АНОТАЦІЯ

Тишко Н.І. Розробка автоматизованого інтерактивного середовища соціалізації мовою програмування Java Script, фреймворк Meteor JS. – Рукопис.

Магістерська робота на здобуття освітнього ступеня магістр за спеціальністю 121 – Інженерія програмного забезпечення. – Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПд-2 // м.Тернопіль, 2019 // С. , рис. – , табл. – , додат. – , бібліогр. – .

Метою дипломної роботи магістра є реалізація концепції мінімалізму, яка полягає у створені продукту призначеного лише для зручного та ефективності обміну даними, без перевантаження додатковими функціями, з врахуванням потреб та дотриманням вимог з юридичної сторони питання.

Суть роботи полягає у створенні програмного продукту, реалізація якого дасть змогу створювати, формувати і розвивати концептуально нове сучасне інтерактивне середовище соціалізації від ефективності функціонування якого залежать успішність, освоєння надбань, ідей, досвіду, дотримання норм та стандартів.

Практичне застосування – розроблене програмне забезпечення призначене дасть змогу ефективно зберегти принцип організації системи, коли мета досягається інформаційним обміном між всіма елементами цієї взаємодіючої системи.

Ключові слова: СУЧАСНІ ТЕХНОЛОГІЇ, JAVASCRIPT, METEORJS, MONGODB, СПЕЦИФІКАЦІЯ, ТЕХНІЧНІ ВИМОГИ, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ, ПРОГРАМНИЙ ПРОДУКТ, ПРАКТИЧНЕ ЗАСТОСУВАННЯ.

АНОТАЦІЯ

Tyshko N.I. Developing of an interactive socialization environment with using Java Script programming language and framework Meteor JS. – Manuscript.

The master degree thesis for the qualification level of magistr in the specialty 121 — Software Engineering. – Ternopil Ivan Pul'ui National Technical University, Faculty of Computer Information Systems and Software Engineering, Software Engineering Department, group SPd-2 // Ternopil, 2019 //

Pages. – , pictures. – , tables. – , supp. – , bibl.ref. –

The purpose of the thesis is to implement the concept of minimalism, which is to create a product designed only for the convenience and efficiency of data exchange, without overloading by additional functions, taking into account the needs and legal requirements.

The essence of the thesis is to create a software product. The implementation will allow to create, form and develop a conceptually new modern interactive socialization environment. Although this is not new, today, depending on the specification of the need, it takes up considerable resources, since success depends on efficiency it.

Practical application - the developed software is intended for all potential users. The product is effectively preserve the principle of organization of the system, when the goal is achieved by the exchange of information between all elements of this interacting system, through which can effectively interact with another user or system.

Keywords: MODERN TECHNOLOGIES, JAVASCRIPT, METEORJS, MONGODB, SPECIFICATION, TECHNICAL REQUIREMENTS, INTERACTIVE ENVIRONMENT, SOFTWARE, PRACTICAL ACTIVITY.

ВСТУП

Сьогодення диктує свої умови і однією з ознак сучасності є інтерактивність між об’єктами різного характеру та ступеню, з особливостями обумовленими областю, напрямком та специфікою діяльності, зокрема в інформатиці, системах комунікацій, теорії інформації та інших. Не виключенням стало і програмування – один із напрямків, який найбільш динамічно розвивається та в якому збережено принцип організації системи, коли мета досягається інформаційним обміном між всіма її елементами, з допомогою яких може відбуватися ефективна взаємодія з іншим користувачем чи системою.

Актуальною задачею на сьогодні все ще залишається вирішення проблеми безпечного обміну даними, зокрема з однодумцями та за спільними інтересами не лише на побутовому рівні а, що найбільш суттєво, спрощення комунікації при реалізації якихось бізнес ідей чи з метою підвищення ефективності у будь-якому з напрямків діяльності. Подібні сервіси сьогодні не заслужено втратили свою колишню популярність і із-за цього мало розвиваються. Проте, ще не так давно вони активно використовувались в приватною метою. Сьогодні ж і взагалі зручніше та звичніше з цією метою використовувати соціальні мережі з вбудованими чатами.

Враховуючи тенденції та актуальність теми було прийняти рішення реалізувати концепцію мінімалізму, яка полягає у створені продукту призначеного лише для надійного, безпечного, зручного та ефективного обміну даними, без перевантаження додатковими функціями, з врахуванням потреб та дотриманням вимог з юридичної сторони питання. Крім цього заплановано створити концептуально нове інтерактивне середовище соціалізації, яке буде вигідно

відрізнятись від існуючого забезпечення подібної специфіки, і стане хорошим інструментом в групуванні за спільними інтересами чи напрямками діяльності, не залучаючи всім відомі популярні соціальні мережі. При чому особливу увагу варта було б зосередити на інтуїтивно зрозумілий та простий дизайн, оскільки лише такий підхід дасть змогу легко працювати та отримати бажаний результат.

Враховуючи все вище сказане та вимоги до розробки програмного забезпечення означено мету роботи, яка полягає в реалізації концепції мінімалізму при створенні продукту призначеного лише для зручного та ефективності обміну даними, без перевантаження додатковими функціями, з врахуванням потреб та дотриманням вимог з юридичної сторони питання. Для досягнення поставленої мети вирішено сформульоване наступне завдання:

- розробити та затвердити технічне завдання;
- проаналізувати технічне завдання, підібрати та проаналізувати бібліографічні матеріали необхідні для виконання дипломної роботи;
- розробити та спроектувати продукт;
- розробити план тестування програмного продукту;
- оформити допоміжну документацію;
- виконати обґрунтування економічної ефективності програмного продукту;
- проаналізувати роботу щодо питань з дотримання положень про охорону праці та безпеку в надзвичайних ситуаціях;
- зробити відповідні висновки за результатами виконаної роботи.

При цьому об’єктом дослідження визначено сучасне інтерактивне середовище соціалізації, як процес, який на сьогодні вже є звичним, проте, на вдосконалення якого, в залежності від

специфікації потреби, витрачаються чималі кошти, оскільки від ефективності його протікання залежать успішність, засвоєння надбань, норм, ідей, досвіду, підтримка стандартів, тощо. А предметом дослідження є власне сама розробка автоматизованого інтерактивного середовища соціалізації мовою програмування Java Script, фреймворк Meteor JS.

Оформлення магістерської роботи має бути оформлена відповідно до діючих стандартів: ДСТУ 2391-94. «Система технологічної документації. Терміни та визначення»; ДСТУ 3008-95. «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення», а також ЕСКД та іншим чинним стандартам.

Роботу апробовано в рамках VII науково-технічнічної конференції «Інформаційні моделі, системи та технології», м. Тернопіль, 2019 р. – Тернопіль: ТНТУ ім. І. Пулюя (м. Тернопіль, 11-12 грудня 2019 року), 2019.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програної системи

1.1.1 Аналіз предметної області

Першим етапом роботи [1] Актуальність вибору саме такого напряму дослідження як комутація зумовлена в першу чергу тим фактом, що на сьогодні це одна з найбільших в світі проблем в сенсі чіткості, надійності, безперебійності, адекватності та безпеки. На мою думку така тенденція зберігатиметься ще не один десяток років. Однак, з часом, у будь-якому напрямку діяльності, в тому числі і там де застосовуються сучасні інформаційні технології, для того щоб не втратити актуальності та бути в тренді, час від часу варта змінювати акценти. Саме така необхідність і спонукала мене обрати даний напрямок та за результатами аналізу предметної області чітко сформулювати задачі, які будуть виконані та зміст яких буде змістовно викладено в пояснювальній записці.

1.1.1.1 Актуальність вибору напряму дослідження

Темою даної дипломної роботи було вибрано соціальну мережу [2-4] (рис. 1.1) (де соціáльна (або суспільна) мере́жа – це така структура соціального напрамку, яку можуть створювати будь-які індивіди чи організації не залежно від форм власності). В загальному ця структура віддзеркалює різні зв'язки між індивідами чи організаціями через соціальні взаємовідносини різного роду, зокрема починаючи з випадкових знайомств і на завершення зв'язки родинного типу. Найраніше цей термін було оприлюднено у першій половині 19

століття (у 1954 році) Дж. А. Барнесом «Class and Committees in a Norwegian Island Parish, «Human Relations». Встановлено, що максимальний розмір соціальних мереж складає приблизно 150 індивідів, середнім вважається на 123 індивіди (Хілл та Данбар 2002).

1.1.1.2 Приклад схеми зв’язків у соціальних мережах

Оскільки відомо, що на сучасному етапі розвитку суспільства, та й людства в цілому, Інтернет представляє собою глобальну мережу, що здебільшого поєднує користувачів різних цільових аудиторів, до прикладу можна навести не лише приватних користувачів різного віку, а й організації різних форм власності, державні установи, приватні фірми [5-7]. Можливості його настільки широкі, що можна говорити про те, що оцінити потенціал цієї мережі досі до кінця не становить реальної можливості, і можна лише прогнозувати, який вплив вона має і матиме на процес соціалізації сучасної та людини майбутнього [8].

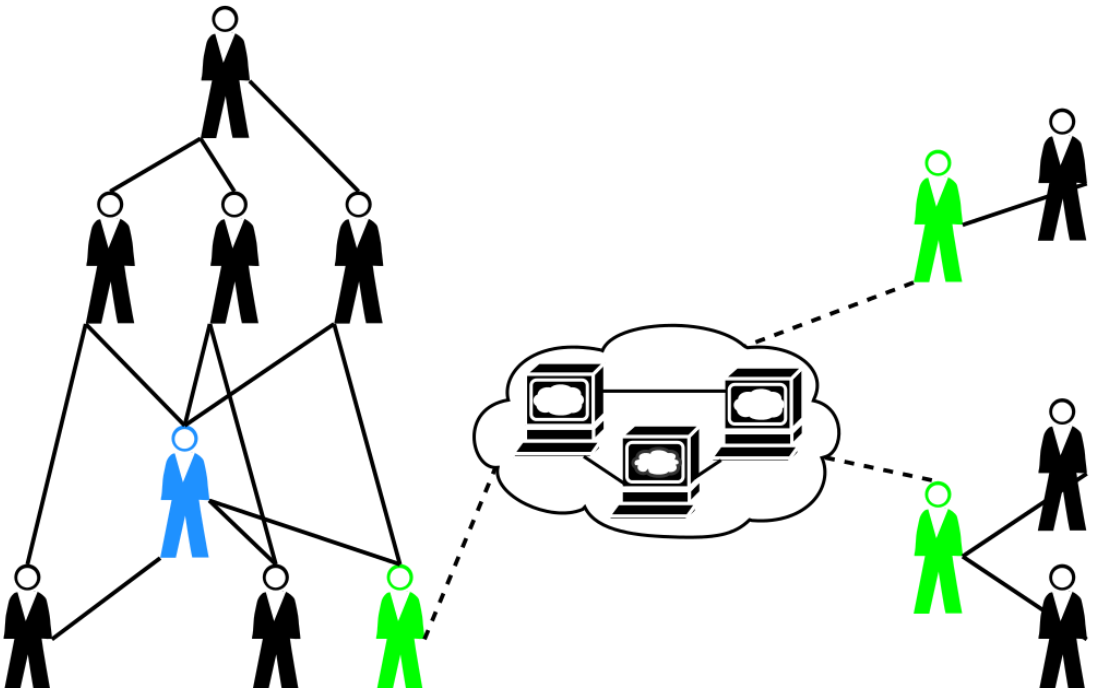

Рисунок 1.1 – Приклад схеми зв’язків у соціальних мережах

На рисунку представлено приклад схеми зв’язків у соціальних мережах на якому в соціальній мережі розташованій зліва індивід, виділений голубим кольором, має найбільшу кількість зв’язків у середині своєї соціальної мережі. У випадку, якщо розглядати випадок наближений до ідеального він повинен був би бути в якості лідера або, не нижче ніж керівником групи чи організації в цілому. Індивіди ж пофарбовані в зелений колір, зображені такими ,що мають зв’язки з іншими соціальними групами і можуть виступати в ролі передавачів інформації між мережами.

Дана предметна область вибрана через актуальність соціалізації зокрема для молоді, хоча люди похилого віку також дуже потребують цього, за допомогою глобальної мережі Інтернет. Таким чином стає зрозумілим той факт, що чим швидше людина соціалізується, тим успішнішим може складатися її розвиток та встановлення як індивіду у суспільстві, тощо.

Соціалізація в сучасному світі, має певні особливості (т.з. специфіку) в тому чи іншому суспільстві, однак є й певні спільні (подібні) характеристики.

Процес соціалізації протікає у постійній взаємодії при якій обов’язково відбувається врахування специфіки тої чи іншої вікової групи з надмірною кількістю різноманітних умов, які достатньо активно впливають на встановлення індивідуумів, та які називаються факторами соціалізації.

Глобальне поширення комп’ютерних ігор та Інтернету в останні роки спричинило значні зміни в інформаційній культурі. Можна говорити про появу нової субкультури так звані «інтернет-спільноти». Це може свідчити лише про модернізацію та трансформацію культури соціуму в цілому. Також можна говорити і про значну видозміну

змісту і певних соціальних ролей. Для прикладу, сьогодні вже нормою поведінки є щоденне використання ресурсів Інтернету не лише в професійній діяльності, а й у побуті, можливий діалог із засобами масової комунікації, вибірковість по відношенню до медійних каналів.

Очевидним є факт того, що людям старшого покоління потрібно прикласти значно більше зусиль, для того, щоби подолати шлях по дорозі до соціалізації, ніж людям молодшого віку та молоді. І зрозумілим цілком є той факт, що першим складніше приймати новіші чи суттєво модифікованіші соціальні ролі. Прикладна сторона цього такого твердження по відношенню до проблем, які виникають при освоєнні Інтернету – це технології, що дають можливість зробити цей процес більш зрозумілим і який пов'язаний з віковою диференціацією користувачів. Молодь, яка перебуває в початкових стадіях вторинної соціалізації, легше та й простіше приймає та засвоює нові соціальні ролі. Оскільки тенденцію є до подальшого бурхливого розвитку Інтернету як засобу масової комунікації, чому сприяє, в тому числі у великій мірі, і збільшення його аудиторії, призведе до того, що велика частина від кількості членів спільноти все ж прийме нові соціальні ролі, обумовлені новою віяннями нової інформаційної культури, в якій в основі є культурою мережі як активний учасник масової комунікації тощо. Варта також відзначити, що вже можна говорити про те, що на сьогодні домашня комп'ютеризація досягла чи найближчим часом досягне такого рівня важливості, що комп'ютер та Інтернет стали чи найближчим часом стаціть аґентами первинної соціалізації поряд з найближчим близьким оточенням кожної людини, не дивлячись на вікову групу.

1.1.1.3 Види соціальної активності

На сьогодні все більшої ваги набирає завдання, яке потребує вивчення всього різноманіття видів активності як молоді так і людей різних вікових категорій у віртуальному світі та наявність можливостей контролю суспільства за відхиленнями при соціалізації, які можуть бути викликані в тому числі й сучасними можливостями для віртуальної комунікації. Важливо не нехтувати можливостями, які нам дають сучасний стан розвитку науки та техніки і з раннього віку вивчати та навчати етиці поведінки у віртуальному просторі, яка ґрунтується на засадах системи суспільних цінностей.

Варта було б назвати види соціальної активності, які найчастіше зустрічаються, зокрема у саме віртуальному просторі, це:

1 – участь у чатах і форумах;

2 – участь в соціальних мережах;

3 – участь в інтерактивному голосуванні,

4 – створення власного сайту;

5 – створення власного блогу і т.п.

Перераховані види соціальної активності вимагають свого вивчення з точки зору мотивації людей з врахуванням віку та специфіки діяльності та створення умов для їх позитивної самореалізації у віртуальних контактах.

Інтернет простір як сфера самореалізації надає не тільки позитивний вплив на психоемоційний стан, а й покращує в певній мірі фізичне і моральне здоров’я. Для прикладу, це може бути емоційна та розумова активність, яка часто забирають багато сил, що згодом може призвести до емоційного і фізичного виснаження, відмови від реальності, апатії по відношенню до навколишнього світу, та й накопичення і врешті решт сплеску негативних емоцій, зайвої

дратівливості і агресії. Дані зміни можуть впливати і на засвоєння інформації та самореалізацію, тощо [6].

1.1.1.4 Характеристики відомих джерел інтернет-комунікації

Все більше набуває актуальності встановлення широти можливостей самовираження і самопрезентації, які власне і можуть надавати соціальні мережі. Відповідь на дане питання важливий не стільки як з'ясування значущості конкретних каналів самопрезентації, скільки як якісна оцінка соціальних мереж як медійного середовища, що надає ресурси для особистісного росту і саморозкриття.

До основних характеристик інтернет-комунікації варта було б віднести:

1) анонімність;
2) добровільність і бажаність контактів;
3) відсутність паралінгвістичних, невербальних компонентів спілкування [9];
4) утрудненість передачі і сприйняття афективного компонента спілкування [10];
5) своєрідність протікання процесів міжособистісного сприйняття в умовах відсутності невербальної інформації;
6) зняття жорстких соціальних конвенцій, кордонів культур.

На сьогоднішній день є зовсім мало соціальних мереж, які дотримуються концепції анонімності і мінімалізму. В основновму всі сучасні соціальні мережі схожі одна на одну і клонують гігантів таких як Facebook, Vkontakte, Instagram [11]. Ідея анонімності більш розвинена в так званих кімнатних чатах, де користувачі спілкуються в публічному чаті(кімнаті), з обмеженою кількістю людей. Метою ж

моєї дипломної роботи є об’єднання рис типової соціальної мережі і кімнатного чату, та досягнення максимальної простоти проте ефективності.

Проаналізувавши всі аспекти даної предметної області, було виявлено ряд проблем:

1) потреба пошуку людей зі спільними інтересами;
2) потреба зручного функціоналу для спілкування;
3) потреба самопрезентації;
4) постійний обмін інформацією між користувачами.

1.1.2 Постановка задачі

Після проведення аналізу предметної області, і виявлення основних проблем, було виокремлено завдання, які потрібно вирішити. Одними з найголовніших є:

1 - який тип бази даних вибрати.

2 - Як буде відбуватися з’єднання з базою даних.

3 - Чи потрібна реєстрація і авторизація.

4 - Чи необхідні додаткові дані для внесення в базу даних, чи достатньо згаданих у попередньому пункті.

5 - Які функції повинна виконувати дана система.

6 - Які процеси система буде виконувати автоматично, а які повинен реалізовувати користувач.

8 - Який буде інтерфейс системи, чи він буде зручним.

9 - Чи достатня кількість описаних акторів і чи пояснені їх основні варіанти використання.

10 - Чи достатньо повно вивчена предметна область, чи необхідно додатковий аналіз.

11 - Як супроводжувати і модернізувати дану систему.

Таким чином, було виокремлено та складено перелік завдань, які потрібно вирішити, а саме:

1) тип бази даних;
2) які дані зберігати;
3) процеси для автоматизації, покращення;
4) необхідні функції для користувача;
5) колекції бази даних;
6) чи є вся необхідна інформація;
7) як система допоможе користувачам.

Крім цього необхідно визначити, які дані нам будуть потрібні для бази даних. З цією метою потрібно узагальнити, тобто описати яка інформація нам буде необхідна в базі даних:

1) про користувачів;
2) кімнатні діалоги;
3) приватні діалоги;
4) про адміністраторів/модераторів;

Таким чином, перелік функцій системи матиме наступний вигляд:

1) пошук користувачів та кімнат;
2) заповнення профілю користувача інформацією;
3) надсилання, перегляд, редагування, видалення постів в кімнатах;
4) створення, перегляд, редагування, видалення приватних повідомлень;

Основні запити, які формує користувач, будуть здійснюватися через графічний інтерфейс, який буде приховувати всю обробку інформації в собі. Будуть запити на внесення і зміну даних, які також

буде виконувати система, після заповнення полів для внесення. Відповідні дані будуть надіслані на сервер.

Оскільки кінцевий користувач буде працювати тільки з графічним інтерфейсом, то необхідно забезпечити його інтуїтивність, зробити його максимально пристосованим до користувачів. Інтерфейс повинен бути простим у розумінні, не перевантажений деталями, повинен реагувати на всі запити користувача, сумісним з потребами і можливостями користувача.

1.1.3 Пошук актантів та варіантів використання

Соціальна мережа передбачає анонімне спілкування багатьох користувачів одночасно у декількох чат кімнатах. Також паралельно можна обмінюватись приватними повідомленням. Ще потрібні працівники, які будуть слідкувати за порядком в кімнатах.

Таким чином, в проекті буде тільки 2 актори: Користувач і Адміністартор.

Загалом система виконує наступні запити і функції:

1) реєстрація нового користувача;
2) авторизація користувача;
3) відображення кімнат;
4) відображення профілю користувача;
5) надсилання, редагування і видалення постів(публічних повідомлень);
6) надсилання, редагування і видалення приватних повідомлень.

Через графічний інтерфейс будуть відбуватися всі ці дії, здебільшого через натискання відповідних кнопок.

Для того, щоб повністю автоматизувати роботу даної інформаційної системи потрібно розглянути ряд функцій, які система буде виконувати автоматично для ще кращого результату, а саме:

7) автоматичне видалення і редагування даних з бази даних;
8) автоматична перевірка правильності введеного паролю чи емейлу.

1.1.4 Опис ключових варіантів використання

Отже, як встановлено в пункті 1.1.3, для повноцінної роботи системи, виявлено оптимальну кількість акторів: два основних актори, а саме: користувач та адміністратор. Звичайно, дана система потребує додаткових акторів, та на даному етапі розробки вони не будуть використовуватись.

Адміністратор – модерує чати кімнат, керує стандартними кімнатами, блокує користувачів-порушників.

Користувач – приймає участь у кімнатних чатах, надсилає приватні повідомлення іншим користувачам, створює власні кімнатні чати, шукає кімнати, які йому імпонують.

Таблиця 1.1 – Актори системи та їх варіанти використання

№	Основний актор	Найменування	Формулювання
1	2	3	4
1	Адміністратор	Створення/редагування/видалення стандартних кімнат	Стартовий контент кімнат складається з стандартних(тобто кімнат, назвою яких є найпопулярніші теми для розмов). Адміністратор займається наповненням стартового контенту

Продовження таблиці 1.1

1	2	3	4
2	Адміністратор	Модерування чату у кімнаті	Адміністратор слідкує, чи не порушуються користувачами правила ведення чату. Видаляє пости, які не відповідають правилам
3	Адміністратор	Блокування користувачів-порушників	При багатократному порушенні правил і ігноруванні попереджень адміністратор блокує користувача
4	Користувач	Приватні повідомлення	Надсилання приватних повідомлень іншим користувачам
5	Користувач	Реєстрація	Користувач реєструється у системі
6	Користувач, адміністратор	Авторизація	Користувач/адміністратор авторизується у системі
7	Користувач, адміністратор	Редагування профілю	Користувач/адміністратор редагує власний профіль
8	Користувач, адміністратор	Вхід у наявну кімнату	Здійснення входу у існуючий чат кімнати
9	Користувач, адміністратор	Створення кімнати	Створення нової кімнати
10	Користувач, адміністратор	Вихід з кімнати	При потребі користувач/ адміністратор покидає кімнату
11	Користувач, адміністратор	Надсилання/редагування/виалення посту у чаті кімнати	Користувач/адміністратор маніпулює постами(публічними повідомленнями) у чаті
12	Користувач, адміністратор	Отримати перелік всіх кімнат	Отримання переліку всіх наявних кімнат у мережі
13	Користувач, адміністратор	Пошук кімнат або користувачів	Користувач/адміністратор здійснює пошук кімнат або користувачів, в яких він зацікавлений

На рисунку 1.2 зображено основні варіанти використання системи.

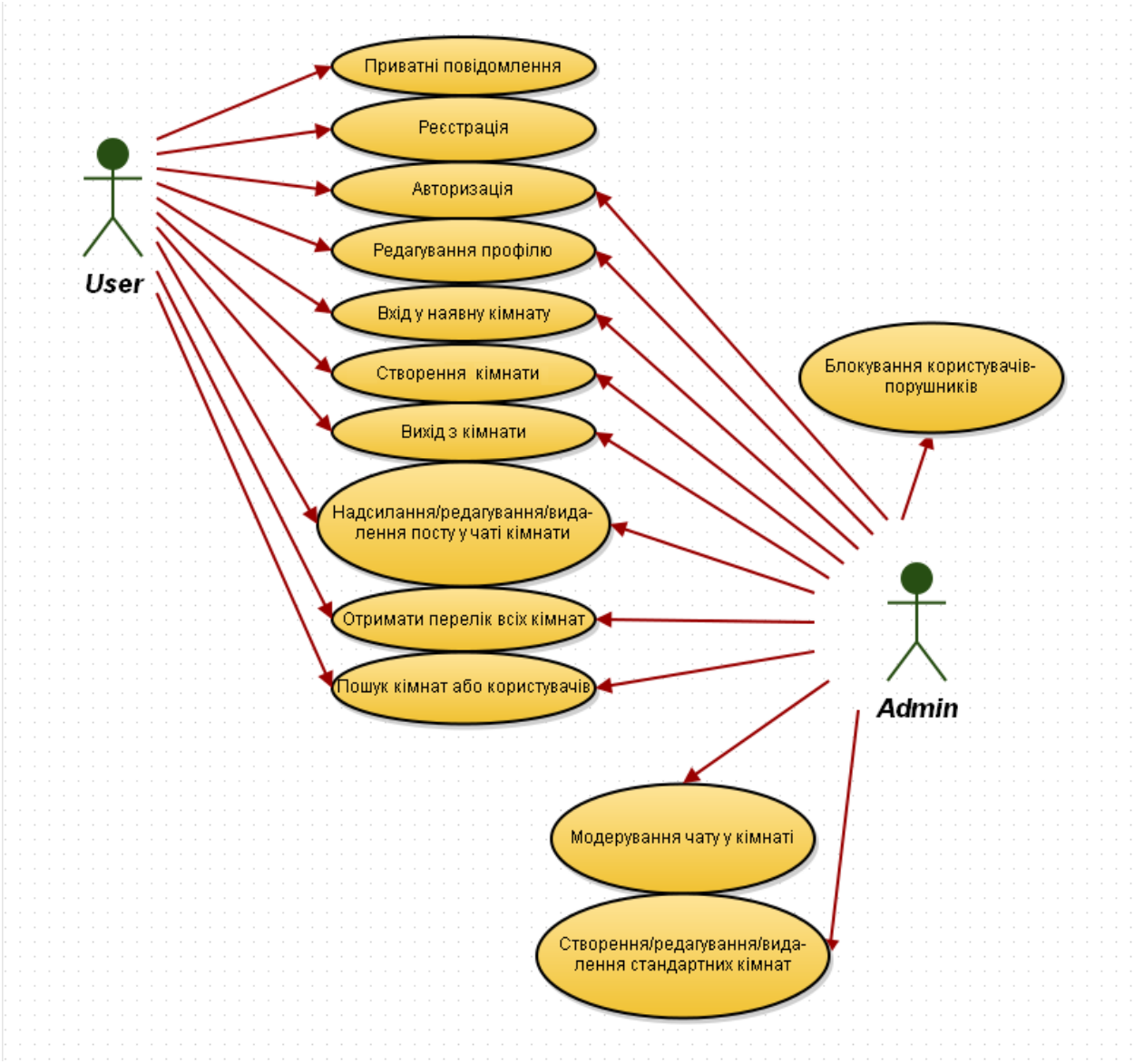

Рисунок 1.2 – Варіанти використання системи

1.2 Проектування програмної системи

1.2.1 Вибір процесу розробки

Структурою називають схему, за якою здійснюється зв’язок всіх сторінок і розділів ресурсу. Вибір конкретної структури може залежити від різних чинників, наприклад, від тематики, формату

сайту, загального обсягу опублікованого контенту, числа тематичних розділів, числа статичних сторінок і т.д.

Що стосується інших факторів, що впливають на тип структури сайту (обсяг контенту, загальне число розділів і т.д.), то тут все досить просто. Чим більше на сайті сторінок і розділів, чим більше статей опубліковано у кожному з розділів, тим складнішою буде структура. Проте в кожному конкретному випадку потрібно прагнути робити структуру максимально простою і зрозумілою, адже від цього безпосередньо залежать поведінкові фактори (чим простіше і зрозуміліше структура, тим користувачам простіше орієнтуватися на сайті, шукати необхідну і потрібну інформацію, переходити від однієї сторінки до іншої, від одного розділу до іншого). Крім того, проста структура сприяє більш якісній і швидкій індексації сайту, що безпосередньо впливає на просування.

Тому при проектуванні структури потрібно намагатися виключати (оптимізувати) всі зайві сторінки і розділи, скорочувати число рівнів їх вкладеності [12].

Найпростішою структурою вважається лінійна. Така структура властива різним електронним журналам і книгам. В даному випадку відсутні будь-які розділи. Є тільки окремі статичні сторінки, які пов'язані один з одним послідовно. Кожна з сторінок містить три основні посилання – на наступну сторінку, на попередню і на головну. Орієнтуватися у вмісті такого сайту досить легко, оскільки він володіє дуже простою системою навігації і відносно невеликим числом сторінок.

Дещо складнішою є лінійна структура з відгалуженнями – це трохи більше складна структура. Як правило, тут також немає розділів, а є тільки окремі статичні сторінки. Але посилання на всі ці сторінки (або на більшість з них) розміщені на головній. Завдяки

цьому система навігації тут дуже проста і інтуїтивно-зрозуміла, а доступ до всіх сторінок здійснюється лише за 1 або 2 кліка.

Найбільш поширеною можна вважати деревовидну структуру, оскільки багато сучасних сайтів і блогів володіють саме такою структурою. Її ключові особливості – наявність декількох рівнів вкладеності, число сторінок на кожному з яких зростає в арифметичній або геометричній прогресії, наявність статичних сторінок і розділів. Крім того, кожен з розділів може містити додаткові підрозділи. На головну сторінку посилається більшість внутрішніх сторінок. Деревовидну структуру найчастіше мають середні і великі за розміром сайти, що складаються як мінімум з декількох десятків сторінок (верхньої межі зазвичай немає). Зрозуміло, що використовувати лінійну структуру тут украй незручно, тому що всі наявні сторінки просто не помістяться на одному рівні.

Своєрідною модифікацією деревовидної структури є ґратчаста. У даному випадку між сторінками крім розвинених і стійких вертикальних присутні ще й горизонтальні зв'язки. Приміром, один на одного можуть посилатися сторінки, що відносяться до одного або суміжних розділів, а також сторінки з абсолютно різних розділів і частин сайту. Дана особливість сприяє збільшенню загального числа внутрішніх посилань. Покращиться або погіршиться навігація, залежить від грамотності розміщення посилань і зв'язки одних розділів з іншими (в деяких випадках структура виходить занадто заплутаною і незручною).

Для розробки даної програмної системи реалізовувалася деревовидна структура. Міститься головна сторінка з загальною інформацією. Для кожного з користувачів є певні розділи на які він зможе перейти. Кожен розділ складається з інших сторінок. Така структура є досить зручною, адже сторінки згруповані і поділені на

розділи. Дана система не буде містити велику кількість сторінок одного рівня. Більш детальніше про структуру даного сайту вестиметься далі.

Об’єктна модель документа (Document Object Model - DOM) є стандартом, що регламентує спосіб представлення вмісту документа у вигляді набору об'єктів. Під вмістом розуміється все, що може перебувати на веб-сторінці: малюнки, посилання, абзаци, текст і т.д.

На відміну від об’єктної моделі браузера (BOM), яка унікальна для кожного браузера, об'єктна модель документа є стандартом і повинна підтримуватися всіма браузерами. І хоча на практиці підтримка DOM реалізована не повною мірою, проте необхідно прагнути слідувати вимогам цього стандарту як виробникам браузерів, так і розробникам веб-сайтів.

Слід зауважити, що DOM може застосовуватися не тільки в веб-сторінках, але і до будь-яких інших документів. Зокрема, вона може використовуватися з будь-якими словниками XML, причому одним з таких словників є HTML, а точніше, XHTML.

DOM розбитий на три рівні. Перший рівень є першою версією стандарту і поки що єдиною закінченою. Він складається з двох розділів: перший є ядром і визначає принципи маніпуляції зі структурою документа (генерація і навігація), а другий присвячений поданням в DOM елементів HTML, що визначаються однойменними тегами.

Другий і третій рівні описують модель подій, доповнюють таблиці стилів, проходи по структурі.

В DOM документ представляється у вигляді дерева (див. рисунок 1.3), що є однією з найбільш уживаних структур у програмуванні. Це забезпечує уніфікований спосіб навігації по документу.

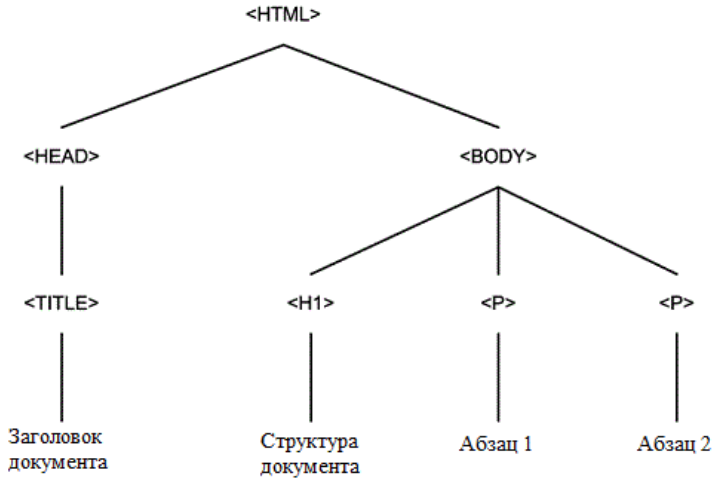

Рисунок 1.3 – HTML-документ у вигляді деревовидної структури

Візуальне уявлення DOM деколи відрізняється від HTML-коду. Деколи у html-коді трапляються помилки. В такому випадку браузер може сам виправити їх. Це може бути пропущений тег, який додається в DOM, який можна потім знайти за допомогою JavaScript, навіть якщо його немає в html-коді (наприклад, тег tbody в таблиці) [13].

1.2.2 Побудова схеми бази даних

Перша колекція – це користувачі системи(Users). Вона містить id користувача, його нікнейм, е-мейл, пароль, роль користувача(user або administrator), стать, день народження, фото профілю, улюблену музику(fmusic – favorite music), улюблені фільми(ffilms – favorite films), улюблені ігри(fgames – favorite games), улюблені серіали(fserials – favorite serials).

Лістинг 1.1 – Колекція користувачів системи

```
Users = new Mongo.Collection('users');
{
```

```
"_id" : "CaNpZv5N5TaXRbdn",
"nickname" : "Рейнор",
"email" : "JimRaynor@gmail.com",
"role" : ["user"]
"password" : "ZergRush",
"sex" : "чоловіча",
"birthday" : "24.04.1991",
"fmusic" : "Killers",
"ffilms" : "Гладіатор",
"fgames" : "StarCraft",
"fserials" : "Гра престолів",
}
```

Наступна колекція – це колекція кімнат. Містить id кімнати, назву кімнати, яка відображає тематику спілкування і кількість користувачів в ній(CoU – count of users). Також кімната містить кількість постів(CoP – counts of posts), та айді залишених постів.

Лістинг 1.2 – Колекція кімнат

```
Rooms = new Mongo.Collection('rooms');

{
"_id" : "eyfddZ2f3qETSTYjR",
"name" : "Дота 2",
"CoU" : "2",
"CoP" : "1",
"Posts_ids" : "fq4TzbALm6KTgPFE"
}
```

Наступна колекція – це колекція постів(публічних повідомлень). В ній пристуні айді посту, час, коли пост був надісланий, текст самого посту, кількість лайків і звісно айді і нікнейм автора посту.

Лістинг 1.3 – Колекція постів

```
Posts = new Mongo.Collection('posts');

{
"_id" : "fq4TzbALm6KTgPFE",
"time" : ISODate("2015-05-16T21:01:31Z"),
"text" : "Привіт народ" ,
"user_id" : "CaNpZv5N5TaXRbdn",
"user_nickname" : "Рейнор"
```

```
"CoL" : "3"
}
```

Наступна колекція – це колекція приватних повідмолень. Вона містить id повідомлення, id користувачів(адресант і адресат), які спілкуються між собою, їх нікнейми, текст самого повідомлення і час, коли воно було надіслане.

Лістинг 1.4 – Колекція приватних повідомлень

```
Messages = new Mongo.Collection('messages');

{
"_id" : "QRfdfsSN5TaXRbdn",
"text" : "Тук Тук",
"sender_id" : "CaNpZv5N5TaXRbdn",
"receiver_id" : "Pqet45sarqeTvB6n",
"sender_nickname" : "Рейнор",
" receiver _nickname" : "Кеpіган",
}
```

Отже, реалізовано 4 колекції для роботи системи (колекція користувачів, колекція кімнат, колекція постів, колекція приватних повідомлень).

1.2.3 Побудова діаграми шаблонів

MeteorJS слабо структурований фреймворк. Це означає, що тут необов’язково дотримуватися якоїсь структури чи використовувати класи для з’єднання частин коди між собою чи наслідування. Використання класів можливе, але воно не першочергове для Meteor, воно дещо утруднює написання коду програми. Роль класів у такому фреймворку замінюють шаблони. З їх допомогою можна оголошувати змінні, робити наслідування шаблонів, описувати і викликати методи.

JavaScript дозволяє створення функцій шаблонів, оголошення змінних можливе завдяки хелперам шаблонів, також шаблони можуть бути вкладені один в інший [14].

Було створено декілька шаблонів з html-кодом, потім до кожного з них додано певну логіку, використовуючи JavaScript.

Загальна діаграма шаблонів наведена на рисунку 1.4.

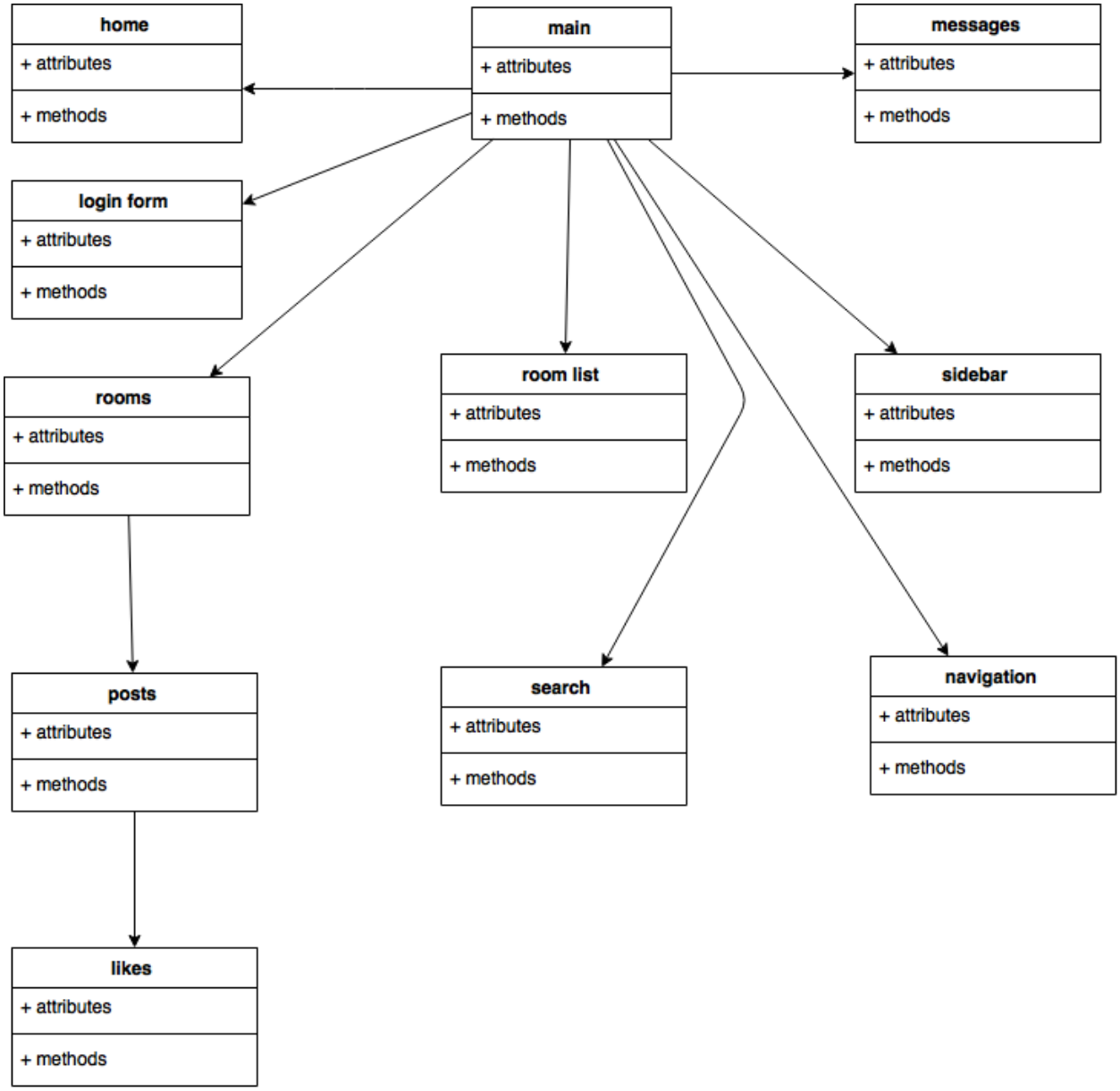

Рисунок 1.4 – Діаграма шаблонів

Шаблони створюються з використанням тегу <template name=“”></template>.

Розглянемо основні шаблони нашого сайту.

Шаблон main містить основний дизайн системи, посилання на інші шаблони, є початковою сторінкою авторизованого користувача.

Шаблон login form призначений для авторизації або реєстрації користувачів/адміністраторів.

Шаблон home призначений для відображення профілю користувачів.

Шаблони navigation і sidebar – функціональні шаблони, які забезпечують роботу бокового і верхнього меню.

Шаблон room list відображає список наявних кімнат.

Шаблон search використовується для пошуку користувачів та кімнат.

Шаблон rooms відображає саму кімнату, містить логіку функцій кімнати. Цей шаблон відповідно включає в себе шаблон Posts, за допомогою якого в чат можна надіслати публічне повідомлення. Posts містить в собі шаблон Likes, який дозволяє натиснути кнопку «мені подобається» під публічним повідомленням. Часткова демонстрація логіки шаблонів відображається в наступних лістинингах:

Лістинг 1.5 – функція додавання постів

```
Template.home.events({
'keyup .posttext':function(evt,tmpl){
if(evt.which === 13){
   var posttext = tmpl.find('.posttext').value;
   var options = {text:posttext,parent:null};
   Meteor.call('addPost',options);
   $('.posttext').val("").select().focus();
}
}
})
```

Лістинг 1.6 – елемент розмітки шаблону navigation

```
<template name="navigation">
    <div class="navbar navbar-static-top navbar-
inverse">
        <div class="navbar-inner">
```

```
            <a class="brand" href="/">Multiroom</a>
            <ul class="nav">
                <li><a href="/">Моя сторінка</a></li>
            </ul>
        </div>
    </div>
</template>
```

Лістинг коду 1.7 – шаблон sidebar

```
<template name="sidebar">
    <ul class="nav nav-list">
      <li class="nav-header">Multiroom </li>
      <li><a href="/home">Моя сторінка </a></li>
      <li><a href="/messages">Мої повідомлення</a></li>
          <li><a href="/rooms">Кімнати </a></li>
          <li><a href="/room_list">Список кімнат</a></li>
          <li><a href="/search ">Пошук </a></li>
    </ul>
</template>
```

1.2.4 Моделювання архітектури системи

На відміну від інших фреймворків чи мов програмування MeteorJS використовує дещо іншу архітектуру побудови своїх застосунків. Відмінність полягає в тому, що meteor-застосунки використовують MVVM архітектуру замість стандартної MVC.

MVVM використовується для розділення моделі і її представлення, що необхідно для зміни їх окремо один від одного при роботі окремо з логікою програми та інтерфейсом.

MVVM зручно використовувати замість класичного MVC і йому подібних в тих випадках, коли в платформі, на якій ведеться розробка, присутнє «зв'язування даних» [15].

Шаблон MVVM ділиться на три частини (див. рисунок 1.5).

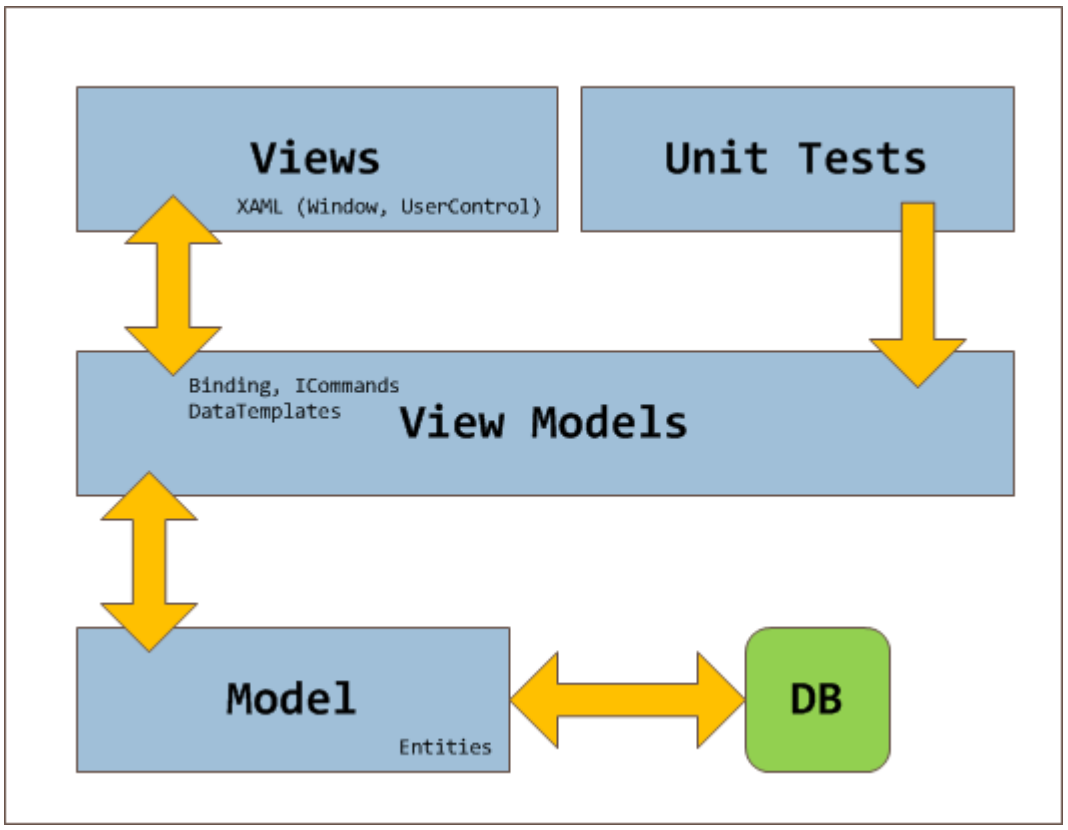

Рисунок 1.5 – MVVM патерн

Модель, так само, як у класичній MVC, являє собою фундаментальні дані, необхідні для роботи програми.

Представлення – це графічний інтерфейс, тобто вікна, кнопки. У випадку, якщо в Моделі представлення змінилася якась властивість, то Представлення, в свою чергу, запрошує оновлене значення властивості. У випадку, якщо користувач впливає на який-небудь елемент інтерфейсу, Представлення викликає відповідну команду, надану моделлю уявлення.

Модель представлення є, з одного боку, абстракцією Представлення, а з іншого, надає обгортку даних з Моделі, які підлягають скріпленню. Тобто, вона містить Модель, яка перетворена до Представлення, а також містить у собі команди, якими може користуватися Представлення, щоб впливати на Модель.

Застосовується зазвичай у вигляді шаблонів, які містять методи для забезпечення роботи системи.

Наведена на рисунку 1.6 архітектура MVVM притаманна нашій програмній системі.

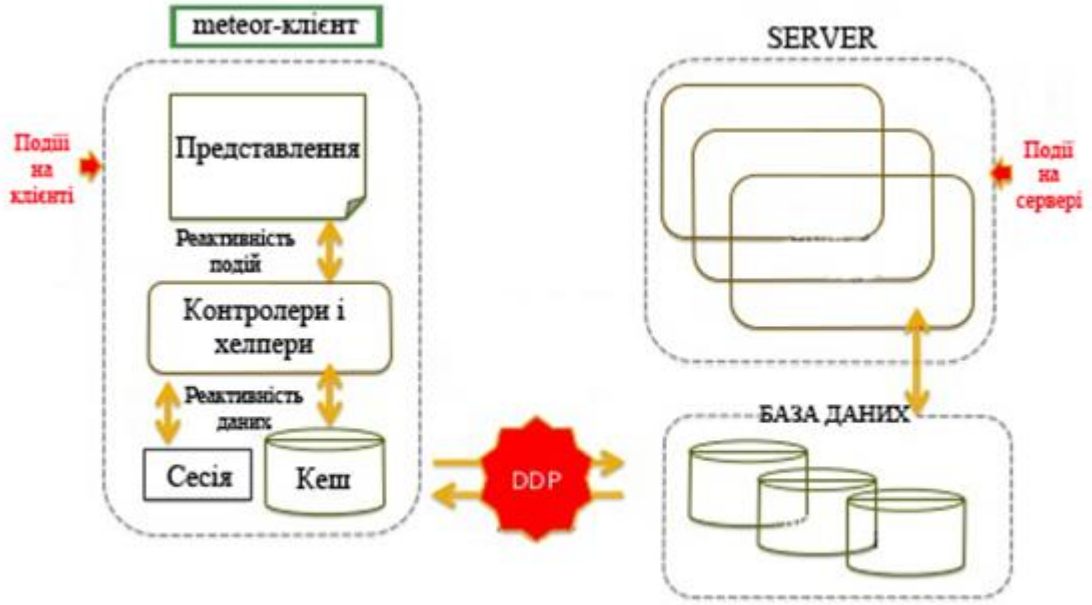

Рисунок 1.6 – Робота протоколу DDP у meteor-застосунку

Клієнт-серверна архітектура використовує Distributed Data Protocol (DDP) для управлінням зв’язком реального часу, який підтримується сучасними браузерами за допомогою WebSockets.

Протокол DDP призначений для роботи з колекціями документів JavaScript Serialized Object Notation (JSON), що дозволяє легко створювати, оновлювати, видаляти, запитувати і, звичайно, переглядати документи JSON. Так як DDP – це протокол з відкритим вихідним кодом, він повинен працювати з будь-яким клієнтом або сховищем даних. Це клієнт-серверний протокол для запитів і оновлення бази даних на стороні сервера і для актуалізації застосунків на стороні клієнта. Він визначає залежність типу «один до багатьох»

між об'єктами таким чином, що при зміні стану одного об'єкта, інші об'єкти, залежні від нього, сповіщаються про це і автоматично оновлюються.

Сам протокол був створений безпосередньо для використання в Meteor. Його силами зазвичай передаються дані, а не готовий HTML-код.

Фактично, Meteor забезпечує дві бази даних MongoDB: буферну базу даних з боку клієнта і базу даних MongoDB з боку сервера. Коли користувач вносить зміни в дані, код JavaScript, що виконується в браузері, оновлює відповідний запис у локальній базі даних MongoDB, а потім робить запит DDP до сервера. Код обробляється негайно, неначе операція виконана успішно, тому що відповіді сервера чекати не потрібно. Тим часом дані на сервері оновлюються у фоновому режимі. Якщо операція на сервері не вдалася, або повертається несподіваний результат, то код JavaScript на стороні клієнта негайно коригує дані відповідно до останньої відповіді сервера.

Це коректування називається компенсацією затримки і забезпечує додаткове відчуття швидкодії у користувача.

Навіть система шаблонів Meteor явно націлена на спрощення зв'язку в режимі реального часу. На більшості Web-платформ в код можна легко впроваджувати мову гіпертекстової розмітки (HTML) або розмітки, еквівалентній HTML, такій як HTML Abstraction Markup Language (Haml). Це дозволяє легко вставляти в сторінки, що відправляються користувачеві, динамічні значення з бази даних. Після цього система повинна стежити за змінами в даних і оновлювати розмітку. Однак система шаблонів в Meteor реєструє, до яких саме даних зверталися через шаблон, і автоматично виконує зворотні

виклики, змінюючи цей HTML-код при зміні відповідних даних, що робить шаблони в режимі реального часу простими і швидкими [16].

1.2.4.1 Структура програмної системи

Особливість meteor застосунків – це те, що код на стороні клієнта і на стороні сервера написаний на JavaScript.

Структура проекту в Meteor повинна містити дві папки – client з всіма файлами, які виконуються на клієнті та server з файлами, які виконується на сервері. Файли в папці client виконуються браузером. Файли в папці server – сервером. Загальна структура meteor за стосунків наведена в лістингу 1.8

Лістинг 1.8 – Структура проекту

```
if (Meteor.isServer) {
// виконується серверний код
}
if (Meteor.isClient) {
// виконується клієнтський код
}
```

При обробці html файлів, теги template конвертуються в JavaScript функції доступні в просторі імен Template.

Це дійсно зручний спосіб відправки HTML шаблонів для клієнта.

Папка Public служить для зберігання зображень.

Кращий спосіб написати застосунок таким чином, щоб не було відчутно в якому порядку завантажуються файли, використовуючи Meteor.startup() або шляхом переміщення коду в Smart Packages, завантаження і порядок яких можна точно контролювати [17]. Однак іноді не вдається уникнути порядку завантаження JS і css файлів, вони завантажаться за такими правилами:

1) файли в каталозі library завантажуються першими;
2) файли у піддиректоріях завантажуються до тих, які вище по дереву папок (але після файлів в library);
3) в одному каталозі, файли завантажуються в алфавітному порядку;
4) файли, які починаються з main.* завантажуються останніми.

Загальна структура даного проекту наведена на рисунку 1.7.

Метеор має вбудовану систему пакетів. Пакети це окремі JavaScript програми. Вони можуть впроваджувати код в клієнтську або серверну частину. Або містити бібліотеку нових функцій. Завдяки їм можна самим розширювати можливості фреймворку.

JQuery і Backbone пакети, дозволяють включити ці ж бібліотеки в застосунок.

Переглянути всі доступні на даний момент пакети можна командою meteor list.

Спочатку завантажаться колекції, які знаходяться в папці library. В папці lib знаходяться оголошення всіх колекцій системи. Потім завантажаться файли в папках в алфавітному порядку і файли в них також. Спочатку завантаження все в папці home, останніми завантажиться файли в sidebar. Після цього завантажаться файли main.css, main.html, router.js.

1.3. Конструювання програмної системи

Розглянемо деякі функції та особливості, які виконуються на сервері.

Для роботи проекту потрібно завантажити iron-router командою [18]:

```
meteor add iron-router.
```

Iron-router – це пакет маршрутизації для Meteor. Він створює односторінкові застосунки [19].

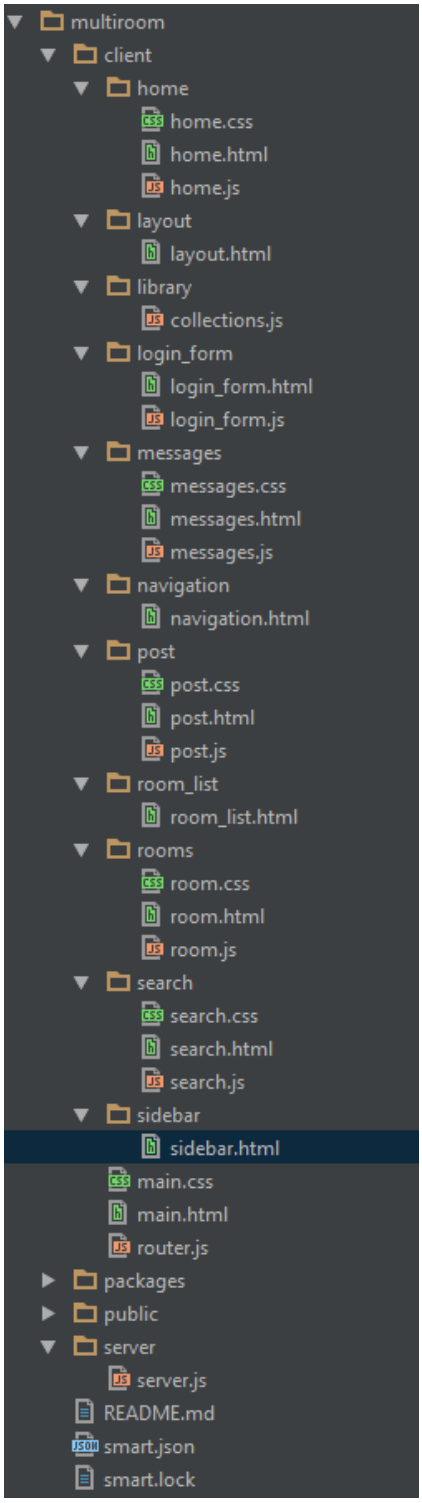

Рисунок 1.7 – Структура проекту

Бере на себе управління тегами, шаблонами на основі поточного URL користувача. Це також допоможе створювати підписки за маршрутом. Код додавання сторінки в проект наведений в лістингу 1.9.

Лістинг 1.9 – Код додавання сторінки

```
Router.route('rooms',{
        path: '/rooms',
    });
```

Щоб при вході на сайт різним користувачам виводилася інша інформація в залежності від ролі, потрібно описати функції авторизації. Код оголошення двох типів користувачів – простого пасажира та користувачів з правами адміністраторів наведений в лістингу 1.10.

Лістинг 1.9 – Код оголошення ролей користувачів

```
if (Meteor.users.find().fetch().length === 0) {
        var users = [{
            name: 'Рейнор',
            email: ' JimRaynor@gmail.com',
            roles: ['user']
        }, {
            name: 'Керіган',
            email: 'Kerrigan@gmail.com',
            roles: ['admin']
        }];
}
```

Є можливість зміни паролю. Код наведений в лістингу 1.11.

Лістинг 1.11 – Код реєстрації користувача та зміни паролю

```
_.each(users, function(userData) {
            var userid = Accounts.createUser({
                "nickname" : "Рейнор",
                 "email" : "JimRaynor@gmail.com",
                 "role" : ["user"]
```

```
                "password" : "ZergRush",
                "sex" : "чоловіча",
                "birthday" : "24.04.1991",
                "fmusic" : "Killers",
                "ffilms" : "Гладіатор",
                       "fgames" : "StarCraft",
                       "fserials" : "Гра престолів",
            username: userData.nickname,
            profile: {
                name: userData.nickname
            }
        })
        Meteor.users.update({
            _id: userid
        }, {
            $set: {
                'emails.0.verified': true
            }
        });
        Roles.addUsersToRoles(userid,['Рейнор']);
    })
```

В MongoDB досить легкий пошук даних з колекцій. Потрібно ввести поле та критерій пошуку.

В лістингу 1.12 наведено функцію пошуку кімнат.

Лістинг 1.12 – Код пошуку кімнат

```
Template.home.room_list = function(){
return Room.find({
    name: “Дота 2”
});
```

Аналогічно відбувається пошук, вставка, редагування і видалення в інших шаблонах.

1.3.1 Вибір мови програмування та середовища розробки

MeteorJS – це нова JavaScript-платформа, призначена для автоматизації та спрощення розробки Web-застосунків, що діють в режимі реального часу.

Було вибрано JavaScript через ряд причин:

1) легкість роботи з деякими елементами (випадаючі вікна);
2) перевірка даних форм і відправка на сервер;
3) можливість виконання обрахунків на стороні клієнта;
4) візуальна ефективність сайту;
5) легкість розробки ПЗ, велика кількість функцій;
6) підтримується фреймворком MeteorJS;
7) постійна підтримка мови.

Сам Meteor написаний на NodeJS і дозволяє створювати застосунки, де практично не існує відмінностей між серверною і клієнтською частиною. Код пишеться в одному стилі як для клієнта, так і для сервера. Завдяки цьому, основна увага приділяється самій логіці програми. Швидкість написання веб-застосунків відчутно зростає.

Meteor може створити реактивні застосунки. Коли користувач вводить дані, він отримує зворотний зв'язок миттєво, без необхідності чекати відповіді сервера. Якщо раніше подібні фреймворки нерідко використовували тільки для швидкого прототипування, то зараз Meteor використовують на цілком серйозних проектах [20].

Платформа спочатку розроблялася з урахуванням простоти навчання. Для початку розробки досить встановити Meteor, багато речей він робить самостійно. Meteor автоматично синхронізує дані між усіма користувачами в реальному часі: не потрібно думати про це, поки не з'явиться необхідність додати які-небудь правила, наприклад, що стосуються ролей користувачів у застосунку. Важливу роль в простоті Meteor грає зрозуміле і прозоре API.

Затребуваність можливості взаємодії з користувачами в реальному часі буде тільки рости.

Основні принципи: передача тільки даних – розмітка не передається в запитах. Відправляються лише дані з серверної частини; одна мова – використання JavaScript як на сервері, так і на клієнті; база даних доступна звідусіль – можна звертатися прямо з клієнтської частини програми.

Також до особливостей можна віднести – на стороні клієнта використовується попередня вибірка й імітація моделі, таким чином зв'язок з БД відбувається практично миттєво; усі шари, від БД до шаблонів мають доступний інтерфейс; Meteor з відкритим вихідним кодом об'єднує, але не замінює інші доступні загалу інструменти та фреймворки; зрозуміле і прозоре API.

Офіційна підтримка Linux і Mac. З недавнього часу Meteor можна використовувати і на Windows. Для Linux і Mac використовуємо команду:

```
curl https://install.meteor.com | /bin/sh
```

Створюємо проект multirooms:

```
meteor create multirooms
```

Запускається проект локально на http://localhost:3000/ командою meteor.

Розгортаємо на безкоштовному хостингу meteor:

```
meteor deploy multirooms.meteor.com
```

Отже, даний фреймворк було вибрано через наступні переваги:

1) Meteor набуває великої популярності;

2) використання JavaScript на серверній і клієнтські частині;
3) легка мова запитів до бази даних;
4) легке, зрозуміле API для роботи;
5) реактивність;
6) швидкість роботи;
7) легке завантаження бібліотек і пакетів;
8) легкість роботи з MongoDB;
9) синхронізація даних в реальному часі;
10) імітація моделі – шаблони.

1.3.2 Вибір СУБД та опис її фізичної моделі

Система управління базами даних (СУБД) — це сукупність програмних засобів, що забезпечує користувачам можливість створення, збереження, оновлення, пошук інформації та контролю доступу в базах даних.

Термін СУБД включає в себе досить велику кількість різних один від одного інструментів для роботи з базами даних (окремі програми і спільні бібліотеки). Так як дані бувають різних видів і типів, було розроблено величезну кількість різних СУБД та інших застосунків для роботи з БД. СУБД ґрунтуються на моделі бази даних – це спеціальні структури призначені для роботи з даними [21].

Хоча існують багато рішень для роботи з БД, популярними і затребуваними стають лише деякі з них. Найбільш часто застосовувана на сьогоднішній день – реляційна система управління базами даних, а з тих, які стають все популярнішими – документно-орієнтовані.

Нижче перераховані основні функції СУБД.

Визначення даних – визначити, яка саме інформація буде зберігатиметься в базі даних, задати властивості даних, їх тип (наприклад, число цифр або символів), а також вказати, як ці дані зв’язані між собою.

Обробка даних – дані можуть оброблятися самими різними способами. Можна вибирати будь-які поля, фільтрувати і сортувати дані. Можна об’єднувати дані з іншою, пов’язаною з ними, інформацією і обчислювати підсумкові значення.

Управління даними – можна вказати, кому дозволено знайомитися з даними, коректувати їх або додавати нову інформацію.

Вхідні до складу сучасних СУБД засоби спільно виконують наступні функції:

1) первинне введення, поповнення інформації в базі даних;
2) видалення застарілої інформації з бази даних;
3) коректування даних для підтримки їх актуальності;
4) впорядкування (сортування) даних по деяких ознаках;
5) пошук інформації по деяких ознаках (для опису запитів є спеціальна мова запитів, що забезпечує також інтерфейс між базою даних і прикладними програмами користувачів, дозволяє цим програмам використовувати бази даних);
6) підготовку і генерацію звітів (засоби підготовки звітів дозволяють створювати і роздруковувати зведення по заданих формах на основі інформації бази даних);
7) захист інформації і розмежування доступу користувачів до неї;
8) резервне збереження і відновлення бази даних, яке дозволяє відновити втрачену при збоях і аваріях інформацію бази даних;

9) підтримку інтерфейсу з користувачами, який забезпечується засобами ведення діалогу;
10) захист від необдуманих дій користувача, запобігаючи втрату інформації у разі поспішних або помилкових команд.

Є 2 види баз даних – структуровані і неструктуровані. Структуровані БД використовують структури даних, тобто структурований опис типу фактів за допомогою схеми даних, більш відомої як модель даних. Модель даних описує об'єкти та взаємовідносини між ними. Найбільш відомими структурованими базами даних є реляційні.

Реляційні БД підключаються до даних в різних файлах для використання загальних елементів або ключів полів. Інформація в реляційних БД зберігається у вигляді різних таблиць, кожна з яких має ключове поле, яке здатне ідентифікувати кожен запис, тобто ключове поле є унікальним для всієї таблиці. Реляційні БД є більш гнучкими в порівнянні з мережевими і ієрархічними базами даних. В реляційних базах даних, таблиці мають назву сутності, рядки або записи називаються кортежами, а стовпчики називаються атрибутами або полями.

Негативною стороною використання реляційного виду БД є те, що при пошуку інформації може бути витрачено більше часу, ніж при використанні інших видів БД.

До неструктурованих БД відносяться повнотекстові бази даних, котрі містять неструктуровані тексти статей чи книг у формі, що дозволяє здійснювати швидкий пошук.

Щодо неструктурованого підходу (NoSQL), то сам термін NoSQL з'явився близько десяти років тому. Як би це не здавалося дивним, але спочатку це було назвою чергової реляційної системи

управління базами даних. Проте передбачала вона трохи інше – уникнути використання стандартів SQL. У наступні кілька років інші підхопили цю ідею і стали використовувати цей термін стосовно нереляційних баз даних.

За задумом NoSQL бази даних і СУБД не мають внутрішніх зв'язків. Вони не ґрунтуються на одній моделі, а кожна база даних залежно від цілей використовує різні моделі.

Було вибрано NoSQL підхід, адже він прибирає всі обмеження реляційної моделі (недостатня продуктивність, трудомістке горизонтальне масштабування, недостатня продуктивність в кластері) і полегшує засоби зберігання і доступу до даних. Такі БД використовують неструктурований підхід, тим самим знімаючи обмеження жорстких зв'язків і пропонуючи різні типи доступу до специфічних даними.

NoSQL бази даних не використовують загальний формат запиту, такий як SQL в реляційних базах даних. Кожне NoSQL рішення використовує власну систему запитів.

Переваги NoSQL:

1) вкладена інформація – документо-орієнтовані сховища відмінно працюють з глибоко вкладеної, складною інформацією;

2) підтримка JavaScript – одна з відмінних особливостей документо-орієнтованих сховищ це те, як вони працюють з іншими за стосунками, підтримка JSON;

3) швидкість – NoSQL бази даних зазвичай швидше, а іноді набагато швидше, коли справа доходить до запису;

4) безсхемна розробка – реляційні СУБД вимагають чітко описану структуру даних до початку роботи, а NoSQL пропонує більш гнучкі рішення;

5) автоматизована (або дуже проста) реплікація/масштабування – NoSQL бази даних швидко розвиваються, тому розробники активно вирішують основні проблеми, одна з яких – реплікація і масштабування. На відміну від реляційних СУБД, NoSQL БД легко масштабуються і працюють з кластерами;

6) широкий вибір – коли справа доходить до вибору NoSQL сховища, існує великий вибір різних моделей.

Існує досить багато різних моделей і функціональних систем для NoSQL баз даних.

Тип БД ключ-значення працює з даними типу ключ-значення, наприклад, як словник. Тут немає місця ні структурі, ні зв'язкам. Після підключення до сервера (наприклад Redis) за стосунок може задати ключ і його значення, а надалі отримувати ці дані за запитом.

Такі СУБД зазвичай використовуються для швидкого збереження базових даних. Вони, зазвичай, дуже швидкі, працездатні або легко масштабуються (добре використовувати такі БД для зберігання сесій, кеша, лічильників відвідувань або переглядів і т.д.).

Розподілене сховище – це двовимірний масив, де кожен ключ (запис) містить одну або кілька пар ключ-значення прив'язаних до нього. Така система дозволяє зберігати і використовувати великі обсяги неструктурованих даних (один запис з великим кількість додаткової інформації). Такі бази даних зазвичай використовуються, коли недостатньо простих пар ключ-значення, а зберігати необхідно великий обсяг записів з різною інформацією. Такі СУБД при відповідному використанні можуть бути дуже продуктивними.

Для розробки даної системи використовувалася неструктурована документо-орієнтована база даних для виконання швидкого пошуку та більшої гнучкості.

Документо-орієнтована СУБД – це СУБД спеціально призначена для зберігання ієрархічних структур даних (документів) і зазвичай реалізована за допомогою підходу NoSQL. В основі документо-орієнтованих СУБД лежать документні сховища, що мають структуру дерева. Структура дерева починається з кореневого вузла і може містити декілька внутрішніх і листових вузлів. Листові вузли містять дані, які при додаванні документа заносяться в індекси, що дозволяє навіть при досить складній структурі знаходити місце (шлях) шуканих даних. API для пошуку дозволяє знаходити за запитом документи та частини документів. На відміну від сховищ типу ключ-значення, вибірка за запитом до документному сховища може містити частини великої кількості документів без повного завантаження цих документів в оперативну пам'ять. Вони допускають набагато більшу вкладеність і складність структури даних.

Документо-орієнтовані сховища відмінно зберігають незв'язану інформацію великих обсягів, навіть якщо вона дуже різниться від сутності до сутності.

Документи можуть бути організовані (згруповані) в колекції. Їх можна вважати віддаленим аналогом таблиць реляційних СУБД, але колекції можуть містити інші колекції. Хоча документи колекції можуть бути довільними, для більш ефективного індексування краще об'єднувати в колекцію документи зі схожою структурою.

Одним з ключових елементів всіх документо-орієнтованих баз даних є те, що вони можуть працювати з набагато більшими структурами і наборами даних, ніж зазвичай. Зокрема, зважаючи на їх розподіленої природи й іншого способу фізичного зберігання даних, вони ідеально підходять там, де потрібно обробляти величезну кількість інформації.

Документо-орієнтовані бази даних мають гнучку структуру, яка забезпечує ряд важливих особливостей. Було вибрано саме такий тип через ряд особливостей.

Відсутність схеми: документо-орієнтованим базам даних не потрібна зумовлена структура даних, яку потрібно зберігати в базі даних. У традиційних РСУБД спочатку визначається структура таблиць, в яких будуть зберігатися дані, для чого потрібно знати наперед зміст, можливі значення і структуру інформації. У документо-орієнтованої базі даних інформацію можна зберігати прямо в документах і не потрібно турбуватися про структуру, достатню кількість полів і в більшості випадків навіть про те, що таке відношення «один до багатьох» і «багато до багатьох». Замість цього можна зосередитися на утриманні самої інформації. Зберігати необроблені документи та інформацію набагато простіше, навіть якщо вона надходить з різних джерел.

Логічні об'єкти: більшість рішень на основі СУБД використовується для моделювання інформації, яка зазвичай зберігається в структурованому форматі. Потім за допомогою SQL і з'єднань ця інформація компонується в об'єкт, який використовується всередині програми. Різні елементи загальної структури даних можна переглядати окремо, але часто інформація скомбінована і надається відповідно з об'єктом, в якому зібрані всі дані.

Мігруюча структура: з плином часу дані змінюються, іноді швидко, іноді повільно. Зміна структури даних – це складний процес, який не тільки впливає на використовувану базу даних, але і вимагає внесення змін у застосунки, які звертаються до цієї інформації і працюють з нею. У разі структури на основі документів, так як структура даних фіксована, адаптація цієї структури для нових версій і різних форматів вихідних даних виявляється важким і складним

справою. Потрібно створити нову таблицю або змінити існуючу для нової структури, а це означає перетворення всіх раніше створених записів. У разі документо-орієнтованої бази даних структура документів може змінюватися.

Документо-орієнтовані бази даних можуть зберігати будь-яку інформацію, але найбільш широко використовується структурний формат JSON, формат запису об'єктів з мови JavaScript. Він дозволяє зберігати дані, що складаються з рядків, чисел, масивів і записів (хеш), а також комбінацій цих основних типів.

Популярні СУБД:

1) Couchbase – засноване на JSON і сумісне з Memcached сховище [22];
2) CouchDB – документо-орієнтоване сховище [23];
3) MongoDB – дуже популярне і функціональне сховище [24].

Найбільш популярні представники цього сімейства – MongoDB і CouchDB. Обидва підтримують реплікацію і шардінг. Крім того, обидва рішення реалізують MapReduce. CouchDB використовує його в якості заміни складним запитам: СУБД створює «відображення» (view), дані яких обчислюються операціями MapReduce і автоматично оновлюються при зміні. MongoDB також підтримує досить складні запити. Для цього в рішенні реалізована власна, не схожа на SQL, мова запитів. Основна відмінність між MongoDB і CouchDB – це різний підхід до конкурентних змін даних. MongoDB не реалізує транзакційність і здійснює лише найпростіші атомарні операції над об'єктами. CouchDB ж реалізує управління конкурентним доступом по моделі Multi-Version Concurrency Control. Таким чином, в CouchDB зміна документів одним користувачем не видна іншим до моменту коміту.

Було вибрано саме документно-орієнтовану базу даних для більшої гнучкості та швидкості роботи, а саме – MongoDB, адже ця база даних більш простіша за аналогічні NoSQL бази даних.

MongoDB – високопродуктивна документо-орієнтована база даних. Особливості цієї СУБД:

1) документне сховище, яке не потребує створення схем (таблиць), адже перевага полягає в тому, що не потребується зберігати порожні клітинки даних у кожному документі;
2) об'єктна мова запитів на основі JSON (яка набагато легша SQL);
3) широкий набір (атомарних) операцій над даними (умовний пошук, складна вставка/оновлення);
4) різні типи даних;
5) підтримка складних масивів, адже кожен елемент масиву може представляти із себе об'єкт;
6) підтримка індексів (B-Tree);
7) автозбереження, шардінг і реплікація;
8) профілювання, зберігання великих об'єктів, адміністративний інтерфейс, серверні функції, MapReduce для розподілених операцій над даними.

2 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Опис додаткових засобів програмування

Для кращої візуалізації використовувався фреймворк Bootstrap. Bootstrap – це чудовий фреймворк для створення сучасних, крос-браузерних інтерфейсів. Структура коду HTML, JavaScript і CSS надає можливість створювати безліч найрізноманітніших елементів інтерфейсу і сітку сайту [25].

Нинішній рівень розвитку цього фреймворку вже дозволяє повністю створити будь-який web-інтерфейс.

Основною перевагою використання Bootstrap є Less – динамічна мова стилів, яка істотно розширює можливості CSS. З допомогою Less можна створювати змінні, вкладені колонки. Ним досить просто користуватися.

Основні інструменти Bootstrap:

1) сітки – наперед задані розміри колонок, які можна відразу ж використовувати;
2) шаблони – фіксований та гнучкий шаблон документа;
3) типографіка – опис шрифтів, визначення деяких класів для шрифтів;
4) медіа – представляє можливість управління зображеннями та відео;
5) таблиці – засоби оформлення таблиць;
6) навігація – класи оформлення для вкладок, сторінок, меню і панелей інструментів.

2.2 Реалізація інтерфейсу системи та тестування

Для користування системою користувач повинен ввійти під їм нікнеймом або електронною поштою (див. рисунок 2.1).

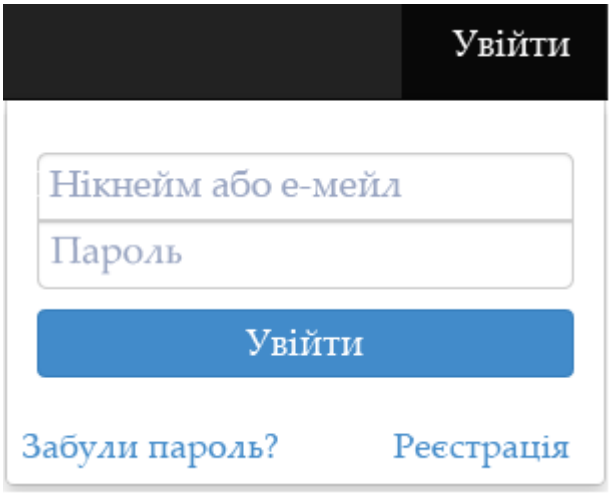

Рисунок 2.1 – Авторизація користувача

При введенні неправильного e-mail-у чи паролю система сповістить користувача про це (рисунок 2.2).

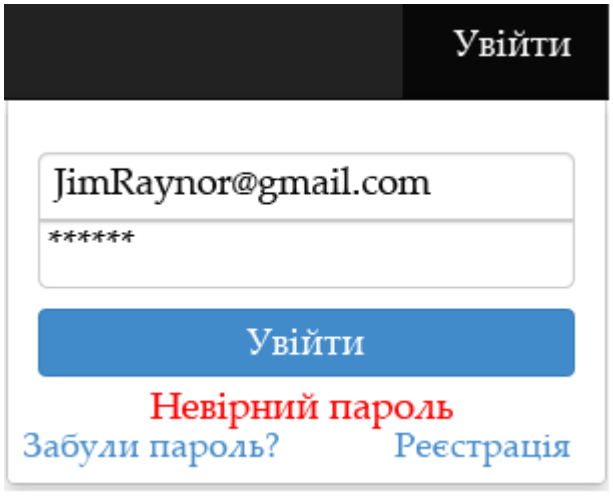

Рисунок 2.2 – Перевірка авторизації

Можлива реєстрація нового користувача (рисунок 2.3.)

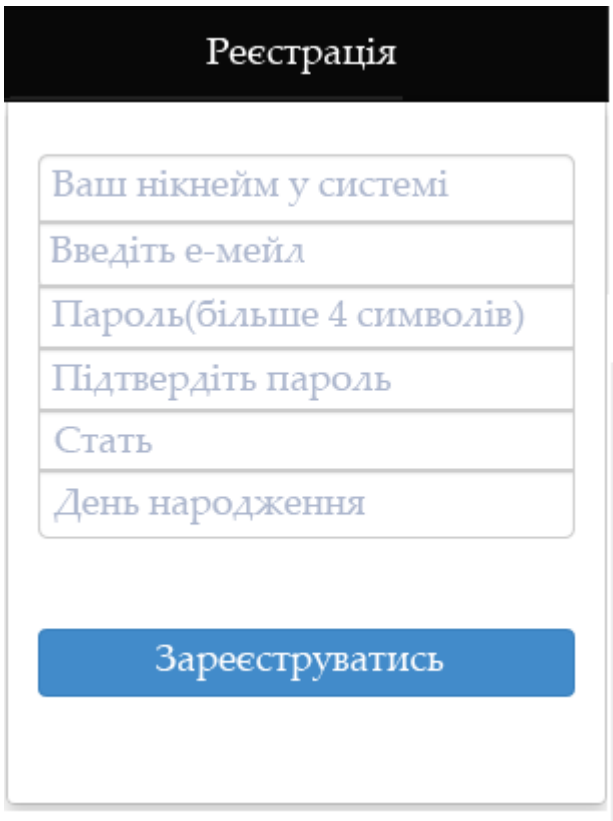

Рисунок 2.3 – Реєстрація нового користувача

Стартова сторінка – це сторінка зі списком кімнат (рис. 2.4)

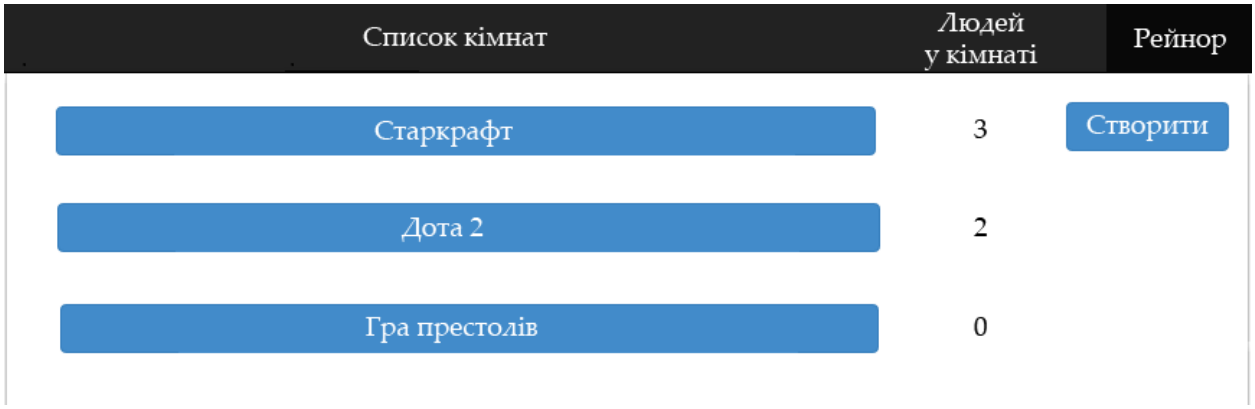

Рисунок 2.4 – Список кімнат

Також у користувача є сторінка профілю (рис. 2.5), на якій вказано його дані, фото профілю, а також є розділ з інтересами, такими як: улюблена музика, ігри, фільми, серіали.

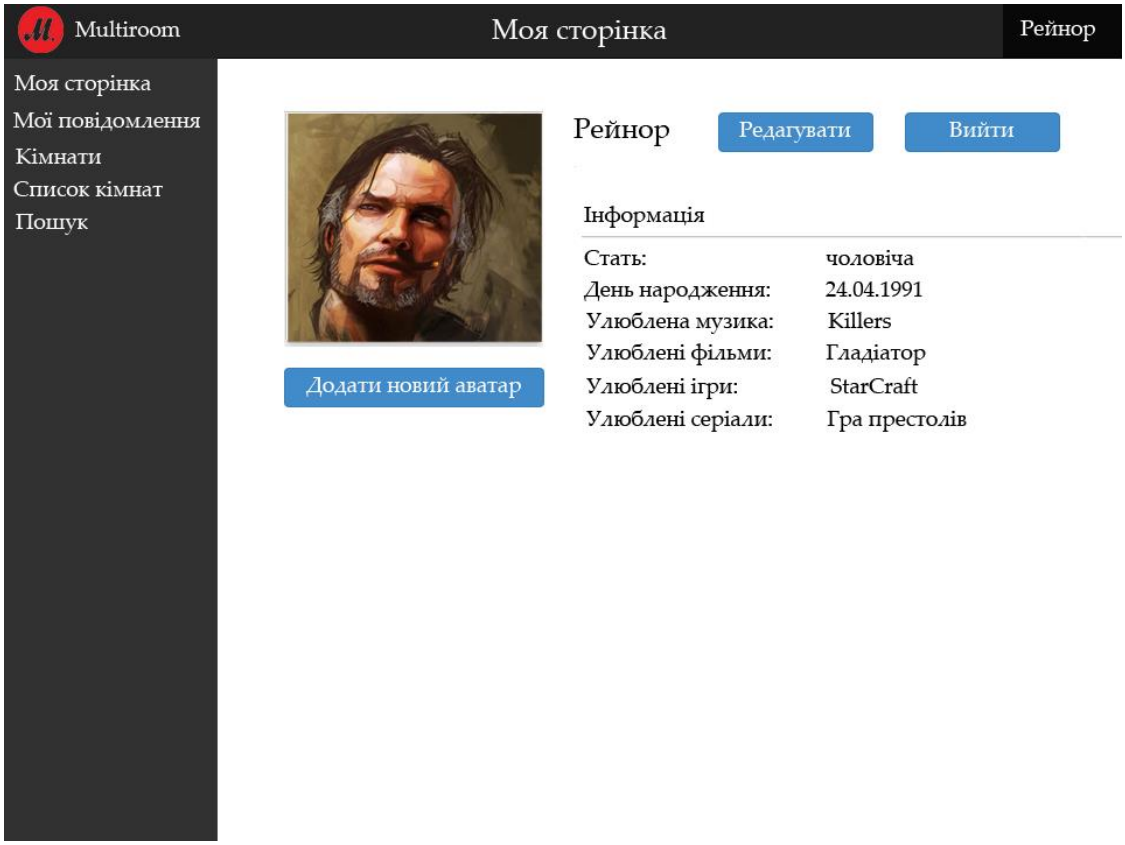

Рисунок 2.5 – Сторінка профілю користувача

Безпосередньо сама кімната (рисунок 2.6) відображає користувачів, які в ній знаходяться, пости цих користувачів, кількість лайків під кожним постом і назву кімнати.

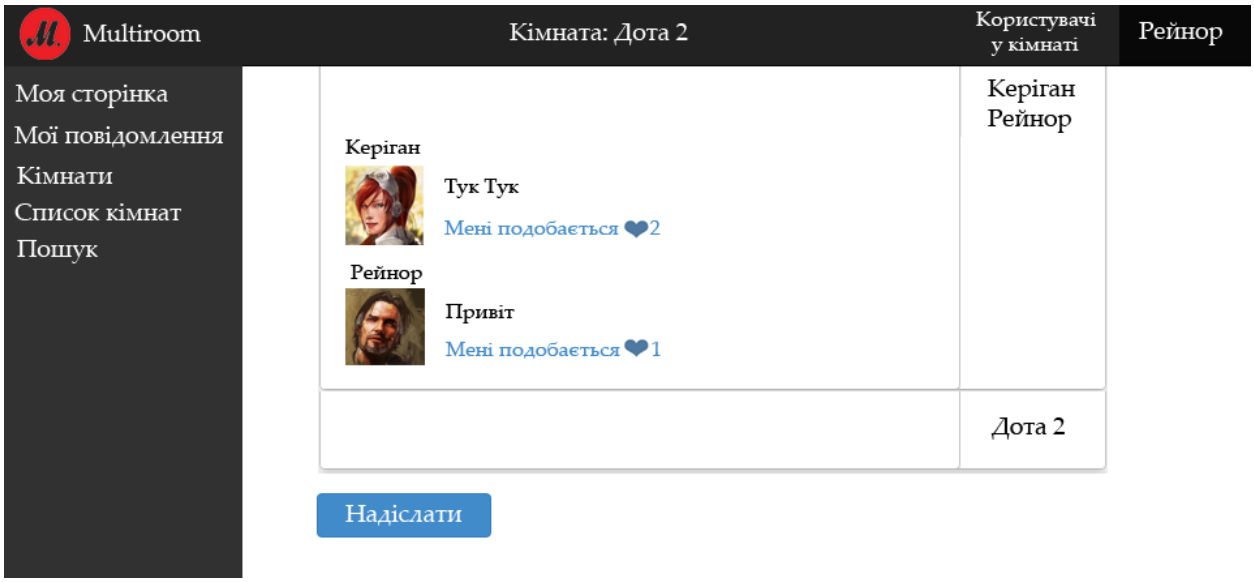

Рисунок 2.6 – Сторінка кімнати

3 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

Для здійснення економічних розрахунків використано реальні дані роботи, описано технологічний процес розробки із зазначенням трудомісткості кожної операції, визначено суму матеріальних затрат, та витрат на оплату праці основного і допоміжного персоналу, включно з відрахуванням на соціальні заходи, обчислено додаткові затрати, а також визначено економічну ефективність та термін окупності продукту.

3.1 Планування стадій та етапів проєктування програмного забезпечення

Відповідно до ст. 1 Закону № 3792, комп'ютерна програма – це набір інструкцій у вигляді слів, цифр, кодів, схем, символів чи у будь-якому іншому вигляді, виражених у формі, придатній для зчитування комп'ютером, які приводять його в дію для досягнення певної мети або результату. В такому самому значенні розуміється цей термін і в контексті податкового обліку (пп. 8.2.2 Закону про прибуток). Статтею 8 Закону № 3792 передбачено, що комп'ютерні програми є об'єктом авторського права.

Діяльність з програмування – це процес створення (розробки) програми, який може бути представлений послідовністю таких кроків: формулювання вимог до програми та розробка алгоритму; кодування (запис алгоритму на вибраній мові програмування); відлагодження; тестування; доопрацювання програми; розробка довідкової системи.

Проєктування ПЗ складається з низки послідовних, цілеспрямованих, взаємозв’язаних та взаємообумовлених етапів, на

які можна поділити весь часовий відрізок, що відводиться на розробку проєкту (таблиця 3.1).

Таблиця 3.1 – Етапи розробки проєкту

Найменування		Вид роботи	
стадії	Етапу	шифр	зміст роботи
1 Підготовча стадія	1.1 Вивчення стану предметної області	1.1.1	Дослідження проблеми
		1.1.2	Вибір платформи для реалізації проєкту
		1.1.3	Вивчення та аналіз аналогічних розробок
		1.1.4	Економічне обґрунтування доцільності виконання проєкту
	1.2 Розробка технічного завдання	1.2.1	Складання ТЗ
		1.2.2	Узгодження ТЗ із зацікавленими сторонами
		1.2.3	Складання плану та розрахунок розробки
2 Технічна пропозиція	2.1 Аналіз ТЗ та техніко-економічне обгрунтування проєкту	2.1.1	Доведення техніко-економічного обгрунтування
		2.1.2	Аналіз ТЗ та визначення приоритетних аспектів розробки
		2.1.3	Доведення та уточнення загального обсягу робіт, строків виконання та витрат
3 Теоретична розробка	3.1 Теоретичне вивчення задачі	3.1.1	Дослідження технічних особливостей інформаційної системи
		3.1.2	Визначення переліку технологій, які використовуватимуться при розробці та мови програмування
		3.1.3	Визначення форматів даних та запитів і їх узгодження із розробниками проєкту
		3.1.4	Розробка алгоритмів роботи програми на високому рівні
		3.1.5	Розробка структури систем забезпечення та схеми взаємодії її компонент
4 Практична реалізація	4.1 Реалізація окремих модулів ПЗ	4.1.1	Розробка алгоритмів роботи програми на низькому рівні
		4.1.2	Розробка окремих класів та компоновка їх у модулі
		4.1.3	Розробка графічного користувацького інтерфейсу для системи
	4.2 Відладка ПЗ	4.2.1	Автономна відладка окремих модулів системи
		4.2.2	Комплексна відладка ПЗ системи

Продовження таблиці 3.1

		4.3.1	Розробка структурної схеми субмодуля
	4.3 Розробка субмодуля	4.3.2	Розробка функціональної схеми субмодуля
		4.3.3	Розробка алгоритму функціонування субмодуля
		4.3.4	Обгрунтування вибору елементного базису та розробка схеми
Найменування		Вид роботи	
5 Доробка всього комплексу	5.1 Тестування всієї системи	5.1.1	Тестування системи у реальних умовах
		5.1.2	Доробка систем із урахуванням результатів тестування
	5.2 Підготовка документації по системі	5.2.1	Підготовка звіту про розробку системи
		5.2.2	Підготовка технічної документації
		5.2.3	Розробка довідкової системи, допоміжної і супроводжувальної документації, запис CD/DVD диска
6 Заключна стадія	6.1 Ознайомлення зацікавлених осіб з проєктом	6.1.1	Підготовка презентації
		6.1.2	Демонстрація системи

Головна мета планування процесу розробки – це визначення необхідних ресурсів на всіх його етапах.

Традиційний процедурний підхід для розробки ПЗ в основі якого лежать алгоритми, процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію.

Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий, покомпонентний процес розробки.

Підготовча стадія, технічна пропозиція і заключна стадія при об’єктно-орієнтованому підході залишаються незмінними. Стадії теоретичної розробки, практичної реалізації та доробки всього комплексу програмного забезпечення описані з точки зору специфіки об’єктно-орієнтованого підходу із врахування ЖЦ розробки ПЗ і

представлені із уточненням фахівців, залучених у кожний робочий процес, та артефактів (таблиця 3.2).

При створенні ПЗ за об’єктно-орієнтованим підходом необхідно залучити більшу кількість фахівців, більшу увагу приділити покроковій покомпонентній розробці, що, можливо, збільшить робочий час і витрати. Однак, цей підхід значно скорочує витрати на подальший супровід і модернізацію, що приводить до зменшення загальних витрат.

Таблиця 3.2 - Робочі процеси життєвого циклу розробки ПЗ

№	Робочі процеси	Діяльності	Співробітники	Артефакти
1	«Визначення вимог»	Ідентифікація актантів (А) і варіантів використання (ВВ), Модель ВВ Описи (формалізація) ВВ і сценаріїв реалізації, Визначення пріоритетності ВВ Створення прототипів інтерфейсів користувача (ІК)	Системний аналітик Специфікатор ВВ	Модель ВВ Актанти Глосарій ВВ Прототип ІК
2	«Проєктування»	Проєктування архітектури, Визначення вузлів та мережевих конфігурацій Визначення систем та їх інтерфейсів Визначення підсистем та зв’язків між ними Визначення інтерфейсів підсистем Визначення архітектурно значущих класів та класів проєктування, Визначення загальних механізмів проєктування Проєктування ВВ Проєктування класів Визначення вимог до реалізації	Архітектор Інженер з ВВ	Модель про-ектування Модель розміщеня Опис арітектури Проєкти реквізитів ВВ Класи про-ектування
3	«Реалізація»	Модель реалізації, Компоненти, Інтерфейси компонентів, Опис архітектури План збирання системи, Реалізація архітектури, Ітеративна реалізація класів Проєктування з генерацією програмного коду, Збірка системи, Планування білдів Реалізація білдів і системи в цілому	Системний інтегратор Інженер-програміст	Модель реа-лізації Опис архітекту-ри План збірки Компоне-нти Підсист-еми реалізації Інтерфейси

Продовження таблиці 3.2

	«Тестування»	Модель тестування Тестові приклади Процедура тестування Тестові компоненти План і оцінка тестування Розробка тестів Тестування цілісності Тестування системи Оцінка результатів тестування	Інженер з тестування Системний тестер	Модель тестування Тестові приклади План тестування Оцінювання тестів

Терміни розробки залежать від замовника, від наданої необхідної інформації (зразків довідників, документів і т.і.), рівня та порядку оплати та інших умов, від замовлення, і можуть становити від 2 тижнів до 6 місяців і більше (за даними сотфверних компаній).

Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки і становить від 200 до 5000 гривень (за даними аутсорсингових компаній). Роботу зі складання технічного завдання веде керівник проєкту разом із програмістами та консультуючись із замовником.

Для визначення загальної тривалості проведення робіт доцільно дані витрат часу по окремих операціях техпроцесу звести у таблицю (таблиця 3.3).

Таблиця 3.3 – Тривалість та трудоємність розробки та реалізації програмного проєкту

Найменування роботи	Виконавець, посада, спеціальність	К-сть виконавців	Тривалість виконання роботи, дні
2	3	4	5
Дослідження предметної області, вибір середовища для реалізації проєкту, вивчення та аналіз аналогічних розробок	Керівник проєкту	2	1
	Програміст		

Продавження таблиці 3.3

Економічне обгрунтування доцільності виконання проєкту, складання ТЗ, узгодження ТЗ із зацікавленими сторонами, складання плану та розрахунок розробки	Керівник проєкту	2	1
	Програміст		
	Програміст		
Доведення техніко-економічного обгрунтування, аналіз ТЗ та визначення приоритетних аспектів розробки, доведення та уточнення загального обсягу робіт, строків виконання та витрат	Керівник проєкту	2	1
	Програміст		
Теоретична розробка процедурний підхід	Програміст	1	3
Теоретична розробка об'єктно- орієнтований підхід	Системний аналітик, специфікатор ВВ, архітектор, інженер з ВВ	4	4
Практична реалізація процедурний підхід	Програміст	1	5
Практична реалізація об'єктно-орієнтований підхід	Системний інтегратор, інженер-програміст	3	7
Тестування програмного забезпечення	Інженер з тестування	1	3
Доробка всього комплексу програмного забезпечення	Програміст	1	4
Заключна стадія	Керівник проєкту	2	1
	Програміст		

Як для процедурного, так і для об'єктно-орієнтованого підходу, підготовча і заключна стадії та технічна пропозиція будуть виконуватися одинаково, теоретична розробка і практична реалізація для об'єктно-орієнтованого підходу вимагатимуть залучення більшої кількості ІТ-спеціалістів, що, можливо, збільшить тривалість виконання роботи.

3.2 Розрахунок витрат на реалізацію проєкту та оцінка економічної ефективності

Оцінка вартості комп'ютерних програм базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Оподаткування заробітної плати програмістів підприємства-розробника, яким встановлено певний посадовий оклад, відбувається аналогічно оподаткуванню зарплати працівників інших галузей господарства.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, розробники часто йдуть на додаткові заходи їх стимулювання, які найчастіше втілюються у:

- певній системі преміювання (за виконання графіку розробки чи його дострокове виконання, оптимізацію етапів розробки тощо). Премії оподатковуються аналогічно витратам на оплату праці;

- авторській винагороді за кожну продану одиницю програми (роялті). Сума роялті не обкладається внесками до Пенсійного фонду та фондів соціального страхування, а використання авторських прав оформлюється відповідним договором (ліцензійний авторський договір).

Разом з тим до створення ПЗ можуть бути залучені також позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем:

- якщо виконавець за договором підряду не зареєстрований фізособою-підприємцем, то виплати за таким договором

оподатковуються ПДФО за ставкою 15 %, також нараховуються (у розмірі 33,2 %) та утримуються (у розмірі 2 %) внески до Пенсійного фонду (п. 1 та п. 4 ст. 4 Закону № 400). Оподаткування провадиться у межах максимальної величини, що дорівнює 15 розмірам прожиткового мінімуму, встановленого для працездатних осіб;

- якщо ж виконавець за договором підряду є фізособою-підприємцем, яка сплачує єдиний податок чи знаходиться на загальній системі оподаткування, виплати, що здійснюються на його користь в рамках договору підряду, не оподатковуватимуться ПДФО.

Вважатимемо, що усі програмісти працюють у штаті підприємства-розробника і їм встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість і основна заробітна плата кожного учасника техпроцесу представлено у таблиця 3.4.

Таблиця 3.4 – Сума часткових заробітних плат

	Місячний оклад, грн.	Денна зар. плата, грн	Трудомісткість, людино-дні		Основна заробітна плата, грн.	
			Процедурний підхід	Об'єктно-орієнтований підхід	Процедурний підхід	Об'єктно-орієнтований підхід
Керівник проєкту	18200	728	4	4	2912	2912
Інженер-програміст	15200	608	17	19	10336	11552
Інженер-тестувальник	12100	484	3	3	1452	1452
Дизайнер	7500	300	3	3	900	900
Всього			27 днів	29 днів	15600 грн.	16816 грн.

Визначимо витрати на основну заробітну плату. Вони складають суму заробітних плат за кожну із робіт. Витрати на основну зарплату для процедурного та об'єкт-орієнтованого підходів:

Додаткова заробітна плата обчислюється за формулою $ЗП_{дод}=ЗП_{осн}\cdot 0.2$.

Обчислимо витрати на додаткову зарплату для двох підходів:

$ЗП_{осн(п)} = 2912 + 10336 + 1452 + 900 = 15600$ (грн.);

$ЗП_{осн(о)} = 2912 + 11552 + 1452 + 900 = 16816$ (грн.);

Це сума часткових заробітних плат за кожну роботу, і вона приведена в таблиці 3.4.

Додаткова заробітна плата обчислюється за формулою:

$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн}$.

$ЗП_{дод(п)} = 15600 \cdot 0.2 = 3120$ (грн.);

$ЗП_{дод(о)} = 16816 \cdot 0.2 = 3363.2$ (грн.);

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$ФЗП = ЗП_{осн}+ЗП_{дод}$.

$ФЗП_{п} = 15600 + 3120 = 18720$ (грн.);

$ФЗП_{о} = 16816 + 3363.2 = 20179.2$ (грн.);

З цієї суми утримуються обов'язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов'язкове соціальне страхування та податкової соціальної пільги.

Таким чином обов'язкові відрахування на заробітну плату для працівників складають:

Утримання єдиного соціального внеску:

$Відр_{ЄСВ1} = 0.036 \cdot ФЗП = 0.036 \cdot 18720 = 674$ грн.;

$Відр_{ЄСВ2} = 0.036 \cdot ФЗП = 0.036 \cdot 20179.2 = 726.5$ грн.

Податок на доходи фізичних осіб:

$Відр_{ПДФО1} = 0.15 \cdot (ФЗП - Відр_{ЄСВ}) = 0.15 \cdot (18720\text{-}674) = 2706.9$ грн.;

$Відр_{ПДФО2} = 0.15 \cdot (ФЗП - Відр_{ЄСВ}) = 0.15 \cdot (20179.2\text{-}726.5) = 2917.9$ грн.

$Відр_1 = Відр_{ЄСВ1} + Відр_{ПДФО1} = 674 + 2706.9 = 3380.9$ грн.;

$Відр_2 = Відр_{ЄСВ2} + Відр_{ПДФО2} = 726.5 + 2917.9 = 3644.4$ грн.

Законом № 1621 підрозділ 10 розділу XX Перехідних положень Податковий кодекс України від 02 грудня 2010 року № 2755-VI зі змінами та доповненнями (далі – ПКУ) доповнено пунктом 16, яким тимчасово, до 1 січня 2015 року, встановлено військовий збір.

Водночас Законом України від 28 грудня 2014 року № 71-VIII «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» (далі – Закон № 71), який набрав чинності з 01.01.2015, оподаткування військовим збором подовжено до набрання чинності рішенням Верховної Ради України про завершення реформи Збройних Сил України (пункт 16 підрозділ 10 розділу XX Перехідних положень ПКУ).

Платниками збору в ромірі 1,5% від заробітної плати є особи, визначені пунктом 162.1 статті 162 цього ПКУ (підпункт 1.1 пункт 16 підрозділу 10 ПКУ).

Військовий збір з фізичних осіб:

$ВЗФО = 0{,}015 \cdot ФЗП$

$ВЗФО_1 = 0{,}015 \cdot 18720 = 280{,}8$ грн.

$ВЗФО_2 = 0{,}015 \cdot 20179.2 = 302{,}7$ грн.

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%, для підприємців розмір єдиного внеску залежно від класу професійного ризику виробництва становить від 36,76 % до 49,70 %. Зокрема, видання програмного забезпечення – 36,77%).

Нарахування на фонд оплати праці (*ФОП*):

$ФОП_{ЄСВ} = 0.3677 \cdot ФЗП$

$ФОП_{ЄСВ1} = 0.3677 \cdot 18720 = 6883.34$ грн.;

$ФОП_{ЄСВ1} = 0.3677 \cdot 20179.2 = 7419.82$ грн.

Всього витрат:

$В_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1} = 18720 + 6883.34 = 25603.34$ грн.;

$В_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2} = 20179.2 + 7419.82 = 27599.02$ грн.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{Bi} = q_i \cdot p_i$$

де q_i– кількість витраченого матеріалу *i*-го виду; p_i– ціна матеріалу *i*-го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{B\,i}$$

Проведені розрахунки занесемо у таблицю 3.5:

Таблиця 3.5 – Зведенні розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	SCRUM-дошка	шт	1	500,00	500,00
2	Папір для друку	листів	120	0,25	30,00
3	Чорнила для принтера	шт	1	65,00	65,00
4	Маркери	шт	4	14,00	56,00
Всього					651,00

Згідно постанов Національної комісії, що здійснює державне регулювання у сфері енергетики (згідно Постанов Національної комісії, що здійснює державне регулювання у сфері енергетики (НКРЕ) від 22.01.2001р. №47 зі змінами і доповненнями, від 06.12.2002р. № 1358, від 22.10.2004р. № 1030, від 28.04.2016р. № 755 на підставі Закону України від 03.12.1999р № 1274-XIV “Про внесення змін до Закону України “Податкового кодексу України”, згідно Закону України “Про міський електричний транспорт” від 29 червня 2004 року № 1914 – IY; також затвердженого постановою Кабінету Міністрів України від 01.06.2011р. № 869, Порядку розрахунку роздрібних тарифів на електричну енергію, тарифів на розподіл електричної енергії (передачу електричної енергії місцевими (локальними) електромережами), тарифів на постачання електричної енергії за регульованим тарифом, затвердженого постановою Національної комісії, що здійснює державне регулювання в сфері енергетики та комунальних послуг (НКРЕКП) "Про ринкове формування роздрібних тарифів на електричну енергію, що відпускається для кожного класу споживачів, крім населення, на території України", ця сума становить 289,85 коп/кВт.г.

Витрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_е = W \cdot T \cdot S$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії.S=2,90грн/кВт.г.

1 $З_{е1} = 1{,}5 \cdot 216 \cdot 2{,}9 = 939{,}6$ грн.;

2 $З_{е2} = 1{,}5 \cdot 232 \cdot 2{,}9 = 1009{,}2$ грн.

Розрахунок суми амортизаційних відрахувань.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Амортизація за час (період) використання обладнання (комп'ютера) складає долю затрат, які припадають на дослідницьку роботу (написання програми), і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}}$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ГОД}$ – річний робочий фонд часу, год; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год.

Таблиця 3.6 – Перелік обладнання, необхідного для розробки програми

Найменування	Кількість, шт.	Ціна за одиницю продукту, грн.	Сума, грн.
1	2	3	4
Комп'ютер	4	15775	63100
Принтер	1	2000	2000
Apple Developer Program Membership	1	3250	3250
Google Play Developer Membership	1	900	900

Продавження таблиці 3.6

1	2	3	4
Хостинг	1	995	995
Доменне ім'я	1	366	366
Всього			70611

$$A_1 = \frac{70611 \cdot 0{,}6 \cdot 216}{2016} = 4539{,}3\,\text{грн.};\ A_2 = \frac{70611 \cdot 0{,}6 \cdot 232}{2016} = 4875{,}52\,\text{грн.}$$

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$НВ = 0{,}5 \cdot ЗП_{осн}$.

$НВ_п = 0{,}5 \cdot 15600 = 7800$ грн.;

$НВ_о = 0{,}5 \cdot 16816 = 8408$ грн.

Проведемо розрахунок вартості створюваного проґрамного продукту. Вартість продукції включає у собі собівартість і планований прибуток.

Повна собівартість проґрамного продукту дорівнює сумі усіх витрат на його виробництво:

$Св = В_{зп} + З_{мв} + З_е + А + НВ;$

$Св_1 = 25603{,}3 + 651 + 939{,}6 + 4539{,}3 + 7800 = 38882{,}2$ (грн.)

$Св_2 = 27599{,}02 + 651 + 951{,}8 + 4875{,}52 + 8408 = 41891{,}74$ (грн.)

Приймемо прибуток на рівні 27%. Для нових інноваційних продуктів, що користуються високим попитом на ринку.

Отже, вартість розробленого програмного забезпечення:

$В_1 = Св_1 + 0,27 \cdot Св_1 = 38882,24 + 0,27 \cdot 38882,24 = 49380,44$ грн.;

$В_2 = Св_2 + 0,27 \cdot Св_2 = 41891,74 + 0,27 \cdot 41891,74 = 53202,51$ грн.

Економічна ефективність (E) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св}$$

де $П$ – прибуток, $П = В - Св$; $Св$ – собівартість.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку.

$E_п = (49380,44 - 38882,24) / 38882,24 = 0,27;$

$E_о = (53202,51 - 41891,74) / 41891,74 = 0,27.$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E}$$

У нашому випадку $T_{ок1} = T_{ок2} = 1/0,27 = 3,7$ років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Загальна вартість пропонованих робіт по розробці програмного продукту становить 18720 грн. для першого варіанту та 20179,2 грн. для другого. Оскільки ефективність для обидвох проєктів відповідно

до встановленого рівня прибутку становить 0,27, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

3.3 Визначення витрат на супровід і модернізацію програмного продукту та уточнений аналіз ефективності вкладених інвестицій

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за процедурним методом 55% від початкових витрат, а за об'єктно-орієнтованим – 25%.

Собівартість модернізації:

$СвМ_п = 0,6 \cdot Св_п = 0,6 \cdot 38882,24 = 23329,35$ грн.,
$СвМ_о = 0,2 \cdot Св_о = 0,2 \cdot 41891,1 = 8378,22$ грн.

Для споживача вартість модернізації:

$М_п = 0,6 \cdot В_п = 0,6 \cdot 47719,7 = 29628,27$ грн.,
$М_о = 0,2 \cdot В_о = 0,2 \cdot 51710,7 = 10640,50$ грн.

Таким чином, уже після першої модернізації, загальні витрати на створення і супровід ПЗ для виробника за об'єктно-орієнтованим методом менші, ніж за процедурним, навіть якщо його собівартість є дещо дорожчою.

$ЗВ_{п\ (вир)} = 38882,24 + 23329,35 = 62211,59$ грн.,
$ЗВ_{о\ (вир)} = 41891,74 + 8378,22 = 50269,96$ грн.

Як і для споживача:

$ЗВ_{п}$ = 49380,44 + 29628,27= 79008,71 грн.,

$ЗВ_{о}$ = 53202,51 + 10640,50 = 63843,01 грн.

Річна економія витрат за всіма можливими напрямами і додатковими витратами, пов'язаними з супроводом і тільки одноразовою модернізацією (у розрахунку на одиницю продукції) при об’єктно-орієнтованому методі порівняно із процедурним:

$\Delta C_{(вир)} = ЗВ_{1(вир)} - ЗВ_{2(вир)}$ = 62211,59 – 50269,96 = 11941,63 грн,

$\Delta C = ЗВ_1 - ЗВ_2$ = 79008,27 – 63843,01 = 15165,70 грн,

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{TBi} - \sum_{i=1}^{t} ІК_i \alpha_{TBi};$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

α_{TBi} – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n};$$

де i – ставка дисконтування або норма дисконту, i = 0,27;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$$\alpha_0 = 1, \quad \alpha_1 = \frac{1}{(1+0,27)} = 0,79$$

Вважатимемо, що обидва програмних продукта одинаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми одинаково вплинуть на розмір його додаткових доходів на вкладений капітал, Тому приймемо цю величину за

постійну, а порівняння дохідності двох проєктів проведемо тільки за витратами.

$$ЧПД_1' = ГП + 0{,}78 \cdot ГП - 47719{,}7 - 0{,}78 \cdot 28631{,}8 = 1{,}78ГП - 70052{,}5 \text{ грн.},$$

$$ЧПД_2' = ГП + 0{,}78 \cdot ГП - 51710{,}7 - 0{,}78 \cdot 10342{,}14 = 1{,}78ГП - 59777{,}57 \text{ грн.}$$

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об'єктно-орієнтованим підходом, становить 10274,93 грн. Отже, сучасну комп'ютеру програму доцільно виконувати по об'єктно-орієнтованій парадигмі,

Усі резутати розрахунків, приведених у даному розділі, зведені у таблицю 3.7

Таблиця 3.7 – Результати розрахунків

№ п.п.	Назва	Процедурний підхід	Об'єктно-орієнтований підхід
1	2	3	4
1	Зарплата основна, грн	15600	16816
2	Зарплата додаткова, грн	3120	3363.2
3	Фонд заробітньої плати (1+2)	18720	20179.2
4	Обов'язкові відрахування на заробітну плату (4.1+4.2), грн	3480.9	3644.4
4.1	Утримання єдиного соціального внеску (3,6%), грн	674	726.5
4.2	Податок на доходи фізичних осіб (15%), грн	2806.9	2917.9
5	Військовий збір (1.5%), грн	280.8	302.7
6	Відрахування на ФОП (36,77%), грн	6883.34	7419.82
7	Разом на оплату праці (3+6), грн	25603.34	27599.02
8	Матеріальні витрати, грн	651	651
9	Електроенергія, грн	939.6	1009.2

Продовження таблиці 3.7

1	2	3	4
10	Амортизація, грн	4539.3	4875.52
11	Накладні витрати, грн	7800	8408
12	Разом на ін. витрати (8+9+10+11)	13929.9	14943.72
13	Собівартість, грн	38882.24	41891.74
14	Прибуток, грн	10498.20	11310.77
15	Вартість розробленого ПЗ, грн	49380.44	53202.51
16	Економічна ефективність	0.27	0.27
17	Термін окупності, років	3.7	3.7
18	Собівартість модернізації, грн	23329.35	8378.22
19	Супровід і модернізація, грн	29628.27	10640.50

Загальна вартість пропонованих робіт по розробці програмного продукту становить 79008,71 грн. для першого варіанту та 63843,01 грн. для другого. Оскільки ефективність для обидвох проєктів відповідно до встановленого рівня прибутку становить 0,27, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

При використанні об’єктно-орієнтовного підходу зменшується кількість працівників, які залучаються у проєкт, та зменшуються витрати на реалізацію проєкту, але для підтримки проєкту і його подальшої модернізації все ж в майбутньому потрібні набагато більші витрати. Для функціонального підходу потрібна більша кількість працівників, часу і коштів, але на модернізацію і підтримку в загальному потрібно менше ресурсів, і у висновку програма виконана по функціональній парадигмі стає дешевшою для споживача, і приносить економію ресурсів для розробників.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Методи аналізу виробничого травматизму

Аналіз виробничого травматизму проводиться з метою встановлення закономірностей виникнення травм на виробництві та розробки ефективних профілактичних заходів [26, 27].

У процесі аналізу травматизму мають бути з'ясовані причини нещасних випадків і розроблені заходи щодо їх попередження.

Для аналізу виробничого травматизму застосовують чотири основних методи: статистичний, монографічний, економічний, метод фізичного і математичного моделювання.

Статистичний метод ґрунтується на вивченні причин травматизму за документами, що реєструють нещасні випадки (акти за формою Н-1, листки тимчасової непрацездатності), за певний період часу (квартал, півріччя, рік); у випадку професійних захворювань аналізуються дані карт обліку професійних захворювань за формою П-5, які складаються на підставі актів розслідування випадків профзахворювань.

Цей метод створює можливість визначити порівняльну динаміку травматизму за окремими галузями, підприємствами, цехами, ділянками одного підприємства і виявити закономірності чи ділянки зниження або підвищення рівня травматизму. Для оцінки рівнів травматизму користуються відносними показниками (коефіцієнтами) частоти, важкості і втрат.

За коефіцієнт частоти травматизму $К_ч$ береться кількість нещасних випадків, що припадають на тисячу працівників за певний період:

$$К_ч = \frac{Т}{Р} 1000 \quad (4.1)$$

де Т – число нещасних випадків за звітний період (за винятком важких та смертельних);

Р – середньооблікова кількість працівників за той же період.

Коефіцієнт важкості травматизму $К_т$ характеризує середня кількість днів непрацездатності, що припадають на один нещасний випадок:

$$К_т = \frac{Д}{Т} \quad (4.2)$$

де Д - сумарна кількість днів непрацездатності за всіма нещасними випадками за звітний період.

За коефіцієнт втрат $К_в$ (показник загального травматизму) береться кількість людино-діб непрацездатності, що припадають на 1000 працівників. У ці показники не включаються групові та смертельні нещасні випадки:

$$К_в = К_ч * К_т = \frac{Д}{Р} * 1000 \quad (4.3)$$

Зміна коефіцієнтів частоти, важкості і втрат протягом ряду періодів характеризує динаміку промислового травматизму й ефективність заходів щодо попередження травматизму.

При поглибленому статистичному аналізі травматизму, крім виявлення причин травматизму, робиться також аналіз нещасних випадків за джерелами і характером впливу на організм; за видами робіт чи виробничими операціями; за характером травм; аналізуються відомості про потерпілих (професія, стаж, стать, вік), дані про час події (місяць, година робочого дня, зміна). Отримана інформація

орієнтує дослідників щодо небезпеки виробничої обстановки та питань розробки індивідуальних захисних засобів, дає змогу вжити попереджувальні заходи.

До різновидів статистичного аналізу відносять груповий і топографічний. Груповий метод аналізу травматизму ґрунтується на повторюваності нещасних випадків незалежно від тяжкості ушкоджень. Наявний матеріал розслідування розподіляється за групами з метою виявлення найчастіше повторюваних випадків (однакових за обставинами). Нещасні випадки групуються за окремими однорідними ознаками: видом робіт, обладнанням, кваліфікацією, спеціальністю, віком потерпілого, причинами нещасних випадків тощо.

Топографічний метод полягає у вивченні причин нещасних випадків щодо місця їх виникнення; ці місця систематично наносяться умовними знаками на плани ділянки, цеху, підприємства. Метод дає наочне уявлення про місця зосередження травматизму, які потребують відповідних профілактичних заходів. Статистичні методи дослідження дають загальну картину стану травматизму, установлюють його динаміку, виявляють певні залежності, але при цьому не вивчаються поглиблено умови, в яких стався нещасний випадок.

Монографічний метод включає детальне дослідження всього комплексу умов, у яких стався нещасний випадок: процеси, устаткування, матеріали, захисні засоби, умови виробничої обстановки та ін. У результаті дослідження виявляються не тільки причини нещасних випадків, а й приховані (потенційні) небезпечні та шкідливі фактори, що можуть призвести до травматизму.

Економічний метод полягає у визначенні економічного збитку від виробничого травматизму, а також в оцінці ефективності витрат,

що спрямовані на попередження нещасних випадків, з метою оптимального розподілу коштів на заходи щодо охорони праці.

Метод фізичного і математичного моделювання застосовується на складних зразках техніки.

Поряд із традиційними методами аналізу травматизму можна відзначити деякі нові напрямки, характерні для дослідження умов безпеки праці та попередження травматизму:

- комплекс методів математичної статистики, наприклад, методи дисперсійного і кореляційного аналізу;
- метод наукового прогнозування безпеки праці. Він служить для ймовірнісної оцінки динаміки травматизму, передбачення утворення несприятливих факторів у нових виробництвах чи технологіях і розробки для них відповідних вимог техніки безпеки;
- розробка автоматизованих систем оперативного обліку і попередження травматизму, що мають стати однією з ланок автоматизованої системи управління охороною праці;
- розробка методик комплексної оцінки безпеки технологічних процесів та устаткування на стадії їх проектування, виготовлення й експлуатації;
- ергономічний метод, що ґрунтується на комплексному вивченні систем "людина - машина - виробниче середовище" (ЛМС) з урахуванням функціональних можливостей людини у процесі праці;
- детерміністичні методи, які створюють можливість виявити об'єктивний закономірний взаємозв'язок умов праці й існуючу зумовленість випадків травматизму (наприклад, метод мережного моделювання застосовується при аналізі випадків травматизму, що стали результатом дії кількох факторів; методи

спостережень, анкетування встановлюють в основному причини психофізіологічного характеру; метод експертних оцінок дає змогу дійти висновків на підставі узагальненого досвіду та інтуїції фахівців, що займаються питаннями охорони праці). Для оперативного обліку та обробки інформації про травматизм і профзахворювання можуть бути використані ручні і машинні системи (ПЕОМ).

Прогнозування травматизму здійснюється звичайно з використанням статистичних даних щодо $К_ч$, $К_т$, $К_в$ за кілька років роботи, це створює можливість екстраполювати криву, що описує застосування зазначених показників, на найближчий календарний період. Прогнозування травматизму і професійних захворювань, а також динаміки зміни умов праці є однією з основ створення систем управління (менеджменту) охороною праці (СУОП).

Методи прогнозування помилок людини. Ці методи ґрунтуються на класичному аналізі, що містить у собі наступні етапи:

- складання переліку основних відмов системи ЛМС;
- складання переліку й аналізу дій людини;
- оцінювання частоти помилок людини;
- визначення впливу частоти помилок людини на інтенсивність відмов розглянутої системи;
- вироблення рекомендацій, внесення необхідних змін у розглянуту систему і обчислення нових значень інтенсивності відмов.

Надійність людини в системі «людина - машина - виробниче середовище» відіграє дуже важливу роль. Але в розглянутих причинах невиконання завдання враховуються не тільки помилки, а й безпомилкові дії, які, проте, призведуть до катастрофи або аварії в

силу тих чи інших обставин. При розгляді причин окремо слід звернути увагу на технічні причини, які, здавалося б, до людини стосунку не мають. Надійність машини визначається її справним станом, за яким наглядає людина.

Якщо людина в силу своєї некомпетентності чи безвідповідальності доводить технічний стан машини до такого, що супроводжується виходом машини з ладу, то першою причиною виходу машини з ладу буде ненадійність людини як фахівця, що обслуговує машину, а потім - технічна несправність. Ризик виходу машини з ладу в цьому випадку є перш за все функцією діяльності людини. Більш докладно ці питання розглядаються в курсі "Безпека життєдіяльності".

Метод дерева несправностей застосовується при аналізі складних систем. Загальна процедура аналізу дерева несправностей полягає у виконанні наступних етапів:

- визначення небажаної (завершальної) події в розглянутій системі;
- ретельне вивчення можливої поведінки і передбачуваного режиму використання системи;
- визначення функціональних властивостей подій вищого рівня для з'ясування причин тих чи інших несправностей системи і проведення більш глибокого аналізу поведінки системи з метою виявлення логічного взаємозв'язку подій нижчого рівня, здатних призвести до відмови системи;
- побудова дерева несправностей для логічно пов'язаних подій на вході. Ці події мають визначатися в термінах ідентифікованих незалежних первинних відмов.

Щоб одержати кількісні результати для завершальної небажаної події дерева, необхідно задати ймовірність відмови, коефіцієнт

готовності, інтенсивність відмов та інші показники, які характеризують первинні події, за умови, що події дерева несправностей не є надлишковими [26].

Більш точний і систематичний аналіз передбачає виконання таких процедур, як:

1) визначення границь системи;

2) побудова дерева несправностей;

3) якісна оцінка;

4) кількісна оцінка.

Приклад. Потрібно побудувати дерево несправностей для простої системи - освітлення робочого місця, у якій є вимикач та електрична лампочка (рис. 4.1). Вважається, що відмова вимикача полягає лише в тому, що він замикається, а завершальною подією є відсутність освітлення в кімнаті. Основними, або первинними, подіями дерева несправностей є:

1) відмова джерела живлення Е};

2) відмова запобіжника Е2;

3) відмова вимикача Е3;

4) перегоряння лампочки Е4.

При аналізі дерево несправностей показує, що первинні події — це входи схем ЧИ: при настанні кожної з чотирьох первинних подій Е., Е2, Е3, Е4 здійснюється завершальна подія (відсутність світла у робочому приміщені).

Переваги і хиби методу дерева несправностей наступні: метод дає уявлення про поведінку системи, але потребує від фахівців глибокого розуміння системи і конкретного розгляду щоразу тільки однієї певної відмови; допомагає дедуктивно виявляти відмови; дає конструкторам, користувачам і керівникам можливе наочне обґрунтування конструктивних змін та аналізу компромісних рішень;

створює можливість виконувати кількісний і якісний аналіз надійності; полегшує аналіз надійності складних систем. Але найголовніша його перевага - це те, що він розвиває навички логічного мислення, що дуже необхідно в практичній діяльності фахівців будь-якого рівня і роду занять.

Метод дуже добре себе зарекомендував при розслідуванні нещасних випадків, коли необхідно розглянути складне нагромадження різних причин у часі. Приклади використання цього методу при розслідуванні наведені у практикумі з «Охорони праці».

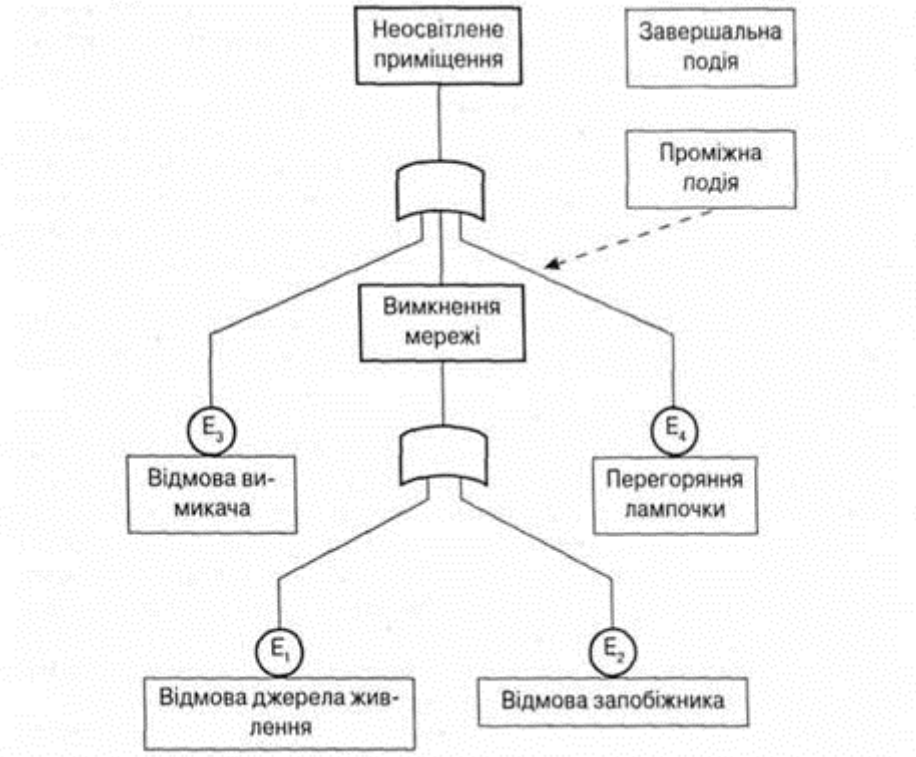

Рисунок 4.1 – Приклад побудування дерева несправностей для лампочки, що перегоріла

4.2 Пристрій автоматичного контролю пошкоджень ізоляції в електроустановках

Захист від небезпеки при переході з вищої напруги на нижчу. При пошкодженні ізоляції між обмотками вищої і нижчої напруг трансформатора виникає небезпека переходу напруги і, як наслідок, небезпека ураження людини, виникнення займання і пожеж. Способи захисту залежать від режиму нейтралі. Мережі напругою до 1000 В з ізольованою нейтраллю, сполучені через трансформатор з мережами напругою вище за 1000 В, мають бути захищені пробивним запобіжником, установленим у нейтралі чи фазі з боку нижчої напруги трансформатора. Тоді у випадку пошкодження ізоляції між обмотками вищої і нижчої напруг цей запобіжник пробивається і нейтраль або фаза нижчої напруги заземлюється. Напруга нейтралі щодо землі $U_з = I_з * R_0$. Заходом захисту є зниження цієї напруги до безпечного заземлення нейтралі з опором R0 < 4 Ом.

Пробивні запобіжники застосовуються, коли вища напруга є більшою за 1000 В. Якщо вища напруга буде нижчою за 1000 В, пробивний запобіжник не спрацює. Тому вторинні обмотки знижувальних трансформаторів для живлення ручного електроінструмента і ручних ламп малою напругою заземлюють.

Контроль і профілактика пошкоджень ізоляції. Профілактика пошкоджень ізоляції спрямована на забезпечення її надійної роботи. Насамперед необхідно виключити механічні пошкодження, зволоження, хімічний вплив, запилення, перегріви. Але навіть у нормальних умовах ізоляція поступово втрачає свої початкові властивості, "старіє". З часом розвиваються місцеві дефекти. Опір ізоляції починає різко зменшуватися, а струм витоку - непропорційно зростати. У місці дефекту з'являються часткові розряди струму,

ізоляція вигорає. Відбувається так званий пробій ізоляції, внаслідок чого виникає коротке замикання, що, у свою чергу, може спричинити пожежу чи ураження людей струмом.

Щоб підтримувати діелектричні властивості ізоляції, необхідно систематично виконувати профілактичні випробування, огляди, видаляти непридатну ізоляцію і заміняти її.

Періодично в приміщеннях без підвищеної небезпеки не рідше одного разу на два роки, а в небезпечних приміщеннях - кожні півроку перевіряють відповідність опору ізоляції нормі. При виявленні дефектів ізоляції, а також після монтажу мережі, її ремонту на окремих ділянках, відключення мережі між кожним проводом і землею та між проводами різних фаз проводять вимірювання. При цьому в силових колах відключають електричні приймачі, апарати, прилади; в освітлювальних - відгвинчують лампи, а штепсельні розетки, вимикачі та групові щитки залишають приєднаними. Перед початком вимірювань необхідно переконатися в тому, що на досліджуваній ділянці мережі (між двома запобіжниками або за останнім запобіжником) або на устаткуванні ніхто не працює і воно відключене. Кабелі, шини, електричні машини, повітряні лінії, конденсатори "розряджають на землю", тобто торкаються заземленим проводом відключених струмопровідних частин кожної фази, знімаючи залишковий ємнісний заряд. Значення виміряного опору ізоляції має бути не нижчим за норму, зазначену в ПУЕ (не менше 0,5 МОм/фазу ділянки мережі напругою до 1000 В).

Для вимірювання використовують прилад - мегаомметр на напруги 500, 1000, 2500 В з межами вимірів 0-100, 0-1000, 0-10000 МОм. Щоб мати уявлення ще й про опір ізоляції всієї мережі, вимірювання потрібно проводити під робочою напругою з підключеними споживачами. Такий контроль можливий тільки в

мережах з ізольованою нейтраллю (у мережі з заземленою нейтраллю постійний струм приладу контролю ізоляції замикається через заземлення нейтралі, і мегаомметр показуватиме нуль).

Застосовується також постійний (безперервний) контроль ізоляції - вимірювання опору ізоляції під робочою напругою протягом усього часу роботи електроустановки без автоматичного відключення. Відлік опору ізоляції здійснюється за шкалою приладу. При зниженні опору ізоляції до гранично допустимого чи нижче, прилад подає звуковий або світловий сигнал або обидва сигнали разом. З вітчизняних приладів контролю ізоляції найбільшого поширення одержали ПКІ, РУВ, УАКІ, М-143, МКН-380, Ф-419. Найпростішим засобом контролю ізоляції є вольтметр. В установках напругою до 1000 В вольтметри підключають безпосередньо до фаз, а в установках з напругою понад 1000 В - через вимірювальний трансформатор.

На підприємствах широко застосовується випробування ізоляції підвищеною напругою. Цей метод є найбільш ефективним для виявлення місцевих дефектів ізоляції і визначення її міцності, тобто здатності довгостроково витримувати робочу напругу. Електричні машини й апарати випробовують струмом промислової частоти, як правило, протягом 1 хв. Подальша дія струму може вплинути на якість ізоляції. Значення випробної напруги нормується залежно від номінальної напруги електроустановки і виду ізоляції.

Захист від випадкового дотику до струмопровідних частин. Щоб виключити можливість дотику або небезпечного наближення до відкритих струмопровідних частин, слід забезпечити недоступність за допомогою захисних засобів, огорож, блокувань чи розташування струмопровідних частин на недоступній висоті в недоступному місці. Огорожі бувають як суцільні, так і сітчасті (сітка 25х25 мм). Суцільні огорожі у вигляді кожухів і кришок використовують для

електроустановок напругою до 1000 В. Сітчасті огорожі застосовують в установках напругою до 1000 В і вище.

За допомогою блокувань захищають електроустановки напругою понад 250 В, у яких часто виконують роботи на необгороджених струмопровідних частинах. Блокування забезпечує зняття напруги зі струмопровідних частин електроустановок при проникненні до них без зняття напруги. За принципом дії блокування поділяють на механічні, електричні й електромагнітні. Електричні блокування розривають коло контактами, встановленими на дверях огорож, кришках і дверцятах кожухів. Механічні блокування застосовують в електричних апаратах (рубильниках, пускачах, автоматах). В апаратурі автоматики, обчислювальних машинах і радіоустановках використовують блокові схеми: коли блок висувається або віддаляється зі свого місця, штепсельне рознімання розмикається. Таким чином, блок відключається автоматично при відкриванні його струмопровідних частин. Використання блокувань є також доцільним для попередження помилкових дій персоналу при переключеннях у розподільних пристроях і на підстанціях.

Для захисту від дотику до частин, що перебувають під напругою, застосовується подвійна ізоляція — електрична ізоляція, що складається з робочої і додаткової. Робоча ізоляція — ізоляція струмопровідних частин електроустановки. Додаткова ізоляція виконується виготовленням корпусу з ізолюючого матеріалу (електропобутові прилади).

Компенсація ємностей складової струму замикання на землю. Струм замикання на землю, як і струм крізь людину в мережі з ізольованою нейтраллю, залежить не тільки від опору ізоляції, а й від ємності мережі щодо землі. Контроль і профілактика пошкоджень ізоляції дають змогу підтримувати її опір на високому рівні. Ємність

фаз щодо землі не залежить від будь-яких дефектів; вона визначається загальною довжиною мережі, висотою підвісу проводів повітряної мережі, товщиною фазної ізоляції живильного кабелю, тобто геометричними параметрами. Тому ємність мережі не може бути знижена. У процесі експлуатації ємність мережі змінюється лише за рахунок відключення і включення окремих ліній, що визначається потребами електропостачання [27].

Оскільки неможливо зменшити ємність мережі, зниження струму замикання на землю досягається шляхом компенсації його ємнісної складової індуктивністю. При цьому компенсаційна котушка включається між нейтраллю і землею, як показано на рис. 4.2. При замиканні на землю в трьохпроводовій мережі з ізольованою нейтраллю струм проходить через перехідний опір r' (провідність g') і далі через опір ізоляції двох інших фаз R_b та R_c (провідності і g_c) і паралельно крізь ємності C_b і C_c(провідності b_b і b_c). Цей струм має дві складові - активну I_r й ємнісну I_c(рис. 4.2 б). На векторній діаграмі показано суму струмів до (рис. 4.2 б) і після (рис. 4.2 в) компенсації.

До активної і ємнісної складових струму замикання на землю додаються активний та індуктивний струми компенсаційної котушки (наявність активної складової пояснюється активними втратами в котушці). Ємнісна й індуктивна складові перебувають у протифазі і при настроюванні в резонанс взаємно знищують одна одну. Активні складові складаються, тобто струм замикання на землю $I_{зк} = I_r + I_{ка}$і стає значно меншим, ніж до компенсації (тут $I_{ка}$ – активний струм компенсаційної котушки). У разі неповної компенсації ємності може бути деяка ємнісна складова струму замикання на землю (при

недокомпенсації); індуктивна – при перекомпенсації. Проте в обох випадках струм замикання на землю знижується.

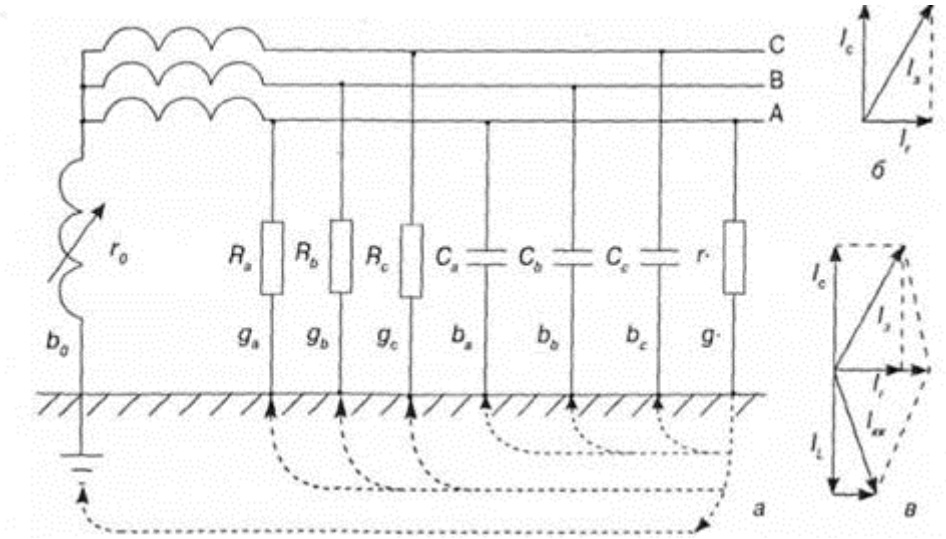

Рисунок 4.2 – Компенсація ємнісної складової струму замикання на землю а – принципова схема; б,в – векторні діаграми струму замикання на землю до і після компенсації

Компенсаційні котушки іноді називають дугогасними, оскільки, зменшуючи струм замикання на землю, вони сприяють гасінню дуги між струмопровіднйми і заземленими частинами і тим самим ліквідації пошкодження, тобто сприяють замиканню на землю. Цей захист застосовується як доповнення до захисного відключення або заземлення.

Захисне заземлення, занулення і захисне відключення. Однофазові замикання струму, які можуть виникнути в електричних машинах, апаратах, приладах, на ЛБП, небезпечні тим, що на корпусах та опорах з'являються напруги, достатні для ураження людини і виникнення пожежі. Струм замикання створює небезпечні

напруги не тільки на самому устаткуванні, а й поблизу нього, розповсюджуючись через основи і фундаменти.

Захист від ураження електричним струмом і загорянь можна здійснити захисним відключенням (відключають пошкоджені ділянки мережі швидкодіючим захистом), або захисним заземленням (знижують напруги дотику і кроку), або зануленням (відключають устаткування і знижують напруги дотику і кроку на період, доки не спрацює апарат, що відключає).

ВИСНОВКИ

Суть роботи полягає у створенні програмного продукту, реалізація якого дасть змогу створювати, формувати і розвивати концептуально нове сучасне інтерактивне середовище соціалізації від ефективності функціонування якого залежать успішність, освоєння надбань, ідей, досвіду, дотримання норм та стандартів.

Для розробки програмного продукту було використано сучасні інформаційні технології, зокрема JavaScript – мову програмування, яка найбільш широко використовується в браузерах для надання їх інтерактивності. Зручність використання її полягає в гнучкості, підтримці більшістю найбільш популярних браузерів, високій швидкодії, широкому виборі налаштувань та корисного функціоналу та безперервному удосконаленню за рахунок популярності. За інфраструктуру програмних рішень взято MeteorJS, оскільки це доступний та, що не менш важливо, відкритий веб-фреймворк, написаний мовою JavaScript, в якому використовується програмна платформа Node.js. Перевагу надано фреймворку MeteorJS через те, що він дозволяє швидко створювати крос-платформові застосунки (веб, Android, IOS) code, а також те, що його можна інтегрувати з MongoDB – швидко доступною, легко масштабованою базою даних зі зрозумілою структурою кожного об’єкту. Перевагою даної бази даних є й те, що вона підтримує динамічні запити документів та не характеризується складними запитами. Цікавим є й те, що на відміну від реляційної бази даних, що має стандартну схему, яка відображає кількість таблиць та зав’язків між ними MongoDB такої схеми не має.

У першому розділі дипломної роботи описано предметну область – всі задачі з якими стикаються користувачі в роботі з подібними засобами комунікації. Розглянуто основних користувачів

системи, їх ролі та варіанти використання – на основі цих варіантів використання було створено систему, яка виконує відповідні функції. Проаналізовано вибір бази даних, мови програмування та фреймворку, розглянуто архітектуру системи, описано основні технології, які використовувалися, а також реалізацію основних методів та особливостей роботи системи.

У другому розділі проілюстровано роботу системи, розглянуто її інтерфейс та роботу основних функцій, проведено тестування системи при введенні різних даних.

У третьому розділі розглянуто підходи до розробки даного продукту, а саме об’єктно-орієнтований і процедурний. Проаналізовано усі витрати і прибутки, потрібні складові для роботи, супровід і модернізацію програмного продукту та уточнено аналіз економічної ефективності вкладених інвестицій, обчислено різницю між двома підходами щодо витрат на виробництво і модернізацію.

В останньому розділі описано заходи щодо профілактики нещасних випадків на виробництві, шляхи вирішення, а також проілюстровано схеми пристроїв захисного відключення.

Отже, в роботі встановлено актуальність роботи, яка полягає у створенні концептуально нової соціальної мережі, яка буде хорошим інструментом соціалізації і об’єднувати людей зі спільними інтересами.

Досягнуто мети роботи, яка полягає в створенні соціальної мережі в основному для підростаючого покоління.

Робота має практичне застосування, оскільки розроблене програмне забезпечення призначене для всіх потенційних користувачів продуктів такого типу, дасть змогу ефективно зберегти принцип організації системи, коли мета досягається інформаційним обміном між всіма елементами цієї взаємодіючої системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання магістерської роботи освітнього рівня ―магістр‖ студентами усіх форм навчання для напряму підготовки 121 – ―Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.

2. Яцишин, А.Band Коваленко, В.В. (2015) Використання електронних соціальних мереж для роботи з дітьми та молоддю з особливими освітніми потребами Освіта та розвиток обдарованої особистості, 8 (39). стор. 32-38.

3. Алмаева В. В. Виртуальные социальные сети как составляющая современного образовательного пространства [Электроный ресурс] / В. В. Алмаева. — Режим доступа : http://ito.edu.ru/2010/Tomsk/III/III-0-1.html

4. Бакулев Г. П. Массовая коммуникация: Западные теории и концепции [Электроный ресурс] / Г. П. Бакулев. — Режим доступа : http://pulib.if.ua/part/9691

5. Белинская Е. Современные исследования виртуальной коммуникации: проблемы, гипотезы, результаты / Е. Белинская, А. Жичкина. — М. : ЮНИТИ, 2009.— 165 с.

6. Войскунский А. Е. Феномен зависимости от Интернета / А. Е. Войскунский // Гуманитарные исследования в Интернете. — М. : Изд-во Эксмо, 2000. — 321 с.

7. Янг К. С. Диагноз интернетзависимость / К. С. Янг // Мир Интернет. — 2000.— № 2. — С. 24—29.

8. Психологія соціалізації [Електронний ресурс] – Режим доступа : URL : http://subject.com.ua/psychology/variy66 – Загл. с экрана.

9. Горелов И.Н. Соотношение вербального и невербального в коммуникативной деятельности // Исследования речевого мышления в психолингвистике. – М., 1985. – С.116-150.

10. Бацевич Ф.С. Основи комунікативної лінгвістики: Підручник. – К., 2004

11. Сазанов В. М. Социальные сети – анализ и перспективы [Электроный ресурс] / В. М. Сазанов. — Режим доступа : http://spkurdyumov.narod.ru/sazonov.htm

12. Основи баз даних [Електронний ресурс] – Режим доступа : URL : http://programing.in.ua/databases/ – Загл. с экрана.

13. Структура сайту [Електронний ресурс] – Режим доступа : URL : http://www.welcomseo.ru/blog/struktura-sayta-osnovnye-vidy-struktur-sovremennykh-saytov.htm – Загл. с экрана.

14. Шаблони в MeteorJS [Електронний ресурс] – Режим доступа : URL : http://ru.discovermeteor.com/chapters/templates/ – Загл. с экрана.

15. Реактивність Meteor [Електронний ресурс] – Режим доступа : URL : http://ru.discovermeteor.com/pdf – Загл. с экрана.

16. MVVM [Електронний ресурс] – Режим доступа : URL : https://ru.wikipedia.org/wiki/Model-View-ViewModel – Загл. с экрана.

17. Шаблони в MeteorJS [Електронний ресурс] – Режим доступа : URL : http://ru.discovermeteor.com/chapters/templates/ – Загл. с экрана.

18. Структура meteor застосунка [Електронний ресурс] – Режим доступа : URL : http://ru.discovermeteor.com/chapters/getting-started/ – Загл. с экрана.

19. Iron-router [Електронний ресурс] – Режим доступа : URL : https://github.com/iron-meteor/iron-router – Загл. с экрана.

20. DOM [Електронний ресурс] – Режим доступа : URL : http://frontender.info/dom/ – Загл. с экрана.

21. Прості застосунки на MeteorJS [Електронний ресурс] – Режим доступа : URL : http://www.ibm.com/developerworks/ru/library/wa-meteor/ – Загл. с экрана.

22. Couchbase [Електронний ресурс] – Режим доступа : URL : http:// couchbase.com – Загл. с экрана.

23. CouchDB [Електронний ресурс] – Режим доступа : URL : http://couchdb.apache.org – Загл. с экрана.

24. MongDB [Електронний ресурс] – Режим доступа : URL : http://mongodb.org – Загл. с экрана.

25. Опис Bootstrap [Електронний ресурс] – Режим доступа : URL : https://ru.wikipedia.org/wiki/Twitter_Bootstrap – Загл. с экрана.

26. Гандзюк М.П.. Желібо Є.П.. Халімовський М.О. Основи охорони праці: Підручник для студентів вищих навчальних закладів / За ред. М.П. Гендзюка. - К.: Каравела. 2003. - 408 с.

27. Основні технічні заходи захисту в електроустановках [Електронний ресурс] – Режим доступа : URL : http://www.pidruchniki.com/14360106/bzhd – Загл. с экрана.

ДОДАТКИ

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра “ Програмної інженерії ”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПІ
“___“ _____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання дипломної роботи
на здобуття освітнього ступеня «магістр»
за спеціальністю 121 – «Інженерія програмного забезпечення» на тему

«РОЗРОБКА АВТОМАТИЗОВАНОГО ІНТЕРАКТИВНОГО СЕРЕДОВИЩА СОЦІАЛІЗАЦІЇ МОВОЮ ПРОГРАМУВАННЯ JAVA SCRIPT, ФРЕЙМВОРК METEOR JS»

Керівник роботи
д.т.н., проф., проф. каф. ПІ Пастух О.А.
“_” _____ 2019 р.
_____ /підпис/

Виконавець
студент групи СПд-2 Тишко Н.І.
“___” _____ 2019 р.
_____ /підпис/

Тернопіль 2019

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Розробка автоматизованого інтерактивного середовища соціалізації мовою програмування Java Script, фреймворк Meteor JS.

1.2 Область застосування – на прикладі інтерактивного середовища.

2 ОСНОВА ДЛЯ РОЗРОБКИ

Основою для розробки є завдання на дипломну роботу, попередньо видане та затверджене кафедрою програмної інженерії факультету комп'ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою роботи є реалізація концепції мінімалізму, яка полягає у створені продукту призначеного лише для зручного та ефективності обміну даними, без перевантаження додатковими функціями, з врахуванням потреб та дотриманням вимог з юридичної сторони питання.

Розроблене програмне забезпечення призначене для всіх потенційних користувачів інтерактивного середовища, дасть змогу ефективно зберегти принцип організації системи, коли мета досягається інформаційним обміном між всіма елементами цієї взаємодіючої системи, з допомогою яких може відбуватися ефективна взаємодія з іншим користувачем чи системою.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами даної розробки є технічна документація, існуючі проґрамні та проґрамні системи, довідники та довідкові системи.

5 ЗАДАЧІ РОЗРОБКИ

5.1 Аналіз предметної області з метою вибору існуючих або розробки нових (удосконалення існуючих) моделей, що описують предметну область, математична постановка задач та вибір моделей їх розв'язку.

5.2 Вибір та обґрунтування вибору архітектури, вибір проґрамного середовища та необхідних додаткових бібліотек для реалізації проґрамної системи, вибір зовнішніх проґрамних та апаратних засобів для забезпечення роботи проґрамної системи.

5.3 Аналіз та уточнення вимог технічного завдання з точки зору обраних моделей, методів, алгоритмів та середовища розробки.

5.4 Проєктування, реалізація та тестування окремих компонент проґрамної системи, розробка принципів взаємозв'язку між ними.

5.5 Тестування проґрамного комплексу на навантаження.

5.6. Організаційно-економічні розрахунки.

5.7. Аналіз роботи в аспекті дотримання вимог положення з Охорони праці та безпеки в надзвичайних ситуаціях.

5.8 Оформлення супровідної документації.

5.9 Здача роботи.

6 ВИМОГИ ДО ПРОҐРАМНОЇ СИСТЕМИ

6.1 Функціональні вимоги

6.1.1 Програмне забезпечення має виконувати наступні дії:

1) реєстрація нового користувача;
2) авторизація користувача;
3) відображення кімнат;
4) відображення профілю користувача;
5) надсилання, редагування і видалення постів(публічних повідомлень);
6) надсилання, редагування і видалення приватних повідомлень.

6.2 Склад та параметри технічних засобів

1) ПК x64, з 512 Мб оперативної пам‘яті, встановленою системою Windows XP, Vista, Seven, 8, 8.1, 10, MacOS. Не менше 200 Мб вільного місця на жорсткому диску. Процесор з тактовою частотою від 1.2 GHz і більше.
2) наявність інтернет-оглядача.

6.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в різних операційних системах (Linux, Windows, MacOS. Програмне забезпечення розробляється з використанням бібліотек та технологій мови JavaScript в середовищі програмування WebStorm 10.0.0.4 з використанням фреймворку MeteorJS.

6.4 Вимоги експлуатації

6.4.1Кліматичні вимоги до експлуатації, при яких забезпечується робота проґрами повинні відповідати кліматичним умовам експлуатації наявних технічних засобів.

6.4.2 Вимоги до кваліфікації та численності персоналу. Мінімальна кількість користувачів, необхідної для роботи проґрамної системи, може складати одну одиницю.

7 ВИМОГИ ОХОРОНИ ПРАЦІ

В четвертому розділі “Охорона праці та безпека в надзвичайних ситуаціях” дипломної роботи необхідно провести аналіз виробничого травматизму, який проводиться з метою встановлення закономірностей виникнення травм на виробництві та розробки ефективних профілактичних заходів.

8 ПОРЯДОК КОНТРОЛЮ І ПРИЙОМУ

8.1 Представлення дипломної роботи до попереднього захисту

8.2 Представлення дипломної роботи до захисту

УДК 004.41
І.Н.Тишко, О.А.Пастух, д. т. н., професор
(Тернопільський національний технічний університет імені Івана Пулюя)

РОЗРОБКА ІНТЕРАКТИВНОГО СЕРЕДОВИЩА СОЦІАЛІЗАЦІЇ МОВОЮ ПРОГРАМУВАННЯ JAVA SCRIPT, ФРЕЙМВОРК METEOR JS

Сьогодення диктує свої умови і однією з ознак сучасності є інтерактивність між об'єктами різного характеру та ступеню, з особливостями обумовленими областю, напрямком та специфікою діяльності, зокрема в інформатиці, системах комунікацій, теорії інформації та інших. Не виключенням стало і програмування – один із напрямків, який найбільш динамічно розвивається та в якому збережено принцип організації системи, коли мета досягається інформаційним обміном між всіма її елементами, з допомогою яких може відбуватися ефективна взаємодія з іншим користувачем чи системою.

Актуальною задачею на сьогодні все ще залишається вирішення проблеми безпечного обміну даними, зокрема з однодумцями та за спільними інтересами не лише на побутовому рівні а, що найбільш суттєво, спрощення комунікації при реалізації якихось бізнес ідей чи з метою підвищення ефективності у будь-якому з напрямків діяльності. Подібні сервіси сьогодні не заслужено втратили свою колишню популярність і із-за цього мало розвиваються. Проте, ще не так давно вони активно використовувались в приватною метою. Сьогодні ж і взагалі зручніше та звичніше з цією метою використовувати соціальні мережі з вбудованими чатами.

Враховуючи тенденції та актуальність теми було сформульовано мету дослідження, яка полягає в реалізації концепції мінімалізму, яка полягає у створені продукту призначеного лише для надійного, безпечного, зручного та ефективного обміну даними, без перевантаження додатковими функціями, з врахуванням потреб та дотриманням вимог з юридичної сторони питання. Крім цього актуальність роботи полягає у створенні концептуально нового інтерактивного середовища соціалізації, яке буде хорошим інструментом в групуванні за спільними інтересами чи напрямками діяльності, не залучаючи всім відомі популярні соціальні мережі. Максимально інформативний, інтуїтивно зрозумілий та простий дизайн дозволить легко працювати та отримувати корисну інформацію.

Для розробки програмного продукту було використано сучасні інформаційні технології, зокрема JavaScript – мову програмування, яка найбільш широко використовується в браузерах для надання їх інтерактивності. Зручність використання її полягає в гнучкості, підтримці більшістю найбільш популярних браузерів, високій швидкодії, широкому виборі налаштувань та корисного функціоналу та безперервному удосконаленню за рахунок популярності. За інфраструктуру програмних рішень взято MeteorJS, оскільки це доступний та, що не менш важливо, відкритий веб-фреймворк, написаний мовою JavaScript, в якому використовується програмна платформа Node.js. Перевагу надано фреймворку MeteorJS через те, що він дозволяє швидко створювати крос-платформові застосунки (веб, Android, IOS) code, а також те, що його можна інтегрувати з MongoDB – швидко доступною, легко масштабованою базою даних зі зрозумілою структурою кожного об'єкту. Перевагою даної бази даних є й те, що вона підтримує динамічні запити документів та не характеризується складними запитами. Цікавим є й те, що на відміну від реляційної бази даних, що має стандартну схему, яка відображає кількість таблиць та зав'язків між ними MongoDB такої схеми не має.

ДОДАТОК В – ДИСК З ПРОҐРАМНИМ ЗАБЕЗПЕЧЕННЯМ