НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Кафедра Автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедрою
_____ Ролік О. І.
(підпис) (ініціали, прізвище)
«__»_____ 2019 р.

Магістерська дисертація

зі спеціальності 126 Інформаційні системи та технології

на тему: «Система колаборативного менеджменту контенту»

Виконав: студент 6-го курсу, групи ІА-82мп
(шифр групи)

Бобович Юрій Володимирович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н, доцент каф. АУТС, Новацький А. О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 126 Інформаційні технології та системи
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік О. І.
(підпис) (ініціали, прізвище)
«_» _____ 2018 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Бобовичу Юрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система колаборативного менеджменту контенту»

науковий керівник дисертації Новацький Анатолій Олександрович, к.т.н.,
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
доцент кафедри АУТС

затверджені наказом по університету від «___» _____ 2019 р. №___

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження система колаборативного менеджменту контенту

4. Предмет дослідження оптимізація роботи при великій навантаженості сторінки.

5. Перелік завдання, які потрібно розробити огляд існуючих рішень, порівняння алгоритмів, модель системи, графічний матеріал

6. Орієнтовний перелік ілюстративного (графічного) матеріалу: схема структурна, ER-діаграма, діаграма класів, діаграма прецедентів, приклади роботи алгоритмів та системи

7. Орієнтовний перелік публікацій: «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення»,

8. Консультанти розділів дисертації

9. Дата видачі завдання – 02.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
1.	Огляд існуючих рішень	2.09.2019 р.	
2.	Опис та математична модель алгоритмів	14.09.2019 р.	
3.	Модель для порівняння алгоритмів	28.09.2019 р.	
4.	Аналіз результатів та вибір архітектури	1.10.2019 р.	
5.	Розроблення моделі системи	15.10.2019 р.	
6.	Аналіз отриманих результатів	31.10.2019 р.	
7.	Розроблення схеми структурної	5.11.2019 р.	
8.	Розроблення схеми функціональної	10.11.2019 р.	
9.	Розробка стартап – проекту	15.11.2019 р.	
10	Оформлення текстової документації	28.11.2019 р.	

Студент _____ Бобович Ю.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Новацький А. О.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація на здобуття ступеня магістру на тему «Система колаборативного менеджменту контенту»: 100с., 47 рис., 40 табл., 9 додатки, 32 джерел.

Об'єкт дослідження - система колаборативного менеджменту контенту.

Мета роботи - оптимізація роботи застосунку з великою кількістю компонентів.

У магістерській дисертації розглядається проблема колаборативного менеджменту великої кількості даних у неформальному вигляді. Пропонується концептуальний підхід до її вирішення на основі аналізу особливостей побудови web-застосунків та рішення основних недоліків існуючих систем. Важливою особливістю є можливість кластеризації та розбиття на шари даних, що усуває проблему поганої роботи системи при роботі користувача з великою кількістю даних одночасно.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОЛАБОРАТИВНИЙ МЕНЕДЖМЕНТ, ВЕБ-ЗАСТОСУНОК, БАЗА ДАНИХ, КЛАСТЕРИЗАЦІЯ

Master's thesis for master's degree on the topic "Collaborative content management system": 100p, 47 figures, 40 tables, 9 annexes, 32 sources.

Object of study – collaborative content management system.

The purpose of the work – optimization of the application with many components.

The master's thesis deals with the problem of collaborative management of a large amount of informal information. A conceptual approach to its solution based on analysis is proposed features of building web-applications and solving the main disadvantages of existing systems. An important feature is the ability to cluster and split into layers of data, which eliminates the problem of poor system performance when the user is running a large amount of data at the same time.

SOFTWARE, COLLABORATIVE MANAGEMENT, WEB APPLICATION, DATABASE, CLUSTERING

ЗМІСТ

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

Віджет – контент-модуль, що вбудовується у веб-сторінку або у браузер.

Iframe – це код, який використовується для вбудовування інтерактивних медіа.

Канвас – елемент HTML5, який можна застосовувати для малювання графіки використовуючи скрипти

Спринт – часовий відрізок протягом якого створюється «готовий», тобто придатний до використання і випуску інкремент продукту.

Spa – це веб-застосунок чи веб-сайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

Бандл – це сукупність будь-яких програмних даних (файлів), об'єднаних за будь-якою ознакою

DBSCAN – Заснована на щільності просторова кластеризация для застосунків з шумами

OPTICS – Впорядкування точок для виявлення кластерної структури

LINQ – Language Integrated Query

REST – Representational State Transfer

DOM – Document Object Model

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

PWN – Progressive Web App

SOA – сервісно орієнтована архітектура

Творчій простір – унікальна сторінка з заповненим данимми canvas елментон для менеджменту контенту

Клстер – елмент творчого простору, який замыняэ собою групу елментів.

Кластеризація – це технологія машинного навчання, яка включає групування точок даних

Azure - це відкрита, гнучка корпоративна платформа хмарних обчислень

IoC – Inversion of Control

ORM – Object-relational mapping

SaaS – Software as a service

СУБД – система управління базами даних

API - application programming interface

MVC - Model-View-Controller

S.O.L.I.D. – single responsibility, open-closed, Liskov substitution, interface segregation, dependency inversion

SOA – сервісно орієнтована архітектура

ВСТУП

Актуальність теми колаборативного менеджменту контенту обумовлена зростанням чисельності людей які працюють віддалено, що призводить до потреби у створенні нових рішень для комунікації між членами команд. Також при зростанні чисельності проектів та людей заохочених до цих проектів виникають ситуації коли користувачам для використання застосунків потрібно завантажити велику кількість даних при використанні застосунків. Потреба у відображенні великої кількості даних на веб-сторінці призводить до нового класу проблем, а саме збереженням швидкості завантаження та швидкодії застосунку.

Об’єктом дослідження застосунки для колаборативного менеджменту контенту.

Предметом дослідження є методи оптимізації відображення даних веб-застосунків.

Метою є оптимізація застосунків для колаборативного менеджменту контенту з великою кількістю елементів.

Для досягнення мети магістерської дисертації було поставлено наступні задачі:

- оптимізація великої кількості елементів;
- оптимізація займаного простору на девайсі користувача;
- пришвидшення завантажування сторінки користувача з великою кількістю елементів.

В рамках магістерської дисертації дослідження предметної області, інструментів, проблем проводилось з використанням загальнонаукових та спеціальних методів, а саме:

– метод моделювання – розробка концепції на ранніх стадіях створення системи;

– аналіз та синтез – проведення дослідження існуючих варіантів синхронізації, створення нових рішень та розробка програмного комплексу з використанням високотехнічних засобів;

– метод оптимізації даних – модифікація системи для поліпшення її ефективності за рахунок кластеризації, розділення та часткового збереження даних.

– метод порівняння – огляд схожих варіантів реалізації синхронізації, порівняння різних алгоритмів знаходження коефіцієнтів кореляції, що дало змогу краще дослідити предметну область та розібратись в технічних вимогах до програмного комплексу.

Основним здобутком дисертації є методологія оптимізації даних основана на принципах розбиття даних на модулі, оптимізації canvas елементів та розбиття даних на шари, що дозволяє при зростанні даних залишати швидкість завантаження, займаний простів застосунку його швидкодії на сталих величинах. Дана система може бути використана як один із інструментів для взаємодії команд в їх повсякденній роботі.

Отримана в результаті виконаної дослідницької роботи методологія оптимізації були опубліковані у статті «Методи оптимізації html5 canvas» в сучасному науковому журналі «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» в 2019 році.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Понад дві третіх людей у всьому світі принаймні один раз на тиждень працюють поза офісом, вважають дослідники[1].

У дослідженні, опублікованому Цугом[2], представником офісу IWG, що базується у Швейцарії, встановлено, що 70 відсотків професіоналів працюють віддалено принаймні один день на тиждень, тоді як 53 відсотки працюють віддалено протягом половини тижня.

З 2014 року додаткові 4 мільйони робітників приєдналися до економіки позаштатної роботи, яка зараз становить 35% робочої сили в США. І чим молодший робітник, тим більше шансів на те, що вони на фрілансі. Згідно з дослідженням 29% робітників бебі-бумерів на фрілансі, 31% покоління X, 40% мілленіалів і 53% покоління Z.

Компанії завжди підстроюються під нові тренди та тенденції[3]. Так на початку 90-х повсюди впроваджувався електронний документообіг, а на початку цього десятиріччя новим трендом стали хмарні технології та рішення.

Так само зі збільшенням кількості позаштатних робітників повинні змінитися підходи та практики ведення роботи. Таким чином вже зараз інструменти які доповнюють та розширюють можливості робітника стали повсюдним стандартом:

1) телекомунікаційні застосунки, які спеціалізуються на наданні функціоналу відео чату, голосових дзвінків та месенджерів між комп'ютерами, планшетами та мобільними пристроями такі як Slack, Teams (Рисунок 1.1), Hangouts, Skype розширили можливість спілкуватися з колегами;

2) хмарні сховища даних які дозволяють користувачам зберігати файли на своїх серверах, синхронізувати файли на різних пристроях та обмінюватися файлами замінили фізичні копії документів. Приклад цього класу застосунків є Google Drive (Рисунок 1.2) або DropBox;

3) комплексне рішення управління ProofHub, Tfs, Jira, Trello (Рисунок 1.3) вивели на новий рівень можливості по формальному менеджменту проектів.

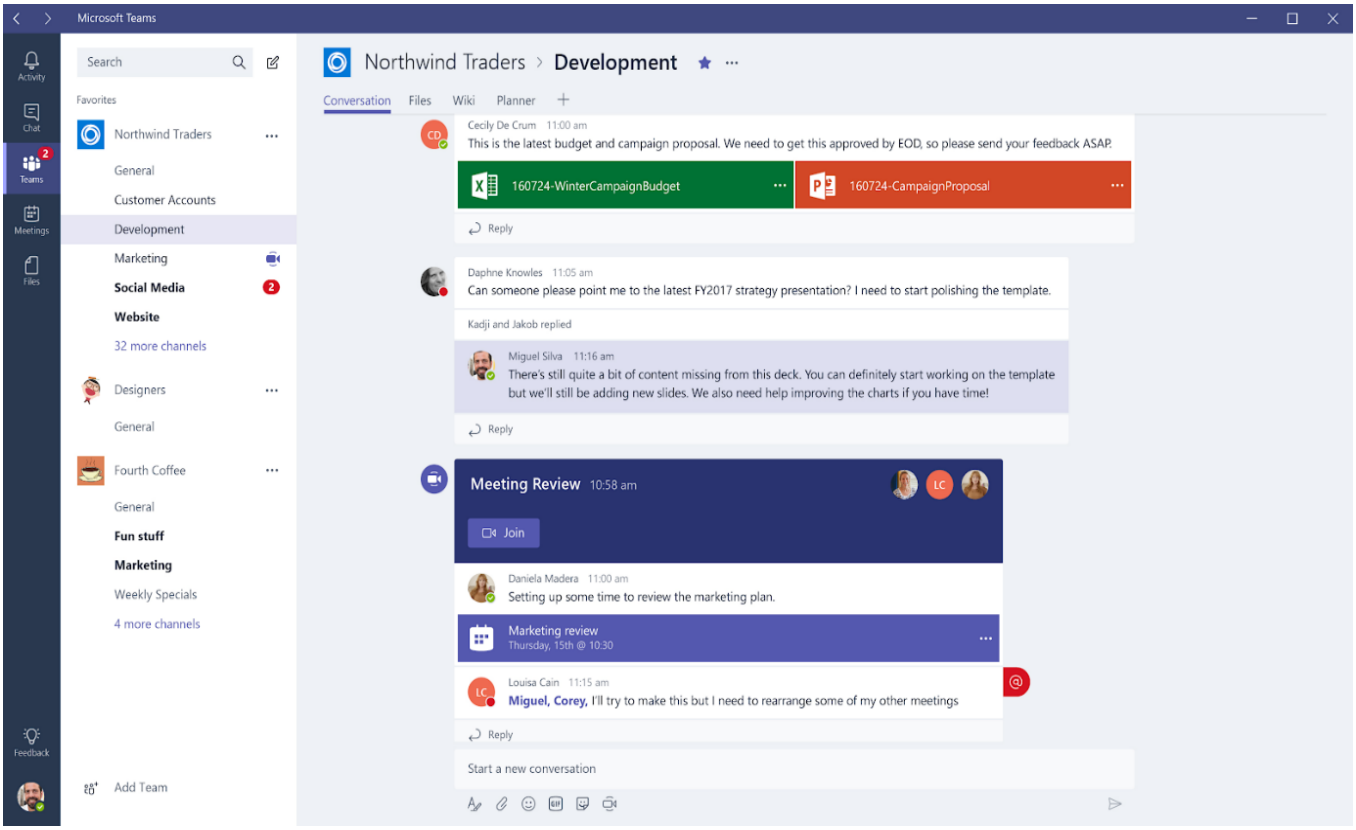

Рисунок 1.1 – Вигляд застосунку Teams[4].

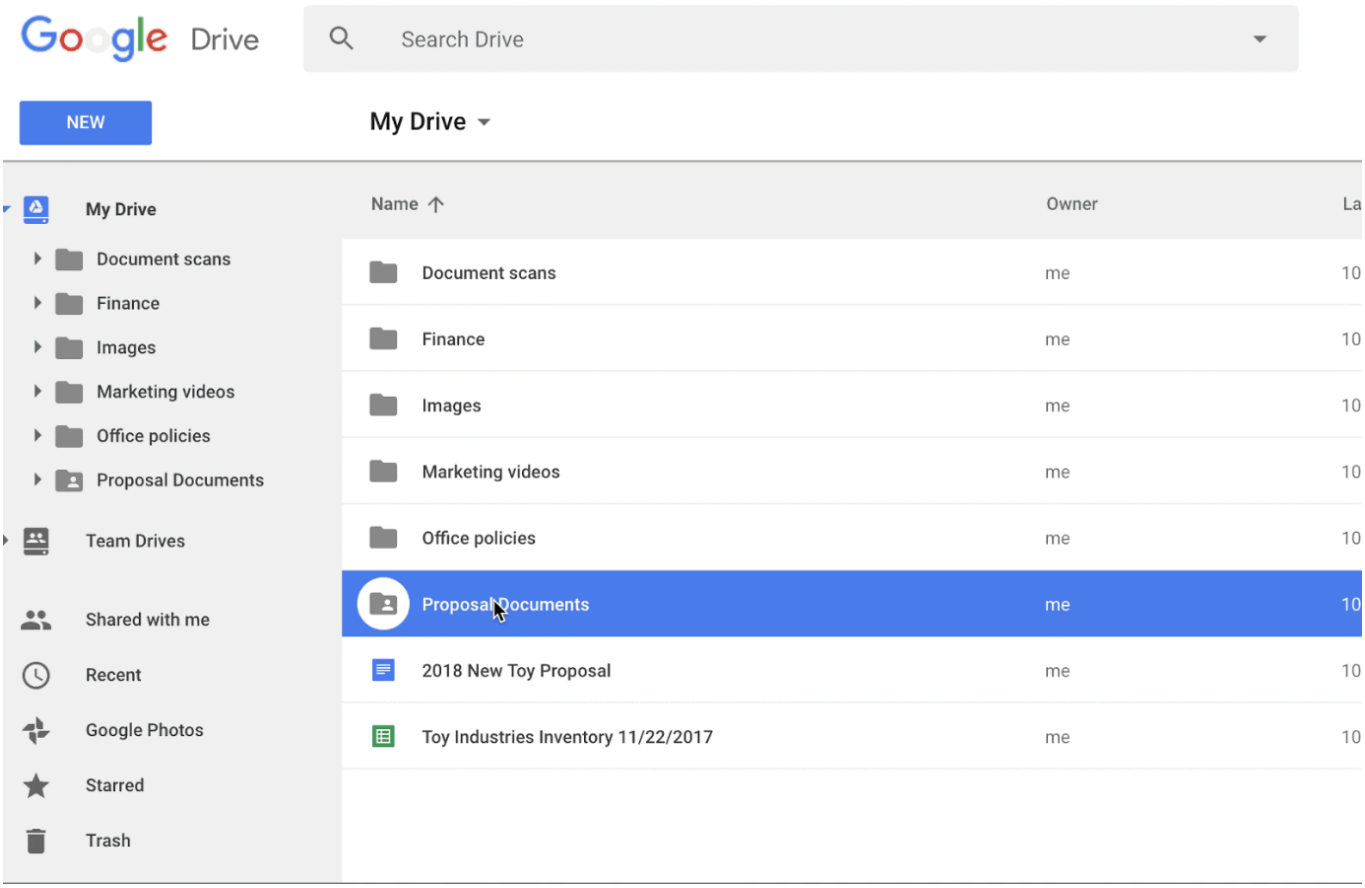

Рисунок 1.2 – Вигляд застосунку Google Drive[5].

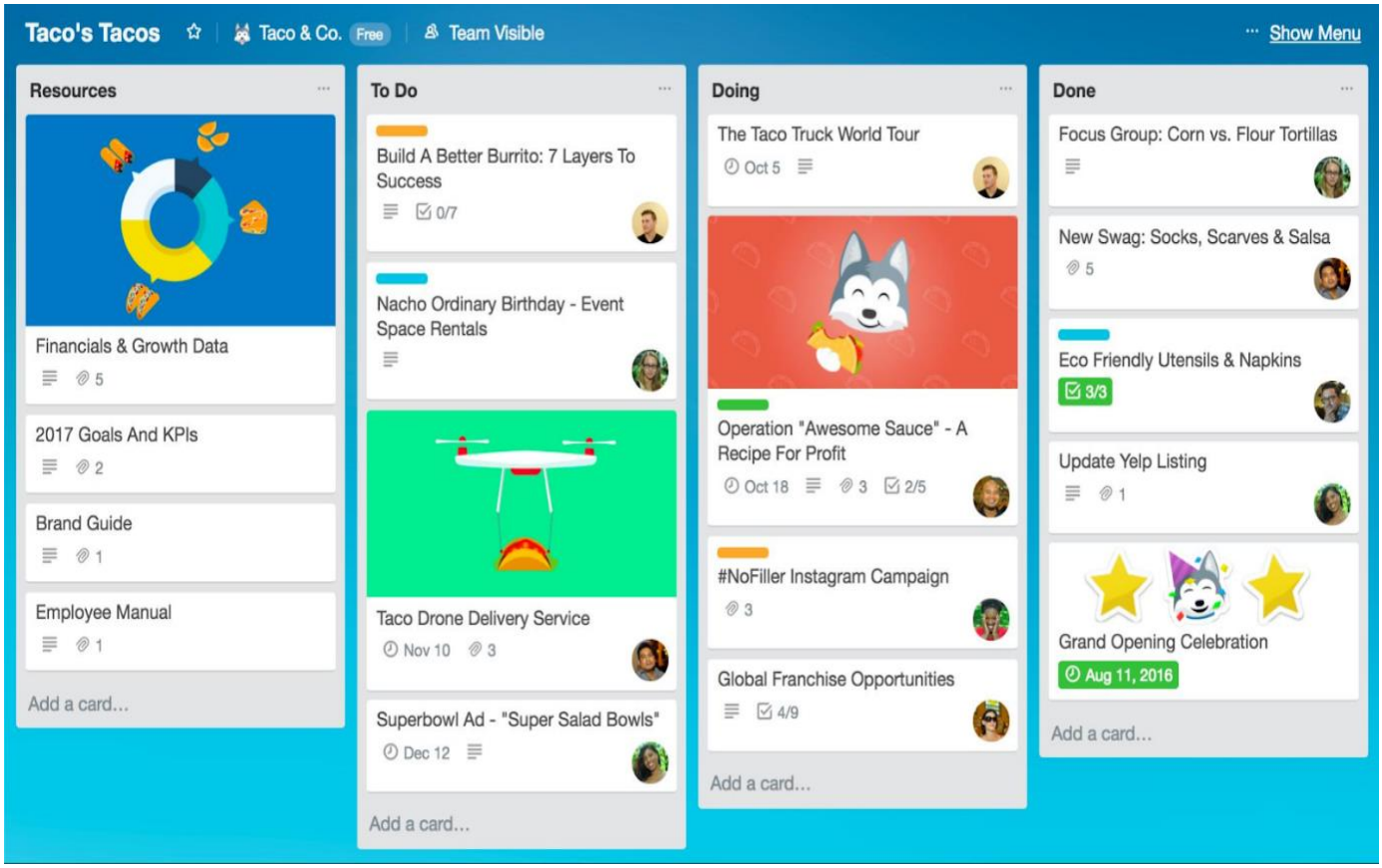

Рисунок 1.3 – Вигляд застосунку Trello[6].

Але поки незайнятою нішею залишається рішення для неформального менеджменту рішень та брейнштorму. Навіть в наш прогресивний час високих технологій велика кількість команд використовує маркерні дошки та інтерактивні дошки. Вже зараз існують застосунки які вдало реалізують старі підходи з новими ідеями рішаючи таки чином широкий клас проблем (проблема присутності, проблема функціональності, проблема колаборативності).

Проблема присутності вирішується реалізацією функціоналу маркерної дошки в вигляді онлайн застосунку або веб ресурсу у комбінації з телекомунікаційними застосунками. Що дозволяє виконувати роботу незалежно від розташування кожного робітника.

Проблема функціональності полягає в тому, що функціонал маркерних дошок обмежений тим що може намалювати людина. Ця проблема в деякому виді усунено в інтерактивних дошках, але веб застосунки дозволять набагато більше (синхронізація з хмарними сховищами даних, велика кількість віджетів таких як календарі, аудіо та

відео).

Проблема колаборативності полягає в тому, що Обмеженням маркерної дошки є кількість людей які можуть зручно одночасно користуватися нею. У програмного рівня такої проблеми немає зовсім у зв'язку зі специфікою предметної області.

Одним із подібних застосунків є Miro. Він був створений в 2014 році та вже зараз ним користуються понад три мільйони користувачів. Також його використовують такі компанії як Netflix, Twitter, Ikea, Cisco, Spotify. Але у даного класу застосунків існує проблема при роботі з великою кількістю даних зображений на одній сторінці що призводить до лагів і довгих завантажування сторінки або в деяких випадках є неможливим у зв'язку з обмеженнями системи.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Зараз основними представниками застосунків для колаборативного менеджменту застосунків[7] є Miro, AWW, Conceptboard, Explain Everything та Mural.

2.1 Miro

Miro – масштабована дошка для спільної роботи розподілених команд та візуальна платформа для співпраці для команд, які хочуть співпрацювати швидше, простіше та забезпечити кращі результати.

За допомогою Miro можна скористатися повним набором можливостей для співпраці, включаючи відео, чат, презентації та обмін даними, щоб полегшити багатофункціональну роботу командну та спростити співпрацю.

Користувацький інтерфейс застосунку показано на рисунку 2.1.

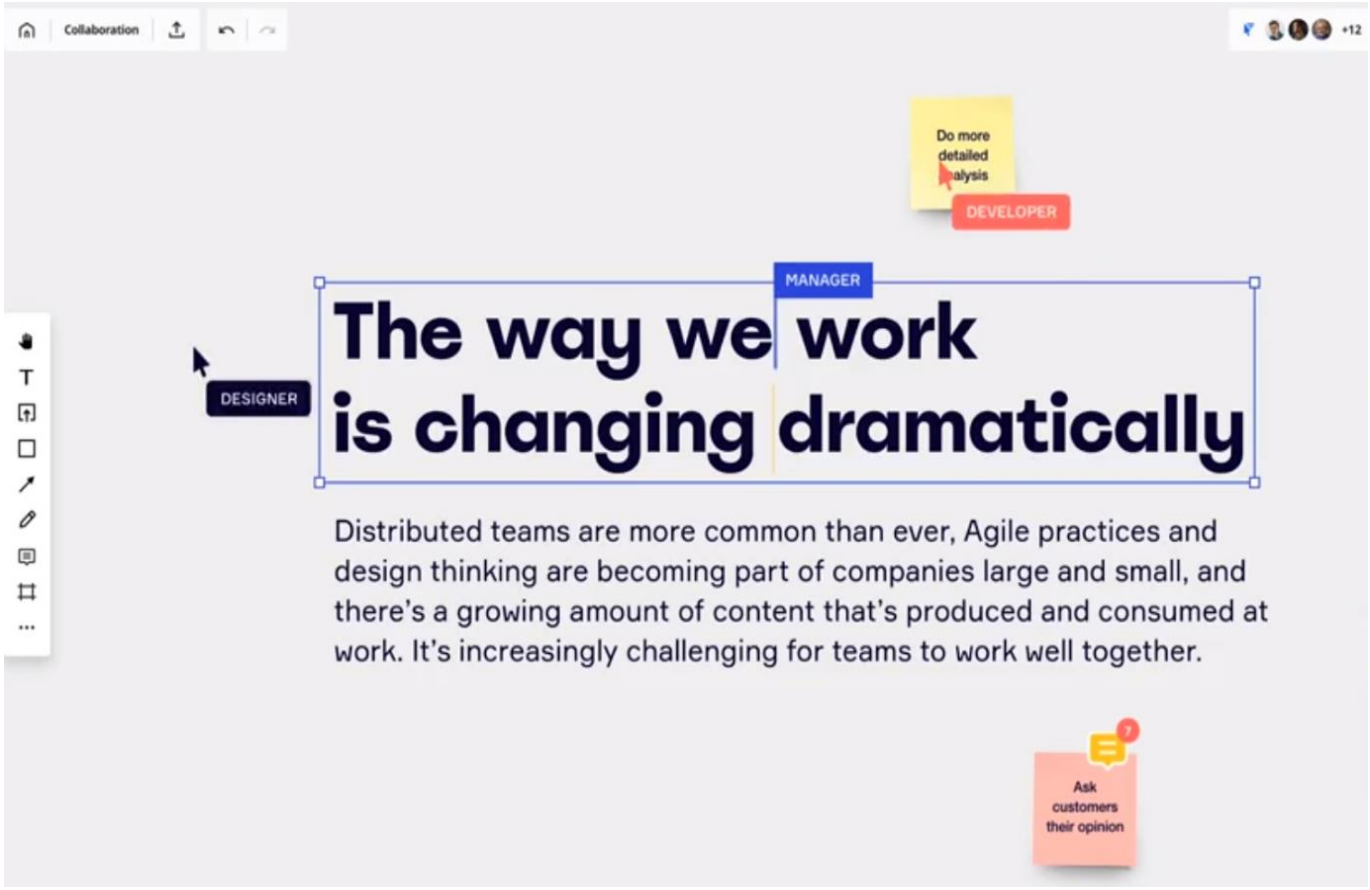

Рисунок 2.1 – Вигляд додатку Miro[8].

Це рішення пропонує розширення можливостей команд з дизайну, розробки та інженерії для створення інновацій у платформі, яка робить все можливим у режимі реального часу. Яка дає можливість створюйте концепції, картографувати історії користувачів або подорожі клієнтів або легко проводити планування дорожньої карти, що дозволяє зосередитись на наданні потрібних продуктів для клієнтів.

Далі наведені переваги, недоліки та вимірювані параметри застосунку Miro.

Переваги:

1) велика кількість віджетів;
2) можливість інтеграції з понад 20 застосунками (google drive, azure cards, jira cards, slack, microsoft teams...);
3) наявність плагінів (відеочат, розширення для хрому);
4) продвинутий текстовий редактор;
5) наявність нативного застосунка для самих поширених платформ.

Недоліки:

1) більша частина можливостей відкривається лише в платній підписці 16 доларів за кожного користувача;
2) погана робота з великою кількістю елементів на одній сторонці.

Вимірювані параметри:

1) швидкість завантажування сторінки при нулю елементів: 2.5с;
2) швидкість завантажування сторінки при невеликій кількості елементів: 3.0с;
3) швидкість завантажування сторінки при великій кількості елементів: 24с;
4) займаний простір при нулю елементів: 48 мб;
5) займаний простір при невеликій кількості елементів: 100 мб;
6) займаний простір при великій кількості елементів: 500 мб.

2.2 AWW

AWW – Найкраща онлайн-дошка для вбудовування в веб-сторінку. AWW видаляє всі перешкоди для початку роботи, оскільки не доведеться входити в систему,

перш ніж створювати свою першу дошку (хоча немає можливості зберегти дошку без останнього входу в систему).

Набір багатих функцій швидко доступний через три панелі інструментів, розташовані на дошці. У лівій бічній панелі розміщено інструменти для написання, текст, форми, завантаження та все інше, що потрібно для додавання вмісту до самої дошки. Праворуч — панель інструментів співпраці для текстового чату, голосових дзвінків та навігації на борту. А панель інструментів вгорі надає доступ до інформації дошки, що дозволяє переходити до інших дошок, збережених в обліковому записі. У той час як три окремі панелі інструментів роблять навігацію надзвичайно простою, це зменшує кількість місця для малювання.

Користувацький інтерфейс застосунку показано на рисунку 2.2.

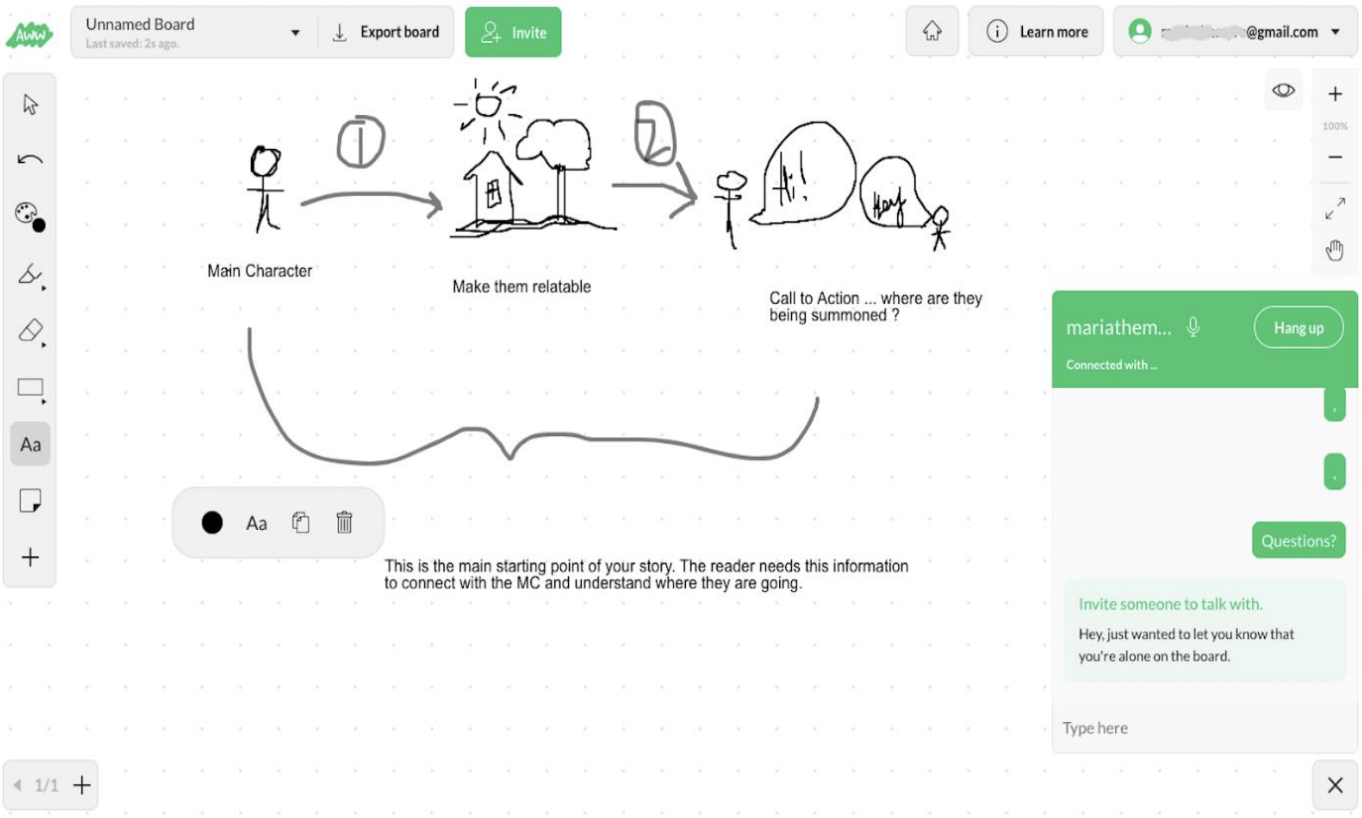

Рисунок 2.2 – Вигляд застосунку AWW[9].

За допомогою AWW можна вставити дошку на будь-яку веб-сторінку, скопіювавши та вставивши код iframe, подібно до того, як працює вбудоване відео YouTube. Усі, хто має доступ до сторінки, можуть скористатися цією

функціональною дошкою.

AWW надає безкоштовні основні інструменти та обмін також від $ 10/місяць для особистого акаунт, що включає необмежену кількість учасників та необмежену кількість преміум-дошок.

Далі наведені переваги, недоліки та вимірювані параметри застосунку AWW.

Переваги:

1) можливість створювати в iframe;
2) доступна ціна.

Недоліки:

1) погана робота з великою кількістю елементів на одній сторонці;
2) малий набір віджетів;
3) немає плагінів.

Вимірювані параметри:

1) швидкість завантажування сторінки при нулю елементів: 2.0с;
2) швидкість завантажування сторінки при невеликій кількості елементів: 2.8с;
3) швидкість завантажування сторінки при великій кількості елементів: 28с;
4) займаний простір при нулю елементів: 40 мб;
5) займаний простір при невеликій кількості елементів: 90 мб;
6) займаний простір при великій кількості елементів: 550 мб.

2.3 Conceptboard

Conceptboard – це чудовий вибір, якщо потрібна дошка для використання разом із командою при створенні великих, складних малюнків, таких як дошки розкладів або перегляд дизайну на багато сторінок. Окрім додавання малюнків, тексту та фігур можна вставляти відео та аудіовміст. Співробітники дошки можуть спілкуватися в чаті, залишати коментарі на самій дошці та навіть призначати завдання один одному в застосунку. Навіть за складністю рішень, яку застосунок дозволяє створювати, інтерфейсом досить легко орієнтуватися, навігація та панелі інструментів розташовані у верхній та лівій частині.

Користувацький інтерфейс застосунку показано на рисунку 2.3.

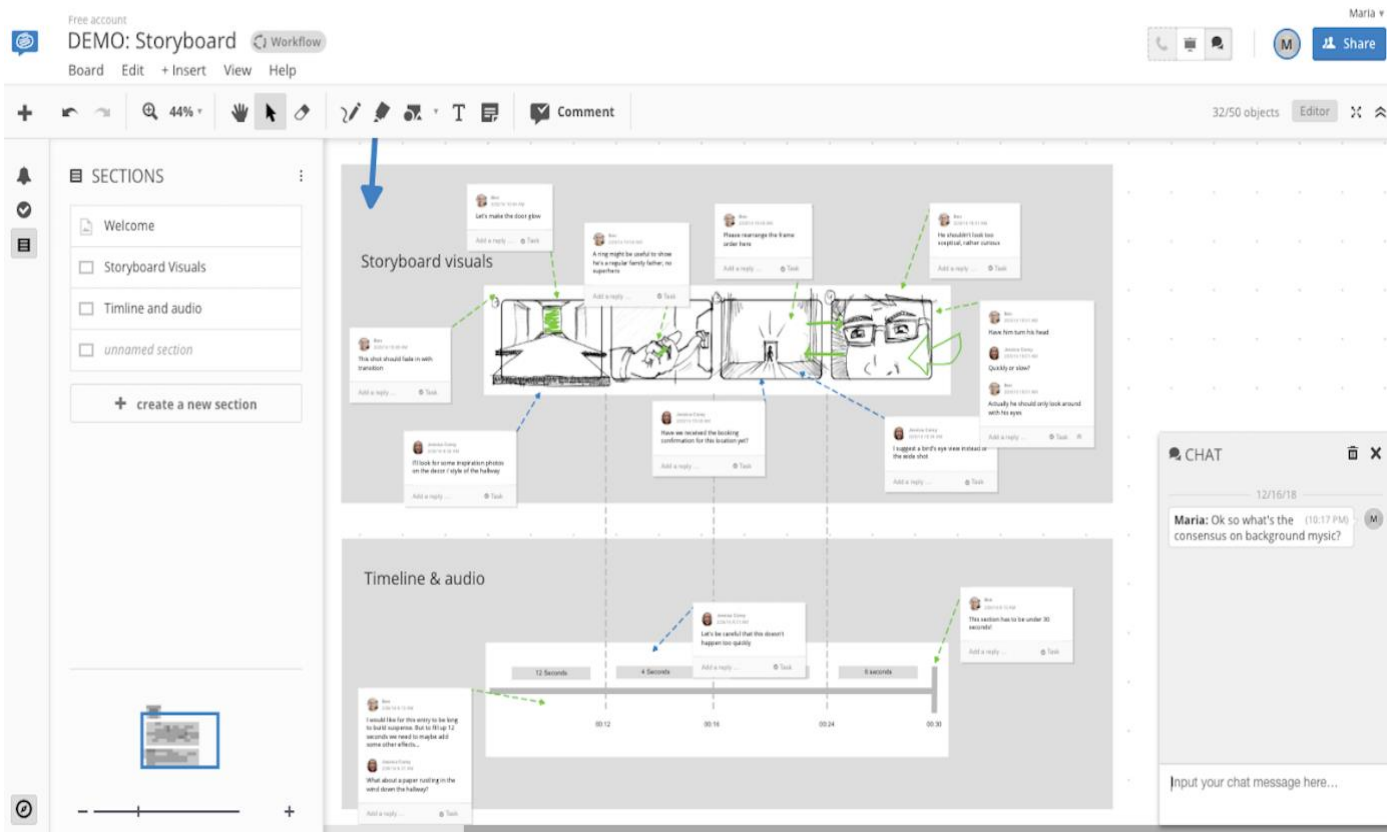

Рисунок 2.3 – Вигляд застосунку Conceptboard[10].

Conceptboard дозволяє створювати будь-яку кількість секцій на дошці. Клацання на заданій секції на бічній панелі автоматично пересуває вас до тієї частини дошки, тому ви можете легко перестрибувати великий або складний малюнок. Розділи також можна експортувати окремо у вигляді зображень або як група у форматі PDF, тому ви можете зберегти або поділитися ними поза застосунком.

Цінова концепція: безкоштовно для необмежених дошок до 50 об'єктів; від $ 9,50 за користувача на місяць для бізнес-плану, який включає в себе від трьох користувачів, необмежену кількість об'єктів на дошках, користувацький брендинг та 1 ТБ сховища.

Далі наведені переваги, недоліки та вимірювані параметри застосунку Conceptboard.

Переваги:

1) покращена робота з великою кількістю елементів;

2) доступна ціна;
3) експорт в PDF;
4) базові елементи комунікації.

Недоліки:

1) малий набір віджетів;
2) немає плагінів.

Вимірювані параметри:

1) швидкість завантажування сторінки при нулю елементів: 3.2с;
2) швидкість завантажування сторінки при невеликій кількості елементів: 3.8с;
3) швидкість завантажування сторінки при великій кількості елементів: 28с;
4) займаний простір при нулю елементів: 52мб;
5) займаний простір при невеликій кількості елементів: 120мб;
6) займаний простір при великій кількості елементів: 570мб.

2.4 Explain Everything

Якщо існує потреба в створенні відео з сеансу роботи на дошці, завжди можна запустити застосунок з запису екрану у фоновому режимі та створити за його допомогою потрібний запис. Або можна спростити процес і використовувати Explain Everything для дошки та відеореєстратора / редактора.

На дошці Explain Everything дозволяє малювати, додавати текст, завантажувати файли та створювати основні фігури. Все досить стандартно. Але оскільки він пропонує можливості запису відео, він дозволяє записувати дошку та голос під час малювання, а потім редагувати або розділяти елементи для вдосконалення готового відео. Після того, як користувач задоволений, він може зберегти відео, яким потом можна поділити в Інтернеті, як посиланням або завантажити у форматі MP4. Таким чином, люди, які не змогли взяти участь у сесії, все ще можуть зрозуміти процес, що стоїть за ним.

Користувацький інтерфейс застосунку показано на рисунку 2.4

Інструменти співпраці також вбудовані, тож доступна можливість запросити

членів команди переглянути або відредагувати дошку та поговорити з ними зі звуком під час роботи разом.

Застосунок безкоштовний для трьох проектів з максимальною тривалістю відео до однієї хвилини; від $ 8,99 / місяць для індивідуального плану з необмеженими проектами, слайдами та підтримкою електронної пошти.

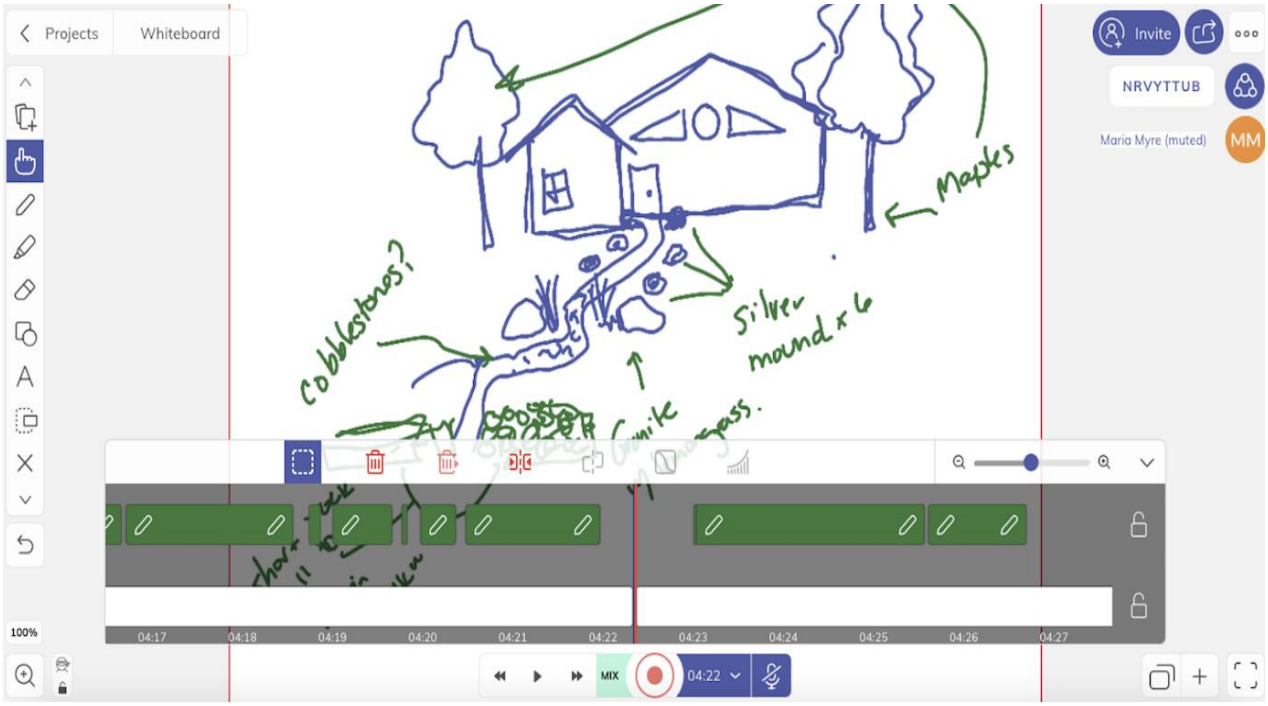

Рисунок 2.4 – Вигляд застосунку Explain Everything[11].

Далі наведені переваги, недоліки та вимірювані параметри застосунку Explain Everything.

Переваги:

1) автозапис;
2) доступна ціна.

Недоліки:

1) обмежений функціонал;
2) малий набір віджетів.

Вимірювані параметри:

1) швидкість завантажування сторінки при нулю елементів: 2.6с;
2) швидкість завантажування сторінки при невеликій кількості елементів: 3.2с;

3) швидкість завантажування сторінки при великій кількості елементів: 26с;
4) займаний простір при нулю елементів: 51мб;
5) займаний простір при невеликій кількості елементів: 104мб;
6) займаний простір при великій кількості елементів: 520мб.

2.5 Mural

На канвасі Mural можна розмістити декілька областей дошки, схожих на artboards в Photoshop, які можна переставити та змінити їх розмір за потребою. Кожна дошка може бути створена зі стандартним порожнім білим фоном або фон може бути вибраним з різних шаблонів, щоб додати заздалегідь вбудовані деталі для якоїсь додаткової структури.

Користувацький інтерфейс застосунку показано на рисунку 2.5

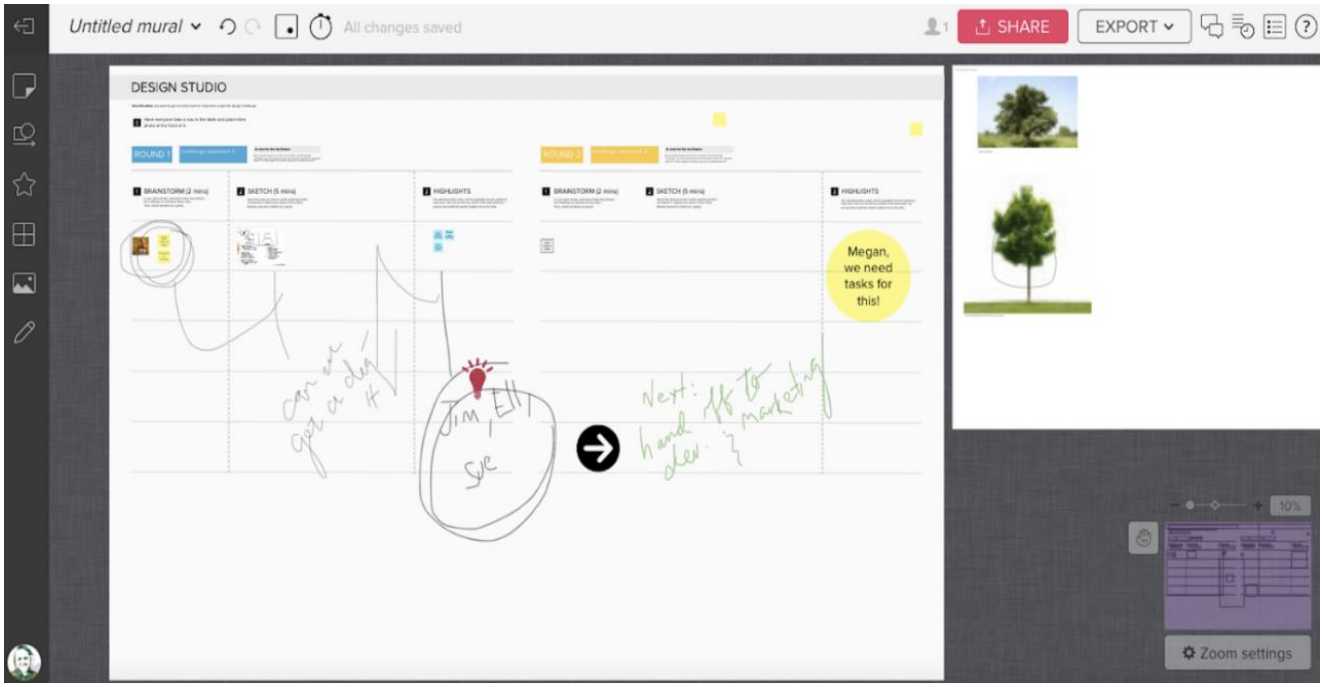

Рисунок 2.5 – Вигляд застосунку Mural [12].

Mural особливо сильний як командний інструмент, що дозволяє створювати кілька «кімнат» для дощок, кожна з яких має різні дозволи для обмеження доступу. Можна створити одну кімнату для команди дизайнерів, одну для маркетингової команди тощо, або можна створити кімнати для проектів.

Віддаленим командам, які використовують MURAL для зустрічей (наприклад,

Zapier), сподобається цифрова версія деяких офісних елементів, від таймерів (які можна використовувати для орієнтованих спринтів) до чатів. Ділитися відвертими відгуками на віддалених зустрічах команди може бути важко. Для цього MURAL є вбудована функція голосування, де можна віддати декілька голосів кожному співробітнику. Кожне голосування є анонімним та розміщується автоматично.

Ціни Mural: від $ 12 в місяць за людину для плану Starter.

Далі наведені переваги, недоліки та вимірювані параметри застосунку Mural

Переваги:

1) погана оптимізація;
2) малий набір віджетів.

Недоліки:

1) обмежений розмір сторінки.

Вимірювані параметри:

1) швидкість завантажування сторінки при нулю елементів: 1.5с;
2) швидкість завантажування сторінки при невеликій кількості елементів: 2.0с;
3) швидкість завантажування сторінки при великій кількості елементів: -;
4) займаний простір при нулю елементів: 38 мб;
5) займаний простір при невеликій кількості елементів: 80 мб;
6) займаний простір при великій кількості елементів: -;

Отже підсумовуючи вищесказане, можна зробити висновок, що хоча на ринку вже існує декілька гравців ніша ще не зайнята. В існуючих застосунків є певні недоліки, а спільним недолік є погана підтримка роботи з великою кількістю елементів, що продемонстровано в таблиці 2.1.

Таблияця 2.1 – порівняння застосунків

Назва		Miro	AWW	Conceptboard	Explain Everything	MURAL
	0 елементів	2.5	2.0	3.2	2.6	1.5
	небагато елементів	3.0	2.8	3.8	3.2	2.0
	багато елементів	24	28	28	26	-
	0 елементів	25	28	32	28	29
	небагато елементів	60	68	72	65	55
	багато елементів	500	550	570	520	-

3 ТЕОРЕТИЧНИЙ ОПИС РІШЕННЯ

3.1 Розробка вимог до системи

Система розробляється як застосунок для колаборативного неформального менеджменту ідей. Функціональні вимоги включають можливість одночасного доступу до спільного креативного простору. Нижче представлений більш детальний список функціональних вимог:

- реалізація менеджменту прав доступу таких як створення акаунтів компанії та користувача, створення сумісних креативних просторів, можливість обмежити або надати доступ користувачу до ресурсів, настройка ролей у групі користувачів;
- можливість розбивати креативний простір на креативні зони та надавати до них доступ в інших креативних просторах;
- підтримка віджетів: аудіо та відео елементів, календаря, голосування, стікерів, абстрактних форм, пензлика, mind-map та інші;
- можливість інтеграції зі сторонніми застосунками:
  - збереження ресурсів в google drive, onedrive, dropbox;
  - інтеграція з месенджерами Stack, Teams.

Список нефункціональних вимог системи:

- оптимізація великої кількості елементів за рахунок розбиття на рівні та шари;
- оптимізація займаного простору на девайсі користувача;
- пришвидшення завантажування сторінки користувача з великою кількістю елементів.

3.2 Загальний підхід до поставленої задачі

Для реалізації нефункціональний вимог було обрано метод кластеризації даних, метод розбиття на рівні, та метод вигрузки елементів.

3.2.1 Кластеризація canvas елементів

Як вже було показано в таблиці 2.1 при великій кількості елементів завантаження може займати більше 20 сукунд а розмір застосунку сягати понад 500 мегабайт. Для оптимізації таких випадків у застосунку можна використати кластеризацію. Враховуючи набір елементів, можна використовувати алгоритм кластеризації для класифікації кожного елементу у конкретну групі(рисунок 3.1). Теоретично елементи, що знаходяться в одній групі, повинні мати подібні властивості та / або особливості, тоді як точки даних у різних групах повинні мати сильно відрізняються властивості та / або особливості. Кластеризація — це метод непідвладного навчання і є загальною методикою статистичного аналізу даних, що використовується в багатьох галузях.

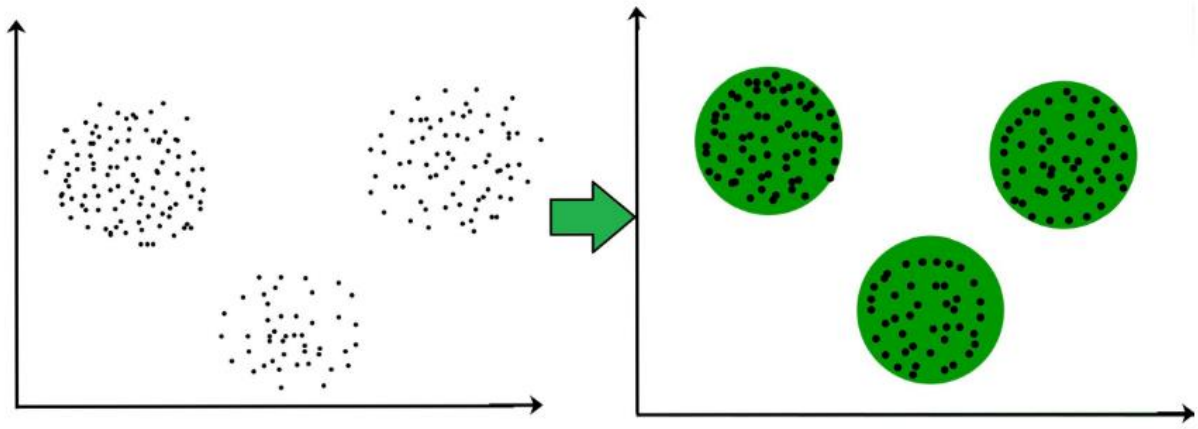

Рисунок 3.1 – Приклад кластеризації

Далі наведені можливі для використання типи алгоритмів кластеризації [13].

Моделі підключення: Як випливає з назви, ці моделі засновані на уявленні, що точки даних, розташовані ближче до простору даних, мають більшу схожість один з одним, ніж точки, що лежать далі. Ці моделі можуть дотримуватися двох підходів. У першому підході вони починають з класифікації всіх точок даних на окремі кластери, а потім агрегують їх у міру зменшення відстані. У другому підході всі точки даних класифікуються як один кластер і потім розподіляються по мірі збільшення відстані. Також вибір функції дистанції є суб'єктивним. Ці моделі дуже легко інтерпретувати, але не вистачає масштабованості для обробки великих наборів даних. Прикладами цих моделей є ієрархічний алгоритм кластеризації та його варіанти;

Центрові моделі: це ітераційні алгоритми кластеризації, в яких поняття подібності виводиться через близькість точки даних до центральної частини кластерів. Алгоритм кластеризації K-Means – це популярний алгоритм, який належить до цієї категорії. У цих моделях немає. про кластери, необхідні в кінці, потрібно заздалегідь згадати, що робить важливим попереднє знання набору даних. Ці моделі виконуються ітераційно, щоб знайти місцеві оптимуми;

Моделі розподілу: ці моделі кластеризації базуються на уявленні про те, наскільки ймовірним є те, що всі точки даних кластера належать одному і тому ж розподілу (наприклад: Нормальний, Гаусський). Ці моделі часто страждають від надягання. Популярним прикладом цих моделей є алгоритм очікування-максимізації, який використовує багатоваріантні нормальні розподіли;

Моделі щільності: ці моделі здійснюють пошук у просторі даних за областями різної щільності точок даних у просторі даних. Він виділяє різні області різної щільності та призначає точки даних у цих регіонах в одному кластері. Популярні приклади моделей щільності - DBSCAN та OPTICS.

У випадку обраної системи кластеризація буде використовуватися для групування елементів які знаходяться близько один до одного. Таким чином у деяких випадках велику кількість елементів можна буде замінити кластером.

В креативному просторі може існувати багато елементів при максимально віддалені користувачі не бачить їх текст, але вони навантажують сторінку і це може призвести до уповільнення сторінки.

Тому при віддалені можна замінити всі елементи (рисунок 3.2) кластеру на один логічний елемент. Тоді система матиме такий як на рисунку 3.3. При натисканні на вибраний елемент будуть відображені всі його елементи. Це буде використано у випадку коли користувач сильно збільшує масштаб. У описаному випадку система відображає велику кількість елементів на екрані, а із-за віддаленості користувач не може розрізняти їх.

У системі є два види кластеризації: автоматичний і клієнтський. В автоматичному режимі при створенні нового елементу він стає частиною найближчого існуючого кластеру автоматично або якщо поблизу немає кластеру то

створюється новий. Автоматичний режим є режимом за замовчуванням.

Рисунок 3.2 – розгорнутий вигляд компонентів

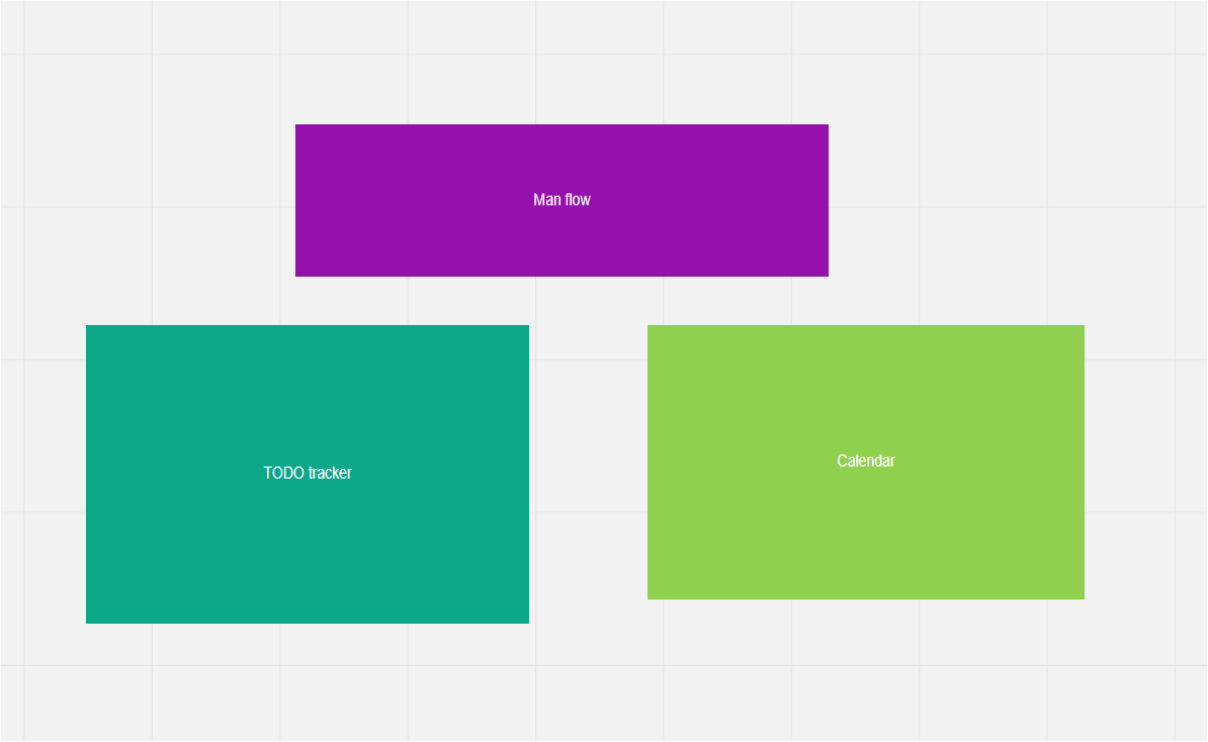

Рисунок 3.2 – розгорнутий вигляд компонентів

Клієнтський режим використовується за бажанням користувача і дає йому можливість самому виділити елементи в кластер.

3.2.2 Розбиття на рівні

Цей режим використовується для оптимізації віджетів (списків, календарів, документів та інших) з великою кількістю даних. Для таких елементів існує декілька видів (рівнів) мінімальний (рисунок 3.3) та максимальний(рисунок 3.4). Так як користувач не може переглядати велику кількість елементів одночасно то при віддаленні будуть відображатися елементи у мінімальному вигляді, тобто буде завантажена і відображена лише частина даних кожного елемента, наприклад лише перші десять елементів списку або перша сторінка документу. При видаленні елементу у мінімальному вигляді він буде замінений елемент у максимальному вигляду (будуть завантажені та відображені всі дані елементу).

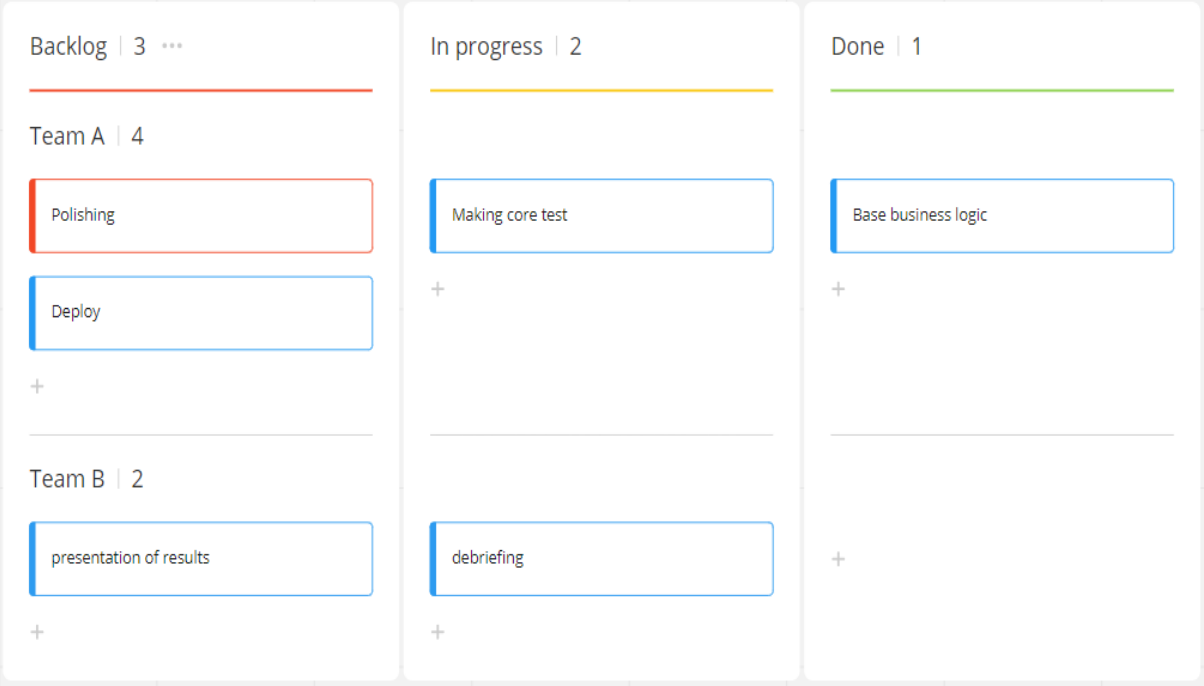

Рисунок 3.3 – розгорнутий вигляд компонентів

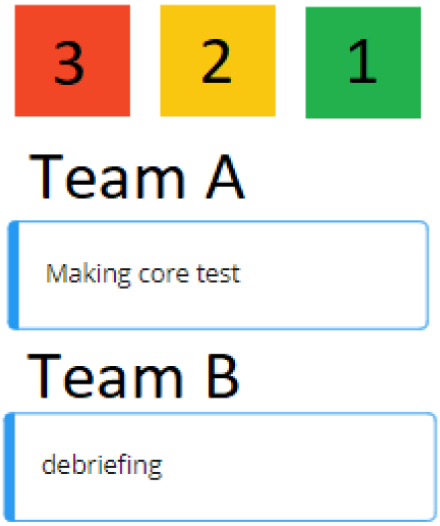

Рисунок 3.4 – розгорнутий вигляд компонентів

3.2.3 Метод вигрузки елементів

Так як перші два методи приводять до частого оновлення canvas то доцільним буде використання заходів оптимізації роботи canvas. Таким чином метод вигрузки елементів дозволяє групувати canvas об'єкти у цілі компоненти та за потребою додавати або видаляти їх на сторінці.

Висновки

Комбінація вищезазначених підходів значний виграш при роботі з великою кількістю даних, що продемонстровано в таблиці 3.1.

Таблиця 3.1 – порівняння власного застосунку з існуючими

Назва		Miro	AWW	Conceptboard	Explain Everything	Власний застосунок
	0 елементів	2.5	2.0	2.8	2.6	2.9
	небагато елементів	30	2.8	3.2	3.2	3.5

Продовження таблиці 3.1

Назва		Miro	AWW	Conceptboard	Explain Everything	Власний застосунок
Швидкість завантаження в секундах	багато елементів	24	28	28	26	3.5
	0 елементів	25	28	32	28	29
	небагато елементів	60	68	72	65	66
	багато елементів	500	550	570	520	80

4 РЕАЛІЗАЦІЯ ТЕХНОЛОГІЧНОГО СТЕКУ

4.1 Архітектурне проектування програмного забезпечення

Для реалізації застосунку було обрано модульну архітектуру[14, 15, 16, 17] (рисунок 4.1)

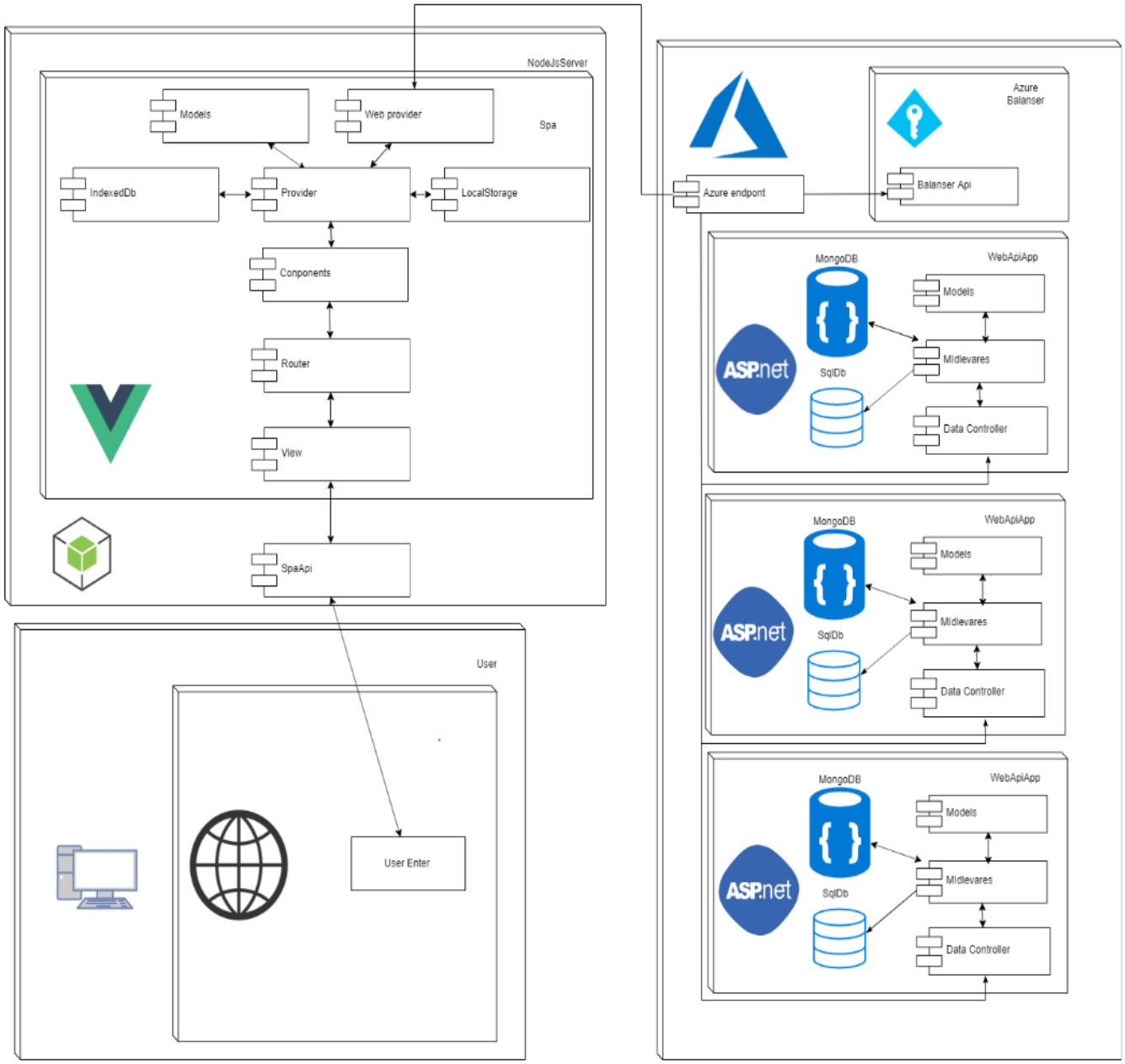

Рисунок 4.1 – Вигляді архітектури

Ця архітектура була вибрана так як у зв'язку зі специфікою рішення існують

груп даних які не пересікаються між собою і можуть існувати в різних інстанції баз даних. Таким чином існуватиме єдина точка входу до якої звертається користувач за spa застосунком та всіма потрібними бандлами та будь-яка кількість екземплярів Api застосунків з бізнес логікою та власними базами даних. Також spa застосунок буде підтримувати інтеграцію зовнішніх застосунків.

4.2 Вибір бекенд технологій

Зараз існує велика кількість бекенд фреймворків, більша їх частина кросплатформна з відкритим кодом.

Основними бекенд фреймворками в наш час виступають[18, 19] фреймворки представлені в таблиці

Таблиця 4.1 – Сучасні бекенд реймворки

Номер	Назва фреймворку	Мова	Рік створенная	Стабільна версія
1	ASP.NET	C#	2002	4.7.1
2	Ruby on Rail	Ruby	204	6.0.1
3	Spring	Java	2002	5.2.0-RC1
4	Django	Python	2005	3.0
5	Node.js	JavaScript	2002	13.2.0

Spa частина додатку виступає у вигляді точки входу і не потребує великої кількості ресурсів тому для неї було обрано Node.js. Node.JS дозволяє розробникам перетворити будь-який сервер на легкий веб-сервер із лише кількома рядками коду.

Він ідеально підходить для розробки невеликих легких веб-прикладних програм, не турбуючись про всі необхідні інфраструктурні сантехніки.А найпростіший сервер на Node.js виглядає наступим чином:

```
var http = require('http');
```

```
var server = http.createServer(function (req, res) {
   var body = 'Amazing lightweight webserver using node.js\n';
   var content_length = body.length;
   res.writeHead(200, {
      'Content-Length': content_length,
      'Content-Type': 'text/plain' });
   res.end(body);
});
server.listen(3939);
console.log('Server is running on port 3939');
```

Найпростіше визначення Node.js, визначає його як JavaScript на стороні сервера. Більш складне визначення, безумовно, буде включати деяку посилання на Node.js з використанням подій на основі процедури виконання сервера, що полегшує розробникам реалізацію архітектури, керованої подіями сервера, у своїх застосунках. Одне з дуже багатьох класних речей про node.js - це те, що це дозволяє легко створювати дійсно прості програми, керовані подіями, з дуже мало рядків коду.

Для Api застосунку було обрано ASP.NET.

Існує маса вагомих причин використовувати ASP.NET при розробці веб-сайту чи програми. Висока швидкість, низька вартість та широка підтримка мови є одними з найбільш значущих переваг. ASP.NET вбудований у звичне середовище сервера Windows, вимагає менших налаштувань та конфігурації, ніж інші платформи веб-розробки, які повинні бути встановлені та налаштовані окремо. Популярність ASP.NET дозволяє легко знайти Інтернет-ресурси та кваліфікованих розробників.

Веб-сайти та програми, створені за допомогою ASP.NET, можуть бути швидшими та ефективнішими, ніж, наприклад, створення веб-сайтів за допомогою PHP. Програми ASP.NET складаються, що означає, що код переводиться в об'єктний код, який потім виконується. Цей процес компіляції займає невелику кількість часу, але відбувається лише один раз. Після компіляції код може бути виконаний знову і знову платформою .Net дуже швидко.

Інтерпретований код не виконується безпосередньо машиною, але його потрібно читати та інтерпретувати кожен раз перед його виконанням. Скомпільований код, як правило, швидший і більш масштабований, ніж інтерпретований код, і він може робити все, що може інтерпретувати код. Приклади інтерпретованих мов включають PHP, JavaScript та Ruby.

Процес компіляції також забезпечує перевірку відповідності всього коду. Наприклад, якщо метод з назвою GetUser перейменований на GetEposleee як частина деяких оновлень коду, будь-яке посилання на GetUser у всій решті програми призведе до помилки під час компіляції, що полегшить її ідентифікацію та виправлення. Інтерпретовані мови не ідентифікують цю помилку, поки код не буде фактично запущений та протестований. У великому додатку дуже багато часу займає тестування кожного сценарію вручну або написання та підтримка додаткового коду, який тестується для кожного сценарію, щоразу, коли код змінюється.

ASP.NET пишеться з використанням об'єктно-орієнтованих мов програмування, таких як C # або VB.net. Об'єктно-орієнтоване програмування забезпечує рамки та структури для організації коду та їх повторного використання. Незважаючи на те, що VB.net є перевагою від застарілого продукту Visual Basic від Майкрософт і значною мірою не прихильнився серед розробників, C # є першокласною мовою програмування і стабільно входить до числа найбільш затребуваних і найбільш використовуваних мов програмування у світі.

Нарешті, незважаючи на те, що ASP.NET є відкритим кодом та безкоштовний у використанні, він активно розробляється та підтримується найбільшою в світі програмною компанією Microsoft. Microsoft вкладає значні кошти в свої платформи розвитку, спільноту розробників та підтримує програмні компанії, які використовуються для запуску цих додатків. Це означає, що вам не потрібно хвилюватися, коли ваше програмне забезпечення стане вчорашньою новиною.

Отже, для spa застосунку було обрано Node.js так як його серверна частина є лише звичайною точкою доступа з мінімальною кількістю бізнес-логіки. Для WebApi застосунку було обрано Asp.Net WebApi так створення високофункціональної бізнес логіки за його допомогою буде зручніше, так як для менеджменту та оркестровки

екземпляр можна буде використати Azure що значно облегшить супровід застосунку та зменшить витрати часу на розробку за рахунок усунення більшої частини задач пов'язаних з розгортанням системи. Реалізація Net WebApi архітектури продемонстрована на рисунку 4.2

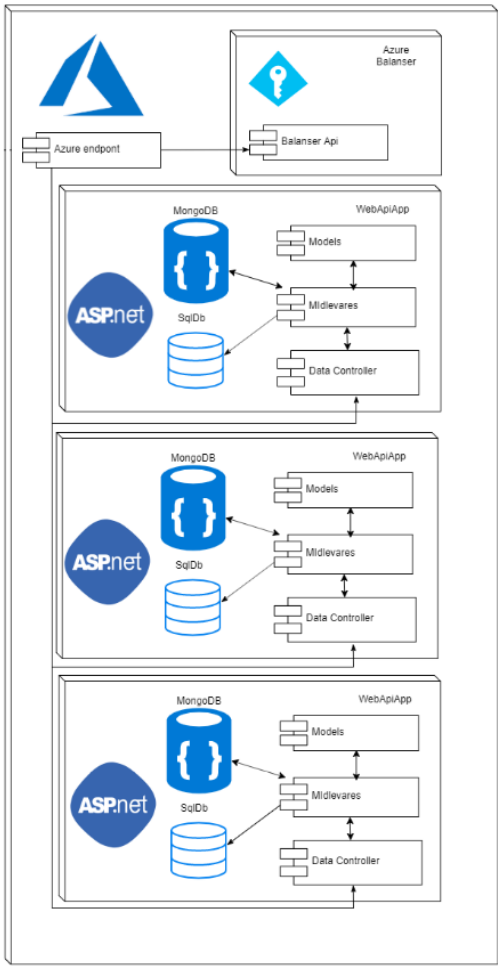

Рисунок 4.2 – Вигляді бекенд архітектури

4.3 Вибір фронтенд технологій

Так як, для фронтенд частина повинна реалізовувати специфічну функцію, а саме високонавантажений canvas з можливістю швидко змінювати значні за розміром canvas гілки, то потрібно було використовувати фреймворк із прогресивний spa можливостями. Вибір проводився між трьома додатками AngularJS Vue.js та React.

Основні фреймворки для розробки фронтенду наведені в наступній таблиці.

Таблиця 4.2 – сучасні фронтенд фреймворки

Назва	Тип	Модель	Версія	Рік створення
AngularJS	фреймворк	mvc	8.2.14	2016
Vue.js	фреймворк	mvvm	2.6.10	2014
React	бібліотека	functionnal	16.12.0	2013

Порівняння продуктивності різних фреймворків показано на рисунку 4.4

Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
create rows Duration for creating 1000 rows after the page loaded.	193.1 ± 7.9 (1.2)	212.2 ± 14.2 (1.3)	166.7 ± 8.6 (1.0)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	197.4 ± 5.3 (1.2)	206.7 ± 7.3 (1.2)	168.5 ± 5.0 (1.0)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	13.0 ± 4.5 (1.0)	18.0 ± 1.6 (1.1)	17.3 ± 2.9 (1.1)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	3.4 ± 2.3 (1.0)	8.7 ± 2.9 (1.0)	9.3 ± 1.7 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	13.4 ± 1.0 (1.0)	17.1 ± 1.3 (1.1)	18.3 ± 1.5 (1.1)
remove row Duration to remove a row. (with 5 warmup iterations).	46.1 ± 3.2 (1.0)	52.4 ± 1.7 (1.1)	52.6 ± 2.7 (1.1)
create many rows Duration to create 10,000 rows	1946.0 ± 41.8 (1.2)	1931.7 ± 35.6 (1.2)	1587.5 ± 33.9 (1.0)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	324.6 ± 10.1 (1.0)	366.4 ± 10.9 (1.1)	399.5 ± 11.0 (1.2)
clear rows Duration to clear the table filled with 10.000 rows.	379.9 ± 11.3 (1.5)	410.9 ± 9.8 (1.6)	254.5 ± 5.0 (1.0)
startup time Time for loading, parsing and starting up	84.3 ± 2.6 (1.5)	93.8 ± 6.9 (1.7)	56.6 ± 2.5 (1.0)
slowdown geometric mean	**1.14**	**1.23**	**1.06**

Рисунок 4.4 – продуктивність фреймворків

Порівняння займаного місця різними фреймворками різних фреймворків показано на рисунку 4.3.

Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
ready memory Memory usage after page load.	4.8 ± 0.0 (1.3)	4.9 ± 0.1 (1.3)	3.8 ± 0.0 (1.0)
run memory Memory usage after adding 1000 rows.	10.9 ± 0.1 (1.4)	10.8 ± 0.1 (1.4)	7.5 ± 0.1 (1.0)

Рисунок 4.3 – займана пам'ять фреймворків

Отже, основуючись на вище описаних параметрах було обрано Vue.js, а обрана ахітектура продемонстрована на рисунку 4.5.

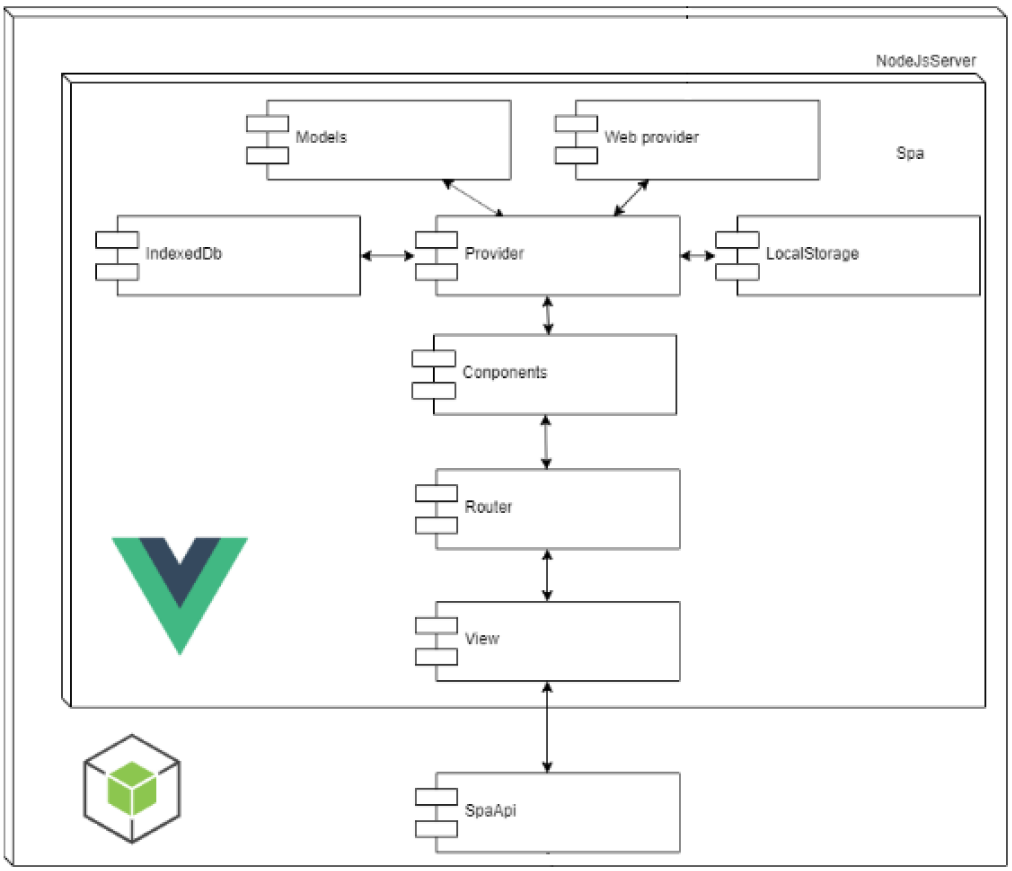

Рисунок 4.5 – Вигляді архітектури фронтенд

4.4 Вибір сховищ даних

Сховище даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування[22, 23] (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в інформаційних системах. На даний час застосунки для роботи з базами даних є одними з найпоширеніших прикладних програм.

4.4.1 Бекенд сховища даних

Опис основних видів сховищ даних представлено у таблиці 4.3

Таблиця 4.3 – порівняння сховищ даних

Тип	Сценарій використання	Приклад
Масштабна RDBMS	Використання семантики ACID звільняє програмістів від необхідності працювати на досить низькому рівні, а саме, відповідати за блокування і несуперечність даних, обробляти колізії.	Застосунки, яким не потрібні оновлення або злиття даних, що охоплюють безліч вузлів.

Тип	Сценарій використання	Приклад
Сховище типу ключ-значення	Підходить для простих застосунків, з одним типом об'єктів, в ситуаціях, коли пошук об'єктів виконують лише	Інтерактивне оновлення домашньої сторінки користувача в Facebook.
Сховище, орієнтоване на документи	Підходить для зберігання об'єктів різних типів.	Транспортний застосунок, що оперує даними про водіїв і автомобілях, працюючи з яким треба шукати об'єкти по різних полях, наприклад - ім'я або дата народження водія, номер прав.
Система зберігання даних з розширюваними записами	Більш висока пропускна здатність і кращі можливості паралельної обробки даних ціною злегка більш високої складності.	Застосунки, схожі на eBay. Вертикальне і горизонтальний поділ даних для зберігання інформації клієнтів.

Так як дані для застосунку можна чітко розділити в дві різні категорії, а саме структуровані дані користувача, творчих просторів, дані що описують їх параметри та дані розгорнутих canvas елементів було вирішено розділити їх на два різні бази даних

Для структурованих даних було обрано реляційне представлення, тому що по існує потреб в постійному пошуку по цим даним (дані користувача, творчих просторів, дані що описують їх параметри).

Для неструктурованих даних було обрано документу базу mongoDb, тому що

даним розгорнутих canvas елементів більшу частину часу будуть зберігатися гарячому стані в застосунку, а звернення до бази буде здійснюватися лише по ключу для створення нового елементу або видалення старого. Перезапис здійсненого за допомогою зміни ідентифікатора в реляційній базі, створення нового елементу в mongoDb та видалення елменту по старому ідентифікатору.

Отже, для бекенд баз даних було обрано дві технології sql-бази даних для зберігання даних профілю користувачів, команд та права доступу, а для зберігання елементів креативного простору було обрана NoSql база даних mongoDb.

4.4.2 Фронтенд сховища даних

Основними сховищами даних на стороні клієнта[26,27] є Web storage, Indexed Database, IndexedDB, Web SQL Database та cookies.

Так як, дані які будуть зберігатися на стороні клієнта мають досить простий форма то основним параметром вибору стала підтримка сховища браузерами

Підтримка Web Storage браузерами показано на рисунку 4.6.

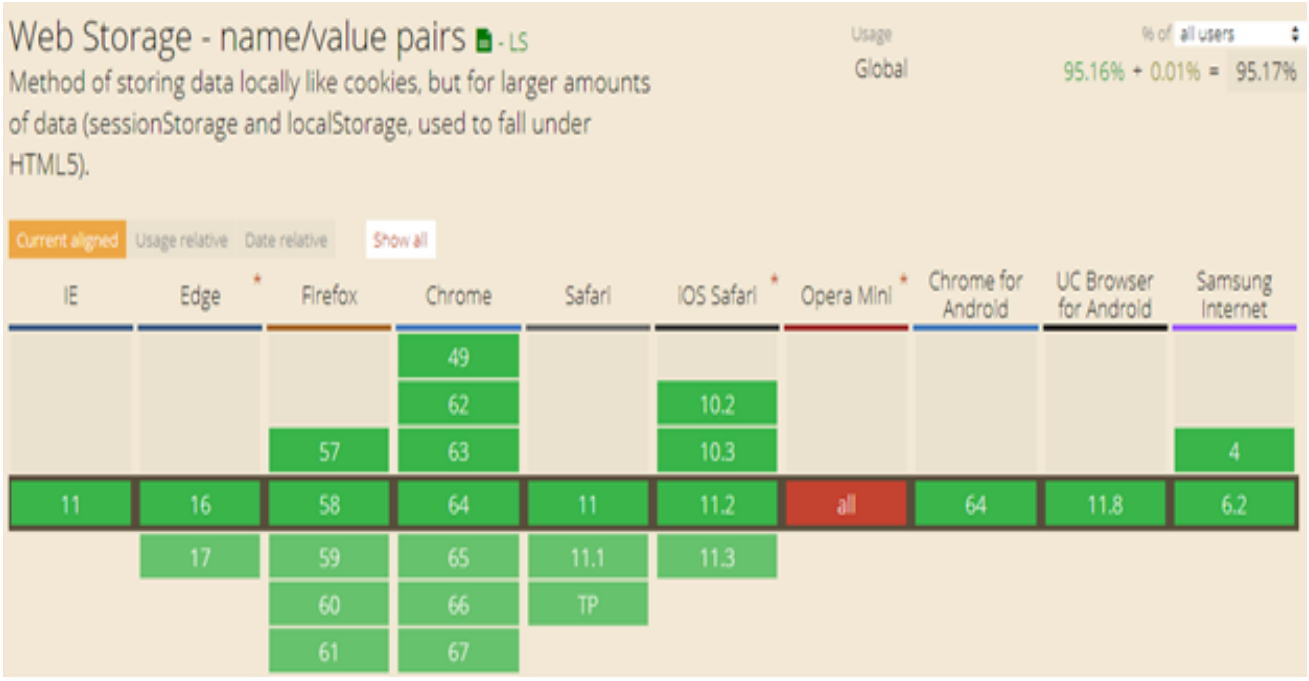

Рисунок 4.6 – підтримка WebStorage

Підтримка IndexedDB браузерами показано на рисунку 4.7.

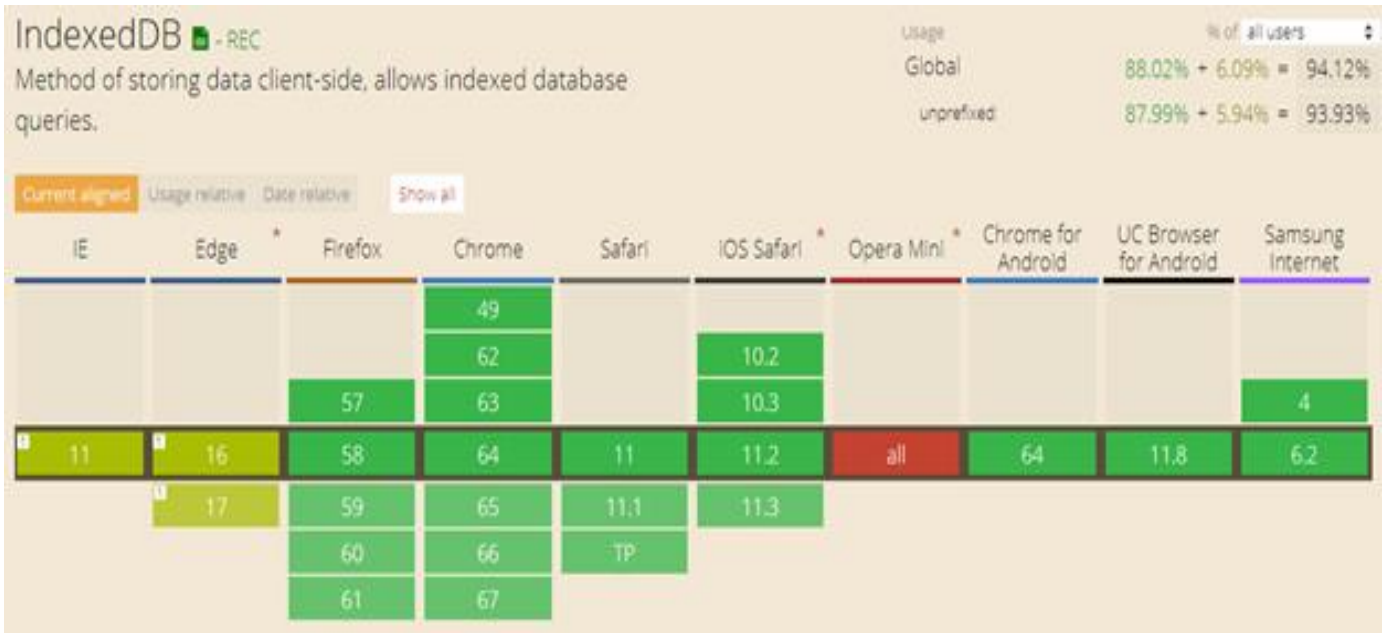

Рисунок 4.7 – підтримка IndexedDB

Підтримка Web SQL Database браузерами показано на рисунку 4.8.

Рисунок 4.8 – підтримка Web Database

З вищезазначених даних було прийнято рішення використовувати IndexedDB для оптимізації застосунку.

IndexedDB - одна з можливостей зберігання, впроваджена в браузери протягом багатьох років.

Це сховище ключів (база даних noSQL), яке вважається остаточним рішенням для зберігання даних у браузерах.

Це асинхронний API, що означає, що виконання дорогих операцій не блокує потік інтерфейсу користувача, надаючи неохайний досвід для користувачів. Він може зберігати необмежену кількість даних, хоча один раз за певний поріг користувачеві пропонується дати сайту більш високі межі.

Він підтримується у всіх сучасних браузерах.

Він підтримує транзакції, версії версій та дає хороші показники.

Хоча ви технічно можна створювати кілька баз даних на одному сайті, як правило, створюється одна єдина базу даних, а всередині цієї бази даних можна створювати кілька об'єктів-сховищ.

База даних є приватною для домену, тому будь-який інший сайт не може отримати доступ до іншого веб-сайту магазинів IndexedDB.

Далі наведено приклад ініціалізації базових функцій для робот з IndexedDB.

```
function openIndexedDB (fileindex) {
  var openDB = indexedDB.open("MyDatabase", 1);
  openDB.onupgradeneeded = function() {
    var db = {}
    db.result = openDB.result;
    db.store = db.result.createObjectStore("MyObjectStore", {keyPath: "id"});};
  return openDB;}
function getStoreIndexedDB (openDB) {
  var db = {};
  db.result = openDB.result;
  db.tx = db.result.transaction("MyObjectStore", "readwrite");
  db.store = db.tx.objectStore("MyObjectStore");
```

```
  db.index = db.store.index("NameIndex");
  return db;}
function saveIndexedDB (filename, filedata, fileindex) {
  var openDB = openIndexedDB(fileindex);
  openDB.onsuccess = function() {
    var db = getStoreIndexedDB(openDB);
    db.store.put({id: filename, data: filedata});}
  return true;}
function findIndexedDB (filesearch, callback) {
  return loadIndexedDB(null, callback, filesearch);}
function loadIndexedDB (filename, callback, filesearch) {
  var openDB = openIndexedDB();
  openDB.onsuccess = function() {
    var db = getStoreIndexedDB(openDB);
    var getData;
    getData.onsuccess = function() {
      callback(getData.result.data);};
    db.tx.oncomplete = function() {
      db.result.close()};}
  return true;}
```

5. РОЗРОБКА СИСТЕМИ

5.1 Сценарії використання системи

Діаграму сценаріїв із функціональними вимогами наведено у додатку Е, детальні описи прецедентів наведено у таблицях нижче.

Таблиця 5.1 – Сценарій використання

Назва	Перегляд доступних творчих просторів
ID	1
Опис	Після логіну користувач потрапляє у власний профіль на якому може переглядати всі доступні креативні простори
Актори	користувач, користувач з правом перегляду, користувач з правом редагування, адмін
Вигоди компанії	Сервіс неможливий без зручного перегляду доступних користувачеві даних
Частота користування	Постійно
Тригери	Користувач входить в систему
Передумови	Користувач проходить крок логіну
Постумови	-
Основний розвиток	Користувач переходить в меню налаштувань
Альтернативні розвитку	-

Винятки	-

Таблиця 5.2 – Сценарій використання

Назва	Перегляд свого профілю
ID	2
Опис	Після логіну користувач потрапляє у власний профіль на якому є меню налаштувань
Актори	користувач, користувач з правом перегляду, користувач з правом редагування, адмін
Вигоди компанії	Якщо користувач не може настроїти систему для зручного користування то він навряд чи буде її використовувати
Частота користування	Невідомо, як часто ця опція буде використовуватися на практиці
Тригери	Користувач переходить в власний профіль з творчого простору або користувача автоматично перенаправляє на сторінку власного профілю після проходження авторизації
Передумови	Користувач проходить крок логіну або натискаючи кнопку повернутися у власний профіль у момент знаходження у творчому просторі
Постумови	-

Основний розвиток	Користувач переходить в меню налаштувань
Винятки	-

Таблиця 5.3 – Сценарій використання

Назва	Запит на отримання прав
ID	3
Опис	Користувач виконує запит до адміна на отримання прав
Актори	користувач, користувач з правом перегляду, користувач з правом редагування, адмін
Вигоди компанії	Важлива функція менеджменту ролей, яка потрібна користувачам в великих командах, може використовуватися як одна із переваг у маркетинговій компанії.
Частота користування	Періодично
Тригери	Користувач вибирає опцію творчого простору «Отримання прав»
Передумови	Користувач знаходиться на творчому просторі або в головному меню
Постумови	Відправляється запит на отримання прав
Основний розвиток	Після того як адмін перегляне запит він може видати користувач обрані права або проігнорувати запит

Альтернативні розвитку	Користувач здійснює запит з власного профілю по ссилці на творчий простір
Винятки	-

Таблиця 5.4 – Сценарій використання

Назва	Перегляд творчого простору
ID	4
Опис	Користувач може переглядати творчий простір
Актори	користувач з правом перегляду, користувач з правом редагування, адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Постійно
Тригери	Користувач переходить на сторінку творчого простору вибравши потрібний елемент списку доступних творчих просторів на сторінці власного профілю
Передумови	Користувач отримує право перегляду творчого простору здійснивши відповідний запит на отримання прав у амінна
Постумови	-
Основний розвиток	Користувач вибирає творчий простір із доступних

	йому, натискає на його іконку та переходить на його сторінку
Альтернативні розвитку	Користувач здійснює перехід по прямому посиланні, яке користувач може зберегти заздалегіть
Винятки	-

Таблиця 5.5 – Сценарій використання

Назва	Взаємодія з активними елементами творчого простору
ID	5
Опис	Користувач може взаємодіяти з деякими елементами творчого простору (голосуванням, медіа елементами, чатами, та віджетами з відповідними настройками)
Актори	користувач з правом перегляду, користувач з правом редагування, адмін
Вигоди компанії	Частина основного мінімального функціоналу
Частота користування	Постійно
Тригери	Користувач знаходячись в творчому просторі може здійснювати відповідні дії відносно активних елементів

Передумови	Користувач отримує право перегляду творчого простору та знаходиться у творчому просторі
Постумови	Зміна стану активного елементу
Основний розвиток	Користувач вибирає творчий простір із доступних йому, натискає на його іконку та переходить на його сторінку для взаємодії з елементом.
Альтернативні розвитку	-
Винятки	Активні компоненти з зміненим параметром взаємодії

Таблиця 5.6 – Сценарій використання

Назва	Додавання компонентів
ID	6
Опис	Користувач може додати новий компонент до творчого простору з меню компонентів
Актори	користувач з правом редагування, адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Постійно
Тригери	Користувач відкриває меню компонентів потім вибирає компонент із меню та перекосить його на творчій простір
Передумови	Користувач отримує право редагування творчого

	простору та знаходиться у творчому просторі
Постумови	Появлення в творчому просторі обраного з меню елементу
Основний розвиток	Користувач знаходячись на сторінці творчого простору переносить елемент з меню елементів на творчій простір
Альтернативні розвитку	Користувач вставляє заздалегідь скопійований елемент з буферу обміну комбінацією клавіш Ctrl+V
Винятки	Заборонені адміном компоненти

Таблиця 5.7 – Сценарій використання

Назва	Видалення компонентів
ID	7
Опис	Користувач може видалити компонент з творчого простору
Актори	користувач з правом редагування, адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Постійно
Тригери	Користувач вибирає пункт видалити у елемента творчого простору

Передумови	Користувач отримує право редагування творчого простору та знаходиться у творчому просторі
Постумови	Видалення елементу з творчого простору та перенесення його у корзину
Основний розвиток	Користувач знаходячись на сторінці творчого простору вибирає компонент натискаючи праву кнопку миші на ньому та у випадаючому меню натискає на пункт видалити
Альтернативні розвитку	Користувач вибирає елемент кліком миші та кнопкою Delete видаляє його
Винятки	Адмін вимкнув можливість взаємодії з компонентом обмеживши можливість користувача або параметри компоненту

Таблиця 5.8 – Сценарій використання

Назва	Редагування компонентів
ID	8
Опис	Користувач може редагувати компоненти на творчому просторі
Актори	користувач з правом редагування, адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Постійно
Тригери	Користувач робить лівий клік миші на елементі та

	обирає потрібний пункт списку
Передумови	Користувач отримує право редагування творчого простору здійснивши запит на отримання прав у адміна та знаходиться у творчому просторі із завантаженими даними обраного кластеру
Постумови	Зміна стану або контенту елементу
Основний розвиток	Користувач знаходячись на сторінці творчого простору вибирає компонент натискаючи ліву кнопку миші на ньому після чого може взаємодіяти з елементом
Альтернативні розвитку	-
Винятки	Адмін вимкнув можливість взаємодії з компонентом обмеживши можливість користувача або параметри компоненту

Таблиця 5.9 – Сценарій використання

Назва	Зміна базових налаштувань компонента
ID	9
Опис	Користувач може редагувати налаштування компоненти на творчому просторі
Актори	користувач з правом редагування, адмін
Вигоди компанії	Гнучка система налаштувань компонентів може використовуватися як сильна сторона застосунку для залучення нових клієнтів

Частота користування	Періодично
Тригери	Користувач користувач робить лівий клік миші на елементі та вибирає пункт меню налаштування
Передумови	Користувач отримує право редагування творчого простору та знаходиться у творчому просторі
Постумови	Зміна параметрів компонентів
Основний розвиток	Користувач користувач робить лівий клік миші на елементі та вибирає пункт меню налаштування
Альтернативні розвитку	-
Винятки	Адмін вимкнув можливість взаємодії з компонентом обмеживши можливість користувача або параметри компоненту

Таблиця 5.10 – Сценарій використання

Назва	Створення команди
ID	10
Опис	Адмін може створити нову команду для ораної компанії
Актори	адмін
Вигоди компанії	Важлива функція менеджменту ролей, яка потрібна

	користувачам в великих командах, може використовуватися як одна із переваг у маркетинговій компанії.
Частота користування	Рідко
Тригери	Адмін натискає на кнопку створити команду, заповнює обов'язкові поля форму та натискає кнопку «створити»
Передумови	Адмін знаходиться на власній сторінці
Постумови	Створення нової команди
Основний розвиток	Адмін знаходячись на власній сторінці натискає на кнопку створити команду
Альтернативні розвитку	Адмін знаходячись на творчому просторі натискає кнопку надати права перегляду команді вибирає пункт створити команду
Винятки	Обмеження підписки, перевищення ліміту команд або при специфічних настройках компанії

Таблиця 5.11 – Сценарій використання

Назва	Видалення користувачі з команд
ID	11
Опис	Адмін може видаляти користувачів із команд
Актори	Адмін

Вигоди компанії	Важлива функція менеджменту ролей, яка потрібна користувачам в великих командах, може використовуватися як одна із переваг у маркетинговій компанії.
Частота користування	Рідко
Тригери	Адмін видалив користувача із команди
Передумови	Адмін переглядає профіль користувача
Постумови	Видалення користувача з команди
Основний розвиток	Адмін в меню підлеглих команд вибирає потрібного користувача натискає праву кнопку миші на у випадаючому меню вибирає пункт Видалити
Альтернативні розвитку	Адмін знаходячись на сторінці творчого простору у меню активних користувачів знаходить потрібного користувача натискає праву кнопку миші на у випадаючому меню вибирає пункт Видалити із команди
Винятки	-

Таблиця 5.12 – Сценарій використання

Назва	Зміна прав підлеглих користувачів
ID	12
Опис	Адмін може видавати та віднімати у користувачів права перегляду та редагування творчого простору

Актори	Адмін
Вигоди компанії	Важлива функція менеджменту ролей, яка потрібна користувачам в великих командах, може використовуватися як одна із переваг у маркетинговій компанії.
Частота користування	Періодично
Тригери	Адмін змінює права користувача
Передумови	Адмін переглядає профіль користувача
Постумови	Зміна прав користувача, що надає користувачеві нові можливості або обмежує його можливості
Основний розвиток	Адмін в меню підлеглих команд вибирає потрібного користувача натискає праву кнопку миші на у випадаючому меню вибирає пункт права вибирає потрібний пункт
Альтернативні розвитку	Адмін знаходячись на власній сторінці вибирає запит користувача на отримання прав та змінює користувачу права
Винятки	-

Таблиця 5.13 – Сценарій використання

Назва	Зміна типу підписки
ID	13
Опис	Адмін може змінити підписку компанії

Актори	Адмін
Вигоди компанії	Надає більш гнучкий спосіб отримання коштів він користувачів, що призведе до збільшення доходів
Частота користування	Рідко
Тригери	Адмін обирає нову підписку, натискає на неї після чого підтверджує зміну типу підписки
Передумови	Адмін знаходячись у власному профілі натискає кнопку «підписка»
Постумови	Надання нових прав, зміна лімітів на творчі простори, команди та налашування
Основний розвиток	Адмін в меню підлеглих команд вибирає потрібного користувача натискає праву кнопку миші на у випадаючому меню вибирає пункт права вибирає потрібний пункт
Альтернативні розвитку	Адмін знаходячись на власній сторінці вибирає запит користувача на отримання прав та змінює користувачу права
Винятки	Незадоволення вимог для зміни підписки

Таблиця 5.14 – Сценарій використання

Назва	Видалення творчих просторів
ID	14

Опис	Адмін може видаляти творчі простори
Актори	Адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Рідко
Тригери	Адмін натискає кнопку видалити на творчому просторі
Передумови	Адмін знаходиться на сторінці доступних творчих просторів
Постумови	Перенесення обраного творчого простору і всіх його даних у корзину та надсилання відповідного повідомлення користувачам
Основний розвиток	Адмін знаходячись на сторінці доступних творчих просторів натискає праву кнопку миші на обраному творчому просторі у випадаючому меню натискає кнопку видалити
Альтернативні розвитку	-
Винятки	Адмін вимкнув можливість взаємодії з компонентом обмеживши можливість користувача або параметри компоненту

Таблиця 5.15 – Сценарій використання

Назва	Створення творчих просторів

ID	15
Опис	Адмін може створювати творчі простори
Актори	Адмін
Вигоди компанії	Частина основного мінімальному функціоналу необхідного для виходу на ринок
Частота користування	Періодично
Тригери	Адмін натискає кнопку створити творчій простір, заповнює обов'язкові поля форми та натискає кнопку «створити»
Передумови	Адмін знаходиться на сторінці доступних творчих просторів
Постумови	Створення нового творчого простору з обраними параметрами, доданими користувачами та зазначеними привілеями
Основний розвиток	Адмін знаходячись на сторінці доступних творчих просторів натискає кнопку додати новий творчій простір і в відкриваючомуся меню вказує потрібні параметри
Альтернативні розвитку	-
Винятки	Обмеження підписки, перевищення ліміту творчих просторів або при специфічних настройках компанії

Таблиця 5.16 – Сценарій використання

Назва	Зміна розширених налаштувань компонента

ID	9
Опис	Користувач може редагувати налаштування компоненти на творчому просторі
Актори	Адмін
Вигоди компанії	Гнучка система налаштувань компонентів може використовуватися як сильна сторона застосунку для залучення нових клієнтів
Частота користування	Періодично
Тригери	Адмін робить лівий клік миші на елементі та вибирає пункт меню налаштування
Передумови	Адмін знаходиться на сторінці доступних творчих просторів
Постумови	Зміна параметрів компонентів, перезавантаження даних компоненту (якщо потрібно), відправлення відповідного повідомлення користувачам
Основний розвиток	Адмін користувач робить лівий клік миші на елементі та вибирає пункт меню налаштування
Альтернативні розвитку	-
Винятки	Адмін вимкнув можливість взаємодії з компонентом обмеживши можливість користувача або параметри компоненту

5.2 Розробка баз даних

В обраній реалізації існують три види баз даних:

1) реляційна база даних для зберігання даних користувача, настройок, та неповних даних творчих просторів. Створена модель описана в наступному розділі;

2) нереляційна mongodb база даних для зберігання повної інформації елементів креативного простору на бекенд частині;

3) indexeddb база даних для зберігання фронтенд частині застосунку отриманих з бекенду даних користувачем, що дозволяє позбутися зайвих запитів, а також дозволяє реалізувати PWA підхід.

5.3 Опис концептуальної моделі даних

Проектування бази даних представляє собою процес, який складається з декількох етапів, послідовність виконання яких забезпечує коректність та ефективність процесів роботи з даними протягом подальшої експлуатації бази даних. Для досліджуваної предметної області побудовано ER-діаграму представлену на рисунку 5.1. ER-діаграма призначена для аналізу предметної області та початкового етапу проектування БД, в якому визначаються найголовніші сутності, над якими надалі потрібно буде визначити атрибути та зв’язки.

В результаті аналізу предметної області виявлено основні сутності, опис яких необхідно здійснити при проектуванні.

Проектування бази даних здійснено за допомогою методу «сутність - зв'язок». При проведенні аналізу предметної області виділено такі сутності:

- company;
- role;
- user;
- team;
- whiteboard;
- cluster;
- item_type;

– item;

– whiteboard-item.

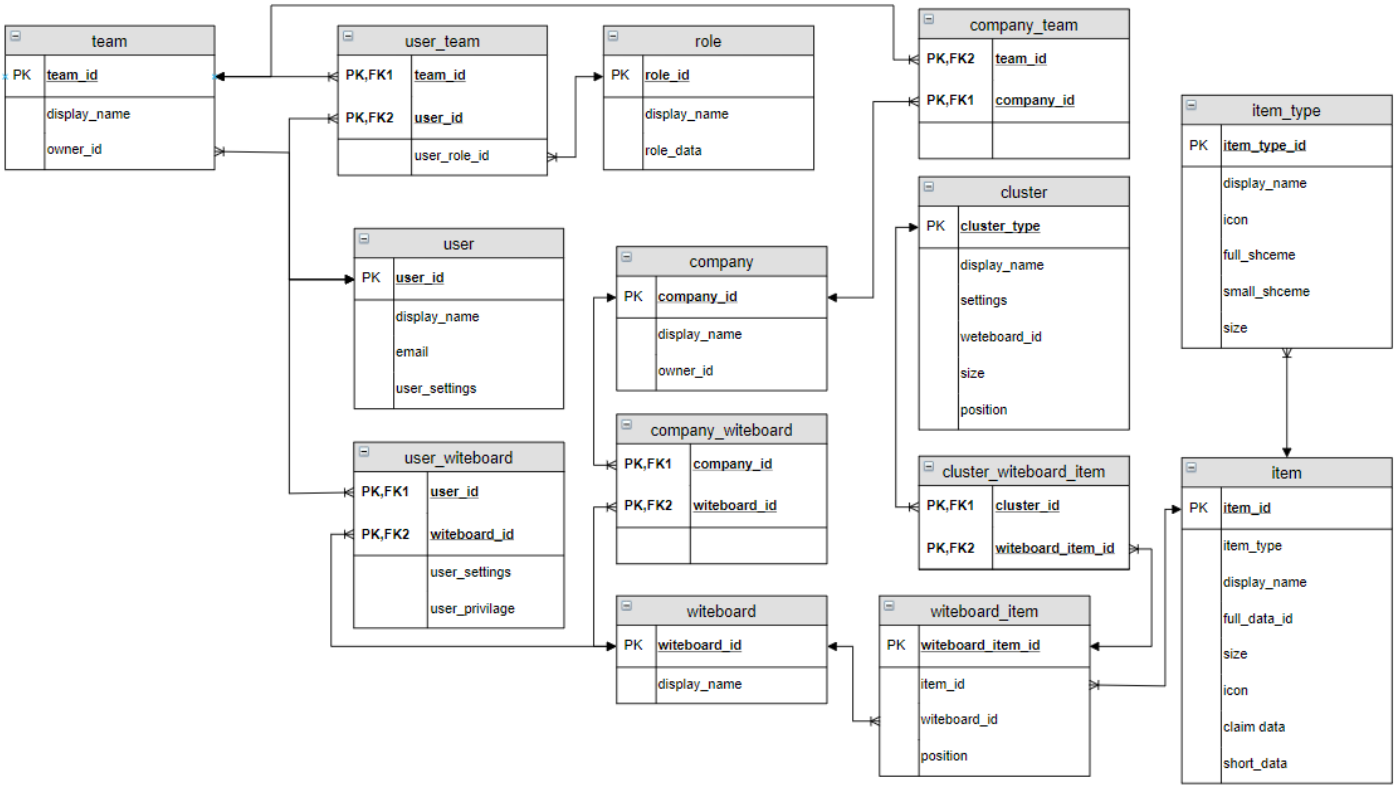

Рисунок 5.1 – ER-діаграма

Для всіх зазначених сутностей виділені атрибути – властивості, характерні для всіх екземплярів певної сутності. Для забезпечення унікальності записів в таблицях баз даних екземпляру кожної сутності відповідає унікальний код.

На рисунку 5.2 зображено сутність для збереження даних про компанію, яка використовується для управління групами та підписками.

На рисунку 5.3 зображено сутність для збереження даних про роль доступу. Атрибут role data це поле для зберігання привілеїв доступу кожної ролі. В цьому атрибуті зберігаються елементи дерева ролей, які використовуються при перевірці на доступ до творчих просторів, віджетів та параметрів.

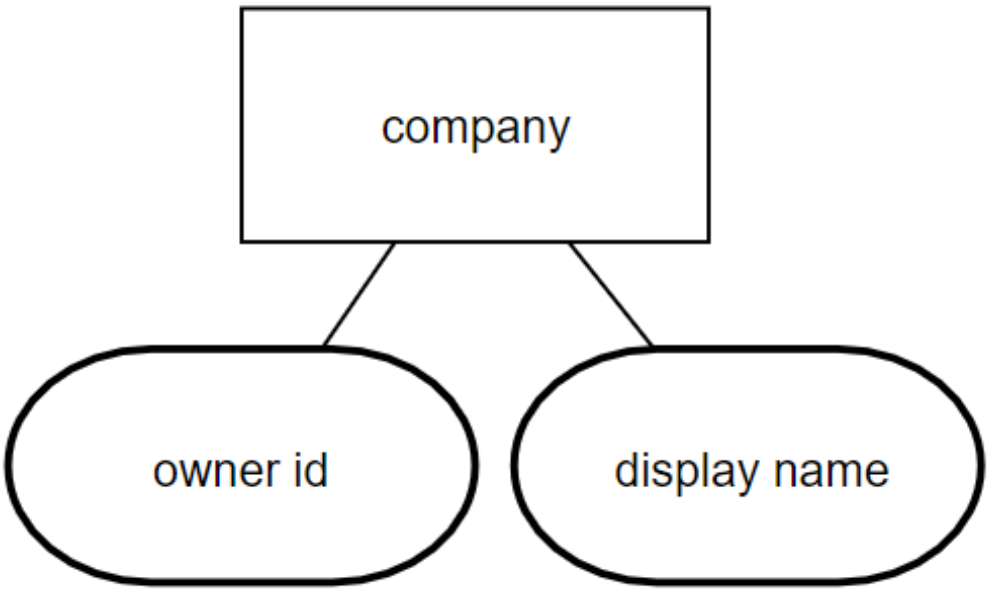

Рисунок 5.2 – Атрибути сутності «company»

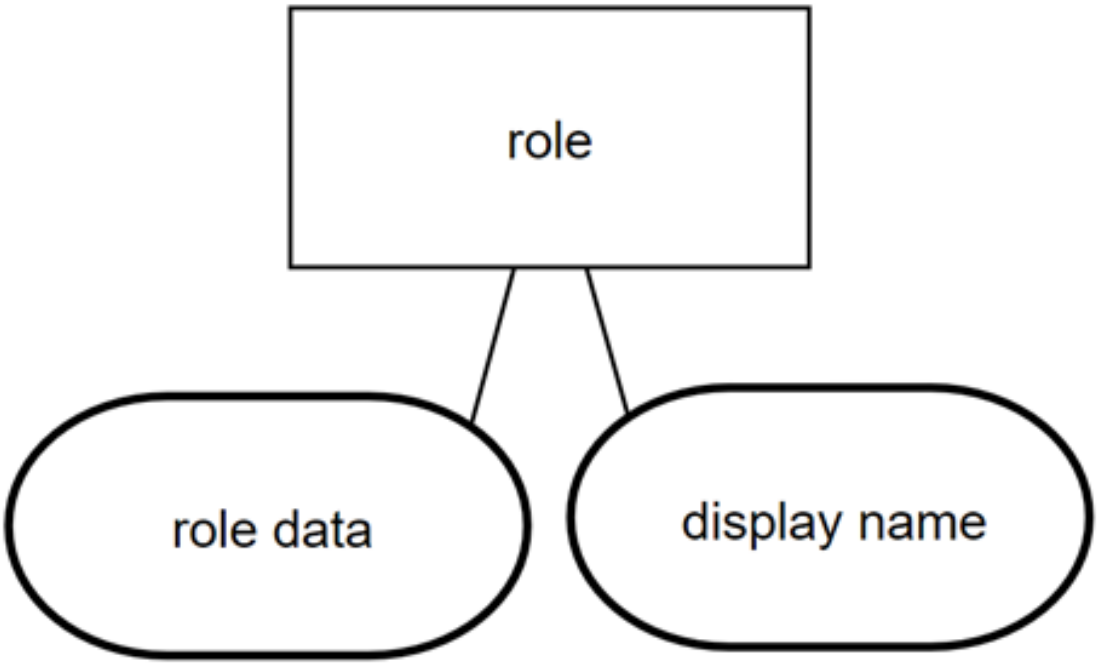

Рисунок 5.3 – Атрибути сутності «role»

На рисунку 5.4 зображено сутність для зберігання даних про користувача: його електронна пошта, набір настройок користувацького інтерфейсу у вигляді json словника параметрів.

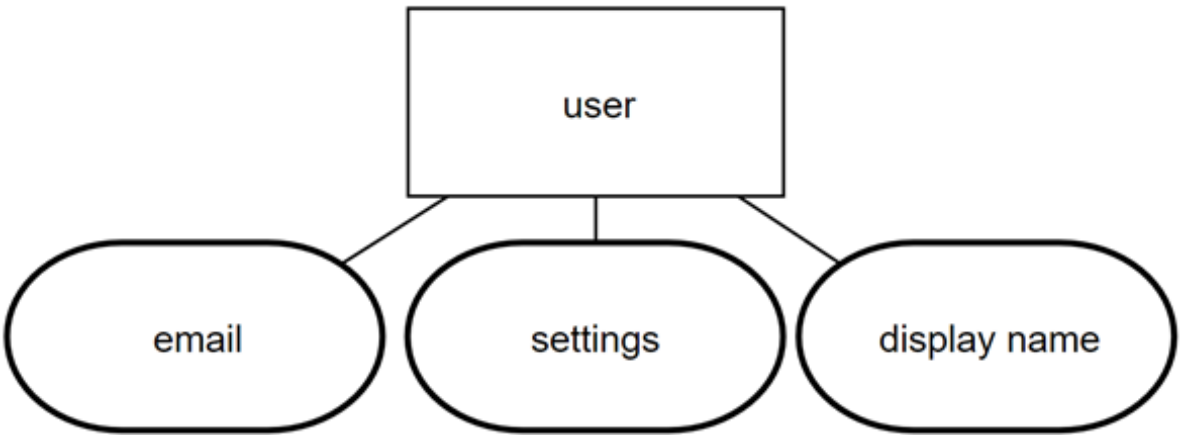

Рисунок 5.4 – Атрибути сутності «user»

На рисунку 5.5 зображено сутність команда, яка використовується для управління групами та підписками.

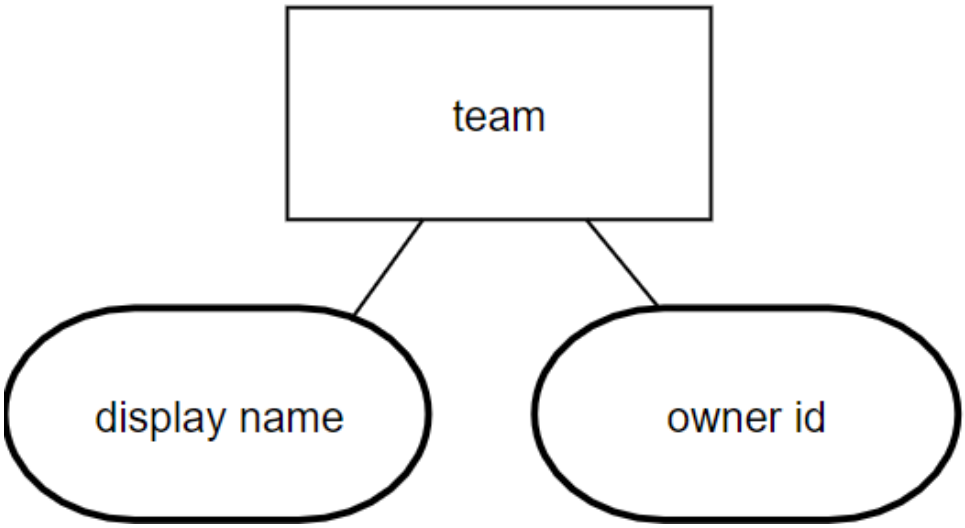

Рисунок 5.5 – Атрибути сутності «team»

На рисунку 5.6 зображено сутність дошки, яка використовується для управління групами та підписками. Атрибут settings використовується для зберігання json словника стандартизованих налаштувань дошки, які є однаковими для всіх дошок. Атрибут claim data зберігає унікальний ключ словника динамічних налаштувань, які зберігаються в документній базі даних та використовуються для зберігання нестандартних налатувань сторонніх бібліотек і вітжетів, які використовуються на конкретній дошці.

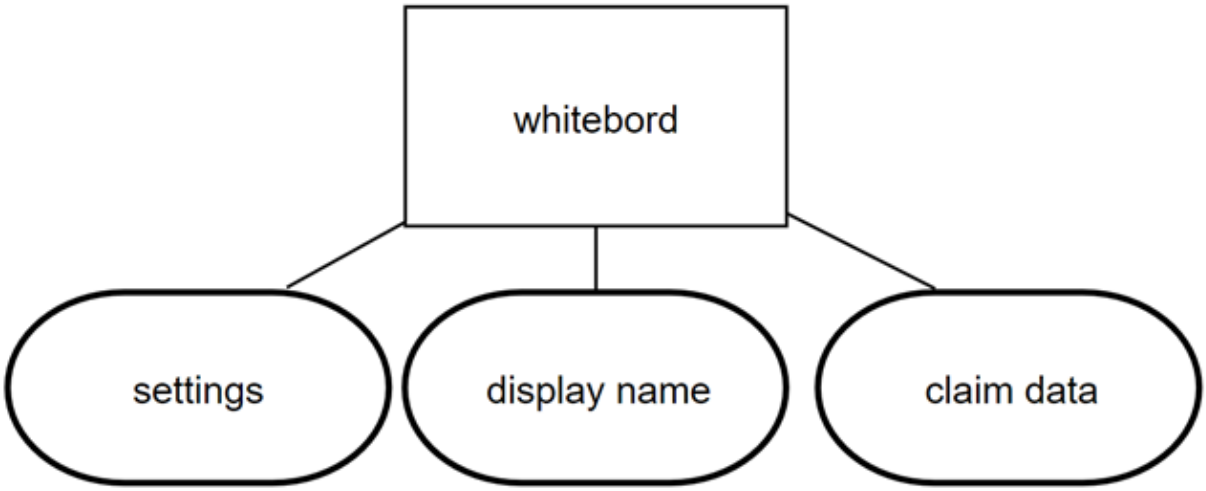

Рисунок 5.6 – Атрибути сутності «whiteboard»

На рисунку 5.8 зображено сутність кластеру, що зберігає дані про кластер який заміняє собою групу елементів на конкретній дошці. Атрибут claim data маж сходу поведінку до такого ж елементу таблиці дошок.

На рисунку 5.9 зображено сутність типа компоненти, яка використовується як основа компоненту дошки. Атрибут small scheme та full scheme – це поля для зберігання схеми параметрів у вигляді таблиці клейм ключів для типу компонентів які використовуються при частковому та повному відображенні компоненту.

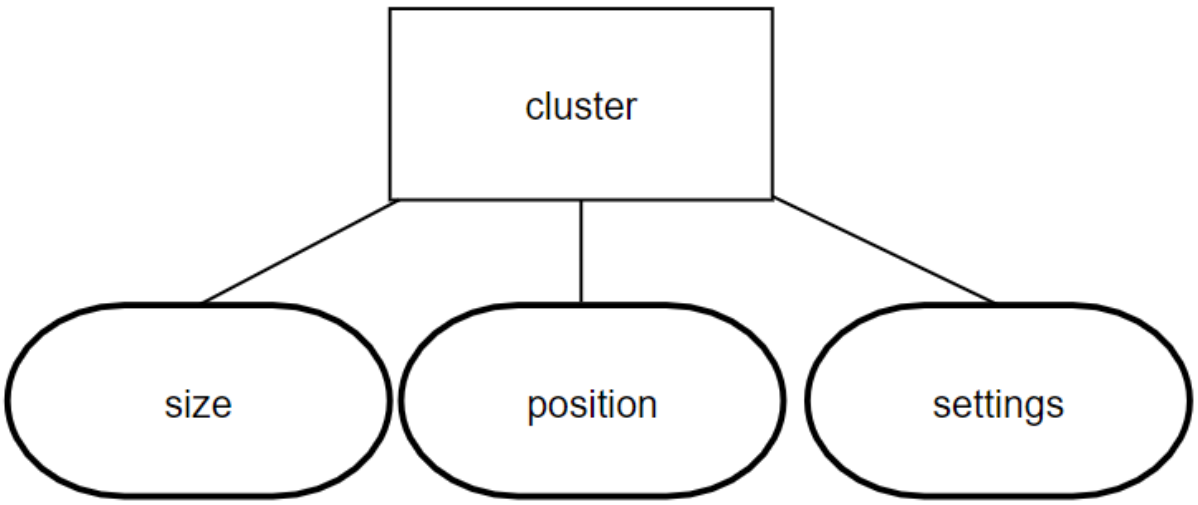

Рисунок 5.8 – Атрибути сутності «cluster»

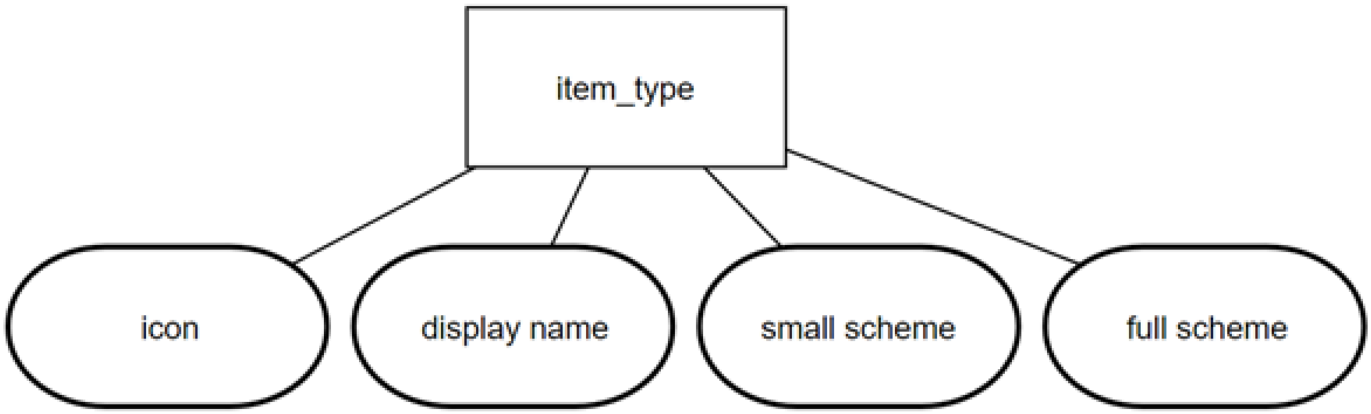

Рисунок 5.9 – Атрибути сутності «item type»

Наведені на рисунку 5.10 атрибут small data це поле для зберігання значень параметрів для таблиці клеймів які при об'єднанні з полем small scheme з попередньої таблиці дають повний опис компоненту який використовуються при частковому відображенні елементу для користувача.

Наведені на рисунку 5.10 атрибут full data id це поле для id повного документу в документній базі даних. В цьому документі зберігається таблиця клеймів які при об'єднанні з полем full scheme з попередньої таблиці дають повний опис компоненту який використовуються при повному відображенні елементу для користувача.

Наведені на рисунку 5.10 атрибут claim data це поле для зберігання json даних налаштувань елементу.

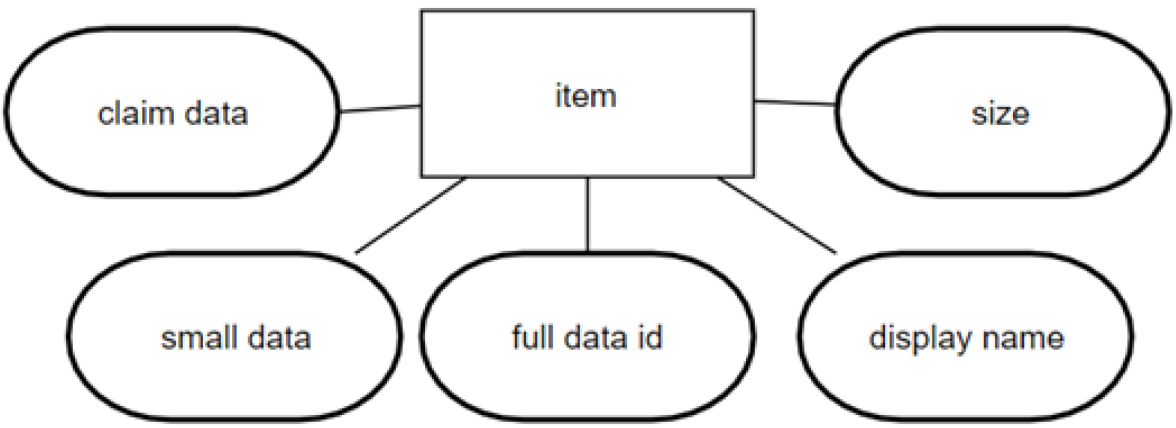

Рисунок 5.10 – Атрибути сутності «item»

Наведені на рисунку 5.10 атрибут full data id це поле для id повного документу в документній базі даних. В цьому документі зберігається таблиця клеймів які при об'єднанні з полем full scheme з попередньої таблиці дають повний опис компоненту який використовуються при повному відображенні елементу для користувача.

Наведені на рисунку 5.10 атрибут claim data це поле для зберігання json даних налаштувань елементу.

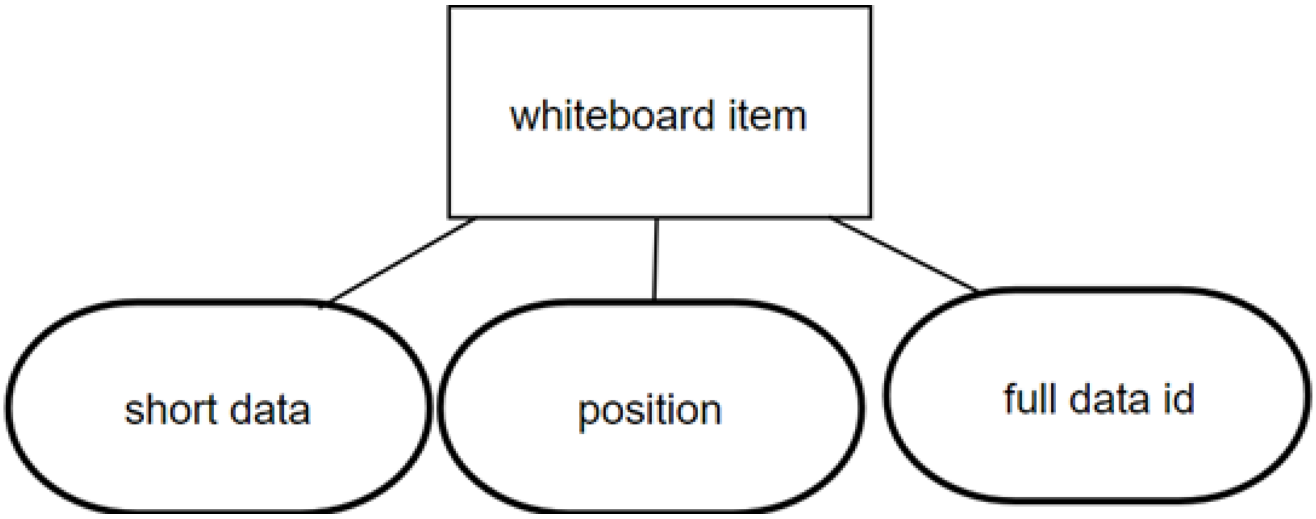

Рисунок 5.11 – Атрибути сутності «whiteboard item»

Зважаючи на наявність атрибутів, які можуть виступати в якості підпорядкованих сутностей, задля забезпечення зв'язку багато до багатьох до моделі даних додано додаткові сутності:

- company team;
- user team;
- user whiteboard;
- company whiteboard;
- cluster whiteboard item.

Для подальшого визначення фізичної структури бази даних та способів адресації зовнішніх ключів згідно методу проектування “сутність-зв’язок” (рисунки 5.10-5.20) при побудові ER-діаграм виникає необхідність встановлення ступенів зв’язків між сутностями.

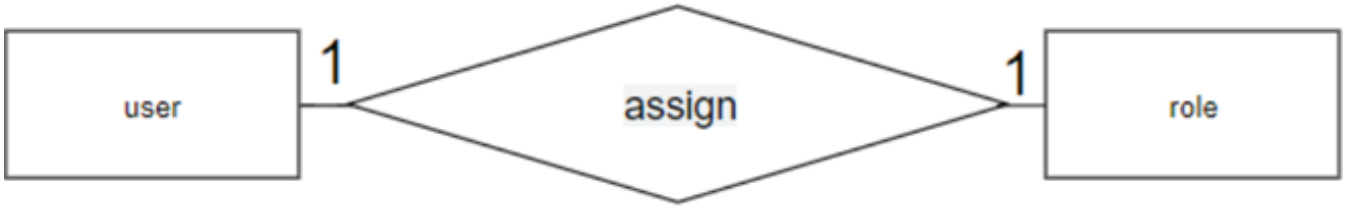

Рисунок 5.11 – Зв'язок між сутностями «user» та «role»

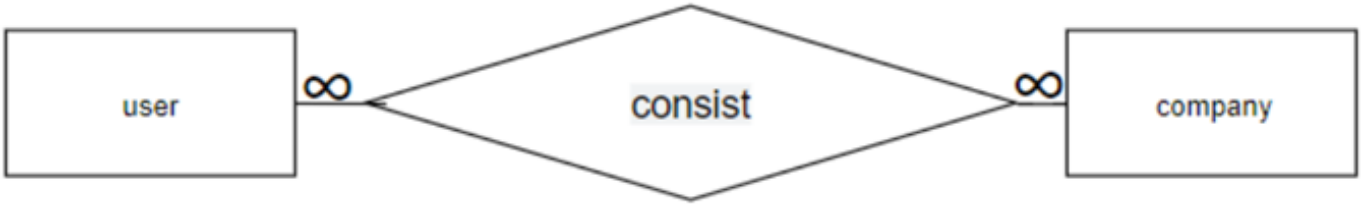

Рисунок 5.12 – Зв'язок між сутностями «user» та «company»

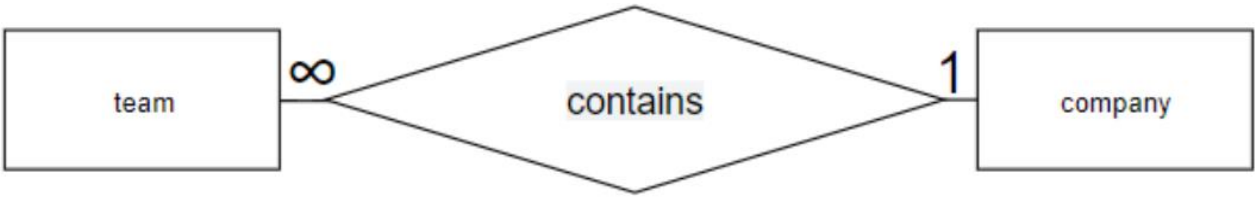

Рисунок 5.13 – Зв'язок між сутностями «team» та «company»

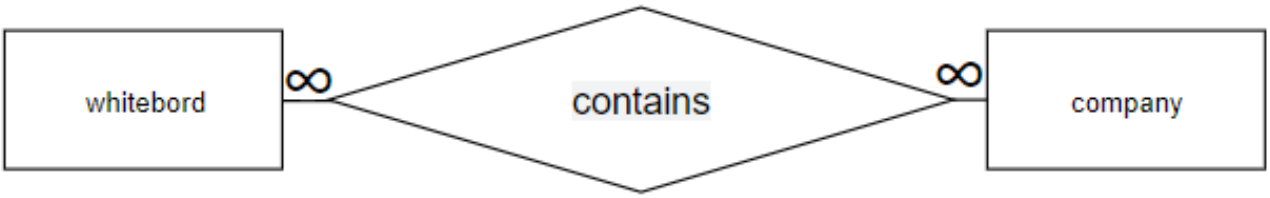

Рисунок 5.14 – Зв'язок між сутностями «whiteboard» та «company»

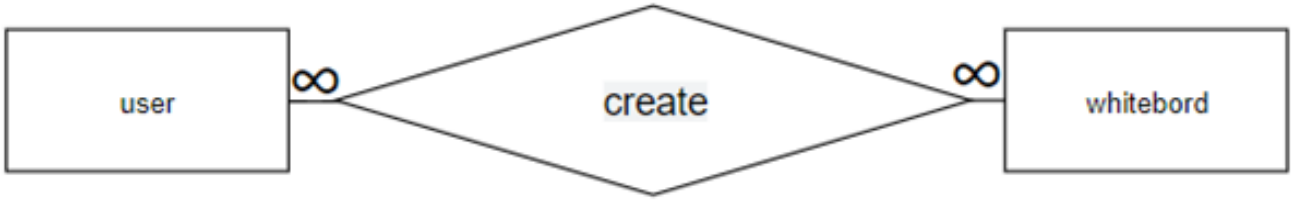

Рисунок 5.15 – Зв'язок між сутностями «user» та «whiteboard»

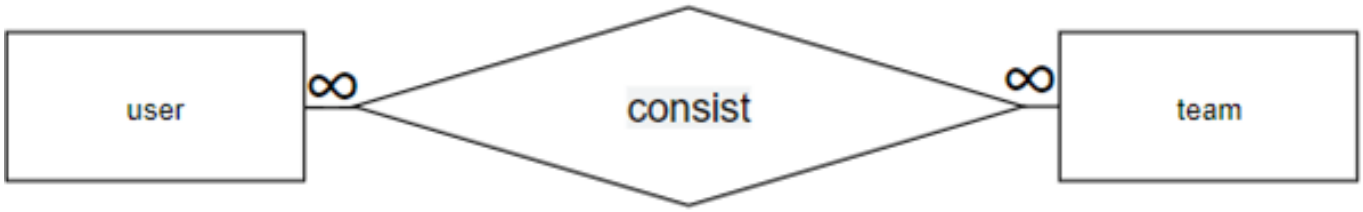

Рисунок 5.16 – Зв'язок між сутностями «user» та «team»

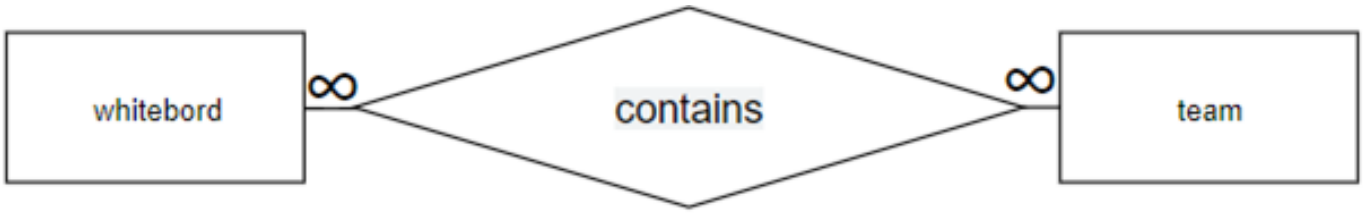

Рисунок 5.17 – Зв'язок між сутностями «whiteboard» та «team»

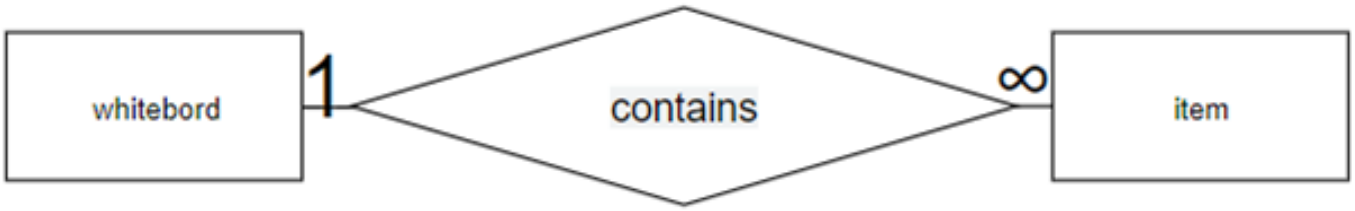

Рисунок 5.18 – Зв'язок між сутностями «whiteboard» та «item»

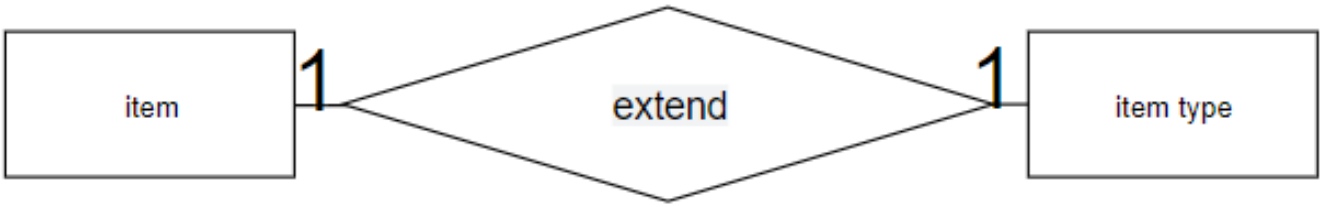

Рисунок 5.19 – Зв'язок між сутностями «item» та «item type»

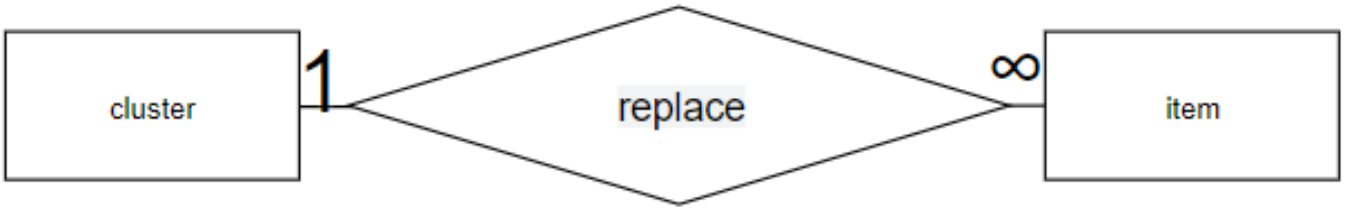

Рисунок 5.20 – Зв'язок між сутностями «cluster» та «item»

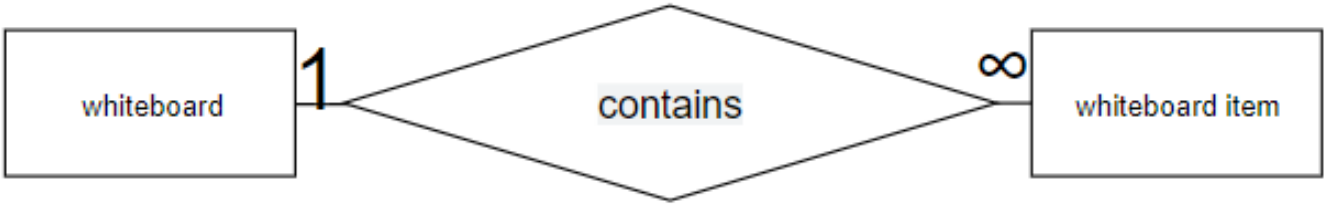

Рисунок 5.21 – Зв'язок між сутностями «whiteboard» та «whiteboard item»

В цілому ER-діаграма надає можливість побачити початкову структуру моделі даних, що будуть використовуватися при розробці платформі та наглядно дозволяє переглянути та проаналізувати всі сутності та їх атрибути на предмет необхідності в даній платформі та за потребою додавання нових елементів на етапі концептуального проектування набагато легше ніж на етапі фізичного проектування бази даних.

5.4 Робота з canvas

Елемент canvas[28, 29] є частиною HTML5 і дозволяє динамічно зображати 2D форми та растрові зображення. Це низька, процедурна модель, яка оновлює растрову карту і не має вбудовані графічні сцени також через WebGL він дозволяє відображати 3D форми та зображення.

Зараз ця технологія підтримується найпопулярнішими браузерами та платформами (таблиця 5.17).

Для відображення фігур на веб-сторінці потрібно лише декілька рядків коду.

Наступний код створює на сторонці червоний квадрат.

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = 'red';
context.fillRect(30, 30, 50, 50);
```

Таблиця 5.17 – Підтримка canvas браузерами

IE	Firefox	Safari	Chrome	Opera	iOS	Android
9.0+	3.0+	3.0+	3.0+	10.0+	3.0+	1.0+

Елемент canvas[30] надає скриптам бітову карту, залежну від розширення екрану, яка може бути використана для візуалізації графіків, ігрової графіки, мистецтва чи інших візуальних зображень на льоту.

Автори не повинні використовувати елемент полотна в документі, коли є більш підходящий елемент. Наприклад, недоцільно використовувати елемент полотна для візуалізації заголовка сторінки: якщо бажане представлення заголовка є графічно інтенсивним, воно повинно бути розміщене за допомогою відповідних елементів (як правило, h1), а потім оформлене за допомогою CSS та підтримуючих технологій, таких як тіньові дерева.

Коли автори використовують елемент canvas, вони також повинні надавати вміст, який, коли він представлений користувачеві, передає по суті ту ж функцію або призначення, що і растровий малюнок полотна. Цей вміст може бути розміщений як вміст елемента полотна. Вміст елемента canvas, якщо такий є, є резервним вмістом елемента.

У інтерактивних візуальних носіях, якщо сценарій для елемента полотна увімкнено, і якщо підтримка елементів полотна включена, то елемент полотна представляє вбудований вміст, що складається з динамічно створеного зображення, растрового зображення елемента.

У неінтерактивних, статичних, візуальних носіях, якщо елемент полотна раніше був пов’язаний з контекстом візуалізації (наприклад, якщо сторінку переглядали в інтерактивному візуальному носії та зараз друкується, або якщо якийсь сценарій, який виконувався під час макета сторінки процес, намальований на елементі), тоді елемент полотна представляє вбудований вміст з поточним растровим зображенням та розміром елемента. В іншому випадку елемент натомість представляє його резервний вміст.

У візуальних засобах масової інформації та у візуальних медіа, якщо сценарій вимкнено для елемента полотна або якщо підтримка елементів полотна відключена, елемент полотна замість цього представляє вміст резервного копіювання.

Коли елемент полотна представляє вбудований контент, користувач все ще може зосередити нащадків елемента полотна (у резервному вмісті). Коли елемент зосереджений, він є ціллю подій взаємодії клавіатури (навіть якщо сам елемент не видно). Це дозволяє авторам зробити доступну інтерактивну канву клавіатури: автори повинні мати індивідуальне відображення інтерактивних регіонів на зону, що фокусується, у резервному вмісті.

Елемент, найближчий прародитель елемента якого є рендерингом і представляє вбудований вміст, це елемент, який використовується як відповідний вміст резервного полотна полотна.

Елемент canvas має два атрибути для управління розміром растрової карти елемента: ширина та висота. Ці атрибути, якщо вони вказані, повинні мати значення, які є дійсними невід'ємними цілими числами. Для отримання їх чисельних значень необхідно використовувати правила розбору невід’ємних цілих чисел. Якщо атрибут відсутній, або при розборі його значення повертається помилка, тоді замість нього слід використовувати значення за замовчуванням. Атрибут ширини за замовчуванням до 300, а атрибут висоти - 150.Під час встановлення значення атрибуту ширини або висоти, якщо контекстний режим елемента полотна встановлений на заповнення, агент користувача повинен кинути DOMException "InvalidStateError" і залишити значення атрибуту незмінним.

Власні розміри елемента полотна, коли він представляє вбудований вміст, дорівнюють розмірам растрової карти елемента.

Агент користувача повинен використовувати щільність квадратного пікселя, що складається з одного пікселя даних зображення на одиницю простору координат для растрових зображень полотна та його контекстів візуалізації.

Растрові елементи полотна, растрові зображення об'єктів ImageBitmap, а також деякі растрові графіки контекстів візуалізації, такі як описані в розділах на об'єктах CanvasRenderingContext2D та ImageBitmapRenderingContext.

Елемент полотна може мати прив’язаний до нього контекст візуалізації. Спочатку він не має обмеженого контексту візуалізації. Щоб відслідковувати, чи є у нього контекст візуалізації чи ні, і що це за контекст візуалізації, на полотні також є режим контекстного полотна, який спочатку не є, але його можна змінити або на заповнювач, 2d, bitmaprenderer, webgl або webgl2 за алгоритмами, визначеними в цій специфікації.

Якщо режим контексту канви не дорівнює, елемент полотна не має контексту візуалізації, і його растрова карта повинна бути прозорою чорного кольору з внутрішньою шириною, що дорівнює числовому значенню атрибуту ширини елемента та внутрішньої висоти, рівній числовому значенню висоти елемента. атрибута, ці значення інтерпретуються у пікселях CSS та оновлюються, коли атрибути встановлюються, змінюються чи видаляються.

Коли режим контексту полотна є заповнювачем, елемент полотна не має контексту візуалізації. Він служить заповнювачем для об’єкта OffscreenCanvas, а вміст елемента полотна оновлюється за допомогою виклику методу commit () контексту візуалізації об'єкта OffscreenCanvas.

Коли елемент полотна представляє вбудований вміст, він надає джерело фарби, ширина якого – це внутрішня ширина елемента, висота якої - внутрішня висота елемента, а зовнішній вигляд – растровий елемент елемента.

Щоразу, коли атрибути вмісту ширини та висоти встановлюються, видаляються, змінюються або надмірно встановлюються у них вже є значення, тоді користувальницький агент повинен виконувати дію з рядка наступної таблиці, що відповідає контекстному режиму елемента полотна.

5.4.1 Методи оптимізації canvas

Далі наведено використані методи оптимізації canvas елементів.

Метод використання примітивів використовується, якщо часто потрібно повторювати одні та ті ж операції малювання на кожному анімаційному кадрі то можна завантажити їх на позаекранне полотно. Потім можна відображати зображення

на екрані на основному полотні так часто, як це потрібно, без зайвого повторення кроків, необхідних для його створення.

```
myCanvas.offscreenCanvas = document.createElement('canvas');
myCanvas.offscreenCanvas.width = myCanvas.width;
myCanvas.offscreenCanvas.height = myCanvas.height;
myCanvas.getContext('2d').drawImage(myCanvas.offScreenCanvas, 0, 0);
```

Заміна чисел з плаваючою комою на цілі значення використовуються для уникнення субпікселів. Наявність субпікселів змушує браузер робити додаткові обчислення для створення ефекту згладжування. Щоб уникнути цього можна округлити всі координати, які використовуються у викликах для drawImage (), наприклад, використовуючи Math.floor ().

Можна кешувати різні розміри своїх зображень у під час завантаження, а не виконувати постійне масштабування їх у drawImage () для уникнення масштабування малюнку.

Використання багатошарових полотен для складних сцен у випадках коли деякі об’єкти потрібно часто переміщувати або змінювати, а інші залишаються відносно статичними. Можлива оптимізація за допомогою розбиття елементів на різні шари, використовуючи кілька елементів canvas.

Наприклад, у випадку з грою інтерфейс користувача знаходиться зверху, інтерактивні об’єкти посередині та статичний фон знизу. У цьому випадку можна розділити гру на три шари canvas. Користувальницький інтерфейс мінятиметься лише після введення користувача, ігровий шар змінюватиметься з кожним новим кадром, а фон залишатиметься незмінним.

```
<div id="stage">
 <canvas id="ui-layer" width="480" height="320"></canvas> <canvas id="game-layer" width="480" height="320"></canvas>
<canvas id="background-layer" width="480" height="320"></canvas>
```

```
 </div>
<style>
#stage { width: 480px; height: 320px; position: relative; border: 2px solid black; }
canvas { position: absolute; }
#ui-layer { z-index: 3; }
#game-layer { z-index: 2; }
#background-layer { z-index: 1; }
</style>
```

Використовування звичайного CSS для великих фонових зображень, якщо потрібно використовувати статичне фонове зображення, можна намалювати його на простому елементі , використовуючи властивість фонового зображення CSS і розмістити його під полотном. Це усуне необхідність відтворення фону на полотні на кожному кадрі.

Масштабування полотна за допомогою перетворень CSS відбуваються швидше, оскільки вони використовують GPU. Найкращий випадок - це не масштабувати полотно, або мати менше полотно і збільшувати його масштаб, а не більше полотно та зменшувати його.

```
var scaleX = window.innerWidth / canvas.width;
var scaleY = window.innerHeight / canvas.height;
var scaleToFit = Math.min(scaleX, scaleY);
 var scaleToCover = Math.max(scaleX, scaleY); stage.style.transformOrigin = '0 0';
//scale from top left stage.style.transform = 'scale(' + scaleToFit + ')';
```

Вимкнення прозорість, якщо на веб-сторінці використовується полотно і не потрібен прозорий фон можна встановити у альфа-параметру значення false, коли створюєте контекст малювання за допомогою HTMLCanvasElement.getContext (). Ця інформація може використовуватися веб-переглядачем для оптимізації візуалізації.

```
var ctx = canvas.getContext('2d', { alpha: false });
```

5.5.2 Реалізація роботи з канвасом

Для отримання робочого прототипу потрібно реалізувати наступну логіку

1) базова drag and drop логіка. Реалізовано за допомогою Paper.js;
2) автоматична кластеризація;
3) перемикання між кластером і набором елементів;
4) перемикання між мінімальною та максимальним виглядом.

Автоматична кластеризація реалізовано за допомогою алгоритму а-квазіеквівалентності[31, 32] яка працює наступним чином:.

Є множина X, кількість елементів у множині - Q.

1) необхідно побудувати матрицю оцінок відстаней між елементами (нормальними заходами подібності) за допомогою формули:

$$\mu_{x_q}(x_i) = 1 - \frac{d(x_q, x_i)}{\max\limits_{k\epsilon[1,Q]}\left(d(x_q, x_k)\right)}, q, i = 1, Q$$

де d (x, y) - відстань Евкліда;

2) далі побудуються відносні міри схожості за допомогою формули:

$$£_{x_q}(x_i, x_j) = 1 - \left|\mu_{x_q}(x_i) - \mu_{x_q}(x_j)\right|, i, j, q = 1, Q$$

Отримана тривимірна матриця показує, наскільки схожа пара даних елементів щодо всіх інших;

3) побудова матриці заходів подібності елементів множини за формулою:

$$£(a, b) = T\left(£_{x_1}(a, b), \dots, £_{x_q}(a, b)\right) = \max_{i=1,Q} £_{x_1}(a, b)\ a, b\epsilon X$$

З тривимірної матриці отримується двомірна, яка містить гірші оцінки заходів подібності для кожної пари точок;

4) для отриманої матриці R побудується її замикання, використовуючи такі формули:

$$R_{£}^{q} = R_{£}^{q-1} * R_{£}$$

Визначення для операції замикання:

$$A = \{a_{lk}\}_{\substack{i=1\dots N \\ k=1\dots Q}}, B = \{b_{kj}\}_{\substack{j=1\dots M \\ k=1\dots Q}}, A * B = \{c_{ij}\}_{\substack{i=1\dots N \\ j=1\dots M}}$$

$$c_{ij} = S(T(a_{i1}, b_{1j}), \dots, T(a_{iQ}, b_{1Q}))$$

У випадку S = max, T = min.

Блок схема та процесу виглядає наступним чином

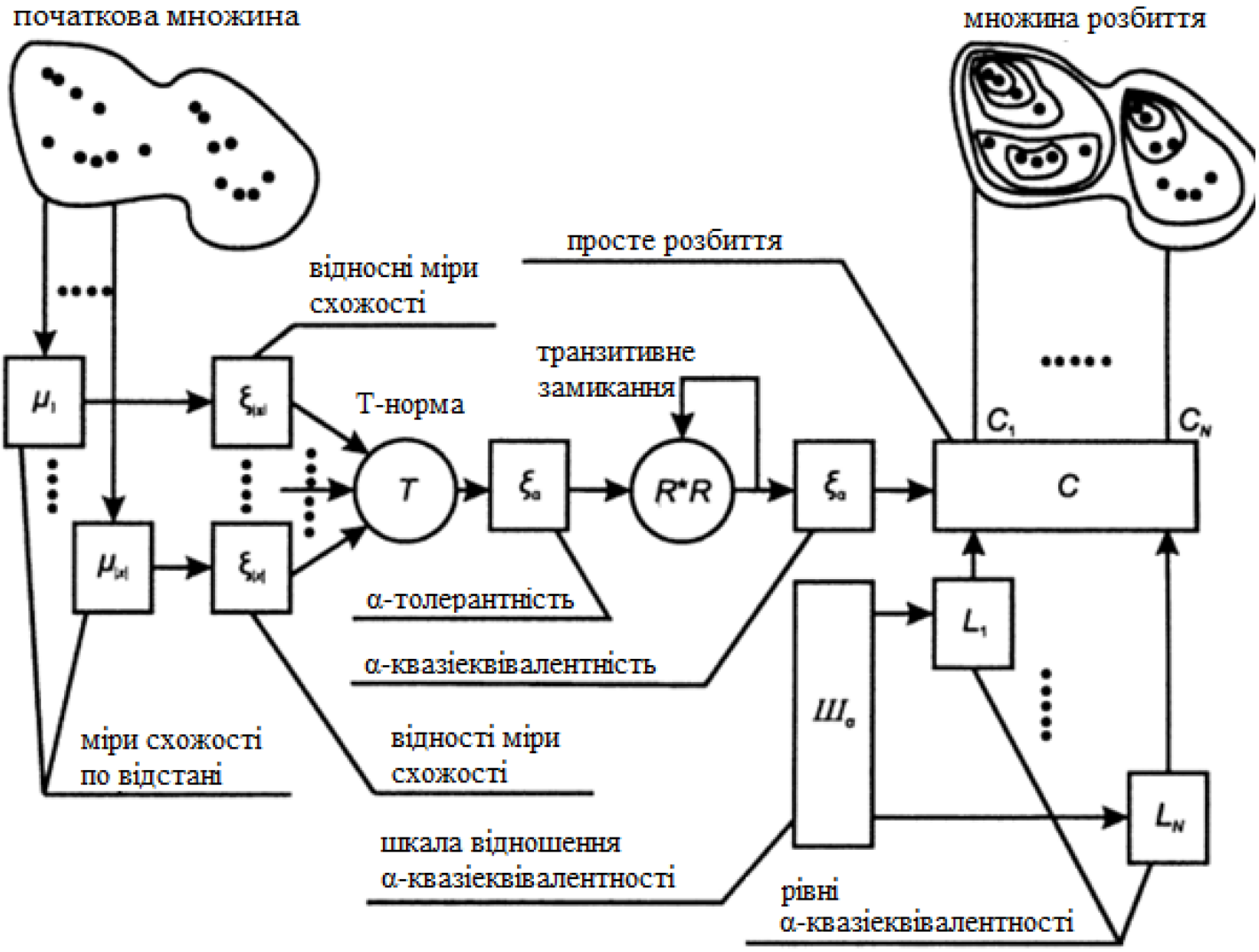

Рисунок 5.22 – Блок схема кластеризації

Наділі отримані розбиття будуть використовуватися для перемикання між кластером і набором елементів.

Блок схема реалізації переходу від кластеру до елементів відображено на рисунку 5.23.

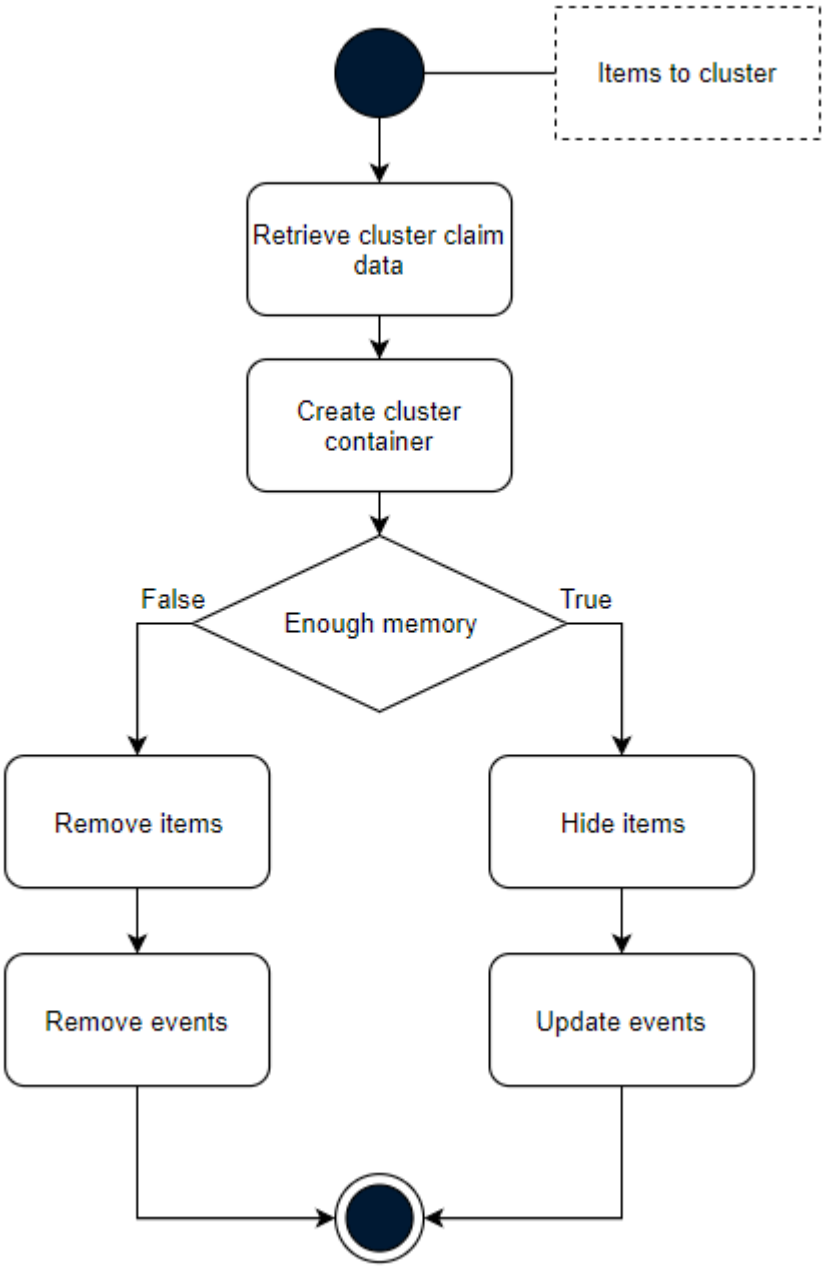

Рисунок 5.23 – Блок схема переходу від кластеру до елементів

Блок схема реалізації переходу від елементів до кластера відображено на рисунку 5.24.

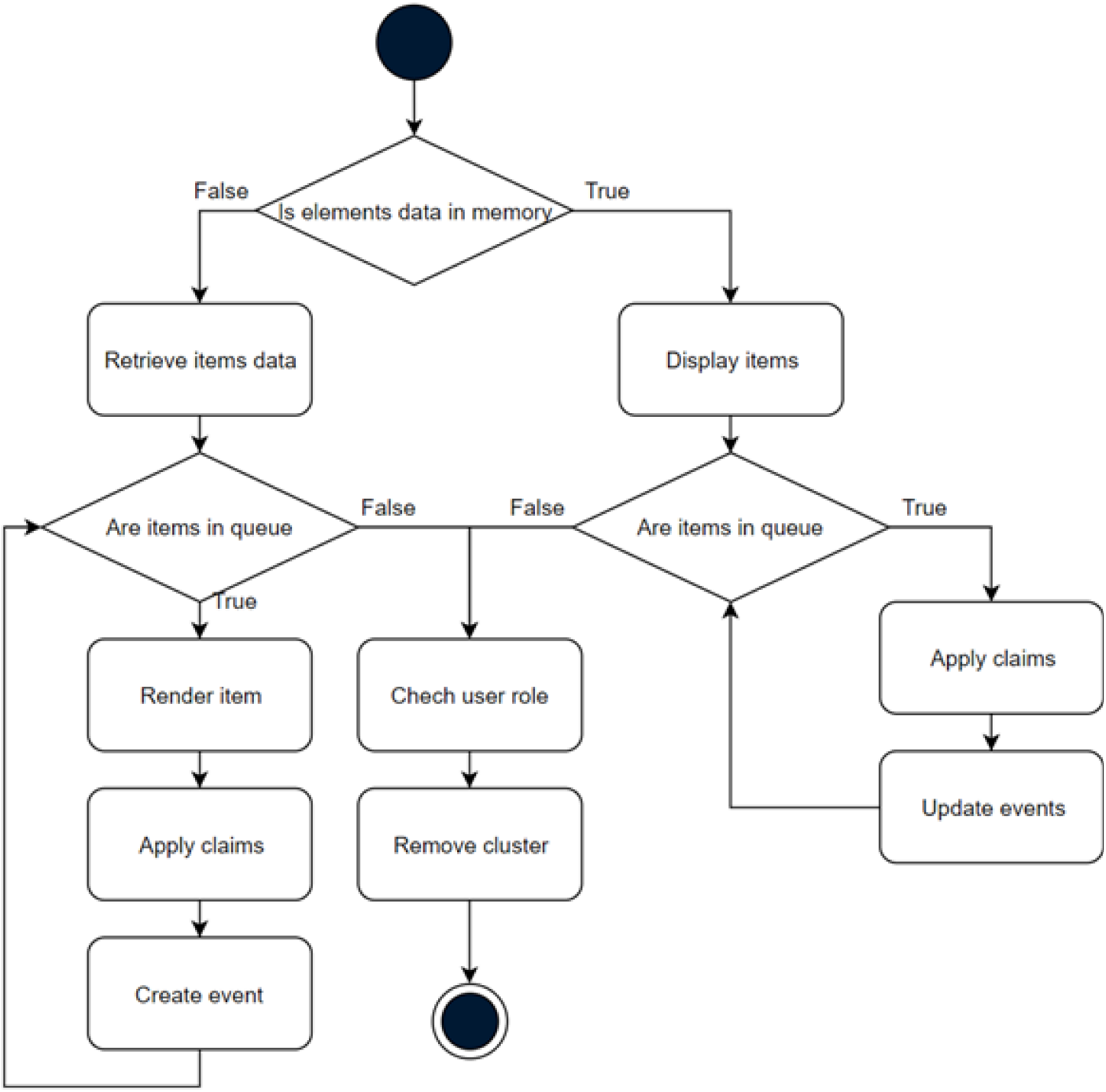

Рисунок 5.24 – Блок схема переходу від елементів до кластера

Блок схема реалізації перемикання між мінімальною та максимальним виглядом відображено на рисунку 5.25.

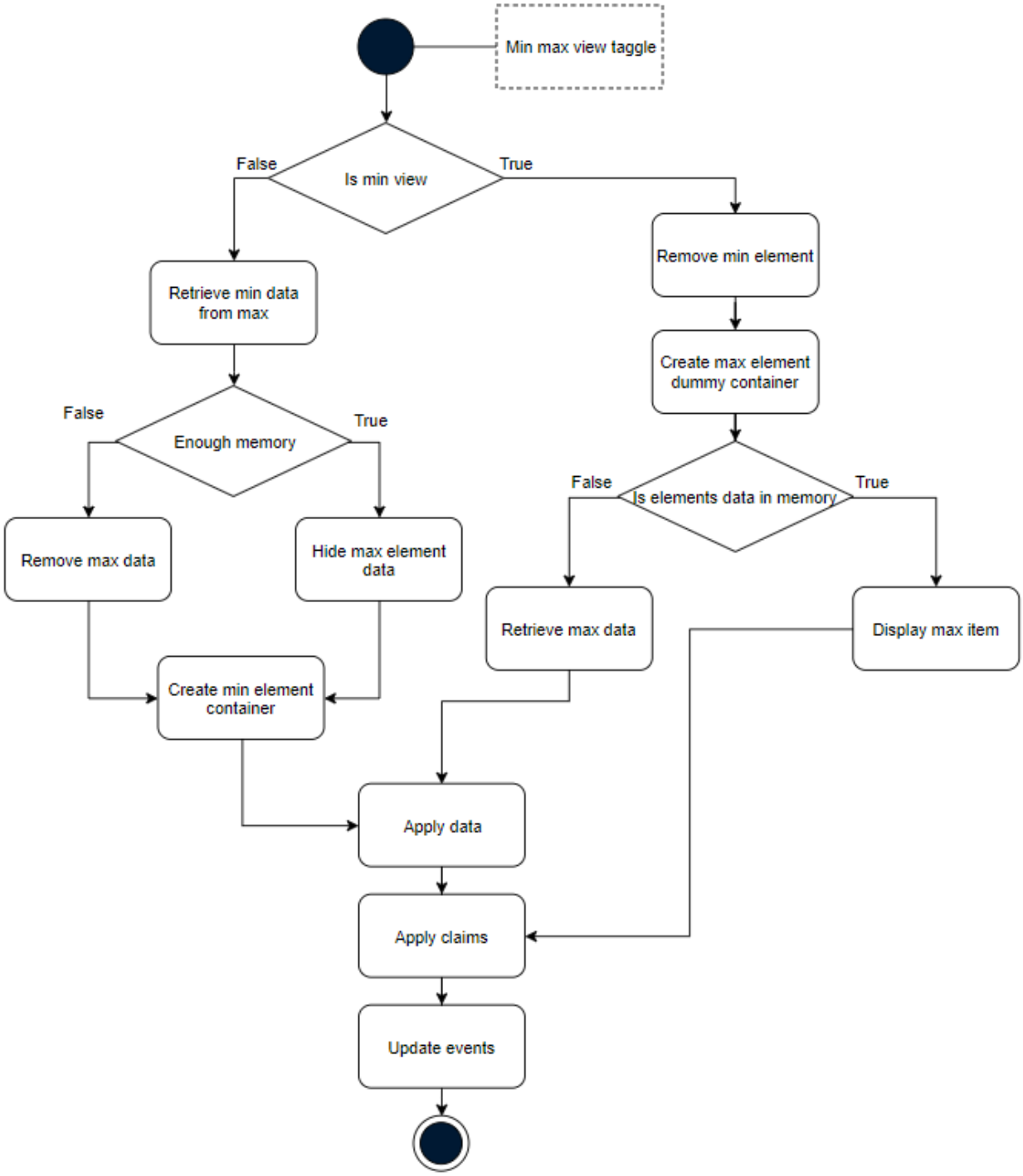

Рисунок 5.25 – Блок схема перемикання між мінімальною та максимальним виглядом

5.6 Розроблення користувацького інтерфейсу

Для виконання замірів продуктивності застосунку з вищеописаними методами було створено демо проект.

Було побудовано мінімальний користувацький інтерфейс за допомогою наступних бібліотек та фреймворків:

– Vue.js – фреймоврк оснований на компонентній архітектурі, який був використаний для побудови віджетів;

– Boostarap – бібліотека стилів для пришвидшення розробки користувацького інтерфейсу;

– Fabric.js – бібліотека для роботи з canvas;

– Dexie.js – бібліотека для облегшення роботи з InedxDb.

На рисунку 5.26 показано користувацький інтерфейс творчого простору та на рисунку 5.27 інтерфейс профілю користувача.

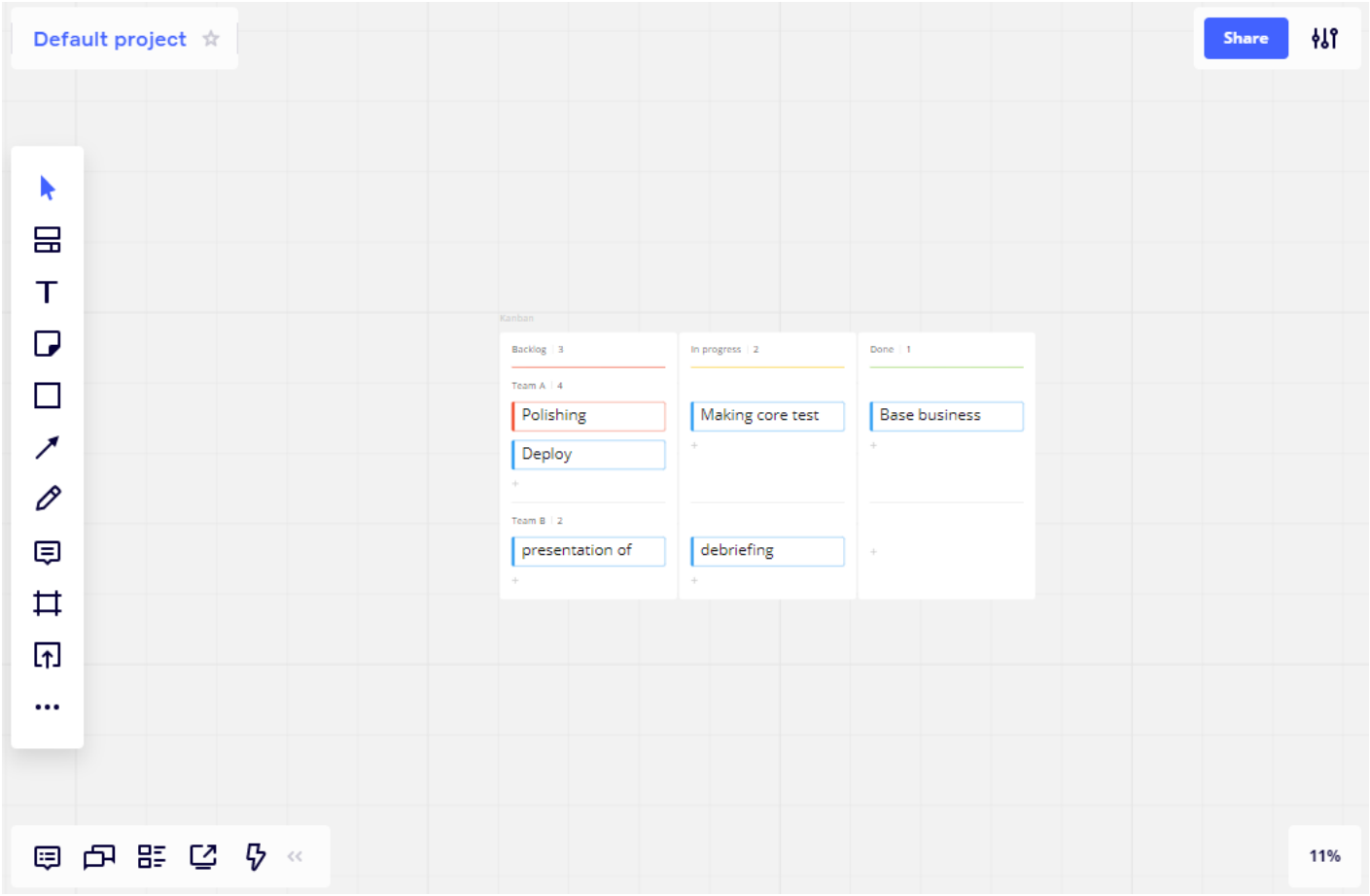

Рисунок 5.26 – Користувацький інтерфейс творчого простору

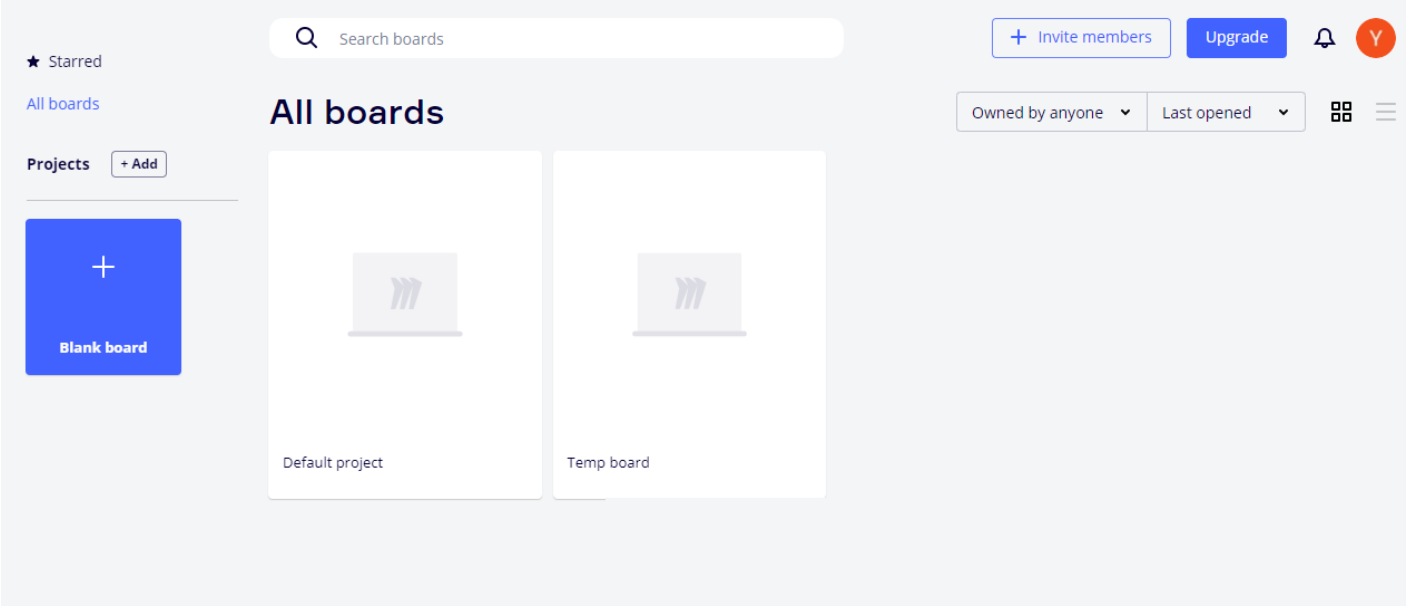

Рисунок 5.27 – Користувацький інтерфейс профілю користувача

Користувацький інтерфейс реалізує всі зазначені функціональні вимоги зазначені, а саме

- реалізація менеджменту прав доступу таких як створення акаунтів компанії та користувача, створення сумісних креативних просторів, можливість обмежити або надати доступ користувачу до ресурсів, настройка ролей у групі користувачів;
- можливість розбивати креативний простір на креативні зони та надавати до них доступ в інших креативних просторах;
- підтримка віджетів: аудіо та відео елементів, календаря, голосування, стікерів, абстрактних форм, пензлика, mind-map та інші;
- можливість інтеграції зі сторонніми застосунками:
  - збереження ресурсів в google drive, onedrive, dropbox;
  - інтеграція з месенджерами Stack, Teams.

6. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

6.1 Опис ідеї проекту

Ідея проекту даної роботи - розробити сервіс, який облегшує комунікацію між членами команди, що у свою чергу зменшить час витрачений на комунікацію що призведе до зменшення вартості розробки. Це є надзвичайно позитивним фактором так як у сфері інформаційних технологій більша частина бюджету витрачається на робітників.

Користувачі системи мають можливість створювати креативні простори та виконувати на них менеджмент контенту у неформальному вигляді.

Узагальнення цих ідей можна побачити в таблиці 6.1.

Таблиця 6.1 Опис ідеї стартап проекту.

Зміст ідеї	Напрямки застосування	Вигоди для користувача
		Скорочення часу, що використовується на комунікацію робітників
		Можливість детального проектування у неформальному вигляді

Основні вимоги до стартап-проектів, як правило, полягають у наступному:

- аналіз ринку на наявність аналогічних рішень;
- ознайомлення можливих споживачів та партнерів зі стартап-проектом;
- обрати найкращий час для подання інформації про стартап-проект;
- забезпечити процеси обробки інформації при реалізації проекту;
- зберегти результати обробки інформації в необхідному вигляді

Загальні обмеження на роботу з інформацією можуть бути представлені як

група характеристик, які дійсно належать сторонам стартап-проекту, а саме:

- об'єкту (джерелу інформації);
- медіатору (засобу отримання, обробки, виробництва і передачі інформації, посереднику між суб'єктом і об'єктом);
- суб'єкту (активній стороні діяльності, одержувачу і перетворювачу інформації).

Складність отримання інформації на сьогодні обумовлена її розподіленням та фрагментарністю, а також різнопрофільним заповненням.

Зараз на ринку існує декілька основних сервісів для колаборативного менеджменту контенту, доведеться конкурувати і з ними, а не тільки з тими, що пропонують рекомендації. Тому із найкрупніших конкурентів можна виокремити Miro, AWW та MURAL.

6.2 Технологічний аудит ідеї проекту

Список технологій реалізації програмного комплексу наведено в таблиці 6.2.

Таблиця 6.2 - Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	REST (Representational State Transfer)	Наявні	Доступні
	Технології її реалізації	Наявність технологій	Доступність технологій

Продовження таблиці 6.2

Ідея проекту			
	Basic access authentication	Наявні	Доступні
	OAuth	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: REST (Representational State Transfer)			
	Basic access authentication	Наявні	Доступні
	OAuth	Наявні	Доступні
	За допомогою токенів	Наявні	Доступні
	На основі сертифікатів	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: За допомогою токенів			
	Asp.net mvc C# .NET	Технології наявні, їх потрібно	Технології наявні, їх потрібно

	Nodejs JavaScript	обєднати міє собою Технології наявні, їх потрібно	обєднати міє собою Технології наявні, їх потрібно

Продовження таблиці 6.2

		обєднати міє собою	обєднати міє собою
Для масштабованої точки входу обрана технологія NodeJs+JavaScript, а для маштабованого рівня для бізнес логіки Asp.net+C#+.NET			

6.3 Аналіз потенційних техніко-економічних переваг

Аналіз сильних, слабких та нейтральних сторін стартап-проекту визначено в таблиці 6.3. Перевагами системи є наявне рішення проблеми роботи з великою кількістю даних.

Таблиця 6.3 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Техніко-економічні характеристики ідеї	товари/концепції конкурентів			Слабка сторона	Нейтральна сторона	Сильна сторона
	Мій проект	Miro	AWW			

Можливість роботи з великою кількістю елементів	Так	Повільно	Повільно			+

Продовження таблиці 6.2

Кількість компонентів	Середня	Велика	Середня		+	
Віджети	Середня	Велика	Велика		+	
Вбудований аудіо та відео зв'язок	ні	так	так	+		
Час завантаження сайту при малому навантаженні	0,5	0,3	0,4	+		
Час завантаження сайту при великому навантаженні	0,7	15	17			+
Масштабування	+	+	+		+	

6.4 Визначення потенційних груп клієнтів

Характеристики потенційних клієнтів стартап-проекту наведені в таблиці 6.4. Це можуть бути представники великого, малого та середнього бізнесу.

Таблиця 6.4 - Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Бажання зменшення часу роботи програміста та приріст швидкості реалізації застосунків	1. Малий середній та великий бізнес галузі IT	1. Ведення бізнесу на платформах інтернет-ресурсів	Можливість роботи з великою кількістю даних

6.5 Аналіз ринкових можливостей запуску стартап-проекту

В таблиця 6.5 наведено дані з попереднього огляду ринку застосунків для колаборативного менеджменту контенту.

Таблиця 6.5 - Попередня характеристика потенційного ринку проекту

Показники стану ринку (найменування)	Характеристика
Кількість головних гравців, од	6
Загальний обсяг продаж, грн/ум.од	3 грн/ум.од
Динаміка ринку (якісна оцінка)	Зростає
Наявність обмежень для входу (вказати характер обмежень)	Недискримінаційні якісні

Продовження таблиці 6.5

Специфічні вимоги до стандартизації та сертифікації	Відсутні
Середня норма рентабельності в галузі (або по ринку), %	69,5%

6.6 Аналіз ринкового середовища

Сфера колаборативного менеджменту контенту є відносно новою і зародилась на початку 2010-х років, тому у більшості країн не розроблена відповідна законодавча база для регуляції даної області, тому необхідно враховувати низку інших загрозливих факторів.

Фактори ризику та загроз наведені в таблиці 6.6.

Таблиця 6.6 - Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Крадіжка власності	ключової інтелектуальної інновації	Відсудження прав інтелектуальної власності Забезпечення якісного захисту інформації зміна методики групування даних
Відмова компаній у співпраці	Керівництво компанії не погоджується на співпрацю	Пропозиція більш вигідних умов співпраці
Недостача стартових капіталовкладень	Недостача початкових інвестицій для реалізації мінімально життєздатного продукту	Пошук нових джерел інвестицій

Поточний розвиток галузі систем колаборативного менеджменту контенту передачі даних показує позитивні темпи росту. Фактори можливостей наведені в таблиці 6.7.

Таблиця 6.7 - Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Отримання інвестицій	Сформований початковий капітал реалізації продукту	Розробка мінімально життєздатного продукту

Висока зацікавленість користувачів	Обіг використання веб-служби становить більше 1000 запитів в день	Підтримка стабільної роботи системи та проведення масштабування системи збільшення цін на використання сервісу
Успішна маркетингова політика	В результаті проведеної маркетингової політики отримана висока зацікавленість користувачів	Підтримка стабільної роботи системи та проведення масштабування системи Збільшення цін на використання сервісу Використання подібної маркетингової стратегії надалі для залучення нових

Продовження таблиці 6.7

Фактор	Зміст можливості	Можлива реакція компанії
		користувачів
Ліквідація конкурента	Конкурент ліквідував свою компанію у результаті власного бажання або зовнішніх чинників	Проведення маркетингової кампанії для монополізації ринку

6.7 Аналіз пропозиції

Також, було розглянуто конкуренцію на ринку і результати наведені в таблиці 6.8.

Таблиця 6.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Особливості конкурентного середовища
Тип конкуренції - чиста	Немає чітко відокремленого лідера серед конкурентів, та відповідно мала їх кількість	Сприятливо впливає на розвиток проекту
Рівень конкурентної боротьби- міжнародний	Сфера зачіпає практично всі країни світу	Можливий швидкий вихід на світові ринки

Продовження таблиці 6.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Особливості конкурентного середовища
За галузевою ознакою - внутрішньогалузева	Загроза появи нових конкурентів Ринкова влада споживачів Висока потреба у товарі	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін
За видами товарів - товарно-родові	Надання різних сервісів одного виду	Маркетингова політика
Цінова	Використання цін для покращення економічних	Зменшення вартості сервісу Використання нових каналів

	умов збуту	розподілу

Аналіз конкуренції за М. Портером дало більш детально інформацію про стан ринку і можливості конкурентів (таблиця 6.9).

Таблиця 6.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
	Miro, AWW та MURAL, Conceptboard	Цінність ідеї та її якісна реалізація	Немає	Споживачі прямо впливають на успішність	Конкуренти можуть стати дешевшими або стануть

Продовження таблиці 6.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
				продукту, оскільки щомісяця сплачуються кошти за ліцензію користування застосунком	надавати якісніші послуги

				за обраним планом	
Ви-сновки	Конкурентів небагато, відсутність монополії, конкурентна боротьба значна	Є можливість виходу на ринок	Немає постачальників	Так, реалізація потреб клієнтів вирішує успішність проекту	Достатньо важко предентувати на клієнтів інших готових продуктів

6.8 Перелік факторів конкурентоспроможності

Навіть при врахуванні конкурентів розроблена система має низку переваг, які наведені в таблиці 5.10.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
У конкурентів наявні функціональні проблеми реалізації	Часткова або повна неможливість працювати з великою кількістю даних
Ідея	розвиток ідеї колаборативної роботи
Швидкий вихід на ринок	Достатньо випустити мінімальну базову працездатну версію продукту на ринок для його використання

Цінова політика	Отримання прибутку здійснюється за рахунок гнучкої моделі оплати

6.9 Аналіз сильних та слабких сторін стартап-проекту

За наведеними факторами конкурентоспроможності в таблиці 5.10 було виконано порівняльний аналіз сильних та слабких сторін програмного рішення (стартап-проекту), який відображено в таблиці 5.11.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін системи

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні із системою управління процесами						
		-3	-2	-1	0	+1	+2	+3
Ідея	11					+		

Продовження таблиці 6.12

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні із системою управління процесами						
У конкурентів наявні функціональні проблеми реалізації	18							+

Швидкий вихід на ринок	15					+		
Цінова політика	12					+		

6.10 SWOT-аналіз

Складений SWOT-аналіз на основі вище отриманих даних відображено в таблиці 6.12.

Таблиця 6.12 – SWOT-аналіз стартап-проекту

Сильні сторони: – можливість роботи з великою кількістю контенту – відносно малий займаний застосунком простір – швидкість завантаження – розвинута система структурування – широкий спектр функціоналу; – мала залежність від інвестицій; – невелика кількість сильних конкурентів;	Слабкі сторони: – потреба в рекламі

Продовження таблиці 6.12

Можливості: – невелика кількість конкурентів; – інвестиції;	Загрози: – вихід на ринок нових конкурентів – зміна тенденцій розвитку ринку

6.11 Опис цільових груп потенційних клієнтів

В таблиці 6.13 наведено характеристики потенційних клієнтів.

Таблиця 6.13 - Характеристика потенційних клієнтів стартап проекту

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Малий бізнес	Середня	70 %	Середня	Просто
Середній бізнес	Висока	85%	Середня	Просто
Великий бізнес	Середня	75 %	Середня	Складно

6.12 Формування базових стратегій розвитку

Базових стратегій розвитку стартап проекту описані в таблиці 5.14.

Базовою стратегією для стартап-проекту є стратегія диференціації, що означає надання програмному комплексу властивостей, які будуть відрізняють його від конкурентів.

Таблиця 6.14 — Визначення базової стратегії розвитку

Обрана альтернатива	Стратегія охоплення ринку	Ключові конку-рентоспроможні	Базова стратегія

розвитку проекту		позиції	розвитку
Пропонування програмного комплексу для IT команд	Диференційований маркетинг	Здатність пропонувати унікальний функціонал	Стратегія диференціації

6.13 Вибір стратегії конкурентної поведінки

В таблиці 6.15 наведено базові стратегії конкурентної поведінки.

Таблиця 6.15 — Визначення базової стратегії конкурентної поведінки

Чи є проект першопроходьцем на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Залучати нових та забирати існуючих	Віджети	Стратегія велику лідера

На основі потреб споживачів з обраних сегментів, а також в залежності від

обраної базової стратегії розвитку в таблиці 5.16 наведено визначені стратегії позиціонування.

Таблиця 6.16 — Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного
Швидкодія та зручність	Стратегія диференціації	Розширення базового функціоналу	Якісний продукт, зрозумілий у використанні

Першим кроком до формування маркетингової концепції товару, який отримує споживач. Для цього у таблиці 6.17 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 6.17 – Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
Швидкодія	Стабільна швидкість завантаження сторінки та плавна її робота	У існуючих застосунків існує застосунку при великій кількості компонентів проблема значного

		уповільнення

Продовження таблиці 6.17

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
Потрібна пам'ять	Оптимізація займаного простору	У існуючі застосунки потребують відносто великих об'ємів пам'яті для роботи з великою кількістю елементів

Наступним кроком є визначення цінових меж представлених в таблиці 6.18, якими необхідно керуватись при встановленні ціни на товар

Таблиця 6.18 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Товари замінників немає	Товари замінники мають високий рівень цін	Цільова група споживачів має високий рівень доходів	Нижня межа – це найменша ціна рішення найближчого найдешевшого конкурента, верхня межа, це на 1% більша ціна на найближчого найдорожчого конкурента

4.7 Висновки

Під час проведення дослідження магістерської дисертації у якості стартап-проекту було створено відповідну аргументовану документацію. Протягом усього

процесу дослідження було досягнуто повне розуміння цілей проекту, його переваги та недоліки, які в свою чергу дали можливість точніше визначити мету проекту та користувацьку аудиторію програмного застосунку, визначити можливі методи монетизації, маркетингову стратегію для IT команд та власного використання.

На даний час попит на такий програмний комплекс може бути не дуже високим через специфіку області застосування.

Перевагами проекту є:

- простий у використанні;
- кросплатформеність;
- зручний інтерфейс користувача;
- оптимізованість застосунку при великому навантаженні.

Унікальність проекту полягає в рівні у вирішені проблем оптимізації роботи застосунку при великому навантаженні. А саме було запропоновано методи, які показали наступні результати :

- скоротили середню швидкість завантаження застосунку з 28 до 3,5 секунд;
- зменшили займаний простір застосунку з 520 до 80 мб.

ВИСНОВКИ

З формуванням тренду на віддалену роботу змінюються самі стандарти та підходи до командної розробки, стандартом стають нові види інструментів для комунікації. Одним з таких видів є інструменти для неформального корпоративного менеджменту контенту. В роботі вирішена актуальна проблема оптимізації швидкодії даного виду інструментів.

Реалізована система для колаборативного менеджменту контенту є сукупністю програмних засобів, використання яких призведе до економії часу на комунікації членів команди за допомогою зручним інструментів менеджменту контенту неформальному вигляді, а також дозволить підвищити якісні та кількісні показники процесів.

Для досягнення мети дослідження, яка полягає у підвищенні якості та ефективності використання інструменту, були вирішені наступні завдання:

– досліджено особливості функціонування платформ для неформального корпоративного менеджменту контенту

– проаналізовано існуючі платформи для розробляємої платформи;

– обґрунтовано вибір підходів і технологій для створення платформи;

– спроектовано архітектуру та базу даних застосунку для колаборативного менеджменту контенту

– створено методологію оптимізації високонавантажених canvas застосунків;

– виконано оптимізацію застосунку розробленою методологією;

– реалізовано інтерфейс користувача для розробляємої платформи;

– охарактеризовано програмне, технічне та організаційне забезпечення платформи.

В ході виконання магістерської дисертації було зроблено такі висновки:

– платформа розроблення програмного забезпечення повинна складатися з двох основних підсистем: spa точки сходу та додатковим сервером з бізнес логікою та базою даних;

– для пришвидшення роботи застосунку необхідно використовувати методи:

- метод кластеризації даних;
- метод розбиття на рівні;
- метод вигрузки елементів;

– для оптимізації запитів до сервера на клієнтській частині слід використовувати IndexDb;

– впровадження платформи у практичне використання повинно супроводжуватися та підкріплюватися запровадженням відповідного стартап-процесу.

Загальним результатом розробки системи колаборативного менеджменту контенту має стати реалізація відповідного стартап-проекту на базі отриманих результатів дослідження та виведення відповідного рішення на ринок корпоративних інструментів комунікації. Розробка стартап-проекту допомогла краще осягнути рамки розроблюваного рішення, його недоліки та переваги, якими він повинен володіти. Це було закріплено у відповідних таблицях розділу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Skilled freelancers [Електронний ресурс] – Режим доступу до ресурсу: https://www.cnbc.com/2019/10/03/skilled-freelancers-earn-more-per-hour-than-70percent-of-workers-in-us.html?&qsearchterm=70%20worker.

2) Freelancing in America 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://www.upwork.com/i/freelancing-in-america/2019/.

3) Telecommuting [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Telecommuting.

4) Microsoft teams [Електронний ресурс] – Режим доступу до ресурсу: https://products.office.com/en-us/microsoft-teams/group-chat-software.

5) Google drive [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/drive/.

6) Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/.

7) Online Whiteboards in 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://zapier.com/blog/best-online-whiteboard/.

8) Miro [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com.

9) AWW [Електронний ресурс] – Режим доступу до ресурсу: https://awwapp.com/.

10) Conceptboard [Електронний ресурс] – Режим доступу до ресурсу: https://conceptboard.com/.

11) Explain everything [Електронний ресурс] – Режим доступу до ресурсу: https://explaineverything.com/.

12) Mural [Електронний ресурс] – Режим доступу до ресурсу: https://mural.co/.

13) Kirk M. Thoughtful Machine Learning / Matthew Kirk., 2014.

14) Kirsten L. Hunter. Irresistible APIs / Kirsten L. Hunter., 2016. – 232 с.

15) Sam Newman. Building Microservices / Sam Newman., 2015. – 102 с.

16) Adam Freeman. Pro ASP.NET Core MVC 2 / Adam Freeman., 2017. –

1017 с. Bill Wagner. NET Microservices Architecture for Containerized NET

17) Applications / Bill Wagner., 2018. – 309 с.

18) Backend Programming Languages [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@Sandra_Parker/top-backend-programming-languages-of-2019-dc8d81836b2d.

19) GUIDE TO BACKEND DEVELOPMENT [Електронний ресурс] – Режим доступу до ресурсу: https://learntocodewith.me/posts/backend-development/.

20) Addy Osmani. A Tinder Progressive Web App Performance Case Study [Електронний ресурс] / Addy Osmani. – 2017. – Режим доступу до ресурсу: https://medium.com/@addyosmani/a-tinder-progressive-web-app-performance-case-study-78919d98ece0.

21) Angular vs. React vs. Vue: Сравнение 2017 [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/post/338068/.

22) Илюшечкин В.М. Основы использования и проектирования баз данных / В.М. Илюшечкин. – М.: Юрайт, 2016 – 214 с.

23) Конноли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание/Пер. с англ. – М.: Издательский дом “Вильямс”, 2017 – 1440 с.

24) SQL или NoSQL — вот в чём вопрос [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/company/ruvds/blog/324936/http://www.mattstine.com/2014/06/30/microservices-are-solid/

25) Kline K. Search Results Web results SQL in a Nutshell: A Desktop Quick Reference Guide / Kevin Kline., 2018.

26) Browser database comparison [Електронний ресурс] – Режим доступу до ресурсу: https://nolanlawson.github.io/database-comparison/.

27) IndexedDB API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

28) Optimizing canvas [Електронний ресурс] – Режим доступу до

ресурсу: https://developer.mozilla.org/en-S/docs/Web/API/Canvas_API/Tutorial/Optimizing_canvas.

29) Wan A. When to use HTML5's canvas [Електронний ресурс] / Alvin Wan. – 2019. – Режим доступу до ресурсу: https://blog.logrocket.com/when-to-use-html5s-canvas-ce992b100ee8/.

30) Jaoko P. HTML5 Canvas Optimizatio [Електронний ресурс] / Patrick Jaoko. – 2019. – Режим доступу до ресурсу: https://code.tutsplus.com/tutorials/html5-canvas-optimization-a-practical-example--active-11893.

31) Обзор алгоритмов кластеризации данных [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://habr.com/ru/post/101338/

32) Guido S. Introduction to Machine Learning / Sarah Guido., 2016.

ДОДАТКИ

Додаток А – Публікація

www.konferenciaonline.org.ua

Міжнародна наукова інтернет-конференція

"Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 43)

14 листопада 2019 р.

Частина 1

Зміст

Частина 1

Секція 1. Інформаційні системи і технології

Бобович Ю.В.
Національний Технічний Університет України "Київський політехнічний інститут ім. Ігоря Сікорського", м. Київ
Кафедра автоматики та управління в технічних системах, студент

МЕТОДИ ОПТИМІЗАЦІЇ HTML5 CANVAS

Елемент "canvas" є частиною HTML5 і дозволяє динамічно зображати 2D форми та растрові зображення. Це низька, процедурна модель, яка оновлює растрову карту і не має вбудовані графічні сцени також через WebGL він дозволяє відображати 3D форми та зображення.

Зараз ця технологія підтримується найпопулярнішими браузерами та платформами.

Основні мітоди оптимізації:

1. Використання примітивів

Якщо часто потрібно повторювати одні і ті ж операції малювання на кожному анімаційному кадрі то можна завантажити їх на позаекранне полотно. Потім можна відображати зображення на екрані на основному полотні так часто, як це потрібно, без зайвого повторення кроків, необхідних для його створення.

2. Заміна чисел з плаваючою комою на цілі значення

Наявність суб пікселів змушує браузер робити додаткові обчислення для створення ефекту згладжування. Щоб уникнути цього можна округлити всі координати, які використовуються у викликах для drawImage (), наприклад, використовуючи Math.floor ().

3. Уникнення масштабування малюнку

Можна кешувати різні розміри своїх зображень у під час завантаження, а не виконувати постійне масштабування їх у drawImage ().

4. Використання багатошарових полотен для складних сцен

У випадках коли деякі об'єкти потрібно часто переміщувати або змінювати, а інші залишаються відносно статичними. Можлива оптимізація за допомогою розбиття елментів на різні шари, використовуючи кілька елементів <canvas>.

Наприклад, у випадку з грою інтерфейс користувача знаходиться зверху, інтерактивні об'єкти посередині та статичний фон знизу. У цьому випадку можна розділити гру на три шари <canvas>. Користувальницький інтерфейс мінятиметься лише після введення користувача, ігровий шар змінюватиметься з кожним новим кадром, а фон залишатиметься незмінним.

5. Використовування звичайного CSS для великих фонових зображень.

Якщо потрібно використовувати статичне фонове зображення, можна намалювати його на простому елементі , використовуючи властивість фонового зображення CSS і розмістити його під полотном. Це усуне необхідність відтворення фону на полотні на кожному кадрі.

6. Масштабування полотна за допомогою перетворень CSS

Перетворення CSS відбуваються швидше, оскільки вони використовують GPU. Найкращий випадок - це не масштабувати полотно, або мати менше полотно і збільшувати його масштаб, а не більше полотно та зменшувати його.

7. Вимкнення прозорість

Якщо на веб-сторінці використовується полотно і не потрібен прозорий фон можна встановити у альфа-параметру значення false, коли створюєте контекст малювання за допомогою getContext().

Список використаних джерел:

1. Alvin W. Optimizing canvas [Електронний ресурс] / Wan Alvin – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Optimizing_canvas.
2. Alan W. When to use HTML5's canvas [Електронний ресурс] / Wan Alan – Режим доступу до ресурсу: https://blog.logrocket.com/when-to-use-html5s-canvas-ce992b100ee8/.
3. Myths and realities of canvas [Електронний ресурс] – Режим доступу до ресурсу: https://www.html5gamedevs.com/topic/7735-myths-and-realities-of-canvas-javascript-performance/.