

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

**Розробка платформи для організації спільного дозвілля та командних
івентів**

**Текстова частина до курсової роботи
за спеціальністю „Програмна інженерія”**

Керівник курсової роботи
Кандидат фізико-
математичних наук,
Гречко А. В.

Виконав студент
Шкаровська Н. С.

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Г. І. Малашонок

(підпис)

„_____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Шкаровська Н. С. факультету інформатики 4-го курсу

ТЕМА: Розробка платформи для організації спільного дозвілля та командних івентів

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Вступ

1. Аналіз предметної області
2. Теоретичні відомості
3. Опис реалізації програмного продукту

Висновки

Перелік використаних джерел

Додатки

Дата видачі „_____” _____ 2020 р. Керівник _____
(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка платформи для організації спільного дозвілля та командних івентів

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання курсової роботи.	29.10.2020	
2.	Дослідження соціального питання організації івентів в умовах карантину.	15.11.2020	
3.	Опис технічного завдання проекту.	23.11.2020	
4.	Розробка основного дизайну платформи.	05.12.2020	
5.	Дослідження сучасних методів та підходів у розробці клієнт-серверних застосунків.	11.12.2020	
6.	Знайомство з бібліотекою React, визначення кращих методів розробки багатосторінкових сайтів.	8.01.2021	
7.	Знайомство з фреймворком Flask, визначення додаткових бібліотек та сервісів, що використовуватимуться.	8.01.2021	
8.	Розробка схеми бази даних.	10.01.2021	
9.	Створення базової архітектури клієнтської частини, верстка перших сторінок.	18.01.2021	
10.	Створення базової архітектури серверу, перші методи API.	27.01.2021	
11.	Опитування цільової аудиторії	03.02.2021	
12.	Огляд аналогів розробки.	03.02.2021	
13.	Верстка основних сторінок сайту.	20.02.2021	

14.	Написання методів основного функціоналу серверу.	20.02.2021	
15.	Інтеграція клієнтської частини з сервером.	28.04.2021	
16.	Розширення функціоналу системи, впровадження обмежень авторизації.	11.03.2021	
17.	Тестування готових компонентів програми та їх відлагодження.	20.03.2021	
18.	Оптимізація систем, впровадження додаткових валідаторів та обмежень.	29.03.2021	
19.	Оформлення документації.	05.04.2021	
20.	Завершення розробки веб-застосування.	05.04.2021	
21.	Завершення написання пояснювальної записки.	09.04.2021	
22.	Створення презентації.	07.05.2017	
23.	Захист курсової роботи.	19.05.2017	

Студент Шкаровська Н. С.

Керівник Гречко А. В.

“ _____ ” _____

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
АНОТАЦІЯ.....	9
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	13
1.1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ТЕМИ	13
1.2 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ РОЗРОБКИ	15
1.2.1 <i>TRN.ua</i>	15
1.2.2 <i>Board Game Arena</i>	16
1.2.3 <i>Microsoft Teams</i>	17
1.3 ПОСТАНОВКА ЗАДАЧІ	18
2 ТЕОРЕТИЧНІ ВІДОМОСТІ.....	20
2.1 ПРЕДМЕТА ОБЛАСТЬ	20
2.2 ОПИС ТЕХНІЧНИХ ВІДОМОСТЕЙ.....	21
2.2.1 <i>Клієнтська частина</i>	22
2.2.2 <i>Серверна частина</i>	25
3 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ.....	28
3.1 ТЕХНІЧНЕ ЗАВДАННЯ	28
3.1.1 <i>Вимоги до інтерфейсу</i>	28
3.1.2 <i>Функціональні вимоги</i>	29
3.1.3 <i>Вимоги до даних</i>	31
3.2 АРХІТЕКТУРА ВЕБ-ЗАСТОСУВАННЯ	32
3.3 ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ	34
3.4 ОПИС ПРОЦЕСУ РОЗРОБКИ ПРОГРАМИ	36
3.5 РОЗРОБКА ОСНОВНОЇ ПРОГРАМИ	37
3.5.1 <i>Клієнтська частина</i>	37
3.5.2 <i>Серверна частина</i>	39
3.6 ОПИС БАЗИ ДАНИХ ТА ІНТЕРФЕЙСІВ ПРОГРАМИ.....	41
3.6.1 <i>Схема даних</i>	41

3.6.2	<i>Користувацький інтерфейс</i>	43
3.6.3	<i>Інтерфейс серверу</i>	44
3.7	ІНСТРУКЦІЯ КОРИСТУВАЧА.....	44
3.8	ПЕРСПЕКТИВИ РОЗРОБЛЕНОЇ ПЛАТФОРМИ	48
	ВИСНОВКИ	50
	ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	52
	ДОДАТКИ	54
1	Додаток А – Опитування цільової аудиторії.....	54
2.2.3	<i>Інтерв'ю №1 – Анастасія:</i>	54
2.2.4	<i>Інтерв'ю №2 – Артем:</i>	55
2.2.5	<i>Інтерв'ю №3 – Олександр:</i>	57
2.2.6	<i>Інтерв'ю №3 – Марія:</i>	59
2	Додаток Б – Додаткові ілюстрації.....	61
3	Додаток Б – Список ілюстрацій	68

Перелік умовних позначень

Eventizer – авторська назва платформи для організації спільного дозвілля та командних івентів, утворена зі слів «event organizer» (з англ. організатор заходів).

Організатор – особа, що займається організацією групових видів діяльності, користувач Eventizer.

Учасник – користувач Eventizer або особа, що зацікавлена брати участь у групових заходах відповідно до власних інтересів.

Вебінар – різновид проведення онлайн-зустрічей, конференцій чи презентацій через Інтернет.

Спам — масове розсилання кореспонденції рекламного чи іншого характеру людям, які не висловили бажання її одержувати. Передусім термін «спам» стосується рекламних електронних листів.

Аватар – фото профілю користувача або невелика картинка, що вказує на основне зображення компоненти чи об'єкту.

Нікнейм – унікальне ім'я користувача в межах системи, може не відповідати реальному імені, складатися з цифр та латинських символів.

Хедер – в даному проекті, елемент верстки, розташований згори сторінки, що містить певний набір навігаційних кнопок.

Мемоізація – це один з методів оптимізації, що зберігає виклики функцій, що потребували складних обчислень та кешує результати, при наступних запитах з такими ж параметрами повертається вже кешований об'єкт.

Лоадер – в даній курсовій роботі – елемент завантаження, що відображає процес обробки або обміну даними.

FAQ (Frequently Asked Question(s)) — часто поставлені, поширені питання.

API (Application Programming Interface) – набір чітко визначених методів для взаємодії з програмою або її компонентами.

HTTP (HyperText Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах та забезпечує функціонування Всесвітньої мережі Інтернет.

TCP (Transmission Control Protocol) – протокол передачі даних, що працює на транспортному рівні моделі даних.

REST (Representational State Transfe) – підхід до архітектури мережевих протоколів, що дозволяє працювати з інформаційними ресурсами такими, як HTTP.

СУБД (система управління базами даних) - набір взаємопов'язаних програм і даних для доступу до цих даних.

JSON (JavaScript Object Notation) – текстовий формат обміну даними.

Анотація

Даною курсовою роботою передбачається розробка платформи для організації спільного дозвілля та командних івентів. Платформа має трирівневу архітектуру, клієнтська частина розроблена на TypeScript за допомогою бібліотеки React; сервер – на мові Python, за допомогою фреймворку Flask, а також SQLAlchemy для реалізації ORM; використовувалася база даних PostgreSQL. Всі програмні компоненти задеплойовані в хмарний сервіс Heroku та доступні в мережі Інтернет. До даного проекту також було підключено сервіси Google: Map API, Map Autocomplete, Main Service, а також imgbb API для завантаження фото та Elasticsearch для реалізації швидкого пошуку подій. Створений сайт містить увесь функціонал вказаний у вимогах, зокрема зручну пошукову систему подій, функціонал особистого кабінету користувачів та систему управління відвіданими чи організованими подіями. У текстовій частині роботи міститься опис роботи використаних технологій, процеси реалізації основної програми веб-застосунку, технічне завдання, а також інструкція користувача готовою системою.

Вступ

Сучасний світ невинно змінюється та рухається вперед, постійно створюючи нові виклики, які людству необхідно вирішувати. Пандемія 2020 року радикально змінила уявлення світового співтовариства про світ, залишивши мільйони ізольованими у власних домівках. Проблема спілкування ще ніколи не була такою болісною, адже ніхто не звик бачити друзів або близьких виключно через екран смартфона. У сучасних реаліях більшості людей складно знайти однодумців та організувати цікаве і змістовне дозвілля. Незважаючи на стрімкий ріст популярності платформ для відеозв'язку, проблема пошуку однодумців або організації цікавої діяльності для компанії друзів залишається. Вирішенням окреслених проблем може стати веб-застосунок для організації спільного дозвілля та командних івентів «Eventizer» - автоматизована платформа, що дозволить зручно організувати онлайн-вечірку, здійснювати пошук однодумців або знайти цікавий спосіб організувати своє дозвілля в будь-який час. Окрім того, сервіс покликаний допомогти організаторам публічних заходів шукати свою аудиторію та популяризувати власні послуги. У періоди послаблення карантину Eventizer також може використовуватись як платформа для анонсів офлайн подій.

Мета курсової роботи – створити платформу, що повною мірою зможе вирішити проблему спілкування, яка виникла під час пандемії в більшості сучасних людей. Основним завданням проєкту є дослідження особливостей онлайн-комунікації для визначення найкращої стратегії розробки онлайн застосунку. Реалізація такої платформи дозволить у зручний спосіб організовувати онлайн-заходи та приємно проводити час, спілкуючись з друзями або знаходячи нових однодумців, не залишаючи власних домівок.

Об'єкт дослідження курсової роботи - процес створення веб-сайту з організації онлайн-заходів, що містить функціонал, необхідний для вирішення вищевказаних проблем. Функціонал сайту базується на дослідженні існуючих способів організації дозвілля, їх недоліків та переваг, а також відповідно до аналізу

потреб цільової аудиторії, як з боку організаторів, так і учасників. Зокрема, об'єктами дослідження, що становить практичний інтерес виключно для дослідника, також є: бібліотека React, що дозволяє будувати складний та швидкодієний користувацький інтерфейс; фреймворк Flask та програмну бібліотеку SQLAlchemy на мові Python, що дозволяють створювати мінімалістичний, швидкодієний та масштабований сервер із застосуванням технології об'єктної бази даних.

Предметом дослідження є поведінкові алгоритми людей під час спілкування в офлайн та в онлайн-форматі, побажання та очікування потенційної цільової аудиторії від платформи, що покликана вирішити проблеми спілкування. Також було проведено дослідження алгоритмів пошуку та сучасних технологій, що дозволяють робити пошук об'єктів за ключовими словами миттєво та з урахуванням допустимих помилок в слові при введенні пошукового запиту користувачем. Основними методами дослідження були моделювання, експеримент, змістовний аналіз ключових проблем та класифікація для визначення оптимальних рішень. Також використовувались методи прогнозування для визначення актуальності такої системи в посткарантинний період.

Проведені дослідження у повній мірі допомогли окреслити основний необхідний функціонал платформи та визначити зовнішній вигляд користувацького інтерфейсу. Окрім того, вдалося чітко окреслити інформацію, що повинна зберігатися в базі даних, критерії та особливості пошуку івентів. Готове рішення може використовуватись як автоматизована платформа для організації та пошуку онлайн-заходів, як система для поширення інформації про публічні івенти або як інструмент для погодження оптимального способу проведення дозвілля для компанії друзів.

Для створення «Eventizer» було використано сучасні методи та засоби розробки. Середовищем розробки клієнтської частини став WebStorm 2020.2.4, що містить нативну підтримку TypeScript. Серверна частина розроблялася в PyCharm

2020.3.3 – інтегроване середовище для створення застосунків мовою Python, а управління базою даних PostgreSQL 12 відбувалося за допомогою DataGrip 2020.3.2. Всі інтегровані середовища розробки є ліцензійними продуктами від компанії JetBrains. Для реалізації функціоналу пошуку на сайті було використано Elastic App Search Service, що дозволяє робити швидкий пошук за ключовими словами використовуючи індекси, що зберігаються в хмарі. Для мануального тестування використовувались браузері Google Chrome та Opera (включаючи вбудовані інструменти розробника), а також Postman – інструмент тестування API. Для деплою серверного та клієнтського веб-застосунку було використано безкоштовну хмарну платформу Heroku, база даних розміщена в репозиторії серверу, як допоміжний сервіс.

Курсова робота містить перелік умовних позначень, вступ, три розділи, висновок, перелік використаних джерел та додатки. Аналіз предметної області описує сучасний стан досліджуваної проблеми та описує основну задачу необхідну для виконання. У розділі теоретичних відомостей детально описуються дотичні до теми терміни та використані технології. Третій розділ містить власне інформацію про реалізацію веб-застосування, що є основною метою даної курсової роботи.

Структура роботи повністю відповідає всім вимогам до курсових робіт.

1 Аналіз предметної області

1.1 Аналіз сучасного стану питання та обґрунтування теми

У сучасному світі не завжди можна віднайти однозначне визначення будь-якого явища чи предмету. З огляду на це даний проєкт можна розглянути як складову технічних інструментів комунікації, сфери розваг або платформу для спілкування. Кожна з цих галузей стрімко розвивається та з року в рік дивує чимось новим, отримуючи схвалення користувачів. Незважаючи на те, що під час пандемії багато світових компаній активно почали створювати чи розвивати власні продукти для комунікації, проблема організації спілкування залишається нерозв'язаною. Адже в основному розробники акцентуються саме на інструментах для спілкування, таких як відеозв'язок, а процеси домовленостей, поширення інформації та управління івентами залишається проблемним питанням для організаторів подій. Загалом можна виділити кілька ключових проблем, які зустрічаються у організаторів:

- складно знайти дійсно зацікавлену аудиторію в заході;
- інструменти для відеозв'язку не дають можливості окремо викладати корисні посилання та матеріали, щоб користувачі мали до них доступ після завершення сесії;
- необхідно постійно власноруч збирати інформацію про учасників і зберігати на окремих сервісах.

З боку учасників існує інша глобальна проблема – пошук релевантного контенту. Таргетована реклама допомагає знайти цікавий вебінар або онлайн-подію, проте доводиться відстежувати безліч заходів на різних соціальних платформах, фіксувати інформацію в сторонніх додатках, щоразу реєструватись на новій платформі для того, щоб мати можливість прийняти участь в онлайн заході. Також в учасників онлайн-подій виникає бажання детальніше поспілкуватися зі спікером або іншими учасниками заходу після його завершення, проте абсолютна

більшість сервісів не має такого функціоналу. Наразі, для того, щоб взяти участь в онлайн-події необхідно зробити наступні кроки:

1. Знайти захід, який задовольняє інтереси учасника. Зазвичай пошуки через інтернет займають достатньо багато часу, а рекомендації знайомих або таргетована реклама не завжди задовольняє всі запити користувачів.
2. Зареєструватися: реєстрація на сайті заходу або ж заповнення онлайн-форми з подальшим підтвердженням участі.
3. Дочекатися події: в переважній більшості випадків учасники просто очікують листа з посиланням на відеоконференцію (або інформацією про скасування) без змоги зв'язатися з організаторами або відстежити статус заходу, при цьому листи часто потрапляють в спам.
4. Відвідати подію

І організатори, і користувачі хотіли б мати можливість переглянути історію відвіданих чи організованих івентів, їх учасників, прикріплені матеріали чи іншу інформацію про захід без реєстрації на десятках платформ.

Окремі проблеми виникають при організації онлайн-ігор, зустрічей для компаній друзів чи знайомих. Щоразу, коли у однодумців є бажання зібратись, щоб весело провести час, виникає безліч суперечок: що саме робити, в який час зібратися, тощо. Для того, щоб реалізувати саму ідею зустрічі великою компанією комусь необхідно взяти на себе роль організатора та особисто опитати всіх на предмет бажання брати участь в запланованій події, а згодом поділитися посиланням на відео чи аудіочат. Для детального аналізу цього питання було проведено декілька інтерв'ю ([Додаток А](#)) на базі який було визначено основні потреби та бажання даної категорії цільової аудиторії платформи:

- платформа, що буде зберігати історію учасників кожної зустрічі;
- можливість створювати подію, щоб всі бажаючі могли приєднатися за посиланням;
- можливість в будь-який момент додати посилання на гру, щоб не робити розсилку всім учасникам;

- функціонал, який дозволить приєднатися до іншої групи людей, якщо немає бажання приймати участь в запропонованому заході;
- можливість знайти контакти людини після зустрічі з якою було приємно спілкуватися.

Саме тому виникає потреба в єдиній системі, що зможе задовольнити базові потреби організаторів та учасників, а в майбутньому стане платформою для популяризації онлайн-подій, пошуку однодумців, відстеження цікавих для себе заходів та полегшення організації онлайн-зустрічей в цілому. Варто зазначити, що єдина система включає в себе функціонал єдиного акаунту організатора та учасника, що може бути доступний в будь-який час та з будь-якого пристрою.

1.2 Огляд існуючих аналогів розробки

Прямі аналоги платформи для організації онлайн-заходів та спільного дозвілля знайти достатньо важко, адже в період пандемії значного поширення набули платформи для організації онлайн-навчання або проведення конференцій, а сферу розваг вважають укомплектованою, адже інструменти власне комунікації та ігор вже існують, проте, окремої уваги потребують питання організації подій дозвілєвого та ігрового характеру, які зазвичай залишаються непоміченими. З іншого боку, оскільки платформа може мати достатньо багато сфер застосування та акумулює в собі функціонал, що дозволяє користуватись іншими платформами, можна зробити припущення, що конкурувати з даною розробкою може будь-що: від платформи для онлайн-вебінарів, сайтів для гри в Мафію або інших командних ігор, платформ для відеозв'язку з вбудованим чатом, до фірм, що надають послуги з організації онлайн-заходів.

1.2.1 TRN.ua

TRN.ua – це всеукраїнський каталог тренінгів, семінарів та конференцій [\[1\]](#). Сайт дозволяє шукати тренінги та семінари за назвою, датою або містом, також є функціонал додати власний тренінг, але лише у випадку, якщо ви представляєте

якусь компанію. Система також містить каталог зареєстрованих компаній та тренерів, а також сторінку новин, де публікуються анонси до обраних редакторами сайту вебінарів. Для того, щоб стати учасником обраної активності замість реєстрації пропонується залишити заявку (прізвище, ім'я, посада в компанії, електронна пошта та номер телефону), а організатори самостійно зв'язуються з кожним та надають необхідну інформацію, що є не дуже зручно.

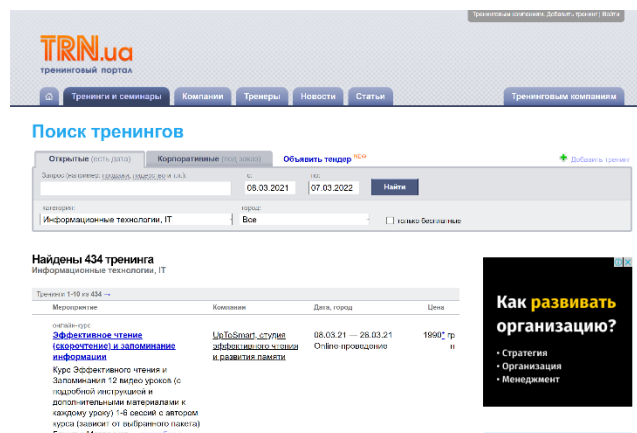


Рисунок 1 – Портал TRN.ua

Платформа є достатньо популярною, адже має півтори тисячі активних тренінгів (станом на лютий 2021-го року), також сайт активно рекламується пошуковою системою Google. Портал, створений у 2008-му році, є безкоштовним, але містить рекламні банери на кожній своїй сторінці, а користувацький інтерфейс вже давно застарів (рисунок 1), тому навряд чи він зможе набути популярності серед молоді. Окрім того, портал, що пропонує свої послуги виключно для українського ринку, реалізовано не українською мовою, відповідно не є релевантною на теренах сучасної України.

1.2.2 Board Game Arena

Board Game Arena – це міжнародна платформа, що пропонує грати в настільні ігри онлайн з друзями або випадковими учасниками. Система містить великий каталог онлайн-версій настільних ігор та дозволяє грати в



Рисунок 2 – Board Game Arena logo

режимі реального часу або по чергову з іншими. Підтримує дуже багато мов (в тому числі українську), має сторінку новин та форум, зручний пошук ігор за назвою, тривалістю, складністю або кількістю гравців. В процесі гри учасники можуть бачити аватари та нікнейми одне одного, спілкуватися в чаті, переглядати правила гри різними мовами та власне грати в гру. Також можна спостерігати за іншими

іграми, які проводяться в режимі реального часу, отримувати нагороди за участь у окремих активностях на сайті, а для преміум-користувачів доступний також аудіо-та відео-зв'язок. Серед недоліків системи можна визначити:

- Незрозумілий інтерфейс та відсутність покрокових інструкцій в іграх: платформа має дійсно багато функціоналу, але не має інструкцій, тому дуже складно розібратись як з нею працювати.
- Непопулярні ігри: каталог налічує близько 300 ігор, проте серед них немає сучасних популярних настільних ігор.

Також платформа не є універсальною для будь-якої сфери діяльності, а підходить виключно для поціновувачів настільних ігор. «BGA – це вебсайт, що підтримується співтовариством» [\[2\]](#) - вказують розробники сайту на сторінці «Про нас», саме тому незважаючи на вищевказані недоліки платформа дійсно є чудовим аналогом настільним іграм в реальному житті.

1.2.3 Microsoft Teams

Microsoft Teams - корпоративна платформа, що поєднує інструменти зустрічей, чатів та вкладень [\[3\]](#). Достатньо зручна платформа для регулярних зустрічей, в межах однієї групи може бути декілька каналів, а різні зустрічі можна організовувати паралельно. Має багато вбудованого функціоналу та дозволяє інтегруватися з іншими продуктами від компанії Microsoft. Може бути аналогом даної розробки для вже сформованих компаній друзів, а також часто використовується для проведення вебінарів або зустрічей неформального спілкування. Платформа дозволяє робити анонси зустрічей, проте виключно всередині каналу, також відсутній функціонал пошуку подій, натомість дуже легко особисто поспілкуватися з учасниками відвіданого заходу, історія відвіданих заходів також зберігається в каналах, проте функція перегляду історії всіх відвіданих подій списком відсутня.



*Рисунок 3 -
Microsoft Teams
logo*

1.3 Постановка задачі

Після ретельного аналізу сучасних потреб в організації дозвілля, базуючись на проведених інтерв'ю та огляді конкурентних систем, що дозволяють організовувати командні івенти, вдалося повною мірою сформулювати основну задачу, що необхідно виконати, та вимоги, яким система повинна відповідати.

Основна задача проекту: створити простий для використання веб-застосунок, що матиме інтуїтивно зрозумілий інтерфейс. Платформа у вигляді багатосторінкового сайту повинна забезпечувати користувачам доступ до єдиного акаунту, де зберігатиметься персональна інформація та інформація про дії на платформі, та бути доступною з будь-якого пристрою. Система передбачає 2 типи користувачів: організатор та учасник, при цьому деякий функціонал системи буде спільним для обох типів користувачів. Також необхідно забезпечити доступ незареєстрованих користувачів до каталогу подій та інформаційних сторінок. Система повинна повністю реалізовувати основні завдання проекту, а саме:

- можливість знайти подію за ключовими словами, часовим проміжком або сферою діяльності;
- створювати приватні події та ділитися посиланням на приєднання;
- створювати публічні події, що будуть доступні в пошуку;
- акумуляція інформації про учасників події та зберігання історії відвіданих або організованих подій;

Користувацький інтерфейс повинен бути приємним та лаконічним, відповідно до проведених опитувань бажано, щоб основна тема сайту була світлою, а кольорова гама – фіолетовою, при цьому бажано зберегти логічні кольорові позначення: помилка – червоного, попередження – жовтого, успішне виконання – зеленого кольорів. Сервіс повинен мати зручну навігацію між сторінками, всі доступні користувачу сторінки повинні відображатися на навігаційній панелі. Сайт повинен мати наступні сторінки:

- «Пошук» - сторінка каталогу івентів з можливістю пошуку та фільтрації запитів;
- «Мої події» - список відвіданих та/або організованих заходів;
- Сторінка авторизації;
- Сторінка реєстрації;
- «Профіль» - особистий кабінет користувача, де можна редагувати власні дані, а також сторінка доступна для перегляду іншим користувачам платформи;
- «Створити подію» - сторінка додавання події з відповідними налаштуваннями;
- Сторінка власне події з її детальним описом та списком учасників;
- «Часті запитання» - сторінка, що міститиме інструкції користування сайтом, а також відповіді на запитання користувачів.

Реалізований сайт повинен бути адаптованим до мобільних пристроїв та планшетів, підтримуватись абсолютною більшістю браузерів та відповідати вимогам до швидкодії. Також базуючись на аналізі схожих сервісів, бажано, щоб платформу було реалізовано за допомогою сучасних фреймворків та бібліотек останніх версій для полегшення її підтримки в майбутньому та з метою довшого збереження актуальності поточної розробки.

2 Теоретичні відомості

2.1 Предмета область

Соціальність – це невід’ємна складова як сучасної людини, так і перших прадавніх людей. Людські потреби дуже різноманітні і, звичайно, наші потреби значно відрізняються від соціальних потреб наших пращурів. Проте дослідники виділяють такі потреби у спілкуванні [\[4\]](#):

- Бути індивідуальністю: виключно у соціумі людина розуміє власну значимість та унікальність;
- Домінування – це прагнення чинити активний вплив на рішення або думки інших, що зазвичай буває шкідливим для взаємовідносин, але задоволення в разі успішного домінування перевищує ризики зруйнувати стосунки;
- Турбота та заступництво за іншого – це спосіб отримати задоволення від допомоги іншим, також це може бути одним із способів самостредження;
- Отримати допомогу, такий взаємообмін присмний і для того, хто надав допомогу, і для того, хто її прийняв та досяг власної мети.

Людина завжди потребує оточення та спілкування, без нього – вона занепадає духом, просуватися по кар’єрній драбині стає все складніше, а комунікація з продавцем в магазині стає надзвичайно складним випробуванням. При цьому потреба виникає не обов’язково в розмовах, людям спілкуються переважно емоційними обмінами [\[5\]](#). Соціальне явище, при якому відбувається усунення особи від інших, через різке скорочення соціальних контактів називають соціальною ізоляцією. Світова пандемія COVID-19, що почалася 2019-го року, спричинила примусову соціальну ізоляцію багатьох людей. Наразі через великий вплив соціальний мереж простого соціального контакту недостатньо, кожен зацікавлений отримати якісне спілкування відповідно до власних інтересів та стилю життя. Саме тому проблема якісного соціального спілкування є такою актуальною.

2.2 Опис технічних відомостей

Якість та швидкодія програми напряду залежить від бібліотек, фреймворків та методологій, які застосовуються при її розробці, а отже важливо провести дослідження ринку сучасних технік та грамотно їх застосувати. Оскільки розробка веб-застосування передбачає постійну його підтримку та розширення функціоналу, в основу повинні лягати принципи SOLID [6] – базові принципи Об’єктно орієнтованого програмування та дизайну:

- Single responsibility (єдиний обов’язок) – один компонент виконує одну функцію;
- Open/closed (відкритості/закритості) – кожна сутність повинна бути доступною для розширення, але забороненою для змін;
- Liskov substitution (підстановка) – будь-який компонент може бути замінений нащадком без зміни коду;
- Interface segregation (розділення інтерфейсів) – декілька інтерфейсів під різні потреби краще за один уніфікований;
- Dependency inversion (інверсія залежностей) – абстрації в системі не повинні залежати від деталей.

Досконало застосувати всі принципи в єдиній розробці надзвичайно складно, проте їх застосування дозволяє значно покращити підтримку та масштабування системи, а також відлагодження та тестування системи. Досягати цих принципів при розробці веб-застосунку дозволяє трирівнева архітектура [7], яка передбачає наявність: клієнта - рівень представлення даних, в даній курсовій роботі – сайт; сервера – рівень обробки даних та зв’язку клієнта з даними; бази даних (надалі БД). Варто зазначити, що обмін даними між клієнтом та сервером, сервером та базою даних відбуваються за різними протоколами. Схема взаємодії рівнів у даній курсовій роботі представлена на рисунку 4.

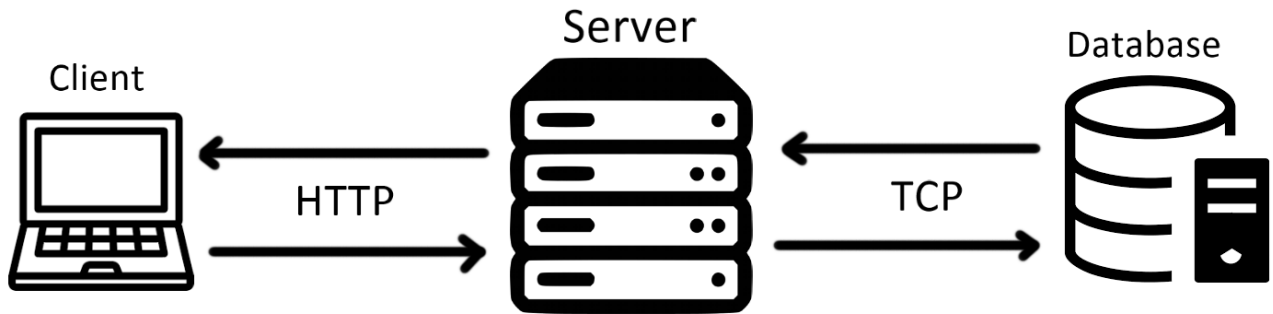


Рисунок 4 - Взаємодія між рівнями клієнт-верверної архітектури

Дана архітектура дозволяє створити «тонкого» клієнта та «товстого» сервера, завдяки такому принципу навантаження з браузера користувача зменшується і веб-застосування можна запускати на будь-яких пристроях. Окрім того, трирівнева архітектура має ряд інших переваг:

- Можливість розділення коду: методи серверу можуть використовуватись декількома різними клієнтами;
- Надійність та безпека: збої на одному з рівнів не впливають на інші, а розділення даних від основної програми дозволяє зберігати дані структуровано з додатковим (серверним) рівнем захисту;
- Масштабованість та зручність розробки: можна обрати будь-які мови програмування та розробляти всі рівні паралельно.

2.2.1 Клієнтська частина

Сучасний сайт повинен мати приємний користувацький інтерфейс, бути швидким, легким та кросбраузерним. В даній курсовій роботі за основу була взята JavaScript-бібліотека React [\[8\]](#), що дозволяє створювати інтерактивні користувацькі застосунки за рахунок роботи зі станами. React побудований за принципами декларативності та базується на використанні компонентів, що дає змогу описати вигляд та функціонал простих компонентів, а потім скласти з цих компонентів основну програму.

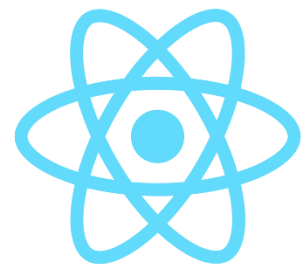


Рисунок 5 - React logo

React працює за принципом станів: один компонент завжди має однаковий вигляд в певному стані. В свою чергу стан об'єкта визначає набір змінних всередині нього. Щосекунди React оновлює сторінку, щоб перевірити чи не змінився один з його станів, проте користувач цього не бачить, адже бібліотека не запускає метод загального рендеру щоразу, натомість відбувається наступний порядок дій:

1. Перевірка станів – необхідно з'ясувати чи змінився хоча б один зі станів.
2. Якщо один зі станів змінився, знаходяться всі компоненти, що були залежні від даного стану.
3. В знайдених компонентах встановлюється які саме частини цих компонентів необхідно оновити відповідно до нового стану.
4. Відбувається оновлення, яке користувач бачить на екрані.

Оновлення сторінки можна налаштувати відповідно до власних вимог. В даній курсовій роботі зміна станів відбувається виключно після дій користувача або при зміні даних, що приходять з сервера. Однією з переваг при розробці з бібліотекою React є свобода дій, адже вона не встановлює строгі правила та обмеження на дії розробника і дозволяє інтегрувати будь-які інші бібліотеки, фреймворки, або ж власний HTML, CSS JavaScript код. Натомість є ряд рекомендацій від розробників бібліотеки, що дозволить максимально оптимізувати роботу сайту.

Одна з найцікавіших переваг React, що стала доступною з версії 16.8.0, це хуки (hooks). Хуки дозволяють виокремлювати та перевикористовувати код із статичною логікою, без впливу на компоненти або їх ієрархію в цілому. Вони підтримують всі вбудовані особливості React та на їх основі можна будувати власні, проте їх не можна використовувати поза React-функціями та всередині циклів, умов чи вкладених функцій. Найпоширеніший React-хук – `useState`, що дозволяє працювати зі станами як із звичайними змінними, синтаксис достатньо простий:

```
const [name, setName] = useState<string>('natalia')
```

де `name` – назва змінної, `setName` – сетер змінної, `string` – тип змінної, `'natalia'` – значення за замовчуванням. Також є безліч інших корисних вбудованих хуків, зокрема:

- `useEffect` – дозволяє щоразу виконувати певний набір функцій при початковому рендері або подальших оновленнях компоненту;
- `useContext` – дозволяє працювати з об'єктом контексту;
- `useMemo` – дозволяє використовувати мемоізацію компонентів, зменшуючи обрахунки при кожній зміні стану;
- `useRef` – дозволяє ініціалізувати та використовувати об'єкт, що відповідатиме за повний життєвий цикл іншого (частіше дочітнього) компоненту.

React також має безліч вбудованого функціоналу для покращення продуктивності та забезпечення взаємозв'язків між різними компонентами програми. `React.lazy` дозволяє динамічно імпортувати компоненти, таким чином компоненти, які повинні відображатись лише у виняткових станах не будуть імпортуватись та завантажуватись, допоки їх не потрібно буде відобразити. Функція `lazy()` також є дуже корисною при розробці багатосторінкового сайту: користувач вказує шлях відповідно до чого підвантажується лише необхідна частина коду, а в цей час користувач може бачити вами встановлений лоадер, який підвантажується за замовченням. Всі дані в React передаються від батьківських компонентів дочірнім, але необхідність в глобальних змінних повною мірою покриває використання контексту (`context`). Контекст дозволяє будь-якому компоненту отримати до нього доступ та використати його змінні, при цьому в контексті можуть зберігатися як дані, так і функції (наприклад `setters`), але варто пам'ятати, що більшість функцій краще виокремлювати в модулі.

TypeScript (TS [9]) – мова програмування з статичною типізацією, що розширює можливості JavaScript (JS). TypeScript передбачає можливі помилки, що виникають через неправильні типи та можуть виникнути під час компіляції або запуску програми, проте після компіляції він перетворюється в оптимізований JS.

Окрім того, TS-код значно легше читається. React рекомендовано використовувати з розширенням JSX, проте в даному проєкті використовується TSX – TS розширення, що дозволяє описувати HTML-теги та власні компоненти як частину TS-коду.

2.2.2 Серверна частина

Хороший веб-сервер повинен повною мірою забезпечувати обробку даних відповідно до бізнес-вимог, мати зручне API для використання клієнтами, бути швидким та незалежним. Один з методів, що досягає досягти даних вимог є мікрофреймворк Flask [10], що дозволяє створювати



Рисунок 6 - Flask logo

швидкодійні веб-API незалежно від обраних типів зберігання даних чи архітектури. Flask містить в собі вбудований обробник запитів, дозволяє легко формувати правильні відповіді, проте він не займається логічною обробкою даних, дозволяючи інтегрувати будь-які інші фреймворки чи бібліотеки, або ж створити власні.

Flask-RESTful – розширення для Flask, що дозволяє будувати REST-API, зокрема для HTTP, що використовується для взаємодії клієнтської та серверної частини в даному проєкті. Flask-RESTful не потребує складних налаштувань, проте надає значні переваги у використанні, зокрема:

- дозволяє розділити сервер на класи, за рахунок чого можна використовувати всі переваги парадигми ООП;
- логічно відділити різні частини програми та уникнути дублювання коду;
- забезпечити уніфікацію методів;
- реалізувати багаторівневість серверу (рівні валідації, попередньої обробки даних, роботи з базою даних, тощо);

Безпека даних та незалежність кожного методу реалізовано за допомогою middleware-валідаторів – функцій перевірки відповідності набору даних до конкретного методу API, що спрацьовують перед виконанням безпосередньо

методу та забезпечують повернення помилок з повідомленням про те, які дані та чому не пройшли валідацію. Також валідатори обмежують доступи до даних: частина методів повинні бути доступні виключно користувачам або обмеженій кількості користувачів, валідатори перевіряють чи є у клієнта необхідні дозволи і в разі чого блокує доступ до методів.

Основна функція веб-серверу – робота з даними, в даному випадку з базою даних. Сучасних варіантів баз даних безліч, одна з них PostgreSQL [\[11\]](#) – об’єктно-реляційна СУБД з відкритим кодом, що підтримує SQL-синтаксис запитів, а також дозволяє створювати значно більше розширень, ніж класичні БД. У даній курсовій роботі підключення бази даних було вирішено реалізувати за допомогою ORM – технології, що дозволяє працювати з даними як з об’єктами. Хороша ORM автоматично вирішує ряд проблем, який при роботі з класичними реляційними СУБД необхідно вирішувати написанням власних класів та перевірок. Одна з найкращих та найпопулярніших ORM для Python - це SQLAlchemy [\[12\]](#), що чудово інтегрується з фреймворком Flask, а також надає можливість працювати з реляційною базою даних як з групою об’єктів за рахунок “Core” сервісу без підключення ORM-компоненту. Використання SQLAlchemy має наступні переваги:

- покращення читання коду та масштабування БД прямо в коді програми;
- ефективність та швидкодія: порівняно з простими SQL-запитами ORM може програвати в швидкості, проте для роботи з великою кількістю взаємозалежних даних оптимальні запити до БД складе сама бібліотека;
- самостійно не генерує непотрібні поля чи додаткову інформацію, схему даних визначає розробник, відповідно в базі зберігаються лише корисні дані;
- мінімізація помилок: використовуючи вбудовані методи для формування запитів виключається можливість сформулювати запит некоректно;
- автоматичне кешування запитів та результатів для покращення швидкодії роботи серверу;
- реалізація перед- та пост-обробки запитів;

- додаткових захист даних від неправильного введення та SQL-ін'єкцій;

Сучасні алгоритми пошуку розвиваються неймовірними темпами, з'являється безліч нових способів та підходів шукати інформацію у великих каталогах за лічені секунди. Для створення пошуку подій в даній курсовій роботі було вирішено застосувати Elasticsearch [\[13\]](#) – пошуковий двигун, що дозволяє здійснювати повнотекстові пошуки по каталогу JSON-документів. Двигун доступний в хмарній версії, тому не потребує окремого сервісу



для хостингу, при цьому користувач може обрати хостинг якої платформи він хоче використовувати (Google Cloud, Azure або AWS) та керувати об'ємами пам'яті, які йому необхідні. Elastic надає можливості: пошуку за текстовими запитами, сортування та фільтрування результатів за значенням текстового поля, поля з номером чи датою. Всі документи за якими відбувається пошук зберігаються у вигляді окремих JSON-об'єктів, що індексуються пошуковою системою, за рахунок чого досягається висока швидкість отримання результатів відповідно до запиту. Система надає також веб-інтерфейс керування пошуковим двигуном, де окрім перегляду доданих об'єктів є можливість налаштовувати результати пошуку, зокрема:

- вказувати за якими полями в документах проводити пошук, а які ігнорувати;
- налаштовувати коефіцієнти ваги кожного поля при пошуку (при співпадінні у двох документах запиту в різних полях більш релевантним вважатиметься той, у якого більший ваговий коефіцієнт у даного поля);
- управляти доступами до пошукового двигуна за допомогою ключів та кодів.

Elasticsearch дозволяє легко масштабувати пошукову систему без часових втрат на пошук релевантних документів. Окрім того, платформа надає безліч додаткових сервісів (більшість з них платні), таких як: збір аналітики, різноманітні метрики, оцінка оптимізації власної програми, що використовує пошуковий двигун, а також різноманітні системи безпеки даних, що зберігаються в хмарі.

3 Опис реалізації програмного продукту







3.1 Технічне завдання

На основі описаних в розділі 1.3 задач, оперуючи знаннями у сфері організації групового дозвілля та розробки сучасних веб-застосунків, можна сформулювати чітке технічне завдання, а також визначити всі вимоги до програми та даних.

Цільовою аудиторією платформи є діти шкільного віку старших класів (від 13-ти років включно), а також студенти та дорослі люди (віком до 65 років в середньому), що добре володіють комп'ютером та зацікавлені в спілкуванні, онлайн-іграх, професійних вебінарах або інших видах групової діяльності в режимі онлайн.

3.1.1 Вимоги до інтерфейсу

Платформа повинна бути реалізована у вигляді кросплатформного сайту. Система з назвою «Eventizer» повинна мати унікальний логотип, який буде впізнаваним для користувачів. Застосунок повинен мати світлу тему, для основної палітри повинні бути використані наступні кольори:

-  #5B58A1 – основний колір для акцентів
-  #8585CC – другорядний колір для акцентів
-  #FF6B6B – колір помилок
-  #9BE564 – колір завершення або успішно виконаних запитів
-  #E0D758 – колір попередження
-  #E5E5E5 – основний відтінок сірого для приміток

Навігаційна панель повинна бути реалізована у вигляді бокового меню та повинна містити посилання на: «Пошук» та сторінку допомоги користувачам (для усіх користувачів); «Профіль», «Мої події» та «Додати подію» (лише для авторизованих користувачів). При цьому кнопки «Log in» (увійти до системи) та

«Log out» (вийти з системи) повинні розташовуватись і хедері. Навігаційна панель повинна бути на усіх сторінках, за винятком сторінок реєстрації та входу.

Сторінка «Пошук» повинна мати велике зручне поле для вводу запиту, панель для фільтрації та сортування результатів пошуку. Всі дані повинні відображатись пагіновано, користувач повинен мати змогу налаштувати кількість відображуваних подій на одній сторінці. Результати пошуку повинні відображатись у вигляді карток івентів, що містять фото, заголовок, короткий опис, з посиланням на сторінку відповідного івенту. Сторінка «Мої події» повинна відображати всі події які користувач вже відвідав або планує, при цьому історія подій користувача повинна логічно відділятися. «Профіль» користувача повинен відображати всі дані про учасника або організатора, які зберігаються в базі даних, сторінка реєстрації повинна містити всі відповідні поля вводу.

3.1.2 Функціональні вимоги

При розробці веб-застосування необхідно виконати наступні загальні вимоги:

- Незареєстрований користувач повинен мати змогу зареєструватися як учасник або організатор;
- Організатори з непідтвердженою електронною поштою не повинні мати змоги увійти до платформи;
- Якщо у профіля користувача або івенту не встановлене фото, воно повинно бути замінене на зображення за замовчуванням;

Будь-який користувач повинен мати змогу:

- Шукати публічні івенти за ключовими словами в назві та описі;
- Фільтрувати результати пошук за типом (офлайн, онлайн), за датою, за обмеженням в кількості учасників;
- Сортувати результати пошуку за назвою, датою, кількістю учасників;
- Переглядати загальну інформацію про публічні івенти та їх організаторів;

- Переглядати сторінку користувацької підтримки та мати змогу задати власне питання;
- Отримати доступ до загальної інформації та посилання (або адреси) на приватну подію, якщо він отримав посилання;
- Авторизуватися в системі за нікнеймом та паролем, якщо раніше він був зареєстрований;

Авторизований користувач повинен мати змогу:

- Ставати учасником публічних івентів;
- Створювати приватні івенти та запрошувати інших користувачів;
- Переглядати історію відвіданих подій, а також перелік подій, до яких він долучився, щоб відвідати в майбутньому;
- Переглядати детальну інформацію про івенти, а також профілі їх організаторів та учасників;
- Переглядати та редагувати інформацію власного профілю;
- Видалити свій профіль;
- Закінчити сесію (вийти з системи);

Організатор повинен мати змогу:

- Створити публічний івент;
- Редагувати інформацію про власні івенти та їх організаторів;
- Видавати права доступу іншим організаторам до власних івентів;
- Видаляти з власних івентів учасників;
- Переглядати історію всіх організованих подій, окремо від історії відвіданих;

Окрім того, доступ до публічних подій повинен бути обмеженим. Лише зареєстровані користувачі платформи можуть переглядати інформацію та приєднуватися до приватних подій, виключно за сформованим кодом.

3.1.3 Вимоги до даних

Всі дані, які використовує система та надає користувачам, повинні зберігатися в базі даних у нормалізованому вигляді. При цьому паролі від особистих кабінетів користувачів повинні зберігатися в хешованому вигляді, а фотографії профілів та івентів повинні бути завантажені на third-party сервіси і зберігатись у базі даних як посилання.

Необхідно в структурованому вигляді зберігати наступну інформацію:

- Персональні дані учасника:
 - обов'язково: нікнейм, пароль, ім'я, дату народження, електронну пошту, роль;
 - за бажанням для учасника, обов'язково для організатора: прізвище, фото, номер телефону;
 - обов'язково для організатора: сферу діяльності;
 - за бажанням: статус;
- Інформацію про івент:
 - обов'язково: назва, початкова дата та час, посилання на онлайн-зустріч або адреса, власник події;
 - за бажанням для приватних івентів, обов'язково для публічних: опис, фото, дата та час закінчення події, сфера діяльності;
 - за бажанням: обмеження, щодо кількості учасників;
- Перелік сфер діяльності, що доступні для вибору організаторам та подіям, що створюються;
- Популярні питання користувачів з відповідями для сторінки FQA.

Системою повинна забезпечуватись валідація даних, при цьому введені користувачів дані повинні валідуватися на клієнтській частині, а потім валідуватися сервером на предмет відповідності типів, символів, довжини, тощо.

До усієї інформації, що зберігається в базі даних повинні бути застосовані наступні корпоративні обмеження цілісності:

- Вік учасника повинен бути більшим за 13 (тринадцять) років;
- Вік організатора повинен бути більшим за 18 (вісімнадцять) років;
- Нікнейм кожного користувача, електронна пошта та номер телефону повинні бути унікальними в межах системи;
- Кожна назва івенту повинна бути унікальною в межах інших івентів, які ще не закінчилися;
- Організатор не може створити два та більше публічні івенти, що перетинаються в часі (мають однакову дату, а часові проміжки перетинаються);
- Сфера діяльності організатора, а також тематика заходу повинні встановлюватись з визначеного переліку;
- Користувач, зареєстрований як учасник, не може створювати публічні події.

3.2 Архітектура веб-застосування

Застосування має базову трирівневу архітектуру, описану в [розділі 2.2](#). Окрім бази даних до веб-серверу підключено:

- пошуковий двигун Elasticsearch (через HTTP протокол);
- сервіс доступу до поштових скриньок SMTP (Simple Mail Transfer Protocol);
- прикладний інтерфейс програмування Google Map (через HTTP протокол), для визначення ат відображення адрес проведення офлайн-івентів.

Усі вищенаведені сервіси доступні для кінцевого користувача виключно через сервер даного застосунку, а не підключені до клієнтської частини напряму. У таких спосіб забезпечується безпека користування сторонніми сервісами та правильна обробка даних, що надсилаються на приходять з сторонніх ресурсів. Окрім того, сервер може одночасно опрацьовувати запити від різних клієнтів, адже

сайтом одночасно можуть користуватися декілька користувачів з різних девайсів. Загальна архітектура наведена на рисунку 8.

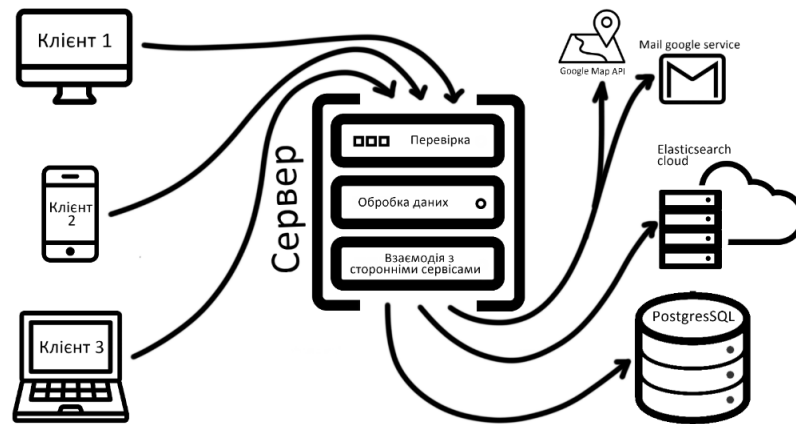


Рисунок 8 – Архітектура застосунку Eventizer

З боку користувача внутрішня архітектура застосунку непомітна, для нього архітектура веб-застосування виглядає як набір доступних для нього дій та переходів. Діаграма використання (Use case diagram) наведена на рисунку 9

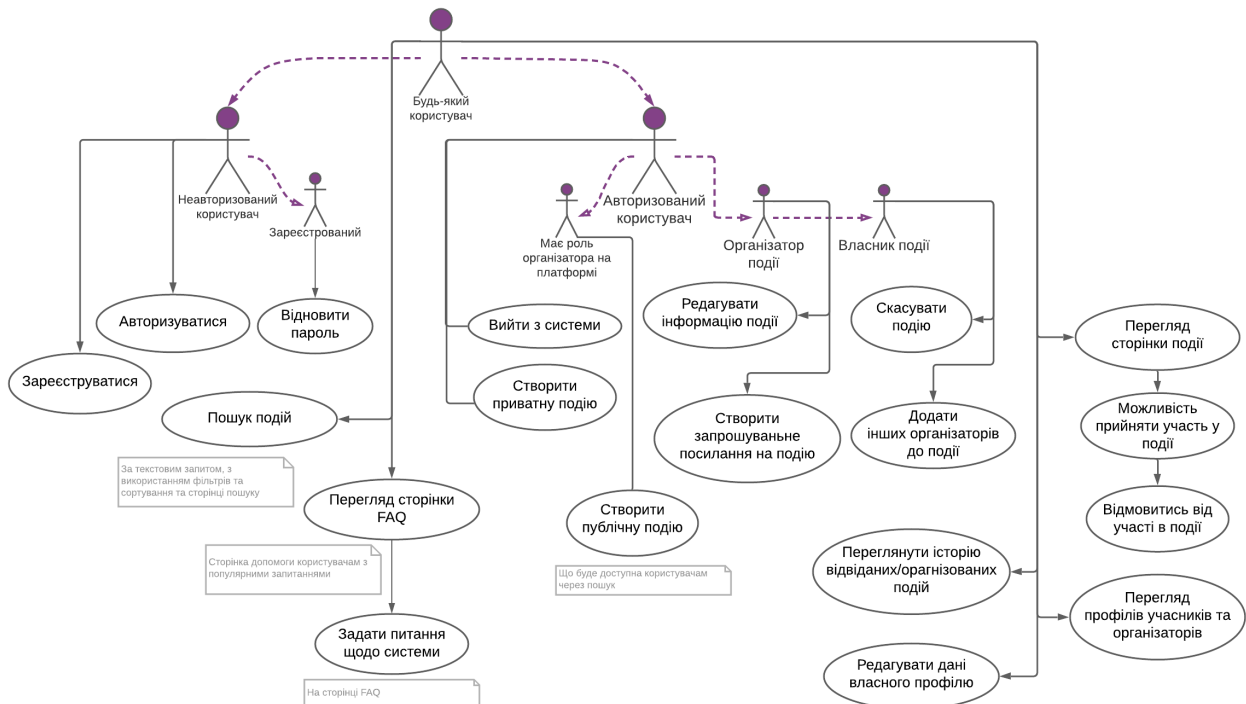


Рисунок 9 - Use case diagram

3.3 Обґрунтування вибору засобів розробки

Сучасний веб-застосунок потребує сучасних підходів до розробки програмного забезпечення. Саме тому клієнтська частина була розроблена на типізованому розширенні класичної мови розробки веб-застосувань - TypeScript за допомогою бібліотеки React. React дозволяє будувати сучасні швидкі та легкі сайти, він дозволяє створювати власні функціональні елементи та перевикористовувати їх для побудови цілих сторінок. Для забезпечення оптимального завантаження багатосторінкового сайті використовувалася бібліотека react-router-dom, що дозволяє використовувати велику частину спільних налаштувань єдиною для всіх сторінок, і змінювати лише обрані відображувані елементи відповідно до вказаного користувачем шляху. Бібліотека styled-components дозволяє перетворювати звичайні HTML-теги в стилізовані функціональні компоненти, що легко імпортуються іншими елементами верстки. Окрім того, для забезпечення єдиного лаконічного стилю системи було використано декілька бібліотек, що втілюють концептуальна філософія дизайну:

- @material-ui/core – містить набір базових компонентів єдиного стилю;
- @material-ui/icons – каталог іконок, для стилізації сайту та покращення зорового сприйняття;
- @material-ui/lab – складні компоненти, що містять власні додаткові функції;
- @material-ui/pickers – стилізовані інпути дати та часу відповідно до компонентів @material-ui/core.

Використовуючи material-ui завжди можна бути впевненими в тому, що сайт виглядає актуально та приємно для користувачів. Ще однією перевагою є те, що кожен окремий компонент вже розроблений з врахуванням адаптивного дизайну, проте за розробником залишається адаптивне розміщення всіх елементів. Для взаємодії з картами та їх коректного відображення використовувалися бібліотеки

рекомендовані розробниками google та React: react-places-autocomplete, react-google-maps.

Оскільки рекоменований стиль написання складових слів для TypeScript (та JavaScript) – це CamelCase (з англ. верблюжий регістр), а для Python, що використовується для написання серверної частини – snake_case (з англ. зміїний регістр), для уникнення конфліктів в назвах змінних було використано бібліотеки:

- snakecase-keys – трансформує дані перед відправкою на сервер;
- camelcase-keys – перетворення даних з сервера в правильний формат для клієнтської частини;

Для створення серверної частини було використано фреймворк Flask з розширенням Flask-RESTful та бібліотекою Flask-SQLAlchemy, переваги яких наведені в [розділі 2.2.2](#), проте окрім власне бібліотеки фреймворку було використано ряд допоміжних бібліотек, зокрема:

- Flask-Cors – для встановлення правильних заголовків у відповідях серверу;
- Flask-Migrate – для створення міграцій – окремих файлів з версіями бази даних, що дозволяє використовувати правильну версію бази даних відповідно до версії коду, а також самостійно оновлювати базу даних до останньої актуальної версії;
- flasgger – бібліотека для створення професійної документації до API;
- Flask-Parameter-Validation та decorator для створення проміжних валідаторів даних, що приходять на сервер, дані бібліотеки дозволяють перевіряти дані перед запуском власне методу;
- validators – набір вбудованих валідаторів, наприклад: електронної пошти, посилання, номеру телефону, тощо;
- uuid – для генерування складних унікальних послідовностей символів, таких як id подій та кодів доступу.

3.4 Опис процесу розробки програми

Веб-застосування – це не просто набір файлів, що виконують певні функції, це система, що дозволяє користувачам вирішувати певні проблеми або задовольняти власні бажання. Відповідно розробка такої системи є складним процесом, що можна поділити на наступні етапи:

1. Формування ідеї: перш ніж створювати веб-застосунок необхідно виділити його основні цілі та визначити цільову аудиторію.
2. Валідація ідеї: опитування цільової аудиторії та визначення ключового функціоналу, необхідного для реалізації.
3. Вибір засобів розробки та написання технічного завдання.
4. Створення дизайну майбутнього застосунку.
5. Розробка архітектури клієнтської та серверної частин, формування схеми майбутньої бази даних.
6. Створення базової верстки на клієнтській частині та базового функціоналу кінцевих методів API на сервері.
7. Інтеграція клієнтської та серверної частини, підключення доступу до певних методів лише за токеном авторизації.
8. Впровадження валідаторів та розширення поточного функціоналу.
9. Деплой готового веб-застосування на хмарних платформах.
10. Тестування програми та її відлагодження.
11. Написання документації та формування звітів.

Важливим етапом є розробка архітектури клієнтської та серверної частин, адже саме цей етап визначає наскільки вдалою буде система для пітримки та масштабування в майбутньому. Не зважаючи на те, що в межах даної курсової роботи розроблялося фактично MVP (minimal valuable product), увесь функціонал був поділений на декілька ітерацій:

- створення подій, їх пошук та перегляд деталей;

- реєстрація користувача, авторизація, функціонал профілю та участі в подіях;
- складні налаштування подій: декілька організаторів, скасування, обмеження доступу до приватних подій за кодом.

Дизайн майбутньої платформи був створений у Figma [14] з використанням вбудованого функціоналу глобальних стилів, компонентів та їх варіантів, а також Auto layout (аналог властивостей групи flex у CSS). Варто зазначити, що під час розробки стиль неодноразово покращувався, а деякі екрани були реалізовані одразу в верстці, оскільки їх реалізація в межах MVP не планувалася.

3.5 Розробка основної програми

Розробка усіх функцій та методів як на клієнтській частині, так і на серверній відбувалася відповідно до груп та модулів до яких належали дані частини. Проте клієнт та сервер значно відрізняються одне від одного.

3.5.1 Клієнтська частина

Файлова структура клієнту наведена на рисунку 10. У кореневому каталозі з назвою репозиторію містяться файли конфігурації серверу та README.md з описом запуску програми. Директорія build містить оптимізовані файли для запуску, node_modules – файли бібліотек, що використовуються, public – вхідний html файл, favicon.ico та robots.txt. Каталог src є корневим, саме в ньому знаходяться всі початкові файли коду, а також зображень, що використовуються (в папці assets).

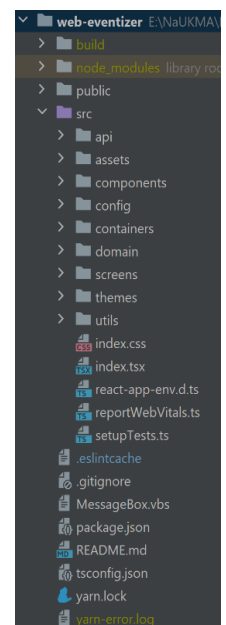


Рисунок 10 –
Файлова
структура
клієнтської
частини

Папки components, containers та screens – це функціональні компоненти, що використовуються програмою та відображаються користувачеві. Варто зазначити, що саме елементи папки screens складаються з усіх інших елементів, containers також складаються з компонентів, проте жоден з них не є фінальним екраном. У config міститься єдиний файл змінних,

що заповнюються даними з змінних середовищ або за їх відсутності має дефолтні значення. Каталог api містить набір функцій та зокрема клас `ApiClient` для підключення та обміну даними з сервером, utils – набір допоміжних функцій, що використовуються в інших частинах програми, domain – типи даних, що повертаються з серверу та якими компоненти обмінюються між собою. Варто зазначити, що всі завантажені користувачем фото профілю або зображення подій завантажуються на сторонній сервіс `imgbb`, після чого на сервер вже надсилається лише посилання на зображення, таким чином розмір даних, що передається суттєво зменшується, а фотографії зберігаються окремо від основних даних. У themes – файли теми (кольорової гами), але на даному етапі присутня лише одна тема за замовчуванням, хоча застосунок дуже легко може розширюватись іншими темами, зокрема темною.

Коли користувач відкриває сайт в браузері, спочатку відбувається рендер об'єкту `App` з директорії `src/screens`, що визначає до якої сторінки користувач хоче отримати доступ. Після цього, відбувається рендер компоненту екрану, що відповідає запиту користувача (компонент імпортується лише після виклику, що забезпечує швидке відмальовування лише необхідного контенту). Поки контент імпортується, користувач бачить лоадер, всі процеси завантаження забезпечені окремими елементами, що відображаються процес завантаження компонентів або очікування запитів. Деякі сторінки, такі як профіль, сторінка івенту, «Мої події», спочатку завантажують дані з серверу, а потім відображають їх, інші здійснюють запити на сервер лише після дії користувача: застосування фільтрів на сторінці пошуку, створення події, логін та реєстрація, функції записатися/виписатися з події, а також інші функції редагування.

Деплой клієнтської частини відбувається за допомогою піклученої `GitHub` репозиторії до `Heroku`, при цьому автооновлення задеплойної версії відбувається, щоразу, при `push` в гілку `master`. Щоб локально запустити програму необхідно перейти в кореневий каталог, після чого в консолі виконати команду: `npm start`, при

цьому в змінних середовища необхідно вказати `REACT_APP_BASE_URL` (або запуснути проект серверу локально) та `REACT_APP_PHOTO_KEY` – для завантаження фото в хмарний сервіс.

3.5.2 Серверна частина

На рисунку 11 наведено файлову структуру серверу, при цьому модулі позначаються іконкою каталогу з крапочною. В кореневому каталозі містяться файли налаштування репозиторії, `requirements.txt` – перелік бібліотек, необхідних для роботи серверу, а також файл конфігурації серверу, де ініціалізуються всі глобальні сервіси, – `configuration.py` та запуску серверу – `app.py`, в якому також оголошуються всі ендпоінти. У `config` містяться файли змінних, що використовуються сервером для налаштування, зокрема ключі підключення до бази даних та Elasticsearch, але програма візьме їх виключно за відсутності цих даних у змінних середовища. В папці `migrations` містяться файли міграцій та допоміжні файли для перевірки поточної версії підключеної бази даних, наразі міститься 8 версій бази даних.

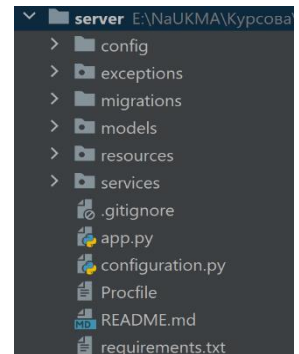


Рисунок 11-
Файлова
структура
серверу

Модуль `exceptions` містить перелік кастомних помилок, що використовуються сервером для формування правильних відповідей з відповідними кодами та повідомленнями; зараз окремо від інших оброблюється 8 різних помилок. У модулі `models` містяться класи, що відповідають таблицям у базі даних. Кожен клас містить від 1 до 4 властивостей серіалізації та використовуються для повернення методами серверу правильних наборів даних. Дані класи також використовуються для автоматичного створення нових міграцій. Наразі існують класи: `Event`, `User`, `Area`, `Token`, `FAQ`, `UserEvent`, `Email`, `JoinCodes`. Модуль `services` містить класи сервісів, що ініціалізуються в файлі конфігурації та виконують логічно обмежений набір функцій, зокрема:

- ElasticSearch – для обміну даними з пошуковим двигуном, розташованим в хмарі, містить окремі методи додавання, редагування та видалення документів, а також метод пошуку;
- TokenManager – клас управління токенами, логіка якого повністю відокремлена від основної програми;
- EmailSender – сервіс для відправки листів, відповідно до методу надсилається певний макет електронного листа у форматі HTML з певними даними.

Модуль resources – це фактично голосний модуль, що реалізує основний функціонал програми. Він містить класи ресурсів, що перевіряють дані на правильність, обробляють їх, використовують сервіси, працюють з об'єктами бази даних та повертають помилки за потреби. Також в модулі наявні каталог flasgger, що містить uml файли з документацією до всіх методів серверу (повний перелік доступний [за посиланням](#)), а також підмодулі decorators, для валідації даних перед безпосереднім викликом методу, та utils з допоміжними функціями, що використовуються класами ресурсів.

Загальний процес обробки запитів наведений на рисунку 12.

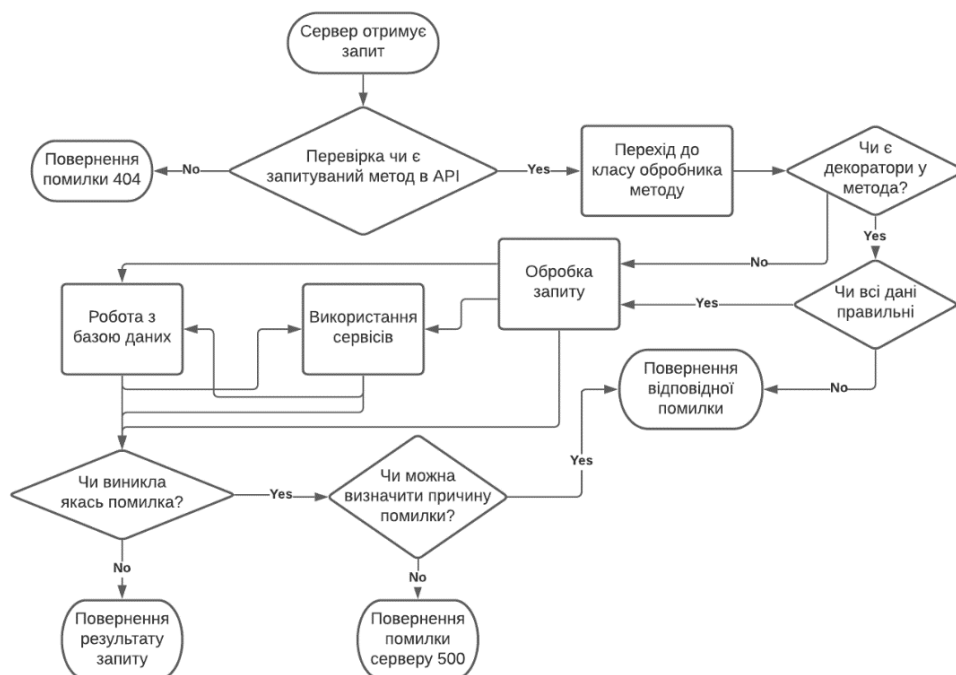


Рисунок 12 – Схема обробки запиту сервером

3.6 Опис бази даних та інтерфейсів програми

У даному проекті всі дані зберігаються в базі даних, схема якої наведена на рисунку 13. Таблиця `alembic_version` використовується бібліотекою міграцій для визначення поточної версії, всі інші таблиці зберігають власне програмні дані.

The diagram illustrates a database schema with the following tables and their attributes:

- events**
 - id: varchar(45)
 - title: varchar(255)
 - description: text
 - start_time: timestamp
 - end_time: timestamp
 - area: integer
 - is_online: boolean
 - is_public: boolean
 - image: text
 - min_member: integer
 - max_member: integer
 - location: text
 - main_link: text
 - person_amount: integer
 - is_deleted: boolean
- areas**
 - id: integer
 - name: varchar(255)
- users**
 - id: integer
 - firstname: varchar(45)
 - lastname: varchar(45)
 - login: varchar(45)
 - password: varchar(255)
 - photo: text
 - birth: date
 - phone: varchar(13)
 - email: varchar(255)
 - role: varchar(45)
 - area: integer
 - status: text
- join_codes**
 - code: varchar(15)
 - event_id: varchar(45)
 - expiration_date: timestamp
 - notes: varchar(255)
- events_users**
 - event_id: varchar(45)
 - user_id: integer
 - role: varchar(45)
- a1enbic_version**
 - version_num: varchar(32)
- emails**
 - email: varchar(255)
 - expiration_date: timestamp
 - code: varchar(10)
 - confirmed: boolean
 - reason: varchar(255)
- tokens_blacklist**
 - id: integer
 - expired_token: text
- faq**
 - id: integer
 - question: text
 - answer: text

Relationships (Foreign Keys) shown by blue arrows:

- events** (id) to **events_users** (event_id)
- events** (area) to **areas** (id)
- events_users** (user_id) to **users** (id)
- join_codes** (event_id) to **events** (id)

Powered by vFiles

Рисунок 13 – Схема бази даних

Таблиця events зберігає повну інформацію про події, при цьому варто зазначити, що поле `is_deleted` визначає чи зможуть користувачі приєднуватися до події, чи подія вважається видаленою (реального видалення подій немає, щоб користувачі завжди могли бачити історію своїх подій). Таблиця подій посилається на таблицю areas, що містить перелік сфер діяльності, які необхідно вказувати для публічних подій та користувачам, що реєструються як організатори. Власне інформація про користувачів зберігається в таблиці `users`, що також має зовнішнім ключем значення сфери діяльності, поля `id`, `email`, `phone` та `LOG IN` є унікальними в межах таблиці, паролі зберігаються в хешованому вигляді (хешуються бібліотекою `passlib`). Таблиця events_users зберігає інформацію про участь користувача в події, при цьому користувач може бути: власником події, організатором чи учасником. Всі ролі надають різний рівень доступу до події. Адміністратор події може створити коди на приєднання до неї, всі вони, а також інформація про термін їх дії, зберігатиметься в таблиці join_codes.

Оскільки сервіс напряму підключений до пошукового двигуна Elasticsearch, який також зберігає деяку інформацію про публічні події, на рисунку 14 наведено схему даних документів `elastic`. Про подію зберігаються лише дані, за якими проводиться пошук, сортування чи фільтрація. Власне пошук відбувається лише за текстовим запитом, при цьому для різних полів встановлено різні коефіцієнти релевантності:

- `title` (назва події) – 1.3,
- `description` (опис події) – 1,
- `area` (сфера діяльності) – 0.7,
- `address` - 0.5.

<code>id</code>	
<code>end_date</code>	date ▾
<code>area</code>	text ▾
<code>address</code>	text ▾
<code>max_member</code>	number ▾
<code>end_time</code>	date ▾
<code>description</code>	text ▾
<code>title</code>	text ▾
<code>min_member</code>	number ▾
<code>start_time</code>	date ▾
<code>person_amount</code>	number ▾
<code>is_online</code>	number ▾
<code>start_date</code>	date ▾

*Рисунок 14 –
Схема даних
Elasticsearch*

3.6.2 Користувацький інтерфейс

Інтерфейс користувача складається з одинадцяти основних сторінок, більшість яких мають хедер, що містить логотип з посиланням на основну сторінку та кнопку «LOG IN» (або «LOGOUT» якщо користувач авторизований), та бокове навігаційне меню з переліком доступних сторінок. Всі сторінки під час завантаження показують відповідний компонент, щоб користувач розумів, що сторінка не зависла. Для дій, що виконуються після дії користувача також забезпечена анімація завантаження, зокрема, якщо дія відбулася після натиску на кнопку, на кнопці відображається ладер.

Система має зручний захист від випадкових дій – всі дії, які неможливо або важко скасувати забезпечені модальними вікнами для підтвердження дії. Окрім того, модальні вікна також присутні для: зміни паролю на сторінці профілю, перегляду всіх учасників події, редагування події та додавання організаторів до події. Також варто зазначити, що у випадку відображення списків інформації, що приходять з сервера, список порожній, користувач побачить відповідне повідомлення. У випадку помилки на стороні серверу, інтерфейс користувача продовжить функціонувати, показавши користувачеві відповідну помилку. Також абсолютна більшість іконок чи надписів, що можуть викликати питання у користувачів системи, забезпечені пояснювальними тултіпами (текстовими підказками графічного інтерфейсу), що відображаються при наведенні на елемент.

Інтерфейс користувача станом на квітень 2021 доступний [за посиланням](#).

3.6.3 Інтерфейс серверу

Інтерфейс серверу – це набір методів API, що доступні для клієнта. Наразі заголовки CORS встановлені виключно для клієнта даного проекту в цілях безпеки, проте з легкістю можуть бути розширені. Наразі API має тридцять два методи, хоча чотири з них клієнтом не використовуються, вони закладені як методи, що повинні першими використовуватись в розробленій адмінці для управління сферами діяльності та частими питаннями. Серед присутніх методів: 14 методів GET, 9 – POST, 5 – PUT, 4 – DELETE. За логічним поділом: 2 методи сфер діяльності, 3 – авторизації, 6 – подій, 4 – приєднання до подій за кодом, 5 – частих запитань, 4 – користувачів, 3 – управління учасниками події, 5 – без виокремленої тематики.

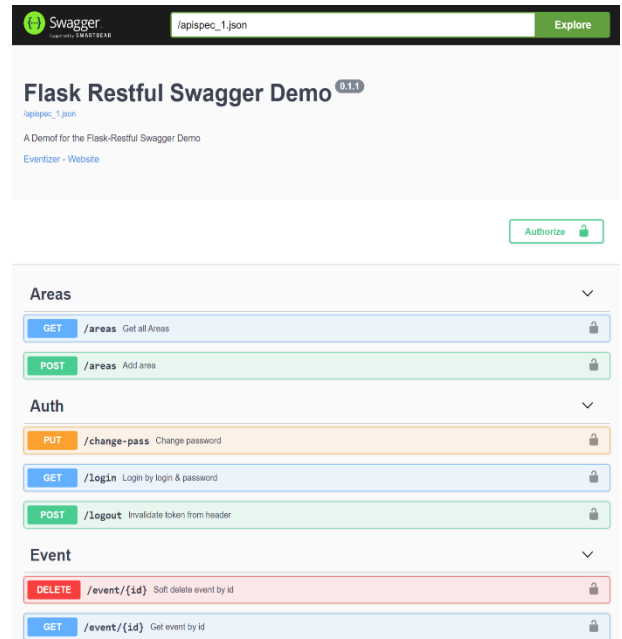


Рисунок 15 – Документація інтерфейсу сервера

Інтерфейс серверу, візуалізований у вигляді документації на рисунку 15, станом на квітень 2021 доступний [за посиланням](#).

3.7 Інструкція користувача

При першому вході на сайт користувач бачить сторінку пошуку подій, при цьому за замовченням відображаються події в порядку їх створення (рисунок 16), користувач також бачить хедер та бокове навігаційне меню. Користувач може сортувати результати пошуку чи фільтрувати їх за сферою

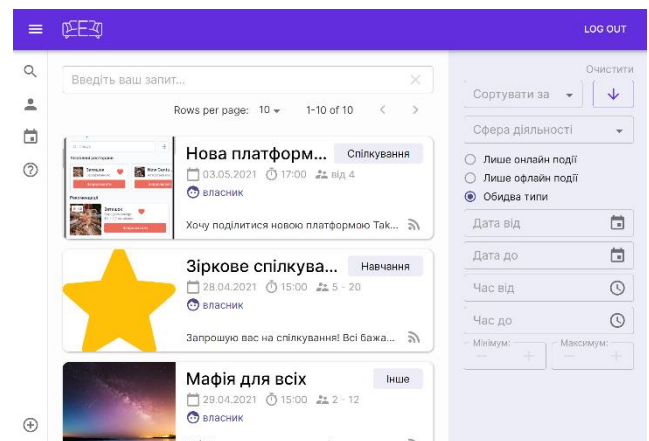



Рисунок 16 – Сторінка пошуку Eventizer

діяльності, за датою та/або часом проведення, за типом (онлайн, офлайн), за кількістю учасників. При натисненні на картку події в результатах пошуку, користувач може перейти на сторінку події ([рисунок 22](#)) та переглянути її деталі, проте доступу до перегляду сторінок організатора чи учасників немає. Натиснувши на іконку  в хедері, можна відкрити пояснення до іконок бокового меню, проте перейти на іншу сторінку за допомогою бокового меню можна і без повного його відкриття.

Сторінка “FAQ” ([рисунок 23](#)) – це сторінка підтримки користувачів, де міститься посилання на дану інструкцію, відповіді на часті питання та форму для власного питання, якщо його немає на сайті. Якщо користувач хоче надіслати власне запитання, він повинен ввести його в полі «Ваше запитання» та натиснути кнопку «Надіслати», після його він отримає повідомлення «Ваше запитання успішно додане!» або «Ваше запитання вже було задано раніше. Відповідь скоро з'явиться на сайті». Відповідь з'явиться в даному розділі щойно адміністратор надасть на нього відповідь.

Щоб авторизуватися в системі необхідно натиснути на кнопку “LOG IN” у хедері, після чого на сторінці входу ([рисунок 17](#)) ввести власний логін, пароль та натиснути «Увійти». Якщо користувач забув пароль можна скористатися системою відновлення паролю натиснувши на «Забули пароль?». Після чого ввести електронну пошту, потім код підтвердження, що був висланий на пошту, а також новий пароль ([рисунок 24-26](#)); тепер користувач може увійти в систему використовуючи новий пароль. У випадку, коли користувачу необхідно зареєструватися на платформі можна натиснути кнопку «Зареєструватися». На сторінці реєстрації необхідно:

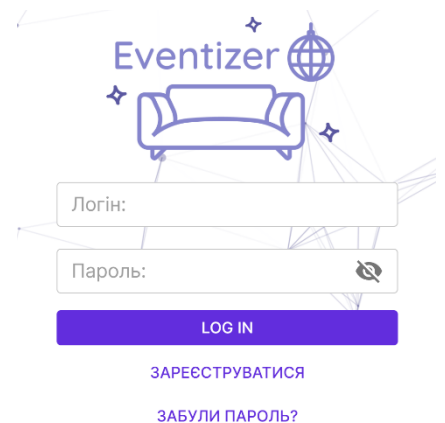


Рисунок 17 – Сторінка входу Eventizer

- 1) Вказати електронну пошту та обрати роль: організатор чи учасник ([рисунок 27](#));
- 2) Якщо обрали роль організатора, необхідно ввести код підтвердження, що надійшов на вказану електронну пошту ([рисунок 25](#));
- 3) Ввести необхідні персональні дані (обов'язкові поля позначені зірочкою) та натиснути «Завершити» ([рисунок 28](#)).

Авторизований користувач в боковому навігаційному меню, окрім сторінок «Пошук» та «FAQ», може перейти на сторінки «Профіль», «Мої події» та «Створити подію». Щоб вийти з системи необхідно натиснути «LOG OUT» в хедері.

Профіль користувача (рисунок 18) містить всю інформацію, яку користувач увів при реєстрації. Щоб редагувати ім'я, фото чи статус необхідно натиснути кнопку «Редагувати профіль», а потім «Зберегти» або «Скасувати» зроблені зміни. Для зміни пароля треба натиснути «Змінити пароль», після чого у модальному вікні ([рисунок 29](#)) ввести поточний пароль, новий та підтвердити новий пароль, потім натиснути «Змінити пароль» для застосування змін. Після зміни паролю при наступному вході використовувати вже новий пароль.

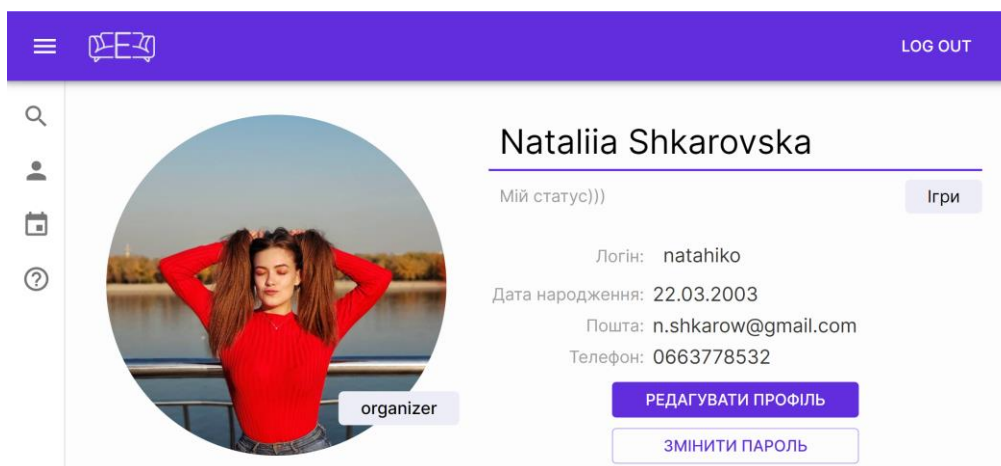



Рисунок 18 – Профіль користувача

Сторінка «Мої події» ([рисунок 30](#)) містить перелік подій, які користувач планує відвідати (або повідомлення, що таких подій немає: [рисунок 31](#)). Натиснувши на іконку «Історія»  можна переглянути вже відвідані події, якщо

такі є. Всі скасовані події відображаються в списку ваших подій, проте вони позначені як «Видалені» ([рисунки 32-33](#)). До скасованих подій не можна приєднатися, проте з них можна виписатися, якщо ви були їх учасником, інформацію про скасовані події редагувати також не можна.

Власне сторінка майбутньої події для авторизованого користувача також виглядає інакше, відповідно до ролі користувача в даній події:

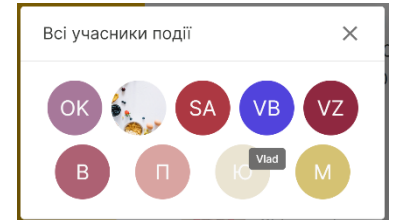


Рисунок 19 – Перегляд учасників події

- Якщо користувач немає жодної ролі в даному івенті: він може стати її учасником натиснувши «Взяти участь» та переглянути всіх користувачів ([рисунок 19](#));
- Якщо користувач є учасником даного івенту: він може перейти на платформу зустрічі натиснувши «Приєднатися до зустрічі» для онлайн зустрічей, або подивитися розташування офлайн зустрічі, натиснувши «Переглянути на карті», або припинити приймати участь в даній події натиснувши «Покинути зустріч» (після цього, подія не відображатиметься в списку подій користувача, якщо подія була приватна користувач не зможе бачити інформацію про неї);
- Якщо учасник є організатором зустрічі він також може редагувати інформацію про неї ([рисунок 34](#)) та видаляти користувачів з події ([рисунок 20](#)), а також управляти посиланнями з кодами доступу до даної події (видаляти їх або копіювати просто натиснувши на посилання чи іконку біля нього - [рисунок 35](#));
- Якщо учасник – власник івенту: окрім всіх дій, доступних для організаторів він також може додавати або видаляти організаторів ([рисунки 36-37](#)), а також скасувати подію.

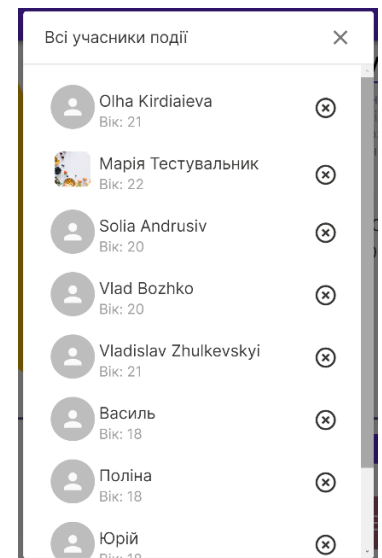


Рисунок 20 – Перегляд учасників події організатором

Перейти на сторінку створення події ([рисунки 38-39](#)) можна натиснувши відповідну кнопку в боковому меню. Варто зазначити, що користувач зареєстрований як учасник не може створити публічну подію (перемикач якого типу подію створити відсутній). Обов'язкові поля для заповнення позначені зірочкою, також допоки всі обов'язкові поля не заповнені кнопка «Створити» не буде активною. У випадку, якщо створена користувачем подія не відповідає певним вимогам платформи, повідомлення про помилку з описом невідповідності з'явиться на сторінці. При успішному створенні події, користувач побачить відповідне модальне вікно, після натиснення «Ок» його буде переадресовано на «Мої події».

Приватні події недоступні в пошуку та за прямим посиланням (при спробі перейти на сторінку приватної події користувачем, що не є її членом буде відповідне повідомлення - [рисунок 40](#)). Щоб приєднатися до події необхідно мати спеціальне посилання на приєднання по якому авторизований користувач зможе приєднатися натиснувши кнопку «Взяти участь» (рисунок 21). Якщо користувач не авторизований при спробі перейти за даним посиланням, він буде переадресований на сторінку входу, проте після авторизації система сама поверне його на сторінку приєднання до події.

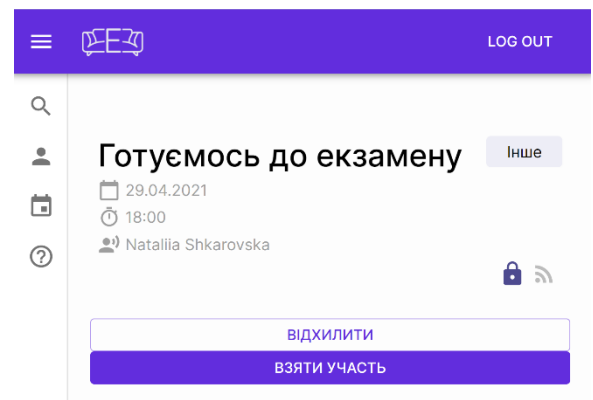


Рисунок 21 – Приєднання до приватної події

3.8 Перспективи розробленої платформи

Дана платформа є дуже актуальним рішенням в умовах світової пандемії, яка, на жаль, невідомо скільки ще триватиме. Більше того, навіть по її завершенню платформа має чудові перспективи розвитку для популяризації офлайн подій. Перша ітерація розробки даної платформи не перевантажена складним для

користувачів функціоналом, проте має безліч функцій, які корисно впровадити на наступних версіях, зокрема:

- портал новин платформи та сторінка анонсів публічних подій;
- окремий блок на сторінці події з прикріпленням корисних матеріалів, посилань, та порад по темі даного івенту;
- соціальна мережа шукачів цікавого дозвілля: надати користувачам змогу відстежувати події обраних організаторів або слідкувати за відвіданими подіями друзів;
- система рекомендацій: на основі інтересів та вподобань пропонувати події, що можуть зацікавити користувача;
- система сповіщень (про початок події або її скасування, налаштовувані сповіщення за декілька годин до початку) та інтеграції з календарями;
- система опитувань учасників події та збору фідбеків про івент, а також система рейтингів організаторів;
- створення платних подій та долучення лише після оплати;
- рекламні інтеграції: для реклами даного сервісу на реклами подій як всередині платформи, так і в інших мережах.

Додатково, якщо платформа набуде популярності, виникне потреба в окремій частині адміністрування сайту: жорсткіша перевірка організаторів для забезпечення якісного публічного контенту; сторінка відповідей на публічні запитання та розширення системи взаємодії з користувачами; управління сферами діяльності; можливість блокування користувачів та публічних подій, що пошурують правила користування платформою.

Платформа Eventizer дійсно має можливість стати єдиною центрацілозаною системою для організації спільного дозвілля, пошуку однодумців, полегшення процесів командних домовленостей, що допоки залишається актуальною проблемою для багатьох людей.

Висновки

Отже, в даній курсовій роботі було досліджено область соціальних взаємодій між людьми під час карантину, зокрема серед української молоді, визначено основні проблеми, що виникають в умовах домашньої ізоляції та запропоновано систему для їх вирішення. Проведені бесіди з представниками цільової аудиторії дали чітке розуміння, який функціонал необхідно реалізовувати першочергово, що власне було зроблено в межах даної курсової роботи. Дизайн інтерфейсу, створений відповідно до філософії Material-ui, вийшов лаконічний та інтуїтивно зрозумілий. Дослідження сучасного стану питання методів розробки веб застосувань дозволило створити продуктивний та швидкодіючий сайт на React (TypeScript), а також незалежний сервер з декількома рівнями захисту даних на Flast (Python), з використанням ORM. Розроблена платформа використовує ряд сторонніх сервісів, що надають незамінний функціонал: Google Mail Service, Google Map API, imgbb API; а також дозволяють значно покращити дану розробку – Elasticsearch. Система повністю протестована на предмет дотримання бізнес-вимог, а також повної працездатності окремих компонентів трирівневої архітектури, реалізованої відповідно до ідеології тонкого клієнта. Клієнтська та серверна частини, включаючи базу даних, повністю забезпечені документацією та задеплойовані за допомогою безкоштовної версії сервісу Heroku [\[15\]](#) [\[16\]](#).

Розробка прикладної програми відбувалася поетапно, відповідно до принципів методології Agile: постійне покращення мінімальної програми. Реалізований веб-застосунок має два типи користувачів: організатор та учасник, проте частина функціоналу доступна і для неавторизованих користувачів. Також реалізовано функціонал: реєстрації (з підтвердженням пошти для організаторів), авторизації, редагування профілю та зміна паролю (включно з відновленням через електронну пошту); пошуку, що відображає результати пагіновано з можливістю сортування та фільтрування, перегляду, редагування та створення подій; участі в подіях, доступу до приватних подій, а також перегляду подій учасника; сторінка

підтримки користувачів. Користувацький сайт адаптивний, доступний з будь-якого браузера чи пристрою.

Отже, в межах даної курсової роботи було розроблено незалежну платформу для організації спільного дозвілля та командних івентів. API серверу застосунку може використовуватись незалежно для інших реалізації платформи (зокрема частини адміністрування). Дана розробка є надзвичайно актуальною в сучасних умовах та має безліч перспектив для розвитку та покращення. Враховуючи вищенаведені результати роботи, сміливо можна стверджувати, що мета даної роботи досягнена, а її завдання виконані повною мірою.

Перелік використаних джерел

- [1] компанія "Реактор". (2008). *Про проект*. Отримано з TRN.ua: <https://www.trn.ua/about/>
- [2] Грегорі Ізабеллі та Еммануель Колін. (2010). *Про нас*. Отримано з Board Game Arena: <https://boardgamearena.com/team>
- [3] компанія Microsoft. (2021). *Отримайте все це безкоштовно*. Отримано з microsoft: <https://www.microsoft.com/uk-ua/microsoft-teams/free>
- [4] Stud. (2015-2021). *Потреба в спілкуванні*. Отримано з Підручники для студентів онлайн: https://stud.com.ua/49635/psihologiya/potreba_spilkuvanni_tsili_funktsiyi_spilku_vannya#:~:text=За%20М.,допомогою%20-%20до%20самопізнання%20і%20самооцінці
- [5] Wells, J. (16 August 2018 p.). *Communication Without Connection*. Отримано з Longitude: <https://longitudedesign.com/communication-without-connection>
- [6] Oloruntoba, S. (17 December 2020 p.). *The First 5 Principles of Object Oriented Design*. Отримано з digitalocean: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
- [7] IBM Cloud. (30 10 2020 p.). *Three-Tier Architecture*. Отримано з IBM: <https://www.ibm.com/cloud/learn/three-tier-architecture#:~:text=Three-tier%20architecture%20is%20a,associated%20with%20the%20application%20is>
- [8] *React Overview*. (2021). Отримано з React: <https://reactjs.org>
- [9] *Typed JavaScript at Any Scale*. (2019). Отримано з TypeScript: <https://www.typescriptlang.org>

- [10] Pallets. (без дати). *Flask 1.1*. Отримано з palletsprojects: <https://flask.palletsprojects.com/en/1.1.x/>
- [11] The PostgreSQL Global Development Group. (September 2020 p.). *About*. Отримано з postgresql: <https://www.postgresql.org/about/>
- [12] SQLAlchemy. (без дати). *The Python SQL Toolkit and Object Relational Mapper*. Отримано з SQLAlchemy: <https://www.sqlalchemy.org>
- [13] Elasticsearch B.V. (2021). *Enterprise Search*. Отримано з Elastic: <https://www.elastic.co/enterprise-search>
- [14] Nataliia Shkarovska. (November 2020 p.). *Eventizer Figma*. Отримано з Figma: <https://www.figma.com/file/AjiwUq9bigbcngfy0Ere1I/Eventizer?node-id=41%3A2>
- [15] Natalia Shkarovska. (April 2021 p.). *Eventizer Search*. Отримано з Eventizer: <http://eventizer-web.herokuapp.com>
- [16] Nataliia Shkarovska. (April 2021 p.). *Swagger*. Отримано з Eventizer API: <https://eventizer-api.herokuapp.com/swagger>
- [17] Babatunde Koiki. (14 Dec 2020 p.). *Building Restful APIs With Flask and SQLAlchemy*. Отримано з medium: <https://medium.com/better-programming/building-restful-apis-with-flask-and-sqlalchemy-part-1-b192c5846ddd>

Додатки

1 Додаток А – Опитування цільової аудиторії

2.2.3 Інтерв'ю №1 – Анастасія:

Дата проведення: 11.02.2021

Розкажіть як відбувається ваше спілкування з друзями під час карантину. Наскільки його стало більше чи менше? Який варіант тобі подобається більше? Чому?

Дуже сильно змінилося: спілкуюсь лише з сусідкою по кімнаті. Важко підтримувати контакти через месенджери. Дуже не вистачає спілкування, мені повезло, що я живу в гуртожитку, бо інакше організувати онлайн-зустріч просто неможливо.

Пригадайте останній раз, коли ви хотіли чимось зайнятися, а ваші друзі були зайняті? Що ви робили в такому випадку?

Замовила їжу, поприбирала, подивилася достатньо цікавий фільм. На жаль, я часто так проводжу свій час, адже просто не можу знайти щось цікавіше.

Розкажіть про своє захоплення та хоббі. Які з них ви поділяєте зі знайомими, а які є виключно вашою зацікавленістю? Чи намагались ви якось знайти однодумців?

Граю в шахи, пишу прозу, читаю книжки. Іноді з хлопцем граємо в онлайн-шахи. Може і було б класно знайти людей, з ким можна це все обговорювати, але на карантині це особливо важко зробити, або я просто не знаю як.

Як ви ставитесь до того, що під час карантину спілкування перейшло в онлайн-формат? Які переваги і недоліки існують для вас?

Я не вмію спілкуватися онлайн, легше коли ти можеш поговорити. Проте я не знаю аналогів, окрім як переписка, але спілкування однозначно різко зменшилось навіть з друзями.

Які складнощі у вас з'являлись під час організації спільного дозвілля або командної роботи?

Особливо великих складнощів немає, адже кожен може підключитися з будь-якої точки світу. Хоча організація кожної такої зустрічі дійсно дуже ресурснозатратна.

Коли ви востаннє збиралися із друзями онлайн? Що було спільного з зустрічами вживу, а чим вони відрізнялись?

Останній раз ми у грудні дивилися фільм, такий досвід мені не дуже сподобався.

Яка платформа для онлайн зустрічей вам подобається найбільше? Чого в ній не вистачає?

Моя улюблена – це ZOOM, він має зрозумілий інтерфейс, але його можливості достатньо обмежені. Це просто інструмент відеозв'язку.

Що вам заважає повноцінно перенести вашу діяльність та комунікацію в онлайн-формат?

Не всі активності загалом доступні в онлайн-форматі, або я просто не можу знайти тих, що підходять моїм інтересам. Дуже важко знайти релевантну компанію для події, а пояснювати кожному що планується надзвичайно довго.

2.2.4 Інтерв'ю №2 – Артем:

Дата проведення: 13.02.2021

Розкажіть як відбувається ваше спілкування з друзями під час карантину. Наскільки його стало більше чи менше? Який варіант тобі подобається більше? Чому?

Я регулярно бачусь з родичами, рідше з колегами, але зовсім не бачусь з одногрупниками, це мінус для мене. До карантину мені подобалось більше, адже я спілкувався з набагато більшою кількістю людей.

Пригадайте останній раз, коли ви хотіли чимось зайнятися, а ваші друзі були зайняті? Що ви робили в такому випадку?

Пішов конструювати робота, дивився відео на YouTube, повезло знайти компанію, що грала в онлайн-мафію на одному з сайтів для мафії.

Розкажіть про своє захоплення та хоббі. Які з них ви поділяєте зі знайомими, а які є виключно вашою зацікавленістю? Чи намагались ви якось знайти однодумців?

Мої хоббі: роботи та настільні ігри. Більшість моїх друзів поділяють мої інтереси, але якщо ніхто не може провести зі мною час доводиться заглиблюватись в мережі інтернету та вишукувати щось на свій смак серед тисячі однотипних сайтів для мафії.

Як ви ставитесь до того, що під час карантину спілкування перейшло в онлайн-формат? Які переваги і недоліки існують для вас?

Погано ставлюсь, не можу часто спілкуватися зі своїми одногрупниками. Переваг особливих не бачу, всі недоліки намагаюсь компенсувати зустрічами в онлайні, але це надзвичайно важко.

Які складнощі у вас з'являлись під час організації спільного дозвілля або командної роботи?

Організовуватись зручно всім, адже не треба кудись їхати. Проте мало хто виступає ініціатором, адже ініціатор фактично повинен задовольняти потреби інших.

Коли ви востаннє збиралися із друзями онлайн? Що було спільного з зустрічами вживу, а чим вони відрізнялись?

Вчора ми з друзями писали прогу і готувались до олімпіади. В відеочатах спілкування схоже на те, що відбувається вживу.

Яка платформа для онлайн зустрічей вам подобається найбільше? Чого в ній не вистачає?

Для мене Discord – це найзручніша, особливо для спілкування під час ігор, всього вистачає, проте щоб там зустрітися необхідно заздалегідь домовлятися і уточняти.

Що вам заважає повноцінно перенести вашу діяльність та комунікацію в онлайн-формат?

Я можу перенести все, що мене цікавить. Це виключно питання часу та бажання, що реалізувати.

2.2.5 Інтерв'ю №3 – Олександр:

Дата проведення: 28.01.2021

Розкажіть як відбувається ваше спілкування з друзями під час карантину. Наскільки його стало більше чи менше? Який варіант тобі подобається більше? Чому?

Не дуже активно, в порівнянні з докарантинним часом, зазвичай це зустрічі в діскорді, якісь онлайн-ігри. Офлайн спілкування мені набагато більше подобається, в переписці не вдається спілкуватись настільки активно та зацікавлено, наскільки це відбувається вживу.

Пригадайте останній раз, коли ви хотіли чимось зайнятися, а ваші друзі були зайняті? Що ви робили в такому випадку?

Зазвичай, коли я хочу щось робити, я хочу робити це оффлайн, адже у людей набагато менше мотивації зустрічатись онлайн. Іноді ідея щось організувати згасає, бо організація дуже важка справа.

Розкажіть про своє захоплення та хоббі. Які з них ви поділяєте зі знайомими, а які є виключно вашою зацікавленістю? Чи намагались ви якось знайти однодумців?

Складно відповісти, я багато чим цікавлюсь: програмування, що є моєю професією також є моїм хоббі, цікавлюсь електротехнікою, музикою, граю на гітарі. З друзями обожнюю грати в настільні ігри, якісь фізичні активності як ковзани. В пошуках нових однодумців не сильно зацікавлений, а от спростити організацію спільного дозвілля з друзями дуже хотів би.

Як ви ставитесь до того, що під час карантину спілкування перейшло в онлайн-формат? Які переваги і недоліки існують для вас?

Погано ставлюсь, не можу часто спілкуватися зі своїми одногрупниками. Переваг особливих не бачу, всі недоліки намагаюсь компенсувати зустрічами в онлайні, але це надзвичайно важко.

Які складнощі у вас з'являлись під час організації спільного дозвілля або командної роботи?

Працювати онлайн мені подобається, а от дружити лише в онлайні не хочеться. Серед переваг можу виділити те, що немає потреби кудись постійно їздити, що займає багато часу та ресурсів. Основний недолік – відсутність різноманіття, все спілкування достатньо одноманітне і знайти щось оригінальне нереально.

Коли ви востаннє збиралися із друзями онлайн? Що було спільного з зустрічами вживу, а чим вони відрізнялись?

Тижні 2 тому, ми збирались погуляти в Code names та мафію онлайн. Схоже з офлайн форматом, але загальна якість зв'язку не дає так сильно розговоритись, як це відбувається в живу. Взагалі ми збираємось достатньо регулярно, я маю більше 10 різних чатів з однаковими людьми для таких зустрічей, хоча це не дуже зручно.

Яка платформа для онлайн зустрічей вам подобається найбільше? Чого в ній не вистачає?

Discord - найменше лагає та у неї найприємніший інтерфейс. В цілому мені вистачає усього.

Що вам заважає повноцінно перенести вашу діяльність та комунікацію в онлайн-формат?

Все, що я хотів би перенести в онлайн вже має свої аналоги, добре, коли маєш друзів, які витрачають безліч часу, щоб їх знайти.

2.2.6 Інтерв'ю №3 – Марія:

Дата проведення: 11.02.2021

Розкажіть як відбувається ваше спілкування з друзями під час карантину. Наскільки його стало більше чи менше? Який варіант тобі подобається більше? Чому?

Намагаюсь підтримувати контакти з друзями, переписуємось, стараюсь придумувати якісь командні ігри, зідзвонююсь з друзями. Однозначно, спілкування стало менше.

Пригадайте останній раз, коли ви хотіли чимось зайнятися, а ваші друзі були зайняті? Що ви робили в такому випадку?

День тому ми з друзями хотіли зібратися у зумі для того, щоб пограти у D&D онлайн, однак усі були зайняті, тому я просто спитала кому коли зручно і ми домовились на іншу дату, коли більшості зручно.

Розкажіть про своє захоплення та хоббі. Які з них ви поділяєте зі знайомими, а які є виключно вашою зацікавленістю? Чи намагались ви якось знайти однодумців?

Люблю дивитися фільми, мене підтримує мама. Музику люблю слухати, але хіба що з братом спілкуємось про це. Настільні ігри люблю також і дуже хотіла б знайти більше однодумців, адже це класно, коли тебе хтось розуміє і підтримує..

Як ви ставитесь до того, що під час карантину спілкування перейшло в онлайн-формат? Які переваги і недоліки існують для вас?

Онлайн спілкування для мене дуже прісне, хоча є багато цікавих форматів, але на них не вистачає часу, щоб самотійно розібратися і реалізувати щось.

Які складнощі у вас з'являлись під час організації спільного дозвілля або командної роботи?

На карантині виникло дуже багато проблем з організацією роботи. Всюди. Онлайн не зрозуміло, що людина щось не зрозуміла або зрозуміла неправильно. Через це виникає багато проблем. Доводиться уточнювати і повторювати багато разів. Також люди ігнорують повідомлення або довго на них відповідають, і це спричиняє затримки по часу.

Коли ви востаннє збиралися із друзями онлайн? Що було спільного з зустрічами вживу, а чим вони відрізнялись?

Позаминулого тижня, зібралися з друзями, щоб пограти у Dnd. Через проблеми з мережею доводиться підключати одразу 2 платформи. З іншого боку завдяки онлайн ми почали збиратись частіше, адже не треба шукати місце, проте я впевнена, що наші зустрічі можна було би покращити, просто немає можливості дізнатись як.

Яка платформа для онлайн зустрічей вам подобається найбільше? Чого в ній не вистачає?

Telegram як месенджер, де ми ділимося посиланнями, а Zoom для відеозв'язку. Подобається зрозумілий інтерфейс.

Що вам заважає повноцінно перенести вашу діяльність та комунікацію в онлайн-формат?

Я обожнюю Just Dance, але ніхто мого захоплення не підтримує і не хоче приймати в цьому участь. Хоча я також розумію, що цей вид діяльності мабуть не для онлайн.

2 Додаток Б – Додаткові ілюстрації

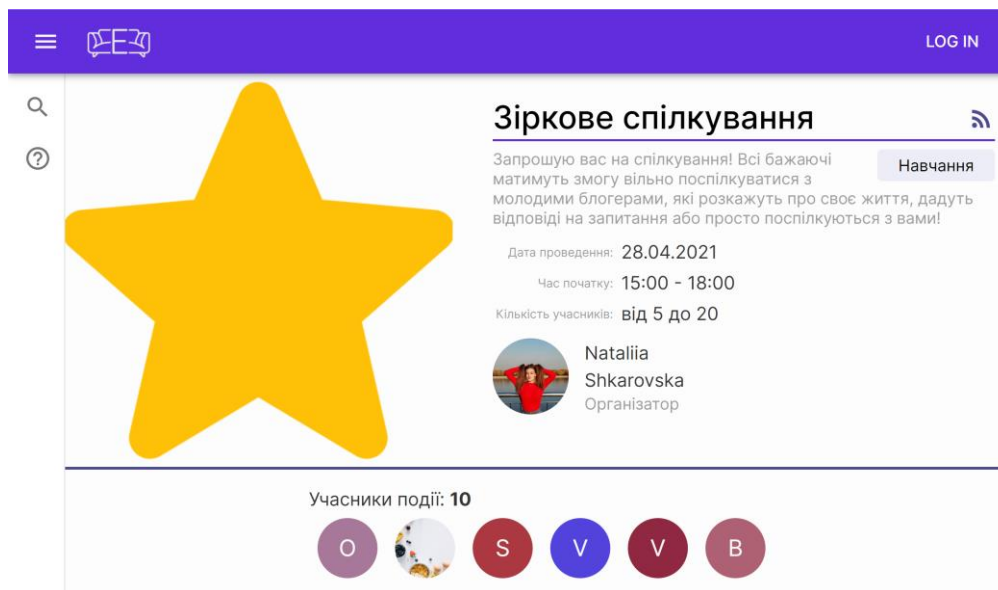


Рисунок 22 – Сторінка події для неавторизованого користувача

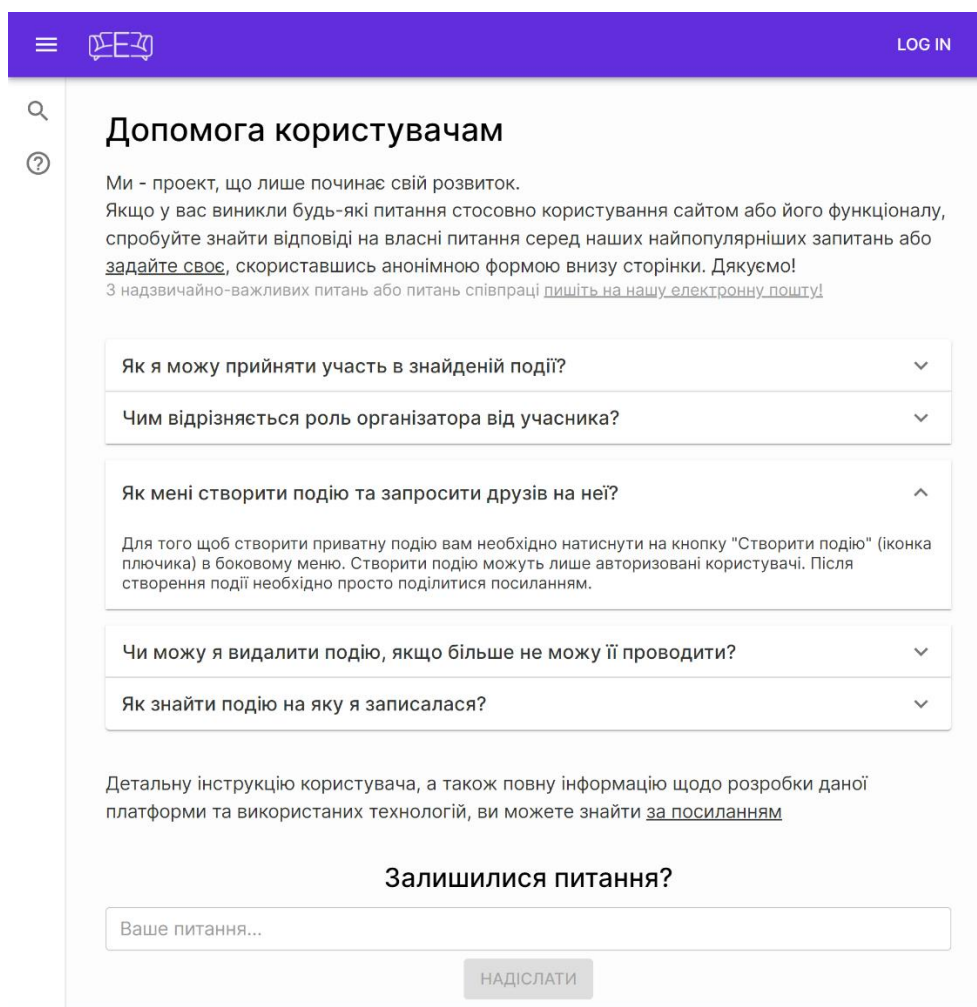



Рисунок 23 – Сторінка FAQ


LOG IN

1

2


3

Введіть пошту
Підтвердіть дію
Введіть новий пароль

Email:

НАЗАД
ДАЛІ

Рисунок 24 – Відновлення паролю (Крок 1)



LOG IN

✓

2

3


Оберіть роль
Підтвердіть email
Введіть персональні дані



Код підтвердження:
452311

НАЗАД
ДАЛІ

Рисунок 25 - Реєстрація або відновлення паролю(Крок 2)


LOG IN

✓

✓

3


Введіть пошту
Підтвердіть дію
Введіть новий пароль

Новий пароль:

Повторіть новий пароль:

НАЗАД
ЗАВЕРШИТИ

Рисунок 26 – Відновлення паролю (Крок 3)

LOG IN

1

2


3

Оберіть роль


Підтвердіть email

Введіть персональні дані

Email:



Я хочу примати участь в заходах та організовувати їх для друзів




Я хочу організовувати публічні події для аудиторії платформи

НАЗАД

ДАЛІ

Рисунок 27 – Реєстрація (Крок 1)

LOG IN

✓

✓

3

Оберіть роль

Підтвердіть email

Введіть персональні дані

Заповніть ваші дані

Ім'я *

Прізвище *

Логін *

Сфера діяльності

Пароль *

Повторіть пароль

Дата народження *

Телефон *

Завантажте ваше фото

НАЗАД

ЗАВЕРШИТИ

Рисунок 28 – Реєстрація (Крок 3 – заповнення даних)

Для того, щоб змінити пароль, уведіть стари та новий паролі.

Старий пароль:

Новий пароль:

Повторіть новий пароль:

СКАСУВАТИ ДІЮ

ЗМІНИТИ ПАРОЛЬ

Рисунок 29 – Модальне вікно зміни паролю в профілі користувача

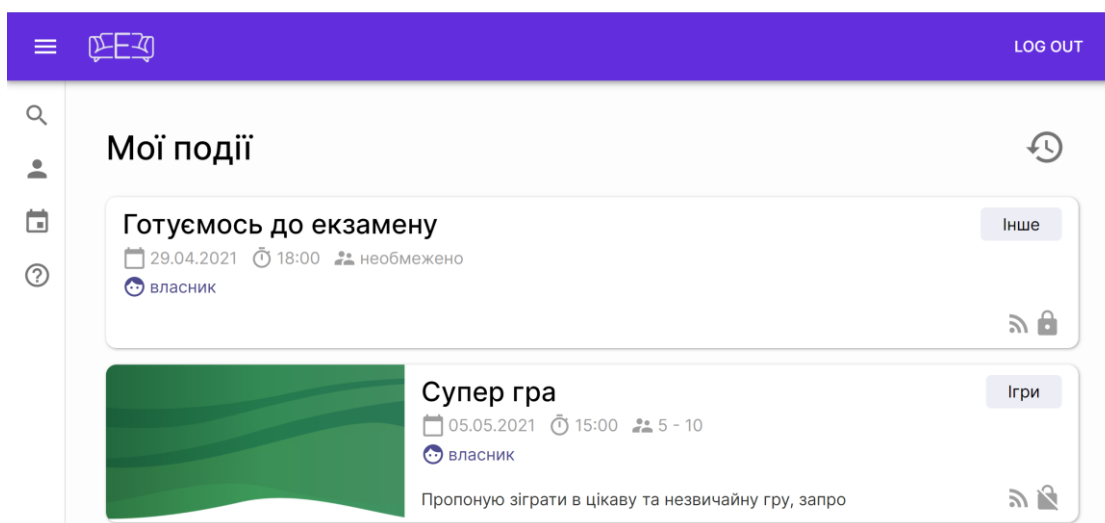


Рисунок 30 – Сторінка «Мої події»

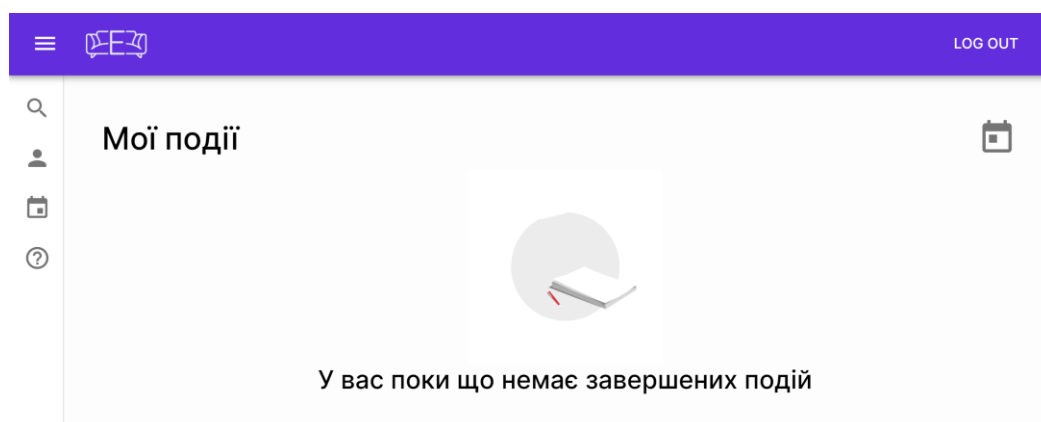


Рисунок 31 – Сторінка «Мої події», якщо список подій відсутній

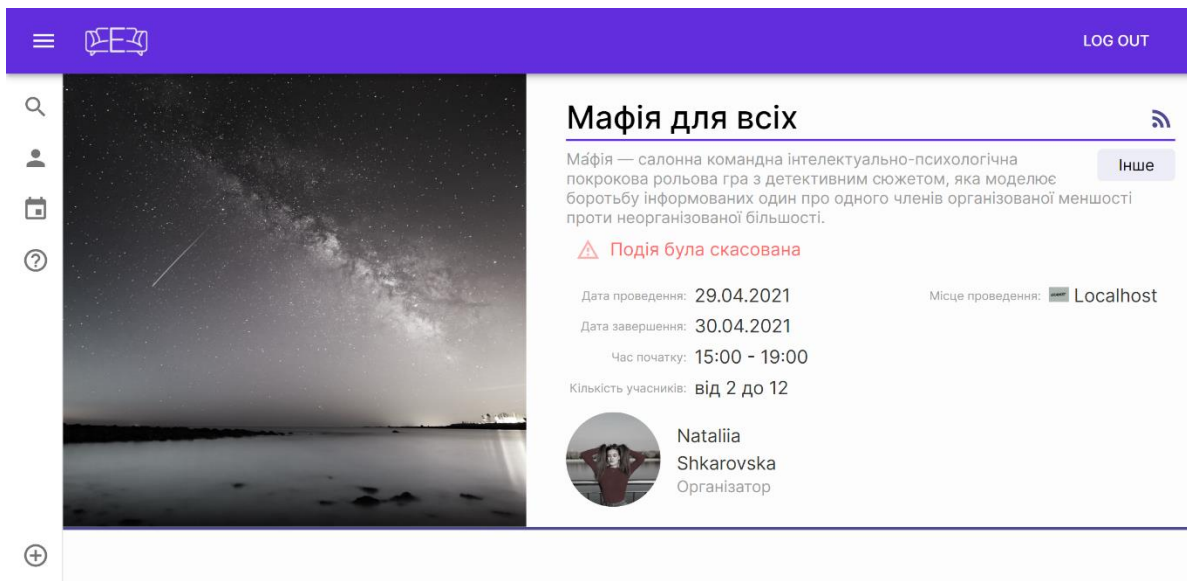


Рисунок 32 – Сторінка видаленої події



Рисунок 33 – Картка видаленої події

× Редагування події
 ЗБЕРЕГТИ

Назва

Зіркове спілкування

Опис події

Запрошую вас на спілкування! Всі бажачі матимуть змогу вільно поспілкуватися з молодими блогерами, які розкажуть про своє життя, дадуть відповіді на запитання або просто поспілкуються з вами!

Дата проведення

28/04/2021

Час проведення

15:00

Дата завершення

28/04/2021

Час завершення

18:00

Посилання на онлайн подію

http://localhost:2303

☐ Необмежена кількість учасників

Мінімум:

— 5 +

Максимум:

— 20 +

СКАСУВАТИ ПОДІЮ

Організатори:

+ ДОДАТИ ОРГАНІЗАТОРА

Рисунок 34 – Вікно редагування інформації про подію

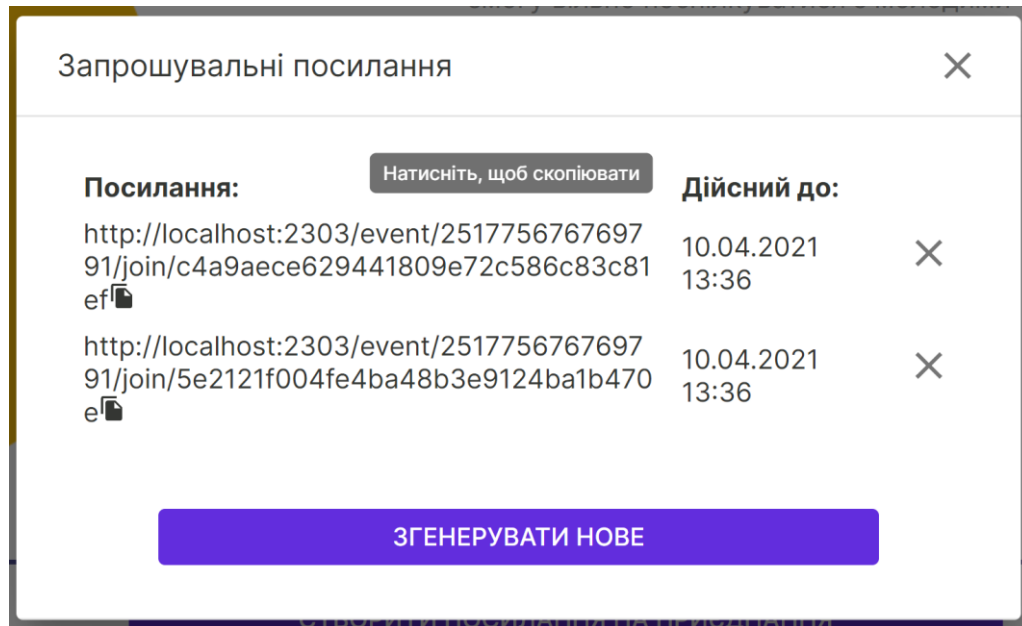


Рисунок 35 – Вікно управління кодами доступу

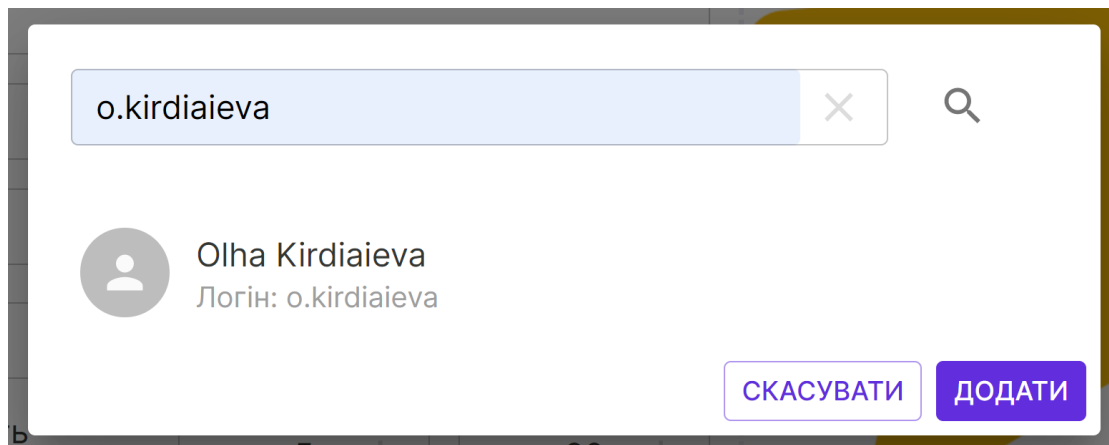


Рисунок 36 – Вікно додавання організатора до події

Організатори:

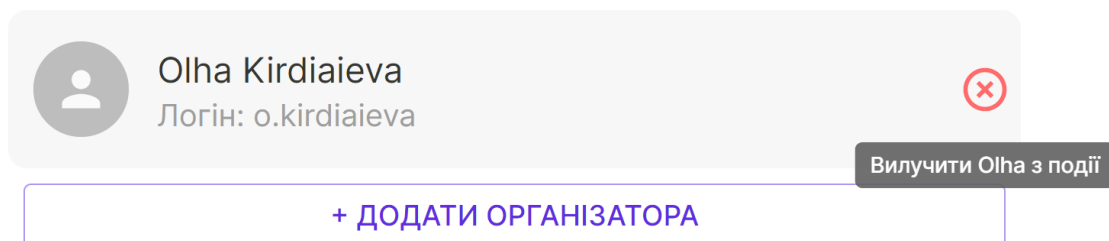


Рисунок 37 – Компонент видалення організатора з події

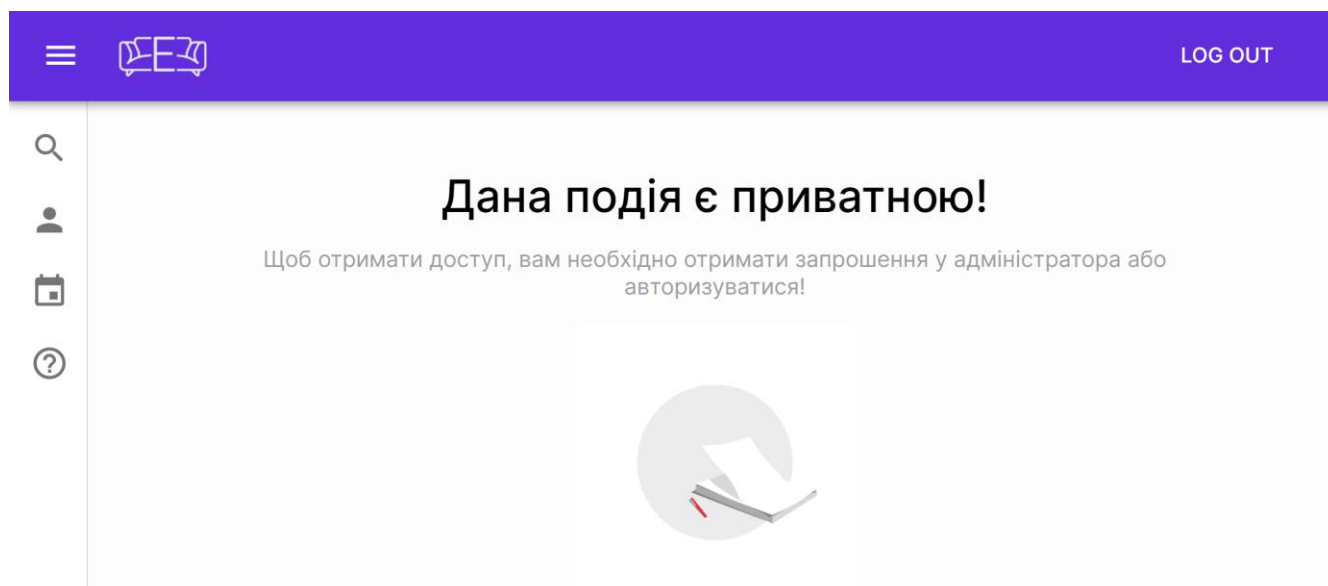


Рисунок 40 – Сторінка обмеження доступу до приватної події

3 Додаток Б – Список ілюстрацій

Рисунок 1 – Портал TRN.ua	16
Рисунок 2 – Board Game Arena logo	16
Рисунок 3 - Microsoft Teams logo.....	17
Рисунок 4 - Взаємодія між рівнями клієнт-верверної архітектури.....	22
Рисунок 5 - React logo	22
Рисунок 6 - Flask logo.....	25
Рисунок 7 – Elastic logo.....	27
Рисунок 8 – Архітектура застосунку Eventizer	33
Рисунок 9 - Use case diagram	33
Рисунок 10 – Файлова структура клієнтської частини.....	37
Рисунок 11- Файлова структура серверу	39
Рисунок 12 – Схема обробки запиту сервером	40
Рисунок 13 – Схема бази даних	41
Рисунок 14 – Схема даних Elasticsearch.....	42
Рисунок 15 – Документація інтерфейсу сервера.....	44
Рисунок 16 – Сторінка пошуку Eventizer.....	44
Рисунок 17 – Сторінка входу Eventizer.....	45
	68

Рисунок 18 – Профіль користувача	46
Рисунок 19 – Перегляд учасників події	47
Рисунок 20 – Перегляд учасників події організатором	47
Рисунок 21 – Приєднання до приватної події	48
Рисунок 22 – Сторінка події для неавторизованого користувача	61
Рисунок 23 – Сторінка FAQ	61
Рисунок 24 – Відновлення паролю (Крок 1).....	62
Рисунок 25 - Реєстрація або відновлення паролю(Крок 2).....	62
Рисунок 26 – Відновлення паролю (Крок 3).....	62
Рисунок 27 – Реєстрація (Крок 1)	63
Рисунок 28 – Реєстрація (Крок 3 – заповнення даних).....	63
Рисунок 29 – Модальне вікно зміни паролю в профілі користувача.....	64
Рисунок 30 – Сторінка «Мої події»	64
Рисунок 31 – Сторінка «Мої події», якщо список подій відсутній.....	64
Рисунок 32 – Сторінка видаленої події.....	65
Рисунок 33 – Картка видаленої події	65
Рисунок 34 – Вікно редагування інформації про подію.....	65
Рисунок 35 – Вікно управління кодами доступу	66
Рисунок 36 – Вікно додавання організатора до події.....	66
Рисунок 37 – Компонент видалення організатора з події.....	66
Рисунок 38 – Створення події (початкова сторінка)	67
Рисунок 39 – Створення події (дані заповнені).....	67
Рисунок 40 – Сторінка обмеження доступу до приватної події.....	68