміток для різних типів пристроїв та розмірів екрану.

### 2.8.1 Історія

Розробка даної технології розпочалась в середині 1990-х років для забезпечення кращого поділу між тим як виглядає документ та його структурою розмітки. На той час для стилізації HTML документів застосовувались спеціальні теги. Таке розділення планувалось ще з початку розробки HTML та зважаючи на відсутність опублікованого синтаксису таблиць, запропонованих Тімоті Бернесом Лі у створеному ним браузері, кожен вебпереглядач мав власний визначений стиль відображення сторінок. У жовтні 1994 року Хокон Віум Лі запропонував перший варіант CSS котрі в

на той час мали назву каскадні таблиці стилів HTML, проте потім назву було змінено зважаючи на можливість їх використання і з іншими мовами розмітки. Звичайно на момент 1994 року вже існували такі технології, проте вони мали обмежений функціонал, складний синтаксис, не були пристосовані для використання на вебсторінках або використовувались лише обмеженим колом розробників браузерів. Окрім нього в той час над власною мовою таблиць SSP (Stream-based Style sheet Proposal) для власного браузера Argo працював Берт Брос, після публікації першого варіанту CSS він почав співпрацю з Лі. Головною перевагою у порівнянні з іншими запропонованими в той час технологіями була можливість об'єднання стилів браузера, сторінки та користувача що надавало можливості для персоналізації. У грудні 1996 року з'явилась перша версія рекомендація W3C першої версії CSS котра називалась CSS рівень 1 (англ. CSS level 1) або скорочено CSS1, як вже було згадано раніше через місяць після цього нова версія HTML вже мала підтримку даної технології, проте ще кілька років браузери підтримували лише частину її функцій. У травні 1998 року було опубліковано рекомендацію для CSS рівень 2 в якій були додані функції що не ввійшли до першого рівня зважаючи на часові або технічні обмеження та нові функції. Розробка CSS рівень 3 почалась у 1998 році , проте рекомендація для неї досі не була опублікована. З нових функцій була додана можливість створення анімацій на сторінці без використання мови JavaScript та нові можливості для налаштування відносин розташування між елементами сторінки перші напрацювання яких були представлені в CSS рівень 2.1. Окрім цього певні функції CSS вже досягли рівнів 4 та навіть 5, для них було обрано створити окремі специфікації на відміну від інших рівнів для яких було створено загальну специфікацію, інші ж нововведення були створені для рівня 1 та в даний час розвиваються.[18]

### 2.8.2 Застосування

В наш час дану технологію використовують для різних типів вебдодатків для надання їм привабливого вигляду, створення відчуття динамічності

сторінок, розташування елементів на екрані таким чином, щоб створена сторінка відповідала розробленим макетам. Використання даної технології дозволяє розмежувати розмітку сторінки, котра розробляється та її зовнішній вигляд, що надасть можливість більш точно розуміти де виникла проблема та як її усунуть, також це дозволяє зменшити кількість тегів, котрі використовуються на сторінці, робить можливим повторне використання стилів оскільки одну таблицю можна використовувати на будь-якій сторінці що дозволяє надавати однаковий стиль усьому проєкту та навіть різним проєктам якщо розробник прагне досягти саме цього.

Також для даної технології існує багато різних технологій для її розширення та доповнення. Мови розширення додають функції створення змінних, написання вкладених виразів, умовних операторів, операцій зі змінними та вцілому полегшують написання CSS. Окрім них також існують різноманітні фреймворки та бібліотеки на меті яких поліпшення існуючих або додавання нових функцій.

Проте слід також зауважити що дана технологія використовується не лише для веб сторінок але й для форматування певних типів електронних книжок та документів, окрім цього її використовують для покращення розмітки користувацьких інтерфейсах в певних додатках котрі використовують мову розмітки що підтримує використання стилів CSS.

У додатку що розробляється в ході виконання дипломного проєкту дана технологія використовується для надання представленням потрібного вигляду, окрім цього вона також надає можливість створення адаптивного дизайну додатку, тобто такого, котрий буде змінювати розташування певних компонентів залежно від розміру екрану користувача.

## 2.9 Мова програмування Javascript та бібліотека jQuery

### 2.9.1 JavaScript

Мова програмування Javascript – це динамічно типізована мова програмування що реалізує стандарт ECMAScript та надає можливість

використання кількох парадигм для написання коду, разом з HTML та CSS є однією з основних технологій на яких побудована всесвітня павутина. Дану технологію використовують для написання певної моделі поведінки клієнтської частини вебдодатків, завдяки ній можна змінювати елементи моделі DOM, додавати нові, встановлювати асинхронне з'єднання з серверною частиною. Окрім цього завдяки можливостям маніпуляції об'єктами DOM раніше дана технологія використовувалась для створення анімацій на вебсторінках.

В наш час існує велика кількість технологій, котрі дозволяють використовувати дану технологію не лише як було написано раніше, проте й для створення серверної частини додатку та полегшення створення клієнтської частини. Саме це дозволило розробникам, котрі використовували лише дану мову, HTML та CSS створювати повноцінні веб додатки. Окрім цього дана мова за допомогою бібліотек та фреймворків надає змогу розробки прогресивних вебдодатків, створення додатків для персональних комп'ютерів тощо.

### 2.9.2 Модель DOM

Об'єктна модель документа(англ. Document Object Model, DOM) — незалежний від мови інтерфейс що представляє XML та HTML документи у вигляді дерева. Кожен вузол якого це об'єкт що відображає частину документу. У вебпереглядачах використовується схожа внутрішня структура для відображення HTML документів. При відображенні документу браузер завантажує його в локальну пам'ять та парсить його для правильного виведення на екран. Взаємодія з об'єктами даної моделі забезпечується шляхом використання спеціальних методів. [19]

### 2.9.3 jQuery

Головним завданням бібліотеки jQuery є спрощення написання звичайних функцій мови JavaScript таких як здійснення AJAX запитів та

проведення маніпуляцій DOM. Завдяки цьому багато складних у написанні та великих за розмірами функцій можна замінити на один метод з даної бібліотеки. Це дозволяє пришвидшити процес розробки клієнтської частини та зменшити кількість помилок пов’язаних з частим написанням коду для виконання однакових дій. Також є можливість маніпулювання елементами CSS та додавання різних ефектів та анімацій на сторінку.

Можливість виконання AJAX запитів та зміни елементів моделі DOM надає можливість отримання нової інформації з серверної частини та її відображення без потреби перезавантаження усієї сторінки. Такий підхід дозволяє створювати більш приємний для користувачів інтерфейс а за умови частого оновлення даних в системі користувач матиме змогу отримувати найновішу інформацію та використовувати інший функціонал одночасно, прикладом цього може слугувати будь-яка система що може отримувати введені користувачем дані та відображати їх на сторінці для всіх, таким чином користувач матиме змогу бачити нові дані отримані системою та в той самий час використовувати поле форми для введення власних даних що за умови постійного перезавантаження усієї сторінки а не лише її частини очищало б поле для введення.

### 2.9.4 Використання

У даному дипломному проєкті ці технології використовуються для постійного оновлення даних чату. Для встановлення зв’язку з сервером використовуючи протокол WebSocket було використано бібліотеки Sock.js та Stomp для мови програмування JavaScript, завдяки цьому є змога встановити зв’язок з визначеними на сервері кінцевими точками використовуючи однаковий протокол.

Бібліотека jQuery допомагає отримувати нові дані при надходженні повідомлень на певні кінцеві точки та перезавантажувати потрібні компоненти веб сторінки для їх відображення в представленні.

# ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі було розглянуто технології що використовуються для розробки додатку для виконання бакалаврського дипломного проєкту. Було наведено їх можливості та основні способи використання. Також було зазначено з якою метою вони використовуватимуться при розробці.

Розглянувши дані технології можна з впевненістю сказати, що вони відповідають вимогам майбутньої системи, нададуть в подальшому можливість правильної реалізації її функцій та можливість легкого розширення за умови виникнення такої потреби. Також дані технології обирались з огляду на те, що вони надають можливість пришвидшення процесу розробки та зменшення вірогідності виникнення помилок оскільки за умови пізнього їх виявлення вони можуть вплинути на роботоздатність усього проєкту та зменшити кількість часу що буде виділена на впровадження необхідного функціоналу.

Окрім цього також розглядалась популярність, актуальність та частота використання оскільки виконавець зможе отримати корисний досвід використання технологій що є затребуваними, здобуде навички розробки з їх використанням що може знадобитись в подальшому. Поєднання даних технологій надає йому можливості створення інших клієнт-серверних додатків навіть використовуючи іншу більш складну архітектуру.

Зважаючи на викладену в даному розділі інформацію можна стверджувати про доцільність використання даних технологій при виконанні завдання дипломного проєкту.

# РОЗДІЛ 3

# ІНСТРУКЦІЯ З ВИКОРИСТАННЯ ТА ОПИС ПРОЄКТУ

## 3.1 Реєстрація/Вхід

### 3.1.1 Інструкція

Для початку роботи даною програмою потрібно зареєструватись або виконати вхід в уже зареєстрований акаунт. Обрати метод авторизації можна на початковій сторінці натиснувши на відповідну кнопку.

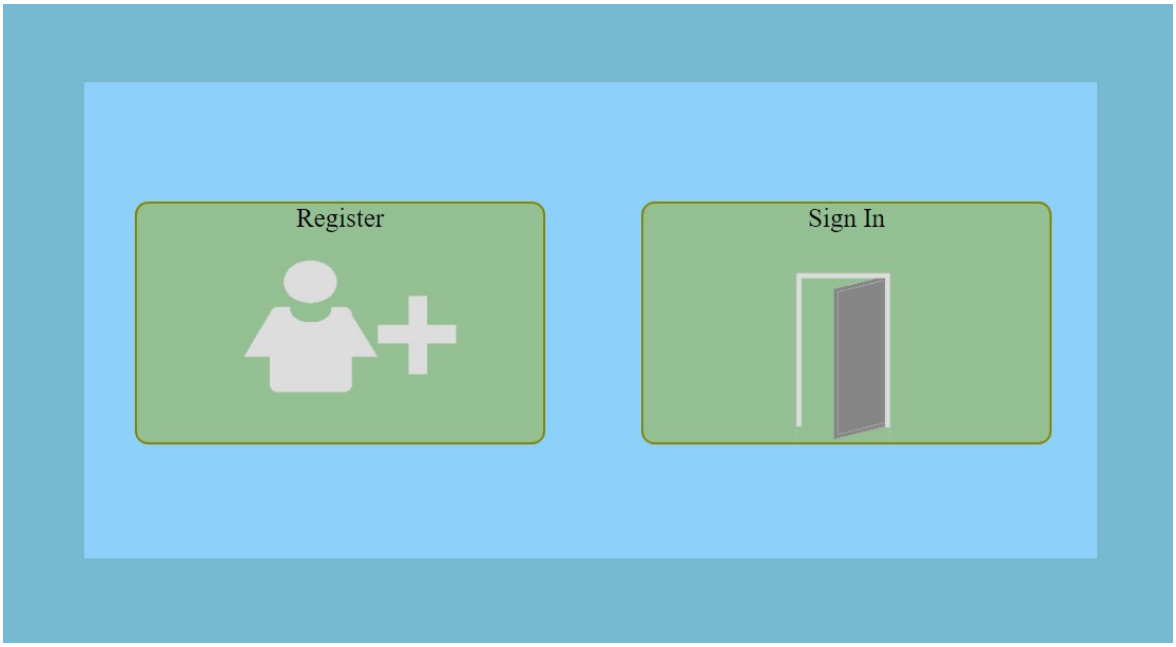



Рисунок 3.1 — Початкова сторінка програми

Після цього буде завантажено відповідну сторінку.

На сторінці реєстрації користувач повинен заповнити потрібні поля та обрати мови. Поле “Username” повинне бути заповнене та містити не менше ніж три символи, “Password” повинне бути заповнене та містити шість або більше символів, “e-Mail” повинне бути заповнене, проте адреса не перевіряється. Поле “Bio” не є обов’язковим, проте воно слугує для більш детального опису користувача. Для вибору декількох мов потрібно натиснути кнопку “Ctrl” на клавіатурі та обрати потрібні. За умови неправильного введення користувацької інформації будуть виведені повідомлення про

помилку біля кожного з полів з неправильними даними. Кнопка, що знаходиться біля поля для введення паролю, дозволяє переглянути його зміст у текстовому форматі.


Рисунок 3.2 — Сторінка реєстрації

На сторінці входу до акаунту потрібно заповнити усі поля та натиснути кнопку “Sign In”. Якщо було введено неправильні дані на сторінці буде відображено повідомлення про це. Окрім цього на даній сторінці також є посилання для зміни паролю у разі що користувач його забув, для цього слід натиснути на посилання “Forgot password?”.


Рисунок 3.3 — Сторінка входу

Для відновлення паролю потрібно ввести адресу електронної скриньки користувача у відповідне поле та натиснути кнопку “get reset link”, за умови, що користувач з такою адресою існує на неї буде відправлено листа з посиланням на сторінку відновлення а на даній сторінці відображено повідомлення що листа було надіслано, якщо ж користувача не було знайдено буде відображено повідомлення що такого користувача не було зареєстровано або введені дані є неправильними.


Рисунок 3.4 — Сторінка відновлення паролю

Після здійснення переходу за посиланням, що було отримано у листі, буде завантажено сторінку зміни паролю де слід заповнити поле та натиснути кнопку “change password” після чого, за умови відповідності нового паролю встановленим обмеженням, пароль буде змінено.

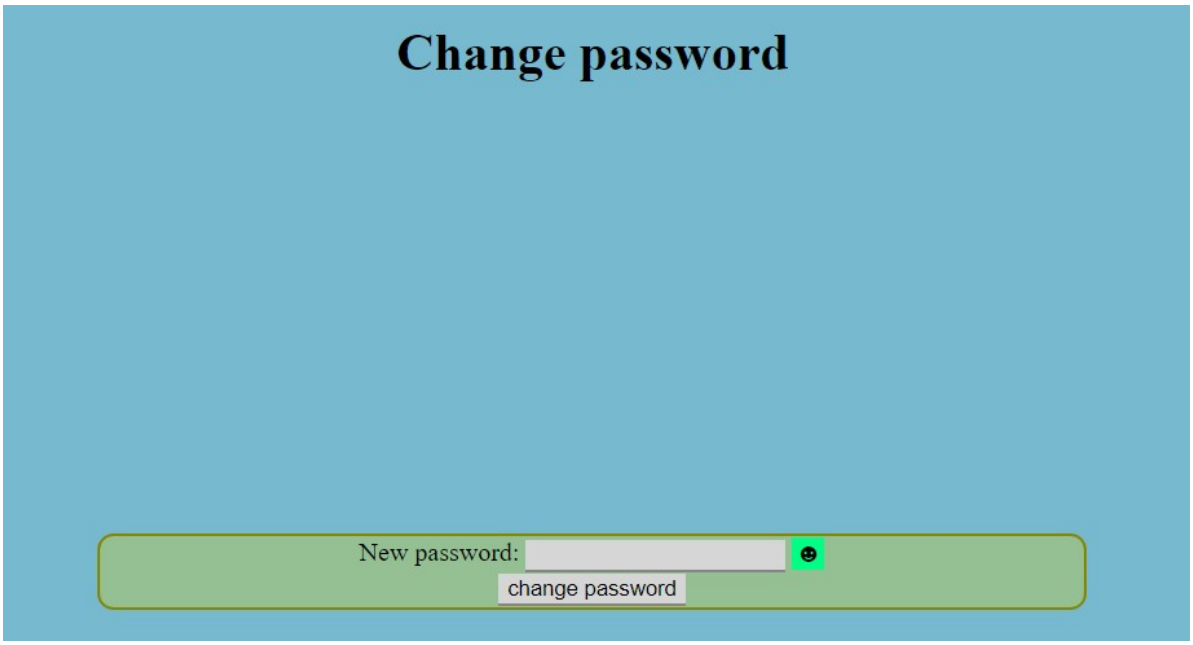



Рисунок 3.5 — Сторінка зміни паролю

Після проходження реєстрації, входу або зміни паролю користувача буде авторизовано та перенаправлено на головну сторінку додатку.

### 3.1.2 Опис

Під час реєстрації створюється об’єкт типу UserModel. Він має відповідні поля анотовані для забезпечення відповідності введених даних очікуванням системи та для створення об’єктно-реляційного представлення. Окрім цього в ному визначені поля для встановлення зв’язків з іншими таблицями (таблицею мов, токенів та груп). Для цього класу визначений інтерфейс UserRepository для забезпечення зв’язку з відповідною таблицею бази даних та надання потрібних методів. Паролі усіх користувачів зберігаються у зашифрованому вигляді. У разі правильного заповнення форм на даній сторінці направляється запит для збереження даного об’єкту після чого користувач авторизується в системі. Посилання, що використовуються для відображення та обробки запитів даних сторінок доступні для не авторизованих користувачів.

Під час виконання входу потрібний користувач авторизується в системі.

Якщо було використано відновлення паролю то для користувача створюється новий об’єкт класу PasswordResetTokenModel та за допомогою

можливостей класу JavaMailSenderImpl надсилається повідомлення на зазначену адресу електронної пошти користувача зі створеним для нього посиланням. Після переходу за даним посиланням користувач може змінити пароль, але лише за умови, що з часу створення токену пройшло менше ніж дві години. Якщо ж пройшло більше часу або користувач вже змінив пароль використовуючи дане посилання, то при наступному переході за ним токен буде видалено. Після успішної зміни паролю користувача буде авторизовано.

Після авторизації до об’єкту користувацької сесії додається потрібний запис (значення адреси його електронної пошти) а користувач перенаправляється на головну сторінку.

## 3.2 Огляд можливостей головної сторінки

### 3.2.1 Інструкція

Головна сторінка програми дозволяє створювати групи, надсилати запити уже створеним групам для приєднання, переглядати інформацію про зареєстрованих користувачів, обирати зі списку власних груп для переходу на їх основну сторінку, здійснювати перехід до сторінки профілю користувача та редагувати користувацьку інформацію.

Інформація про групи, до яких належить користувач знаходяться в лівому стовпчику. Після натискання на одну з них користувача буде перенаправлено на основну сторінку відповідної групи.

Інформація про зареєстрованих користувачів знаходиться в середньому стовпчику. Тут відображаються лише ті користувачі, котрі під час реєстрації обрали хоча б одну з мов, котру обрав і даний користувач, сам користувач тут відсутній.

Групи, до яких даний користувач не входить знаходяться в правому стовпчику. Це групи, мова котрих також є однією з тих, що користувач обрав. Завдяки кнопці, котра знаходиться на елементі, можна надіслати запит на приєднання до відповідної групи.

У верхній частині сторінки знаходяться функціональні елементи. Вони дозволяють перейти до сторінки користувача, створити групу та вийти з акаунту.

Для створення групи слід заповнити поля в верхній центральній частині сторінки. Поле “Group Name” обов’язкове для заповнення та довжина введених даних повинна бути щонайменше три символи, “Group Description” не є обов’язковим та слугує для опису створюваної групи максимальний розмір опису — тисяча символів.

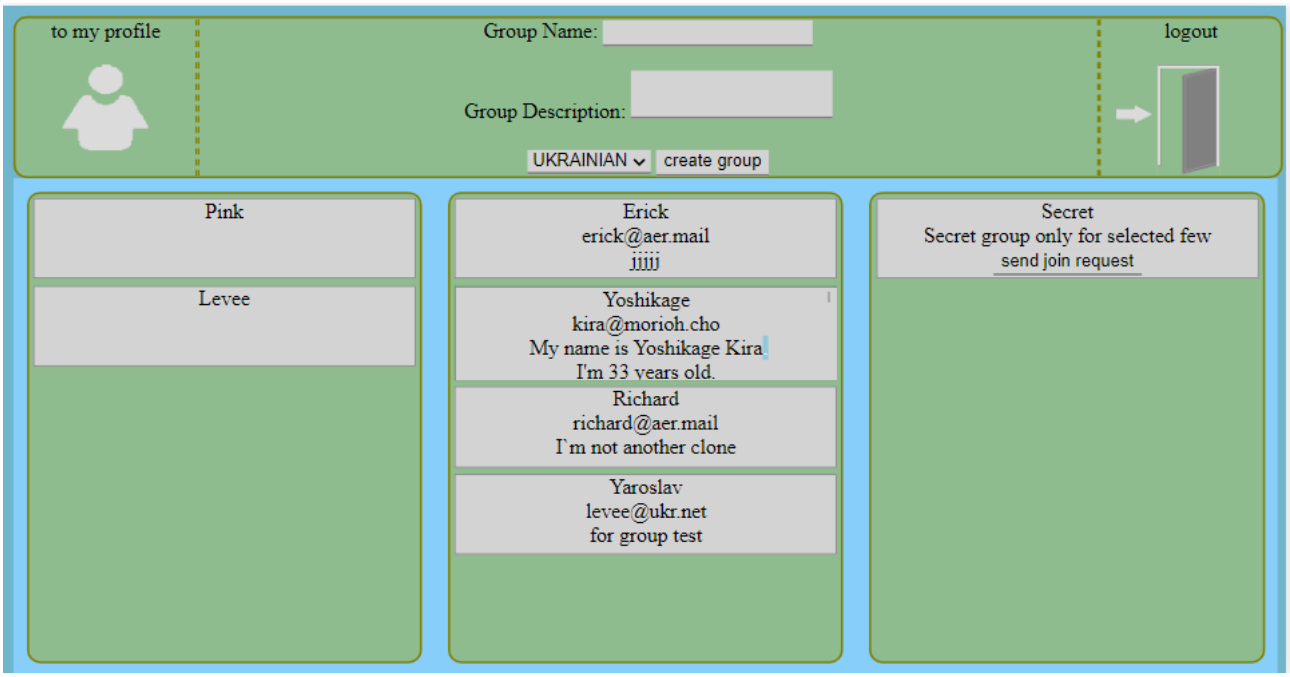



Рисунок 3.6 — Головна сторінка додатку

Після натискання кнопки “to my profile” буде завантажено сторінку користувацького профілю. Тут можна переглянути інформацію про користувача.

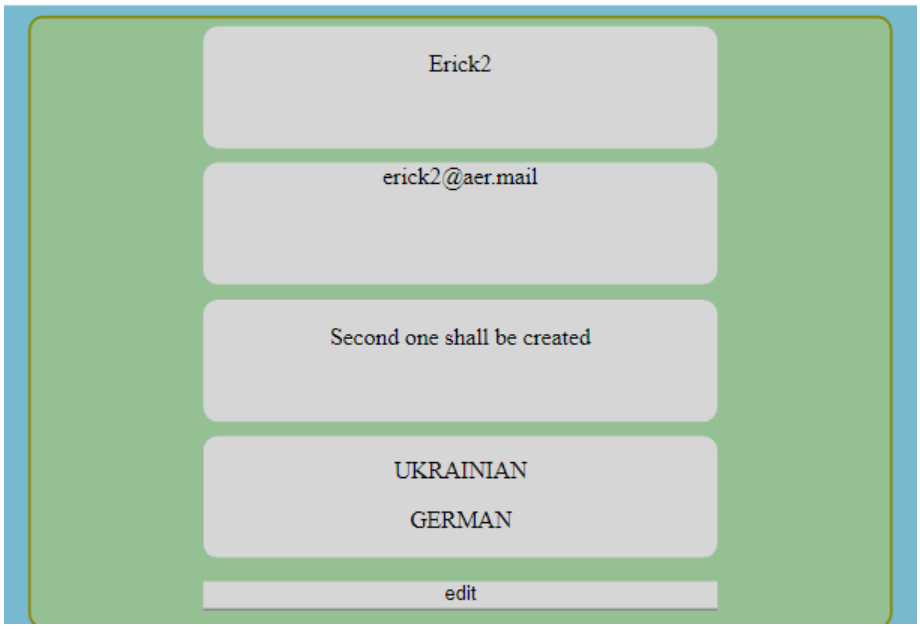



Рисунок 3.7 — Сторінка користувацького профілю

Також на даній сторінці можна редагувати поля імені та біографії користувача натиснувши кнопку “edit”, відповідні елементи буде змінено на поля для введення та буде додано кнопку “save changes” для збереження змін.


Рисунок 3.8 — Сторінка користувацького профілю після натиснення кнопки „edit“

### 3.2.2 Опис

Доступ до головної сторінки обмежено, перейти за даним посиланням можуть лише ті користувачі, котрі авторизувались. Зробити це дозволили

інструменти Spring Security. Окрім цього користувач, котрий перейшов на сторінку користувацького профілю, повинен пройти перевірку що акаунт, з якого здійснено запит, відповідає акаунту, котрий користувач бажає переглянути або редагувати.

При створенні нової групи в системі ініціалізується об'єкт класу GroupModel з потрібними полями, генерується серійний номер, котрий використовується як ідентифікатор групи. Окрім цього створюються колекції об'єктів GroupUser, MessageModel та TaskModel, в першу додається користувач з роллю "CREATOR" та відповідними правами, для неї також створюється об'єкт класу композитного ключа GroupUserKey. Усі ці об'єкти забезпечують необхідні для роботи зв'язки між таблицями, у цьому також допомагають анотації котрі, також, як і в класі користувача, використовуються для створення обмежень отриманих даних. Для даного класу створено інтерфейс GroupRepository в якому визначені усі необхідні для роботи з відповідною таблицею методи.

Для надсилання запитів для приєднання використовуються можливості бібліотек Sock.js та STOMP.js для встановлення зв'язку за протоколом вебсокет та використання підпротоколу STOMP. Після натиснення кнопки користувача буде приєднано до кінцевої точки, на яку буде надіслано запит, а користувача від'єднано.

Для виходу з акаунту використовується відповідна кнопка котра викликає функцію що здійснює POST запит за посиланням "/logout" котре робить користувацьку сесію недійсною та видаляє об'єкт авторизації. Після цього здійснюється перенаправлення користувача на початкову сторінку.

## 3.3 Огляд можливостей основної сторінки групи

### 3.3.1 Інструкція

Основна сторінка групи є головним інструментом організації процесу праці над проєктом. Вона дозволяє обмінюватись повідомленнями між користувачами та керувати їх правами в групі. Окрім цього за допомогою

кнопок, котрі знаходяться зліва від панелі повідомлень, можна перейти на сторінки опису групи, перегляду та додавання завдань, користувачів та перегляду запитів для приєднання.


Рисунок 3.9 — Основна сторінка групи

Проте дані кнопки відображаються лише за умови того, що користувач має відповідні права.

На панелі користувачів можна переглянути тих, хто вже приєднався до групи та їх права. Окрім цього, якщо користувач має право приймання запитів, він може змінювати користувацькі права натиснувши на кнопку “manage rights”.

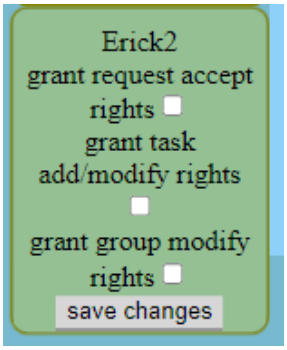



Рисунок 3.10 — Панель користувачів після натиснення кнопки “manage rights”

Після цього можна проставити галочки напроти відповідних значень та натиснути “save changes” для збереження змін. Слід зазначити, що кнопка “manage rights” відображається лише для тих, в кого є відповідні права, також права того користувача, котрий створив групу, змінити не можна.

Після натискання на кнопку “to group details” буде завантажено сторінку з інформацією про групу. Тут можна переглянути всю потрібну інформацію про групу.


Рисунок 3.11 — Сторінка з інформацією про групу

Щоб редагувати назву групи та її опис можна скористатись кнопкою “edit”, вона замінить відповідні елементи полями для введення та додасть кнопку “save changes” для збереження змін так само як і на сторінці користувацького профілю.

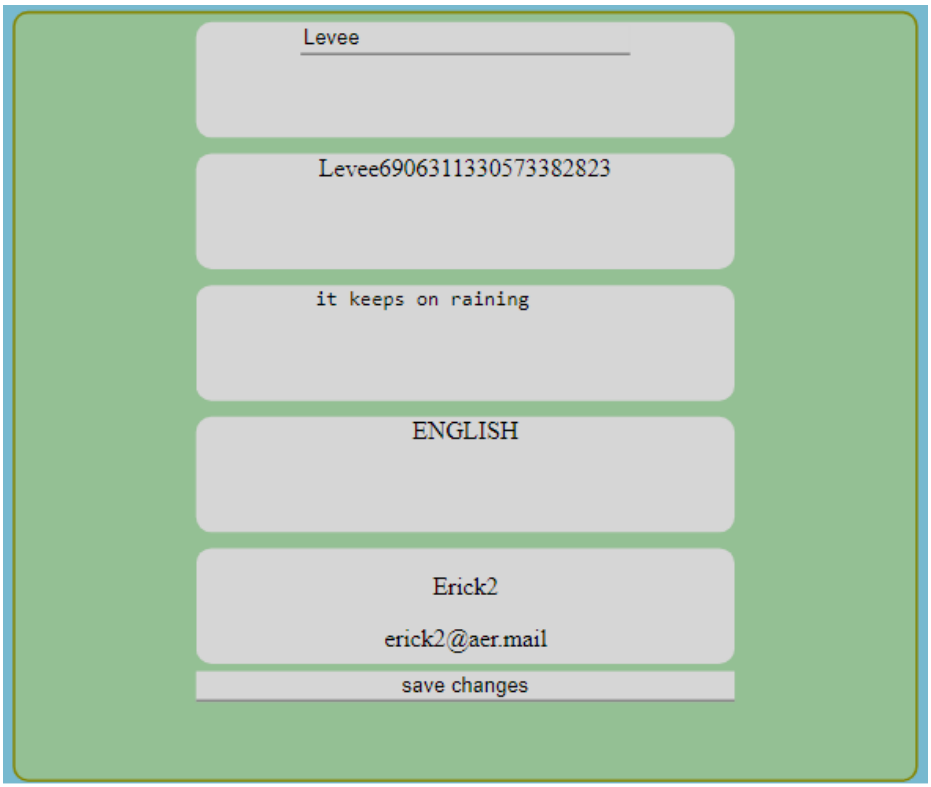



Рисунок 3.12 — Сторінка з інформацією про групу після натиснення кнопки “edit”

### 3.3.2 Опис

Після завантаження основної сторінки групи користувача буде під’єднано до відповідної кінцевої точки вебсокету групи для отримання сповіщень про нові надіслані повідомлення та надання можливості надсилання повідомлень. Панель повідомлень оновлюється кожного разу, коли на дану точку приходить нове повідомлення. Для відправлення повідомлень кожного разу створюється JSON об’єкт з користувацькими даними та надсилається до потрібної точки на сервері, там створюється новий об’єкт класу MessageModel з полями заповненими відповідними значеннями що були введені користувачем та його інформацією про адресу електронної пошти, даний клас має потрібні анотації для забезпечення правильної роботи системи. Цей об’єкт зберігається до таблиці за допомогою об’єкту створеного для інтерфейсу MessageRepository, котрий було отримано за допомогою анотації @Autowired, зі значенням поля group_id котре відповідає даній групі. Після цього JSON об’єкт пересилається на іншу

кінцеву точку, до якої приєднаний кожен користувач що на даний момент знаходиться на головній сторінці групи для повідомлення про появу нового запису що викликає оновлення інформації в панелі повідомлень за допомогою функцій бібліотеки jQuery.

Для зміни користувацьких прав використовуються можливості бібліотеки jQuery для надсилання асинхронних запитів на сервер та оновлення інформації. На сервері змінюються значення відповідних полів у створених з інформації що отримали з таблиці об’єктах класу GroupUser, після чого зміни зберігаються за допомогою методів визначених в інтерфейсі GroupRepository.

Посилання основної сторінки групи доступне лише для користувачів, що належать до неї, запити для зміни прав користувачів, переходу на сторінку з інформацією про групу та її редагування мають лише ті користувачі, у котрих є відповідні дозволи, для перевірки цього використовується клас PathAccessGuard методи якого викликаються Spring Security при спробі надсилання запитів за посиланням.

## 3.4 Огляд можливостей сторінки завдань

### 3.4.1 Інструкція

На сторінці усіх завдань можна переглянути інформацію про завдання усіх користувачів та додати нові. Для того щоб потрапити на неї користувач повинен мати дозвіл додавання та управління завданнями.

Завдання, що ще виконуються знаходяться у лівій частині панелі завдань. Ті завдання, що повинні бути виконані користувачем, що переглядає сторінку мають форму для їх завершення шляхом введення посилання на ресурс з результатами виконання та натискання кнопки “Complete task”. Поле посилання можна залишити порожнім.

Вже виконані завдання знаходяться у правій частині панелі завдань, в них також представлені введені при їх завершенні посилання.

Додати нові завдання можна за допомогою панелі додавання завдань, котра знаходиться у лівій частині екрану. Для створення завдання слід заповнити поля назви та опису завдання, обрати виконавця дату та час виконання та натиснути кнопку “register task”. Назва повинна містити два або більше символи, опис не може містити більше ніж 255 символів.

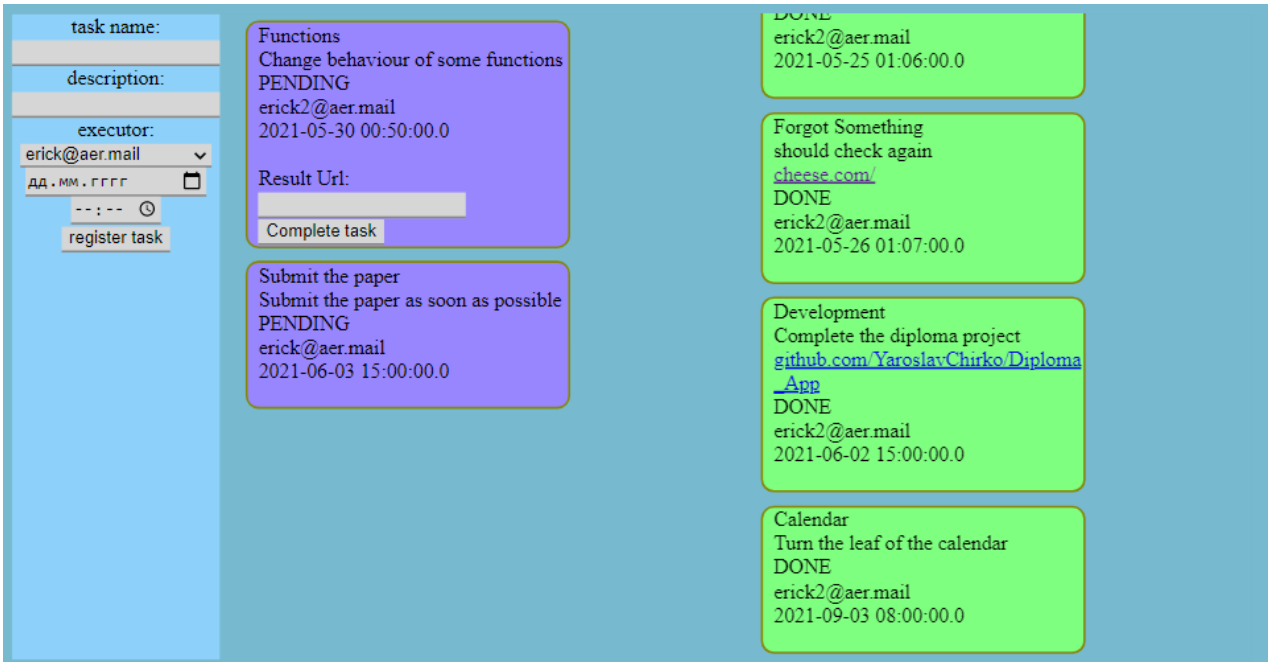



Рисунок 3.13 — Сторінка усіх завдань

Окрім даної сторінки також існує сторінка завдань користувача, на неї може потрапити лише той користувач, завдання котрого буде представлено. Вона відрізняється лише тим, що на ній будуть показані лише ті завдання, котрі даний користувач вже виконав або повинен виконати. Також, якщо у користувача немає прав додавання та управління завданнями, панель додавання завдань не відобразиться.

### 3.4.2 Опис

Для додавання завдань створюються об’єкти класу TaskModel з полями заповненими користувацькими даними. Ці поля анотовані для забезпечення правильності отриманих даних та створення об’єктно-реляційного представлення. Для даного класу створено інтерфейс TaskRepository в якому визначено необхідні методи для забезпечення потрібних для роботи системи функцій.

Після введення необхідних даних та натискання кнопки “register task” на панелі створення завдань дані відправляються на сервер де форматуються та зберігаються до таблиці.

Після натискання кнопки “Complete task” дані, що знаходяться в полі “Result Url” будуть відправлені на сервер, якщо адреса електронної скриньки користувача та виконавця співпадають, то дані будуть додані до завдання, а його стан змінено на “DONE”.

Для того щоб мати змогу використовувати функцію додавання завдань та переглядати усі завдання потрібно мати дозвіл додавання та управління завданнями. Для переходу на сторінку користувацьких завдань потрібно мати таку ж адресу електронної пошти що й вказана як частина шляху. Для перевірки цього Spring Security використовує методи класу PathAccessGuard.

## 3.5 Огляд можливостей сторінок додавання користувачів та запитів для приєднання

### 3.5.1 Інструкція

Сторінки додавання користувачів та запитів для приєднання використовуються для надання можливості додавання нових користувачів до групи.

На сторінці додавання користувачів можна скористатись одним з двох способів додавання користувачів: додати користувача за допомогою адреси електронної пошти або обрати з переліку користувачів.

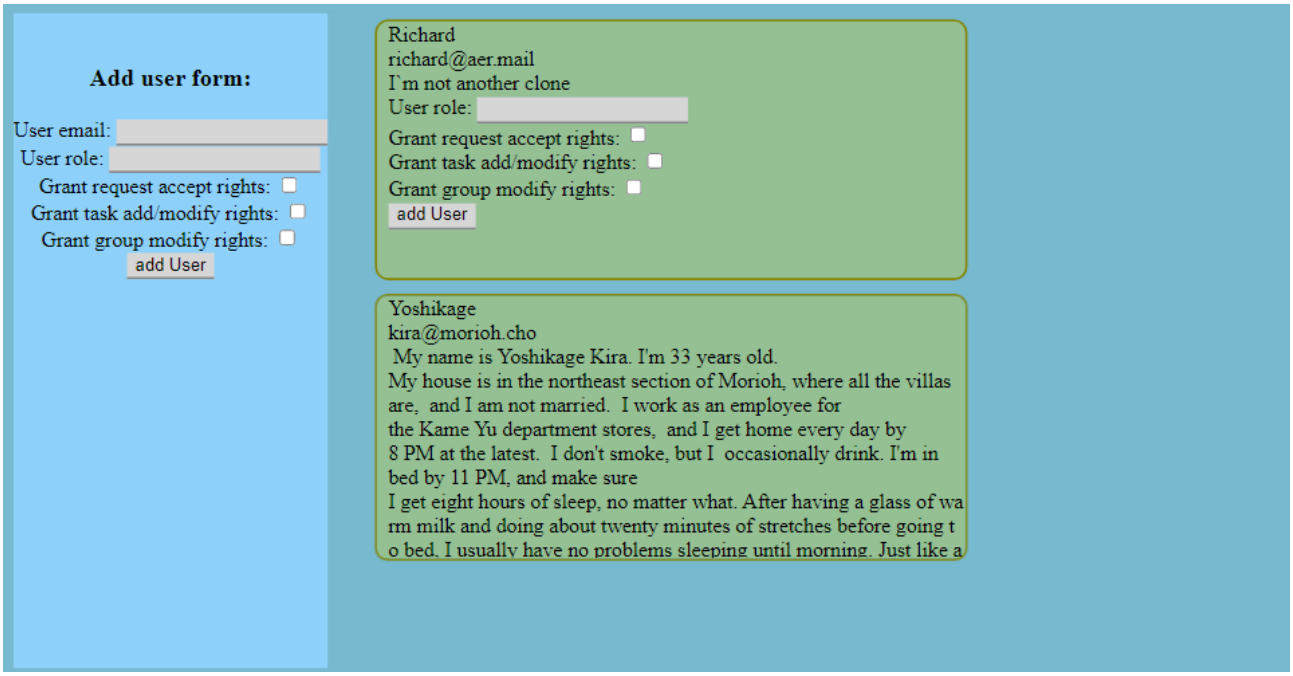



Рисунок 3.14 — Сторінка додавання користувачів

Панель додавання користувачів за допомогою адреси електронної пошти знаходиться зліва, за її допомогою можна додати зареєстрованого користувача використовуючи відому адресу електронної пошти. Для цього слід заповнити поля, обрати права котрі будуть надані користувачу з переліку на натиснути на кнопку “add User”.

У центральній та правій частині знаходиться панель переліку користувачів. Тут представлені усі користувачі, котрі при реєстрації обрали однією з мов ту, що використовується в даній групі та ще не були додані неї. Заповнивши поле ролі користувача та надавши потрібні права його буде додано до групи після того як буде натиснута кнопка “add User”.

На сторінці запитів для приєднання можна переглянути надіслані користувачами в групу запити для приєднання до неї. Для того, щоб додати користувача потрібно заповнити поля та натиснути кнопку “accept” а для того, щоб відмовити слід натиснути кнопку “deny”. Незалежно від обраної кнопки усі запити даного користувача буде видалено.


Рисунок 3.15 — Сторінка запитів для приєднання

### 3.5.2 Опис

Під час додавання нового користувача до групи система отримує об’єкт класу UserModel з відповідної таблиці та додає його до колекції користувачів об’єкту GroupModel, та заповнює необхідні значення ролі та дозволів. Після цього викликається функція збереження котра створює новий запис таблиці group_user.

Для своєчасного отримання запитів для приєднання користувача, котрий зайшов на відповідну сторінку, приєднують до кінцевої точки вебсокет з’єднання на яку надсилаються запити для цієї групи. Для забезпечення можливості підтвердження та відмови кнопкам надано відповідні значення а на сервері створено два різних методи для обробки.

Право доступу до цих сторінок та посилань, з якими вони працюють мають лише ті користувачі, котрі мають права приймання запитів, це забезпечується Spring Security та методами класу PathAccessGuard.

# ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було розглянуто створену систему, описано інструкцію з використання для забезпечення правильного розуміння того як з нею слід взаємодіяти та створено опис роботи компонентів для кращого розуміння того, як саме надається змога виконання певних дій.

В інструкції користувача наявні необхідні пояснення для забезпечення правильності використання системи.

Інформація, котру було отримано у ході виконання двох попередніх розділів, посприяла процесу розробки остаточного проєкту, надала можливість покращити початковий план розробки та, завдяки обраним технологіям, виконати роботу швидше та більш якісно.

# ВИСНОВОК

У ході даної роботи було розроблено Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Також було проведено порівняльний аналіз аналогічних систем з метою покращення розуміння основних потреб користувачів системи, що розробляється та обрано технології, що дозволили ефективно організувати процес розробки та зменшити кількість можливих проблем.

Усі компоненти системи працюють відповідно до очікувань, проте для того, щоб випустити її у якості комерційного продукту слід провести більш повне тестування та внести певні зміни в компоненти.

Завдяки використанню патерну MVC систему можна легко доповнювати а її компоненти підтримувати та вносити зміни до них. Окрім цього розділення обов'язків між компонентами дозволяє вносити зміни до однієї частини програми без потреби змінювати інші.

Також, для розширення даної програми можна реалізувати ті частини, що не вдалось додати у рамках дипломного проєкту зважаючи на обмеження в часі. Окрім цього можна оптимізувати роботу програми та змінити певні компоненти клієнтської частини для забезпечення кращого користувацького досвіду.

Усі поставлені на початку розробки проєкту цілі було досягнуто, а програма знаходиться у робочому стані (альфа версія) за потреби її можна розширити, доповнити та покращити.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. monday.com:Work the way that works for you [Електронний ресурс]. Режим доступу: https://monday.com/
2. Бітрікс24: сервіс автоматизації та оптимізації бізнес-процесів компанії [Електронний ресурс]. Режим доступу: https://www.bitrix24.ua/
3. Trello [Електронний ресурс]. Режим доступу: https://trello.com/
4. History of Java programming language [Електронний ресурс]. Режим доступу: https://www.freejavaguide.com/history.html
5. Differences between Java EE and Java SE [Електронний ресурс]. Режим доступу: https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html
6. History of Java: An Important Guide In Just 4 Points [Електронний ресурс]. Режим доступу: https://www.jigsawacademy.com/blogs/java/history-of-java/
7. Spring Framework Overview [Електронний ресурс]. Режим доступу: https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html
8. History of Spring Framework and Spring Boot [Електронний ресурс]. Режим доступу: https://www.quickprogrammingtips.com/spring-boot/history-of-spring-framework-and-spring-boot.html
9. spring-overview.png [Електронний ресурс]. Режим доступу: https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/images/spring-overview.png
10. Modules of the Spring Architecture – Dzone Java [Електронний ресурс]. Режим доступу: https://dzone.com/articles/4-modules-of-spring-architecture
11. Spring Security – Wikipedia [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Spring_Security

12. MVC3.png [Електронний ресурс]. Режим доступу: https://www.freecodecamp.org/news/content/images/2021/04/MVC3.png

13. The Evolution of MVC | Stephen Walther [Електронний ресурс]. Режим доступу: http://stephenwalther.com/archive/2008/08/24/the-evolution-of-mvc

14. ws02.png [Електронний ресурс]. Режим доступу: https://www.tutorialdocs.com/upload/2018/10/ws02.png

15. WebSockets – A Conceptual Deep Dive | Ably Realtime [Електронний ресурс]. Режим доступу: https://ably.com/topic/websockets

16. Thymeleaf – Wikipedia [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Thymeleaf

17. История развития HTML • Vertex Academy [Електронний ресурс]. Режим доступу: https://vertex-academy.com/tutorials/ru/html_history/

18. A brief history of CSS until 2016 [Електронний ресурс]. Режим доступу: https://www.w3.org/Style/CSS20/history.html

19. Document Object Model – Wikipedia [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Document_Object_Model

# ДОДАТОК 1

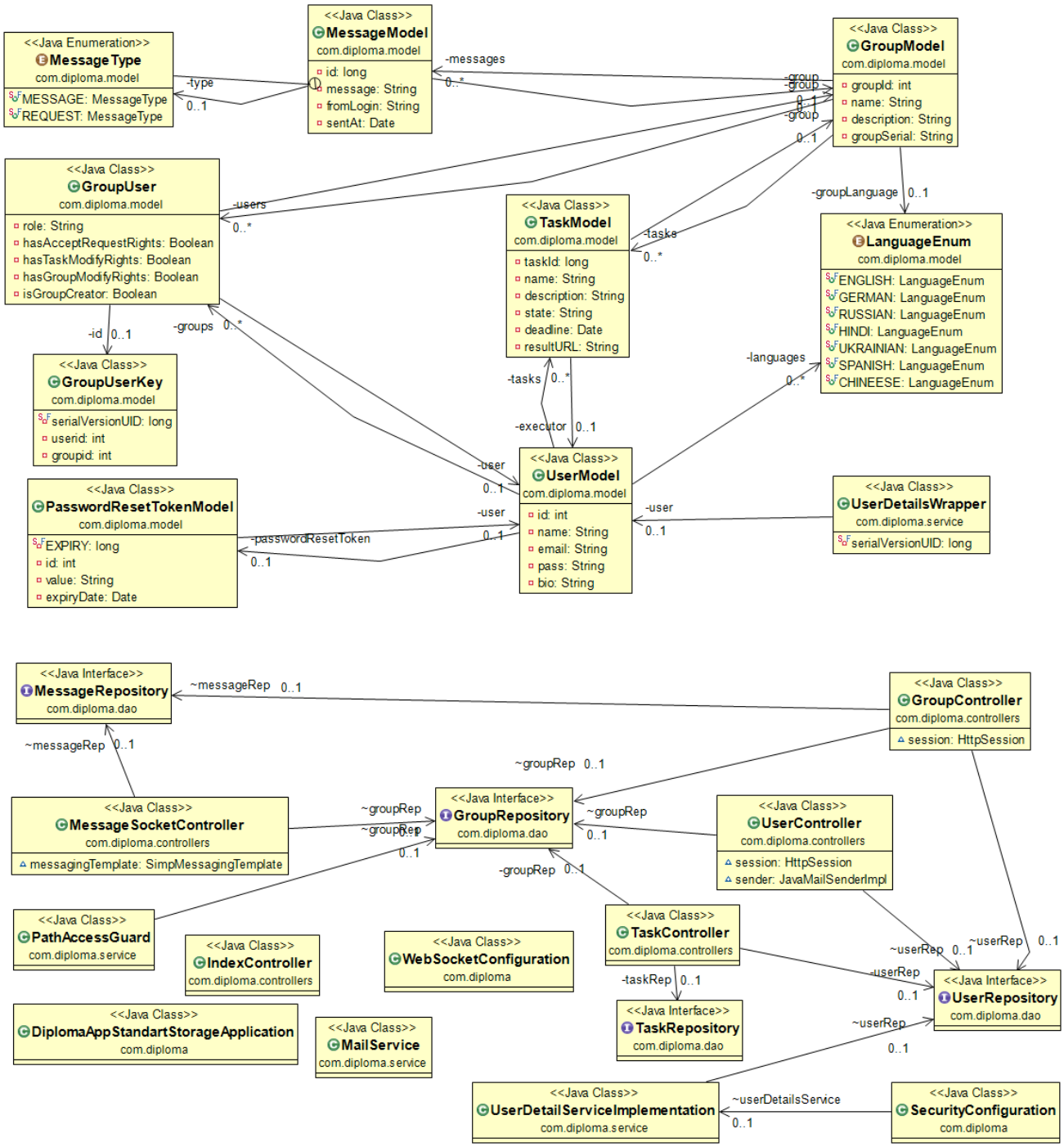
Клієнт-серверний додаток для створення груп та організації процесу праці над проектом

**Схема структурна - діаграма класів**

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2021 р.




| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.004 Д1 | |
| | | | | | Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Додаток 1. | Літ. / Маса / Масш. |
| Змн | Арк. | ПІБ | Підпис | Дата | | |
| Розробив | | Чирко Я.Ю. | | | | Аркуш / Аркушів |
| Перевірив | | Алещенко О.В. | | | КПІ ім. Ігоря Сікорського ФІОТ ІП-74 | 1 / 1 |
| Н/Контр | | Сімоненко В.П. | | | | |
| Зав.Каф | | Стіренко С.Г. | | | | |

# ДОДАТОК 2

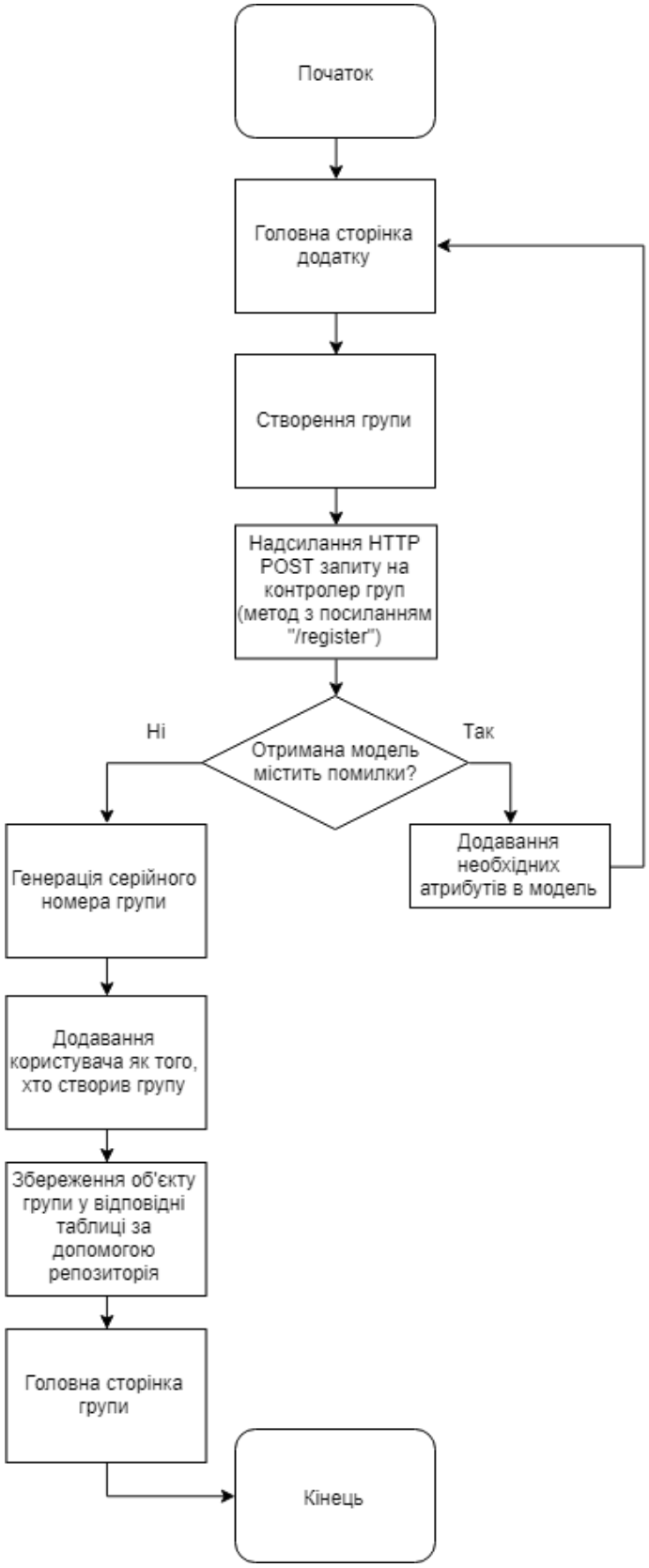
Клієнт-серверний додаток для створення груп та організації процесу праці над проектом

**Схема функціональна — блок-схема алгоритму створення групи**

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2021 р.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.005 Д2 | | |
| | | | | | Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Додаток 2. | Літ. | Маса | Масш. |
| Змн | Арк. | ПІБ | Підпис | Дата | | | | |
| Розробив | | Чирко Я.Ю. | | | | Аркуш | Аркушів | |
| Перевірив | | Алещенко О.В. | | | | 1 | 1 | |
| Н/Контр | | Сімоненко В.П. | | | КПІ ім. Ігоря Сікорського ФІОТ ІП-74 | | | |
| Зав.Каф | | Стіренко С.Г. | | | | | | |

# ДОДАТОК 3

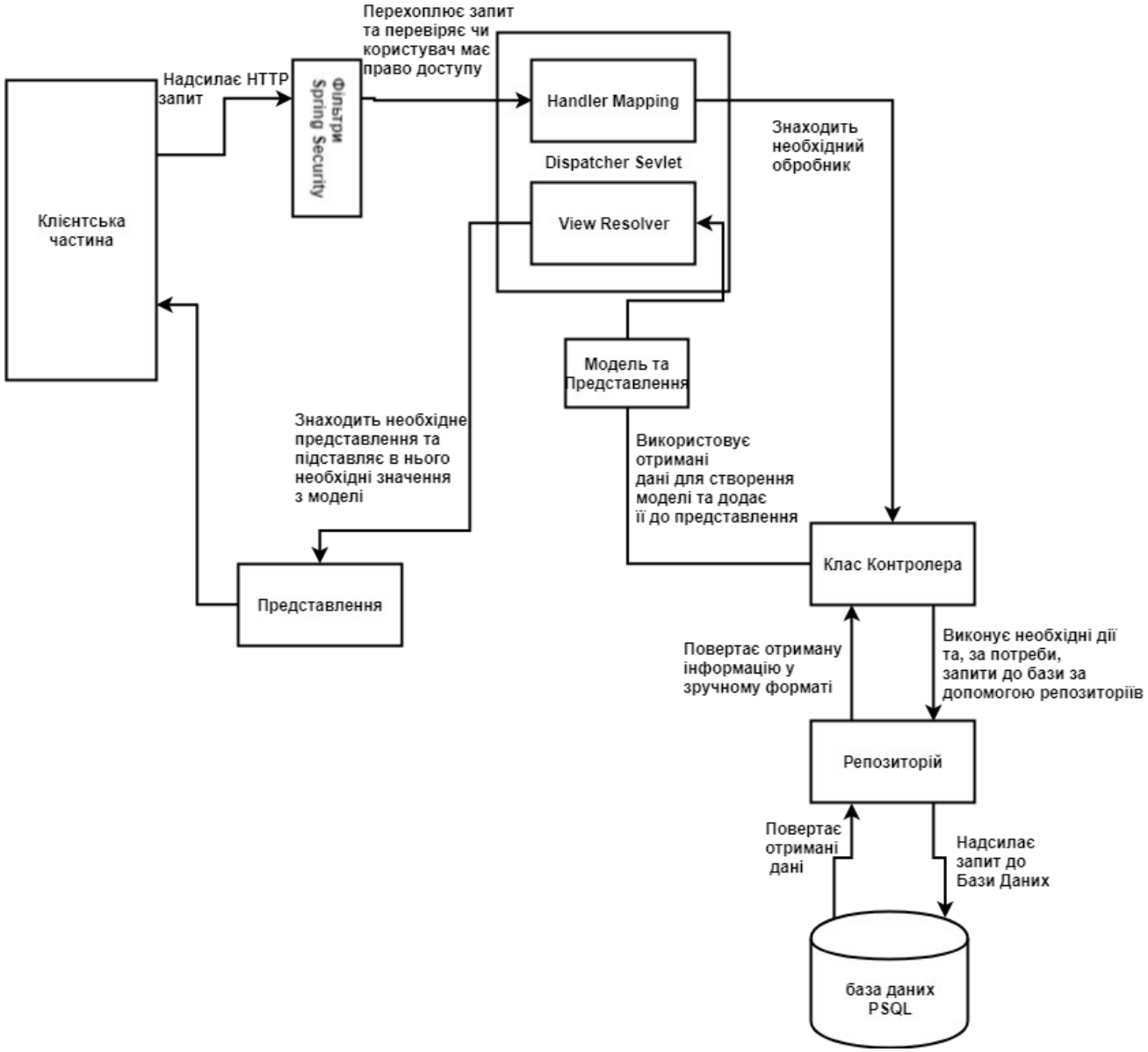
Клієнт-серверний додаток для створення груп та організації процесу праці над проектом

**Схема взаємодії**

ІАЛЦ.467100.006 Д3

Аркушів 1

Київ 2021 р.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *ІАЛЦ.467100.006 Д3* | | |
| | | | | | *Клієнт-серверний додаток для створення груп та організації процесу праці над проектом. Додаток 3.* | *Літ.* | *Маса* | *Масш.* |
| *Змн* | *Арк.* | *ПІБ* | *Підпис* | *Дата* | | | | |
| *Розробив* | | *Чирко Я.Ю.* | | | | *Аркуш* | *Аркушів* | |
| *Перевірив* | | *Алещенко О.В.* | | | | | | |
| | | | | | *КПІ ім. Ігоря Сікорського ФІОТ ІП-74* | *1* | *1* | |
| *Н/Контр* | | *Сімоненко В.П* | | | | | | |
| *Зав.Каф* | | *Стіренко С.Г.* | | | | | | |

# ДОДАТОК 4

Клієнт-серверний додаток для створення груп та організації процесу праці над проектом

***Ключові елементи коду програми***

ІАЛЦ.467100.007 Д4

Аркушів 11

Київ 2021 р.

# GroupController.java

```
package com.diploma.controllers;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Set;
import javax.servlet.http.HttpSession;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.SessionAttribute;
import org.springframework.web.servlet.ModelAndView;
import com.diploma.dao.GroupRepository;
import com.diploma.dao.MessageRepository;
import com.diploma.dao.UserRepository;
import com.diploma.model.GroupModel;
import com.diploma.model.GroupUser;
import com.diploma.model.MessageModel;
import com.diploma.model.MessageModel.MessageType;
import com.diploma.model.UserModel;
@RestController
@RequestMapping("/group")
public class GroupController {
	@Autowired
	GroupRepository groupRep;
	@Autowired
	HttpSession session;
	@Autowired
	UserRepository userRep;
	@Autowired
```

```
	MessageRepository messageRep;


	@PostMapping("/register")
	private ModelAndView registerGroup(@Valid @ModelAttribute("groupModel")
GroupModel group,
			BindingResult result,
			@SessionAttribute("email") String email) {
		if(result.hasErrors()) {
			ModelAndView mav = new ModelAndView("redirect:/user/fetch-all-
compatible-users"
					,result.getModel());
			return mav;
		}
		UserModel user = userRep.getUserByEmail(email);
		group.setGroupSerial(group.getName()+(new Random()).nextLong());
		group.setMessages(new ArrayList<MessageModel>());
		group.addUser(user, "CREATOR",true,true,true,true);
		groupRep.save(group);
		return new ModelAndView("redirect:/group/"+group.getGroupSerial()+"/main");
	}


	@GetMapping("/{groupSerial}/main")
	private ModelAndView getGroupMainView(@PathVariable("groupSerial") String serial,
			@SessionAttribute("email")String email) {
		List<GroupUser> users = groupRep.getGroupUsers(serial);
		List<MessageModel> messages =
messageRep.getMessagesByGroupGroupSerialAndTypeOrderBySentAtAsc(serial,
MessageType.MESSAGE);
		GroupUser user = groupRep.getUserFromGroup(serial, email);
		ModelAndView mav = new ModelAndView("group");
		mav.addObject("serial", serial);
		mav.addObject("messages",messages);
		mav.addObject("requestRigths",user.isHasAcceptRequestRights());
		mav.addObject("groupRights", user.getHasGroupModifyRights());
		mav.addObject("taskRights", user.isHasTaskModifyRights());
		return mav.addObject("users",users);


	}
```

```
	@PostMapping("/{groupSerial}/addUser")
	private void addUserToGroup(@ModelAttribute("email")String email,
			@ModelAttribute("role")String role,
			@RequestParam(name="request-rights",defaultValue ="false")Boolean
requestAcceptRights,
			@RequestParam(name="task-rights",defaultValue ="false")Boolean
taskModifyRights,
			@RequestParam(name="group-rights",defaultValue ="false")Boolean
groupModifyRights,
			@PathVariable("groupSerial")String serial) {
		if(groupRep.isUserInGroup(serial,
email)==0&&userRep.existsUserModelByEmail(email)) {
			GroupModel group = groupRep.getGroupByGroupSerial(serial);
			group.addUser(userRep.getUserByEmail(email), role.toUpperCase(),
				requestAcceptRights,taskModifyRights,groupModifyRights,false);
			groupRep.save(group);
		}

	}

	@GetMapping("/{groupSerial}/add-user-form")
	private ModelAndView showAddUserForm(@PathVariable("groupSerial") String serial)
{
		List<Integer> userIds
=userRep.getUserIdByLanguage(groupRep.getGroupByGroupSerial(serial).getGroupLanguage().
name());
		List<UserModel> users = new ArrayList<>();
			users = (List<UserModel>) userRep.findAllById(userIds);
		List<Integer> groupUserIds = groupRep.getGroupUserIds(serial);
		users.removeAll((List<UserModel>) userRep.findAllById(groupUserIds));
		ModelAndView mav = new ModelAndView("show-available-users");
		mav.addObject("users", users);
		mav.addObject("serial", serial);
		return mav;
	}

	@GetMapping("/{groupSerial}/requests")
	private ModelAndView getGroupRequestsView(@PathVariable("groupSerial") String
serial) {
```

```
            List<MessageModel> requests =
messageRep.getMessagesByGroupGroupSerialAndTypeOrderBySentAtAsc(serial,
MessageType.REQUEST);
            ModelAndView mav = new ModelAndView("requests");
            mav.addObject("serial", serial);
            mav.addObject("requests",requests);
            return mav;


      }


      @GetMapping(value = "/{groupSerial}/requests/process", params="accept")
      private ModelAndView acceptGroupRequestsView(@PathVariable("groupSerial") String
serial,
                  @ModelAttribute("user_login")String userLogin,
                  @ModelAttribute("role") String role,
                  @RequestParam(name="request-rights",defaultValue ="false")Boolean
requestAcceptRights,
                  @RequestParam(name="task-rights",defaultValue ="false")Boolean
taskModifyRights,
                  @RequestParam(name="group-rights",defaultValue ="false")Boolean
groupModifyRights) {
            if(groupRep.isUserInGroup(serial, userLogin)==0) {
                  GroupModel group = groupRep.getGroupByGroupSerial(serial);


                  group.addUser(userRep.getUserByEmail(userLogin), role,


      requestAcceptRights,taskModifyRights,groupModifyRights,false);
                  messageRep.deleteMessagesByGroupGroupSerialAndTypeAndFromLogin(serial,
MessageType.REQUEST, userLogin);
                  groupRep.save(group);
                  return new ModelAndView("redirect:");
            }else {
                  ModelAndView mav = new
ModelAndView("redirect:process").addObject("user_login",userLogin);
                  mav.addObject("deny","deny");
                  return mav;
            }


      }


      @GetMapping(value = "/{groupSerial}/requests/process", params="deny")
```

```
	private ModelAndView denyGroupRequestsView(@PathVariable("groupSerial") String serial,@ModelAttribute("user_login")String userLogin) {
		GroupModel group = groupRep.getGroupByGroupSerial(serial);

		messageRep.deleteMessagesByGroupGroupSerialAndTypeAndFromLogin(serial, MessageType.REQUEST, userLogin);

		groupRep.save(group);
		return new ModelAndView("redirect:/requests");
	}

	@GetMapping("/{groupSerial}/details")
	private ModelAndView getGroupDetailsForm(@PathVariable("groupSerial") String serial,
			@RequestParam(name="error_name",required = false)String nameError,
			@RequestParam(name="error_description", required=false)String descriptionError) {
		GroupModel group = groupRep.getGroupByGroupSerial(serial);
		ModelAndView mav = new ModelAndView("group_details");
		mav.addObject("group", group);
		mav.addObject("error_name" ,nameError);
		mav.addObject("error_description",descriptionError);
		return mav;

	}

	@PostMapping("/{groupSerial}/details/edit")
	private ModelAndView editGroupDetails(@PathVariable("groupSerial") String serial,
			@ModelAttribute("name") String name,
			@ModelAttribute("info") String description) {
		ModelAndView mav = new ModelAndView("redirect:/group/"+serial+"/details");
		GroupModel group = groupRep.getGroupByGroupSerial(serial);
		if(name!=null&&name.trim().length()>1) {
			group.setName(name);
		}else {
			mav.addObject("error_name","name is too short or consists of white spaces");
		}
```

```
            if(description.length()<1000) {
                  group.setDescription(description);
            }else {
                  mav.addObject("error_description","name is too short or consists of
white space");
            }

            groupRep.save(group);

            return mav;

      }

      @PostMapping("/{groupSerial}/manage-rights/{user_id}")
      private boolean manageUserRights(@PathVariable("groupSerial") String serial,
                  @PathVariable("user_id") int userId,
                  @ModelAttribute("requestRights") boolean requestRights,
                  @ModelAttribute("groupRights") boolean groupRights,
                  @ModelAttribute("taskRights") boolean taskRights) {
            GroupUser user = groupRep.getUserFromGroup(serial, userId);
            if(user==null||user.getIsGroupCreator()) {
                  return false;
            }
            GroupModel group = groupRep.getGroupByGroupSerial(serial);
            Set<GroupUser> users = group.getUsers();
            users.remove(user);
            user.setHasAcceptRequestRights(requestRights);
            user.setHasGroupModifyRights(groupRights);
            user.setHasTaskModifyRights(taskRights);
            users.add(user);
            groupRep.save(group);
            return true;

      }

}
```

# MessageSocketController.java

```
package com.diploma.controllers;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.DestinationVariable;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.web.bind.annotation.RestController;
import com.diploma.dao.GroupRepository;
import com.diploma.dao.MessageRepository;
import com.diploma.model.GroupModel;
import com.diploma.model.MessageModel;
import com.diploma.model.MessageModel.MessageType;

@RestController
public class MessageSocketController {
	@Autowired
	SimpMessagingTemplate messagingTemplate;

	@Autowired
	GroupRepository groupRep;
	@Autowired
	MessageRepository messageRep;

	@MessageMapping("/{groupSerial}")
	private void sendToGroup(@DestinationVariable("groupSerial")String serial,
MessageModel message) {

		GroupModel group = groupRep.getGroupByGroupSerialJoinMessages(serial);
		System.out.println(group);
		message.setType(MessageType.MESSAGE);
		group.addMessage(message);
		groupRep.save(group);

		MessageModel sendable = new
MessageModel(message.getMessage(),message.getFromLogin(),message.getSentAt());
		messagingTemplate.convertAndSend("/user/"+serial,sendable);
	}
```

```
	@MessageMapping("/{groupSerial}/requests")
	private void sendRequestToGroup(@DestinationVariable("groupSerial")String serial,
MessageModel message) {

		GroupModel group = groupRep.getGroupByGroupSerialJoinMessages(serial);
		message.setType(MessageType.REQUEST);
		message.setMessage("");
		group.addMessage(message);
		groupRep.save(group);

		MessageModel sendable = new
MessageModel(message.getMessage(),message.getFromLogin(),message.getSentAt());
		messagingTemplate.convertAndSend("/user/"+serial+"/requests",sendable);
	}
}
```

## TaskController.java

```
package com.diploma.controllers;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.SessionAttribute;
import org.springframework.web.servlet.ModelAndView;

import com.diploma.dao.GroupRepository;
import com.diploma.dao.TaskRepository;
import com.diploma.dao.UserRepository;
import com.diploma.model.GroupModel;
import com.diploma.model.GroupUser;
```

```
import com.diploma.model.TaskModel;
import com.diploma.model.UserModel;

@RestController
@RequestMapping("group/{groupSerial}/tasks")
public class TaskController {
	@Autowired
	private TaskRepository taskRep;

	@Autowired
	private GroupRepository groupRep;
	@Autowired
	private UserRepository userRep;

	@GetMapping("/{userEmail}")
	public ModelAndView showUserTasks(@PathVariable("userEmail") String email,
			@PathVariable("groupSerial") String groupSerial) {

		ModelAndView mav = new ModelAndView("task");
		GroupUser user =  groupRep.getUserFromGroup(groupSerial,email);

		List<TaskModel> userTasksP =
taskRep.getTasksByExecutorEmailAndGroupGroupSerialAndStateOrderByDeadlineAsc(email,
groupSerial, "PENDING");
		mav.addObject("pending_tasks", userTasksP);
		List<TaskModel> userTasksD =
taskRep.getTasksByExecutorEmailAndGroupGroupSerialAndStateOrderByDeadlineAsc(email,
groupSerial, "DONE");
		mav.addObject("done_tasks", userTasksD);
		if(user.isHasTaskModifyRights()) {
			mav.addObject("users",user);
			mav.addObject("taskRights",
groupRep.canUserAcceptRequests(groupSerial, email));
		}

		return mav;
	}

	@GetMapping("/all-tasks")
```

```
	public ModelAndView showAllTasks(@PathVariable("groupSerial") String groupSerial,
				@SessionAttribute("email")String email) {
			List<GroupUser> users = groupRep.getGroupUsers(groupSerial);
			ModelAndView mav = new ModelAndView("task");


			List<TaskModel> userTasksP =
taskRep.getTasksByGroupGroupSerialAndStateOrderByDeadlineAsc( groupSerial, "PENDING");
			mav.addObject("pending_tasks", userTasksP);
			List<TaskModel> userTasksD =
taskRep.getTasksByGroupGroupSerialAndStateOrderByDeadlineAsc( groupSerial, "DONE");
			mav.addObject("done_tasks", userTasksD);
			mav.addObject("taskRights", groupRep.canUserAcceptRequests(groupSerial,
email));


			mav.addObject("users", users);
			mav.addObject("serial", groupSerial);
			return mav;
	}


	@PostMapping("/add-task")
	public ModelAndView addTask(@ModelAttribute("task-name") String name,
				@ModelAttribute("task-description") String description,
				@ModelAttribute("executor-email") String email,
				@ModelAttribute("deadline_date") String deadlineDateString,
				@ModelAttribute("deadline_time") String deadlineTimeString,
				@PathVariable("groupSerial") String groupSerial) {
			GroupModel group = groupRep.getGroupByGroupSerial(groupSerial);
			UserModel user = userRep.getUserByEmail(email);
			Date deadline = new Date();


			try {
				deadline = new SimpleDateFormat("yyyy-MM-dd'
'hh:mm").parse(deadlineDateString+" "+deadlineTimeString);
			} catch (ParseException e) {
				e.printStackTrace();
			}
			TaskModel task = new TaskModel(name,description,group,user,
					"PENDING", deadline);
			group.addTask(task);
```

```
            groupRep.save(group);
            ModelAndView mav = new ModelAndView("redirect:all-tasks");
            return mav;
      }



      @PostMapping("/complete-task")
      public ModelAndView completeUserTask(@PathVariable("groupSerial") String
groupSerial,
                  @ModelAttribute("taskID")Long taskId,
                  @ModelAttribute("resultURL")String resultURL,
                  @ModelAttribute("executorEmail")String executorEmail,
                  @SessionAttribute("email")String email) {
            if(!email.equals(executorEmail)) {
                  System.out.println("wrong user");
                  return new ModelAndView("redirect:"+email);
            }
            TaskModel task = taskRep.findById(taskId).get();
            if(resultURL.startsWith("http://")) {
                  resultURL = resultURL.replaceFirst("http://", "");
            }else if(resultURL.startsWith("https://")) {
                  resultURL = resultURL.replaceFirst("https://", "");
            }
            task.setResultURL(resultURL);
            task.setState("DONE");
            taskRep.save(task);
            ModelAndView mav = new ModelAndView("redirect:"+email);
            return mav;
      }


}
```